Tissue thickness measurement tool for craniofacial reconstruction

Hari Kinan Gopal Boddupalli
West Virginia University

Tissue Thickness Measurement Tool for Craniofacial Reconstruction

Hari Kiran G. Boddupalli

Masters Thesis
Submitted to
The College of Engineering and Mineral Resources
Of
West Virginia University
in
partial fulfillment of the requirements for the degree of
Master of Science
in
Electrical Engineering

Computer Science & Electrical Engineering Department
West Virginia University
Morgantown, WV 26506

1999
Keywords: craniofacial reconstruction, tissue thickness measurement, forensic identification

Abstract

Tissue Thickness Measurement Tool for Craniofacial Reconstruction

Hari Kiran G. Boddupalli

Craniofacial Reconstruction is a method of recreating the appearance of the face on the skull of a deceased individual for identification purposes. Older clay methods of reconstruction are inaccurate, time consuming and inflexible. The tremendous increase in the processing power of the computers and rapid strides in visualization can be used to perform the reconstruction, saving time and providing greater accuracy and flexibility, without the necessity for a skillful modeler.

This thesis introduces our approach to computerized 3D craniofacial reconstruction. Three phases have been identified. The first phase of the project is to generate a facial tissue thickness database. In the second phase this database along with a 3D facial components database is to be used to generate a generic facial "mask" which is draped over the skull to recreate the facial appearance. This face is to be identified from a database of images in the third phase.

Tissue thickness measurements are necessary to generate the facial model over the skull. The thesis emphasis is on the first phase of the project. An automated "facial tissue thickness measurement tool" (TTMT) has been developed to populate this database.

Acknowledgements

I would like to express my sincere thanks to Dr.Bojan Cukic, my thesis advisor, for his guidance and invaluable suggestions that he has provided during the entire course of my thesis work. I am also extremely grateful to him for granting me the freedom to explore and learn.

My sincere thanks to Dr. Wils Cooley and Dr. Larry A. Hornak, my graduate committee members for their support and guidance.

I would like to thank Dr. Phillip Noble and Vobor Paravic from the Institute of Orthopedic Research and Education, Baylor College of Medicine, Houston, TX for their assistance and guidance during the initial stages of the project. I also would like to thank Dr.Frank McFadden for his suggestions during the course of my research.

Finally I would like to thank my friends and colleagues, Satish, Shekar and Goran who have been a constant source of encouragement and support.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Forensic identification is the science of determining an individual's appearance prior to his death for the purposes of identification. Usual methods include analysis of the body of the deceased victim and adding the evidence from the crime scene to identify the individual. In the absence of any external evidence for identification, analysis of the skull is the primary means of accomplishing this task. This is done through reconstruction of facial features based on their known relation to skull features.

Craniofacial Reconstruction [1] is a method of recreating the appearance of the face on the skull of a deceased individual for identification purposes. Current manual methods of estimating the appearance of the individual from his or her skull are extremely expensive and time-consuming.

Normally using manual methods, a single facial estimate is produced and publicized. As a result, the success rate for identification resulting from facial reconstruction is on the order of only 50% [2]. This single facial estimate is not necessarily the best estimate of a person's appearance due to the difficulty in determining, solely from cranial information, both body fat content prior to death and the appearance of features such as the nose, lips and eyes. Another factor that affects the success rate is the variability introduced by artistic interpretation: different artists will produce quite differing facial estimates from the same skull. Conversely, the same artist may create several similar faces from a set of completely different skulls. Low success rates are also caused by the lack of statistically significant tissue depth data, since facial reconstruction relies on knowing the tissue depths at specific landmarks on the skull. Most studies have very small population samples.

The goal of this thesis to improve the success rates for victim or suspect identification by finding a way to quickly and cost-effectively generate a number of plausible facial representations. Multiple, fast and accurate reconstructions without the necessity of a skillful modeler that will closely approximate the individual’s actual appearance, will facilitate recognition by family, friends or acquaintances and improve the identification success rates.

The tremendous increase in the processing power of the computers and rapid strides in 3D visualization can be used to perform the reconstruction, saving time and providing greater accuracy and flexibility, without the necessity for a skillful modeler. A software application that can produce a number of plausible three-dimensional facial reconstructions for a given skull would be of benefit to law enforcement agencies. The software would allow faster, easier and more efficient generation of multiple representations of an individual that would take into account differences in facial appearance not apparent from cranial analysis.

1.1 Origin of the Craniofacial Reconstruction Project

Many fields of academic studies are showing growing interest in the development of graphical tools to aid both in research and in commercial applications. The Craniofacial Reconstruction Project was started at West Virginia University in early spring of 1998 in collaboration with the Institute of Orthopedic Research and Education (IORE), Baylor College of Medicine, Houston, TX. The WVU research group developed a plan to build a tool to perform facial reconstruction based on tissue thickness measurements. The aim of the project is to build a "Craniofacial Reconstruction and Identification System" that can recreate the appearance of an individual given his/her skull and identify the person from a database of images.

1.2 Approach

The approach is to first register the 3D model of the skull and drape a "modifiable face mask" over the registered skull. 3D models of the eyes, nose, ears etc. are to be added from a database followed by additional features like hair, glasses etc. to make the person identifiable. This person would then be identified using a face matching algorithm from a database of facial images.

In order to recreate the face of the person on the skull, i.e., the "modifiable face mask", a database of facial tissue thickness measurements is needed.

1.3 Thesis Objective

This thesis introduces our approach to computerized 3D craniofacial reconstruction and describes the "Tissue Thickness Measurement Tool (TTMT)" that has been developed to automatically register facial tissue thickness.

1.4 Outline

Chapter 2 gives an overview of the existing methods of craniofacial reconstruction. A comparison of our approach and the existing methods is done highlighting the advantages and disadvantages.

Chapter 3 describes the background work and the algorithms and modeling techniques that are used in the project.

Chapter 4 contains detailed description of the 3D reconstruction from CT data and reconstruction studies at various CT spacing to determine the optimal CT spacing for tissue thickness measurement.

Chapter 5 describes the methods adopted for tissue thickness measurement. Head stack division for identifying sub stacks with similar boundary structures and extraction of outer and inner contour are discussed.

Chapter 6 is a description of the "Tissue Thickness Measurement Tool" and its user interface.

Finally, chapter 7 summarizes the work that has been done in the context of this research project, looks at some open issues involved in the use of our system, and outlines avenues for future research.

CHAPTER 2

Related Work

This chapter describes the currently existing methods of craniofacial reconstruction. This is followed by a description of our approach to the problem and the comparison of the methodologies and their relative advantages and disadvantages.

2.1 Existing Techniques for Craniofacial Reconstruction

A large number of craniofacial reconstruction techniques are available. They can be broadly classified into three categories. Sometimes a combination of two methodologies is used for better results.

1. Manual reconstruction techniques

Manual reconstruction techniques include clay or plaster based reconstructions. These are the traditional methods of reconstruction. A plaster cast of the original skull is first prepared. Plaster or plastic eyeballs are inserted into the orbits and small holes are drilled into the skull cast at 21 specific anatomical points [3]. Small wooden pegs are then cut to precise lengths and glued into the holes. The pegs project the surface of the skull to a distance that corresponds to the average thickness of the soft tissue to be found at those sites. These measurements are selected from an appropriate table. Once all the measurement pegs are in position, the basic muscles and muscle groups are built on the plaster skull using modeling clay. This is followed by gluing clay models of facial features like nose, ears etc. When the basic structure of the face has been built over the cast, further layers of clay are put in position simulating subcutaneous tissue and finally, skin.

It is clear that the manual reconstruction is time consuming and expensive. The accuracy of the reconstruction depends on the skill of the modeler and a substantial effort produces only a few models. Modification of clay models is difficult. Despite it being such a cumbersome process, surprisingly, a lot of forensic identification departments around the world still use this technique.

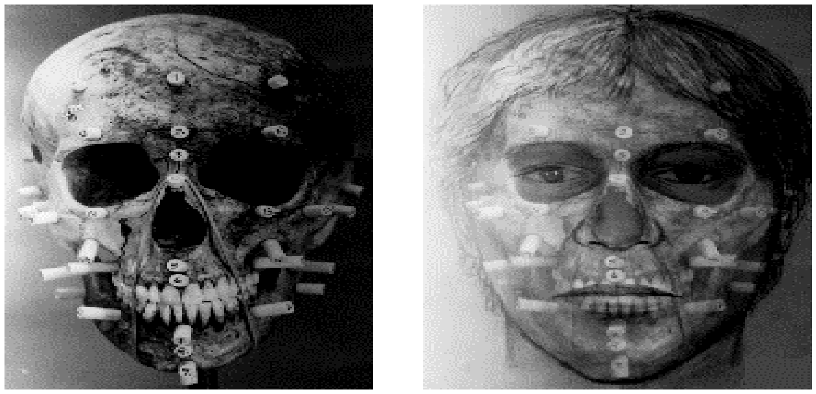


Figure 1: Traditional clay based reconstruction. Ref [2]

2. Two dimensional reconstruction techniques

The simplest and the earliest two - dimensional technique is using artists to draw the face over the photograph of a skull. Early attempts tried to match historical figures with a skull. In 1883, Welcker identified Immanuel Kant's skull [4] by matching orthogonal perspective drawings of outlines of the skull with the death mask of Kant.

Another two-dimensional method is called the *superimposition technique [4].* Superimposition consists of matching a photograph of an individual with the skull. The skulls of the victims are photographed in positions matching photographs of possible victims. The photographs of the skulls are then enlarged to life size and outline drawings of the pictures are compared.

The problems with this technique include determining the enlargement of the photographs of the individual and the skull (reference objects are needed) and positioning the skull to match the orientation of the photograph. The availability of large data banks of mugshots showing individuals in standardized positions speeds up the identification process. However, it is rare that the victim is a person with a criminal record. Regular portrait or studio photographs are often taken from the most flattering angle, which is often not the optimal angle for superimposition.

The advent of video cameras and mixing and editing apparatus also helped alleviate these problems. A video-animation compositor is used to simultaneously project images of the skull and photograph onto a screen. The compositor adjusts the intensities of both images, so that the photograph and the skull over which it is to be superimposed, are both visible. Sizing is accomplished through zooming, instead of multiple photographic enlargements or reductions.

Orientation correction is much simpler. One person repositions the skull on its mounting apparatus based on feedback received from a second person observing the combined images on the monitor. The results can be more easily recorded for replay in courtrooms. Any point on the bony surface can be projected to the soft tissue surface using a model for soft tissue thickness.

Errors may still be introduced if important morphological points are not precisely associated with the corresponding areas of the head in the photograph. Rotation and tilting of the head may not be properly accounted for.

Another notable two-dimensional technique is a computerized technique developed in Japan in 1992 [5]. The system consists of an image processing unit for skull morphometry and an image-editing unit for editing facial components on the skull images. The skull image is first registered to create a framework. To this framework suitable facial features are added from a database of facial components. After provisional reconstruction the facial image is retouched by correcting skin color and shades by an electronic painting device. This system has a limited database of facial components and the painting in the skin gave artificial look to the images generated as they were based on random choices of the facial features.

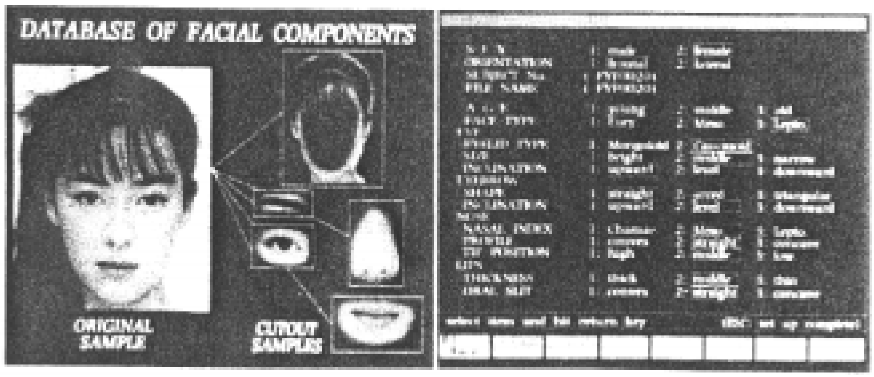


Figure 2: Two-dimensional facial reconstruction system developed in Japan. Ref [5]

3. Three dimensional reconstruction techniques

Three-dimensional techniques involve first registering the 3D model of a skull using a scanner and generating a "face mask" over it. Craniofacial Reconstruction Project at the University of British Columbia [6] first digitizes and generates a 3D model of the skull. "Dowels" that simulate tissue thickness are then interactively placed and oriented by the user [2]. Dowels are markers that simulate the pegs in normal clay reconstructions. A generic facial model, created using hierarchical B-splines, is placed over the skull model. This generic face is then fit to the dowels, resulting in the final facial estimate.

This method, however, needs a modeler who must place the dowels on the 3D model of the skull. The number of dowels that are placed is limited and the location of their placement is ambiguous. Many dowels are needed to produce a good smooth fit. Attempts are being made to add extra dowels using an automated technique, so that dowel placement on the skull model will not be more time-consuming than dowel placement on an actual physical cast of the skull.

The facial surface is editable after fitting, so the artist can modify such features as nose and lips. The lengths of the dowel will be automatically updated in the future to reflect race, gender and body fat content.

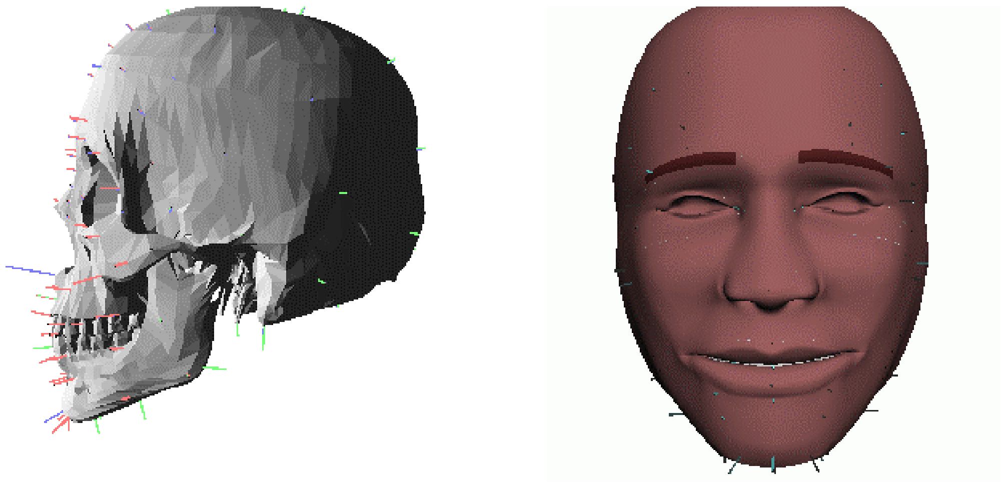


Figure 3: UBC's dowel based approach. Ref [2]

In common with many of the techniques used in facial reconstruction, the three-dimensional model can never produce a 100% accurate portrait, for there are too many variables in the details of a face. There are many obvious difficulties when reconstructing features such as mouth, nose, eyes and ears. The type and color of hair, the form of the eyebrows and the possibility of facial hair will also have profound effect on the appearance of any individual.

2.2 Our Approach

The 3D model of the skull is registered using a laser scanner. From a database of tissue thickness over the skull, a 3D "mask" is "wrapped" on the registered skull. A "mask" is a generic facial template that is generated from the facial thickness database, which is to be wrapped on the skull for generating the facial appearance. 3D models of facial features (nose, mouth, eyes etc) are then superimposed on the mask from a database of facial components. These models are modifiable and will be placed based on the bone structure of the underlying feature and its dimensions. Then the user is free to use his creativity to modify the image based on other parameters.

Our approach is somewhat similar to the method followed by the University of British Columbia Craniofacial Reconstruction Project described above. The following are the differences:

- The reconstruction will be based on a database of tissue depth measurements instead of markers on the skull. Even though a reasonable reconstruction can be obtained from few points on the skull, as demonstrated in the UBC project [6], we adopted the approach with numerous locations because we aim to apply this research to other fields where the precision is necessary. One application area is in plastic surgery, where a bone- tissue relationship based reconstruction can be very helpful in predicting how a patient may look after surgery. Other areas include identification of missing children, age based facial reconstruction, defense (war victim identification) and archeology. Also, since the reconstruction is based on tissue depth measurements the accuracy of reconstruction is expected to improve, thus improving the chances of identification.

- In the UBC project, the "face mask" is modified to generate the facial features like the nose, eyes, ears etc. This process is very time consuming and tedious and needs an expert artist to perform this task. In our approach, modifiable, generic 3D templates of facial features are to be developed. They are to be placed on the generated face -mask. This will speed up the process of reconstruction and avoid the need of a modeler.

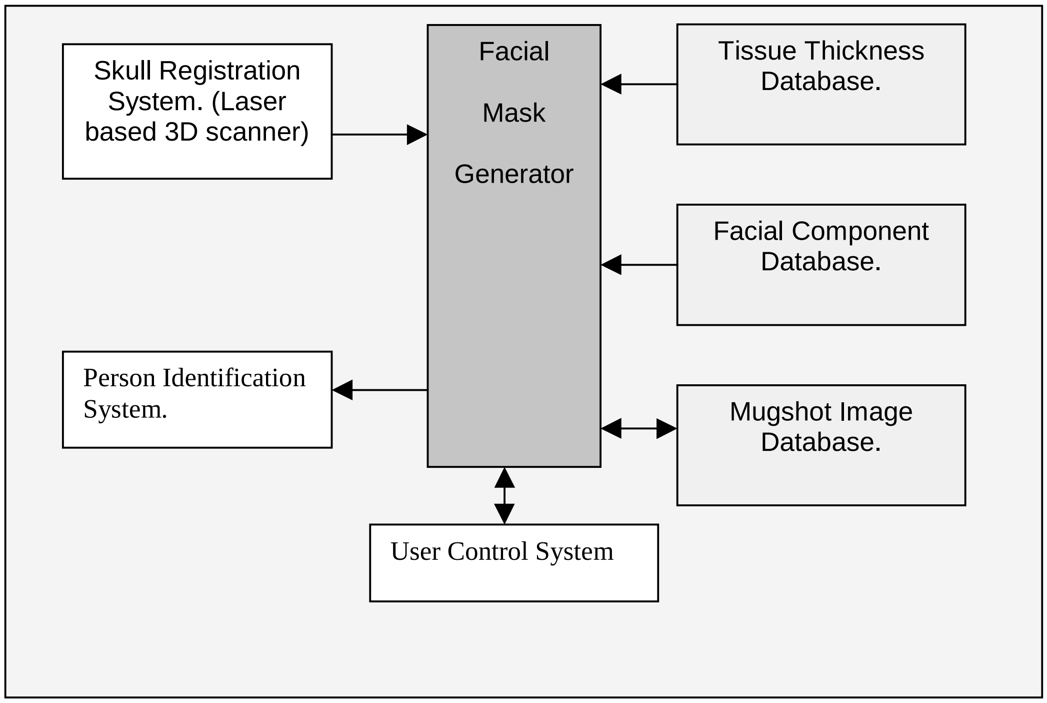

Skull Registration System. (Laser based 3D scanner)
Facial Mask Generator
Tissue Thickness Database.
Facial Component Database.
Person Identification System.
Mugshot Image Database.
User Control System

Figure 4: WVU Craniofacial Reconstruction System model.

2.3 Project Plan and Goals

A fully automated system for Craniofacial Reconstruction and Identification, currently under development at West Virginia University, is shown in Fig 4. It will consists of the following components:

- Skull Registration System: This component consists of a laser-based scanner for 3D-skull registration.

- Databases: The reconstruction system uses three databases to generate the facial tissue mask. The three databases are the Tissue Thickness Database, Facial Component Database and the Mugshot Image Database. The tissue thickness database will consist of average tissue thickness over the skull stratified over age, sex and race. The facial component database will consist of generic 3D models of facial components, which include ears, nose, eyes and mouth. The Mugshot Image database is to be used to identify the individual.
- Facial Mask Generator: An automated system is to generate the modifiable facial "mask". This module takes inputs from the other components, which include the Skull Registration System, Tissue Thickness Database, Facial Component Database and the Mugshot Image Database. It , then, based on the inputs, generate facial features and tissue on the registered skull in 3 dimensions. The user will have the flexibility to interactively modify the proposed facial appearance, if needed.
- User Interaction System: This is the user interface for the system. This will enable a user to modify the template generated by the Face Mask Generator and add additional features like facial hair, skin color adjustments, scars etc.
- Person Identification System: This system is to be used to identify the person from the Mugshot Image database.

The Skull Registration System will be used to register the 3D model of the skull by digitizing it with the laser scanner. Laser scanners [7] are already commercially available.

To realize the remaining components of the system, the project is organized into three phases.

Phase 1:

The first phase of the project is to generate a database of tissue thickness. An automated tool that measures the tissue thickness over the face from CT (Computed Tomography) data has been developed for this purpose which is the main focus of this thesis. This tool will be used to populate the Tissue Thickness Database.

Phase 2:

The second phase of the project is to generate a 3D model of the face over the skull based on the tissue thickness data and database of facial features. The face is created using a "face mask template" which is generated from the Tissue Thickness Database from phase 1. 3D models of nose, eyes and mouth will be superimposed on to the mask from the Facial Component Database (this database needs to be populated with 3D facial components before it can be used). Additional features such as facial hair, color etc. can be added to complete the reconstruction.

The modifiable, interactive, 3D " face mask" will be generated by fitting in Bezier or B-Spline [8] surfaces through the tissue thickness data set instead of using regular polygon fitting algorithms. These surfaces have the advantage they can be easily modified based on a set of control points. Details about the method of generation of these surfaces are given in Appendix A.

Phase 3:

In this phase the generated face is identified from the Mugshot Image Database. Face identification techniques like eigenface [9] are to be used during this phase. View based eigenface [10] technique will be used to increase identification success rates, as this technique is tolerant to rotation in the images.

2.4 Phase 1: Automated Tissue Thickness Measurement Tool

The goal of this thesis is to develop a tool to realize phase 1 of the project. An Automated Tissue Thickness Measurement Tool is developed which will be used to generate a database of tissue thickness measurements. Tissue thickness is defined as the distance between the outer layer of the bone and the outer layer of the skin.

Before tissue thickness can be measured, 3D reconstruction of the head and skull are performed. This is done to study the methods that are to be used for mask generation and to determine the optimal CT spacing to be used for tissue thickness measurement and facial reconstruction.

CHAPTER 3

Overview of Algorithms

This chapter describes principles behind the visualization processes, image processing operators and algorithms that are used in the later sections of the thesis.

3.1 *Visualization*

Visualization is the process that transforms data into simple graphics primitives. The methods of computer graphics are then used to convert these primitives into pictures.

3.1.1 Components of a 3D visualization model

A 3D model consists of the following components.

Scene:

A complete representation of the components required to generate an image or animation including lights, camera, actors, properties, transformations, geometry, texture and other pertinent information.

Lights:

One of the major factors controlling the rendering process is the intersection of light with the actors in a scene. It is the interaction of the emitted light and the surface of the actors in the scene that define what we see. Different types of lights are used in computer graphics. Simplest is an infinitely distant point light source that is used in the reconstruction.

Actors or Objects:

These are the components that are to be viewed or rendered.

