Design and Analysis of a System for 3D Fabrication of Synthetic Anatomical Structures

By

Raymond D. Nell

A DISSERTATION PRESENTED TO THE HIGHER DEGREES COMMITTEE OF CAPE PENINSULA UNIVERSITY OF TECHNOLOGY IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER TECHNOLOGY (ELECTRICAL ENGINEERING)

2005

DECLARATION

I, Raymond D. Nell, herby declare that this dissertation represents my own work and the opinions are my own and not necessary those of the university. All references used have been accurately reported.

Signed: .... Name: Raymond D. Nell

SUMMERY

This dissertation is the reading and display of DICOM medical images (Digital Imaging and Communication in Medicine) and production of model artifacts of anatomical organs using Rapid Prototyping

An algorithm to read these DICOM medical images was developed. It also displays pixel information of the image. When the DICOM image has been read and displayed, the information required to produce the anatomical artifact is extracted. These 2D slice images, MRI (Magnetic Resonance Imaging) and CT Scan (Computer Tomography) images are written to 3D file in SLC (Slice files) and STL (Stereolithography File Format) format. A 3D softcopy of the anatomical structure is created. At this stage, the clinician or surgeon can make any changes or require additional information to be added to the anatomical structure.

With the 3D model available in STL format, a physical artifact is produced using Rapid Prototyping. The external edge of the anatomical structure can be produced using Rapid Prototyping as well as the outer rim with the internal structures.

To produce the external surface of the structure, an outer rim edge detection algorithm has been developed. This will only extract the external surface of the structure. In addition to the softcopy of the structure, multiple organs can be displayed on the same image and this will give a representation of the interaction of neighboring organs and structures. This is useful as both the normal anatomy as well as the infiltration of the abnormal pathology can be viewed simultaneously.

One of the major limitations of displaying the information in a 3D image is that the files are very large. Since 3D STL files use triangles to display the outer surface of a structure, a method to reduce the file size and still keep the image information was developed. The triangle reduction method is a method to display the 3D information and to decrease the STL file size depending on the complexity of the outer surface of the structure.

To ensure that the anatomical model s represented as in the DICOM files, an Interpolation Algorithm was developed to reconstruct the outer of the model from 2D MRI or CT-Scan images.

A word about computer models: Some of the programs and presentations are based on the real world. They model the real world and anatomical structures. It is very important to note that the models are created with software. Obviously a model is useful if it resembles reality closely, but it is only a prediction about the model itself. Models are useful because they help to explain why certain things happen and how interaction takes place. Models provide suggestions for how structures might look. Computer models provide answers very quickly. These are computer models representing the real structure. (Czes Kosniowski, 1983)

Chapter overview

Chapter 1: Introduction, project objective and the outcome of the research.
This chapter gives a background on medical imaging, X-rays, ultrasound, MRI and CT-Scans. The objective of the research and the expected results is explained.

The objective is to design a system for reading DICOM medical files and produce a physical artifact of an organ. A physical model of one slice of an MRI scan of a human scull was developed by creating an STL file from a DICOM image, filtering out the tissue and using the STL file of the bone for Rapid Prototyping.

Medical diagnoses are essentially the extraction of anatomical and physiological information from a subject (the patient) and the interpretation of this information in such a way for corrective treatment. Diagnostic radiology provides one method by which the information can be obtained. The radiological image presents the information in a visual form, where information of the anatomical structure can be extracted.

Chapter 2: Imaging and mathematical background.
How the eye view an image. This chapter gives an explanation about the difference between a real image and a mental image and the different methods of viewing an image.

Chapter 3: Description of the DICOM format. X-ray, MRI and CT-Scan image examples.
This chapter explaines the principles of X-ray imaging and the layout of a DICOM file.

The DICOM standard defines a set of protocols to be followed by devices claiming DICOM conformance. DICOM files are image files. DICOM files have a fixed layout that is identical in all files, consisting of a header block with pointers to information in the file and tags that indicate the beginning of a set of information in the file.

DICOM files are medical image files that contain the following information: patient identification, patient positioning during examination and the image information of the examination performed on the patient. This can be an X-RAY examination, CT SCAN examination or MRI examination on the patient.

Chapter 4: Rapid Prototyping. Describing the STL and SLC file format and an explanation of Rapid Prototyping.

Rapid Prototyping is the speedy fabrication of sample parts for demonstration, evaluation, or testing. It typically utilizes advanced layer manufacturing technologies that can quickly generate complex three-dimensional objects directly from computer-based models devised by Computer Aided Design (CAD).

Chapter 5: Manipulation of the medical image.
This chapter gives an overview of the algorithm on reading DICOM images and filtering out the area of interest. To convert the medical images like the CT-scan slice or MRI slice to Rapid prototyping, the medical DICOM image need to be displayed and the information of interest need to be extracted.

Charter 6: Converting Medical DICOM images to Rapid prototyping.
This chapter gives an explanation on the techniques used to read and display medical images. The medical image in an image window is described by a specific height and width and with each pixel information describes a specific RGB (RED, GREEN and BLUE) value. This information is obtained from the DICOM tags, which describe the height and width. The image is completed with a set of pixels of which the position is obtained from the DICOM tag (7FE0, 0010).

Chapter 7: Improvements on Rapid Prototyping files
This chapter explains multi-slice STL files and the triangle reduction method. The outer rim edge detection is explained and an example of a multi-slice 3D image is given.

The medical examination in the form of CT scan or MRI scan is done in a step-by-step manner. All these slices are evaluated separately as describe by the methods above and then added together to form the major structure.

Chapter 8: Theoretical method for detecting the 3D point in space with one X-ray exposure.

This chapter gives a new method and developed to detect the position of an artifact with only one X-ray exposure.

To detect the position of an artifact in space with an X-ray, normally two exposures are done. In these cases two exposures are done by moving the X-ray tube or we move a patient to obtain a 90-degree view from the previous one. In the movement of the patient and the X-ray tube, the position of the artifact from the Anterior Posterior (AP) to Lateral (Lat) position is obtained. This will expose the radiographer and patient to radiation and to limit the amount of radiation for detection of an artifact, the method to detect the position in space with only one X-ray exposure was developed.

Chapter 9: Future work improvements. Organ interaction.

Additional improvements can be added to deliver a much better 3D product. This method delivers the 3D model in color. One of the improvements is minimize the time from reading in the DICOM image to produce a 3D model.

Communication improvements. This chapter gives a method developed to improve commutation between medical officers by viewing the same image at two stations simultaneously.

After the image has been reconstructed, the results and presentation of the image need to be discussed and displayed to other users.

This chapter explains the remote image viewing to display medical images simultaneously on two or more PC's.

Chapter 10: Conclusion: Visualization of arterial structures

The research has developed a method for reading of DICOM medical files and displays the information in an image window. The algorithm developed in the work done can display these DICOM files that can be a CT Scan or an MRI image

After the DICOM image has been read and displayed, the anatomical structures of interest has been extracted using the filtering method where the gray scale value of a specific area or organ is inserted and the information is extracted using the filtering method.

Arteries of the heart and the brain can be damaged, dilated or present abnormally especially in a stroke or a heart attack. Visualization of these complex arteries can be done using the method described above. For this the arteries need to be identified and tracking of those arteries can be done through the slices. Not just only small structures like the coronary arteries in the heart can be isolated, but also major arteries like the Aorta can also be visualized.

Chapter 11: Operators Manuel of the Software Code developed
This chapter gives a brief overview of the operation of the software code developed.

Chapter 12: Publication
An article submitted for publication.
Theoretical method for detecting the 3D point in space with one X-ray exposure

Chapter 13: Software code
This chapter gives the source code of the software developed to read and display medical images.

Acknowledgements

I wish to thank the following for their input and support.
Cape Peninsula University of Technology.

Mr. Miles Jacobs: Department of Electrical Engineering
Cape Peninsula University of Technology.

Mr. Aladdin Speelman: School of Radiography
Grootte Schuur Hospital

Prof. Lorna Martin
Head
University of Cape Town (UCT)
Department of Health Sciences
Forensic Medicine & Toxicology

Greg Flash
Research Manager
LODOX: Critical Imaging Technology
South Africa

TABLE OF CONTENTS

List of figures

NOMECLATURE

ARC	: American College of Radiologists
SLC	: Slice files
STL	: Stereolithography File Format
DICOM	: Digital Imaging and Communication in Medicine
MRI	: Magnetic Resonance Imaging
CT Scan	: Computer Tomography
US	: Ultrasound
ROI	: Region of Interest

Glossary

NEMA	: National Electrical Manufacturers Association
LOM	:Laminated object manufacturing
RP	:Rapid Prototyping
3-D	:Three Dimensional Representation
CSIR	:Council for Scientific and Industrial Research
MRI	: Magnetic Resonance Imaging
CT Scan	: Computer Tomography Scan
X-ray	: X-ray Imaging

Keywords: Stereoscope, horopter, pseudoscopic, point focus, finite focus, penumbra, Z co-ordinate

CHAPTER 1

1) Introduction

Design and Analysis of a System for 3D Fabrication of Synthetic Anatomical Structures describes the reconstruction anatomical structures in 3D from DICOM (Digital Imaging and Communication in Medicine) medical files.

Diagnostic imaging as known today, originates back to November 1895 when the German professor WIHELM KONRAD RONTGEN discovered the high energy "rays" which, after having penetrated solid material, still induced a chemical process on a photographic plate.

On the 28 December of the same year he published his discovery in a speech at the University of Wurzburg, Germany, under the title "Uber eine neue Art von Strahlen".

Although modern equipment looks very different from what was used a hundred years ago, the basic physics and principles behind X-ray examinations have not change drastically.

As new and completely different imaging modalities like ultrasound (US) and Magnetic Resonance Imaging (MRI) are coming into use, the need for X-ray based examinations to benefit both the patient and assist the medical practitioner is still necessary.

The need for a 3D image that can be rotated in any geometric position and specific measurements of any anatomical section is required.

With Design and Analysis of a System for 3D Fabrication of Synthetic Anatomical Structures, multiple organs can be viewed on the same 3D image and any pathology can be included in these representations. The alignment, size, shape or manner in which abnormal pathology infiltrates normal anatomical structures can be viewed and any orientations of all the structures are available. The 3D image can also be converted to a Rapid Prototyping artifact to produce a physical model of the structure.
Rapid Prototyping is the physical production of artificial model using computer aided manufacturing.

A physical model of one slice of a MRI scan of a human brain was developed by creating an STL file from a DICOM image, filtering the bone and using the STL file of the scull for Rapid Prototyping.

The objective is to design a system from reading DICOM medical files to produce a physical artifact of human organs.

To obtain these final results multiple steps are investigated and developed from reading in the DICOM file, extracting the information as well creating this physical artifact.

1.1) X-Ray Imaging
X-rays (German: Röntgenstrahlen) are a form of electromagnetic radiation with a wavelength approximately in the range of 5 pm - 10 nanometers (corresponding to frequencies in the range 30 PHz - 60 EHz).

X-rays with a wavelength longer than 0.1 nm are called soft X-rays. At wavelengths shorter than this, they are called hard X-rays. Hard X-rays overlap the range of long-wavelength (low energy) gamma rays, however the distinction between the two terms depends on the source of the radiation, not its wavelength: X-ray photons are generated by energetic electron processes, gamma rays by transitions within atomic nuclei. X-rays are primarily used for diagnostic medical imaging and crystallography.

1.2) Ultrasound
Ultrasound waves can be bounced off of tissues using special devices. The echoes are then converted into a picture called a sonogram. Ultrasound imaging, referred to as ultrasonography, allows physicians and patients to get an inside view of soft tissues and body cavities, without using invasive techniques. Ultrasound is often used to examine a fetus during pregnancy. There is no convincing evidence for any danger from ultrasound during pregnancy.

1.3) Magnetic Resonance Imaging (MRI)
MRI is a procedure in which radio waves and a powerful magnet linked to a computer are used to create detailed pictures of areas inside the body. These pictures can show the difference between normal and diseased tissue. MRI makes better images of organs and soft tissue than other scanning techniques, such as CT or x-ray. MRI is especially useful for imaging the brain, spine, the soft tissue of joints, and the inside of bones. Also called nuclear magnetic resonance imaging.

1.4) Computed Tomography Scan (CT-Scan)
CT Scan is a series of detailed pictures of areas inside the body taken from different angles. A computer linked from an x-ray machine creates these pictures. Also called computerized tomography and computerized axial tomography (CAT).

1.5) Rapid Prototyping (RP)
Rapid Prototyping is the speedy fabrication of sample parts for demonstration, evaluation, or testing. It typically utilizes advanced layer manufacturing technologies that can quickly generate complex three-dimensional objects directly from computer-based models devised by Computer Aided Design (CAD). This computer representation is sliced into two-dimensional layers, whose descriptions are sent to the fabrication equipment to build the part layer by layer. Rapid prototyping includes many different fabrication technologies. Stereolithography (STL), selective laser sintering (SLS), laminated object manufacturing (LOM), and fused deposition modeling (FDM) are a few examples.

1.6) Digital Imaging and Communications in Medicine (DICOM)
Digital Imaging and Communications in Medicine (DICOM) is a comprehensive set of standards for handling, storing and transmitting information in medical imaging. It includes a file format definition and a network communication protocol.

A standard for interconnection of medical digital imaging devices developed by the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA). DICOM improves interconnect ability of equipment on a network and interactivity with other communications standards. (http://www.med.uni-giessen.de/ipl/dicomvl.html)

1.7) Project and Research objectives

•This research is to demonstrate and developed the reading and display of DICOM medical images, like CT Scan an MRI images and reproducing a physical artifact of an anatomical structure using Rapid Prototyping.

•Medical diagnoses are essentially the extraction of anatomical and physiological information from a subject (the patient) and the interpretation of this information in such a way for corrective treatment. Diagnostic radiography provides one method by which the information can be obtained. The radiographic image presents the information in a visual form, where information of the anatomical structure on the radiographic image can be extracted.

•The research will contribute to the description and display of DICOM medical images. Developed an algorithm and writing a program to read the DICOM tags and image data and display the data in an image window.

•Extracting normal anatomy and abnormal pathological structures of the human body from a non-invasive examination of the patient. This will be achieved by extracting the pathological structures and create an extraction algorithm to obtain only the pathological structure of interest.

•Create a 3D model in software where measurements and changes can be made. The pixel information on an MRI slice will be converted to 3D co-ordinates in space.

•Develop an STL file from the structure to create a physical model. STL files are used in Rapid Prototyping. The pixel information and extraction of anatomical structures will be converted to STL format in color to produce a solid STL 3D file for Rapid Prototyping.

•Develop different STL files of different sections of the model when after rapid prototyping these sections can be build up to create the major organ. Create an algorithm and method to add different anatomical organs, example tumor and normal anatomy onto one single 3D representation.

•Creating a 3D model of an anatomical structure in software where measurements and changes can be made. The measurements of the STL model must be the same as that of the information described in the DICOM file.

•Create an STL file for Rapid Prototyping from the structure to create a physical model. Use the laminated object manufacturing (LOM) at the Mechanical Department at Cape Peninsula University of Technology.

•Create different STL files of different sections of the model when after rapid prototyping these sections can be build up to create the major organ. Explain how different organs can be manufactured differently.

•Provide a physical artifact from a DICOM medical image slice. Create an artifact of the human scull.

1.8) The following are additional suggestions for future work:

• Create a new method and algorithm to detect the position of an artifact in space with only one X-ray exposure.

•Improve communication between medical officers and researches by developing the Remote Image view where the image can be displayed on any PC for discussion and changes.

•Create a color model of the anatomical structure. This will be achieved since LOM machine can produce color artifacts.

•Introduce mechanical structures of the anatomical structures where the interaction of the 3D anatomical structure can be seen how it react with other organs. This can be achieved by using PYTHON software where stretching of the anatomical organ can be introduced.

To obtain the final product, certain sub problems need to be identified. The final product has to be broken down in step-by-step way to achieve to results.

The following describe the background information of medical imaging and DICOM files as well as a description on achieving the end results.

1.9) Describe and display DICOM medical images

DICOM is Digital Imaging and Communication in Medicine with the following information present in a file:

- Image data
- Patient identification and technical information about the scanner examination, series, slice and image data.

DICOM files are fixed with a layout identical in all files. The DICOM file consists of a header block with pointers to information in the file and tags that indicate the beginning of a set of information in the file.

1.10) Statement of the problem

The main intention of this project is the conversion of medical diagnostic images like Magnetic Resonance Images (MRI) or Computer Tomography (CT SCAN) images for visualization and fabrication of solid model organs. These solid tissue models can then be fabricated using Rapid Prototyping.

1.11) Awareness of the problem

There is a need for new types of Software to provide 3-dimensional reconstruction of tissue models from MRI, CT SCAN or DICOM Images. This is to allow surgeons to see what they are going to work with. These 3D images are also useful in the case of pathology like fractures or cancer legions to produce a 3-dimensional graphic image to take measurements and to see how they interact with neighboring organs.

1.12) Hypothesis

Extracting normal anatomical and abnormal pathological structures of the human body from a non-invasive examination of the patient.

- Creating a 3D model in software where measurements and changes can be made
- Create an STL file of the structure to produce a physical model.
- Create different STL files of different sections of the model when after rapid prototyping these sections can be build up to create the major organ.

1.13) Obtaining only the structure of interest from the medical image slice

In an MRI or CT scan not only one organ or anatomical part is displayed, but all the relevant organs and structures in that specific anatomical slice. To extract only the anatomical structure of interest, a method is needed to extract and highlight the organs and structures of interest. The extracting algorithm is very important as all the anatomical structure on a specific MRI or CT-scan image is displayed on one slice.

1.14) Creating a 3D model of an anatomical structure in software where measurements and changes can be made

The organ that has been reconstructed in software can be rotated and edited by the user. This 3D software model must be displayed on a PC for discussion and planning.

1.15) Develop an STL file for Rapid Prototyping from the structure to create a physical model.
To produce a physical model using Rapid prototyping, the information of the model has to be converted to STL format. A method for creating an STL file to produce an artifact has to be introduced. Interpolation to recreate the outer surface of the model is also developed.

1.16) Create different STL files of different sections of the model when after rapid prototyping these sections can be build up to produce a model of the major organ.
The anatomical organ consists of a number of anatomical structures and these structures models can be reconstructed separately and then be added together to complete the organ or structure in full.

1.17) Provide a physical artifact from a DICOM medical image slice
The STL file provided must be used to produce the physical model. This STL file must be exported to any Rapid Prototyping machine and the physical artifact must be produced.

CHAPTER 2

2) Background on imaging

Medical diagnoses are essentially the extraction of anatomical and physiological information from a subject (the patient) and the interpretation of this information in such ways that corrective treatment may be prescribe. Diagnostic radiology provides one method by which the information can be obtained. The radiographic image presents the information in a visual form, which is relatively easy for a trained person to understand.

This chapter gives also a background on the mathematics required to alter and change the image data in an image. To change the image data in an image, matrices are used. An image that has been produced, can be changed in the form of stretching and shrinking in any direction by using matrices.

2.1) What is an image?

An image can be a physical presentation, like a photograph or an X-ray image, a painting or a sketch, which has a real physical existence.

An image can also be applied to an idea or a concept, which has a mental rather than a physical existence.

If a person is asked to imagine an object like an apple, a mental image comes to mind. The existence of this visual image helps us to grasp the concept of an apple. Being showed a photograph of an apple has a similar effect.

Of course the photograph of an apple represents only one aspect of an apple: its visual appearance. It provides no representation of its taste, smell or feel of the apple. It is left to our mental processors to imagine these other non-physical characteristics of the apple.

Thus we get the following:

2.1.1) Real image: Those images having a real, physical existence such as a photograph or a radiographic image, like an X-ray CT-Scan and MRI slice. These images are accessible to scientific measurement and objective study.

2.1.2) Mental images: Those images generated as mental pictures within our minds and which are accessible only to subjective study.

2.2) Viewing an image by reflected light

In a real image the surface contains pigments, which reflect varying amounts (and/or colors) of the light incident upon them (fig 2.1). The text and illustration on a printed-paper like a book are viewed by reflected light. Photographic images viewed in this way are known as prints. (Ball et all…,1989)

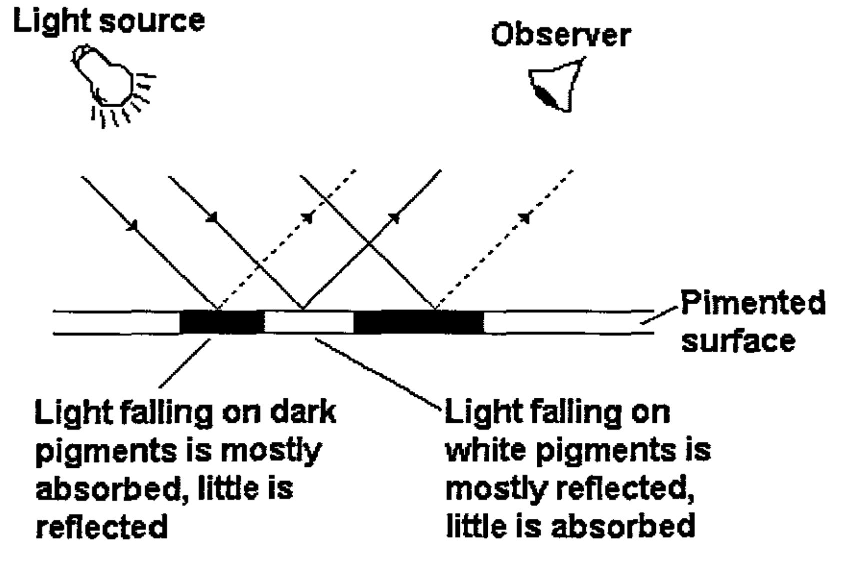


Fig 2.1: Viewing an image by reflected light

2.3) Viewing an image by transparent light

When an image is viewed by transparent light, the image layer contains pigments, which transmits varying amounts (and/or colors) of the light incident upon them (fig 2.2). The image on the conventional radiograph is viewed by transmitted light. In this case the light-absorbent pigment is metallic silver. A photographic image viewed by transparent light is known as a transparency.

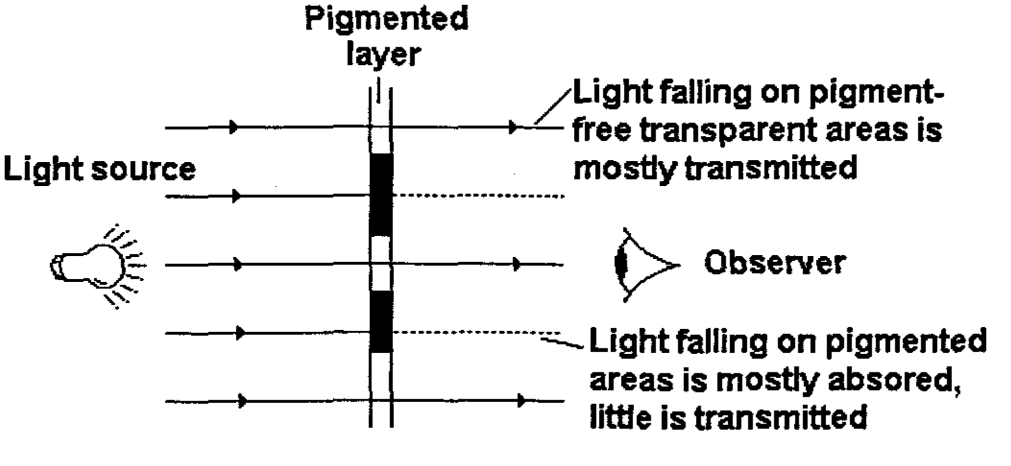


Fig 2.2: Viewing an image by transparent light

2.4) Viewing an image by lamination (on a computer screen)
Exposure to an electron beam stimulates the fluorescent material to emit varying amounts of light. The florescent screen image and the image on a computer screen is viewed by limination.

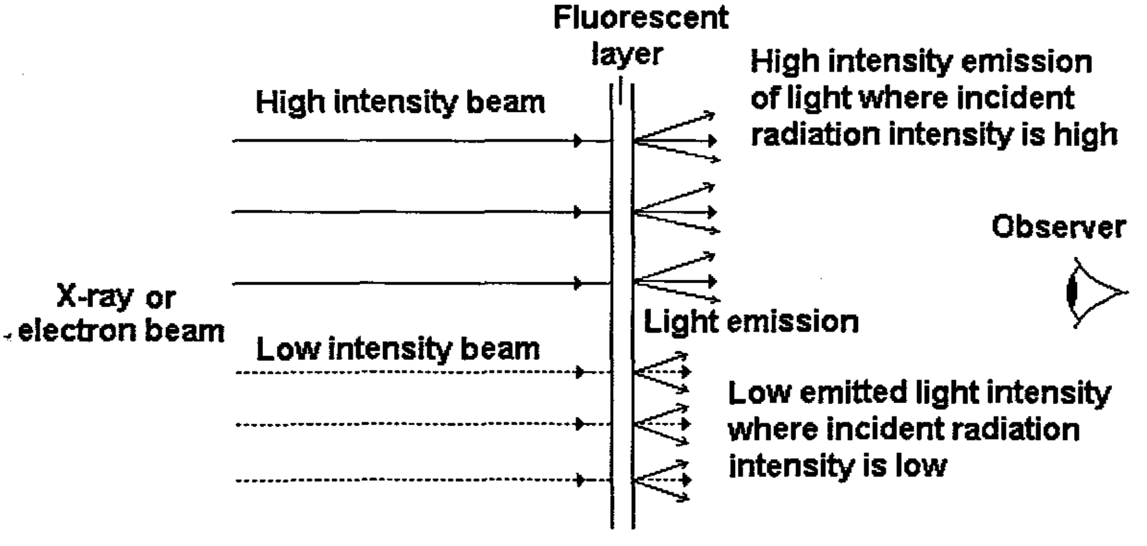


Fig 2.3: Viewing an image by lamination (on a computer screen)

2.5) Mathematical representation for the product of numbers

If a set of numbers are represented by the variable *Xn* then the product *p* representing the product of the numbers are given by:

$$p = \prod_{i=1}^{n} X\,i = X_1 X_2 \cdots X_n$$

where *X1, X2,...,Xn,* are numbers specified as data.

2.6) Mathematical representation for the sum of numbers

The calculation of a sum of numbers are given by the formulae:

$$s = \sum_{i=1}^{n} X\,i = X_1 + X_2 \cdots + X_n$$

Where s is the sum and *Xi, Xn,* are the numbers.

2.7) Matrices

A matrix is a two dimensional array of numeric data, where each row or column consists of one or more numeric values. Arithmetic operations which can be performed with matrices include addition, subtraction, multiplication and division. The size of a matrix is defined in terms of the number of rows and columns. A matrix with M rows and N columns is defined as a MxN matrix. Individual elements of the matrix are referenced using two index values. Using mathematical notation these are usually assigned the variables 'i' and 'j'. The order is row first, column second for example, if a matrix M with order 4x4 exists, then the elements of the matrix are indexed by the following row:column pairs:

```
        | 00 01 02 03 |
    M = | 10 11 12 13 |
        | 20 21 22 23 |
        | 30 31 32 33 |
```

The element at the top right of the matrix has i=0 and j=3.

In computer animation, the most commonly used matrices have either 2, 3 or 4 rows and columns. These are referred to as 2x2, 3x3 and 4x4 matrices respectively. 2x2 matrices are used to perform rotations, shears and other types of image processing. General purpose NxN matrices can be used to perform image processing functions such as convolution. 3x3 matrices are used to perform low-budget 3D animation.

Operations such as rotation and multiplication can be performed using matrix operations, but perspective depth projection is performed using standard optimized into pure divide operations. 4x4 matrices are used to perform high-end 3D animation.

Operations such as multiplication and perspective depth projection can be performed using matrix mathematics.