Camera:

Camera is the viewer of the light reflected from the objects. It captures the light from the 3D scene and projects it onto a 2D plane. A number of factors determine how a 3D scene gets projected onto a plane to form a 2D image. These include position, orientation and focal point of the camera, the method of camera projection and the location of the camera clipping planes.

3.1.2 Rendering and image formation

Computer graphics is the process of generating images using computers. This is done through a process called *rendering*. Rendering can be viewed as the process of converting graphical data into an image. There are many types of rendering processes ranging from 2D paint programs to 3D techniques. In data visualization the main goal is to transform data into graphical primitives that are then rendered.

Physical description of rendering:

Consider the figure given below.

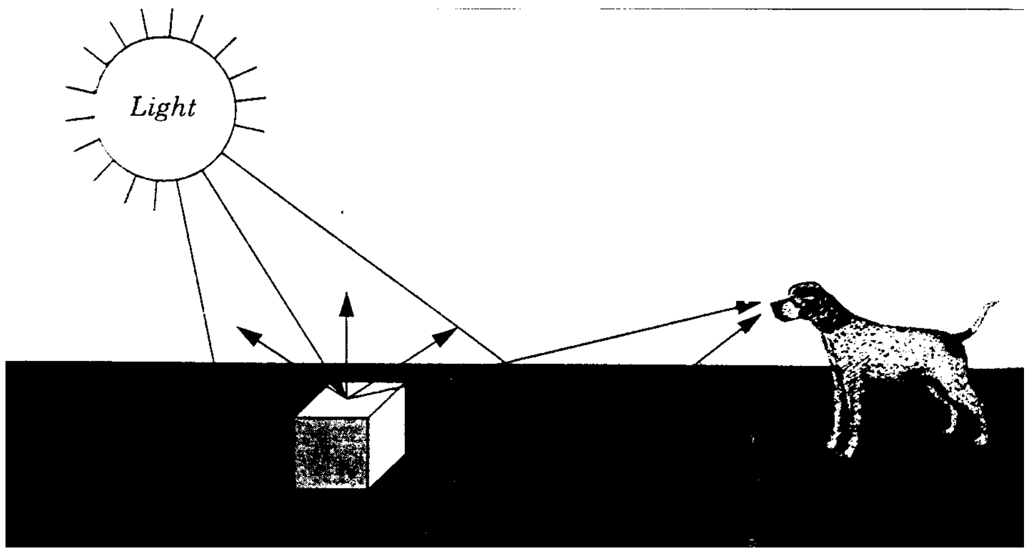


Figure 5: Physical generation of an image.

Rays of light are emitted from a light source in all directions. Some of these rays strike the cube whose surface absorbs some of the incident light and reflects the rest. A part of this reflected light may head towards us and enter our eyes. If this happens we “see” the object. This physical model is used for rendering in computer graphics too.

A common and effective technique for 3D computer graphics is called ray tracing or ray casting. Ray tracing simulates the interaction of light with objects by following the path of each light ray. Typically, the ray is followed backwards from the viewer’s eye and into the world to determine what the ray strikes. The direction of the ray is the direction we are looking at. When a ray intersects an object we can determine if that point is lit by our light source. This is done by tracing a ray from the point of intersection towards the light. If the ray intersects the light, the point is being lit. If the ray intersects something else before it get to the light then that light will not contribute to illuminating the point. For multiple light sources the total contribution from all the light sources plus any ambient scattered light, will determine the total lightning or shadow for that point.

Rendering categories:

Rendering processes can be broken into two categories: image order and object order. Ray tracing is an image order process. It works by determining what happens to each ray of light, one at a time. An object order process works by rendering each object, one at a time.

Surface and volume rendering:

It is generally assumed while rendering an object we are viewing the surfaces of objects and their interactions with light. However, common objects such as clouds, water, fog etc are translucent or scatter light that passes through them. Such objects cannot be rendered using a model based exclusively on surface interactions. Instead we need to consider the changing properties inside the object to properly render them. These two rendering models are referred to as surface rendering (i.e., render the surface of the object) and volume rendering (i.e., render the surface and the interior of an object).

When we render an object using surface rendering techniques, we mathematically model the object with a surface description such as points, lines, triangles, polygons etc. The interior of the object is not described. Volume rendering technique allows us to see the inhomogeneity inside objects.

3.1.3 Data set representation and interpolation

As described earlier visualization is mapping information into graphics primitives. This involves transformation of data from one form into another. Visualization data is discrete. Because of the discrete character of the data we do not know anything about regions in between the data values. This poses a serious problem. The obvious solution to this problem is “interpolation”. A relationship between neighboring data values is assumed, usually a linear function (quadratic, cubic, spline etc., can also be used) to generate data values between known points.

Volumetric visualization uses voxels as the building blocks of data set. A voxel is a primary three-dimensional cell. The voxel is topologically equivalent to the hexahedron with additional geometric constraints. Each face of the voxel is perpendicular to one of the coordinate x-y-z axes.

3.1.4 Contouring

When a surface colored with data values is viewed, the eye separates similarly colored areas into distinct regions. Contouring is effectively constructing the boundary between these regions. These boundaries correspond to contour lines (2D) or surfaces (3D) of constant scalar value. Examples of 2D contouring include weather maps, annotated with lines of constant temperature or topological maps drawn with lines of constant elevation. Three-dimensional contours are called *“isosurfaces”* and are approximated by many polygonal primitives. Examples of isosurfaces include constant medical image intensity corresponding to body tissue such as skin, bone etc. (These isosurfaces are used to extract the bone and tissue features while reconstruction).

Contour generation techniques:

a) Boundary tracking method

Consider the 2D structured grid sown in the figure.

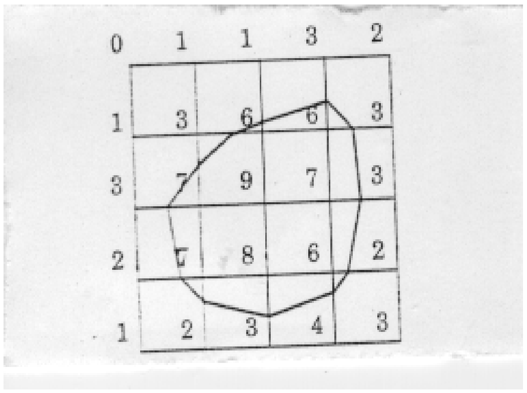


Figure 6: Contouring a 2D- structured grid with contour line value 5. Ref [11]

Scalar values are shown next to the points that define the grid. Contouring begins by selecting a scalar value, or a contour value that corresponds to the contour line or surface to be generated. To generate a contour, some form of interpolation should be used. This is because we have scalar values at finite set of points in the data set and the contour value can lie between the point values. Since the most common interpolation technique is linear, points on the contour surface are usually generated by linear interpolation along the edges. If an edge has scalar values 10 and 0 at its two end points and if we are trying to generate a contour line of value 5, then edge interpolation computes that the contour passes through the mid point of the edge.

Once the points on cell edges are generated, all these points are connected into contours using different approaches. One approach detects an edge intersection (i.e., the contour passes through an edge) and then “tracks” this contour as it moves across cell boundaries. It is obvious that if a contour edge enters a cell, it must exit a cell as well. The contour is tracked until it closes back on itself or exits a data set boundary. If it is known that only a single contour exists, then the process stops. Otherwise, every edge in the data set is checked to see whether the contour lines exist.

b) Marching squares (2D) and cubes (3D) algorithm

Another approaches uses a divide and conquer technique, treating cells independently. The marching squares algorithm is in 2D and marching cubes in 3D. The basic assumption of these techniques is that a contour can only pass through a cell in a finite number of ways. A case table is constructed that enumerates all possible topological states of a cell, given combinations of scalar values at the cell points. The number of topological states depends on the number of cell vertices and the number of inside /outside relationships a vertex can have with respect to a contour value. A vertex is considered inside a contour if its scalar value is larger than the scalar value of the contour line. Vertices with scalar values than the contour value are said to be outside the contour. For example, if a cell has four vertices and each vertex can either be inside or outside there are 2^4 possible ways that the contour can pass through the cell. In the case table we are not interested in where the contour passes trough the cell (geometric intersection), just how it passes through the cell (i.e. topology of the contour in the cell).

The figure below shows the sixteen combinations for a square cell.

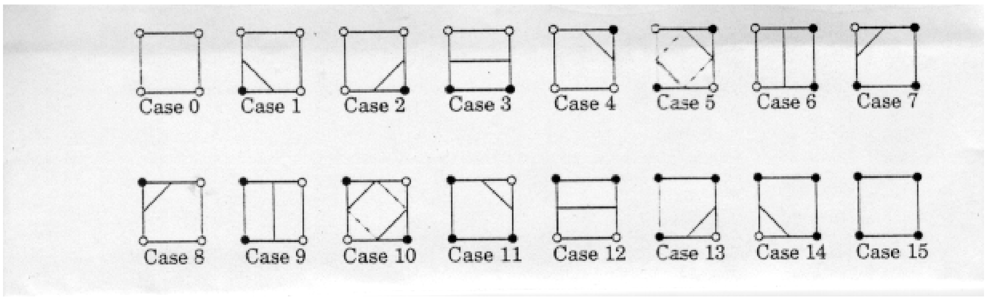


Figure 7: Sixteen different marching square cases. Ref [11]

An index into the case table can be computed by encoding the state of each vertex as a binary digit. For a 2D data represented on a rectangular grid we can represent the 16 cases with a 4 – bit index. Once the proper case is selected, the location of the contour line/cell edge intersection can be calculated using interpolation. The algorithm processes a cell as it moves or marches to the next cell. After all the cells are visited, the contour will be completed. In summary the marching algorithm proceeds as follows:

1. Select a cell
2. Calculate the inside/outside state of each vertex of the cell.
3. Create an index by storing the binary state of each vertex in a separate bit.

4. Use the index to look up the topological state of the cell in a case table.
5. Calculate the contour location (via interpolation) for each edge in the case table.

This procedure will construct independent geometric primitives in each cell. At the cell boundaries duplicate vertices and edges may be created. Using a special coincident point merging operation may eliminate these duplicates. Another important aspect is interpolation along each edge should be done in the same direction.

The 3D analogy of the marching square's algorithm is the marching cube's algorithm, which is used here for CT reconstruction. Here, there are 256 combinations of scalar value, given that there are eight points in a cubical cell. The figure below shows these combinations reduced to 15 cases by using

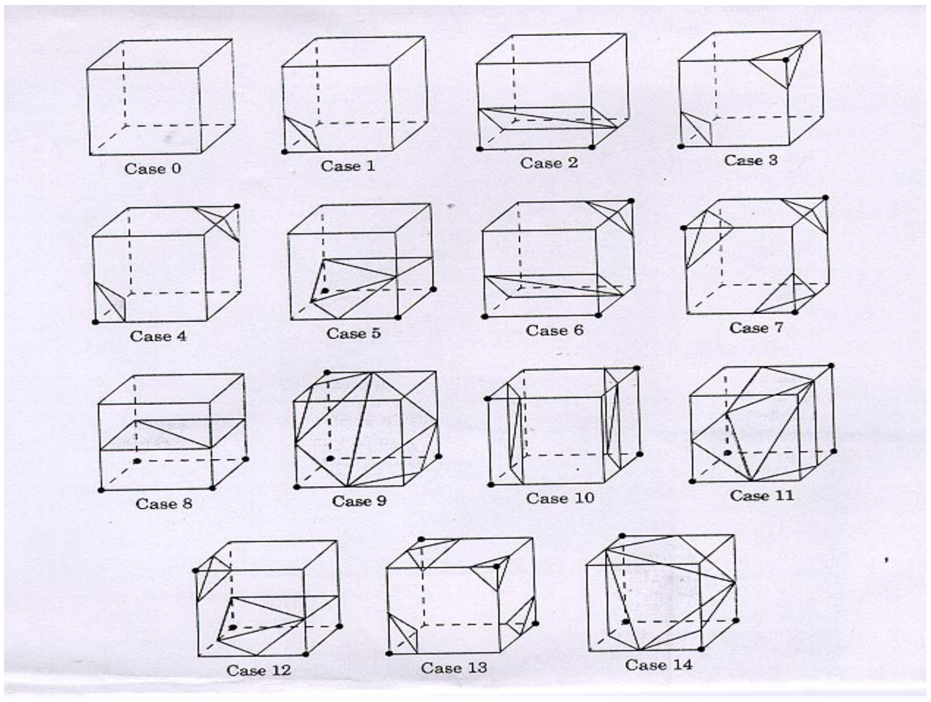


Figure 8: Marching cube cases for 3D isosurface generation. Ref [11]

the arguments of symmetry.

More information and mathematical details about the marching squares and the marching cubes algorithm can be found in [12].

3.2 Digital Image Processing

Digital Image Processing is concerned with the manipulation and analysis of images discretized from continuous signals. Algorithms to manipulate these images mathematically involve integrals, however, the necessity of sampling these images for computer use simplifies these formulae in terms of replacing integration operations by summation operations.

3.2.1 Edge detection

Edge detection is the process of determining the edges in an image. “Edges” are points or regions in an image where there is a sharp change in the intensity function [13]. In other words edges are pixels where this function changes abruptly. This usually occurs at the boundaries of images across which there is a sharp image contrast. Hence edge detection is used to detect boundaries in images.

An image function depends on two variable-coordinates (x,y) in the image plane. A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

An edge is a property attached to an individual pixel and is calculated from the image function behavior in the neighborhood of a pixel. It is a vector variable with a magnitude and direction. The edge magnitude is the magnitude of the gradient and the direction is rotated with respect to the gradient direction by -90^o.

Edge detection operators can be divided into three categories [14].

Category 1 operators:

Operators approximating derivatives of the image function using differences. Some of them are rotationally invariant (e.g. Laplacian) and thus are computed from one convolution mask only. Others, that approximate first derivatives, use several masks. The orientation is estimated on the basis of the best matching of several simple patterns. Operators approximating the first derivative of an image function are sometimes called compass operators because of the availability to determine the gradient direction.

Roberts operator:

The Roberts operator is one of the oldest operators. It is very easy to compute as it uses only a 2X2 neighborhood of the current pixel. Its convolution masks are

$$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix} \qquad \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$$

so the magnitude of the edge is computed as

$$|g(i,j) - g(i+1, j+1)| + |g(i,j+1) - g(i+1, j)|$$

The primary advantage of Roberts's operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Laplacian operator:

The Laplacian operator ∇^2 is a very popular operator approximating the second derivative, which gives the gradient magnitude only. The Laplacian equation is approximated in digital images by a convolution sum. A 3 X 3 mask h is often used; for a 4-neighborhood and a 8-neighborhood it is defined as

$$h = \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix} \qquad h = \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$$

The Laplacian operator has the disadvantage that it responds doubly to some edges in the image.

Prewitt operator:

The Prewitt operator approximates the first derivative. The gradient is estimated in eight (for a 3X3-convolution mask) possible directions and the convolution result of greatest magnitude indicates the gradient direction. Larger masks are possible.

Following are 3 X 3 masks for each operator.

$$h_1 = \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix} \qquad h_2 = \begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix} \qquad h_3 = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$$

Sobel operator:

$$h_1 = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \qquad h_2 = \begin{matrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{matrix} \qquad h_3 = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

The Sobel operator is often used as a simple edge detector of horizontality and verticality of edges in which case only mask h_1 and h_3 are used. If the h_1 response is y and the h_3 response x, we can then derive the edge strength magnitude as

$$(x^2+y^2)^{1/2} \text{ or } |x| + |y|$$

and the direction as $\tan^{-1}(y/x)$.

Other operators in this category include Robinson operator and Kirsch operator.

Category 2 operators:

These operators based on the zero crossings of the image function second derivative (e.g. Marr- Hilderth or Canny edge detector). These are based on the fact that the first derivative of the image function should have an extremum at the position corresponding to the edge in the image, and so the second derivative should be zero at the same position. It is much easier and more precise to find zero crossing positions than an extremum.

Marr-Hilderth edge detector:

Proposed by David Marr and Hilderth in their paper in 1991, this is also called the Laplacian of the Gaussian detector. A 2D Gaussian smoothing operator G(x,y) is first applied on the image to remove the noise. The Gaussian is given by

$$G(x,y) = e^{-x2+y2/2\sigma 2}$$

Where x and y are the image coordinates and σ is the standard deviation of the associated probability distribution. The standard deviation is the only parameter of the Gaussian filter – it is proportional to the size of the neighborhood on which the filter operates. Pixels more distant from the center of the operator will have smaller influence and pixels further away than 3σ have negligible significance. After the application of the 2D filter, Laplacian operator is used to obtain the second derivative. Finally, the convolution mask of a zero crossing detector is given by

$$h(x,y) = c(x^2+y^2/\sigma^2 - 1)\, e^{-x2+y2/2\sigma 2}$$

Canny Edge Detector [15]:

Canny - introduces 3 criteria for edge detection

- good detection
- good localization
- only one response to a single edge

He derives edge detector optimizing the above mentioned criteria.

The Canny operator works in a multi-stage process. First of all the image is smoothed by Gaussian convolution. Then a simple 2-D first derivative operator (somewhat like the Roberts Cross) is applied to the smoothed image to highlight regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as "non-maximal suppression". The tracking process exhibits "hysteresis" controlled by two thresholds: T1 and T2, with T1 > T2. Tracking can only begin at a point on a ridge higher than T1. Tracking then continues in both directions out from that point until the height of the ridge falls below T2. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments. More details about the Canny edge detector can be found form Canny's paper in [15].

Category 3 operators:

Operators which attempt to match an image function to a parametric model of edges (e.g. snakes, parametrically deformable models etc). These categories of edge detectors use the higher-level image knowledge to identify regions based on a parametric template, which is deformed according to the image parameters.

One example is that of a converging contour (snakes) which drive the contour to its minimum energy. These methods can be efficient where there is a lot of higher-level image knowledge and pattern matches can be used. Usually these are faster than the low-level techniques. A description and how they can been applied to detect multiple boundaries for tissue thickness measurements is discussed in Appendix B.

The next chapter describes the 3D reconstruction of soft tissue and facial features from CT data.

CHAPTER 4

3D Reconstruction from CT

This chapter of the thesis describes the reconstruction methods adopted to generate 3D models of soft tissue and bone features from CT (Computed Tomography). This is done to study the methods that are to be used for mask generation and to determine the optimal CT spacing to be used for tissue thickness measurement. Reconstruction techniques and algorithms described in Chapter 3 are used to perform the reconstruction.

4.1 Test Data: Extraction and Formatting

CT data of the head is used to perform the reconstruction. CT or Computed Tomography measures the attenuation of the x-rays as they pass through the body. CT's counterpart MRI, or the Magnetic Resonance Imaging modality comprises of the actual images (unlike images generated by attenuation of x-rays) of the cut cross-sections of the body.

4.1.1 Data sets

Four sets of data were used in our study. The first two sets were from the NIH's Visible Human Project and the other two were CT's of sinus patient's from the Methodist Hospital's Radiology Department, Houston, TX.

Visible Human Data Set:

In 1989, the National Library of Medicine (NLM) began an ambitious project to create a digital atlas of the human anatomy. The NLM Planning Panel on Electronic Image Libraries [16] recommended a project to create XRAY Computed Tomography (XRAY-CT), Magnetic Resonance Imaging (MRI) and physical sections of a human cadaver. The project is called "The Visible Man." Another cadaver, that of a 59 year-old woman, "The Visible Woman", was released in the fall of 1995.

Visible Male Data Set:

MRI - The male data set consists of axial MRI images of the head and neck taken at 4-mm intervals and longitudinal sections of the rest of the body also at 4-mm intervals. The MRI images are 256 pixel by 256-pixel resolution. Each pixel has 12 bits of gray tone resolution.

CT - The CT data consists of axial CT scans of the entire body taken at 1 mm intervals at a resolution of 512 pixels by 512 pixels where each pixel is made up of 12 bits of Grey tone. The axial anatomical images are 2048 pixels by 1216 pixels where each pixel is defined by 24 bits of color, about 7.5 megabytes. The anatomical cross-sections are also at 1-mm intervals and coincide with the CT axial images. There are 1871 cross-sections for each mode, CT and anatomy, obtained from the male cadaver.

Visible Female Data Set:

MRI – The female data set consists of axial MRI images of the head and neck taken at 4-mm intervals and longitudinal sections of the rest of the body also at 4-mm intervals. The MRI images are 256 pixel by 256-pixel resolution. Each pixel has 12 bits of gray tone resolution.

CT – The data set from the female cadaver has the same characteristics as the male cadaver with one exception. The axial anatomical images were obtained at 0.33-mm intervals instead of 1.0-mm intervals. This results in over 5,000 anatomical images. The data set is about 40 gigabytes in size. Spacing along the "head to toe" (Z axis) direction was reduced to 0.33 mm in order to match the pixel spacing in the "XY" plane, which is 0.33 mm.

More details about the data set and the visible human project and the data set can be found at [16].

Sinus Patient Data Set:

This data set was obtained from the radiology lab at Methodist Hospital, Houston TX. The CT data of the sinus patient’s consisted of 45 slices, each slice being 16 bit unsigned 512 X 512 pixels. These are axial cross- sections of the head.

The CT of the sinus patient was available on an optical disk and was extracted on a Macintosh Power PC using the CT Extractor software. This gives individual slices in the .tif format.

4.1.2 Head stack generation

Individual slices extracted have to be stacked to generate a head stack in order to generate volumetric data for reconstruction. "Head Stack" is a stack of individual slices placed one on top of other in order. The head stack is also stored in a tiff format that can be easily read by any visualization tool.

Scion Image [17] is freely downloadable software from NIH which, can be used to perform this task. Scion Image is used to capture, display, analyze, enhance, measure, annotate, and output images. Scion Image extensively supports all Scion frame grabber boards, and provides a powerful and complete image acquisition environment.

NIH ‘s Scion Image was used to generate a "head stack" by first removing the header from each slice and overlaying consecutive slices. The resulting head stack is stored in tiff format, which is later used for reconstruction.

The following are the details of the head stacks obtained after formatting the data sets using Scion Image.

Visible Male CT Head Stack:	
No of slices	*: 164*
Slice names	*: c_vm****.fre*
Header size	*: 3416 bytes*
Byte order	*: BigEndian*
Dim of the slice	*: 512 X 512*
Type	*: 16 bit unsigned*

Table 1: Visible Male Head Stack specifications.

Visible female CT Head Stack	
No of slices	*: 208*
Slice names	*: c_vf****.fre (1 –208)*
Header size	*: 3416 bytes*
Byte order	*: BigEndian*
Dim of the slice	*: 512 X 512*
Type	*: 16 bit unsigned*

Table 2: Visible Female Head Stack specifications.

Sinus patient-1 CT Head Stack	
No of slices	*: 45*
Slice names	*: RobertLee.**
Header size	*: 748 bytes*
Byte order	*: BigEndian*
Dim of the slice	*: 512 X 512*
Type	*: 16 bit unsigned*

Table 3: Sinus patient-1 Head Stack specifications.

Sinus patient-2 CT Head Stack (2 sets)	
No of slices used	*: 18*
Slice names	*: Wright Karla.**
Header size	*: 748 bytes*
Byte order	*: BigEndian*
Dim of the slice	*: 512 X 512*
Type	*: 16 bit unsigned*

Table 4: Sinus patient-2 Head Stack specifications.

4.2 3D Reconstruction from CT Data

4.2.1 Software tools

- *Visual Numerics PV Wave 6.2*

Visual Numerics PV Wave is a mathematical modeling and data visualization tool. It can transform raw data into high quality, full color, graphics and maps. PV-WAVE utilizes a powerful array-oriented fourth-generation language (4GL) specifically designed for visual data analysis. Large data sets are imported and manipulated easily. Its efficient and compact 4GL helps you develop and debug applications quickly, and can reduce software coding efforts by up to 80 percent compared to traditional languages. Scripts are written as .pro files which are executed by the PV Wave Interpreter. PV wave comes with three main libraries: Mathematics and Statistics, Image Processing and Signal Processing. The main advantage of PV Wave is it not only allows visualization of data but also can be used to perform numerical analysis and mathematical modeling on it. X-Motif Widgets in PV Wave support creation of advanced user interfaces can be built for applications.

Further details about PV Wave can be obtained at [19].

Visual Numerics PV Wave 6.2 provides the visualization libraries and also can perform mathematical operations on data sets. Hence this tool was used later to perform the reconstruction.

4.2.2 3D Reconstruction using PV-Wave 6.2

PV-Wave uses the ray tracing technique (described in chapter 3) to perform advanced rendering. Ray tracing is the process of following the path of light rays from a light source into a scene.

There are six basic steps to rendering using PV-Wave.

Step 1: Import or generate data to be rendered.

In order to use the rendering functions the data needs to be set up or generated so that the rendering functions can be called on it. The rendering functions of PV Wave accept data in the form of *vertex arrays and polygon lists*. The vertex arrays and polygon lists are PV Wave's formats for representing polygons. It consists of array of vertices and a flat one-dimensional array of polygons.

vertex_arrays: A (3, n) array containing the three-dimensional coordinates of each vertex.

polygon_list: A array containing the number of sides for each polygon and the subscripts into the vertex_array.

Many different functions in PV-Wave can be used to generate the vertex_array and polygon_list based on the type of data we are using (2D, volumetric etc.). While reading data, like in our case from the head stack, the data is 16-bit volumetric and read into an array in the form of bytes. A vertex_array and poly_list is generated using the SHADE_VOLUME function for volumetric data.

SHADE_VOLUME function:

SHADE_VOLUME computes the polygons that describe a three-dimensional contour surface. Given a 3D volume and a "*contour value (isosurface value)*" the function generates a list of vertices and polygons describing the contour surface for the particular contour value. Each voxel (Chapter 3) is visited to find the polygons formed by the intersections of the contour surface and the voxel edges. This function is the "Marching Cubes Algorithm" described in chapter 3. Isosurface is a region with a constant density in a volumetric data. It is also called the *threshold value*. The bone features in the volumetric data (skull) have an isosurface or threshold value of *140*. The skin or the soft tissue features a isosurface value of 220.

Syntax:

SHADE_VOLUME, volume_data, isosurface_value, vertex_array, poly_list

Input Parameters:

volume_data: An array with three dimensions containing the dataset to be contoured. If the volume array is dimensioned (D0, D1, D2), the resulting vertices range as follows:

- In X, they range between 0 and D0 - 1.
- In Y, they range between 0 and D1 - 1.
- In Z, they range between 0 and D2 - 1.

isosurface_value: A scalar containing the contour or the isosurface value.

Output Parameters:

vertex_array: The name of a variable to receive the vertex array. This variable will be set to a (3, n) floating-point array, suitable for input to the MESH or POLYSHADE procedure to generate the 3D model.

poly_list: The name of a variable to receive the polygon list, an *m-element longword array*. This list describes the vertices of each polygon and is suitable for input to MESH or POLYSHADE.

The surface produced by SHADE_VOLUME may then be displayed as a shaded surface with the POLYSHADE procedure.

Step 2: Manipulate and convert the data.

This step is optional depending on the type of data being used. In our case this is not necessary. This step is essential in generating grids. Gridding is a method that generates a uniform grid from an irregularly spaced data. The method interpolates or extrapolates new data from a given set of data and then creates a uniform grid that maps this data. Other operations in this stage include coordinate conversion, volume manipulation and polygon manipulation.

Step 3: Set up the data for viewing.

This step is used to set up the data for viewing before rendering. There are three parameters in which data can be set for viewing.

CENTER_VIEW: This procedure sets system viewing parameters to display data in the center of the window.

SET_VIEW3D: This generates a 3 dimensional view given a view position and a view direction.

T3D: This is a procedure that accumulates one or more sequences of translation, scaling, rotation, perspective, or oblique transformations and stores the result. It is used with two keywords which specify the oblique and the perspective projections.

Oblique: A two-element vector containing the oblique projection parameters. Points are projected onto the XY plane at Z=0 as follows:

X’ = X + Z(d * cos(a))
Y’ = Y + Z(d * sin(a))
where Oblique(0) = d and Oblique(1) = a

Perspective: A scalar (p) indicating the z distance to the center of the projection. Objects are projected onto the XY plane at Z=0, and the "eye" is located at point (0, 0, p).

The default is an identity matrix.

In our case the T3D is used to set up the data viewing as it fits the generated model to the window. The default identity matrix is used as the keyword. Other keywords with T3D include, ROTATE, SCALE and TRANSLATE.

Step 4: Setting up the coordinate system for rendering

Before the data can be rendered a coordinate system for rendering its dimensions within which the data is to be rendered has to be specified. The light source is assumed to be at the origin. Since the model is rendered from the vertex_array, the dimensions of the required region for rendering is determining the maximum and the minimum vertex lengths along the x, y, and z directions in the *vertex_array.* Hence the maximum and minimum values of the vertex array along the x, y and z axis give the enclosing region. Once this is determined a coordinate system is defined with the given dimensions using the SHADE_SURF function. The SHADE_SURF function is actually used to create a shaded surface representation of a regular or nearly regular gridded surface, with shading from a light source model. This function when used with the *No_Data* and *Save* keywords generates the coordinate system of a specified size and initializes the light source at the origin.

Syntax:

SHADE_SURF, INDGEN(2,2), XRange= [min_x, max_x], YRange= [min_y, max_y], ZRange = [min_z, max_z], /NoData, /Save, Az=0, Ax=0

Where max_x and min_x are the maximum and minimum bounds of the coordinate system along the x axis. These are the maximum and the minimum x values in the vertex_array.

Step 5: Use the rendering routines to render the image

Once the data is generated, the data is setup for viewing and the coordinate system defined the next step is rendering the data. POLYSHADE constructs the shaded surface using the scan line algorithm. The shading model is a combination of diffuse reflection and depth cueing. Polygons are shaded in one of two ways:

a) With constant shading, where each polygon is given a constant intensity.

b)With Gouraud shading, where the intensity is computed at each vertex and then interpolated over the polygon.

POLY_SHADE function

Constructs a shaded surface representation of one or more solids described by a set of polygons.

Syntax:

result = POLYSHADE(vertex_array, poly_list)

Input Parameters:

vertex_array: A (3, n) array containing the x-, y-, and z-coordinates of each vertex.

poly_list: An integer or longword array containing the indices of the vertices of each polygon. The vertices of each polygon should be listed in counterclockwise order when observed from outside the surface.

result: A 2D byte array containing the shaded image.

The SET_SHADING keyword is used to set the direction of the light source. In our case it is set to its default value 1.

Step 6: Display the data.

Once an image is generated, the image is displayed using the TVSCL function, which scales the intensity values of an input image into the range of the image display, usually from 0 to 235 on Windows systems, and outputs the data to the image display at the specified location.

Three different methodologies were tried on PV-Wave to reconstruct the face and the skull features from the CT data. These are described below.

Method 1:

The CT data formatted using Scion Image is read into a 3- Dim byte array. All the data manipulations for visualization are done on this data set. The head stack was read in as bytes. A vertex and a polygon list is generated using the *SHADE_VOLUME* function which uses a modified Marching Cubes Algorithm and a surface is fit through this point data using the ray based surface fitting technique using the *POLY_SHADE* function of the PV wave. Threshold values of 220 for the skull and 140 for the skin are used to obtain the soft tissue and bone features of the face. The resulting images are displayed.

Results: The results are shown at the end of the section.

Method 2:

In the second approach each slice of the data set was individually digitized to obtain spatial (x,y,z) data instead of using the volumetric voxel data. NIH Scion

Image was used to digitize the slices by marking points on the edge of the slice. The edge of each slice was read in as a data file containing the X and Y coordinates of the points on the digitized slice. The data in the file is organized as column data. This number of points in this data set was increased to 360 points per slice using linear interpolation.

After the data set is obtained the read (x, y) data is aligned so that there is no "twist error" (point 1 on slice 1 is right above point 1 of slice 2) between consecutive slices. This is done in the following way. The X and Y data column data from the file was read in into two arrays for two successive slices. A reference point is chosen on the first slice is chosen and its distance from the all the points on the next slice is calculated. The point with the minimum distance is the corresponding point on the next slice. After the matching point on the next slices is calculated the array of the next slice is checked for clockwise or anti-clockwise rotation and the two arrays are aligned based on whether the rotation is clockwise or anti- clockwise. Then the next two slices are taken. After all the slices are aligned interpolation is done in the Z direction to generate the final X, Y Z volumetric data. A surface is fit through this data using the ShadeSurfIrr function of PV-Wave.

Data Used: Ten slices of the Male CT data around the nose region were used to test the reconstruction. 360 points on each slice chosen and aligned using the technique above.

Results: Average (not very good) results are produced. This method is very cumbersome and manual digitization of slices is required. The surface of the reconstruction was not smooth.

Method 3:

This approach is similar to the method adapted earlier. Here instead of directly using the generated (x,y,z) data for surface generation, "slice images" consisting of just the edge are generated which are stacked and read in as a stack. This is done to utilize the smoothing algorithms from PV Wave. This is explained in more detail below.

Scion Image is used to digitize a slice as described earlier. The (x,y) co-ordinates of corresponding slice was matched to minimize "twist error" by calculating the minimum X-Y distances as described earlier. After all the slices are interpolated and matched each slice points are plotted on an X-Y co-ordinate system. This plot is read in as a slice image and all such images are stacked. 3D model is generated from this stack as in method 1.

Data Used: Ten slices of the Male CT data around the nose region were used to test the reconstruction. 360 point's on each slice were chosen and aligned using the technique above. These slice points were plotted to obtain a stack of 10 slices which was reconstructed using method 1.

Results: Here again not very satisfactory reconstruction was obtained. This method is also very cumbersome and the problem of smoothing still prevailed.

Of all the above methods, method 1 is the simplest and works best with the data. In the other two methods (though it was thought that these methods would give better results than compared to the earlier methods as the noise features were removed by extracting the edge) the edge points have to be individually extracted and produced sharp edges in the reconstructed image. The problem of smoothening sharp edges and islands on the surface is automatically taken care of in the first method as the POLY_SHADE function of PV wave automatically calls the SMOOTH function with appropriate width for the smoothing window to give the final smoothed image.

SMOOTH function:

Smooths an array with a boxcar average of a specified width.

Syntax:

result = SMOOTH(array, width)

array: The array to be smoothed. If the array has more than one dimension, the algorithm is applied over each dimension.

width: The width of the smoothing window (should be an odd number).

result: A copy of array smoothed with a boxcar average of the specified width. It has the same type and dimensions as array.

The SMOOTH function was applied on method 2 and 3 but the width of the smoothing window had to be manually adjusted until the desired smoothness in the image is obtained. Hence the last two methods were abandoned. The best reconstruction is obtained form the first method.

Results:

The following are the reconstruction obtained using PV Wave following the first method described above.

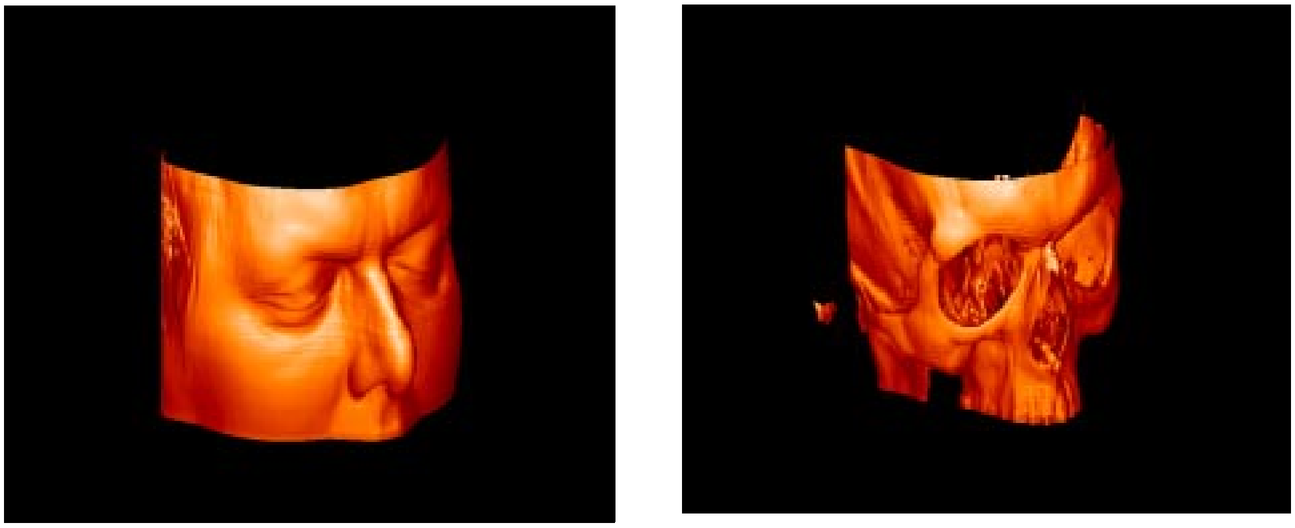


Figure 9: Reconstruction of head stack Lee (Ref. Table 3). Isosurface values skull: 140, skin: 220.

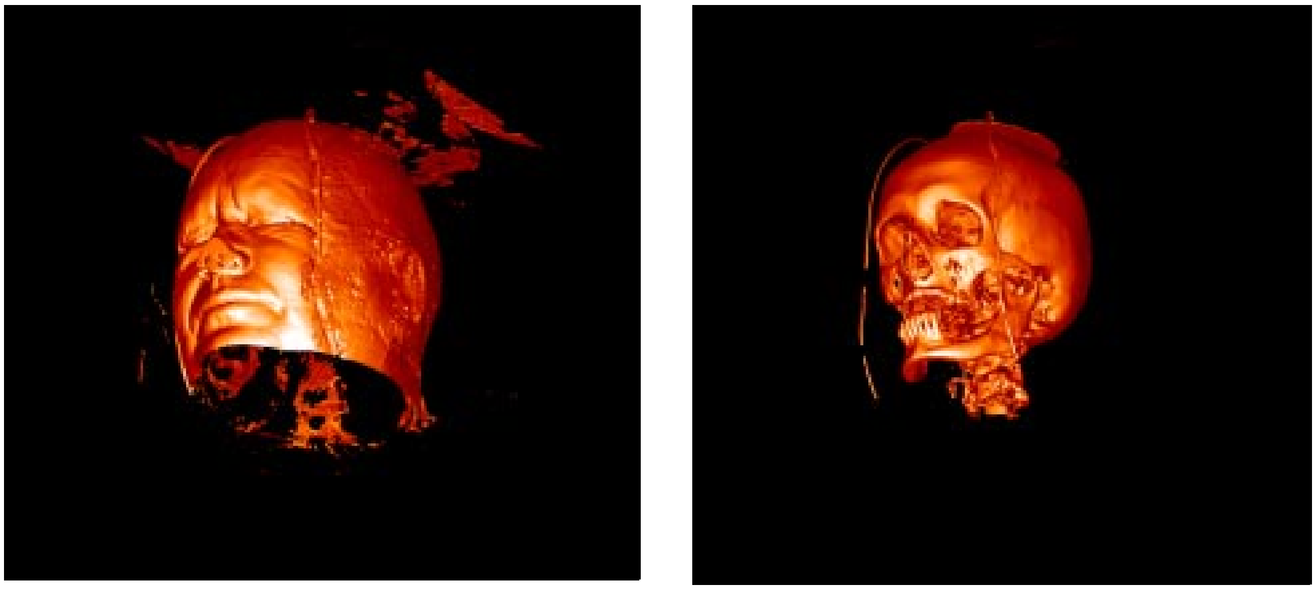


Figure 10: Reconstruction of head stack Visible Female (Ref. Table 2). Isosurface values skull: 140, skin: 220.

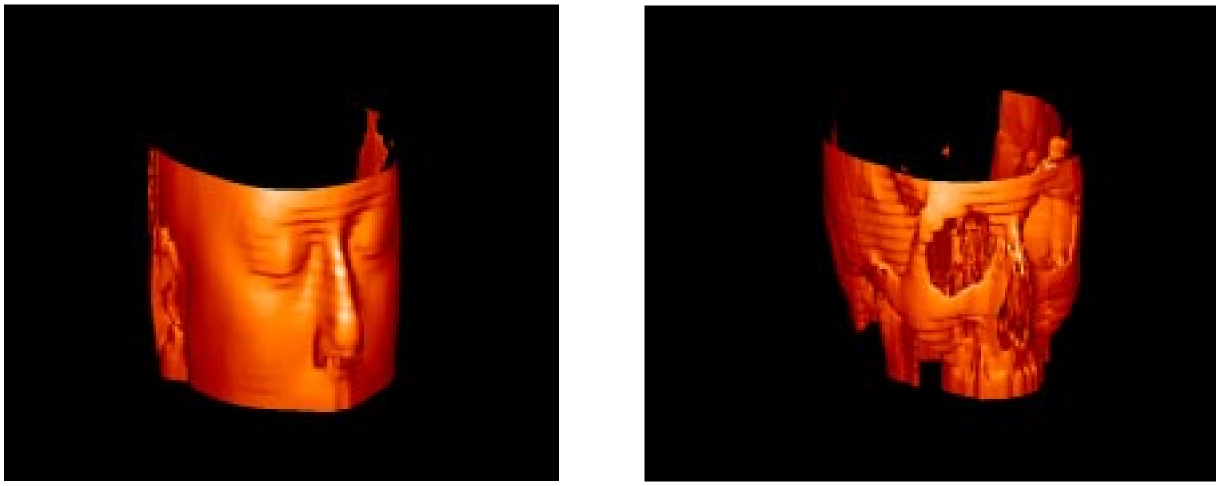


Figure 11: Reconstruction of head stack Karla1 (Ref. Table 4). Isosurface values skull: 140, skin: 220.

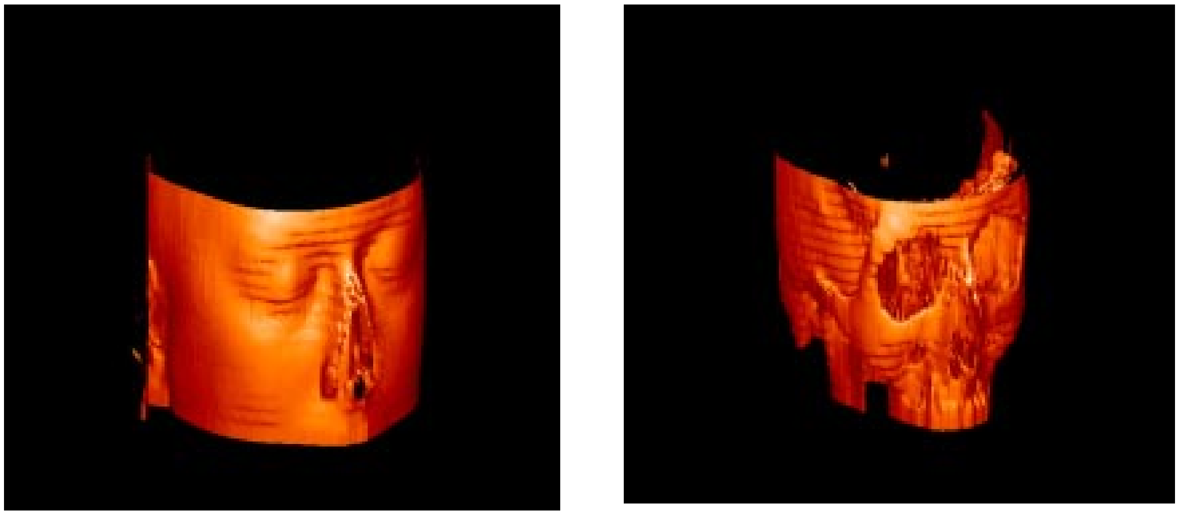


Figure 12: Reconstruction of head stack Karla2 (Ref. Table 4). Isosurface values skull: 140, skin: 220.

4.3 Reconstruction Study for Optimal CT Spacing Determination.

Facial reconstruction is to be done by draping a 3D mask over a given skull. This mask is to be generated from a database of tissue thickness. The thickness data is placed over the skull as markers and surfaces are fit through this data set to generate the facial appearance.

Before a tissue thickness database is generated, it is necessary to know the approximate number of thickness measurements that are required so that a proper identifiable mask is generated from the data set (i.e. density of points in space required for proper reconstruction). If the thickness measurements are too far apart the reconstruction will not be smooth.

CT can be obtained at various spacing. Each spacing corresponds to a particular density of points in space. The purpose of this study was to determine the CT spacing that is optimal for the reconstruction of the facial tissue and bone features from it. This optimal spacing CT will be used to obtain tissue thickness.

4.3.1 Approach

In order to obtain the optimal CT spacing, reconstruction is performed from CT at various spacing and their relative accuracy and identifiability is compared. The Visible Female data set was used to carry out this study. The Visible female head CT data set details are given below.

No of slices	*: 209*
Dim of slice	*: 512 X 512*
No of bits/pixel	*: 16*
Byte Order	*: BigEndian*
Spacing	*:1mm*
Header Size	*: 3416bytes*

Seven separate stacks were generated from the main head stack. Each of these stacks contains slices at a different spacing. This is done by skipping intermediate slices in the stack. These stacks are reconstructed using method 1 described earlier. The resulting images are shown below.

Case 1:

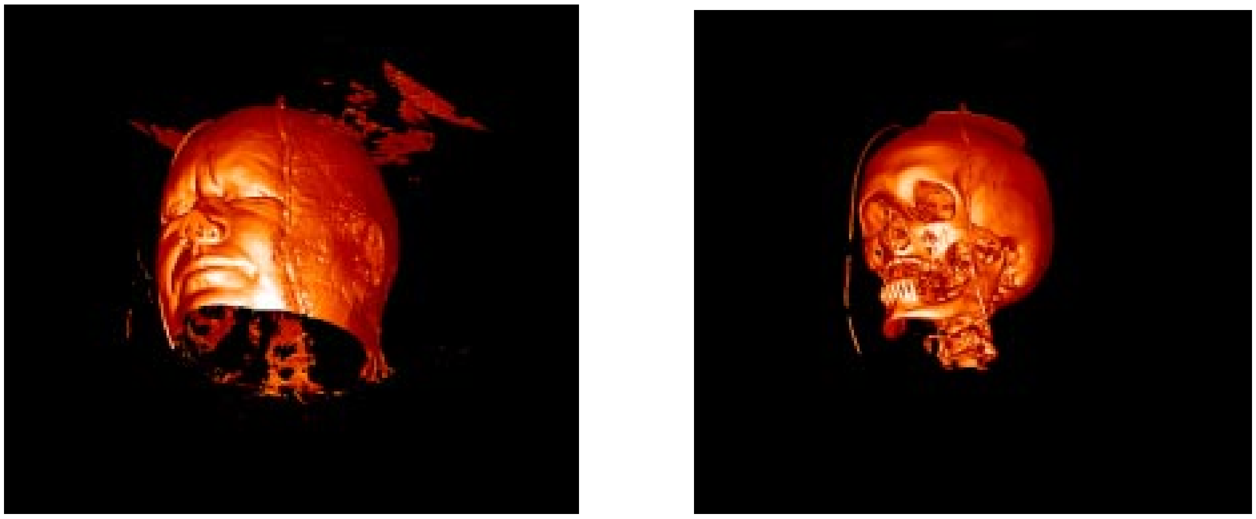


Figure 13: CT reconstruction at 1-mm spacing. No of slices in the stack: 209

As shown in Fig. 13, the density of slices is large and the spacing close hence the reconstruction looks good. All the facial features are clearly visible.