2.8) Matrix calculations

To review a matrix is a rectangular array of numbers (or functions) enclosed in brackets. These numbers or functions are called entries or elements of the matrix.

For example, in a system of equations in the following:

$$5x - 2y + z = 0$$
$$3x + 0y + 4z = 0$$

Then the coefficients of the unknowns x,y,z are the entries of the coefficients of the matrix A.

$$A = \begin{bmatrix} 5 & -2 & 1 \\ 3 & 0 & 4 \end{bmatrix}$$

The general form of a matrix is given by

$$A = \left[a_{jk}\right]$$

$$A = \left[a_{jk}\right] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Where A is the matrix, *a* the elements of the matrix and it is a, (m x n) matrix where m is the number of rows and n is the number of columns.

2.8.1) Vectors

A vector is a matrix that has only one row. This is called I row vector and a colum vector.

$$a = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

2.8.2) Addition of matrices

Addition is defined only for matrices

$$A = \begin{bmatrix} a_{jk} \end{bmatrix}$$

and

$$B = \begin{bmatrix} b_{jk} \end{bmatrix}$$

of the same size and their sum **A + B** is obtained by adding the corresponding entries. Both the matrices must be of the same size.

2.8.3) Matrix multiplication

The product **C = AB** (in this order) of an (m x n) matrix

$$A = \begin{bmatrix} a_{jk} \end{bmatrix}$$

and an *r x p* matrix

$$B = \begin{bmatrix} b_{jk} \end{bmatrix}$$

is defined if and only if $r = n$, that is the number of rows in the 2nd factor B = number of colums in the 1st factor A, and is then defined as a (m x p) matrix

$$C = \left[c_{jk}\right]$$

with entries

$$c_{jk} = \sum_{l=1}^{n} a_{jl} b_{lk} = a_{j1} b_{1k} + a_{j2} b_{2k} \cdots a_{jn} b_{nk}$$

2.9) Application of Matrix calculations on an image

Matrices and matrix calculations can be used to draw images on the screen and to adjust the image by multiplying a matrics with a scalar. Suppose the matrix represents the following image:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

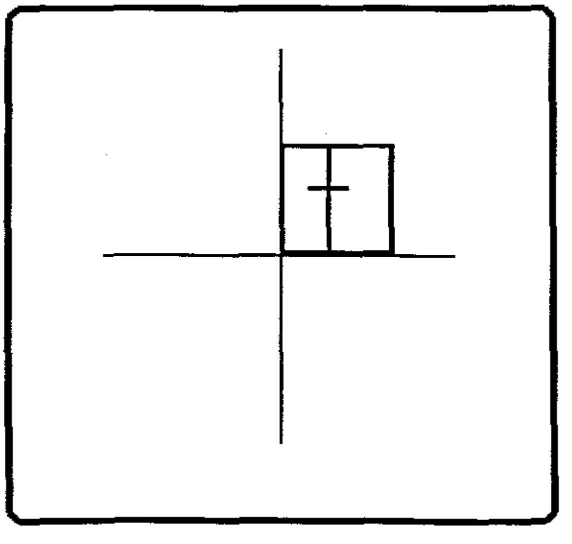


Fig 2.4: Image represented by a matrix

By stretching the image in the X direction then the value of the horizantal or X co-ordinants.If the point was originally plotted at (X,Y) then the new co-ordinants are (2*X,Y). The resulted matrix and image will be:

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

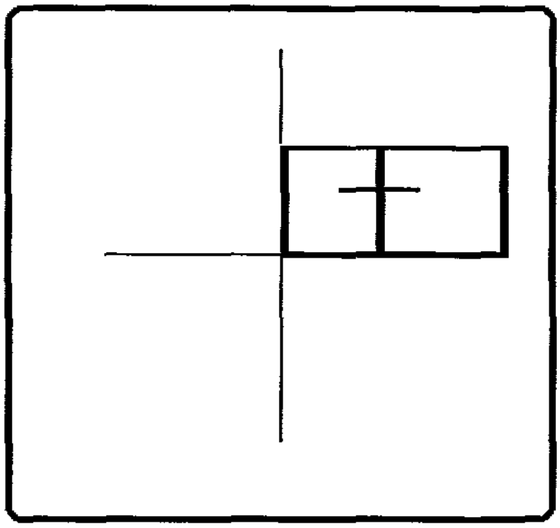
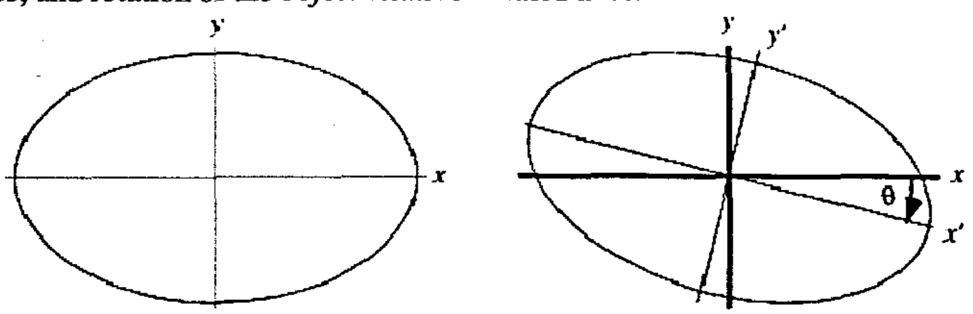


Fig 2.5: Image stretched in the X direction

2.9.1) Rotation Matrices

When discussing a rotation, there are two possible conventions: rotation of the *axes*, and rotation of the *object* relative to fixed axes.



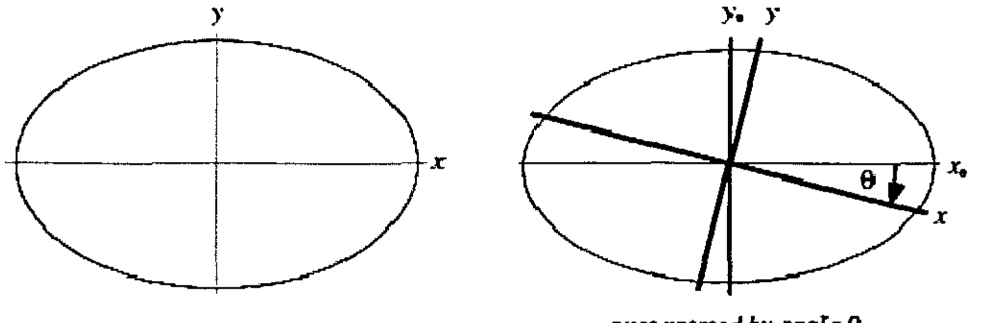
curve rotated by angle θ

In $\mathbb{R}^2$, let a curve be rotated by a clockwise angle θ, so that the original axes of the curve are $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, and the new axes of the curve are $\hat{\mathbf{x}}'$ and $\hat{\mathbf{y}}'$. The matrix transforming the original curve to the rotated curve, referred to the original $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ axes, is

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \qquad (1)$$

i.e.,

$$\mathbf{x} = R_\theta \mathbf{x}'. \qquad (2)$$



axes rotated by angle θ

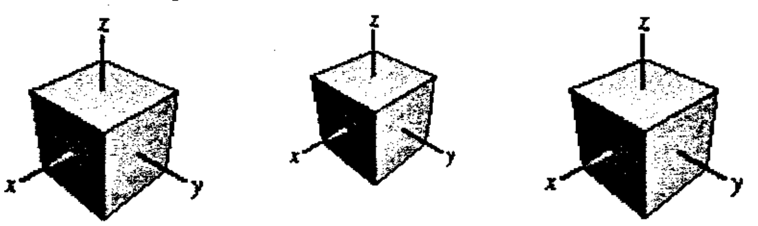
On the other hand, let the *axes* with respect to which a curve is measured be rotated by a

clockwise angle θ, so that the original axes are $\hat{x}_0$ and $\hat{y}_0$, and the new axes are $\hat{x}$ and $\hat{y}$. Then the matrix transforming the coordinates of the curve with respect to $\hat{x}$ and $\hat{y}$ is given by the matrix transpose of the above matrix:

$$R'_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{3}$$

i.e.,

$$\mathbf{x} = R'_\theta \, \mathbf{x}_0. \tag{4}$$



In $\mathbb{R}^3$, rotations about the *x*-, *y*-, and *z*-axes in a clockwise direction when looking towards the origin give the matrices

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{5}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{6}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

(Goldstein 1980, pp. 146-147 and 608; Arfken 1985, pp. 199-200).

Any rotation can be given as a composition of rotations about three axes (Euler's rotation theorem), and thus can be represented by a 3X3 matrix operating on a vector,

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \quad (8)$$

We wish to place conditions on this matrix so that it is consistent with an orthogonal transformation (basically, a rotation or rotoinverson).

In a rotation, a vector must keep its original length, so it must be true that

$$x_i' x_i' = x_i x_i \quad (9)$$

for $i=1$, 2, 3, where Einstein summulation is being used. Therefore, from the transformation equation,

$$(a_{ij} x_j)(a_{ik} x_k) = x_i x_i. \quad (10)$$

This can be rearranged to

$$a_{ij}(x_j a_{ik}) x_k = a_{ij}(a_{ik} x_j) x_k \quad (11)$$

$$= a_{ij} a_{ik} x_j x_k = x_i x_i. \quad (12)$$

In order for this to hold, it must be true that

$$a_{ij} a_{ik} = \delta_{jk} \quad (13)$$

for $j, k = 1$, 2, 3, where δ_{ij} is the Kronecker delta. This is known as the orthogonality condition, and it guarantees that

$$A^{-1} = A^T, \quad (14)$$

$$A^T A = I, \quad (15)$$

where A^T is the matrix transpose and I is the identity matrix. The identity gives the orthogonal matrix its name. Orthogonal matrices have special properties which allow them to be manipulated and identified with particular ease.

Let A and B be two orthogonal matrices. By the orthogonality condition, they satisfy

$$a_{ij} a_{ik} = \delta_{jk}, \quad (16)$$

and

$$b_{ij} b_{ik} = \delta_{jk}, \tag{17}$$

where δ_{ij} is the Kronecker delta. Now

$$c_{ij} c_{ik} = (a b)_{ij} (a b)_{jk} = a_{is} b_{sj} a_{it} b_{tk} = a_{is} a_{it} b_{sj} b_{tk} \tag{18}$$
$$= \delta_{st} b_{sj} b_{tk} = b_{tj} b_{tk} = \delta_{jk}, \tag{19}$$

so the product $C \equiv A B$ of two orthogonal matrices is also orthogonal.

The eigenvalues of an orthogonal rotation matrix must satisfy one of the following:

1. All eigenvalues are 1.

2. One eigenvalue is 1 and the other two are -1.

3. One eigenvalue is 1 and the other two are complex conjugates of the form $e^{i\theta}$ and $e^{-i\theta}$.

An orthogonal matrix A is classified as proper (corresponding to pure rotation) if

$$\det(A) = 1, \tag{20}$$

where $\det(A)$ is the determinant of A, or improper (corresponding to inversion with possible rotation; rotoinversion) if

$$\det(A) = -1.$$

(http://mathworld.wolfram.com/RotationMatrix.html)

2.10) Quaternion

The quaternions are members of noncommutative division algebra. The fundamental formula of quaternion algebra is

$$i^2 = j^2 = k^2 = i j k = -1, \tag{1}$$

The set of quaternions is denoted $\mathbb{H}$, H, or Q_8, and the quaternions are a single example of a more general class of hypercomplex numbers discovered by

Hamilton. While the quaternions are not commutative, they are associative, and they form a group known as the quarternion group.

By analogy with the complex numbers being representable as a sum of real and imaginary parts, $a \cdot 1 + b\,i$, a quaternion can also be written as a linear combination

$$H = a \cdot 1 + b\,i + c\,j + d\,k. \tag{2}$$

The quaternion

$$a + b\,i + c\,j + d\,k$$

is implemented as

Quaternion [*a*, *b*, *c*, *d*]

The quaternions can be represented using complex [2 X 2] matrices

$$H = \begin{bmatrix} z & w \\ -\overline{w} & \overline{z} \end{bmatrix} = \begin{bmatrix} a+ib & c+id \\ -c+id & a-ib \end{bmatrix}, \tag{3}$$

where z and w are complex numbers, a, b, c, and d are real, and $\overline{z}$ is the complex conjugate of z.

Quaternions can also be represented using the complex [2 X 2] matrices

$$\mathrm{U} \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

$$\mathrm{I} \equiv \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} \tag{5}$$

$$\mathrm{J} \equiv \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{6}$$

$$\mathrm{K} \equiv \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} \tag{7}$$

(Arfken 1985, p. 185). Note that here U is used to denote the identity matrix, not I. The matrices are closely related to the Pauli spin matrices σ_x, σ_y, σ_z, combined with the identity matrix.

From the above definitions, it follows that

$$\mathsf{I}^2 = -\mathsf{U} \tag{8}$$
$$\mathsf{J}^2 = -\mathsf{U} \tag{9}$$
$$\mathsf{K}^2 = -\mathsf{U}. \tag{10}$$

Therefore I, J, and $\mathsf{K}$are three essentially different solutions of the matrix equation

$$\mathsf{X}^2 = -\mathsf{U}, \tag{11}$$

which could be considered the square roots of the negative identity matrix. A linear combination of basis quaternions with integer coefficients is sometimes called a Hamiltonian interger.

In $\mathbb{R}^4$, the basis of the quaternions can be given by

$$i \equiv \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{12}$$
$$j \equiv \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$
$$k \equiv \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \tag{14}$$
$$1 \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

The quaternions satisfy the following identities, sometimes known as Hamilton's rules.

$$i^2 = j^2 = k^2 = -1 \tag{16}$$

$$i\,j = -j\,i = k \tag{17}$$
$$j\,k = -k\,j = i \tag{18}$$
$$k\,i = -i\,k = j. \tag{19}$$

They have the following multiplication table.

	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

The quaternions ± 1, $\pm i$, $\pm j$, and $\pm k$ form a non-Abelian group of order eight (with multiplication as the group operation).

The quaternions can be written in the form

$$a = a_1 + a_2\,i + a_3\,j + a_4\,k. \tag{20}$$

The quaternion conjugate is given by

$$\overline{a} = a_1 - a_2\,i - a_3\,j - a_4\,k. \tag{21}$$

The sum of two quaternions is then

$$a + b = (a_1 + b_1) + (a_2 + b_2)\,i + (a_3 + b_3)\,j + (a_4 + b_4)\,k, \tag{22}$$

and the product of two quaternions is

$$a\,b = (a_1\,b_1 - a_2\,b_2 - a_3\,b_3 - a_4\,b_4) + (a_1\,b_2 + a_2\,b_1 + a_3\,b_4 - a_4\,b_3)\,i + (a_1\,b_3 - a_2\,b_4 + a_3\,b_1 + a_4\,b_2)\,j + (a_1\,b_4 + a_2\,b_3 - a_3\,b_2 + a_4\,b_1)\,k. \tag{23}$$

The quaternion norm is therefore defined by

$$n(a) = \sqrt{a\overline{a}} = \sqrt{\overline{a}a} = \sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}. \tag{24}$$

In this notation, the quaternions are closely related to four-vector.

Quaternions can be interpreted as a scalar plus a vector by writing

$$a = a_1 + a_2 i + a_3 j + a_4 k = (a_1, \mathbf{a}), \tag{25}$$

where $\mathbf{a} \equiv [a_2\ a_3\ a_4]$. In this notation, quaternion multiplication has the particularly simple form

$$q_1 q_2 = (s_1, \mathbf{v_1}) \cdot (s_2, \mathbf{v_2}) = (s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2}, s_1 \mathbf{v_2} + s_2 \mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2}). \tag{26}$$

Division is uniquely defined (except by zero), so quaternions form a division algebra. The inverse of a quaternion is given by

$$a^{-1} = \frac{\overline{a}}{a\overline{a}}, \tag{27}$$

and the norm is multiplicative

$$n(ab) = n(a)\, n(b). \tag{28}$$

In fact, the product of two quaternion norms immediately gives the Euler four-square identity.

A rotation about the unit b vector $\hat{\mathbf{n}}$ by an angle θ can be computed using the quaternion

$$q = (s, \mathbf{v}) = (\cos(\tfrac{1}{2}\theta), \hat{\mathbf{n}} \sin(\tfrac{1}{2}\theta)) \tag{29}$$

(Arvo 1994, Hearn and Baker 1996). The components of this quaternion are called Euler parameters. After rotation, a point $p = (0, \mathbf{p})$ is then given by

$$p' = q\, p\, q^{-1} = q\, p\, \overline{q}, \tag{30}$$

since $n(q) = 1$. A concatenation of two rotations, first q_1 and then q_2, can be computed using the identity

$$q_2 (q_1\, p\, \overline{q}_1) \overline{q}_2 = (q_2\, q_1)\, p\, (\overline{q}_1\, \overline{q}_2) = (q_2\, q_1)\, p\, \overline{q_2\, q_1} \tag{31}$$

2.11) The use of quaternions to represent rotations.

We can use quaternions to represent rotations in 3D. To do this we restrict the quaternions to those with unit magnetude and we use only multiplications and not addition to represent a combination of different rotations. When quaternions are normalised in this way, together with the multiplication operation to combine rotations they form a mathematical group.

When quaternions are used in this way we can think of them as being similar to axis-angle except that real part is equal to cos(angle/2) and the complex part is made up of the axis vector times sin(angle/2). It is quite difficult to give a physical meaning to a quaternion, and many people find this similarity to axis-angle as the most intuative way to think about it, others may just prefer to think of quaternions as an interesting mathematical system which has the same properties as 3D rotations.

The quaternion in terms of axis-angle is:

q = cos(a/2) + i (x * sin(a/2)) + j (y * sin(a/2)) + k (z * sin(a/2))

where:

- a=angle of rotation.
- x,y,z = vector representing axis of rotation.

So it is closely related to the axis angle representation of rotations.

	complex number	quaternion number
can be used to represent:	2D vector	3D rotation
subsequent operations:	combined by addition	combined by multiplication
'i' represents:	rotation by 90 degrees	rotation by 180 degrees

The quaternion 'i' represents a rotation of 180 degrees about the x axis, the quaternion 'j' represents a rotation of 180 degrees about the y axis, the quaternion 'k' represents a rotation of 180 degrees about the y axis. So i*i = -1 represents a rotation of 360 degrees about the x axis.

It may seem strange that -1 represents a rotation of 360 degrees, since this is 'no change' I would expect it to be 1.

However it turns out that if (a + b i + c j + d k) represents a rotation then (- a - b i - c j - d k) represents the same rotation. In other words if we negate all the terms we get a different quaternion but it represents the same rotation. This makes sense in terms of axis angle representation, if we take the reverse angle and also reverse the axis this will produce the same result.

So both 1 and -1 represent the identity (do nothing) rotation. An object which, if rotated by 360 degrees it is inverted, is known as a spinor.

Quaternions are therefore spinors.

CHAPTER 3

3) Medical imaging

3.1) Description of DICOM format

The DICOM standard defines a set of protocols to be followed by devices claiming DICOM conformance. DICOM files are image files. DICOM files are fixed with a fixed layout identical in all files; consist of a header block with pointers to information in the file and tags that indicate the beginning of a set of information in the file.

DICOM files are medical image files that contain information: patient identification, patient positioning during examination and (very important) the image information of the examination performed on the patient. This can be an X-RAY examination, CT SCAN examination or MRI examination on the patient.

The DICOM images of the CT SCAN or MRI examinations can be used to produce a 3-D image of the patient part examined for rapid-prototyping (RP). A DICOM file consists of a header on top and image data at the bottom.

In a DICOM file there are tags that direct you to the information needed and by setting the file pointer to that offset, the information can be extracted from a file that is needed.

The image can be 8bit, 12bit, 15bit or 16bit with a specific width (X-value) and a specific length (Y-value). All the information is available in the tags in the header and is explained as follows:

DICOM FILE

```
   /-------  *****************************
   /         *****************************
HEADER       *****************************
   \         *****************************
   \-------  *****************************

   /-------  *****************************
   /         *****************************
   /         *****************************
IMAGE        *****************************
DATA         *****************************
VALUES       *****************************
   \         *****************************
   \         *****************************
    \        *****************************
    \        *****************************
     \-----  *****************************
```

The DICOM tags are as follows:
The groups are organized as follows:

0000	Command
0008	Identifying
0010	Patient
0018	Acquisition
0020	Relationship
0028	Image Presentation
4000	Text
6000-601E (even)	Overlay
7FE0	Pixel Data

Some of the more interesting elements are:

(nnnn,0000) BD S Group Length # of bytes in group nnnn
(nnnn,4000) AT M Comments

(0008,0010) AT S Recognition Code # ACR-NEMA 1.0 or 2.0
(0008,0020) AT S Study Date # yyyy.mm.dd
(0008,0021) AT S Series Date # yyyy.mm.dd
(0008,0022) AT S Acquisition Date # yyyy.mm.dd
(0008,0023) AT S Image Date # yyyy.mm.dd
(0008,0030) AT S Study Time # hh.mm.ss.frac
(0008,0031) AT S Series Time # hh.mm.ss.frac
(0008,0032) AT S Acquisition Time # hh.mm.ss.frac
(0008,0033) AT S Image Time # hh.mm.ss.frac
(0008,0060) AT S Modality # CT,NM,MR,DS,DR,US,OT

(0010,0010) AT S Patient Name
(0010,0020) AT S Patient ID
(0010,0030) AT S Patient Birthdate # yyyy.mm.dd
(0010,0040) AT S Patient Sex # M, F, O for other
(0010,1010) AT S Patient Age # xxxD or W or M or Y

(0018,0010) AT M Contrast/Bolus Agent # or NONE
(0018,0030) AT M Radionuclide
(0018,0050) AN S Slice Thickness # mm
(0018,0060) AN M KVP
(0018,0080) AN S Repetition Time # ms
(0018,0081) AN S Echo Time # ms
(0018,0082) AN S Inversion Time # ms
(0018,1120) AN S Gantry Tilt # degrees

(0020,1040) AT S Position Reference # e.g. iliac crest
(0020,1040) AN S Slice Location # in mm (signed)
(0028,0010) BI S Rows
(0028,0011) BI S Columns
(0028,0030) AN M Pixel Size # row\col in mm
(0028,0100) BI S Bits Allocated # e.g. 12 bit for CT
(0028,0101) BI S Bits Stored # e.g. 16 bit
(0028,0102) BI S High Bit # e.g. 11
(0028,0102) BI S Pixel Representation # 1 signed, 0 unsigned

(7FE0,0010) BI M Pixel Data # as described by grp 0028

3.2) X-ray image principles

The basic principles behind X-Ray imaging techniques are very much the same as these known for ordinary photography. The generation of the X-RAY beam is highly technical. The high-energy X-Ray beam induces changes in a photographic film or electric detector. X-Rays can penetrate solid material such as the human being.

The amount of X-Ray penetrating a material (or in our case the human body) depends upon how the specific material is built up. More specifically, it depends upon the type of atoms contained in the material. In general, light atoms with low atomic numbers allow more of the X-Rays to pass through than heavier atoms, i.e. with higher atomic numbers.

In broad sense the human body consists of three types of "material":

Soft tissue → containing mainly high atoms
Bone → containing heavier atoms (minerals)
Air → (or some sort of gas) built up by very light atoms

Accordingly a film exposed to X-Rays that have penetrated a human body will have white or very bright areas (little exposure), gray areas (more exposure) or nearly black areas (heavy exposure) depending upon the amount of X-Rays having penetrating various parts of the body.
For example bones letting a small amount of X-Rays pass through them will appear very bright or white on a X-Ray film and gas or air bubbles letting a large amount of X-Rays through them will appear nearly black on the X-ray film.

At most soft tissues, be it muscles, blood vessels, liver, kidneys or other, are built up by nearly the same type of atoms (mainly hydrogen, oxygen, nitrogen and carbon), it is often impossible to distinguish between them on an X-ray film without using more complicated procedures (contrast agent for conventional X-Rays or Computer Tomography (CT)).

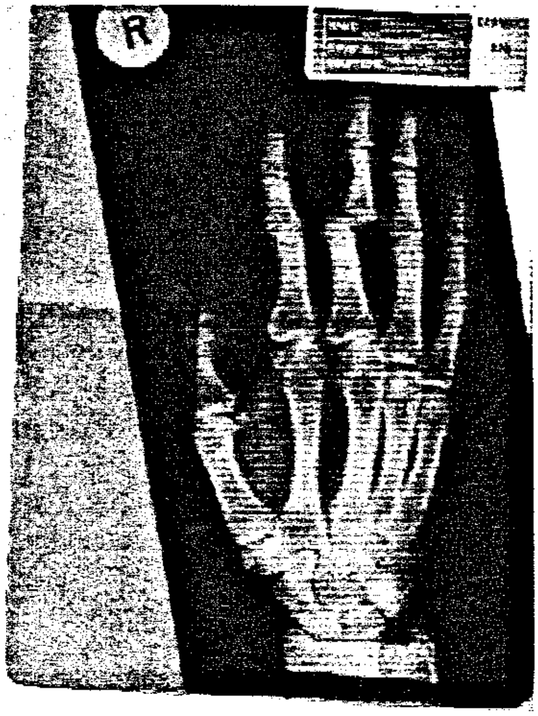


Fig 3.1: X-ray image of the hand

3.3) Example of a Computer Tomography (CT Scan) Image
Computer Tomography images are generated according to the same principles as conventional X-Ray images. The main difference is that the X-Rays, after penetrating the human body, induce electrical signals in electrical detectors instead of creating a chemical process on an X-Ray film, and that the sensitivity of the system is much higher than that of conventional X-Ray systems. With these benefits, various types of soft tissues can easily be distinguished from each other. Furthermore the CT Scan image is build up digitally and can be saved in a file format.

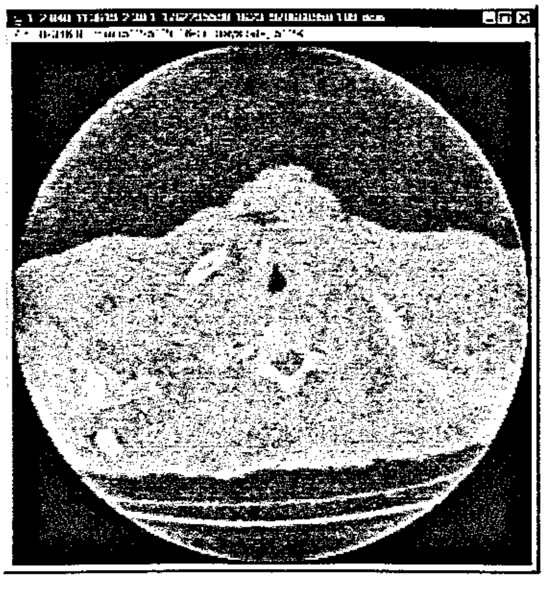


Fig 3.2: CT Scan image

3.4) Example of a Magnetic Resonance Image (MRI Scan)

Magnetic Resonance Imaging is a good way of demonstrating human anatomy, especially those of the central nervous system and those of anatomical abnormalities such as congenital heart malformations.

The main part of MRI equipment:

1) Very strong magnet normally in the range of 0.2 - 2.0 Tesla
2) A radio transmitter and receiver
3) A computer

The magnet is so large that a patient or part of the human body to be examined can be placed in it. In that sense it may look similar to a CT scanner although the principles for imaging are fundamentally different.

Important is that MRI is a different way of diagnostic imaging, different from conventional X-ray examinations and CT Scan as there is no radiation involvement.

MRI is a procedure in which radio waves and a powerful magnet linked to a computer are used to create detailed pictures of areas inside the body. These pictures can show the difference between normal and diseased tissue. MRI makes better images of organs and soft tissue than other scanning techniques, such as CT or x-ray. MRI is especially useful for imaging the brain, spine, the soft tissue of joints, and the inside of bones. MRI is also called nuclear magnetic resonance imaging.