Case 2:

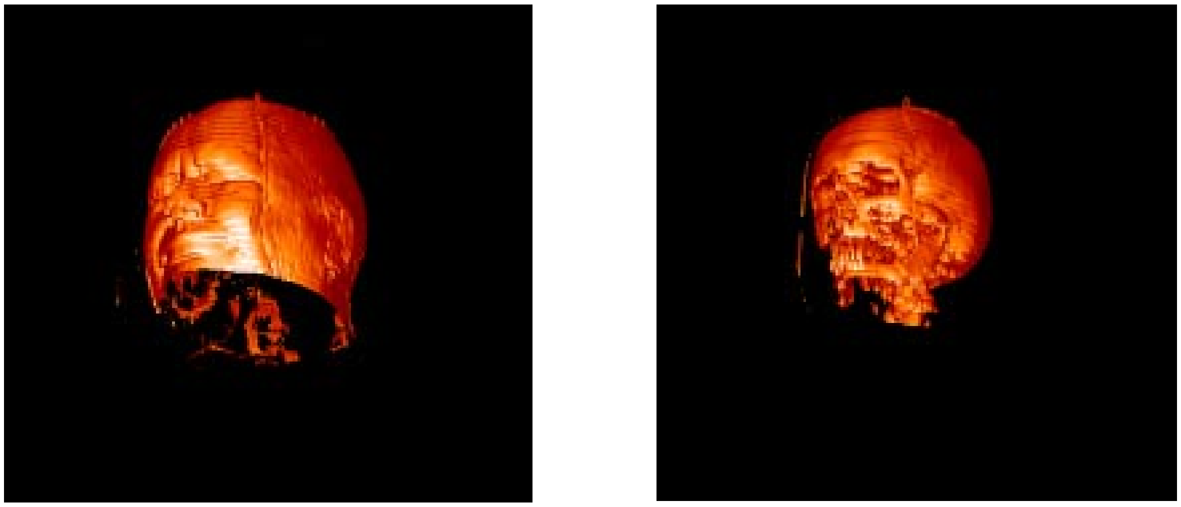


Figure 14: CT reconstruction at 7-mm spacing. No of slices in the stack: 30

As shown in Fig. 14, the density of slices is less and the spacing wide hence the reconstruction is not smooth. Step like features can be seen on the face.

Case 3:

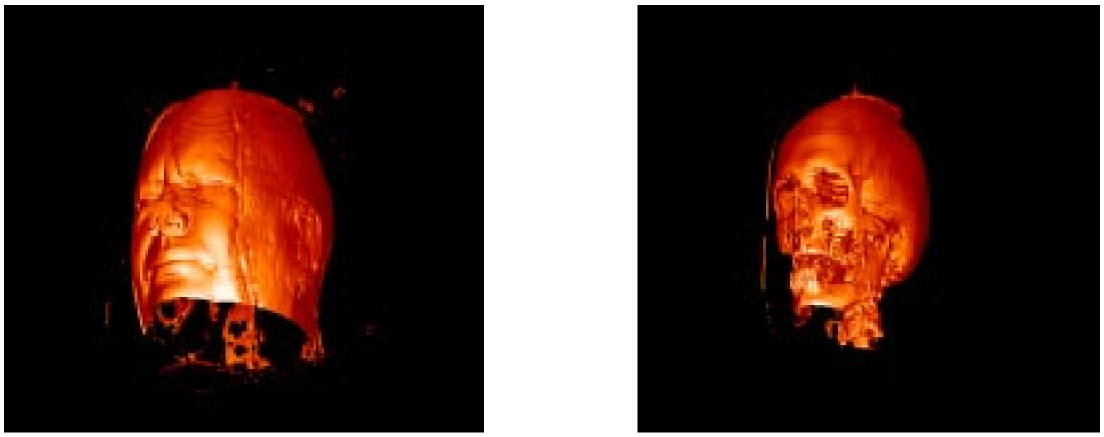


Figure 15: CT reconstruction at 4-mm spacing. No of slices in the stack: 50

As shown in Fig. 15, since the density of slices is large and the spacing close hence the reconstruction looks good. All the facial features are visible but there is still some degree of step formation.

Case 4:

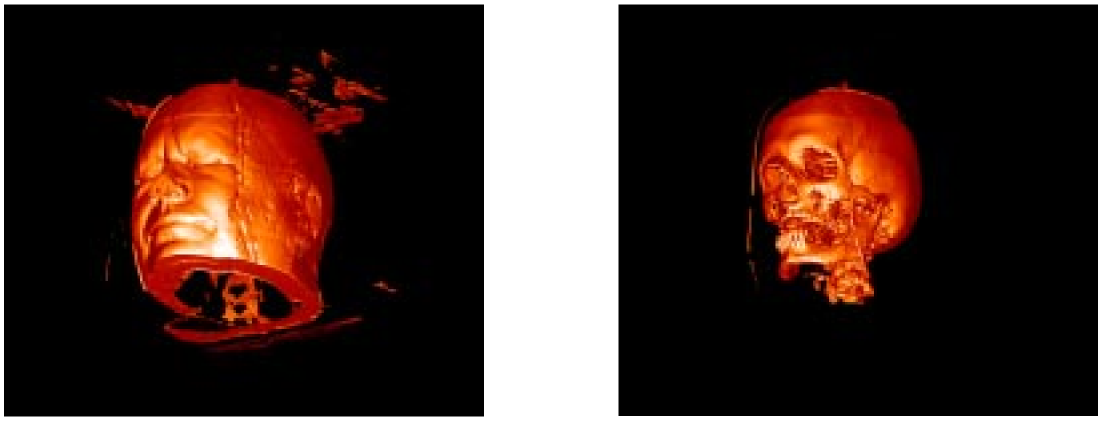
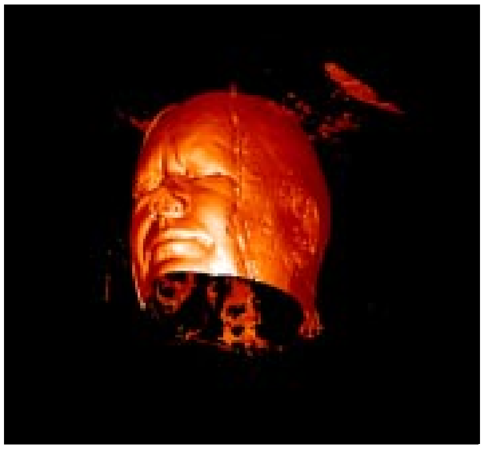


Figure 16: CT reconstruction at 3-mm spacing. No of slices in the stack: 70

As shown in Fig.16, since the density of slices is large and the spacing close hence the reconstruction looks good. All the facial features are clearly visible and there is no step like features on the reconstruction.

Case 5:



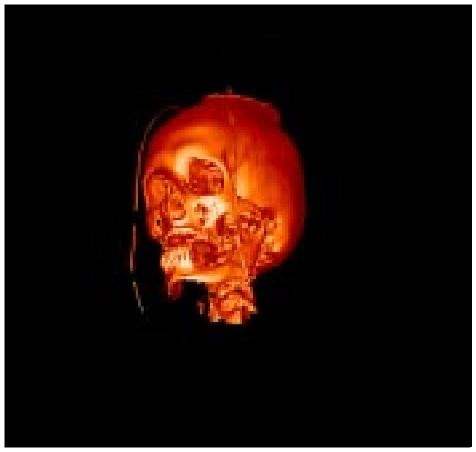


Figure 17: CT reconstruction at 2-mm spacing. No of slices in the stack: 109

As shown in Fig. 17, since the density of slices is large and the spacing close hence the reconstruction looks good. All the facial features are clearly visible.

4.3.2 Analysis

From the above reconstructions it can be seen that the images with the 3mm or more spacing have a marked step like formations. These step like formations on the reconstructions were eliminated by an application of a smoothing filter. The SMOOTH function in PV-Wave uses a boxcar averaging to smooth the image. Different widths for the smoothing window were tried. This resulted in blurring of facial features, hence this method was abandoned.

4.3.3 Conclusions

The above study shows that the data to be used for reconstruction must have at the most a spacing of 3mm between each slice. The most appropriate spacing would be either 1 or 2mm. At this spacing the reconstruction is good and smooth, which can be used for identification purposes. Hence the tissue depth measurement should be from slices with at most 3mm spacing. All these results are based on assuming the visible women data to be the standard.

4.4 Determination of optimal CT spacing with minimum bone feature deformations after interpolation

As mentioned earlier, this research is also to be used in medical reconstruction where there is a need to generate models based on accurate bone-tissue relations. For medical applications errors in tissue depth measurements can be critical.

3D model of the head is generated through interpolation of the head stack data. Interpolation adds new slices in between the existing slices in a head stack. The larger the distance between actual slices the greater the interpolation. The newly added slices through interpolation may be different/deformed from the actual slices at the same spacing. Tissue thickness measurements from such slices may produce erroneous results. The degree of deformation in the bone features varies with the actual CT spacing. This study is to determine optimal CT spacing between actual slices so that the errors in the interpolated slices are minimum. Comparing the interpolated slices with the corresponding original slices to check for errors does this.

4.4.1 Approach

The four head stacks generated for the earlier study are used for this study too. A head stack with a particular spacing was taken and interpolated to obtain same number of slices as in the original data set. For example with an original data set of 209 slices at 1mm spacing, a head stack with 42 slices at 4mm interval was taken (case 3 in the above section) and the data is interpolated to obtain 209 slices as in the original data set. Then individual slices from the data set were taken and compared with the actual slices to check for similarity. Care was taken not to compare the original slices of the two sets and that only the interpolated slices are compared to the original slices.

Interpolation is done along the z axis using the CONGRID function of PV-Wave.

CONGRID function:

CONGRID shrinks or expands the number of elements in image by interpolating values at intervals where there might not have been values before. The resulting image is of the same data type as the input image. It can interpolate data using two interpolation techniques. The nearest neighbor interpolation method sets new data equal to the nearest existing value of image. The other kind of interpolation is bilinear interpolation. The type of interpolation is set based on the keyword *Interp.* In our case bilinear interpolation is used along the " z" axis.

Syntax:

result = CONGRID(image, col, row)

image: The two-dimensional image to resample.

col: The number of columns to be in the resulting image.

row : The number of rows to be in the resulting image.

result: The resampled image or array.

Keywords:

Interp: Specifies the interpolation method. If zero, uses the nearest neighbor method. If nonzero, uses the bilinear interpolation method.

The test data stack is read in and is interpolated using the CONGRID function to obtain the same number of slices using bilinear interpolation.

Data Sets:

Original Data Set: *Visible Female CT Data*

No of slices in the original stack	*: 209*
Spacing between slices	*: 1mm*
Size	*: 512 X 512*
Header	*: 3416*
Type	*: 16 bit gray level*

Test data stacks generated form the original data set:

Data Set 1:

No of slices in the original stack	*: 104*
Spacing between slices	*: 2mm*
No of slices in the stack after interpolation	*: 209*

Data Set 2:

No of slices in the original stack	*: 70*
Spacing between slices	*: 3mm*
No of slices in the stack after interpolation	*: 209*

Data Set 3:

No of slices in the original stack	*: 54*
Spacing between slices	*: 4mm*
No of slices in the stack after interpolation	*: 209*

Data Set 4:

No of slices in the original stack	*: 30*
Spacing between slices	*: 7mm*
No of slices in the stack after interpolation	*: 209*

4.4.2 Results

The following are the results that are obtained.

Data Set 4:

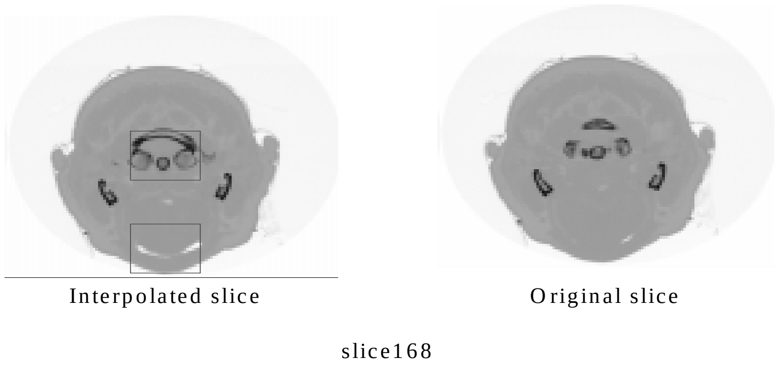


Figure 18: Comparison of slices in data set 4.

In Fig. 18, there is a large variation in the bone features between the original slice and the interpolated slice. Newer bone features are generated in the interpolated slice that is not present in the original slice. Errors in tissue thickness measurements will be high.

Data Set 3:

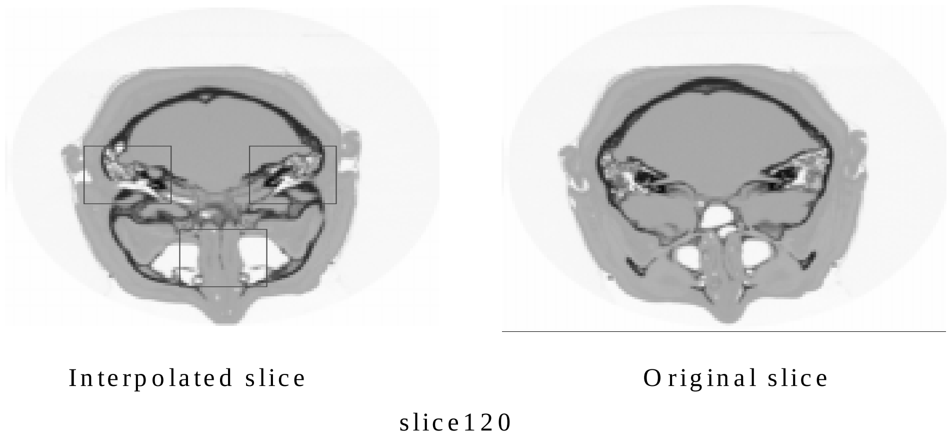


Figure 19: Comparison of slices in data set 3.

In Fig. 19, there is a large variation in the bone features between the original slice and the interpolated slice. Newer bone features are generated in the interpolated slice that is not present in the original slice. Errors in tissue thickness measurements will be high.

Data Set 2:

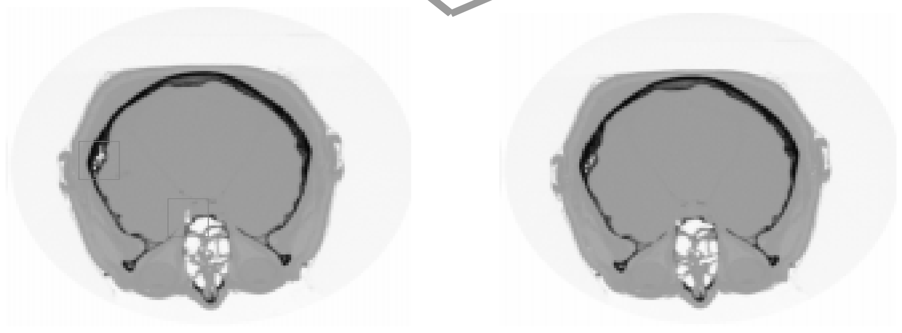


Figure 20: Comparison of slices in data set 1.

In Fig. 20, the original slice and the interpolated slice are similar. Errors in tissue thickness measurements will be low.

Data Set 1:

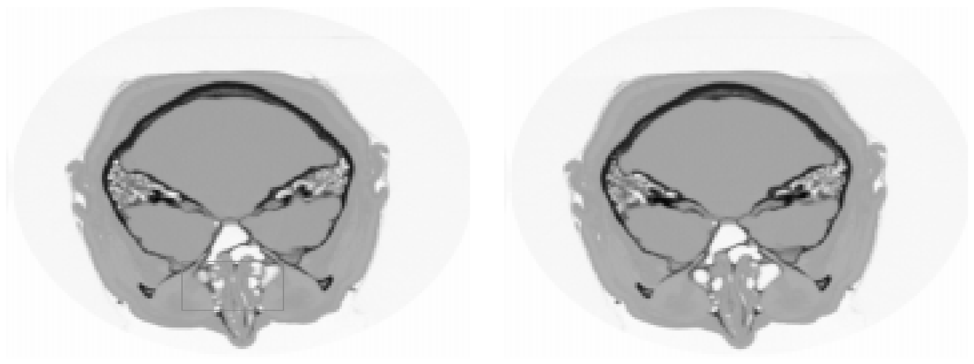


Figure 21: Comparison of slices in data stack: noSlices104.

In Fig. 21, the original slice and the interpolated slice are similar. Errors in tissue thickness measurements will be low.

4.4.3 Conclusions

From the above images it can be seen that the skull and the soft tissue features are accurate after interpolation up to spacing of 3mm between the actual slices. Beyond 3 mm the soft tissue and bone features are distorted and thus undesirable errors can be produced while measuring tissue thickness.

4.5 Conclusion from the Studies

From the above two studies it has been shown that 3mm or less spacing is the optimal CT for generation of tissue thickness database.

CHAPTER 5: *Measurement of tissue thickness over the skull*

As mentioned in chapter 2, the craniofacial reconstruction project is divided into three phases. The first phase involves measurement of tissue thickness over the skull. This chapter describes the methods employed for tissue thickness measurement over the skull. This data set will be used to generate the facemask in Phase 2 of the project.

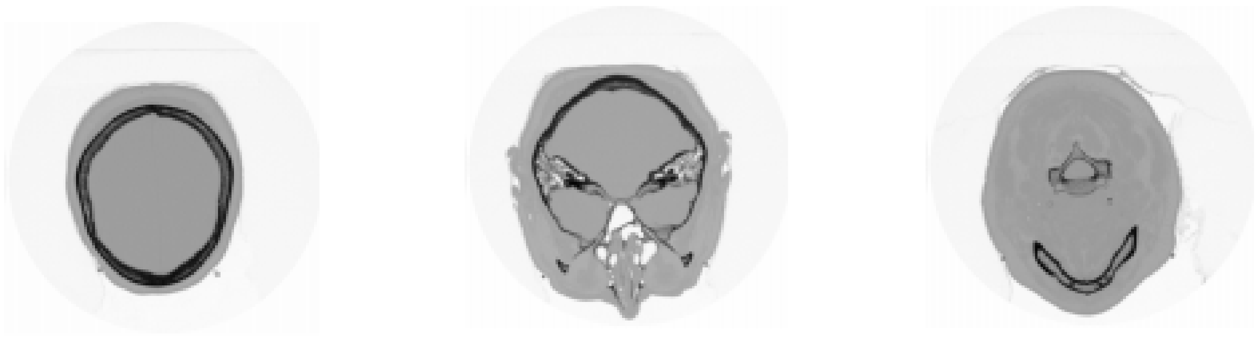
5.1 Tissue thickness measurement considerations

Tissue thickness is measured as the distance between corresponding points, on the outer layer of the bone and the outer layer of the skin on a ray originating from the center of the slice. The center of the slice is defined as the centroid of the points of the outermost contour of the slice image. In order to measure tissue thickness, the outer edge of the skull and the outer edge of the skin need to be identified and then distances have to be measured between them.

Image processing techniques, mentioned in chapter 3 can be used to obtain the boundaries. However, one algorithm cannot be used to extract the edges and measure the tissue thickness in all the slices of the head stack because the boundary contours of the skin and skull vary over the head stack. Regions of certain slices where tissue thickness cannot be measured (like on the nose and on the eye) need to be excluded. The solution to this problem is to divide the head stack into regions with similar boundary structure and apply different boundary detection algorithms in different regions. Since the boundary structures are similar with in a region a single algorithm can be used for the region.

Based on the CT slices three regions of the head stack can be identified, which have similar boundary structures. They are the upper forehead region, the region from the upper eyes to the lower part of the nose and the third is the region below the nose.

The slices in the upper forehead region have elliptical contours (see fig: 26). The slices in the second sub stack (i.e. the region from the upper eyes to the lower part of the nose) have similar boundaries where the thickness measurements over the nose and the eyes are not to be registered (eyes and nose are not part of the facemask). In the third stack, the mouth region needs to be eliminated.



a) Sub Stack *1*　　b) *Sub Stack 2*　　*c) Sub Stack 3*

Figure 22: Sub stack slices.

To apply edge detection algorithms on these sub-stacks, the head stack has to be divided into sub stacks. An automatic method of CT head stack division has been developed based on [20]. This algorithm gives an approximate location of the position of eye’s, nose and mouth based on the skull contour. These detected locations can also to be used to place 3D models of nose, eyes and mouth during the reconstruction [20] in Phase 2.

5.2 Head stack division for tissue thickness measurement

Caludio et. al. [20] in their paper developed a method to identify the location of craiometric points (eye center locations, location of the nose etc.). Based on statistical data on location of these cranial points on the skull and their relationship to the skull morphology, these points are identified from the frontal view image of the skull, by using image-processing techniques. In order to determine the sub stacks this algorithm is slightly modified where first, the location of the craniometric points (location of eyes, nose end) is done in accordance with [20] and then these located points are mapped on to the corresponding slice number of the stack. The stack is divided into sub stacks based on these stack numbers. The following are the steps involved in dividing the head stack into sub stacks with similar boundary structures:

1. Reconstruct the 3D model of the skull using the head stack. Capture the front view image of the skull.
2. Apply noise reduction filters on the image.
3. Identify the positions of eyes, nose and mouth in the image using the algorithm [20] described in 5.2.1.
4. Map the location of these points to the corresponding slice numbers.

5. Divide the stack into sub stacks based on these slice numbers.

Stage 1: Skull Image Capture

The 3D model of the skull is reconstructed as described in chapter 4. The difference here being the reconstruction is performed so that the front view of the skull is visible. Setting the rotational parameters in the SHADE_SURF function to zero does this. In our case the rotation parameters during reconstruction about the X- axis and the Y-axis are set to zero. The function in PV- Wave that does this is given below.

```
SHADE_SURF, INDGEN(2,2), XRange= [min_x, max_x], YRange=
[min_y, max_y], ZRange = [min_z, max_z], /NoData, /Save, Az=0, Ax=0
```

This image is then used for the identification of craniometric points (location of eyes, nose, etc.). The following stages are involved in locating the craniometric points.