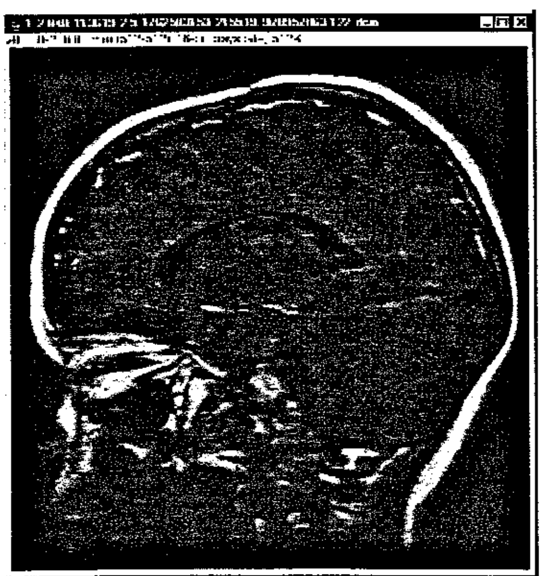


Fig 3.3: MRI image

CHAPTER 4

4) Rapid Prototyping

Rapid Prototyping is the speedy fabrication of sample parts for demonstration, evaluation, or testing. It typically utilizes advanced layer manufacturing technologies that can quickly generate complex three-dimensional objects directly from computer-based models devised by Computer Aided Design (CAD). This computer representation is sliced into two-dimensional layers, whose descriptions are sent to the fabrication equipment to build the part layer by layer. Rapid prototyping includes many different fabrication technologies. Stereolithography (STL), selective laser sintering (SLS), laminated object manufacturing (LOM), and fused deposition modeling (FDM) are a few examples.

4.1) Rapid Prototyping (Medical models)

As medical treatment is becoming more and more advanced, the need for a solid representation of a human organ or area of pathology has also grown.

Since the DICOM standard is the standard of representing medical images, the DICOM files are used to develop a 3D image of organ, extremity (bone) or area of interest.

4.2) Format of an SLC file

SLC (Stereo Lithography Contour) files are used in the fused deposition modeling (FDM) process. They describe the contours on, and the thickness of, each build layer. SLC files are used in Rapid Prototyping when designing a physical model.

SLC files cut 2D contours of the 3D database. These contour lines are poly lines. The advantage to using this file format is that the geometry does not need to be tessellated and therefore fewer translations are required between the original geometry and the data sent to the SLA machine to be built.

The SLC file Rapid Prototyping file format is an ASCII text file format. Comments may be included. Comments begin with a "#" and end at the end of that line. Units are in inches.

SLC files can be arranged in an order: X, Y, Z
X, Y, Z should be positive floats in the range from 0 to 22.86cm. Layer thickness (dz) range from 0.0127cm to 0.0381cm. The standard layer thickness is 0.0254cm.

4.2.1) Example of an SLC file

The bold characters should be typed as is. The italic characters should be substituted with their corresponding values.

```
Slice V0       # 0 = zero
FILE(filename)# filename may include full path
SLICES()       #
Z z dz         # z = absolute z of layer, dz = layer thickness
x y            # Provide the coordinates of the corner points
x y            # along each contour. Do not duplicate the
x y            # first point. The is no upper limit on the
x y            # number of points in a contour.
x y            #
C              # Terminate the contour with a "C".
x y            # Begin the second contour on this level...
x y            #
x y            #
x y            #
x y            #
C              # Terminate the second contour.
Z z dz         # Begin a new layer (z = z.old + dz)
x y            # Begin a new contour on a new level...
x y            #
x y            #
x y            #
x y            #
C              # Terminate the contour.
END            # All following lines are ignored.
```

If you have a 3D image of the product in matrix form, by having the lengths x,y,z the ASCII floating point numbers x,y,z can be written to a file using C, Matlab or C++.

4.3) Describe the format of an STL file

The stereolithography format (STL) is an ASCII or binary file used in manufacturing. It is a list of the triangular surfaces that describe a computer generated solid model. This is the standard input for most rapid prototyping machines.

4.3.1) ASCII STL file

The first line is a description line that must start with the word "solid" in lower case, it then normally contains the file name, author, date etc. The last line should be the keyword "endsolid". The lines between the above contain descriptions of 3 vertex facets including their normals, the ordering of the vertices should comply with the right hand rule.

4.3.2) Example of an ASCII STL file

```
solid
  facet normal 0.0 0.0 -1.0
  outer loop
    vertex -1.5 -1.5 1.4
    vertex 0.0 1.7 1.4
    vertex 1.5 -1.5 1.4
  endloop
  endfacet
endsolid
```

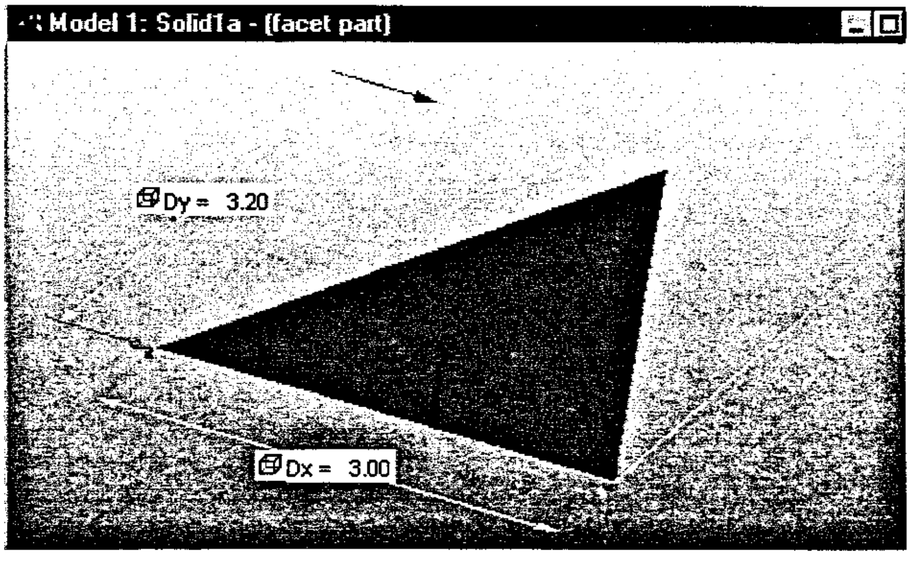


Fig 4.1: Triangle describe by ASCII STL file example

Often the normals need not be provided and they will be generated by the parsing software/system. The main restriction placed upon the facets in STL files is that all adjacent facets must share two common vertices.

4.4) Binary STL file

Binary STL files consist of an 80-byte header line that can be interpreted as a comment string. The following 4 bytes interpreted as a long integer give the total number of facets. What follows is a normal and 3 vertices for each facet, each coordinate represented as a 4-byte floating-point number (12 bytes in all). There is a 2-byte spacer between each facet. The result is that each facet is represented by 50 bytes, 12 for the normal, 36 for the 3 vertices, and 2 for the color.

4.4.1) Color in an STL file

Only use 15bit color not 32bit color as conventionally. One bit is used to indicate if the color is used. The RGB component that that is usually described by an 8bit integer between 0 - 255 is now described by a 5bit inter between 0 -17. Gray scale is created when RED, GREEN and BLUE are the same value example 0,0,0 describes black where 255,255,255 describe white and 127,127,127 describe a Gray offset.

2 (previously unused)	bit 15	always 1 to distinguish from normal STL files (For Materialise: 1 The part colour will be applied for this triangle 0 The part colour is not used, the triangle colour is described in the following 15 bits.)
	[redacted]	[redacted]

Fig 4.2: Color in an STL file

4.5) Vertex-to-Vertex rule and triangles

All facets in an STL must share two common vertices.

The most common error in an STL file is non-compliance with the vertex-to-vertex rule. The STL specifications require that all adjacent triangles share two common vertices. This is illustrated in figure 4.3. The figure on the left shows a top triangle containing a total of 4 vertex points. The outer vertices of the top triangle are not shared with and one other single triangle. The lower 2 triangles each contain one of the points as well as the fourth invalid vertex point. To make this valid under the vertex-to-vertex rule the top triangle must be subdivided as in the example on the right.

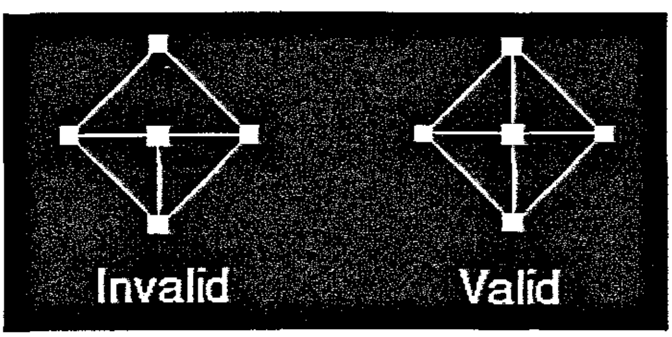


Fig 4.3: Vertex-to-Vertex rule and triangles

CHAPTER 5

5) Manipulation of medical imaging

To convert the medical image slice like the CT-scan slice or MRI slice to Rapid prototyping, the medical DICOM image needs to be displayed and the information of interest needs to be extracted.

5.1) Reading and displaying DICOM medical images.

To read and display the information in a DICOM file, the information described by the tags are read and this information is used to create the image window and display the different pixels.

The format of a DICOM file is the header on top and the image information at the bottom. The tags in the DICOM file describe the information in the DICOM image. The tags can be detected and the information can be obtained to display the DICOM image.

The tags required to display the image are:

```
(0028,0010) BI S Rows
(0028,0011) BI S Columns
(0028,0030) AN M Pixel Size          # row\column in mm
(0028,0100) BI S Bits Allocated      # e.g. 12 bit for CT
(0028,0101) BI S Bits Stored         # e.g. 16 bit
(0028,0102) BI S High Bit            # e.g. 11
(0028,0102) BI S Pixel Representation  # 1 signed, 0 unsigned

(7FE0, 0010) BI M Pixel Data         # as described by group 0028
```

The tag (0028,0010) is read and the information after the tag displays the number of rows in the image. Then the columns tag (0028,0011) and the number of columns after the tag, the height and the width of the image are obtained.

To display the different pixel values on the image plane of the width and height previously read, detect the tag (7FE0, 0010) that gives the start of the pixel information. The pixel information can be 8bit, 12 bit or 16 bit as describe by the tag (0028,0101).

To display an 8 bit image, and if 12 bits are stored the 12 bit information are shifted 3 bits to the left for display. This will ensure that a 8 bit gray scale image is displayed.

The final image will be a medical image of an X-ray, CT or MRI of width described by the tag (0028,0010) and height (0028,0011) with pixel information obtained from the tag (7FE0, 0010).

```
00 31 2E 32 2E 38 34 30 2E 31 31 33 36 31 39 2E 32 2E 35 2E 31
37 36 32 35 38 33 31 35 33 2E 32 31 35 35 31 39 2E 39 37 38 39   Header information
35 37 30 36 33 2E 37 38 00 43 00 62 10 04 00 00 00 35 38 30 31
43 00 6F 10 14 00 00 00 20 30 5C 30 2E 30 30 30 30 30 30 5C 30
2E 30 30 30 30 30 30 E0 7F 00 00 04 00 00 00 0C 00 08 00 [E0 7F] <- Pixel Data tag
[10 00] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   Pixel information
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Fig 5.1: Information in a DICOM file

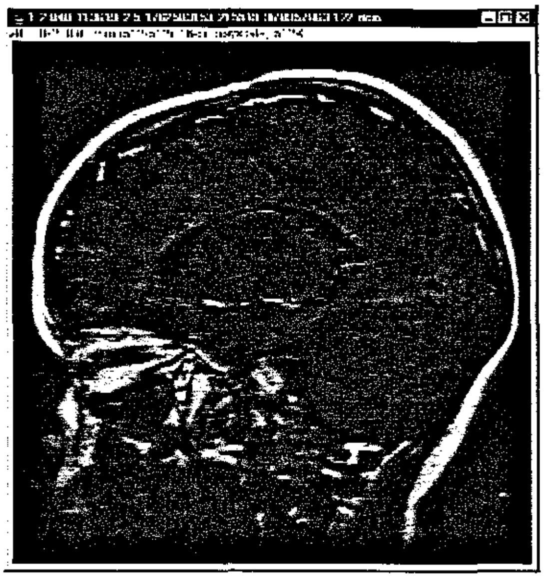


Fig 5.2: Image Displayed as pixels described in pixel data

5.2) Alter the brightness of the DICOM image

To view the image optimally, adjustments to the image can be made and for this the brightness of the image can be changed.

To change the brightness of the image, the pixel values that describe pixels in the image is adjusted either higher for brighter of lower for darker. The color on the gray scale image is represented by an eight bit value between 0 - 255. By changing the brightness the pixel value is adjusted by 1,10 or more up or down.

5.3)Detection of 1D and 2D structures in an image. Classification and identification of such structures.

5.3.1) Filter out the structure of interest

In a medical image of the human body all structures in a specific image slice is presented in that specific image slice. In a CT scan of the chest, both the lungs as well as the heart, major arteries and the ribs are presented on that slice.
If the interest is in the lungs, the image of the lungs must be filtered out.

On the DICOM slice of the chest, the pixel value of the lungs are presented by a specific number between 0 - 255. Normally the pixel value is in the range of 80 - 100. Now the image can be scanned and all the pixel values between 80 and 100 are highlighted and filtered out. This will provide only the structure of interest that is is this case the lungs on a CT slice of the chest.

5.4) Outer rim edge detection

After the structure of interest has been filtered out, the structure must be converted to 3D modeling for rapid prototyping. For 3-D modeling, the outer rim of the image is required. To convert the 2D medical images to 3D images, outer rim edge detection is needed. A method for outer rim edge detection was developed in this thesis.

In outer rim edge detection only the outer rim of the structure is detected. Outer rim edge detection is the process that the image is scanned for a specific pixel value. The image is scanned in a specific manner in order to obtain only the outer rim of the structure.

The method of scanning is that every line is scanned from left to right, right to left, top to bottom and bottom to top. The scan starts at the left upper corner of the image and the first line is scanned from left to right. The reason for scanning is that a specific pixel value needs to be detected. When the line is scanned from left to right, and the set pixel value is detected, then the position of that pixel position is obtained and stops scanning from left to right.

Repeat the scanning process of the same line but now from right to left. Keep looking for the set pixel value and if the set pixel value is detected, the position is obtained and the scan of that line is stopped.

Repeat the same process for every line from top bottom, until the bottom line is reached. This will give the left and right outer rim of the structure without the internal pixel value.

Do the same to the image by scanning the lines from top to bottom by starting from the left upper line. After completing scanning from top tom bottom, the outer rim from the top and bottom is detected.

The whole method of outer rim edge detection both from left to right and top to bottom will provide the outer rim of the of the structure without the internal pixels.
Mathematical method for aouter rim detection.

If P(x,y,z) >= Pmax
Pi(x,y,z)= P(x,y,z)

for n, 0, 1, 2, 3, ...
 for X = Xmax, Xmax-1, Xmax – 2, ...
  for Y = 0, 1, 3, ...
   for Y = Ymax, Ymax -1, Ymax – 2, ...
   for Z = Zmax, Zmax -1, Zmax – 2, ...

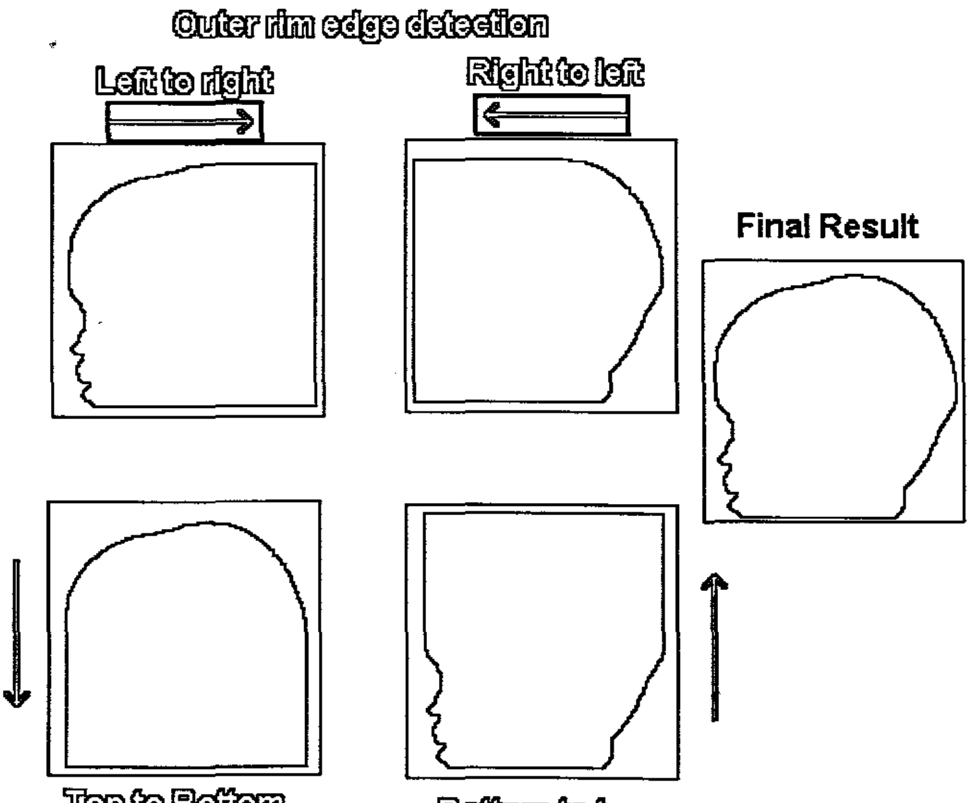


Fig 5.3: Result of outer rim detection

5.5) Edge detection of the anatomical structure

Even if the outer rim is being detected, specific edge detection of a complex anatomical structure is required.

For this the image is converted to a two-color-image color presentation. Every pixel is evaluated and the surrounding pixels will determine if that specific pixel is an edge pixel or not.

If one pixel surrounded by eight other pixels, as in the image below, then the pixels can be represented by 1 and zeros. A zero(0) represent a pixel with a different value as that in the centre.

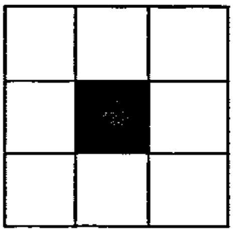


Fig 5.3.1: Centre pixel in an 8 pixel matrix

If a black pixel is epresented by 1 and awhite pixel is represented by 0, the a pixel is an edge pixel if it is represented b one of the following rouines:

000 100 010 001 000 000 000 000 000
000
010 or 010 or 010 or 010 or 110 or 011 or 010 or 010 or 010 or
010
000 000 000 000 000 000 000 100 010
001

A pixel is an edge pixel if that pixel is connectec to one other pixel in the 8 pixel matrix in the following ways.

000 100 010 001 000 000 000 000 000
000
010 or 010 or 010 or 010 or 110 or 011 or 010 or 010 or 010 or
010
000 000 000 000 000 000 000 100 010
001

Now scanning all the pixels and checking the surrounding pixels of a specific pixel will detect the edge of the whole image. The edge is detected when the difference is calculated between the center pixel and any surrounding pixel. If the difference is one then the center pixel is an edge pixel and if the difference is zero then the center pixel is not an edge pixel.

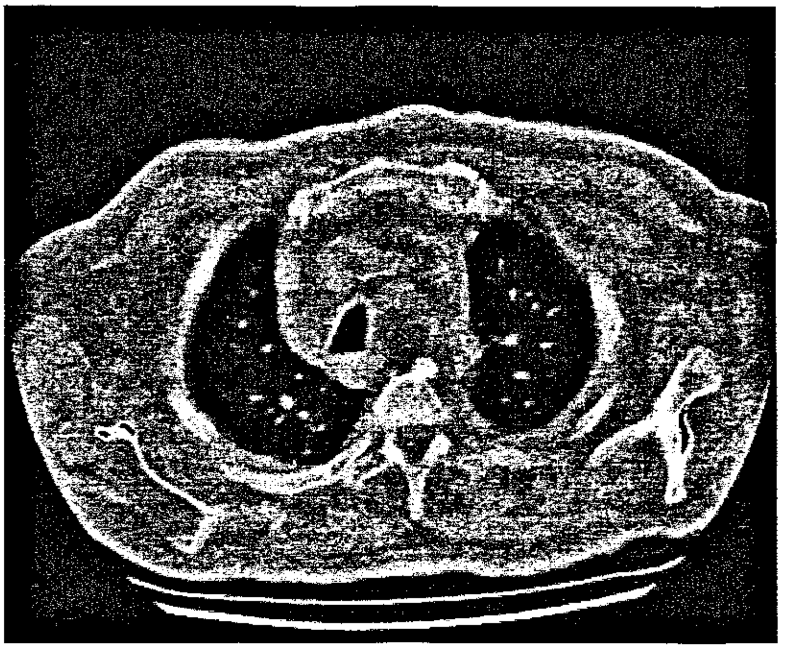


Fig 5.4: Display the edge of an anatomical structure

5.6) Display the anatomical structure of interest on the medical slice

It is necessary to display the anatomical part of interest on the original medical image. After the outer rim and the edge of a specific structure has been demonstrated, the image can be highlighted on the original image slice.

To highlight the structure on the image slice, the edge image of the slice is loaded as a mask. Now the original image is displayed and the mask of the edge image is displayed on the image.

Algorithm to demonstrate the anatomical structure of interest

1) Load the DICOM image by reading the pixel information and display these gray scale information in the image window.
2) Filter out the anatomical structure of interest by detecting the gray scale value of that specific part.
3) Do outer rim edge detection on the extracted part.
4) Now with only the outer rim, load this as a mask.
5) Display the original DICOM image in the image window.
6) Ad the mask extracted on the original image.

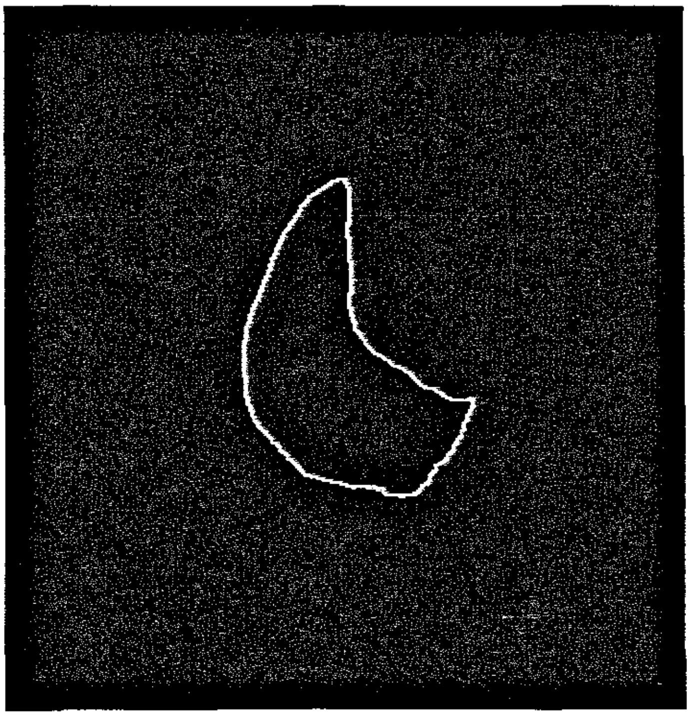


Fig 5.5: Demonstrate the anatomical structure of interest

CHAPTER 6

6) Description converting medical images to rapid prototyping

The medical image in an image window is described by a specific height and width and each pixel represent a specific RGB (RED, GREEN and BLUE) value. This information is obtained from the DICOM tags, which describe the height and width. Now the image is completed with a set of pixels of which the position is obtained from the DICOM tag (7FE0, 0010).

Every pixel has a pixel value, x-position, y-position and also a pixel height and a pixel width. The information of the pixel width and pixel height is obtained from the DICOM tags.

6.1) The image information describing the DICOM file.

0002,0010 Transfer Syntax UID: 1.2.840.10008.1.2

0008,0005 Specific Character Set: ISO_IR 100
0008,0008 Image Type: ORIGINAL\PRIMARY\AXIAL
0008,0016 SOP Class UID: 1.2.840.10008.5.1.4.1.1.2
0008,0018 SOP Instance UID:

1.2.840.113619.2.30.1.1762295590.1623.978668950.109
0008,0020 Study Date: 20010105
0008,0021 Series Date: 20010105
0008,0022 Acquisition Date: 20010105
0008,0023 Image Date: 20010105
0008,0030 Study Time: 083501
0008,0031 Series Time: 083709

0008,0032 Acquisition Time: 083848
0008,0033 Image Time: 083852
0008,0050 Accession Number: 0000000001
0008,0060 Modality: CT
0008,0070 Manufacturer: GE MEDICAL SYSTEMS
0008,0080 Institution Name: Toronto Hosp, West Div.

0008,0090 Referring Physician's Name: PHYSICIAN
0008,1010 Station Name: TWD1_OC0
0008,1030 Study Description: CHEST
0008,103E Series Description: HELICAL CHEST
0008,1060 Name of Physician(s) Reading Study: RADIOLOGIST
0008,1070 Operator's Name: BK
0008,1090 Manufacturer's Model Name: HiSpeed CT/i

0010,0010 Patient's Name: PATIENT1
0010,0020 Patient ID: 0000001
0010,0030 Patient's Birth Date: 19700101
0010,0040 Patient's Sex: M
0010,1010 Patient's Age: 089Y

0010,1030 Patient's Weight: 0.000000
0010,21B0 Additional Patient History: WIGHT LOSS ,COUGH

0018,0022 Scan Options: HELICAL MODE
0018,0050 Slice Thickness: 5.000000
0018,0060 KVP: 120

0018,0088 Spacing Between Slices: 6.500000
0018,0090 Data Collection Diameter: 480.000000
0018,1020 Software Versions(s): 05

0018,1100 Reconstruction Diameter: 346.000000
0018,1110 Distance Source to Detector: 1099.3100585938
0018,1111 Distance Source to Patient: 630.000000
0018,1120 Gantry/Detector Tilt: 0.000000
0018,1130 Table Height: 167.100006
0018,1140 Rotation Direction: CW
0018,1150 Exposure Time: 800
0018,1151 Transmitting Coil: 200
0018,1152 Exposure: 200
0018,1160 Screen Type: BODY FILTER
0018,1190 Focal Spot(s): 0.700000
0018,1210 Convolution Kernel: STANDARD
0018,5100 Patient Position: FFS

0020,000D Study Instance UID:
1.2.840.113619.2.30.1.1762295590.1623.978668949.886
0020,000E Series Instance UID:
1.2.840.113619.2.30.1.1762295590.1623.978668949.890
0020,0010 Study ID: 40933
0020,0011 Series Number: 2
0020,0012 Acquisition Number: 1
0020,0013 Image Number: 1
0020,0032 Image Position (Patient): -161.399994\ -148.800003\4.700000
0020,0037 Image Orientation (Patient):
1.000000\0.000000\0.000000\0.000000\1.000000\0.000000
0020,0052 Frame of Reference UID:
1.2.840.113619.2.30.1.1762295590.1623.978668949.886.8493.0.12
0020,0060 Laterality:

0020,1040 Position Reference Indicator: SN
0020,1041 Slice Location: 4.6999998093

0028,0002 Samples per Pixel: 1
0028,0004 Photometric Interpretation: MONOCHROME2
0028,0010 Rows: 512
0028,0011 Columns: 512
0028,0030 Pixel Spacing: 0.675781\0.675781
0028,0100 Bits Allocated: 16
0028,0101 Bits Stored: 16
0028,0102 High Bit: 15
0028,0103 Pixel Representation: 1
0028,0120 Pixel Padding Value: 32768
0028,1050 Window Center: 40
0028,1051 Window Width: 400
0028,1052 Rescale Intercept: -1024
0028,1053 Rescale Slope: 1

7FE0,0010 Pixel Data: 6417

Title: '1.2.840.113619.2.30.1.1762295590.1623.978668950.109.dcm'
Width: 346.00 mm (512)
Height: 346.00 mm (512)
Resolution: 1.5 pixels per mm
Bits per pixel: 16 (unsigned short)
Display range: 30768.00 - 35170.00
No Threshold

Calibration Function: y = a+bx
a: -32768.000000
b: 1.000000
Unit: "gray value"

6.2) To convert the DICOM file to STL format the following information are needed from the image information:

Rows: 512
Columns: 512
Width: 346.00 mm (512)
Height: 346.00 mm (512)
Slice Thickness: 5.000000
Pixel Spacing: 0.675781\0.675781
Unit: "gray value"

Algorithm to convert DICOM slice to STL file

1) Read in the DICOM file
2) Go to the left top pixel and scan every pixel
3) Use the width and height with the pixel spacing to create 2 STL triangle to produce a 3D STL pixel
4) Complete the whole image area by reading in the relative x, y positions and multiply it with the pixel spacing.
5) Also read in the gray scale value of the pixel and convert it to the gray scale triangle color.

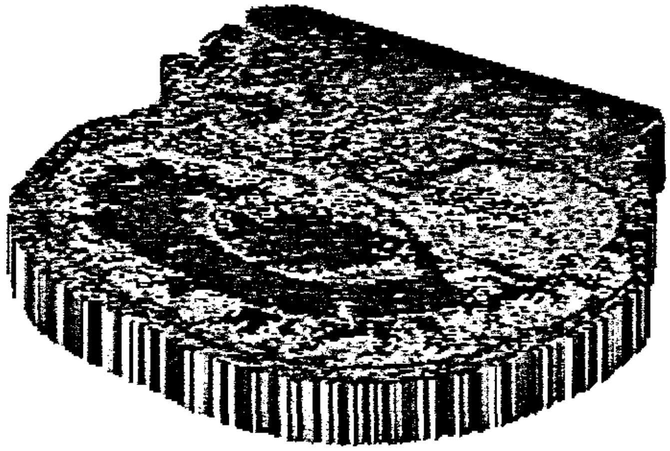


Fig 6.1: Converting the DICOM file to STL format (2D of 3D one slice)

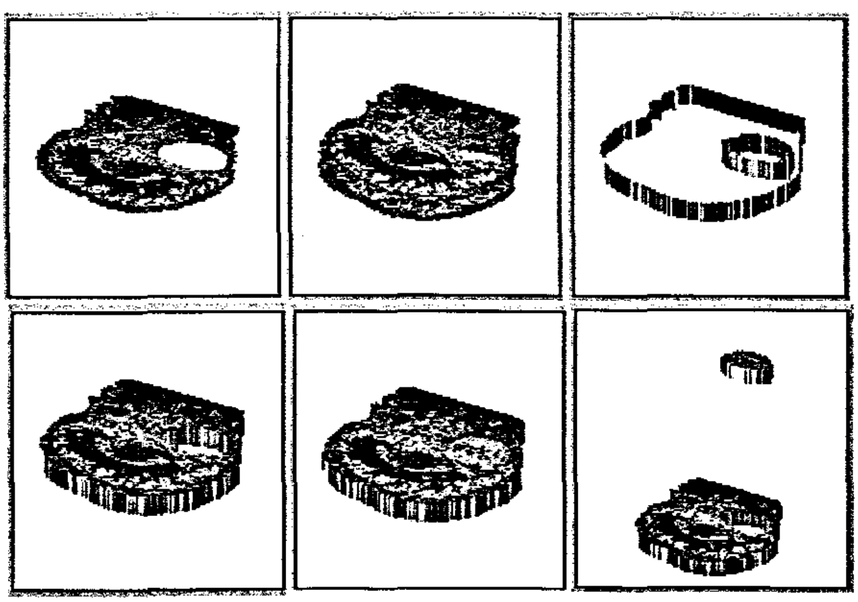


Fig 6.2: Steps for converting anatomical extraction to STL format

6.3) Converting the anatomical structure to STL format (no color)
STL format is the outer rim representation of information for rapid prototyping.

The STL file can be in ASCII form as below or in binary form.

The information in an STL file represents the pixel information of the outer surface of the structure. The x, y, z position of the pixel is described in the STL file.

After the image is read and displayed in an image window, all the pixel positions are available and using this information the data is obtained to write the STL file.

In the example of this MRI image of the scull, then:

Absolute z of layer = 5mm as read from DICOM tag OFE0
Layer thickness = 5mm
Now scan the every pixel in the image and when a pixel value of 255 is detected the x, y position is calculated with the pixel width, pixel height and pixel position on the image.

With the keywords required to complete the STL file, the whole STL file can be written.