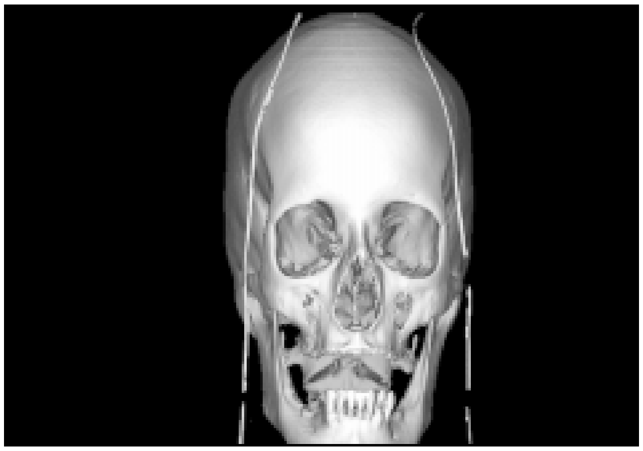


Figure 23: Front View of the skull for craniometric point determination.

Stage 2: Pre-Processing

Preprocessing removes the noise from the image. Blurring the skull image with a linear filter does noise reduction. A 5X5 spatial averaging mask is used to reduce the influence of spurious data, simplifying the segmentation technique. The results of the convolution can be expressed as *f'(x, y) = 1/25 Σ Σ f(x+i, y+i),* where f'(x, y) is the resulting image function after the application of the averaging mask and f(x, y) is the original image and (x, y) are the coordinates of a pixel in the image matrix.

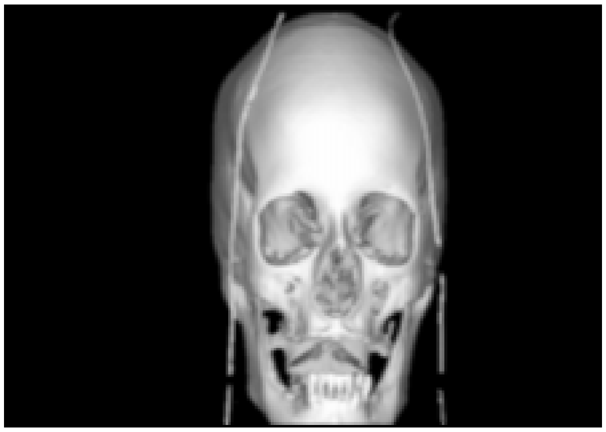


Figure 24: Smoothed skull.

Stage 3: Contour Extraction and Identification of Craniometric Points

The outer contour of the skull is extracted. First Roberts edge detector is applied (described in Chapter 3) to detect the boundaries in the image. The outer contour of the skull is extracted using a rotating ray method. The outer boundary points are the maximum radii points along a rotating ray, originating at the centroid of the image. The algorithm that describes the outer contour extraction is given below. The figure below shows the outer contour of the skull using the algorithm.

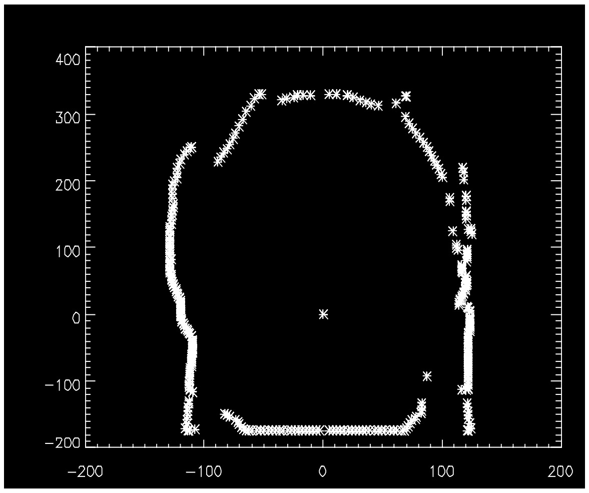


Figure 25: Outer contour of the skull.

The algorithm for skull contour extraction is given in the flow chart below.

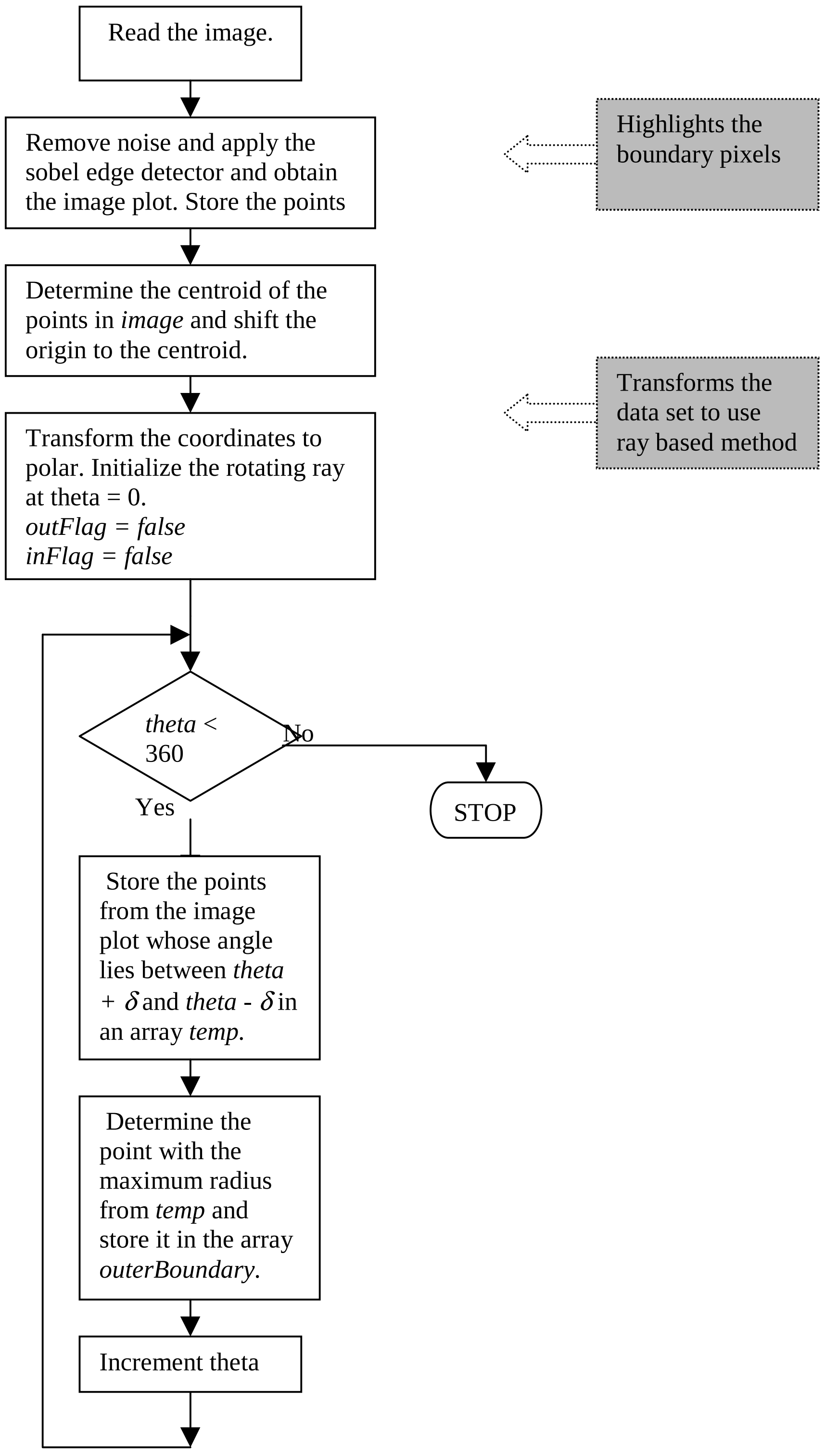

Read the image.
Remove noise and apply the sobel edge detector and obtain the image plot. Store the points
Highlights the boundary pixels
Determine the centroid of the points in *image* and shift the origin to the centroid.
Transforms the data set to use ray based method
Transform the coordinates to polar. Initialize the rotating ray at theta = 0. *outFlag = false* *inFlag = false*
theta < 360
No
STOP
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta

Figure 26: Flow chart for outer contour determination.

Centroid determination

Once the outer edge of the skull is determined the centroid of the skull contour is determined by the first moment of the contour points. The centroid of the points is give by

$$x_c = \frac{\sum \sum x.\, C_0(x, y)}{\sum \sum C_0(x,y)} \qquad y_c = \frac{\sum \sum y.\, C_0(x, y)}{\sum \sum C_0(x,y)}$$

where $C_0(x,y)$ is the skull contour points obtained in stage 3 and (x_c,y_c) is the centroid of the contour.

Location of craniometric points

After determining the centroid, the location of the craoimetric points is based on a vertical axis which is defined as the vertical line through the x coordinate of the centroid. These locations of craniometric points are based on the statistical data collected from skulls by [21] [22]. They give the location of the eye centers and nose with respect to the vertical axis. The location of the points is shown in the figure below.

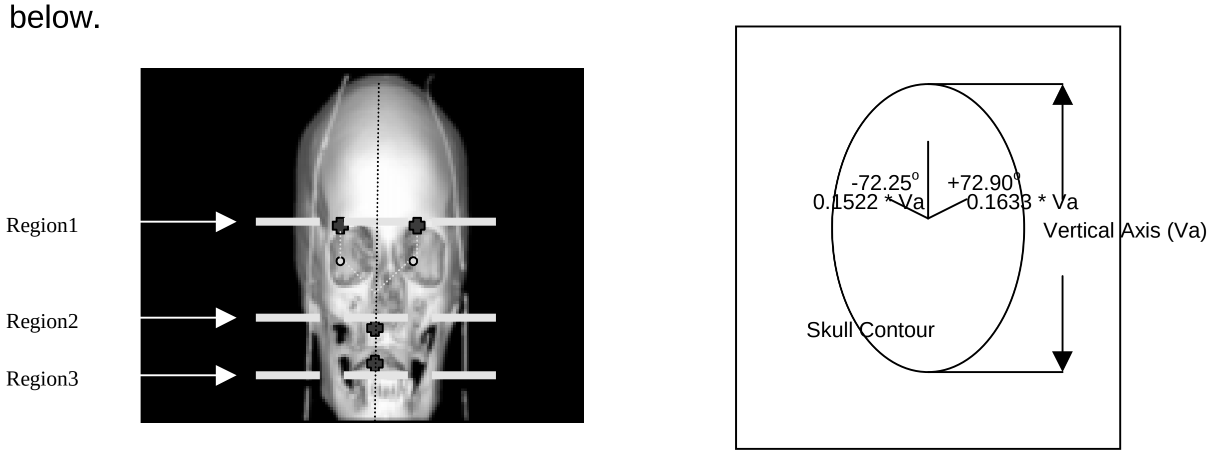


Figure 27: Location of craniometric points.

Element to Centroid	*Distance/Vertical Length*
Left orbit center	0.1633 +/- 0.0167
Right orbit center	0.1522 +/- 0.0126
Element to the vertical axis	*Tilt*
Left orbit center	+72.90^o +/- 11.50^o
Right orbit center	-72.25^o +/- 10.80^o

Table 5: Location of craniometric points. Ref [20]

Left/right orbit center is the location of the center of the left/right eye orbit in the skull image.

Determining the Length of the Vertical Axis

The vertical axis is defined as the line that passes through the centroid dividing the skull contour in half in the vertical plane (Fig. 31). From the skull contour data set the points whose angle is +90^0 and - 90^0 are determined (these are the points corresponding to the two maxima points on the skull contour). The distance between the two points (in this case the sum of the radius) gives the length of the vertical axis.

From the skull contour obtained previously, the vertical points obtained are at 230 pixels and 246 pixels.

Upper vertex length = 230 pixels

Lower vertex length = 246 pixels

Length of the vertical axis = *230 + 246 = 476 pixels*.

Based on the skull image the distances to various points on the skull are calculated from table 1. Table 2 below gives the resulting distances.

Vertical Axis Length:	*476 pixels*
Position of left orbit center	*72.44 pixels*
Position of right orbit center	*77.73 pixels*

Table 6: Lengths of craniometric point locations.

Left orbit center is the location of the center of the left eye.

Stage 5: Mapping to slice numbers:

The location of these craniometric points with respect to the vertical axis gives the approximate contribution of each region in the entire head stack. Based on the individual contributions the head stack is divided. This is explained below.

The visible female head stack has 209 slices. The length of the vertical axis calculated above is directly proportional to the number of slices. From the location of the cranoimetric points, the head sub stack slices are calculated below.

From Fig. 27, the contribution of the forehead region to the length of the vertical axis is = (Upper vertical axis length – distance to left orbit center cos(-72.25))

The distance between nasion (the center between the two eyes) and nasospinale (lower tip of the nose) is approximately 5 times the distance to the left orbit center (calculated above Table6) [20]. Hence the contribution of the nose and the eyes region to the length of the vertical axis is = 5 X distance to left orbit center.

If the total length of the vertical axis is proportional to "N" slices in the head stack, the slices in each region is given by

$$\frac{\textit{N X contribution of region to vertical axis}}{\textit{length of the axis}}$$

Calculations:

Sub Stack 1 (Forehead Region slices):

$$\frac{209 \text{ X contribution of forehead slices}}{\text{length of the axis}} = \frac{209 \text{ X } (246 - 72.44 \cos(-72.25))}{476} = 98$$

Sub Stack 2 (Nose Regions Slices):

$$\frac{209 \text{ X contribution of nose slices}}{\text{length of the axis}} = \frac{209 \text{ X } (5 \text{ X } 72.44 \cos(-72.25))}{476} = 49$$

Sub Stack 3 (Mouth Region Slices):

Total no of slices - sub stack 1 - sub stack 2 = 209 - 98 - 49 = 62 slices

Comparison with the original stack:

Region	Slice number from observing the stack	Calculated from algorithm
Sub Stack 1	92	98
Sub Stack 2	53	49
Sub Stack 3	64	62

Table 7: Comparison of sub stack division.

It should be noted that this sub stack generation algorithm only gives an approximate division. Different results were obtained in different trials. The results presented here are the "*best results"* obtained in the trials which were closest to the actual stack numbers. The location of craniometric points is highly sensitive to the location of the centroid and the determination of the vertical axis length. The locations are based on statistical data available on skull and may not be true in all the cases. Hence in the tool this algorithm should be used to give only an approximate head stack division. Once the algorithm gives the sub stack division, it should be fine-tuned to get the actual stack division. This facility for fine-tuning is provided later in the tissue thickness measurement tool. If the sub stack division is already known while generating the head stack, these values can be directly used in the tool, bypassing this algorithm.

The sub stacks in the visible female head stack are

Sub Stack 1: Slices 0 -98
Sub Stack 2: Slices 99- 128
Sub Stack 3: Slices 129 - 209

After the sub stack division is complete, the boundaries of the outer layer of the skin and the outer layer of the skull need to be extracted for tissue thickness measurement.

5.3 Outer/ Inner boundary detection and thickness measurement.

The following procedure is adopted to determine the inner and outer boundary in the slices for tissue thickness measurements.

Stage 1: Edge Detection

In each sub stack, edge detection procedures are applied. There are three categories of edge detection procedures as described in chapter 3. Initial trials using thresholding and histogram based edge detection did not prove to be very satisfactory. Sobel edge detector provided very sharp edges but also enhanced the noise in the image giving rise to spurious edges. Roberts edge detector has a smoothing effect, which reduces the noise in the images. Hence this edge detector was used. The convolution masks for the edge detector are given by

$$h1 = \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix} \quad \text{and} \quad h2 = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$$

The magnitude of the edge is computed as

$$|g(i,j) - g(i+1, j+1)| + |g(i,j+1) - g(i+1, j)|$$

Category 3 operators were also tried which utilize the higher level knowledge present in the image. A model for multiple boundary detection using Active Contour Models was developed. The advantage of using models was that they utilized the higher level knowledge in the image. The slices in a sub stack have similar boundary structures. They only differ in their relative sizes. The idea behind using a dual active contour model was if the active contour could be initialized on one slice, the remaining slices could be tracked based on the position of the contour in the original slice. This model however abandoned as it has problems of driving the contour to boundary concavities.

Stage 2: Outer boundary determination

After the edge detector was applied the outer boundary (the outer edge of the skin) is extracted using the following procedure. The outer contour of the skull is extracted. First Roberts edge detector is applied (described in Chapter 3) to detect the boundaries in the image. The outer contour of the slice is extracted using a *rotating ray method*. The outer boundary points are the maximum radii points along a rotating ray, originating at the centroid of the image. The edge detected data is converted to polar coordinates with the origin at the centroid of the image. A rotating ray is initialized at the center. This ray sweeps across the slice in steps of 1degrees. In each step the maximum radii point on the ray is obtained which is a point on the outer boundary. All such points for a 360-degree sweep are obtained which gives the outer boundary of the slice.

Stage 3: Inner Boundary Determination

Once the outer boundary is determined, the inner boundary is obtained by eliminating the outer boundary from the image and repeating the rotating ray method. The inner boundary is the boundary of the skull. After the outer boundary is obtained as described earlier, the outer boundary points are removed from the original edge detected data set. This gives the inner slice. The contour is determined by repeating the procedure described in Stage 1.

Stage 4: Exclusion of irrelevant regions

Regions where tissue thickness is not measurable or irrelevant like the nose, eyes etc. are to be removed from the inner and outer contours. The curvature at various points on the boundary contour is determined and the regions where there is a sudden change in the curvature are excluded from the boundary. The reason behind using curvature to exclude unwanted regions is on the outer and the inner contours where these regions (nose, eyes, mouth etc.) are present, there is a sudden change in the boundary curvature at the start and the end of such regions. These two *sudden curvature change* points are identified and the region between them is excluded from the boundary contour.

Stage 5: Tissue thickness measurement

Once the inner and the outer boundaries are obtained and the unwanted region excluded, tissue thickness are a determined by measuring the distances between the inner and the outer boundary points. A point on the inner contour is determined and the corresponding point on the outer contour is determined. The distance between the two points gives the tissue thickness.

The following section describes the implementation of the above methods in each of the sub stacks.

Sub Stack 1 (fore head region):

The following flow chart describes the for tissue thickness measurement in the forehead region.

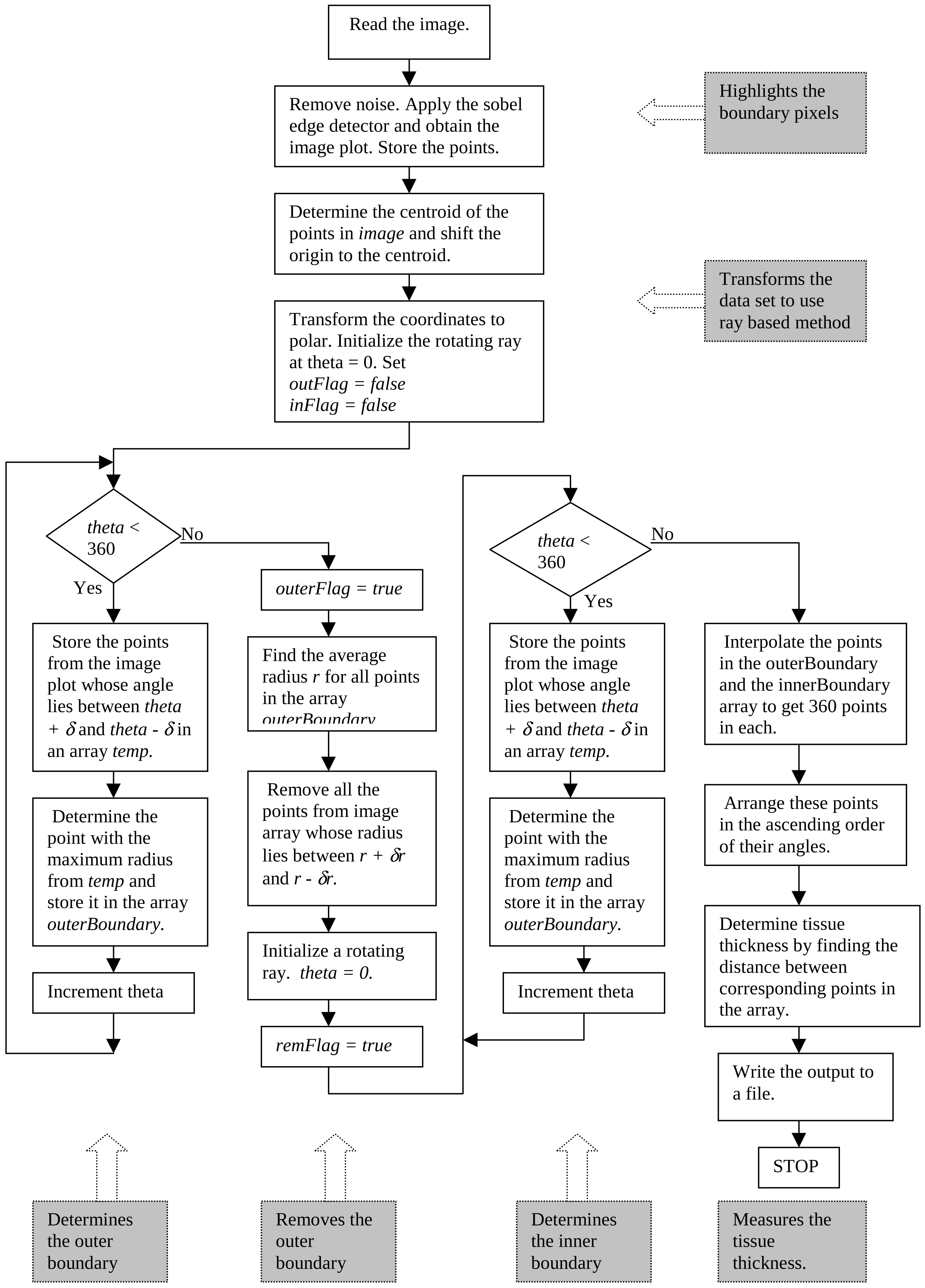
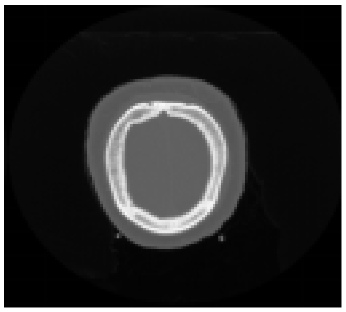

Read the image.
Remove noise. Apply the sobel edge detector and obtain the image plot. Store the points.
Highlights the boundary pixels
Determine the centroid of the points in *image* and shift the origin to the centroid.
Transform the coordinates to polar. Initialize the rotating ray at theta = 0. Set *outFlag = false* *inFlag = false*
Transforms the data set to use ray based method
theta < 360
No
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta
outerFlag = true
Find the average radius *r* for all points in the array outerBoundary
Remove all the points from image array whose radius lies between *r* + δr and *r* - δr.
Initialize a rotating ray. *theta = 0.*
remFlag = true
theta < 360
No
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta
Interpolate the points in the outerBoundary and the innerBoundary array to get 360 points in each.
Arrange these points in the ascending order of their angles.
Determine tissue thickness by finding the distance between corresponding points in the array.
Write the output to a file.
STOP
Determines the outer boundary
Removes the outer boundary
Determines the inner boundary
Measures the tissue thickness.

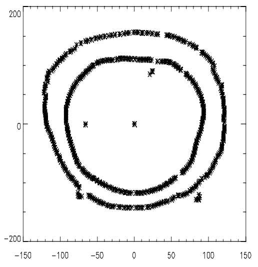
Figure 28: Flow chart for tissue thickness measurement in head stack 1.

Results:

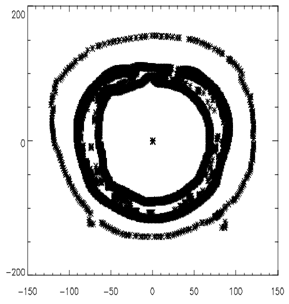
The following are the results after using the above algorithms on sub stack 1.


a) Original CT slice through the forehead region.