```
solid
  :
  :
  facet normal 0.0 0.0 1.0
  outer loop
    vertex  1.0  1.0  0.0
    vertex -1.0  1.0  0.0
    vertex  0.0 -1.0  0.0
  endloop
  endfacet
  :
  :
endsolid
```

6.4) Converting the anatomical structure to STL format (in color)
The STL file can be represented both in binary as well in ASCII format. Color in an STL file is represented in the binary file format. The RGB component of the STL file is represented in 15-bit integer. The first bit indicates if the color information is used and 5 bits represent the RGB components.

Normally the RGB components are represented by 8bit values and now only 5bit RGB values are used that gives a color scale of 0 to 15364.

6.5) Creating the STL edges from the DICOM file
In the case of creating an STL file from an MRI image, the STL file has to be enclosed from all sides. To enclose the STL file the edges need to be created.

Edges are form XY if the edges are described from left to right and YX if the edges are described from right to left. To obtain XY edges, the same process is used as outer rim detection where the image is scanned from left to right and when an edge is detected, an XY STL pixel is created. The image is scanned from left to right only and not from top to bottom.

The YX edges are obtained when the image is scanned from right to left and when an edge is detected, a YX STL pixel is created.

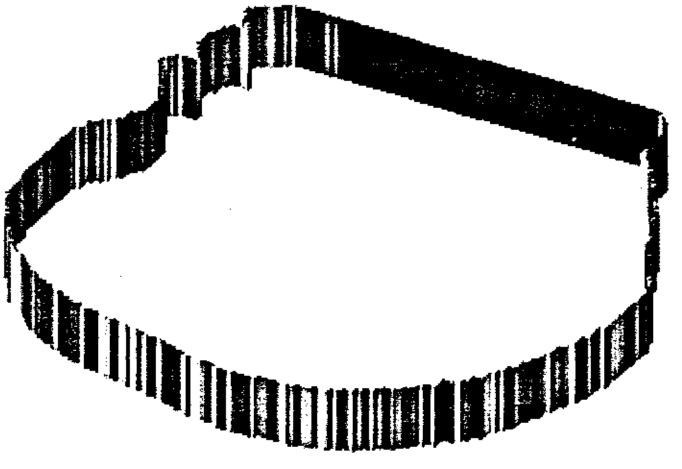


Fig 6.3 Edge STL file of the image of scull (2D of 3D)

6.6) Converting the anatomical structure to SLC format

SLC format is the slice representation of information for rapid prototyping.

The format of an SLC file to represent the pixel information is given below. The x, y position of the pixel is described in the SLC file.

After the image is read and displayed in an image window, all the pixel positions are available and using these information the data is obtained to write the SLC file.

In the example of the MRI image of the scull, then:

absolute z of layer = 5mm as read from DICOM tag OFE0
layer thickness = 5mm

Now scan the every pixel in the image and when a pixel value of 255 is detected the x,y position is calculated with the pixel width, pixel height and pixel position on the image.

With the keywords required to complete the SLC file, the whole SLC file can be written.

```
Slice V0        # 0 = zero
FILE(filename)# filename may include full path
SLICES()        #
Z z dz          # z = absolute z of layer, dz = layer thickness
x y             # Provide the coordinates of the corner points
x y             # along each contour. Do not duplicate the
x y             # first point. The is no upper limit on the
x y             # number of points in a contour.
x y             #
C               # Terminate the contour with a "C".
x y             # Begin the second contour on this level...
x y             #
x y             #
x y             #
x y             #
C               # Terminate the second contour.
Z z dz          # Begin a new layer (z = z.old + dz)
x y             # Begin a new contour on a new level...
x y             #
x y             #
```

```
x y      #
x y      #
C        # Terminate the contour.
END      # All following lines are ignored.
```

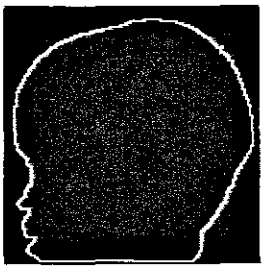


Fig 6.4: Original image

6.7) SLC file created from DICOM image

Slice V1 #
FILE(c:\ScullMRIPlane1.slc) # filename with full path
SLICES() #
Z 1.000 0.010 #
4.799 0.404 #Coordinates of corner piont (273 ,23) along counter 1 . [Format (x,y)]
C #Terminate contour : 1 (1 point in countour 1) ,(1 points up till contour 1)
4.852 0.422 #Coordinates of corner piont (276 ,24) along counter 2 . [Format (x,y)]
4.869 0.422 #Coordinates of corner piont (277 ,24) along counter 2 . [Format (x,y)]
4.887 0.422 #Coordinates of corner piont (278 ,24) along counter 2 . [Format (x,y)]
4.904 0.422 #Coordinates of corner piont (279 ,24) along counter 2 . [Format (x,y)]
4.922 0.422 #Coordinates of corner piont (280 ,24) along counter 2 . [Format (x,y)]
4.939 0.422 #Coordinates of corner piont (281 ,24) along counter 2 . [Format (x,y)]
4.957 0.422 #Coordinates of corner piont (282 ,24) along counter 2 . [Format (x,y)]
4.975 0.422 #Coordinates of corner piont (283 ,24) along counter 2 . [Format (x,y)]
4.992 0.422 #Coordinates of corner piont (284 ,24) along counter 2 . [Format (x,y)]
5.010 0.422 #Coordinates of corner piont (285 ,24) along counter 2 . [Format (x,y)]
5.027 0.422 #Coordinates of corner piont (286 ,24) along counter 2 . [Format (x,y)]
5.045 0.422 #Coordinates of corner piont (287 ,24) along counter 2 . [Format (x,y)]
5.062 0.422 #Coordinates of corner piont (288 ,24) along counter 2 . [Format (x,y)]
5.080 0.422 #Coordinates of corner piont (289 ,24) along counter 2 . [Format (x,y)]
5.098 0.422 #Coordinates of corner piont (290 ,24) along counter 2 . [Format (x,y)]
5.115 0.422 #Coordinates of corner piont (291 ,24) along counter 2 . [Format (x,y)]
5.133 0.422 #Coordinates of corner piont (292 ,24) along counter 2 . [Format (x,y)]
5.150 0.422 #Coordinates of corner piont (293 ,24) along counter 2 . [Format (x,y)]
5.168 0.422 #Coordinates of corner piont (294 ,24) along counter 2 . [Format (x,y)]
5.186 0.422 #Coordinates of corner piont (295 ,24) along counter 2 . [Format (x,y)]

5.203 0.422 #Coordinates of corner piont (296 ,24) along counter 2 . [Format (x,y)]
5.221 0.422 #Coordinates of corner piont (297 ,24) along counter 2 . [Format (x,y)]
5.238 0.422 #Coordinates of corner piont (298 ,24) along counter 2 . [Format (x,y)]
5.256 0.422 #Coordinates of corner piont (299 ,24) along counter 2 . [Format (x,y)]
5.273 0.422 #Coordinates of corner piont (300 ,24) along counter 2 . [Format (x,y)]
5.291 0.422 #Coordinates of corner piont (301 ,24) along counter 2 . [Format (x,y)]
5.309 0.422 #Coordinates of corner pionts (302 ,24) along counter 2 . [Format (x,y)]
5.326 0.422 #Coordinates of corner piont (303 ,24) along counter 2 . [Format (x,y)]
5.344 0.422 #Coordinates of corner piont (304 ,24) along counter 2 . [Format (x,y)]
5.361 0.422 #Coordinates of corner piont (305 ,24) along counter 2 . [Format (x,y)]
5.379 0.422 #Coordinates of corner piont (306 ,24) along counter 2 . [Format (x,y)]
5.396 0.422 #Coordinates of corner piont (307 ,24) along counter 2 . [Format (x,y)]
5.414 0.422 #Coordinates of corner piont (308 ,24) along counter 2 . [Format (x,y)]
5.432 0.422 #Coordinates of corner piont (309 ,24) along counter 2 . [Format (x,y)]
5.449 0.422 #Coordinates of corner piont (310 ,24) along counter 2 . [Format (x,y)]
5.467 0.422 #Coordinates of corner piont (311 ,24) along counter 2 . [Format (x,y)]
5.484 0.422 #Coordinates of corner piont (312 ,24) along counter 2 . [Format (x,y)]
5.502 0.422 #Coordinates of corner piont (313 ,24) along counter 2 . [Format (x,y)]
5.520 0.422 #Coordinates of corner piont (314 ,24) along counter 2 . [Format (x,y)]
5.537 0.422 #Coordinates of corner piont (315 ,24) along counter 2 . [Format (x,y)]
5.555 0.422 #Coordinates of corner piont (316 ,24) along counter 2 . [Format (x,y)]
5.572 0.422 #Coordinates of corner piont (317 ,24) along counter 2 . [Format (x,y)]
5.590 0.422 #Coordinates of corner piont (318 ,24) along counter 2 . [Format (x,y)]
5.607 0.422 #Coordinates of corner piont (319 ,24) along counter 2 . [Format (x,y)]
C #Terminate contour : 2 (44 points in countour 2) ,(45 points up till contour 2)

4.641 0.439 #Coordinates of corner piont (264 ,25) along counter 3 . [Format (x,y)]
4.658 0.439 #Coordinates of corner piont (265 ,25) along counter 3 . [Format (x,y)]
4.676 0.439 #Coordinates of corner piont (266 ,25) along counter 3 . [Format (x,y)]
4.693 0.439 #Coordinates of corner piont (267 ,25) along counter 3 . [Format (x,y)]
4.711 0.439 #Coordinates of corner piont (268 ,25) along counter 3 . [Format (x,y)]
4.729 0.439 #Coordinates of corner piont (269 ,25) along counter 3 . [Format (x,y)]
4.746 0.439 #Coordinates of corner piont (270 ,25) along counter 3 . [Format (x,y)]
4.764 0.439 #Coordinates of corner piont (271 ,25) along counter 3 . [Format (x,y)]
...
...
...
6.908 8.754 #Coordinates of corner piont (393 ,498) along counter 476. [Format (x,y)]
C #Terminate contour : 476 (22 points in countour 476) ,(34871 points up till contour 476)
1.020 8.771 #Coordinates of corner piont (58 ,499) along counter 477. [Format (x,y)]
1.037 8.771 #Coordinates of corner piont (59 ,499) along counter 477. [Format (x,y)]
1.055 8.771 #Coordinates of corner piont (60 ,499) along counter 477. [Format (x,y)]
1.072 8.771 #Coordinates of corner piont (61 ,499) along counter 477. [Format (x,y)]
1.090 8.771 #Coordinates of corner piont (62 ,499) along counter 477. [Format (x,y)]
1.160 8.771 #Coordinates of corner piont (66 ,499) along counter 477. [Format (x,y)]
1.178 8.771 #Coordinates of corner piont (67 ,499) along counter 477. [Format (x,y)]
1.266 8.771 #Coordinates of corner piont (72 ,499) along counter 477. [Format (x,y)]
1.283 8.771 #Coordinates of corner piont (73 ,499) along counter 477. [Format (x,y)]
6.768 8.771 #Coordinates of corner piont (385 ,499) along counter 477. [Format (x,y)]
6.785 8.771 #Coordinates of corner piont (386 ,499) along counter 477. [Format (x,y)]
6.803 8.771 #Coordinates of corner piont (387 ,499) along counter 477. [Format (x,y)]
6.820 8.771 #Coordinates of corner piont (388 ,499) along counter 477. [Format (x,y)]

6.838 8.771 #Coordinates of corner piont (389 ,499) along counter 477. [Format (x,y)]
6.873 8.771 #Coordinates of corner piont (391 ,499) along counter 477. [Format (x,y)]
6.891 8.771 #Coordinates of corner piont (392 ,499) along counter 477. [Format (x,y)]
C #Terminate contour : 477 (16 points in countour 477) ,(34887 points up till contour 477)
1.037 8.789 #Coordinates of corner piont (59 ,500) along counter 478. [Format (x,y)]
1.055 8.789 #Coordinates of corner piont (60 ,500) along counter 478. [Format (x,y)]
1.072 8.789 #Coordinates of corner piont (61 ,500) along counter 478. [Format (x,y)]
1.090 8.789 #Coordinates of corner piont (62 ,500) along counter 478. [Format (x,y)]
1.107 8.789 #Coordinates of corner piont (63 ,500) along counter 478. [Format (x,y)]
1.125 8.789 #Coordinates of corner piont (64 ,500) along counter 478. [Format (x,y)]
1.160 8.789 #Coordinates of corner piont (66 ,500) along counter 478. [Format (x,y)]
1.178 8.789 #Coordinates of corner piont (67 ,500) along counter 478. [Format (x,y)]
1.195 8.789 #Coordinates of corner piont (68 ,500) along counter 478. [Format (x,y)]
6.803 8.789 #Coordinates of corner piont (387 ,500) along counter 478. [Format (x,y)]
6.820 8.789 #Coordinates of corner piont (388 ,500) along counter 478. [Format (x,y)]
6.838 8.789 #Coordinates of corner piont (389 ,500) along counter 478. [Format (x,y)]
6.855 8.789 #Coordinates of corner piont (390 ,500) along counter 478. [Format (x,y)]
6.873 8.789 #Coordinates of corner piont (391 ,500) along counter 478. [Format (x,y)]
6.891 8.789 #Coordinates of corner piont (392 ,500) along counter 478. [Format (x,y)]
C #Terminate contour : 478 (15 points in countour 478) ,(34902 points up till contour 478)
1.090 8.807 #Coordinates of corner piont (62 ,501) along counter 479. [Format (x,y)]
1.107 8.807 #Coordinates of corner piont (63 ,501) along counter 479. [Format (x,y)]
1.125 8.807 #Coordinates of corner piont (64 ,501) along counter 479. [Format (x,y)]
1.143 8.807 #Coordinates of corner piont (65 ,501) along counter 479. [Format (x,y)]
1.160 8.807 #Coordinates of corner piont (66 ,501) along counter 479. [Format (x,y)]

1.178 8.807 #Coordinates of corner piont (67 ,501) along counter 479. [Format (x,y)]
1.195 8.807 #Coordinates of corner piont (68 ,501) along counter 479. [Format (x,y)]
6.803 8.807 #Coordinates of corner piont (387 ,501) along counter 479. [Format (x,y)]
6.820 8.807 #Coordinates of corner piont (388 ,501) along counter 479. [Format (x,y)]
6.838 8.807 #Coordinates of corner piont (389 ,501) along counter 479. [Format (x,y)]
6.855 8.807 #Coordinates of corner piont (390 ,501) along counter 479. [Format (x,y)]
6.873 8.807 #Coordinates of corner piont (391 ,501) along counter 479. [Format (x,y)]
C #Terminate contour : 479 (12 points in countour 479) ,(34914 points up till contour 479)
1.125 8.824 #Coordinates of corner piont (64 ,502) along counter 480. [Format (x,y)]
1.143 8.824 #Coordinates of corner piont (65 ,502) along counter 480. [Format (x,y)]
1.160 8.824 #Coordinates of corner piont (66 ,502) along counter 480. [Format (x,y)]
1.178 8.824 #Coordinates of corner piont (67 ,502) along counter 480. [Format (x,y)]
1.195 8.824 #Coordinates of corner piont (68 ,502) along counter 480. [Format (x,y)]
C #Terminate contour : 480 (5 points in countour 480) ,(34919 points up till contour 480)
END #End of slice 1

CHAPTER 7

7) Improvements on converting Medical image slices to rapid prototyping
The medical examination in the form of CT scan or MRI scan is done in a step-by-step manner. All these slices are evaluated separately as describe by the methods above and then added together to form the major structure.

7.1) Multi-slice STL files
Multi-slice STL files are created when each slice is converted to STL format and the added together in one STL file.

The STL files created from each slice can be created separately and then added together to investigate the structure as required. Not just the separate slices can be added together, but also extractions made of anatomical structures can be added together.

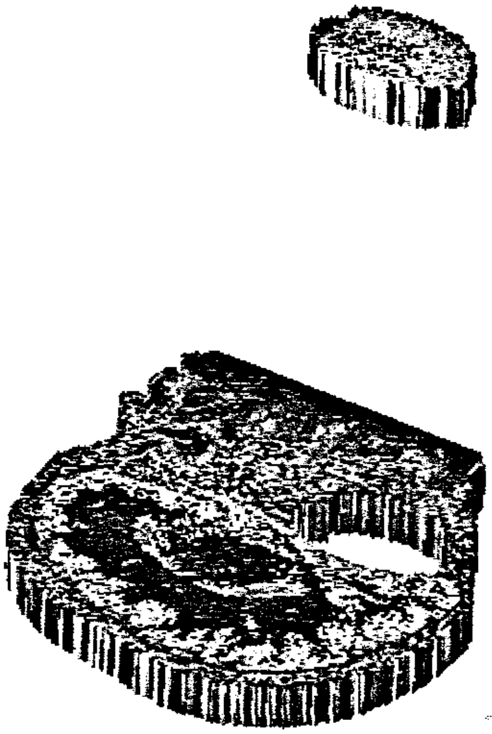


Fig 7.1: Extraction of anatomical structure of an MRI presentation (2D)

The following is an example of Multi-slice STL file where the reconstruction of the lung of a patient was created from CT Scan images.

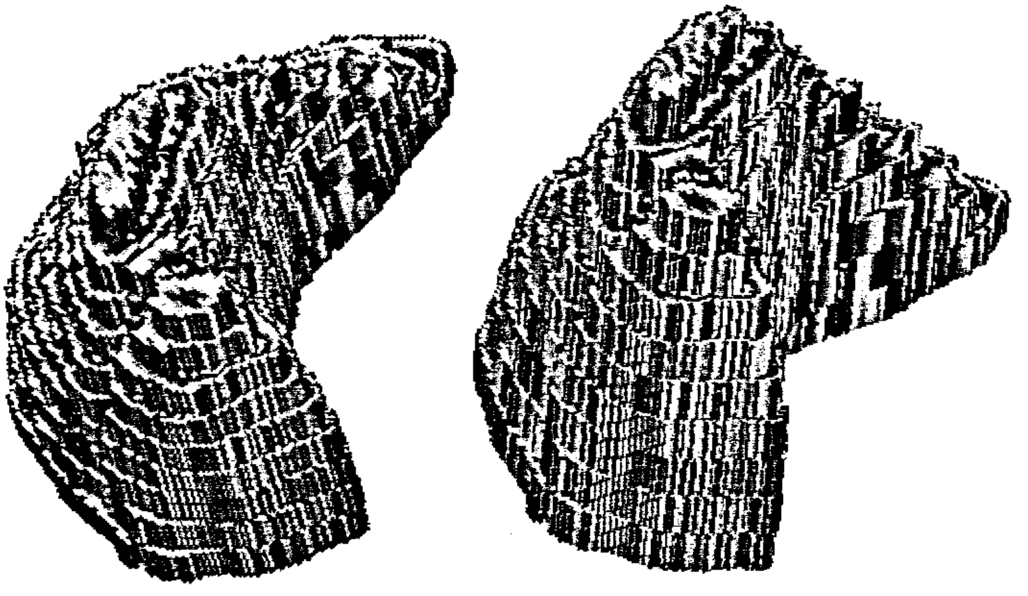


Fig 7.2: 3D reconstruction of the right lung from CT Scan images

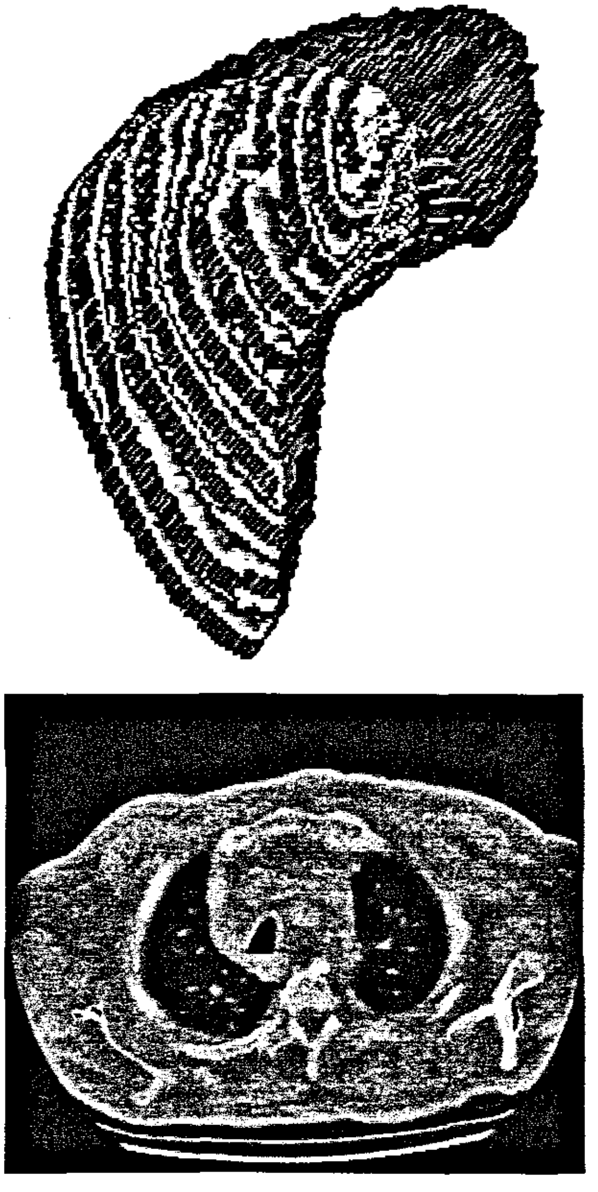


Fig 7.3: 3D reconstruction of the right lung and CT Scan images

7.2) Triangle reduction

In Rapid Prototyping the outer surface of a model is described by a set of triangles. Depending on the complexity of the model, these triangles can escalate very quickly. The amount of triangles required can easily be over 10 000 triangles. If each triangles described by a set of vertices XYZ and a normal, the resources required to display these 3D STL files is very large.

Bahattin Koc, Yawei Ma and Yuan-Shin Lee Presents a method of *Max-Fit* biarc curve fitting technique to improve the accuracy of STL files and to reduce the file size for rapid prototyping.

(http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=Published/EmeraldFullTextArticle/Articles/1560060304.html)

A method to minimise the amount of triangles, but still keep the shape and size of the model was developed.

The rule for converting the outer surface to STL file format is that the triangles have to obey the Vertex-to-Vertex rule. This method explains the reduction in the amount of triangles, but has to comply by the vertex-to-vertex rule.

Firstly a any flat surface in 3D space is detected. Now working on that specific surface, the surface area is scanned firstly by searching for the biggest surface area, and then reduces the surface area by one pixel both in length and with. For this purpose, we use a square surface area and the search if there is a continuous square area in the original surface.

When a solid area if detected that is bigger than one pixel size, that area is treated as one big surface area. Continue this process until only one surface area is over to be scanned.

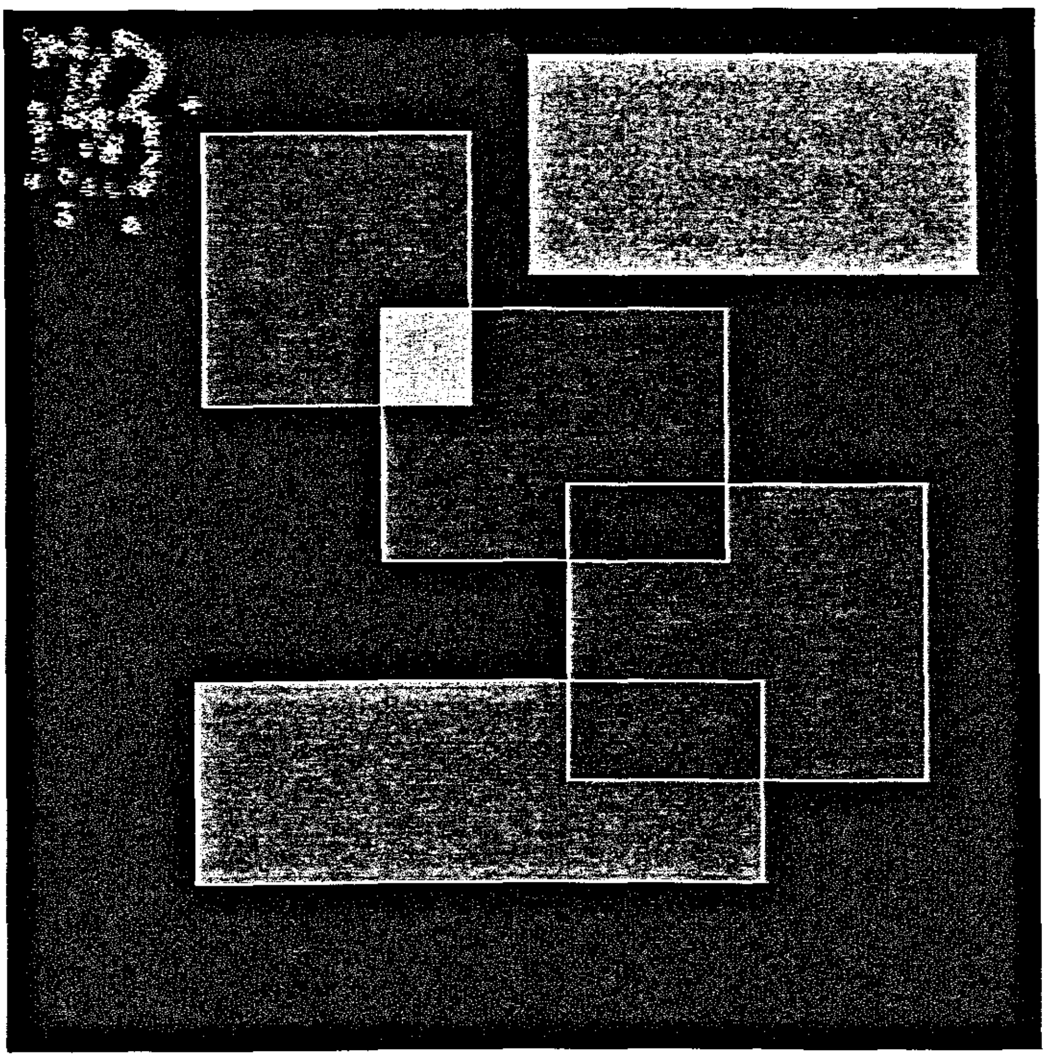


Fig 7.4: Original image before triangle reduction with squeares.

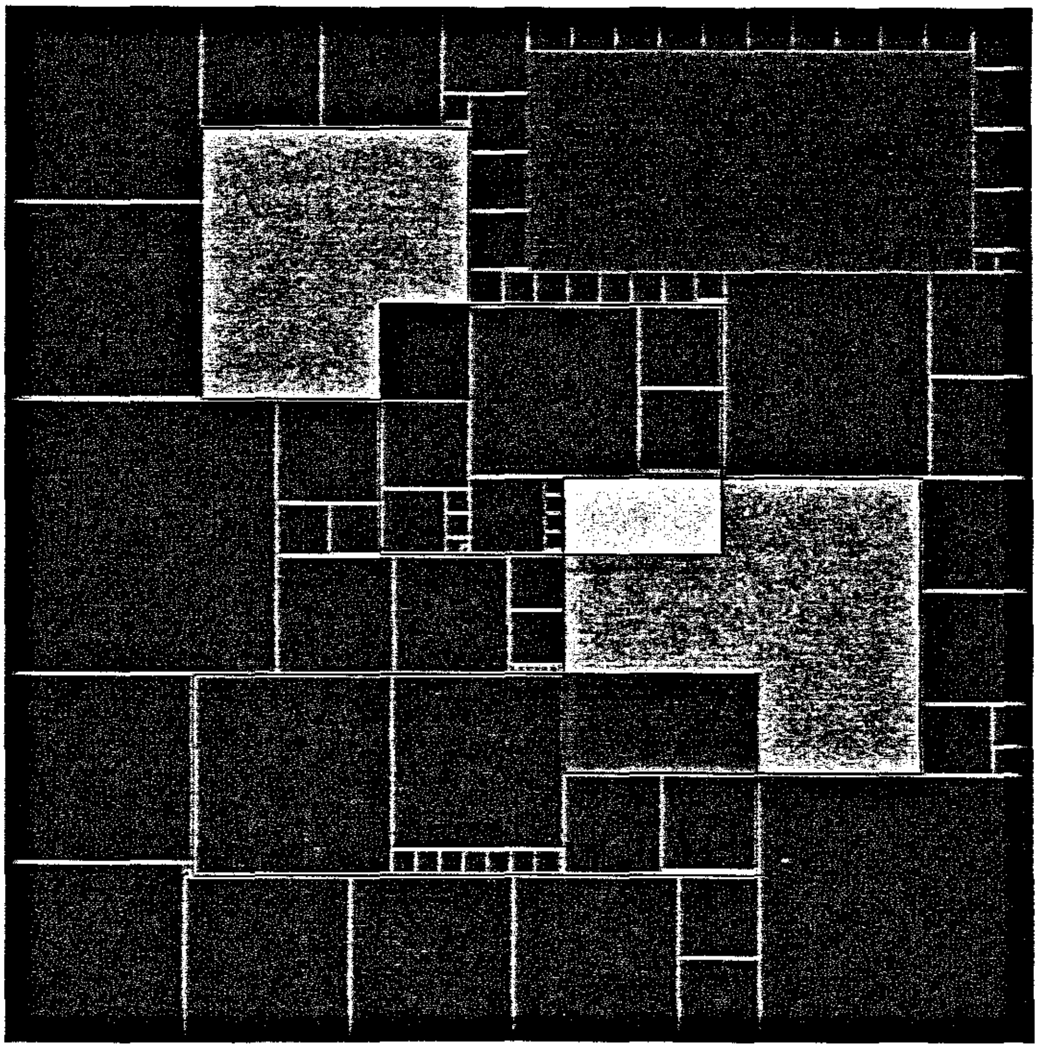


Fig 7.5: Graphical explanation of using triangle reduction with squares.

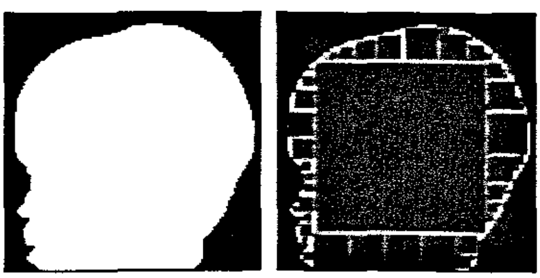


Fig 7.6: Applying triangle reduction on an MRI slice for creating the STL file

7.3) Point cloud simulation

Point clouds are created when an object is scanned with a laser scanner and XYZ information of all the points on the object is obtained. Because the outer surface of the object is demonstrated in the point cloud information, an STL representation will be useful to reconstruct the outer surface.

To reconstruct the point cloud, a specific line thickness is scanned to obtain a line in that thickness area. Now one step is moved down and the next line is obtained. By repeating the process, a set of lines is obtained that represent a mesh of the outer surface.

To give a solid outer surface, this mesh of lines and XY positions has to be connected to triangles that cover the outer surface of the structure.

To obtain the triangles the following algorithm is applied:

The line points are already obtained by scanning a certain thickness.
Detect two lines one space of each other.
From the top line detect only two points from each other.
From the bottom line detect only the points within the range of the two top points.
Complete the first triangle by connecting the first two top points and the first point of the bottom row.
Now used the right point of the top line as one triangle point and follow the bottom line two points at a time to complete the triangles.

7.4) C Code to display the point cloud in a Mesh

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <io.h>
#include <alloc.h>
#include <fcntl.h>
#include <process.h>
#include <sys\stat.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>


int X[1000]={0},Y[1000]={0},Z[1000]={0};
FILE *stream_surf,
    *STLstream;


void setup_graphics(void)
{
// request auto detection
int gdriver = DETECT, gmode, errorcode;
```

```
// initialize graphics and local variables
initgraph(&gdriver, &gmode, "C:\\lang\\tc\\bin");

// read result of initialization
errorcode = graphresult();
if (errorcode != grOk)  // an error occurred
 {
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch();
  exit(1); // terminate with an error code
 }
}

//-----
void stl_triangle(float x1,float y1,float z1,
                  float x2,float y2,float z2,
                  float x3,float y3,float z3,
                  FILE* STLstream)
{
float normal[3]={0};
unsigned int colour;

colour=0xFC00;//RED
//colour=0x83E0//GREEN;
//colour=0x801F//BLUE;

//Write co-ordinants of normal to STL file
fwrite(normal, sizeof(normal), 1, STLstream);

//Write co-ordinants of point_1 to STL file
fwrite(&x1, sizeof(x1), 1, STLstream);
fwrite(&y1, sizeof(y1), 1, STLstream);
fwrite(&z1, sizeof(z1), 1, STLstream);

//Write co-ordinants of point_2 to STL file
fwrite(&x2, sizeof(x2), 1, STLstream);
fwrite(&y2, sizeof(y2), 1, STLstream);
fwrite(&z2, sizeof(z2), 1, STLstream);

//Write co-ordinants of point_3 to STL file
fwrite(&x3, sizeof(x3), 1, STLstream);
fwrite(&y3, sizeof(y3), 1, STLstream);
fwrite(&z3, sizeof(z3), 1, STLstream);

//Write color to STL file
fwrite(&colour, sizeof(colour), 1, STLstream);
}
```

```
//----------------------------
void stl_line(float x1,float y1,float z1,
              float x2,float y2,float z2,
              FILE* STLstream)
{
stl_triangle(x1,y1,z1,
             x2,y2,z2,
             x2,y2,z2,
             STLstream);
}
//----------------------------
void stl_point(float x1,float y1,float z1,
              FILE* STLstream)
{
stl_line(x1,y1,z1,
         x1,y1+0.01,z1,
         STLstream);
}
//----------------------------
void Xlines_in_rectangle(int left, int top, int right, int bottom,int points)
{
int n,x,y,x0=0,y0=0,z0=0;

 for(x=left;x<=right;x++)
 for(y=top;y<=bottom;y++)
  for(n=0;n<points;n++)
        if(
          (X[n]==x)&&(Y[n]==y)
         )
         {
          //stl_line(x0,y0,z0,X[n],Y[n],Z[n],STLstream);
          //line(x0,y0,X[n],Y[n]);
          x0=X[n];
          y0=Y[n];
          z0=Z[n];
          x=right+1;
          y=bottom+1;
          n=points;
         }


 for(x=left;x<=right;x++)
 for(y=top;y<=bottom;y++)
  for(n=0;n<points;n++)
        if(
          (X[n]==x)&&(Y[n]==y)
         )
          {
          stl_line(x0,y0,z0,X[n],Y[n],Z[n],STLstream);
          line(x0,y0,X[n],Y[n]);
```

```
            x0=X[n];
            y0=Y[n];
            z0=Z[n];
           }
}
//---------------------------
void Ylines_in_rectangle(int left, int top, int right, int bottom,int points)
{
int n,x,y,x0=0,y0=0,z0=0;

  for(y=top;y<=bottom;y++)
  for(x=left;x<=right;x++)
   for(n=0;n<points;n++)
          if(
            (X[n]==x)&&(Y[n]==y)
           )
           {
            //stl_line(x0,y0,z0,X[n],Y[n],Z[n],STLstream);
            //line(x0,y0,X[n],Y[n]);
            x0=X[n];
            y0=Y[n];
            z0=Z[n];
            x=right+1;
            y=bottom+1;
            n=points;
           }


  for(y=top;y<=bottom;y++)
  for(x=left;x<=right;x++)
   for(n=0;n<points;n++)
          if(
            (X[n]==x)&&(Y[n]==y)
           )
           {
            stl_line(x0,y0,z0,X[n],Y[n],Z[n],STLstream);
            line(x0,y0,X[n],Y[n]);
            x0=X[n];
            y0=Y[n];
            z0=Z[n];
           }
}
//---------------------------
void main(void)
{
char stlfile[]="stlbin.stl",
    header[80]={0x20};
unsigned long  number_of_trangles;
int f1,f2,f3,n=0,m;
//---------------------------
```

```
setup_graphics();
//-----------------------------
//-----------------------------
STLstream = fopen(stlfile, "wb+");
fwrite(header, sizeof(header), 1, STLstream);
number_of_trangles=2;
fwrite(&number_of_trangles, sizeof(number_of_trangles), 1, STLstream);
//-----------------------------
//stl_triangle(1,1,0,
//          0,0,0,
//          1,0,0,
//          STLstream);
//-----------------------------
//stl_line(0,0,0,
//        1,0,0,
//        STLstream);
//-----------------------------
//stl_point(0,0,0,
         //STLstream);
//-----------------------------
if ((stream_surf = fopen("surfi.c", "r"))== NULL)
  fprintf(stderr, "Cannot open input file.\n");
else
{
 while(!feof(stream_surf))
 {
 fscanf(stream_surf,"%d %d %d ",&f1,&f2,&f3);
 X[n]=f1;
 Y[n]=f2;
 Z[n]=f3;
 //if(n<=20)
 //stl_point(X[n],Y[n],Z[n],STLstream);
n++;
}
}
//-----------------------------
//Xlines_in_rectangle(1,1,500,20,n);
//Ylines_in_rectangle(1,1,20,500,n);
//-----------------------------
for(m=0;m<=500;m=m+20)
Xlines_in_rectangle(1,1+m,500,20+m,n);

for(m=0;m<=500;m=m+20)
Ylines_in_rectangle(1+m,1,20+m,500,n);
//-----------------------------
fclose(stream_surf);
fclose(STLstream);
//-----------------------------
// clean up
   //printf("\nPress any key...");
```

```
getch();
closegraph();
}
```

7.5) Point Cloud Co-ordinants

207	10	2
14	11	13
182	11	8
259	11	10
118	14	5
163	17	16
359	18	3
83	19	7
89	20	8
150	20	9
264	21	13
19	22	18
104	22	14
255	22	4
144	23	3
333	23	4
378	23	1
146	26	5
235	26	9
360	27	8
175	31	3
392	31	2
228	33	19
393	33	9
141	34	0
237	36	11
209	37	4
37	39	4
10	42	13
75	42	3
255	42	4
272	43	10
41	44	19
96	44	5
169	44	14
171	46	7
373	46	0
134	47	6
301	48	15
117	50	1
256	50	0
286	51	14
352	56	15
270	60	17
24	62	17
223	62	9
236	63	3
265	63	5
254	66	13

68	68	11
193	68	18
110	70	18
138	70	1
308	71	13
313	71	5
287	75	1
161	80	19
206	80	17
192	82	10
264	83	7
162	85	19
244	86	1
88	88	16
224	88	15
362	88	16
81	91	16
244	93	5
127	94	7
389	95	0
111	96	19
152	96	6
76	97	9
292	97	0
179	98	12
289	99	19
354	99	15
133	101	8
190	104	9
173	106	0
205	106	19
94	109	1
248	111	6
257	111	3
282	112	6
348	113	17
357	115	12
88	116	16
372	116	9
324	117	9
41	118	15
251	121	0
351	121	2
244	122	10
13	123	3
207	124	12
191	127	15
260	128	15
253	129	16
282	133	6

274	134	0
32	139	19
340	141	4
77	142	11
127	143	1
67	144	0
104	144	10
127	144	11
187	144	12
282	145	5
235	147	18
271	148	8
268	149	10
379	153	0
35	156	12
219	158	7
56	161	13
391	161	12
233	163	14
194	164	2
319	165	6
141	170	5
152	171	18
282	171	10
55	172	7
247	172	3
10	173	3
107	174	13
151	174	7
91	176	5
375	177	0
57	178	18
81	179	19
31	180	4
50	180	2
97	180	14
111	181	15
147	182	16
257	182	2
308	183	15
119	187	7
288	188	3
368	188	5
163	189	10
322	193	12
13	195	9
159	196	16
229	198	10
222	201	3
193	203	18

279	204	9
382	205	1
199	206	16
228	207	2
327	209	1
110	211	14
110	212	3
305	213	7
348	213	5
163	214	6
168	214	1
231	214	15
379	216	9
325	217	3
60	218	16
194	222	14
78	225	11
108	227	7
255	229	1
79	231	6
275	232	8
60	233	12
178	234	1
398	234	5
267	235	15
106	237	18
230	238	10
277	238	10
80	239	12
269	239	11
160	242	17
265	243	2
61	245	3
398	245	19
169	246	14
274	249	1
333	250	7
190	251	2
106	252	10
219	261	4
333	261	12
67	262	6
128	264	17
124	265	19
119	267	8
225	268	17
354	270	4
377	271	3
272	274	6
319	275	9

293	276	18
58	278	12
162	281	12
97	287	5
371	287	13
52	288	12
53	289	6
55	290	12
260	293	18
269	293	8
55	294	18
267	296	9
311	296	19
344	296	4
391	296	17
128	297	10
229	298	5
374	299	6
348	300	3
213	302	5
296	302	9
372	304	10
349	305	3
151	312	17
248	312	7
284	315	14
258	318	9
126	322	19
388	322	17
114	325	17
243	325	16
130	331	10
78	332	18
209	332	18
160	333	6
167	334	10
379	334	6
76	336	14
399	337	16
216	339	19
357	339	4
218	345	9
327	345	0
391	345	19
380	347	14
101	353	17
347	353	9
211	354	15
302	355	6
149	357	2

151	357	4
322	358	8
374	358	3
221	361	12
306	362	11
141	364	10
145	364	10
269	364	17
313	365	11
380	365	1
255	367	10
260	367	16
133	369	5
257	369	4
97	373	3
398	376	12
27	378	5
400	378	5
305	379	2
279	380	18
266	381	15
45	384	14
151	385	4
15	386	7
259	389	8
290	389	14
112	390	5
255	391	5
383	391	12
38	392	6
341	393	3
36	394	18
372	395	9
111	399	19

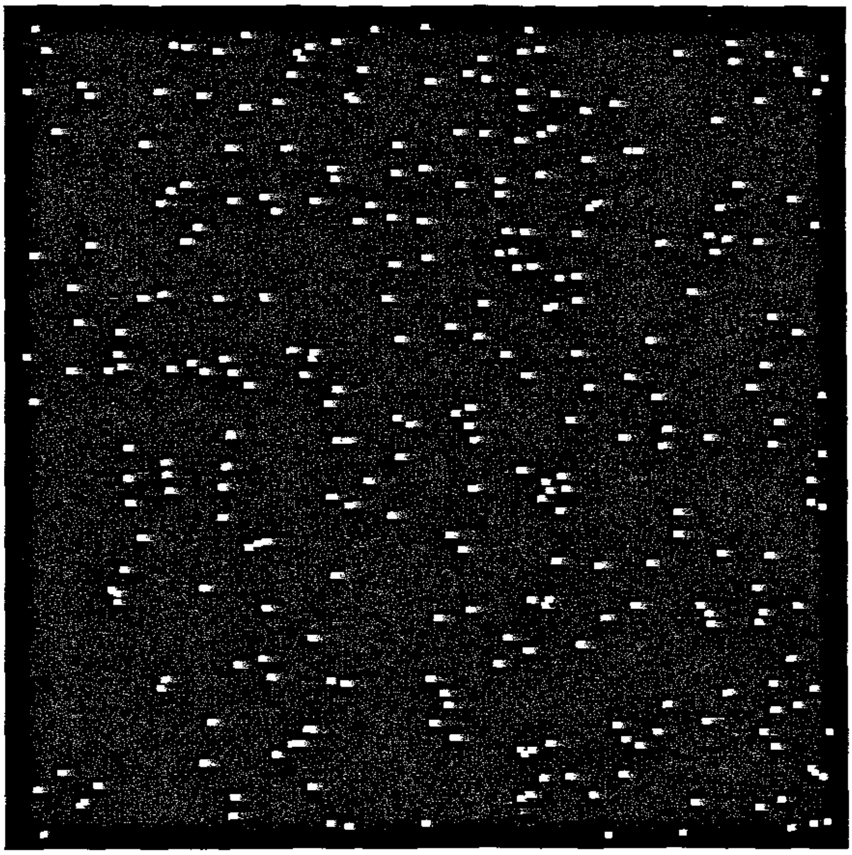


Fig 7.7: Point cloud created with different XYZ co-ordinants.

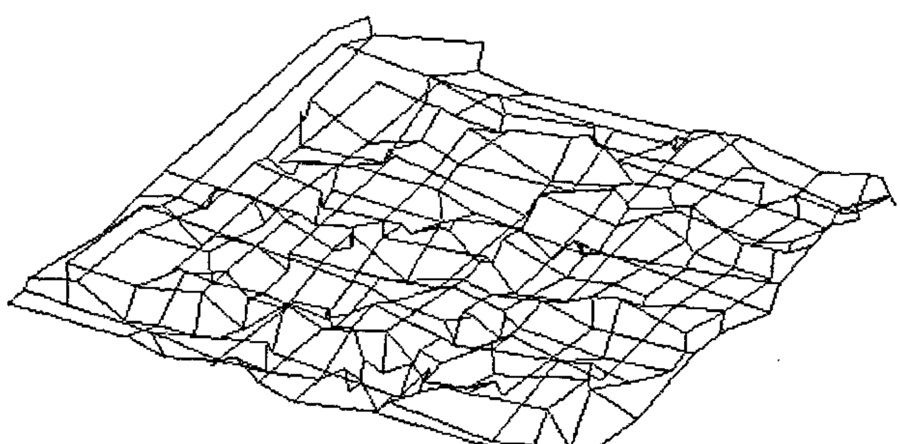


Fig 7.8: Mesh created by connecting the points together in 3D plane

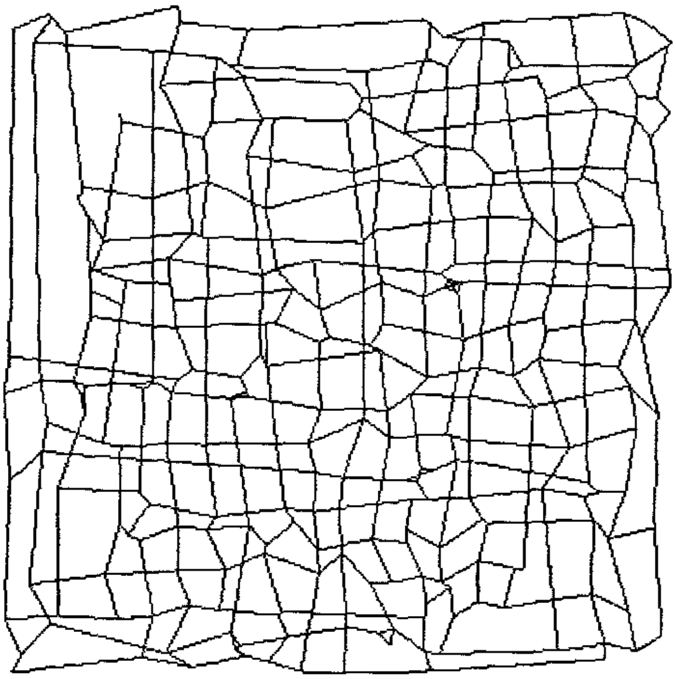


Fig 7.8: Mesh created by connecting the points together in XY plane

7.6) Interpolation

MRI and CT-Scan images are created from different levels on the body. To recreate the orgran of interest, anatomical extraction, external edge detection and edge detection is required (methods for anatomical extraction, external edge detection and edge detection is previously explained).

To comlete the structure of interest, the edges of these image slices must be added together to re-create the outer surface of the structure.

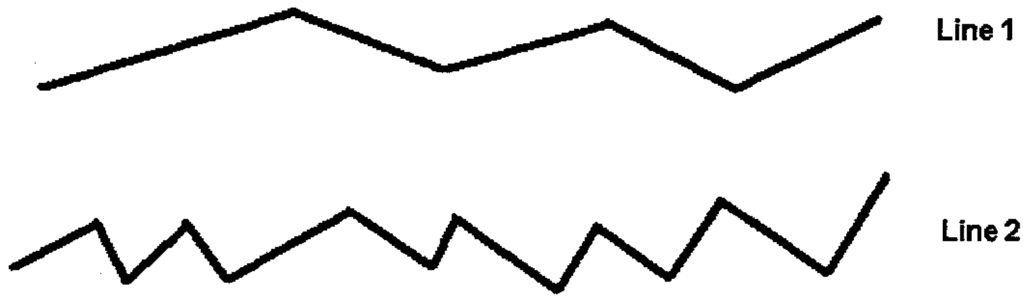
To recreate the outer surface of the structure, the points on the external surface of the organ is obtained. This method is similar to the point cload simulation if viewed from a specific direction. To obtain the completed image of the organ, the external surfaces are added together as in the outer rim edge detection.

7.6.1) Algorithm for Interpolation

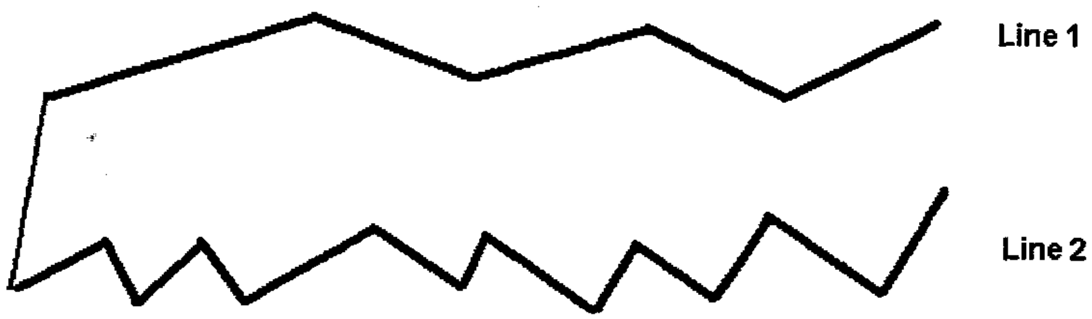
1) Obtain the points in one line of a specific part of the structure.
2) Obtain only two lines next to each other, line 1 and line 2.
3) From the left point of line 1, obtain the first and the second point on line 1.
4) From line 2, obtain all the points between point 1 and point 2 of line 1.
5) Connect the 1st point of line 1 to the first point of line 2.
6) Connect the 1st point of line 2 to the 2nd point of line 2.
7) Connect the 1st point of line 2 to the 2nd point of line 1.
8) This will create the 1s triangle described by the surface of line 1 and line 2.
9) Now connect the 2nd point of line 2 to the previous point in line 2.
10) Also connect the 2nd point in line 2 to the 2nd point in line 1.
11) Repeat the process until all points in line 2 are connected to the 2nd point in line 1.
12) This will create a surface between line 1 and line 2.