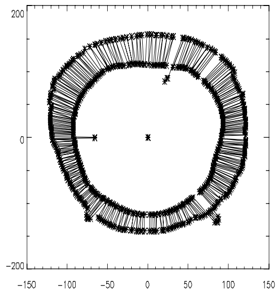
b) Roberts edge detector applied .



c) Inner and outer edge plots.



d) Inner and outer plot superimposed with the original slice.



e) Thickness plot. 360 measurements per slice.

Figure29: Tissue thickness measurement in sub stack 1.

Tissue Thickness Measurement in Sub Stack 2 (eyes and nose region):

In this sub stack the regions with the eyes and nose need to be removed and tissue thickness is measured in the remaining portion of the slice. The inner and the outer contour are extracted by the method described in section 5.2. There is a sudden change in the curvature of the outer boundary at the start and end of the nose region. These two points are identified and the region between the two points is excluded. Tissue thickness is measured over the remaining part of the slice. The following flow chart describes the tissue thickness measurement in the nose/eye region.

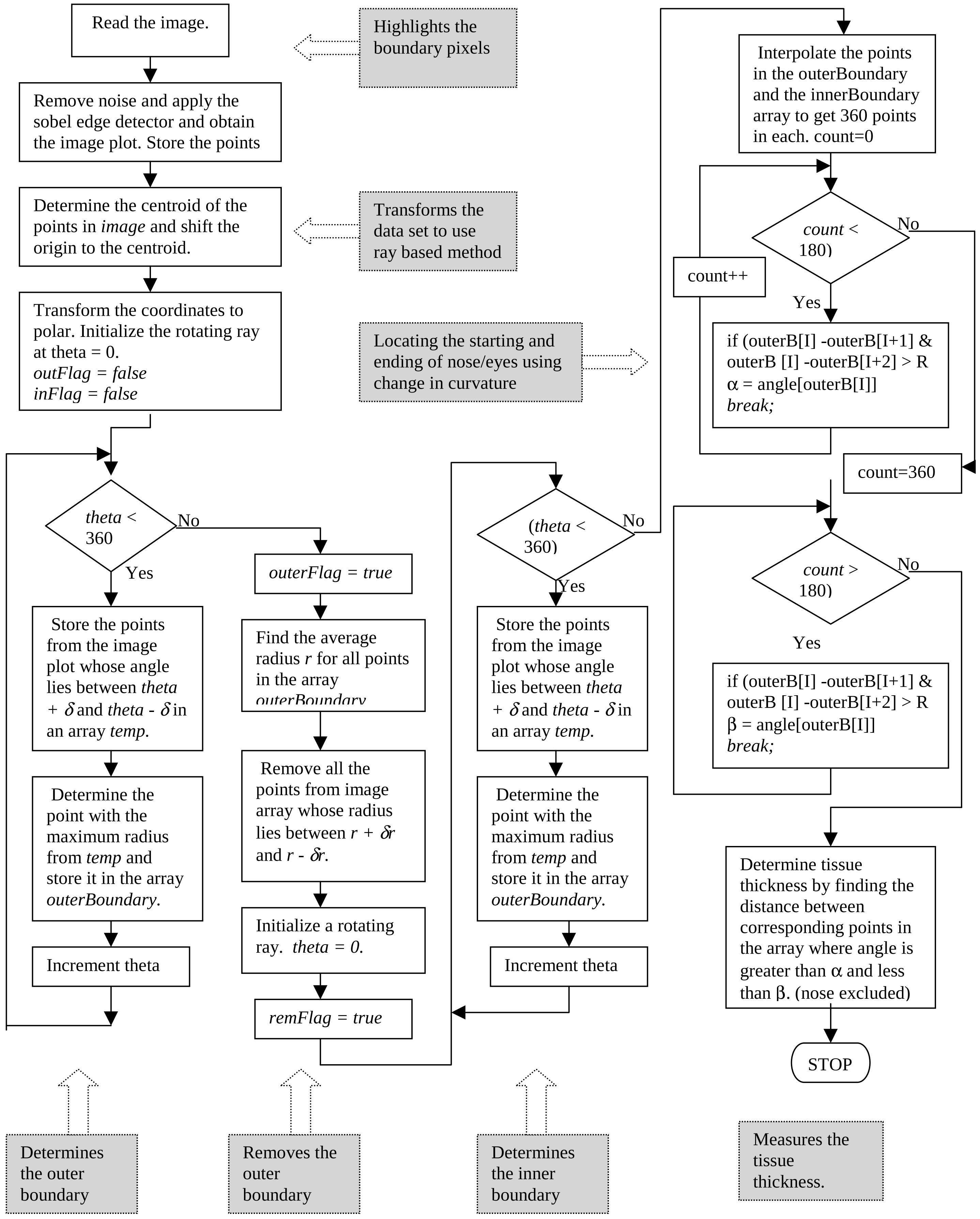
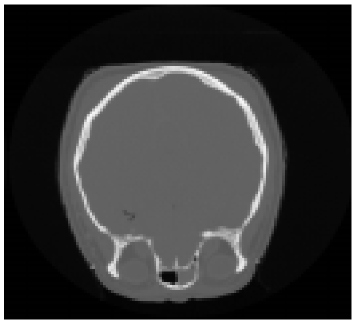

Read the image.
Highlights the boundary pixels
Remove noise and apply the sobel edge detector and obtain the image plot. Store the points
Determine the centroid of the points in *image* and shift the origin to the centroid.
Transforms the data set to use ray based method
Transform the coordinates to polar. Initialize the rotating ray at theta = 0. *outFlag = false* *inFlag = false*
Locating the starting and ending of nose/eyes using change in curvature
theta < 360
No
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta
outerFlag = true
Find the average radius *r* for all points in the array *outerBoundary*
Remove all the points from image array whose radius lies between *r* + δr and *r* - δr.
Initialize a rotating ray. *theta = 0.*
remFlag = true
(*theta* < 360)
No
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta
Interpolate the points in the outerBoundary and the innerBoundary array to get 360 points in each. count=0
count < 180)
No
Yes
count++
if (outerB[I] -outerB[I+1] & outerB [I] -outerB[I+2] > R α = angle[outerB[I]] *break;*
count=360
count > 180)
No
Yes
if (outerB[I] -outerB[I+1] & outerB [I] -outerB[I+2] > R β = angle[outerB[I]] *break;*
Determine tissue thickness by finding the distance between corresponding points in the array where angle is greater than α and less than β. (nose excluded)
STOP
Determines the outer boundary
Removes the outer boundary
Determines the inner boundary
Measures the tissue thickness.

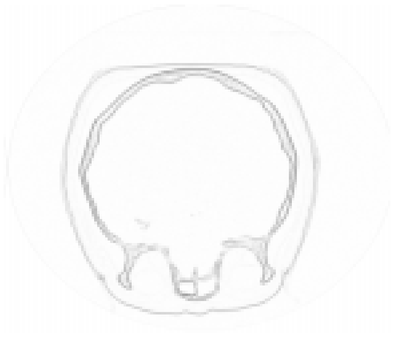
Figure 30: Flow chart for tissue thickness measurement in sub stack 2.

Results:

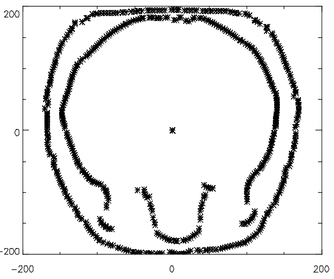
The following results are obtained using the above algorithm on sub stack 2.



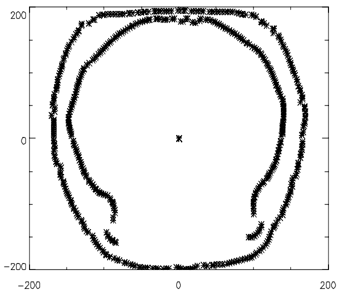
a) Original CT slice through the nose region.



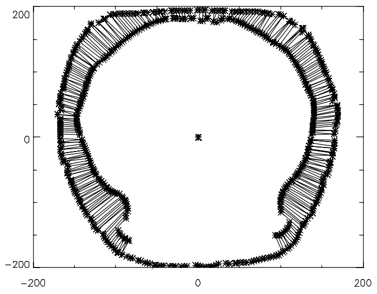
b) Roberts edge detector



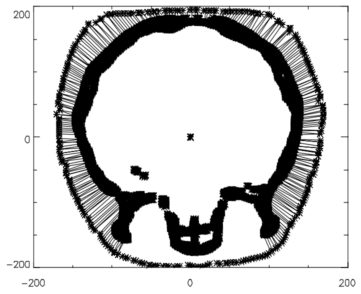
c) Inner and outer edge plots.



d) Inner and outer edge plots with the eyes and the nose regions removed.



e) Thickness plot. 360 measurements per slice.

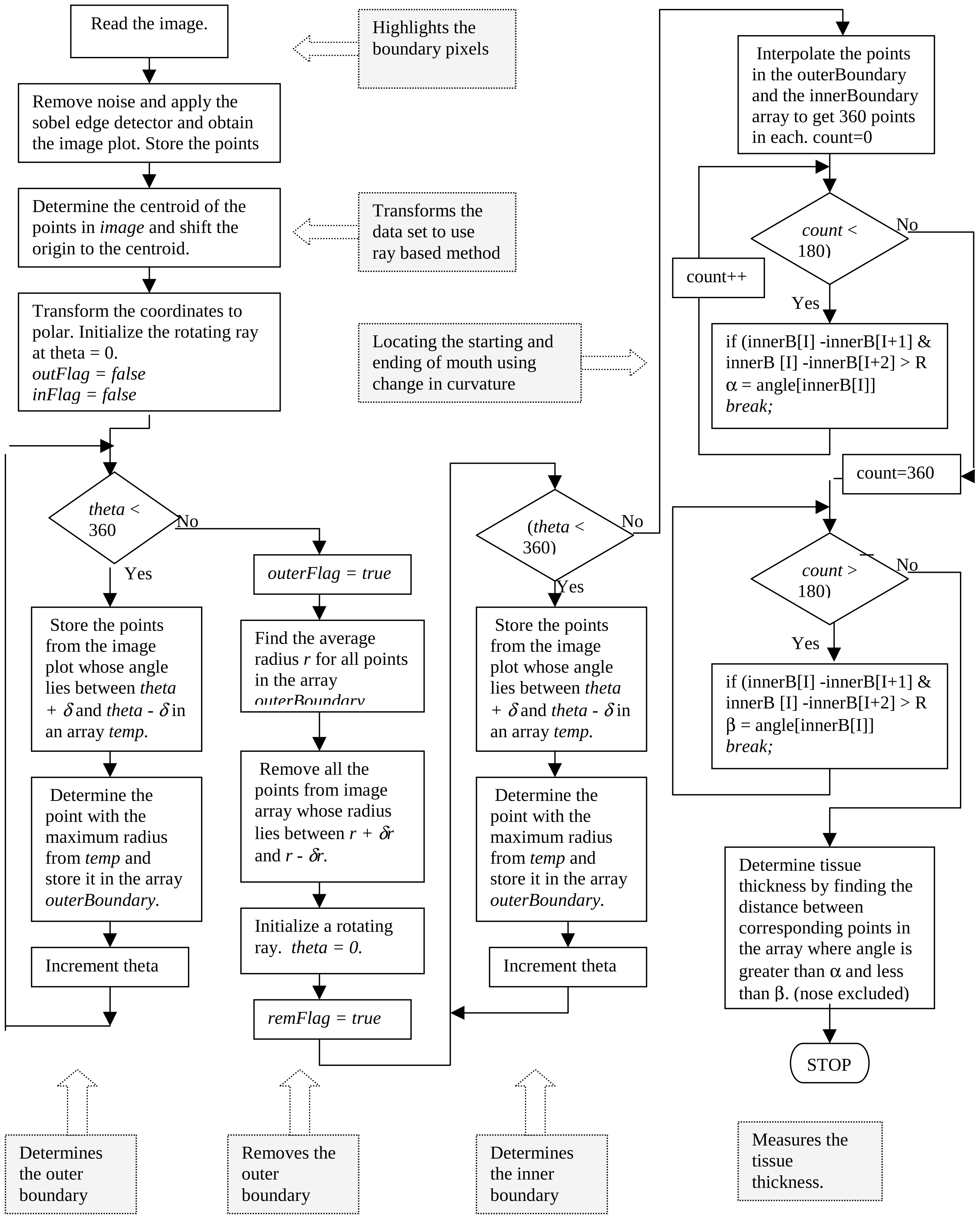


f) Thickness plot superimposed with the original slice.

Figure 31: Tissue thickness measurement in sub stack 2.

Tissue Thickness Measurement in Sub Stack 3 (eyes and nose region):

The inner and the outer contour are extracted by the method described for sub stack 1. The mouth region is excluded (if required) by curvature studies as in method 2. The difference here is that there is a sudden change in the curvature of the inner boundary instead of the outer boundary. The starting and ending points of the mouth are identified. Tissue thickness is measured over the remaining part of the slice. The following flow chart describes the tissue thickness measurement in sub stack 3.

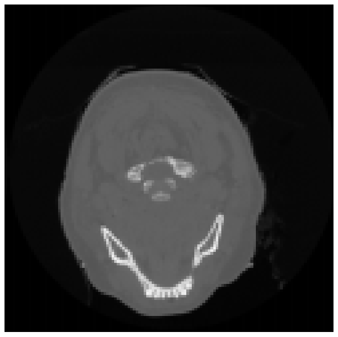


Read the image.
Remove noise and apply the sobel edge detector and obtain the image plot. Store the points
Determine the centroid of the points in *image* and shift the origin to the centroid.
Transform the coordinates to polar. Initialize the rotating ray at theta = 0. *outFlag = false* *inFlag = false*
Highlights the boundary pixels
Transforms the data set to use ray based method
Locating the starting and ending of mouth using change in curvature
theta < 360
No
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta
outerFlag = true
Find the average radius *r* for all points in the array *outerBoundary*
Remove all the points from image array whose radius lies between *r* + δr and *r* - δr.
Initialize a rotating ray. *theta = 0.*
remFlag = true
(*theta* < 360)
No
Yes
Store the points from the image plot whose angle lies between *theta* + δ and *theta* - δ in an array *temp*.
Determine the point with the maximum radius from *temp* and store it in the array *outerBoundary*.
Increment theta
Interpolate the points in the outerBoundary and the innerBoundary array to get 360 points in each. count=0
count < 180)
No
Yes
count++
if (innerB[I] -innerB[I+1] & innerB [I] -innerB[I+2] > R α = angle[innerB[I]] *break;*
count=360
count > 180)
No
Yes
if (innerB[I] -innerB[I+1] & innerB [I] -innerB[I+2] > R β = angle[innerB[I]] *break;*
Determine tissue thickness by finding the distance between corresponding points in the array where angle is greater than α and less than β. (nose excluded)
STOP
Determines the outer boundary
Removes the outer boundary
Determines the inner boundary
Measures the tissue thickness.

Figure 32:Flow chart for tissue thickness measurement in sub stack 3.

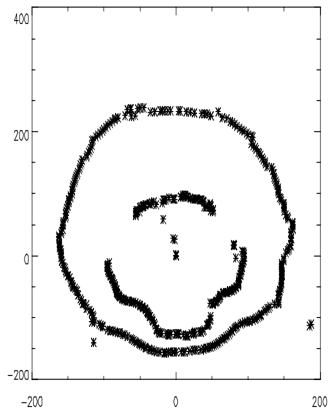
Results:

The following are the results from the algorithms above for sub stack 3.

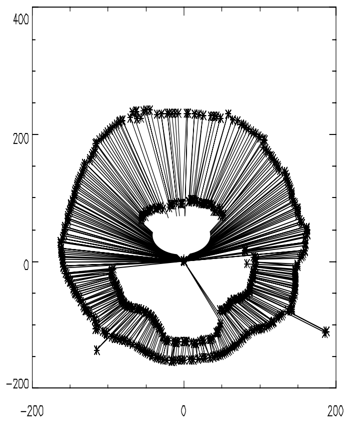


a) Original CT slice through the mouth region.

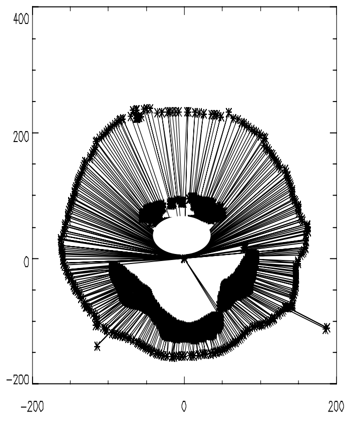
b) Roberts edge detector applied .



c) Inner and outer edge plots.



d) Thickness plot. 360 measurements per slice.



e) Thickness plot superimposed with the original slice.

Figure 33: Tissue thickness measurement in sub stack 3

5.4 Accuracy of Tissue Thickness Measurement:

The tissue thickness measured in the earlier sections is based on edge detection techniques. A statistical analysis of the data was done to determine the accuracy of tissue thickness measurements in each sub stack.

Approach

Twenty slices form each sub-stack were identified and tissue thickness is measured at 10 points on each slice using the automated methods mentioned in the earlier sections. Physical measurements were then taken to determine the actual tissue thickness. The two measurements were then compared to determine the accuracy of tissue thickness measurement.

Test Data

In each sub stack 20 slices were isolated. The visible female CT head stack was used for the study.

Tissue Thickness Measurement Locations

Tissue thickness was measured in the sub stacks at following locations.

Sub stack 1

5 points at an angle of 18^o each, starting at 0^o till 90^o.
5 points at an angle of -18^o each, starting at 180^o till 90^o.

Number of points per slice	:10
Number of slices in the sub stack	: 20
Total number of thickness measurements in the sub stack	:200

Table 8: Measurements in sub stack 1

Sub Stack 2

5 points at an angle of 9^o each, starting at 0^o till 45^o.
5 points at an angle of -9^o each, starting at 180^o till 135^o.

Number of points per slice	:10
Number of slices in the sub stack	: 20
Total number of thickness measurements in the sub stack	:200

Table 9: Measurements in sub stack 2

Sub Stack 3

5 points at an angle of 9^o each, starting at 0^o till 45^o.
5 points at an angle of -9^o each, starting at 180^o till 135^o.

Number of points per slice	:10
Number of slices in the sub stack	: 20
Total number of thickness measurements in the sub stack	:200

Table 10: Measurements in sub stack 3

Data files generated per sub stack

Number of data files generated from automated measurements	: 20 (1 per slice)
Number of data files generated from physical measurements	: 20 (1 per slice)
Total number of data files	: 40 per sub stack

Tissue thickness from automated measurement

Tissue thickness at the above mentioned points were measured, using the automated algorithms described earlier. These measurements were written in to a set of files.

Physical measurements

Physical measurements were carried out using Scion Image software toolkit. Scion Image provides a tool to determine the number of pixels between two points. The following procedure was adopted in each slice to determine the tissue thickness measurement.

1. Identify the center of the slice. Since all the slices are 512X 512 and positioned at the center, the point (256, 256) was taken as the centroid.
2. Mark radial lines from the center at angles for which tissue thickness is to be measured.
3. Magnify the region where tissue thickness is to be measured by 50%.
4. Using the distance-measuring tool measure the distance between the outer edge of the skull and the outer edge of the skin.
5. This tool gives the thickness in pixels.
6. All such measurements are recorded and written to a file.

Figure 38 shows the Scion Image tool.

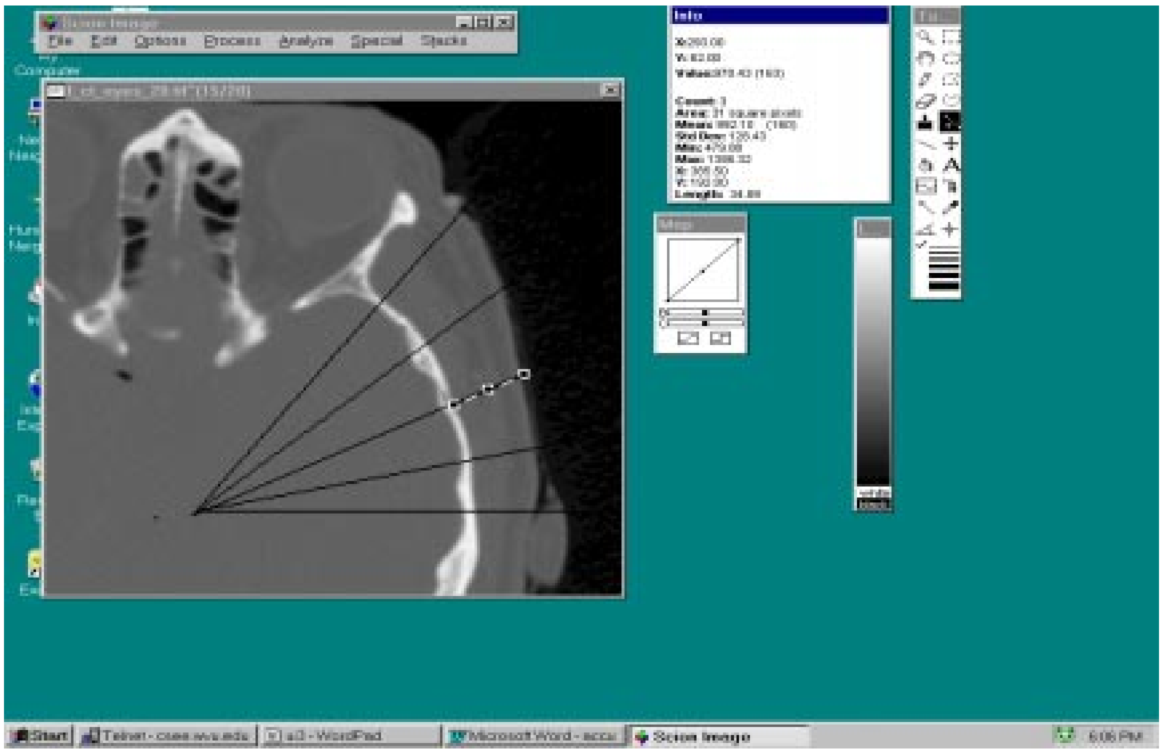


Figure 34: Using Scion Image for physical tissue measurement.

Analysis:

The following plots show the actual (from Scion Image) and the measured (from the tool) tissue thickness measurements in each sub stack.