7.6.2) Graphical Representation for Interpolation

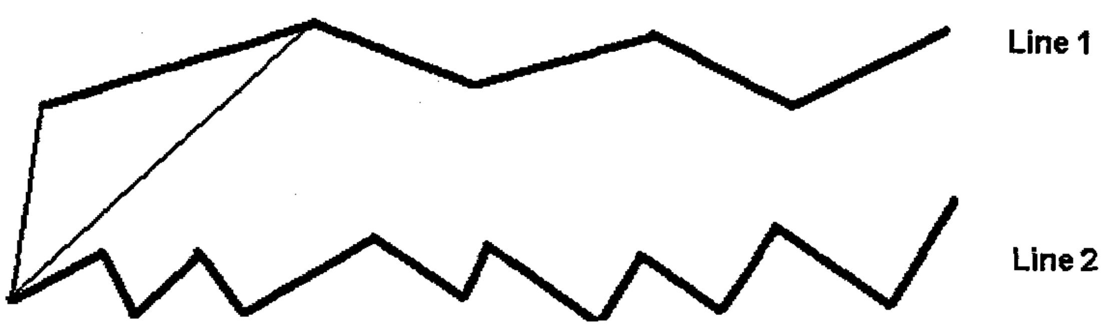
1) Obtain only two lines next to each other, line 1 and line 2.



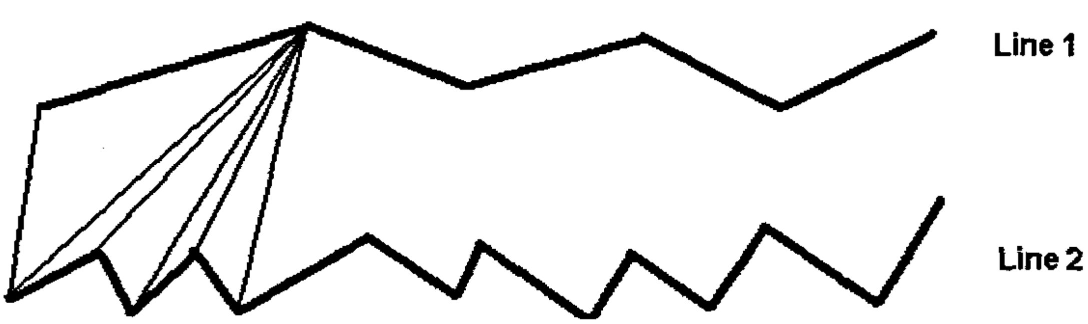
2) Connect the 1st point of line 1 to the first point of line 2



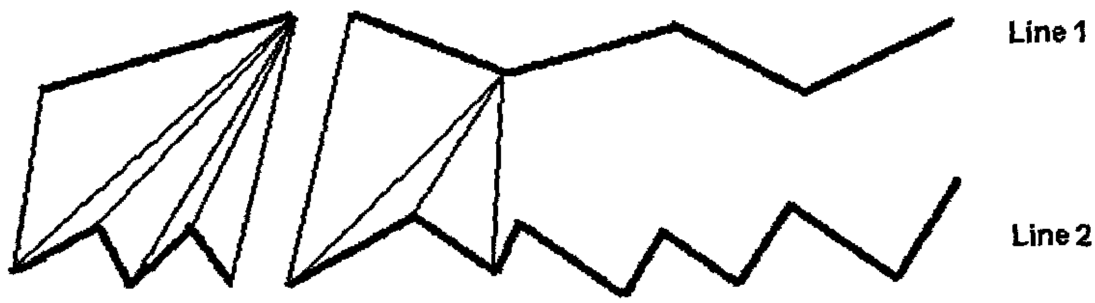
3) Connect the 1st point of line 2 to the 2nd point of line 2.



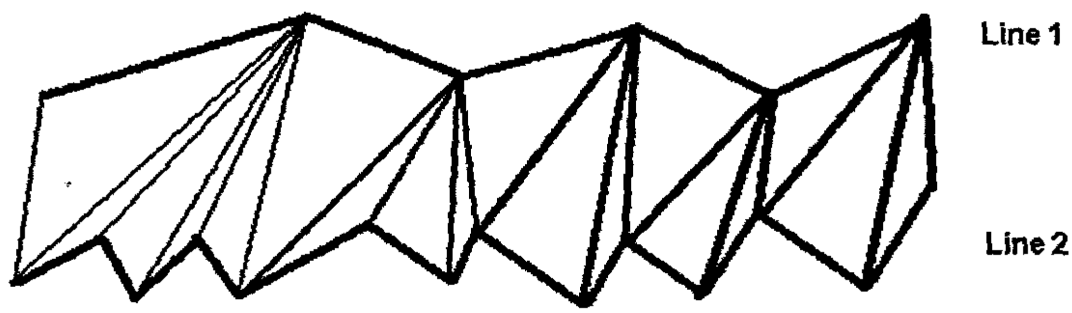
4) Repeat the process until all points in line 2 are connected to the 2nd point in line 1.



5) Segment 1 and segment 2



6) This will create a surface between line 1 and line 2.



CHAPTER 8

8) Theoretical method for detecting the 3D point in space with one X-ray exposure

8.1) Background to 3D detection in space

To detect the position of an artifact in space with an X-ray, we normally do two exposures. In these cases we either use two X-ray exposures by moving the tube or we move a patient to obtain a 90-degree view from the previous one. In the movement of the patient or the X-ray tube, we are moving the position of the artifact from the Anterior Posterior (AP) to Lateral (Lat) position. This will expose the radiographer and patient to radiation and to limit the amount of radiation for detection of an artifact, the method to detect the position in space with only one X-ray exposure was developed.

The work done on detecting the point in 3D is in the beginning and although planning was done and seeking new methods, improvements can be made to deliver better quality of position detection in space.

8.2) Background on detecting the 3D point in space with one X-ray exposure

In review of the literature, the following write up about X-Ray holograms were mentioned where it is stated that if such holograms become possible, one exposure could provide a radiographic image which the radiologist or clinician could examine from different angles presenting a different radiographic presentation to the observer. [1] Although this aspect is in the beginning stage, the techniques developed can be used for position detection of artifacts.

8.3) Stereoscopic vision

Humans are capable of stereoscopic vision, which is fundamental to depth perception; because they are able to focus both eyes on a single object which (see figure 1 and figure 2). Stereoscopic vision can be described in terms of the vision process involved in the use of a stereoscope, which presents an image from two slightly different angles so that the eyes can merge them into a single image in three dimensions. A horopter is the projection of the points in the visual field corresponding to the aggregate of points registering on the two retinas.

The figure below shows L and R representing the two eyes and SS a line (the "horopter") [Encarta] drawn through the point A where the optic axes LA and RA intersect, and parallel to a line joining the two eyes L and R. The point A is seen in corresponding points of the two eyes, axially situated. Two points r and l, however, may be so placed-either in the plane of the horopter or outside it-that the two eyes together perceive the points r and l as one point, B (in Fig. 8.1 point B is nearer to the eye, and in Fig. 9.2 farther from the eye than the horopter SS itself.) Now, in Fig. 1, a diagram is made representing l and A, and another representing r and A. Then suppose the former is laid before the left eye and the latter before the right eye. The two optic axes are thus made to converge, so that the image of A is formed in corresponding points in the two eyes. As a result, the point's l and r will appear to blend into one, situated either nearer the eye than A or further from it. This explains the action of the stereoscope and also the "pseudoscopic" effect produced when the pictures are reversed.

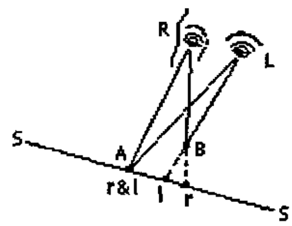


Figure 8.1 Viewing of an object (No: 1)

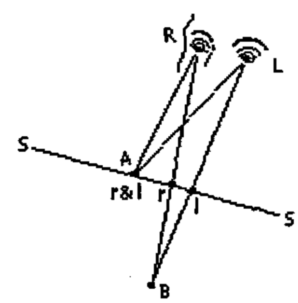


Figure 8.2 Viewing of an object (No: 2). Using unfocused section of the image as well

8.4) Image of X-ray beam entering patient

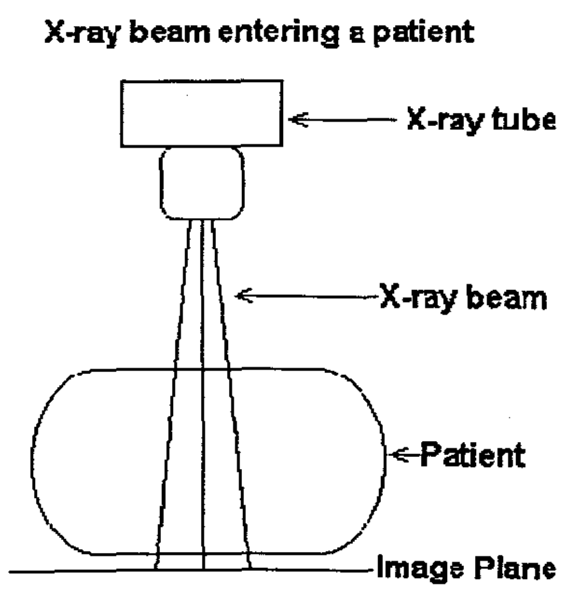


Figure 8.3: X-ray set up for patient examination

8.5) Examination Anterior Posterior (AP) and Lateral (Lat) position using two exposures

To detect the position in space we normally do two exposures. [McMullan J.T]

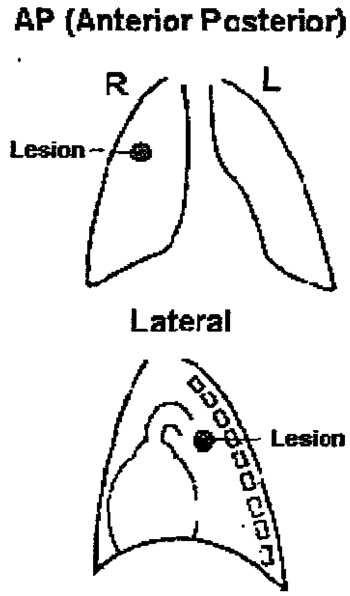


Figure 8.4: Example of localising a lesion in an X-ray examination of the chest

8.6) Method to detect the position in space with only one exposure
Two techniques have been developed to detect the position of an artifact in 3D space.

1) Graphical method using the enlargement technique and the penumbra of the image.
2) Theoretical method using the characteristics of the X-ray image system.

8.7) Definitions of point focus
Point focus
This is the ideal situation in which the source of the X-rays is a point focus. See Figure (Point source of X-ray)

X-ray Image production from a point source

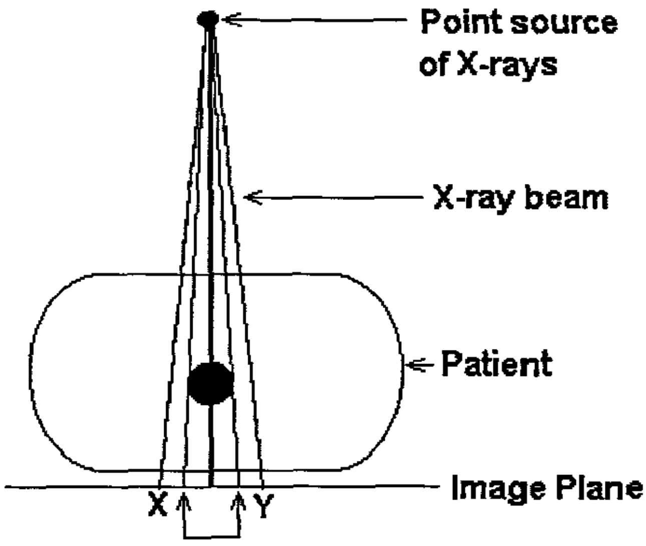


Figure 8.5: X-ray source from a point focus

8.7) Definitions of finite focus
Finite focus
The X-ray source has a finite area.
See Figure (Finite source of X-ray)

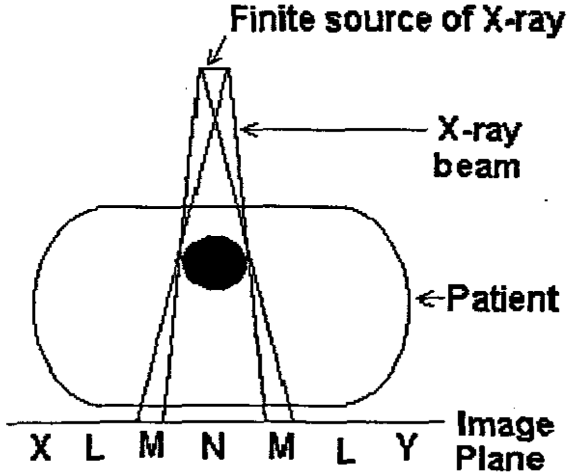


Figure 8.6: X-ray source from a finite area

8.9) Image from Finite area

In figure 6 consider the image production from a source of finite area. Rays traced from the edges of the source divide the image plane into regions L, M and N were M is the penumbra.

8.10) Penumbra

This is a partial shadow when the image is created and is called the penumbra. See Figure (Geometry of image formation)

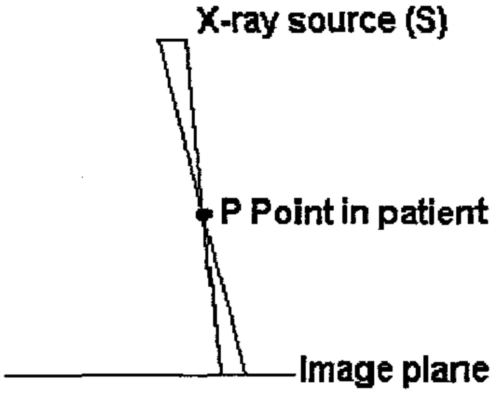


Figure 8.7: Penumbra produced from a point

8.11) Technique 1): Graphical method using the enlargement technique.

In this case we use two X-ray image planes as positioned in figure below.

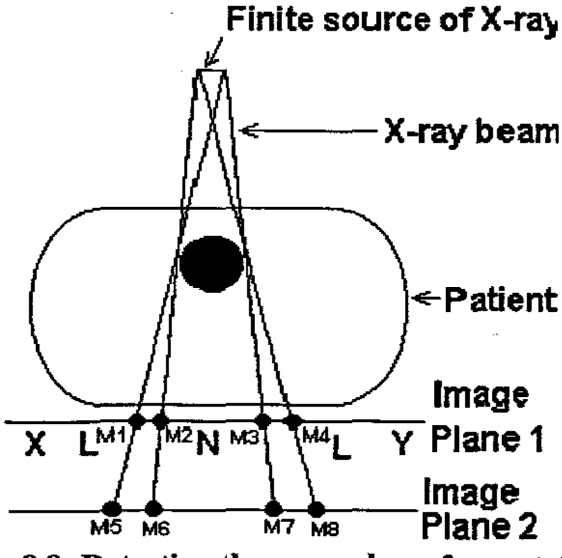


Figure 8.8: Detecting the penumbra of an artefact

As can see only one exposure will produce two different images in size and using this enlargement technique the position of the artifact can be determined.

The image on image plane 1 can be projected at another position as from image two with a different setup.

If viewed from the sides, two different images are obtained with slightly different co-ordinates.

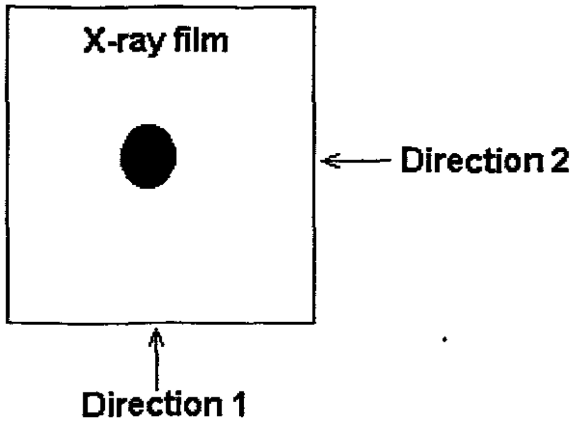


Figure 8.9: X-ray image view from the above

The image as projected from the side can be viewed in two directions, namely direction 1 (see on figure 8.9) and direction 2 (see on figure 8. 9). The penumbra on image plane 1 and image plane 2 are projected with different co-ordinates.

8.12) Lets examine the images as viewed from direction 1

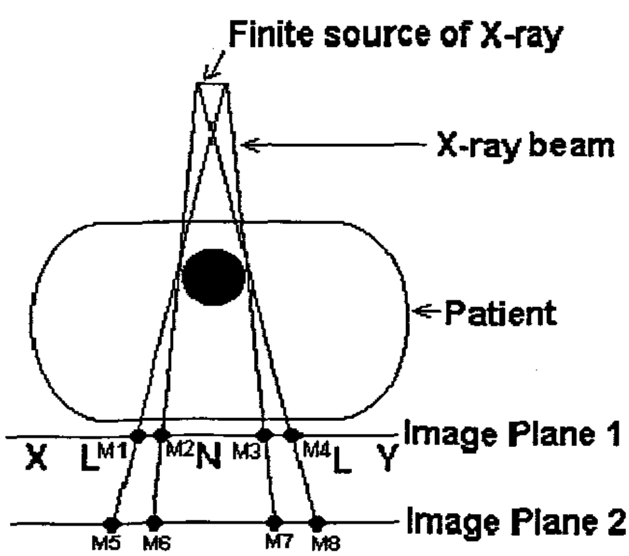


Figure 8.10: Using the penumbra to detect the position (Unfocused section)

The distance between Image Plane 1 and Image Plane 2 is a set value. With this known distance and the points M1, M2, M3, M4, M5, M6, M7, M8 marked and determined, there are eight co-ordinates (four per image plane) and this information is used to determine the position.

Now lets extend line M5 through M1 and line M2 through M6. The position of the left point of the artifact is given by the intersection of lines M5, M1 and line M6,

M2 on figure 10. By doing the same with M7, M3 and M8 and M4 the right hand side is determined.
The same process can be followed when viewing the image from direction 2. Certainly a practical consideration must be considered and this aspect gives an explanation for 3D position detection with one exposure.

8.13) Detecting a single point in space
Lets consider figure 8.11 below, which shows a finite source of X-ray (S), a point (P) in an object, and the two image planes [I1] and [I2].

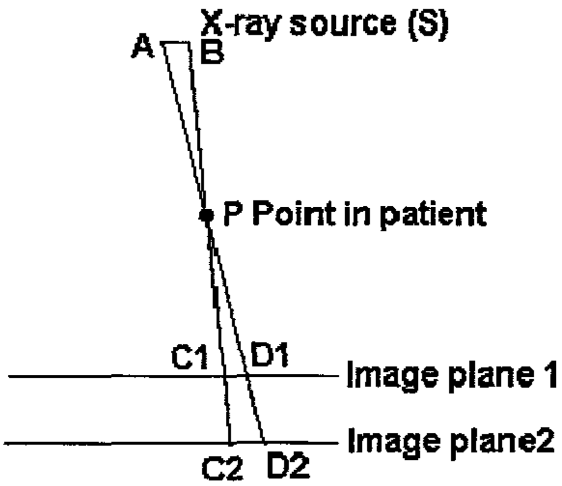


Figure 8.11: Detecting a single point in space

At the image planes the images of P is represented by C1, D1 and C2, D2. The image P should be a point, but because of the finite size of the X-ray source it is imaged as a disk whose diameter is C1, D1 and C2, D2.

Since the distance between image plane 1 and image plane 2 is a set distance, the co-ordinates of C1, D1, C2, D2 can be determined. Using the graphical method lines C1 and C2 can be extended and also lines D1 and D2. This will provide two lines that will cross at point P.

For the above methods two image planes are required, but with the next method only one-image plane is needed.

8.14) Technique 2): Theoretical method using the characteristics of the X-ray System
Lets consider the image below with only on image plane. Here certain characteristics of the system must be known, namely the focus size as well as the position of the focus in the system.

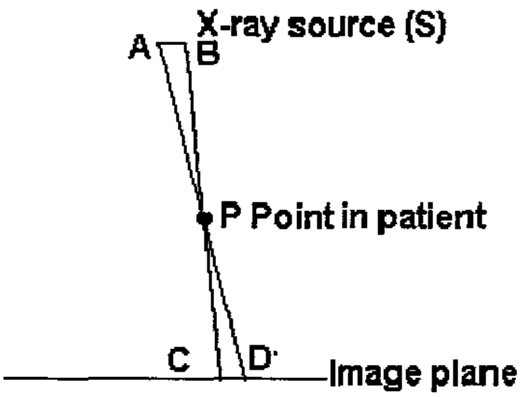


Figure 12.11: Detecting a single point in space one plane

The penumbra is also used, but the position in space is calculated using the characteristics of the image system. The formation of the penumbra is a consequence of the finite size of the X-ray source. Consider the figure below which shows a finite source of X-ray (S), a point P, in an object and the image plane (I) which would normally be accompanied by a film, screen cassette or image intensifier input phosphor. At the image plane, CD represents the image of P. The image P should be a point, but because of the finite size of the X-ray source, it is imaged as a disk whose diameter is CD. Triangles ABP and CDP are geometrically similar. Therefore AB/CD=BP/CP. Re-arranging this gives CD=(AB X CP)/BP.

Lets get the co-ordinates of the positions of A, B, C and D. Since the position of C and D is projected on the image plane, their co-ordinates can be determined. If the size of the focus (S) is known and it's position the co-ordinates of A, and B must be determined. Now we have the co-ordinates of A, B, C and D and we form a line from A to D and from B to C. By getting the mathematical formula of these lines and at their crossing point, will give the position of point P.

By using the same method as previously explained, the position of point P can be calculated with only one image plane and only one X-ray exposure by using the characteristics of the system and references.

8.15) Point Spread Function

The tissues displayed by X-ray imaging consist of a number of minute points of anatomical detail. A point structure in an object should produce a point image. In practice the image of a point will be unsharp and will appear as a small blurred disk. This will make the recognizing of the penumbra very difficult.

8.16) Limitations to consider

One of the major limitations is the measurement of the penumbra. In many cases the penumbra will be very small due to the long source to object distance and small size of the focal spot. [Ball J, Price T] The size of the penumbra is dependent on the position of the object from the X-ray source. Another limitation is the fact than most of the objects blocks X-ray partially only (Radio lucent material). The point-spread function will also contribute to the unsharpened of the point in space.

8.17) Concluding Remarks

This is a theoretical method of determining the height of a point in space. Here are two methods of determining the Z co-ordinate by using only one X-ray exposure. Certainly the technique and methods can be refined to obtain better results. By having these two methods, they can be used as a checking method for each other.

CHAPTER 9

9) Future work

Additional improvements can be added to deliver a much better 3D product. This method delivers the 3D model in color. One of the improvements is minimize the time from reading in the DICOM image to produce a 3D model.

9.1) Increase the time for converting the process

To increase the time to deliver a 3D model, automation of the process can be introduced. For this the user only has to identify the DICOM series. After the DICOM series has been identified, the user can identify a threshold range, for example between 120 and 130, and filtering of all the slices will be done automatically. After the filtering process, the conversion to STL file will be done automatically, and the STL file will also be created.

9.2) Non-invasive autopsies

9.2.1) Introduction to Virtual autopsies

The application of human organ reconstruction can be used in the forensic field as described by CATHERINE NANGINI at http://www.jsonline.com/alive/news/aug03/164448.asp?format=print.

Virtual autopsy may be next innovation in forensic field. Medical imaging useful in solving crime, but cost, training debated

In 1999, a woman's dismembered body - minus most of her skin, her ears and nose - was found floating in trash bags in the Wisconsin River.

Police had a grisly murder case on their hands with little to go on - no identity, no missing person alert, no witness.

But with the help of imaging technology developed for the living, the victim was identified as Mwivano Kupaza, and the killer, her cousin Peter Kupaza, was sent to prison.

Medical imaging may not be used on the next episode of "CSI: Miami," but some scientists believe it is the next major advance in forensic science.

Mwivano Kupaza's skull and soft tissues could not be further examined for clues because the skull was evidence, according to Leslie Eisenberg, forensic anthropologist and archaeologist at the Wisconsin Historical Society.

Instead, the remains were imaged using computed tomography (CT), a three-dimensional X-ray technique, and the images were sent to the Milwaukee School of Engineering's Rapid Prototyping Center. Based on the images, very thin strips of paper were used to build "the exact replica of the victim's skull without the soft tissue," Eisenberg said.

After realistic facial features were added to the model, posters were made and distributed. Finally, someone recognized Mwivano Kupaza from Tanzania.

The Kupaza case is just one example of how researchers can use imaging to help solve a crime.

William Oliver, a forensic pathologist at the Armed Forces Institute of Pathology in Washington, D.C., uses MRI to examine the eyes of babies who have been shaken to death.

The head will flop backward and forward, Oliver said, causing bleeding at the back of the eye. It's difficult to determine whether a baby has died because of shaking or other trauma such as a fall, he said. But Oliver believes that the amount and pattern of bleeding in the eye provides an important clue.

Physically preparing the eye for dissection is very labor-intensive and takes months of work per eye, he said. By using MRI as a high-resolution microscope, the exact amount of blood in the eye can be measured, and more quickly. With enough cases, Oliver can build a database with his findings, ultimately making it easier to identify damage caused by shaking.

MRI can estimate the angles and forces used in blunt force trauma such as motor vehicle accidents, according to work shown at the International Society for Magnetic Resonance Imaging last month in Toronto.

Victims of bioterrorism or infectious diseases such as severe acute respiratory syndrome could also be good candidates for the virtual technique because it minimizes the pathologist's exposure to airborne agents, said Michael Thali of the University of Berne in Switzerland. Thali is project manager of a five-year-old virtual autopsy - "virtopsy" - program, whose research was presented in Toronto.

9.2.2) Virtopsy goes beyond the realm of crime.

For most people, the idea of an autopsy is uncomfortable at best, especially in situations such as the death of an infant. Some religious groups forbid autopsy outside of legal necessity.

Without making a single cut, imaging can slice the body in fine detail - and produce a 3-D image - in a matter of minutes.

CT scans can show damage to the face, typically untouched during an autopsy out of consideration for the family, and in areas that might otherwise be overlooked.

"MRI can diagnose injuries that are not necessarily . . . anticipated," said Milwaukee County Medical Examiner Jeffrey Jentzen.

A breakdown of unexplained deaths in Milwaukee County shows how imaging could be applied.

In Milwaukee County, 55% of the 1,876 deaths investigated by the medical examiner's office in 2001 required autopsies to determine the cause of death, records show. Of these, 64% were classified as natural, mostly due to heart disease. Accidents made up 16% of deaths, primarily from falls, followed by homicide (8%), suicide (7%) and motor vehicle accidents (3%).

MRI is very good at detecting heart failure, said Rob Bisset, a radiologist at North Manchester General Hospital in England, but blockages in small vessels are still too small to be seen with current imaging technology. Bisset also consults for a private MRI facility funded by the Jewish community to perform virtual autopsies.

Thali said developments in MRI scanning show promise for detecting heart muscle damage during heart attacks. If doctors knew which area of the heart was affected, they could be guided right to that area, like using a map to navigate an unfamiliar town.

9.2.3) Expensive and specialized equipment

Not everyone is jumping on the virtual autopsy bandwagon. The main problems are money and training.

"CT scanning and MRI scanning comes at a cost," said Michael Bell, the deputy chief medical examiner in Broward County, Fla.

They also have their limitations. Bisset said that MRI cannot replace the autopsy in cases involving drugs, diabetes and leukemia, for example. If combined with the real exam, virtual techniques could double the cost of an autopsy

Interpretation of medical images requires either a radiologist trained in forensic science or a forensic scientist trained in radiology. Either way, collaborations between radiologist and pathologists need to be established. Forcing scientists to learn new ways in a traditional field such as forensic science is an uphill battle, Thali said.

Pathologists are less willing to do autopsies when they could make more money doing lab work for living patients, said Michael Steir, an assistant professor of forensic pathology at the University of Wisconsin-Madison.

For now, imaging technology can be a valid adjunct to the standard autopsy.