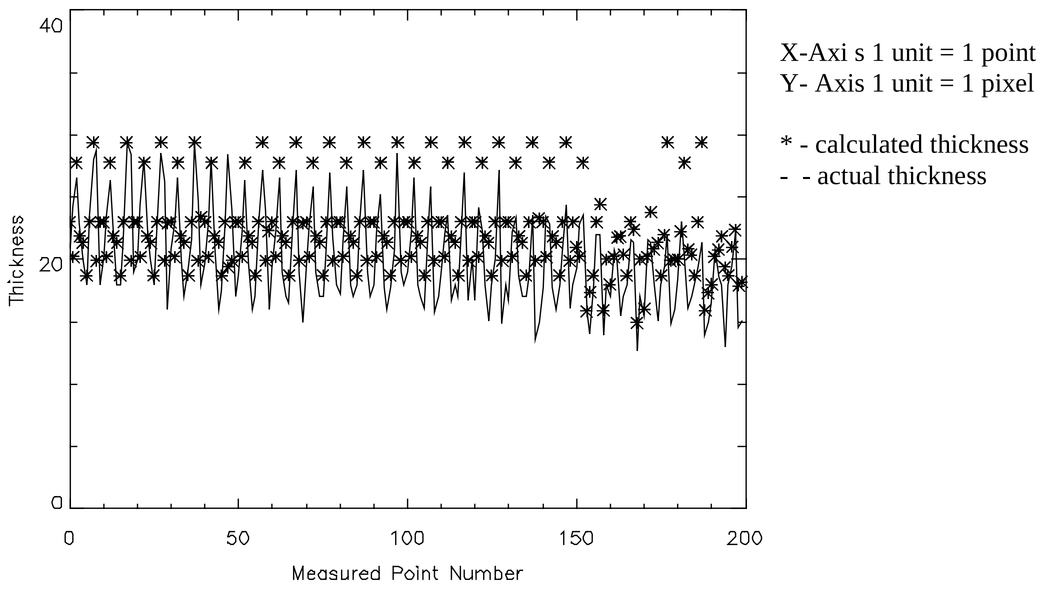


Figure 35: Actual Thickness overlapped with Measured Thickness plot in sub stack 1.

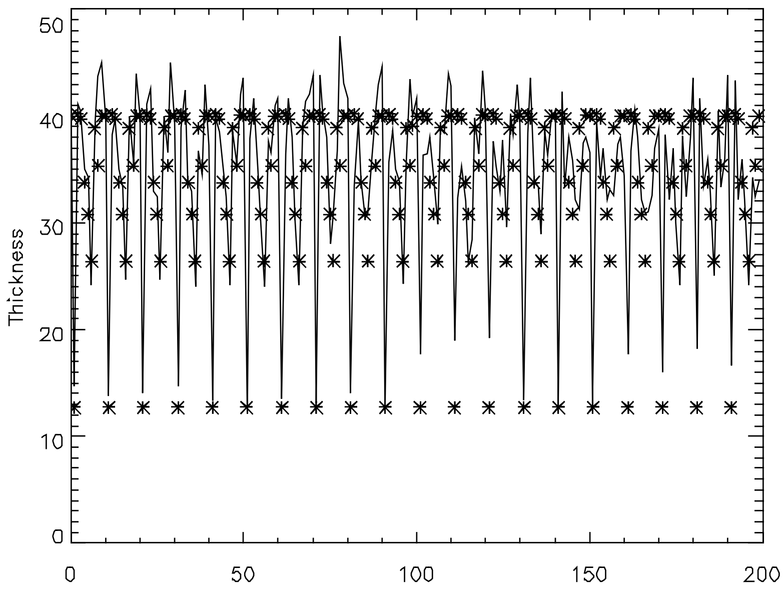


Figure 36: Actual Thickness Overlapped with Measured Thickness plot in sub stack 2.

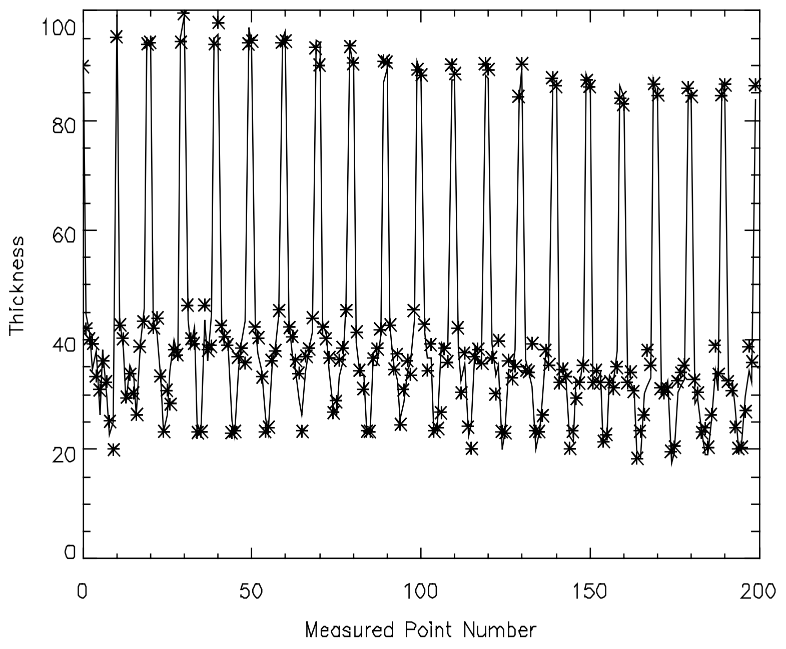


Figure 37: Actual Thickness Overlapped with Measured Thickness plot in sub stack 3.

Error Plots: The error in tissue thickness is determined for each slice in a sub stack and plotted. Error (pixels) = actual thickness – measured thickness

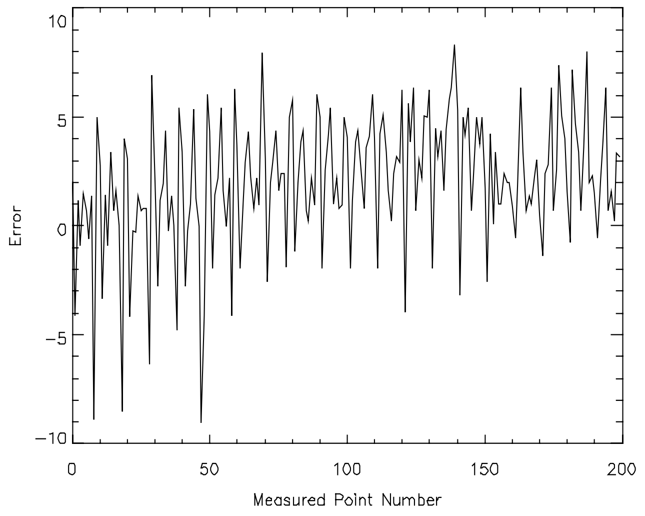


Figure 38: Error plot in sub stack 1.

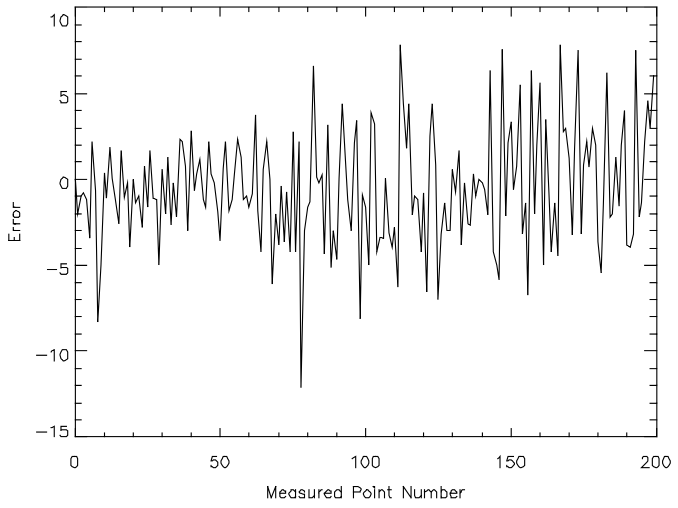


Figure 39: Error plot in sub stack 2.

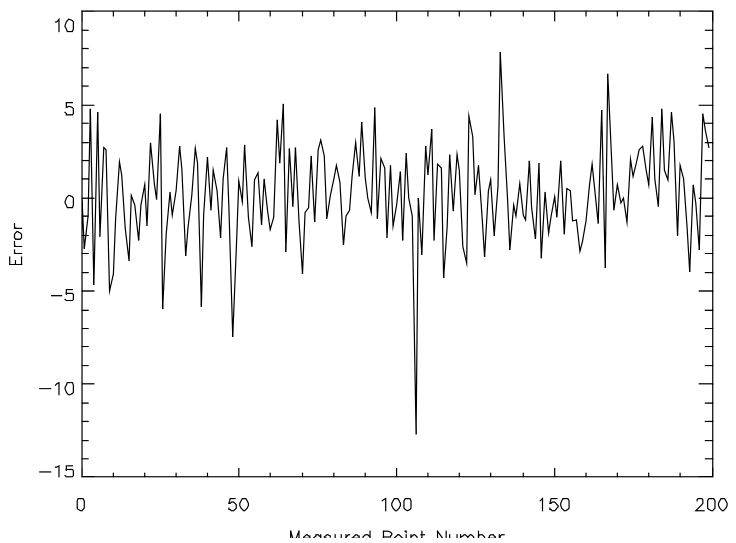


Figure 40: Error plot in sub stack 3.

Calculations:

From the above plots the standard deviation of the error in tissue thickness measurement is determined. In each sub sack, the error array is determined where,

error_ array = Actual thickness – Measured Thickness

The standard deviation of the error array is computed. The following are the standard deviations of the error array in each sub stack.

Sub stack 1 error array standard deviation : 2.5184 pixels

Sub stack 2 error array standard deviation : 3.8134 pixels

Sub stack 3 error array standard deviation : 2.9975 pixels

Average error in the head stack: *2.9184+3.8134+2.5975 /3 =* 3.1097 pixels

Conclusion:

The length of each pixel being 0.28 mm the error in measuring the tissue thickness is in the order of 0.87mm over the head stack.

CHAPTER 6

Tissue Thickness Measurement Tool

This chapter describes the "Tissue Thickness Measurement Tool" and its user interface. PV Wave’s support for GUI using X-Motif is used to develop the tool user interface.

Tissue thickness measurement is done in two steps. These are

Step1: Formatting the CT Data.

Step2: Using the tool to measure the tissue thickness.

These steps are discussed below.

6.1 Formatting the CT data

Before the CT slices can be used to measure tissue thickness, a head stack has to be generated and headers are to be removed from each slice. The stack needs to be saved in a .tif format to be read by PV Wave. All this is done using Scion Image [17].

The following are the steps involved for CT stack generation using Scion Image.

Step 1: From the *file* menu select *import*.

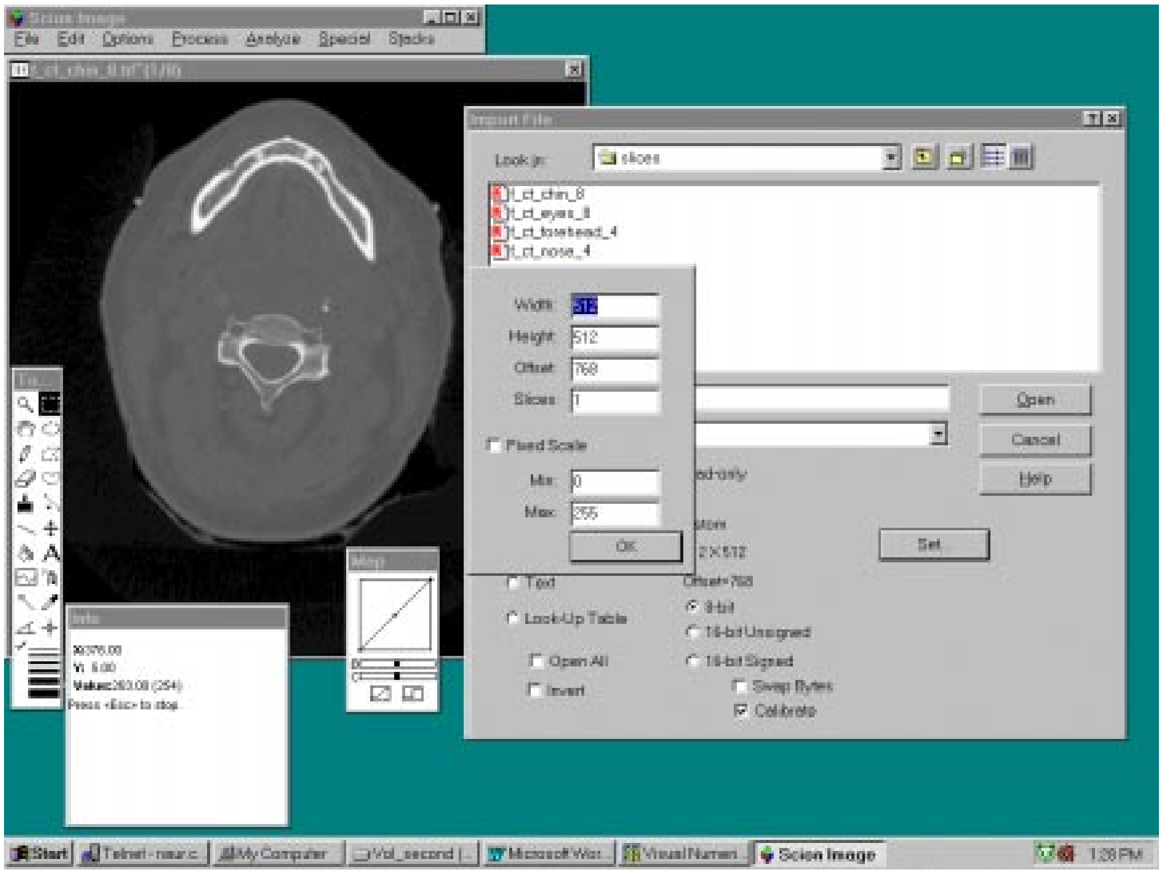
Step 2: In the dialog box that appears select all the files that are to be included in the head stack.

Step 3: Click the *set* option and specify the slice dimensions and the header size of the slices.

Step 4: Select the suitable file types (16 - bit unsigned, Swap Bytes) options based on the data type. Scion Images displays all the selected slices in consecutive order, with each slice in a separate window.

Step 5: Select the *Stack* menu option and click on *Windows to Stack*. Scion Image collects all the open windows into a stack. One can navigate through the stack using the *> or <* keys. Unwanted slices can be removed form the stack by traversing to the particular slice using the > or < key and then selecting *Add Slice or Delete Slice* from the *stack* menu.

Step 6: Save the stack as a .tif.



Look in: slices

f_ct_chin_8
f_ct_eyes_8
f_ct_forehead_4
f_ct_nose_4

Width: 512
Height: 512
Offset: 768
Slices: 1
Fixed Scale
Min: 0
Max: 255
OK

512 X 512
Offset=768
Text
Look-Up Table
Open All
Invert
[✓ 8-bit]
16-bit Unsigned
16-bit Signed
Swap Bytes
[✓ Calibrate]

X: 376.00
Y: 5.00
Value: 263.00 (254)
Press <Esc> to stop

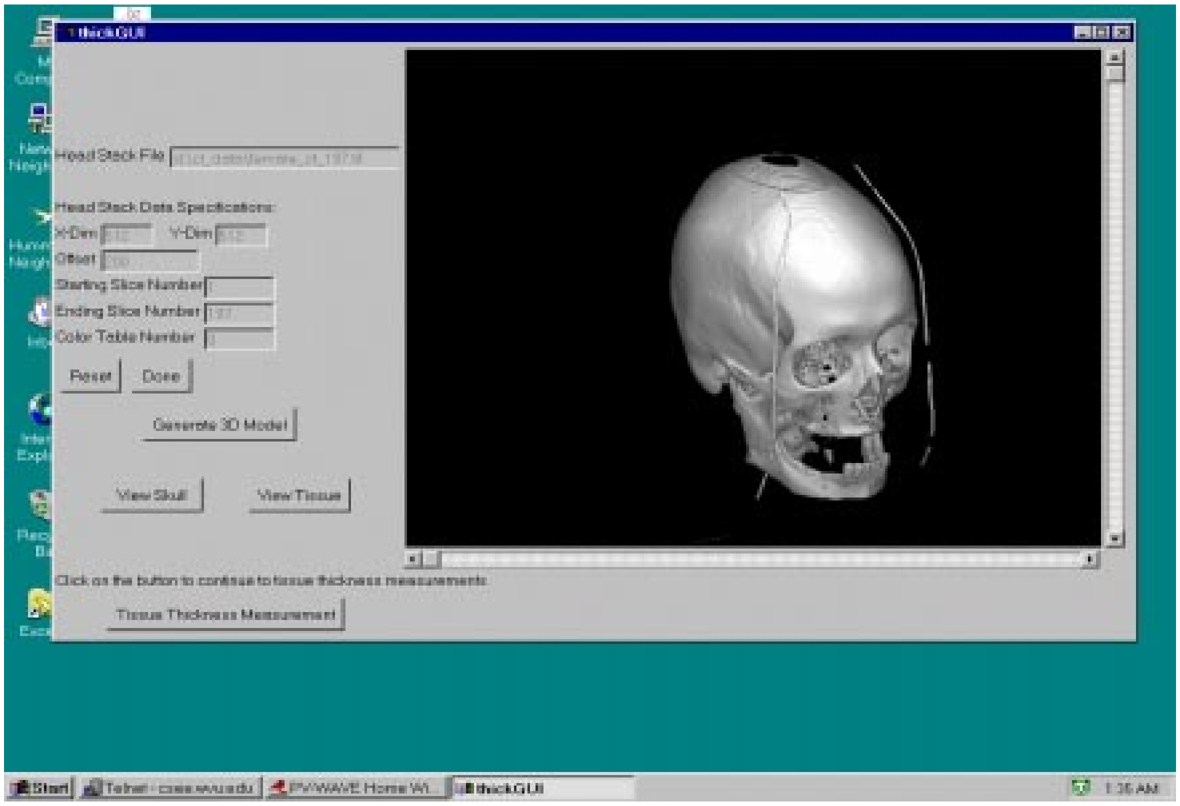
Figure 41: Data formatting using Scion Image.

6.2 Tissue Thickness Measurement Tool User Interface

The tissue thickness measurement tools user interface contains two screens. Screen 1performs 3D reconstruction’s described in chapter 4. Skin and the bone 3D models are displayed in the presentation window. Screen 2 is used to perform tissue thickness measurements on the CT data.

The two screens are shown below.

Screen 1:



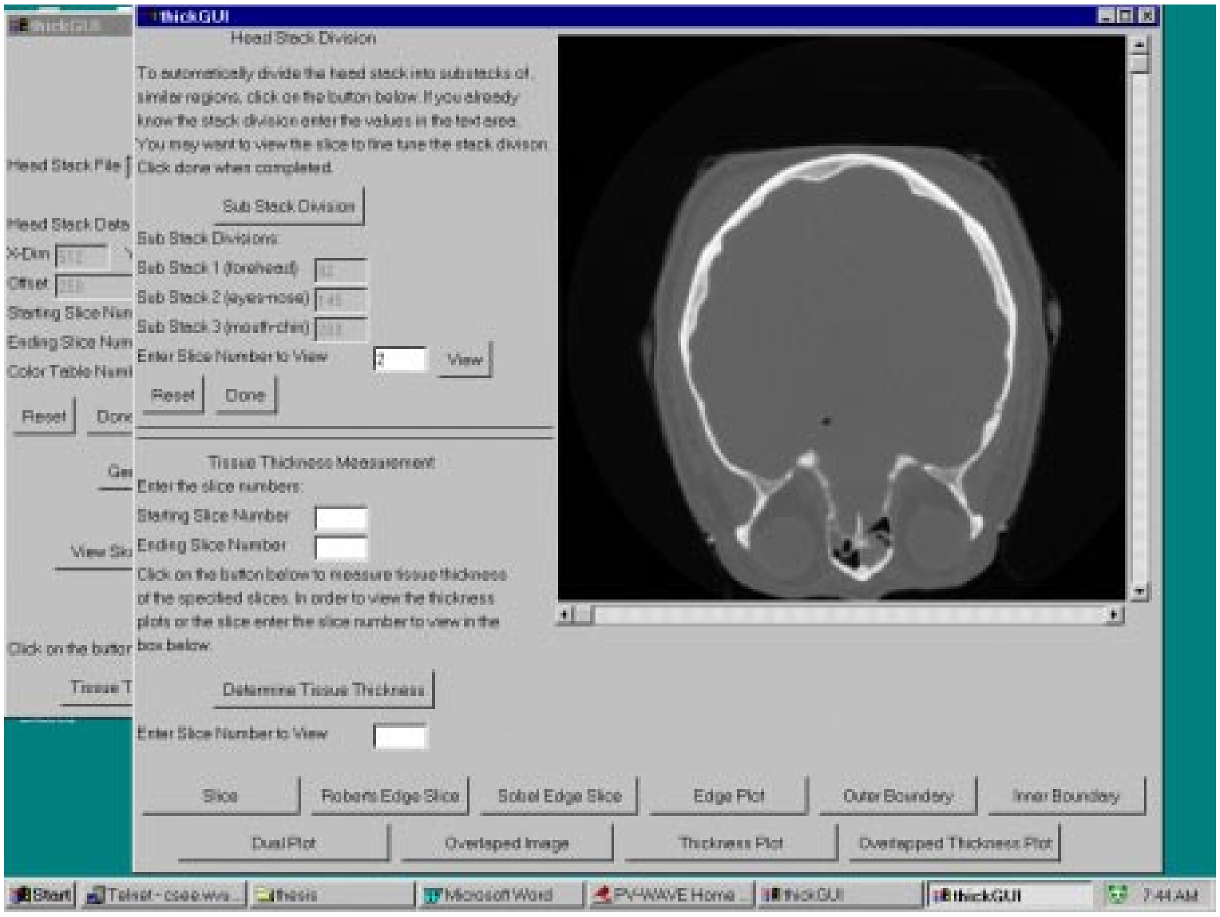
Click on the button to continue to tissue thickness measurements

Figure 42: Tissue thickness measurement tool user interface screen 1.

Head Stack File	: Path of the head stack
X-Dim	: X Dimension of the slice
Y-Dim	: Y Dimension of the slice
Offset	: Header size of the stack to be excluded

Starting Slice Number	: First slice number of the stack to be read
Ending Slice Number	: Last slice number of the stack to be read
Color Table Number	: Color settings for the reconstruction (default 8)
Reset	: Resets the input data
Done	: Reads the input data specified in the text boxes
Generate 3D model	: Generates the 3D models for the
View Skull / View Skin	: Displays the skull and skin model in the window
Tissue Thickness Measure	: Moves to the tissue measurement screen (screen 2)

Screen 2:



To automatically divide the head stack into substacks of similar regions, click on the button below. If you already know the stack division enter the values in the text area. You may want to view the slice to fine tune the stack divison. Click done when completed.

Click on the button below to measure tissue thickness of the specified slices. In order to view the thickness plots or the slice enter the slice number to view in the box below.

Figure 43: Tissue thickness measurement tool user interface screen 2.

Sub Stack Division	: Divides the head stack into sub stacks (chapt 5)
Sub Stack 1 (forehead)	: Gives the starting slice number for sub-stack 1
Sub Stack 2 (eyes-nose)	: Gives the starting slice number for sub-stack 2
Sub Stack 3 (mouth-chin)	: Gives the starting slice number for sub-stack 3
View	: Displays the specified slice in the window
Reset/ Done	: Resets or Reads the data in text areas
Starting Slice Number	: First slice number of the head stack to be read
Ending Slice Number	: Last slice number of the head stack to be read
Determine tissue thickness	: Measures the tissue thickness
Slice Number to View	: Slice number to view in the window
Slice	: Displays the slice
Roberts Edge Slice	: Displays the Roberts edge detector applied slice
Sobels Edge Slice	: Displays the Sobel edge detector applied slice
Edge Plot	: Displays the edge plot of the slice
Outer Boundary	: Displays the outer boundary of the slice
Inner Boundary	: Displays the inner boundary of the slice
Dual Plot	: Displays the inner and outer boundary plots
Overlapped Image	: Displays the dual plot overlapped with the image
Thickness Plot	: Displays the thickness plot
Overlapped Thickness Plot	: Displays the thickness plot overlapped with image

6.3 Working With the Tool

The CT slices are formatted and a head stack is generated using Scion Image as described in section 6.2. After the head stack is obtained the steps are performed to obtain the tissue thickness:

Step 1: Enter the head stack location in the *Head Stack File* text area. Enter the dimensions of the slice and offset values in the corresponding text entry areas. Click *Done* when complete.