"I think complementary would be an excellent word to use," Jentzen said. However, it is not clear who would be willing to pay for either the equipment itself - about a million dollars for an MRI scanner - or for renting machines of a private clinic, which could run a couple thousand dollars per scan. It's still early to tell.
"Call me again in 10 years," Thali said. "I promise you that the field will change."
In this process 3D visualization of the organs are created, and during an autopsies the organs are investigated. Since CT and MRI images are used, a CT scan or an MRI scan can be done on a diseased body and these images can be used for reconstruction of the organs. Although this process is in the beginning stage, for Non-Invasive post mortems the method designed can be used for reconstruction.

9.3) Display of interaction of cancer and normal anatomical structures

If a cancer lesion of an organ is visualized the 3D visualization of the lesion can be demonstrated. In cases where the cancer lesion infiltrates normal anatomy, it is useful to demonstrate how the cancer lesion infiltrates the normal anatomy. In the process developed, both normal and anatomy can be visualized in 3D. Here this method can be used to 3D visualization of the abnormal pathology separately of normal anatomy as well as demonstrate both on the single 3D presentation.

9.4) Mechanical structure and interaction of anatomical structures

Here we can use the model as prescribed in Pyhton code describing the spring mass system with each mechanical structure with its own characteristics.

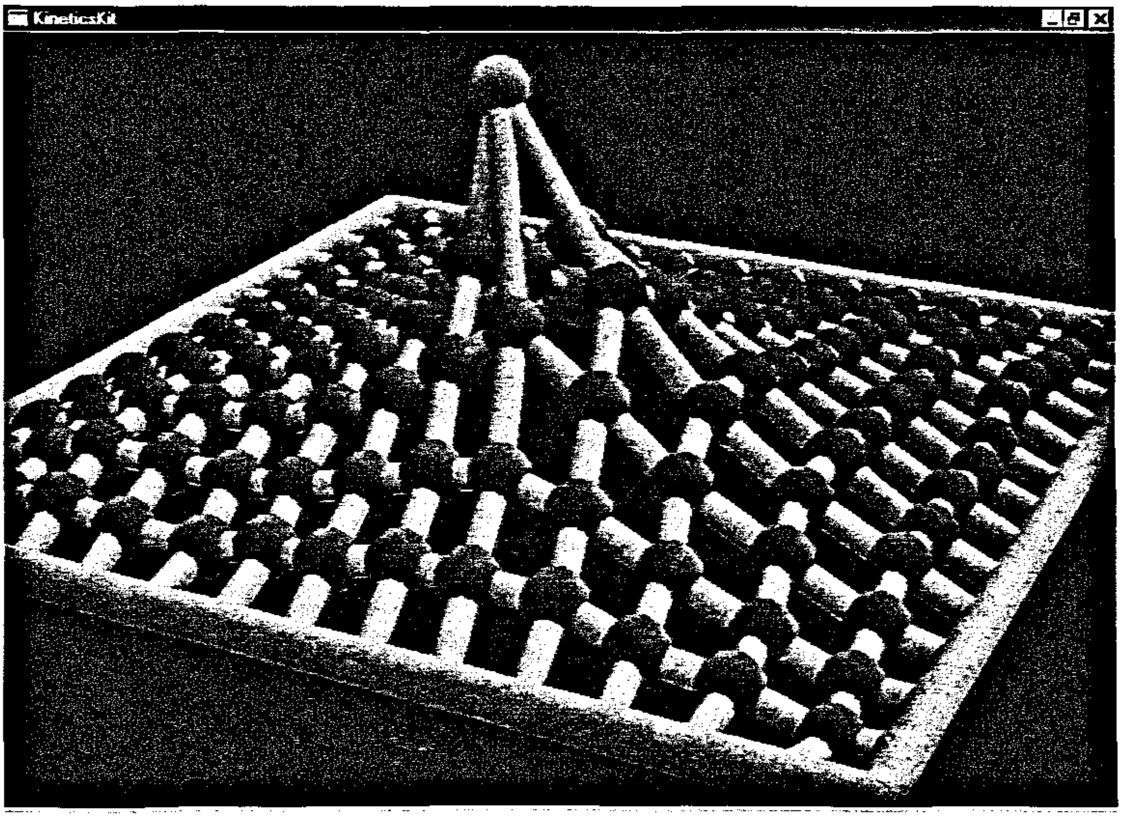


Fig 9.1) Python model representation to introduce mechanical structure on medical MRI image

9.5) Communication improvements

After the image has been reconstructed, the results and presentation of the image need to discussed and displayed to other uses. Since CT scan used X-ray radiation, a new method to detect the 3D position with only one X-ray exposure has been developed.

9.5.1) Remote medical image view

This is to transmit the medical image from one PC to another. On the server PC the medical officer will have the X-ray Image on screen. Communications between Community Health Centres and those at other Hospitals can be simplified. One person can point on the X-ray Image the area of interest and the other person will be able to see the pointer.

For this a PC with a modem and a landline is required. Internet connection is can also be used. Cost is minimized since only a landline is required.

Setup Dial-Up Network and run the VNC software. The viewer at the Hospital will have a visual of the same X-ray Image that is at the other hospital. This method can also be used to save the X-ray Image for 5 to 10 years.

9.5.2) Reasons for remote viewing of medical Images

The patient condition is displayed better. By viewing the medical image the discussion can be much better and both can point to the area of interest on the X-Ray image.

9.5.3) What does each party have for remote image view?

Both will have the same medical image in front of them on screen.
The image can be controlled on both sides making it lighter or darker or zoom to an area of interest.

9.5.4) Equipment needed

You only need a PC with a modem, Internet connection.

9.5.5) Setup for remote medical image view

You only need to setup your PC with Dial-Up networking to connect to a remote PC with a telephone line. The protocol must be TCP/IP. You run the remote viewing software.
Run the server on the one PC and the viewer on the other.

CHAPTER 10

10) Conclusion

The work done and research have developed a method for the reading of DICOM medical files and display these information. The algorithm developed in the work done can display these DICOM files that can be CT Scan or MRI.

After the DICOM image has been read and displayed, the anatomical structures of interest has been extracted using the filtering method where the gray scale value of a specific area or organ is inserted and the information is extracted using the filtering method.

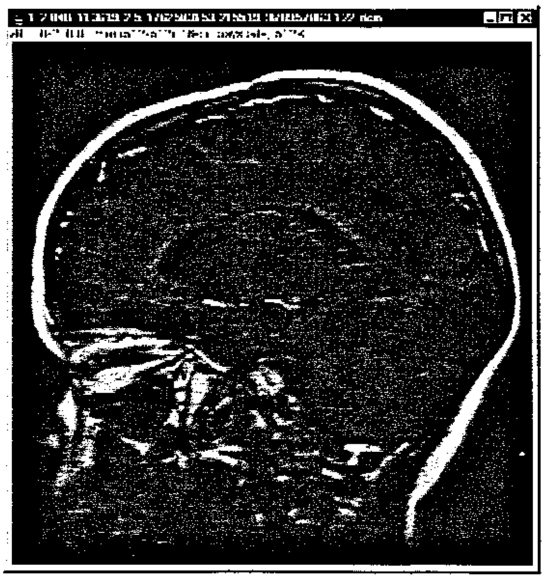


Fig 10.1: Resulted image after applying the DICOM read algorithm

10.1) Visualization of arterial structures

Arteries of the heart and the brain can be damaged, dilated or present abnormally especially in a stroke or a heart attack. Visualization of these complex arteries can be done using the method described above. For this the arteries need to be identified and tracking of those arteries can be done through the slices. Not just only small artery like the coronary arteries in the heart, but also major arteries like the Aorta can be visualized.

10.1.1) Interpolation

The reconstruction of the structure can be completed using interpolation algorithm. After the information is extracted from the MRI or CT-Scan slices, the surface of the structure is reconstructed with triangles.

An algorithm to reconstruct the outer surface was developed.

10.2) Methods developed

10.2.1) Create an SLC file

An SLC file from the structure was created. The pixel information and extraction of anatomical structures was developed to create the SLC file.

```
Slice V1        #
FILE(c:\ScullMRIPlane1.slc) # filename with full path
SLICES()        #
Z 1.000 0.010 #
4.799 0.404     #Coordinates of corner piont (273 ,23 ) along counter 1  .
[Format (x,y)]
C       #Terminate contour : 1   ( 1    point in countour 1  ) ,( 1    points up till
contour 1  )
4.852 0.422     #Coordinates of corner piont (276 ,24 ) along counter 2  .
[Format (x,y)]
4.869 0.422     #Coordinates of corner piont (277 ,24 ) along counter 2  .
[Format (x,y)]
4.887 0.422     #Coordinates of corner piont (278 ,24 ) along counter 2  .
[Format (x,y)]
4.904 0.422     #Coordinates of corner piont (279 ,24 ) along counter 2  .
[Format (x,y)]
4.922 0.422     #Coordinates of corner piont (280 ,24 ) along counter 2  .
[Format (x,y)]
4.939 0.422     #Coordinates of corner piont (281 ,24 ) along counter 2  .
[Format (x,y)]
4.957 0.422     #Coordinates of corner piont (282 ,24 ) along counter 2  .
[Format
```

Fig 10.1) SLC file created from DICOM image

10.2.2) Create an STL fie

STL files of different sections of the model was developed when after rapid prototyping these sections can be build up to create the major organ. An algorithm and method to add different anatomical organs, example tumor and normal anatomy onto one single 3D representation was developed and implemented.

Te following is an example of an ASCII STL file and a binary STL file was created that include the color of the pixel.

10.2.3) ASCII STL file

```
solid
  facet normal 0.0 0.0 -1.0
  outer loop
    vertex  -1.5 -1.5 1.4
    vertex 0.0 1.7 1.4
    vertex  1.5 -1.5 1.4
  endloop
  endfacet
endsolid
```

Fig 10.2) ASCII STL file

10.2.4) 3D model of an anatomical structure

A 3D model of an anatomical structure was developed in software where measurements and changes can be made. The measurements of the STL model must be the same as that of the information described in the DICOM file.

An STL file for Rapid Prototyping was developed from the structure to create a physical model. The laminated object manufacturing (LOM) at the Mechanical Dept. at CPUT was used to manufacture the physical slice of the scull.

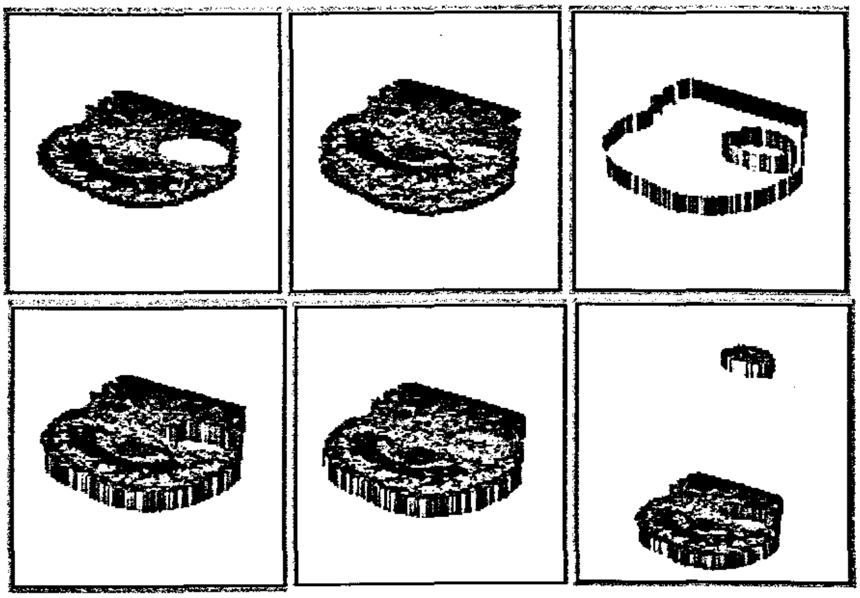


Fig 10.3) 3D model of an anatomical structure

10.2.5) Remote Image view

A remote image view was developed to improve communication between the clinician and surgeon to make any changes or require additional information to be added to the anatomical structure.

10.2.5.1) Outer rim edge detection

An outer rim edge detection algorithm was developed that detects only the outer edge of the anatomical structure. Another method to display the outer edge of an organ is to scan the external surface of the organ and create a point cloud. A software code was written in C programming to display these point clouds information in a mesh.

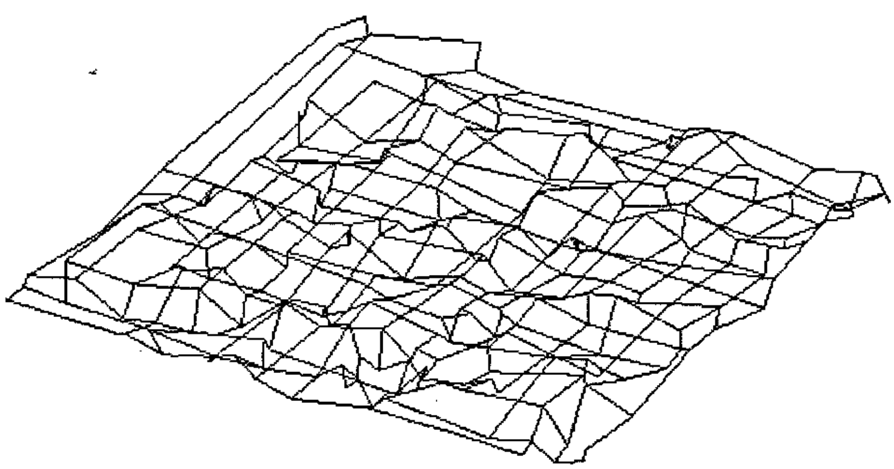


Fig 10.4) Results of the mesh creation of the point cloud reconstruction

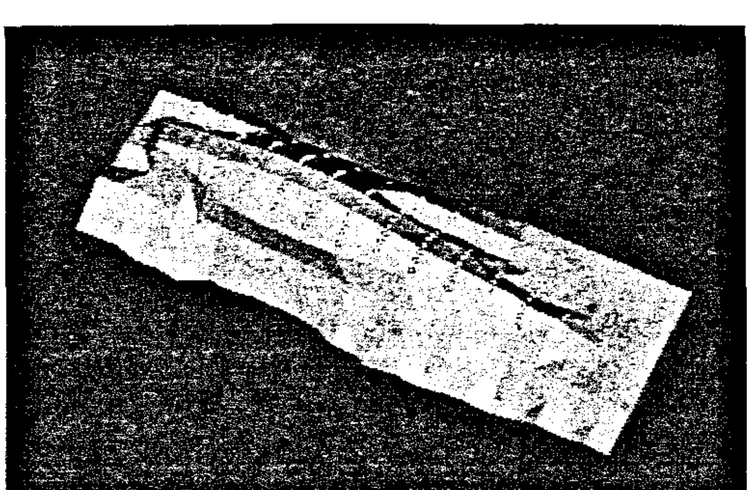


Fig10.5) Example of the outer surface reconstruction

10.2.6) Reduction of a 3D STL file size

A method to reduce the file size of a 3D STL file was developed and still keep the image information was developed. The triangle reduction algoritm is used to display the 3D information but reduce the filesize of an STL fille.