Step 2: Generate the 3D model of the skin and the skull by clicking on the *Generate 3D model* button. The models can be view in the view window by clicking on *View Skull/ View Skin* buttons.

Step 3: Click *Measure Thickness* button to continue to the next screen. The second screen is used to measure the tissue thickness.

Step 4: Generate sub-stacks by clicking on *Sub Stack Division* button. If you already know the slice number to divide the stack, the slice numbers can be entered in the *Sub Stack* text areas. Click *Done* when complete.

Step 5: Enter the slice numbers for which you want tissue thickness in the *Starting Slice* and *Ending Slice* text areas. Click on *Determine Tissue Thickness* to generate the thickness files in the *C:\thickness\ directory.*

Step 6: In order to view the thickness plot for a particular slice, enter the slice number to be displayed in the *Enter Slice Number to View* text area. Different plots of the slice which include edge plot, outer boundary, inner boundary, overlapped plots, thickness plots etc. can be viewed by clicking on respective buttons .

The code for the Tissue Measurement Tool is given in appendix B. The next chapter summarizes the results from this thesis and discusses the future work in the project.

CHAPTER 7

Conclusions and Future Work

This chapter summarizes the work done for the thesis and gives the pointers for future work.

7.1 Summary

This work explores the utilization of visualization and image processing technologies for craniofacial reconstruction. It can been seen that computerized tools for craniofacial reconstruction can save a lot of time, effort and the need for a modeler and increasing the chances for identification. This thesis lays the groundwork for using computers to simulate the work of a craniofacial modeler.

Initial work on the project was carried out at the Institute of Orthopedic Research and Education, Baylor College of Medicine, Houston, TX. An approach to craniofacial reconstruction and identification was formulated from a study of existing techniques and requirements for such a system. Three phases have been identified to realize the project described in chapter 2. The first phase of the project is to generate a tissue thickness database, the second phase uses this database with a facial component database to generate a generic face mask over the skull. In the third phase this image is to be identified from a database of facial images.

The first phase of the project is to generate a database of tissue thickness, which is the main focus of this thesis.

3D reconstruction from CT data was carried out and a study was done to determine the optimal CT spacing for reconstruction and accurate tissue thickness measurement. After the initial study the focus was shifted to tissue thickness measurement. A single algorithm cannot be applied on the slices of the head stack for tissue thickness determination. This is because the boundary structures are varied over the head stack. Hence an automated head stack division algorithm is developed that divides the head stack into sub stacks with slices having similar boundary structures. Similar algorithms can be applied on the sub stacks. Three sub stacks are identified. Algorithms for tissue thickness measurement are developed for each sub-stack. Tissue thickness is the distance between the outer layer of the bone and the outer layer of the skin. In the first sub stack the outer contours of the skin and the bone is identified using edge detection techniques and the distance between points is measured. In the second and the third sub-stacks the nose, eyes and mouth features are isolated from curvature studies on the outer and inner contours and excluding regions where there is a sudden change in the curvature. All the above algorithms are integrated with a GUI to build the "Tissue Thickness Measurement Tool".

7.2 Further Work

There are many open issues that need further thorough investigation. I would like to emphasize that this research is still in its rudimentary stage and this work provides the framework for the project. A few of the issues are discussed below.

7.2.1 Tissue Thickness Database

This research has resulted in the development of a tool for tissue thickness measurement. The purpose of this tool is to develop a database of tissue thickness. Phase 1 of the project is still incomplete without a tissue thickness database. Tissue thickness from a large number of CT data from different people needs to be measured and inserted into the database. The main problem will be to obtain a statistically significant set of scans. These measurements need to be organized and classified based on age groups, gender, race etc. Statistical analysis of the data needs to be done so that a template for tissue thickness is obtained for a particular age, sex, color and race. The accuracy of reconstruction and success rate for identification will depend on the database set.

7.2.2 Work on the Remaining Phases

Work on phase two and phase three can be simultaneously started as the database is being developed. Before the database is generated a format is to be decided on to store the tissue thickness data. This is so that they can be easily retrieved and directly used for mask generation without many manipulations to the data set. The data format would depend on the reconstruction algorithm to be used. In the second phase instead of using the regular polygon fitting algorithms b-spline or bezier surfaces are to be used [23]. They need 16 control points for surface generation (routines for surface generation are already written appendix A). Therefore the data format needs to be suitable for such surface generation. Once an algorithm and a data format is determined the facemask can be generated. Another major aspect is creating 3D models of nose and eyes which, are to be placed on the mask. 3D models for nose and eyes already exist which can be used. Phase three of the project deals with using face identification techniques for identification. This field is already well developed and references are provided in the reference list that point to sources for face identification techniques.

References

1. Helmer R. P. and Iscan M.Y. (eds.). Forensic Analysis of the Skull. Willy-Liss, NewYork, 1993.

2. Archer, Katrinaa, Craniofacial Reconstruction Using Hierarchical B-Spline interpolation. M.A.Sc thesis, University of British Columbia, 1997.

3. Vanezis P., Blowes R.W., Linney A.D., et.al. Application of 3-D computer graphics for facial reconstruction and comparison with sculpting techniques. Forensic Sciences International (42), 1989, pp. 69-84.

4. Shahrom A.W, Vanezis P., Chapman P., Gonzalez R.C., et.al. Techniques in facial identification: Computer aided facial reconstruction using a laser scanner and video superimposition.

5. Miyasaka S., Yoshino M., Seta S. The computer aided facial reconstruction system. Forensic Scinences International, 1995, pp. 155-165.

6. Kevin M.C. Simulating Craniofacial Growth, MSc Thesis, University of British Columbia, 1997.

7. Michael W. V., Pilgrim T., Bhatia G., et.al. Facial Surface Scanner. IEEE Computer Graphics and Applications, 1991, pp. 72-80.

8. Foley, Van D., Feiner et.al. Computer Graphics, Principles and Practice. Addison Wesley Publication, 1997, pp. 516- 529.

9. Turk M.and Pentland A. Eigenfaces for recognition. Journal of Cognitive Neuroscience, Vol 3, No.1, 1991, pp. 71-86.

10. Pentland A., Moghaddam B. and Starner T. View based Modular eigenspaces for face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994, pp. 84-91.

11. Schroeder W., Martin K., Lorensen B. The Visualization Toolkit, Prentice Hall Publication, 1997.

12. Lorensen W.E., Cline H.E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, Vol 21(3) , 1987, pp. 163-169.

13. Gonzalez C.R., Woods R. E., Digital Image Processing, Addison Wesley Publications, 1992.

14. Sonka M., Hlavac V., Boyle R., Image Processing, Analysis, and Machine Vision, Pws Publishing, 1998.

15. Canny J. F, A computational approach to edge detection. IEEE Transactions on Pattern Analysis and machine Intelligence, Vol 8 (6), 1986, pp. 679-698.

16. Visible Human Project. http://www.nlm.nih.gov/research/visible/visible_human.html

17. Scion Image. www.scioncorp.com

18. Visualization Tool Kit. www.kitware.com

19. PV Wave. www.vni.com

20. Claudio L., Augusto M., et.al. Automatic detection of craniometric points for craniofacial identification. SIBGRAPI, 1996, pp. 189-196.

21. Fedosyutkin B.A. Nainys J.V. The relationship of skull morphology to facial features. Forensic Analysis of the Skull. (R.P.Helmer and M.Y.Iscan eds.), Willy-Liss, NewYork, 1993, pp. 199 – 213.

22. Spencer L. R. The Human Skull. Its Mechanics, Measurements and Variations. Charles C. Thomas publisher, 1984.

23. Samual H. R., Markus H.R., et.al. A Bernstein-Bezier based approach to soft tissue simulation. CS Technical report#282, Computer graphics Research Group, ETH Zurich, 1998.

24. Young H., Niels V., Age Classification from facial images. IEEE Pattern Recognition 1994, pp.762-767.

25. Kas, Witkin, Terzopolous. “Snakes: Active Contour Models” International Journal of Cognitive Vision, 1987, pp. 321-331.

26. William, Shah M. “ A Fast Algorithm for Active Contour and Curvature Estimation”, 1991, CVGIP, Vol 55, pp. 14-26.

27. Xu, Prince. “Snakes, Shapes and Gradient Vector Flow”. IEEE Transactions on Image Processing 1998, March, Vol7, No. 3, pp. 359-369

Appendix A : Surface Generation Procedure

Bezier Surface

A surface defined by mathematical formulae, used in computer graphics. A surface P(u, v), where u and v vary orthogonally from 0 to 1 from one edge of the surface to the other, is defined by a set of (n+1)*(m+1) "control points" (X(i, j), Y(i, j), Z(i, j)) for i = 0 to n, j = 0 to m. Bezier surfaces are an extension of the idea of Bezier curves, and share many of their properties.

Mathematical Representation

These curves can be represented mathematically by the following equations.

X(s, t) = S Mb Px Mb^t T^t
Y(s, t) = S Mb Px Mb^t T^t
Z(s, t) = S Mb Px Mb^t T^t

Applications

Bezier surfaces are used to fit in surface through a set of point's data. This is especially useful in modeling soft tissue features of the face. 16 control points define a Bezier surface. Varying the control points can easily vary the concavity and shape of the surface. The following is a routine written in P-V Wave that can fit in a surface through a set of 16-control point's data. The control points are read in through the points.dat file.

Code:

```
;        cd, 'D:\0_programs'
;        .run bezier_surface.pro
;        bezier_surface

; Program BEZIER_SURFACE
; this program fits  a surface through a set of 16 control point data
; the data points are read from the file in the file points1.txt

PRO BEZIER_SURFACE

;no of points in the s and t regions
no_points_s =10
no_points_t =10

;read the control points points in the space
cd, 'd:\0_Programs'
;string to store the name of files of slice data to be read in
in_file_list = 'points1.txt'

;read the x,y,z column data
status= DC_READ_FREE(in_file_list, X, Y, Z, /Column)

Mb= [[-1, 3, -3, 1], [3, -6, 3, 0], [-3, 3, 0, 0], [1,0,0,0]]
Mb_t = TRANSPOSE(Mb)


 ;Generate the P matrix
Px = FLTARR(4,4)
Py = FLTARR(4,4)
Pz = FLTARR(4,4)

          k = 0
 FOR i=0, 3 DO BEGIN
         FOR j=0, 3 DO BEGIN
                  Px(i, j) = X(k)
                  Py(i, j) = Y(k)
                  Pz(i, j) = Z(k)
                  k = k+1
         ENDFOR
 ENDFOR

;create the X1,Y1 and Z1 form the matrices
X1 = FLTARR(no_points_s, no_points_t)
```

```
Y1 = FLTARR(no_points_s, no_points_t)
Z1 = FLTARR(no_points_s, no_points_t)


;defining the ranges of s and t
s= INDGEN(no_points_s) /10.0
t= INDGEN(no_points_t) /10.0

FOR i=0, no_points_s -1 DO BEGIN
        FOR j=0, no_points_t-1 DO BEGIN
                S_mat= [(s(i))^3, (s(i))^2, (s(i)), 1]
                T_mat= [(t(j))^3, (t(j))^2, (t(j)), 1]
                T_t = TRANSPOSE(T)
                X1(i, j) = S_mat # Mb # Px # Mb_t # T_mat
                Y1(i, j) = S_mat # Mb # Py # Mb_t # T_mat
                Z1(i, j) = S_mat # Mb # Pz # Mb_t # T_mat
        ENDFOR
ENDFOR

SHADE_SURF_IRR, Z1, X1, Y1
END
```

Test Case:

x	y	z
2	1	2
2	3	2
2	5	1
2	7	1
4	1	2
4	3	3
4	3	1
4	7	0
6	1	0
6	3	1
6	5	0
6	7	3
8	1	0
8	3	0
8	5	0
8	7	1

Table 1:Bezier Surface Control Points

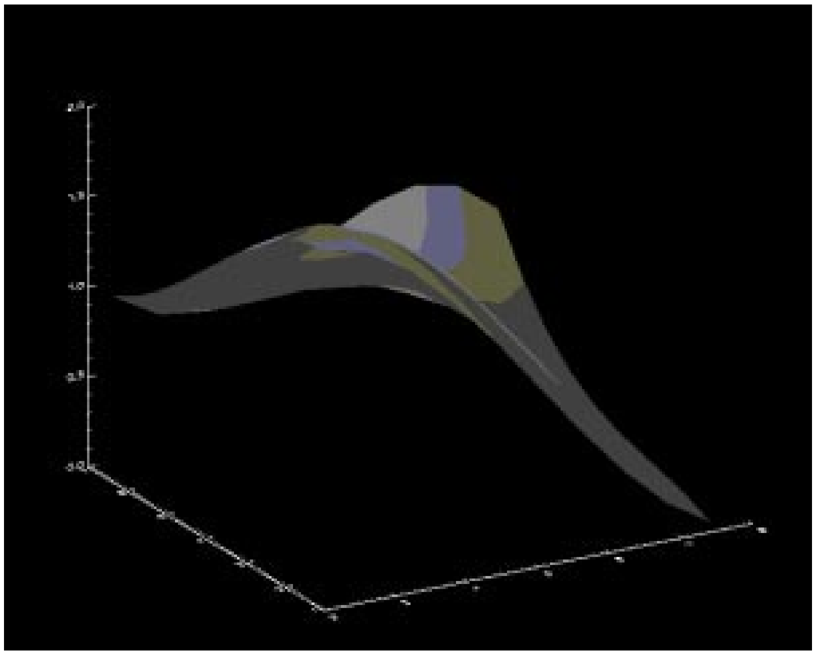


Fig 1: Bezier Surface

- *B-Spline Surface*

The B-Spline surface does not pass through any control point but is continuous and has continuity of tangent vector and of curvature while the Hermite and Bezier surfaces have only first derivative continuity.

Mathematical Representation:

These curves can be represented mathematically by the following equations.

x(s, t) = S Ms Px Ms^t T^t
y(s, t) = S Ms Py Ms^t T^t
z(s, t) = S Ms Pz Ms^t T^t

Applications:

These surfaces are used to fit in an approximate surface through a set of control points. These surfaces are smoother and . However since the surface does not pass through the control points it is difficult to visualize the exact surface by varying the control points.

Code: Bspline_Surface.pro

```
; Program BSPLINE_SURFACE
; this program fits  a surface through a set of 16 control point data
; the data points are read from the file in the file points1.txt

PRO BSPLINE_SURFACE

;no of points in the s and t regions
no_points_s =10
no_points_t =10

;read the control points points in the space
cd, ’d:\0_Programs\’
;string to store the name of files of slice data to be read in
in_file_list = ’points1.txt’

;read the x,y,z column data
status= DC_READ_FREE(in_file_list, X, Y, Z, /Column)
```

```
Ms= [[-1.00, 3.00, -3.00, 1.00], [3.00, -6.00, 0.00, 4.00], [-3.00, 3.00, 3.00, 1.00],
[1.00,0.00,0.00,0.00]]
Ms= Ms/6
Ms_t = TRANSPOSE(Ms)

 ;Generate the P matrix

Px = FLTARR(4,4)
Py = FLTARR(4,4)
Pz = FLTARR(4,4)

           k = 0
 FOR i=0, 3 DO BEGIN
          FOR j=0, 3 DO BEGIN
                    Px(i, j) = X(k)
                    Py(i, j) = Y(k)
                    Pz(i, j) = Z(k)
                    k = k+1
          ENDFOR
 ENDFOR
;create the X1,Y1 and Z1 form the matrices

X1 = FLTARR(no_points_s, no_points_t)
Y1 = FLTARR(no_points_s, no_points_t)
Z1 = FLTARR(no_points_s, no_points_t)

;defining the ranges of s and t
s= INDGEN(no_points_s) /10.0
t= INDGEN(no_points_t) /10.0

FOR i=0, no_points_s -1 DO BEGIN
          FOR j=0, no_points_t-1 DO BEGIN
                    S_mat= [(s(i))^3, (s(i))^2, (s(i)), 1]
                    T_mat= [(t(j))^3, (t(j))^2, (t(j)), 1]
                    T_t = TRANSPOSE(T)
                    X1(i, j) = S_mat # Ms # Px # Ms_t # T_mat
                    Y1(i, j) = S_mat # Ms # Py # Ms_t # T_mat
                    Z1(i, j) = S_mat # Ms # Pz # Ms_t # T_mat
          ENDFOR
ENDFOR

SHADE_SURF_IRR, Z1, X1, Y1
END
```

Test case:

x	y	z
2	1	2
2	3	2
2	5	1
2	7	1
4	1	2
4	3	3
4	3	1
4	7	0
6	1	0
6	3	1
6	5	0
6	7	3
8	1	0
8	3	0
8	5	0
8	7	1

Table 2:B-Spline Surface Control Points

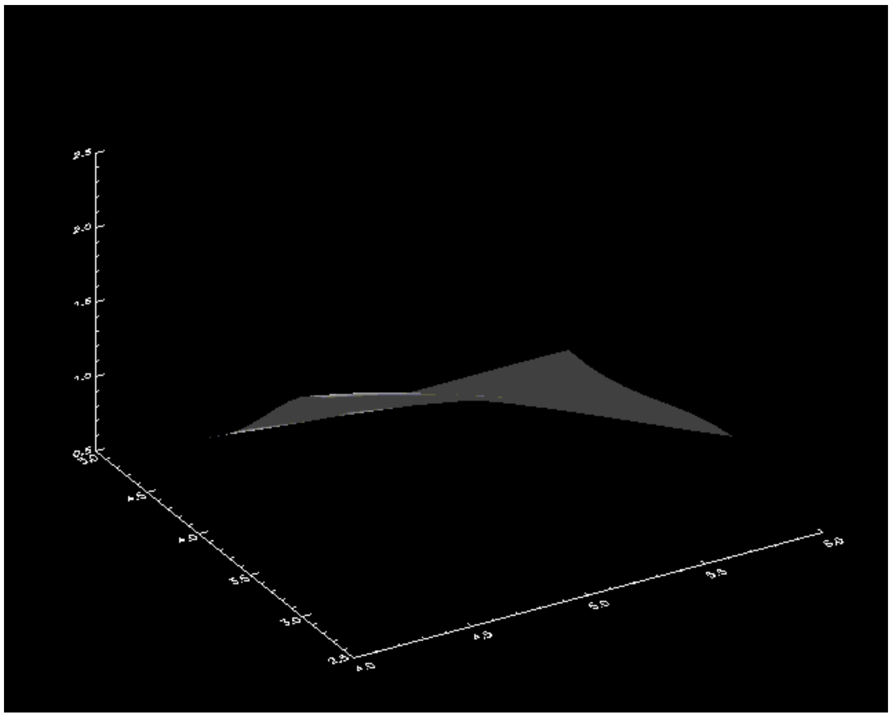


Fig 2: B-Spline Surface

- ***Hermite Surface***

The Hermite surface is the basic surface from which the Bezier and the B-Spline are derived. However its application for surface fitting is limited as this surface is generated based on the position and tangents at the curve's end points. This is difficult to determine in case of surfaces where most of the time the data is in the form of x, y, z points.

Mathematical Representation:

x(s,t) = S Mh Qx Mh^t T^t
y(s,t) = S Mh Qy Mh^t T^t
z(s,t) = S Mh Qz Mh^t T^t

Application:

Places where the tangency data is available this type of surface fitting can be used.

Code:

```
;	cd, ’D:\0_Programs\’
;	.run hermite_surface.pro
;	hermite_surface


; Program HERMITE_SURFACE
; this program fits  a surface through a set of 16 control point data
; the data points are read from the file in the file points1.txt

PRO HERMITE_SURFACE

;no of points in the s and t regions
no_points_s =10
no_points_t =10

;read the control points points in the space
cd, ’d:\0_Programs\’
;string to store the name of files of slice data to be read in
in_file_list = ’points1.txt’

;read the x,y,z column data
status= DC_READ_FREE(in_file_list, X, Y, Z, /Column)

Mh= [[2, -3, 0, 1], [-2, 3, 0, 0], [1, -2, 1, 0], [1, -1, 0, 0]]
Mh_t = TRANSPOSE(Mh)


 ;Generate the P matrix
Px = FLTARR(4,4)
Py = FLTARR(4,4)
Pz = FLTARR(4,4)

	 k = 0
 FOR i=0, 3 DO BEGIN
	FOR j=0, 3 DO BEGIN
		Px(i, j) = X(k)
		Py(i, j) = Y(k)
		Pz(i, j) = Z(k)
		k = k+1
```

```
          ENDFOR
 ENDFOR

;create the X1,Y1 and Z1 form the matrices
X1 = FLTARR(no_points_s, no_points_t)
Y1 = FLTARR(no_points_s, no_points_t)
Z1 = FLTARR(no_points_s, no_points_t)

;defining the ranges of s and t
s= INDGEN(no_points_s) /10.0
t= INDGEN(no_points_t) /10.0

FOR i=0, no_points_s -1 DO BEGIN
          FOR j=0, no_points_t-1 DO BEGIN
                    S_mat= [(s(i))^3, (s(i))^2, (s(i)), 1]
                    T_mat= [(t(j))^3, (t(j))^2, (t(j)), 1]
                    T_t = TRANSPOSE(T)
                    X1(i, j) = S_mat # Mh # Px # Mh_t # T_mat
                    Y1(i, j) = S_mat # Mh # Py # Mh_t # T_mat
                    Z1(i, j) = S_mat # Mh # Pz # Mh_t # T_mat
          ENDFOR
ENDFOR
SHADE_SURF_IRR, Z1, X1, Y1
END
```

Test case:

x	y	z
2	1	2
2	3	2
2	5	1
2	7	1
4	1	2
4	3	3
4	3	1
4	7	0
6	1	0
6	3	1
6	5	0
6	7	3
8	1	0
8	3	0
8	5	0
8	7	1

Table 3: Hermite Surface Control Points

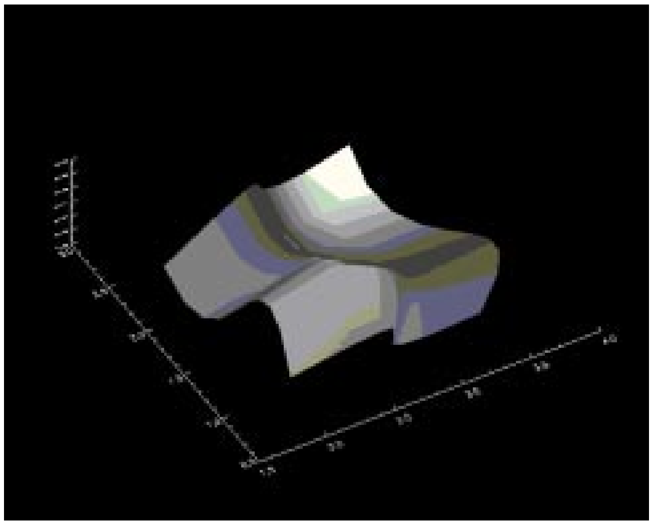


Fig 3: Hermite Surface