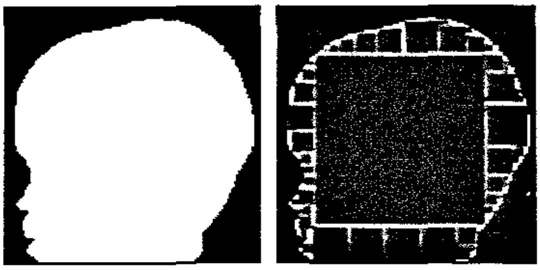


Fig 10.6) Results of triangle reduction algorithm

10.2.6.1) Comparison display the original flat surface of the scull

10.2.6.2) 3D Display without triangle reduction

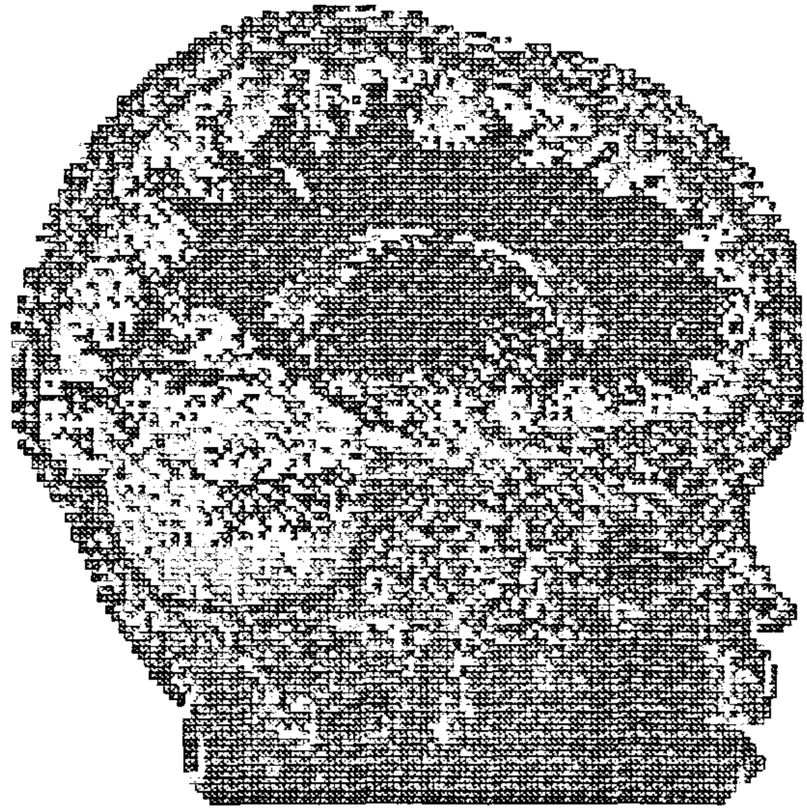


Fig 10.7) Image without triangle reduction

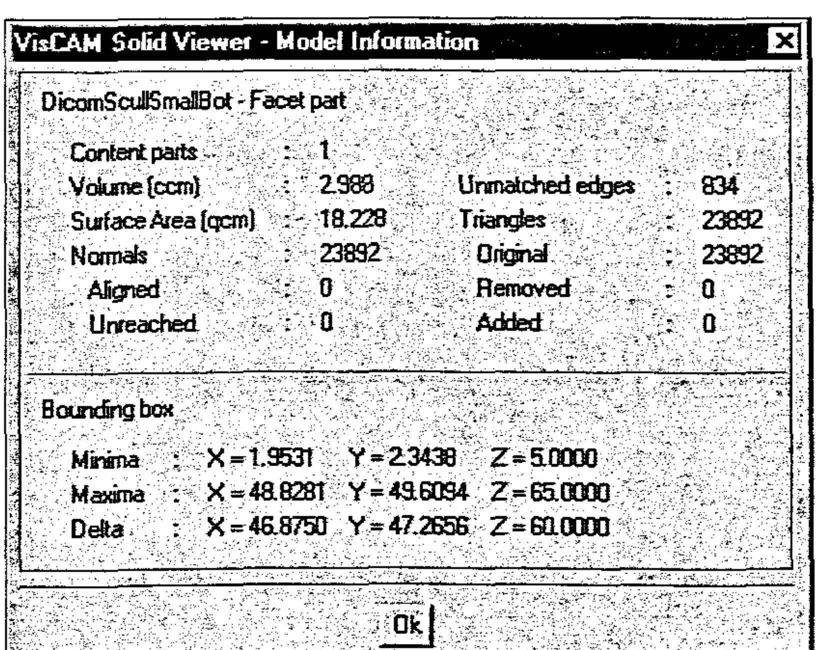


Fig 10.8) Information of triangles without triangle reduction

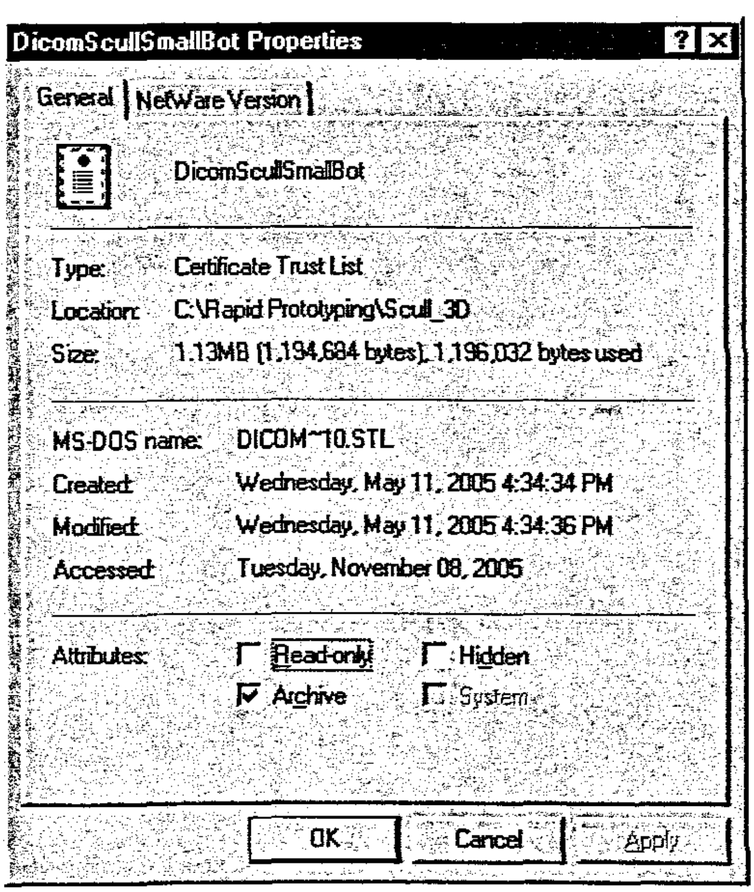


Fig 10.9) Information of file size without triangle reduction

10.2.6.3) 3D Display with triangle reduction

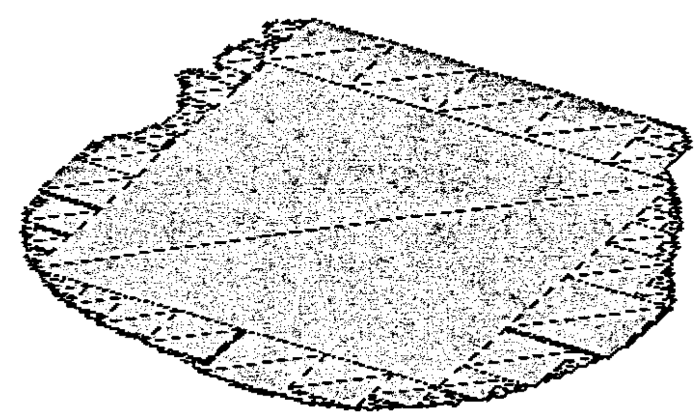


Fig 10.10) 3D Image with triangle reduction

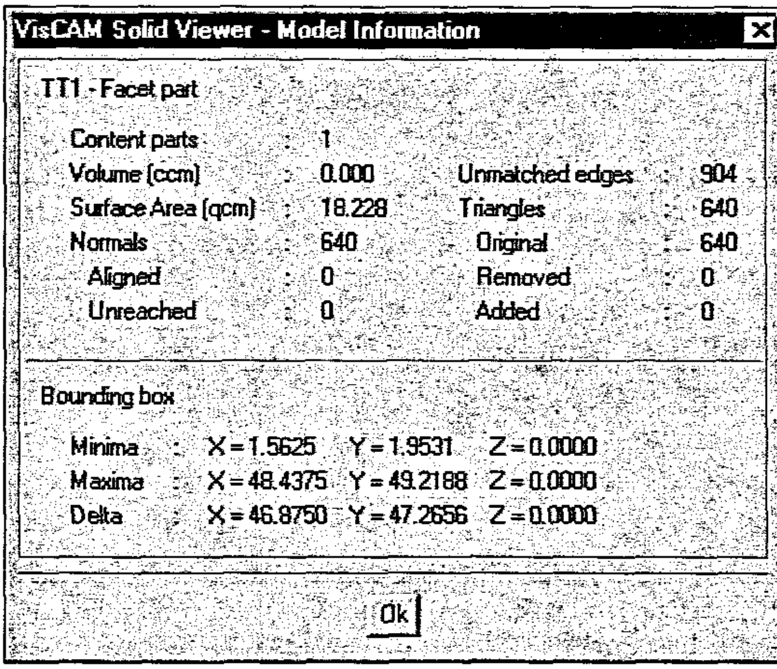


Fig 10.11) Information of triangles with triangle reduction

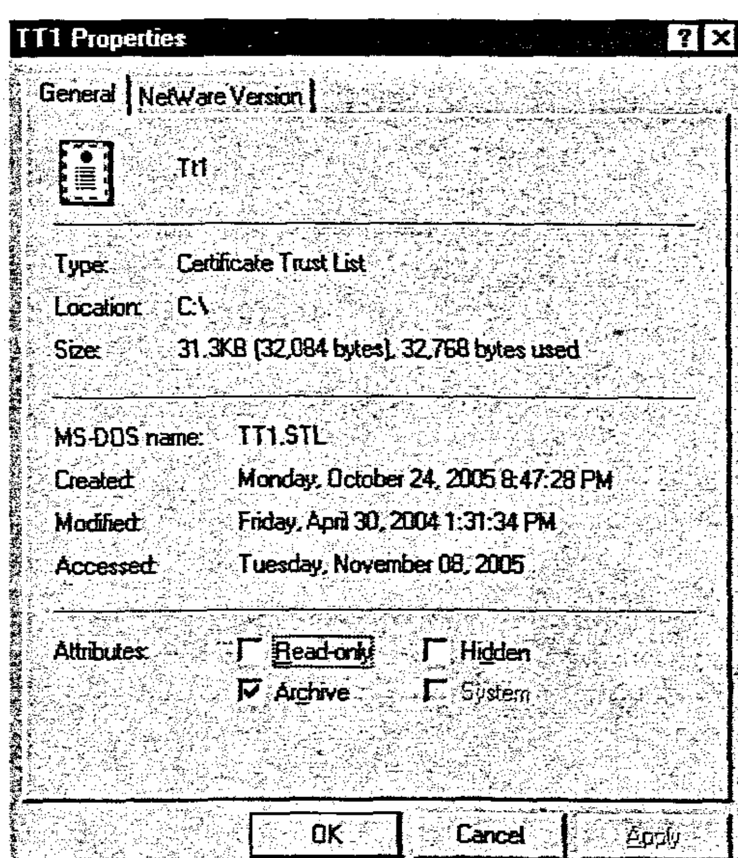


Fig 10.12) Information of file size with triangle reduction

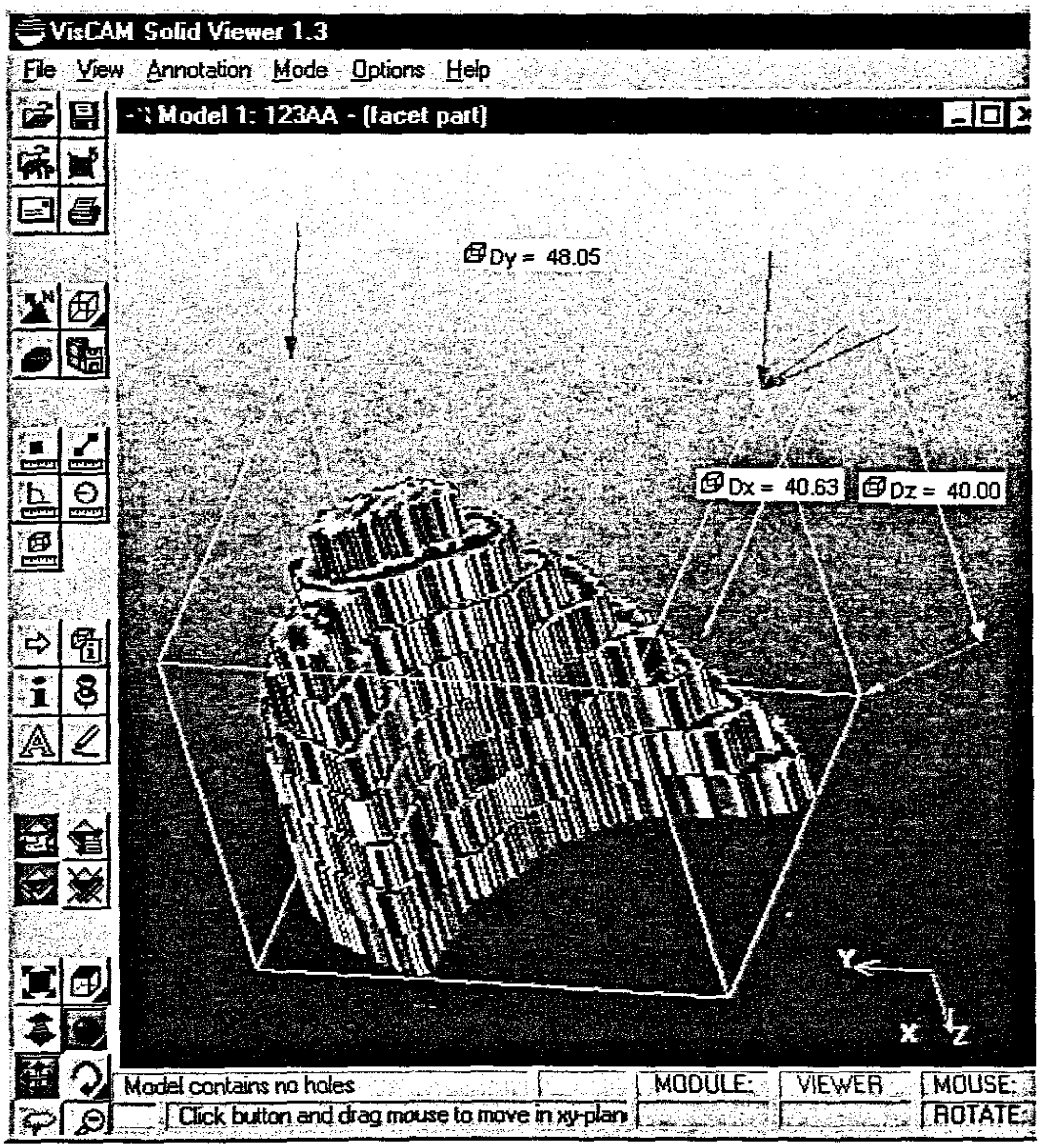


Figure 10.13): 3D product for Rapid Prototyping

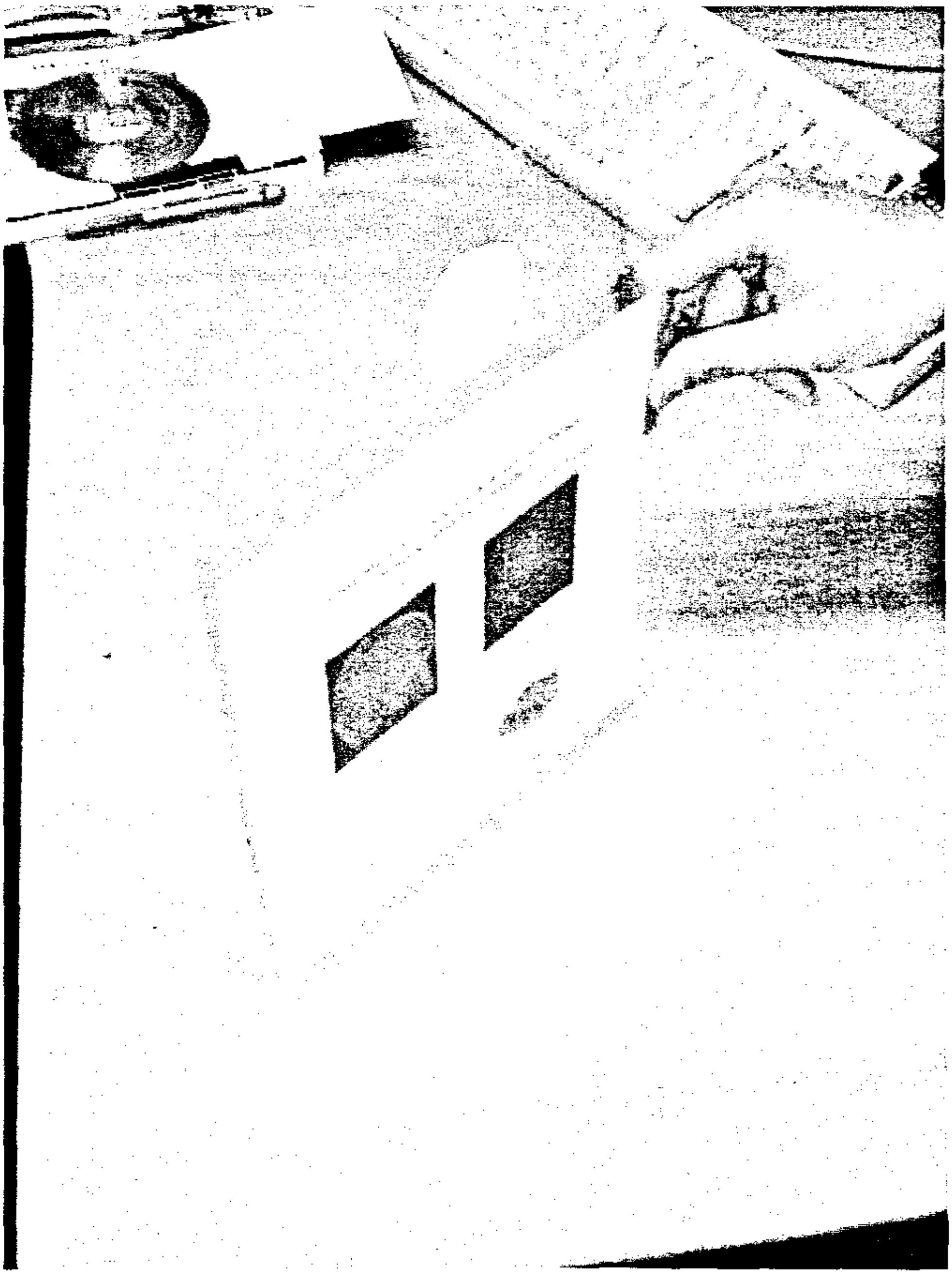
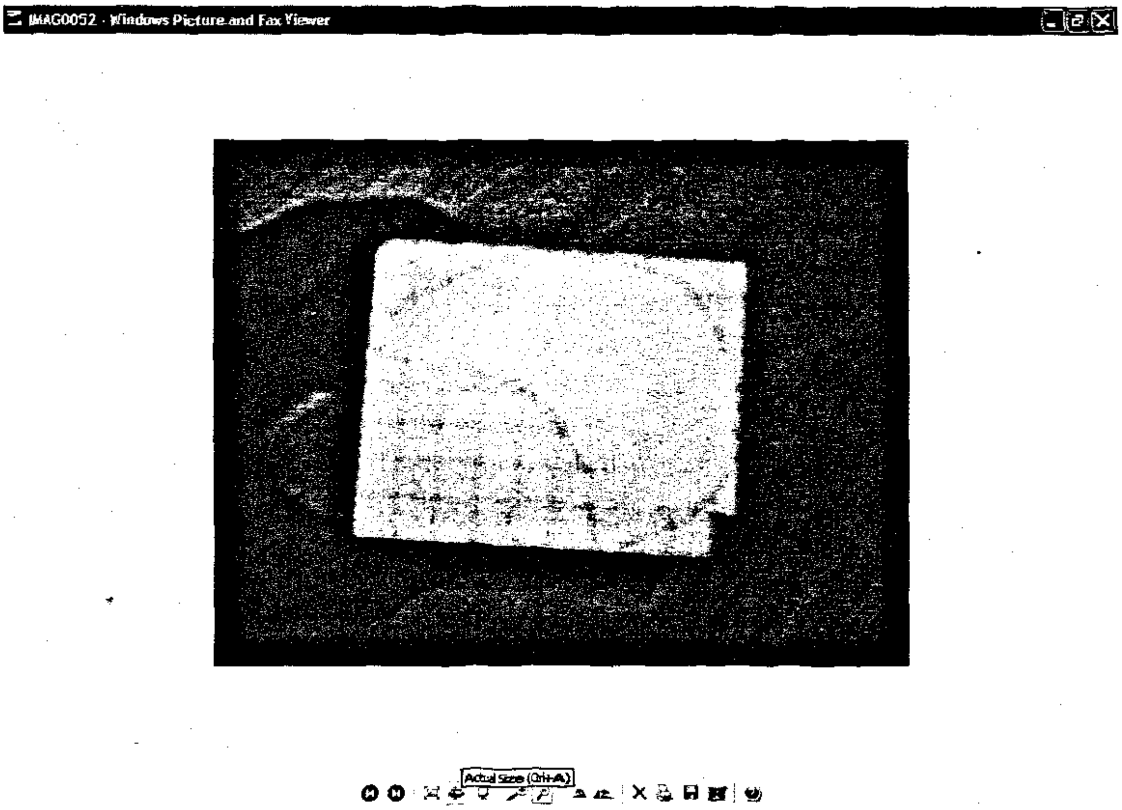
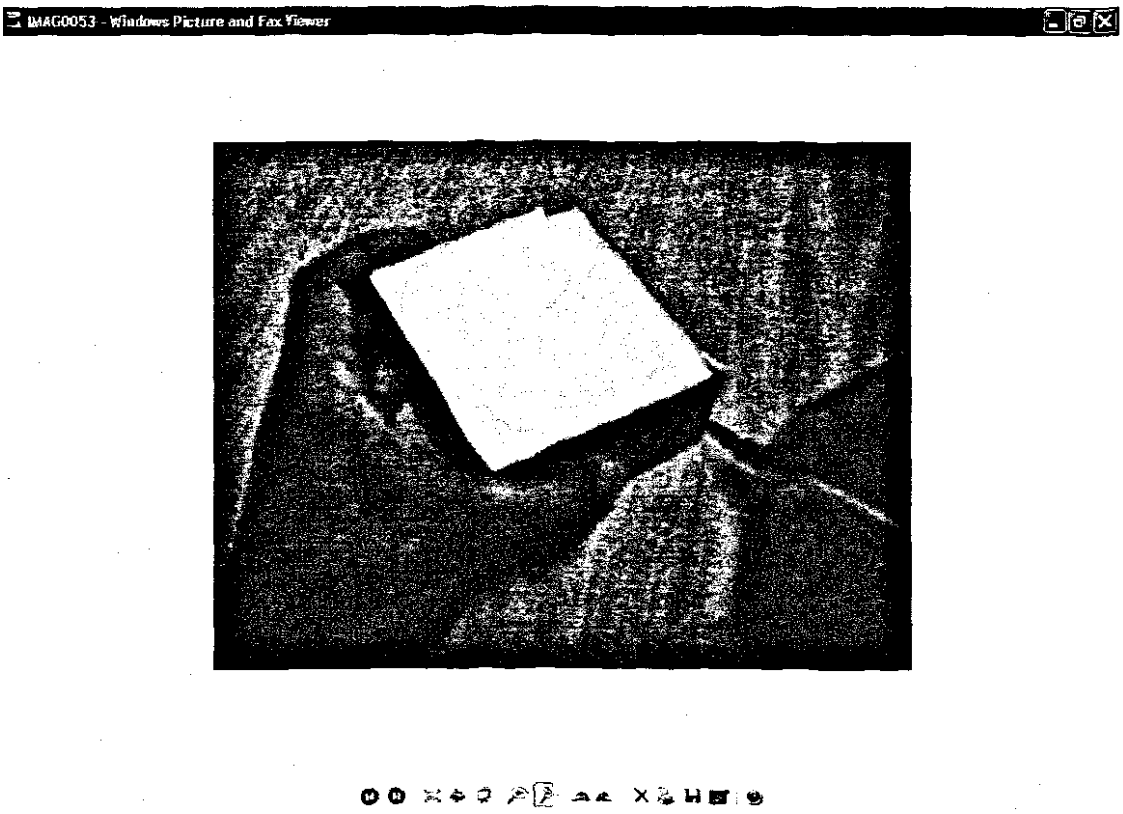


Fig 10.14) Rapid Prototyping of a Scull slice produced (2D)

10.2.6.4) RP Model developed






IMAG0054 - Windows Picture and Fax Viewer


IMAG0055 - Windows Picture and Fax Viewer

10.2.7) Software used and developed

The software required was developed both in C software language, C++ Builder as well as PYTHON programming.

The final result was a comprehensive method to read in these medical image slices, filter out information and produce a physical artifact using Rapid Prototyping.

10.2.8) Edge detection methods

A number of additional work was done by introducing algorithms for edge detection, creating 3D edges both in XZ an YZ plane and adding these results together.

These information can also be used for future work where non invasive post mortems can be used to re-create anatomical organs and have physical models developed as references.

Methods for 3D reconstruction of anatomical organs are also used by BIOBUILD and Materialise.

CHAPTER 11

11) Operators Manuel of the Software Code developed

This chapter gives a brief overview of the operation of the software code developed.

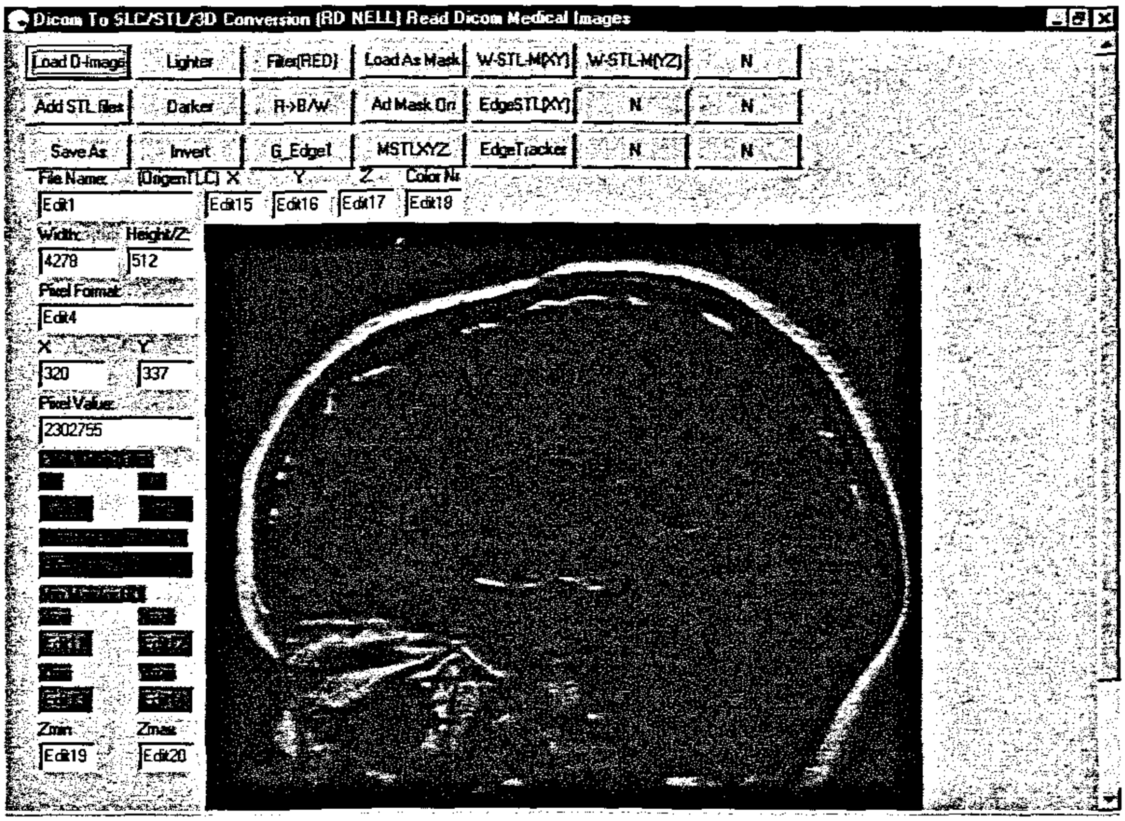


Fig 11.1): Reading in a DICOM file (MRI of the Scull) from the Software created.

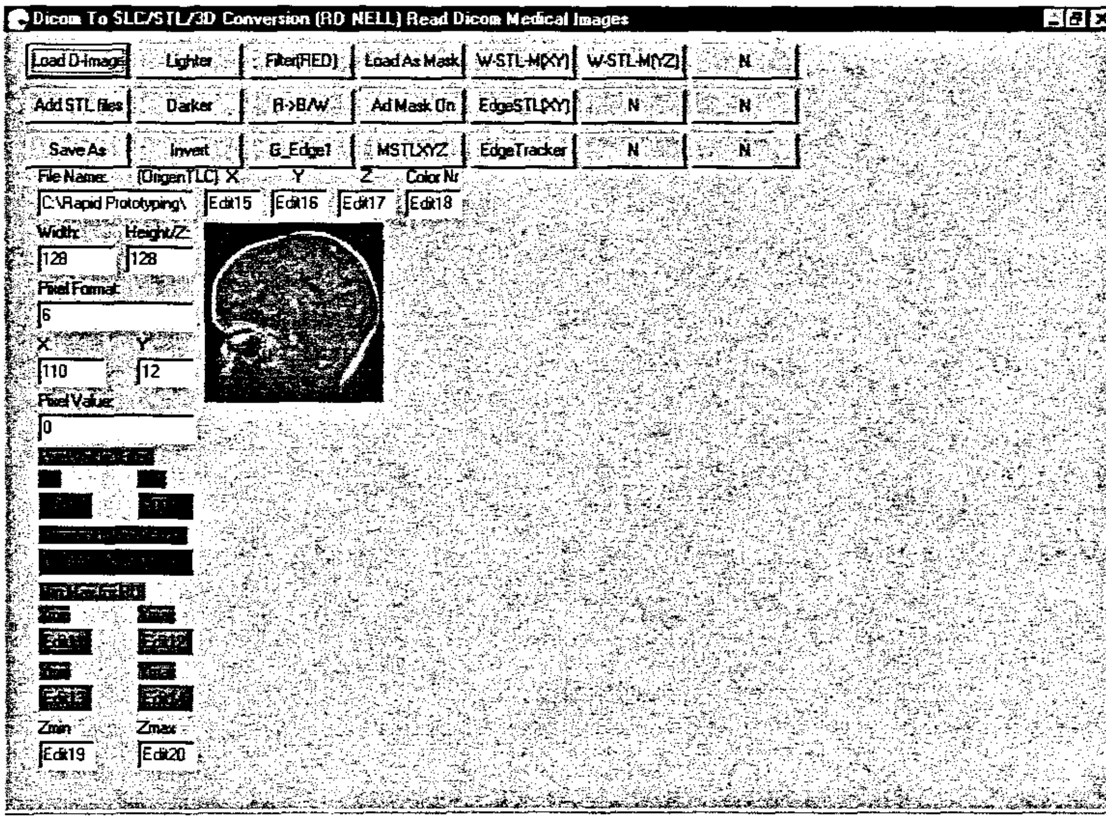


Fig 11.2) The image is reduce and create STL file for physical model

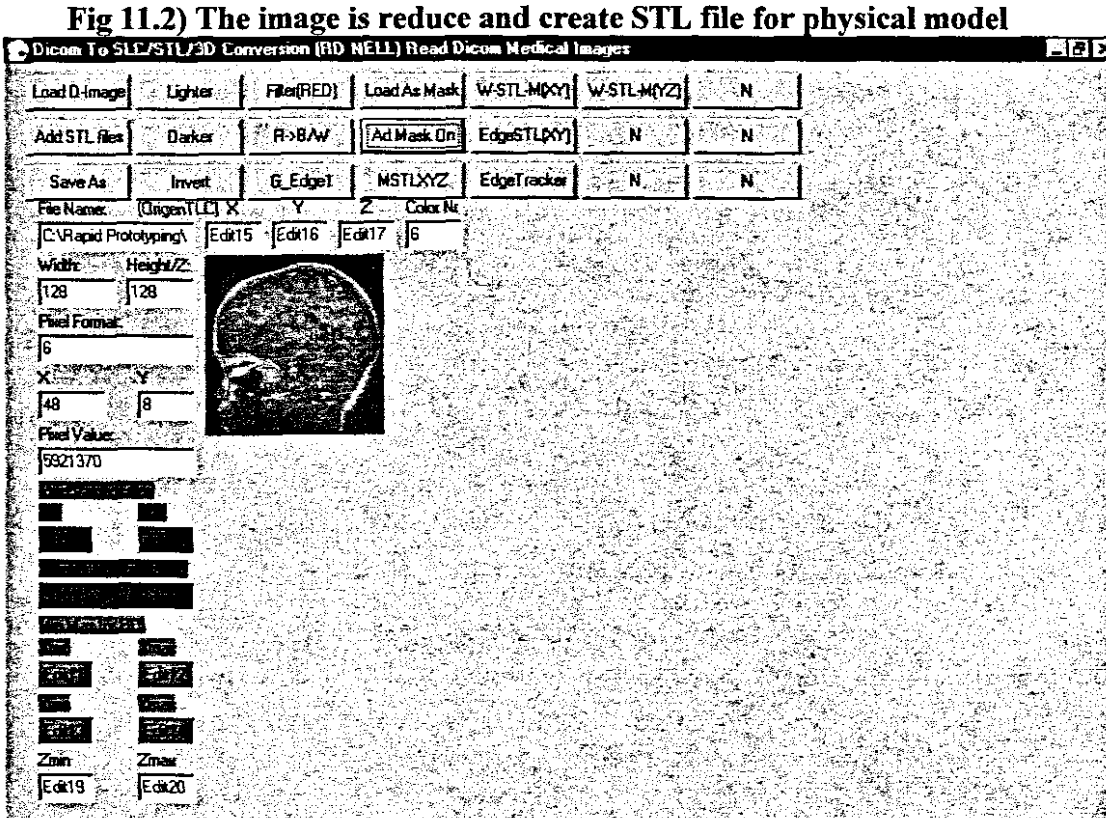


Fig 11.3) Detecting the edge

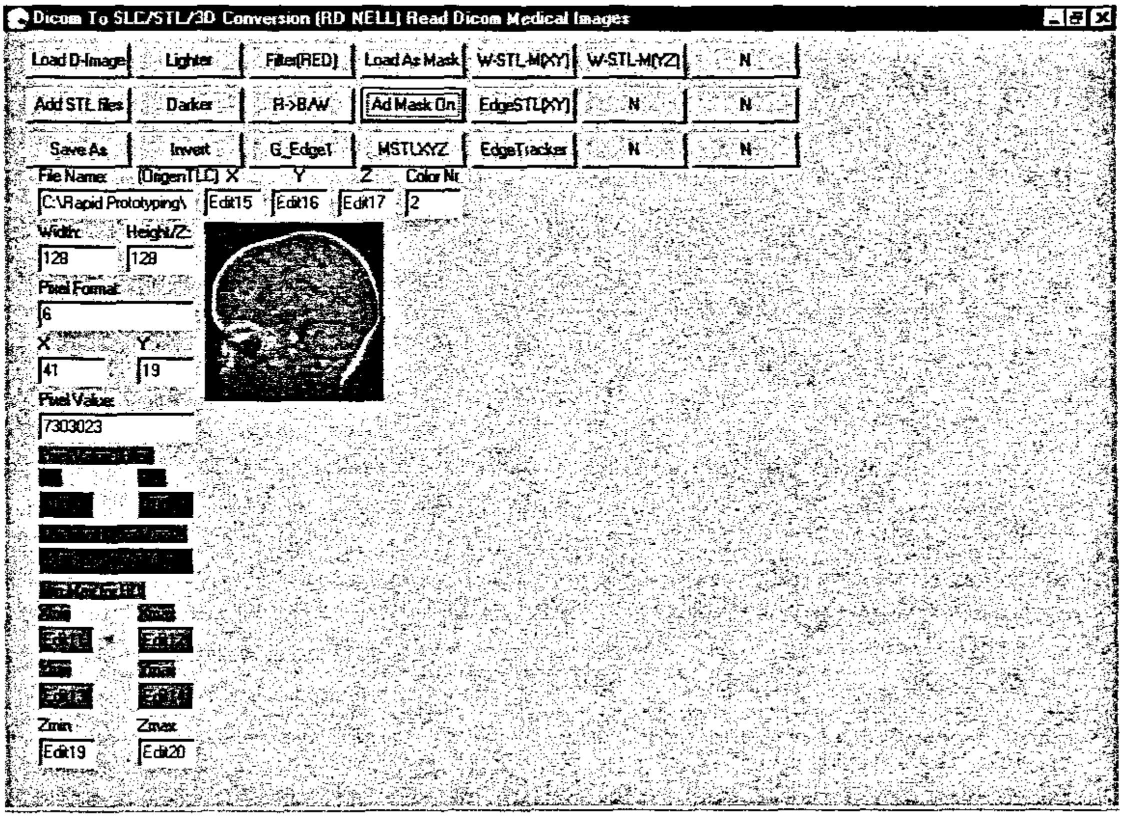


Fig 11.4) Extract the anatomical structure of interest

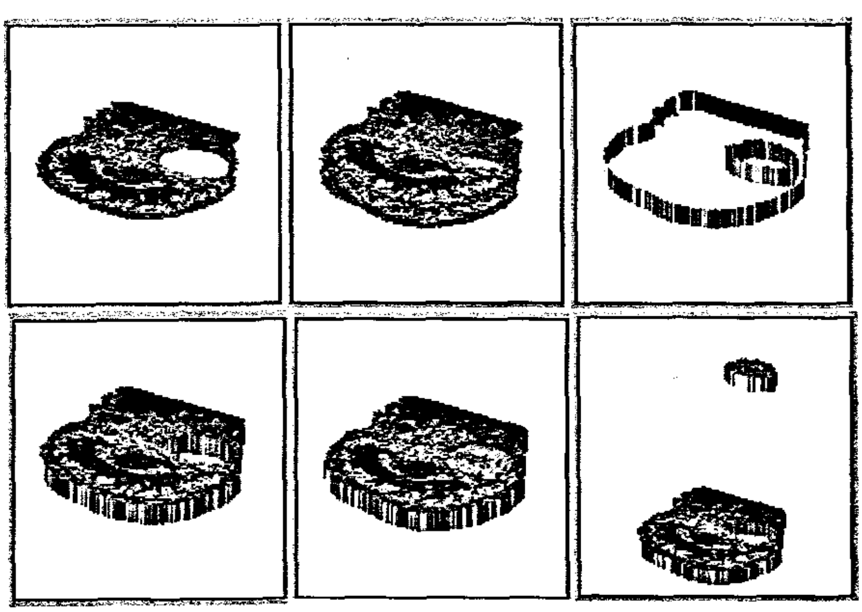


Fig11.5) Create the STL file of the Scull

CHAPTER 12

12) Publication

The following paper was submitted for publication.

Conditions of submission

Re manuscript entitled: **Theoretical method for detecting the 3D point in space with one X-ray exposure**

By: R.D. Nell

Submitted for publication in *The South African Radiographer* ISSN 0258 0241

....

Reserved rights

All proprietary rights other than copyright are reserved to the authors, as well as the right to reproduce original figures and tables from this item in their future works, provided full credit is given to the original publication *The South African Radiographer* ISSN 0258 0241

Originality

The authors warrant that this submission is an original works and that neither this work nor one with substantially similar content has been published elsewhere in whole or in part [except in abstract form], that it has only been submitted for this publication, and that it will not be submitted elsewhere pending written notice of acceptance or rejection by this journal.

Avowal of authorship

The undersigned certify that each has participated sufficiently in the conception and design of this work and the analysis of the data [where applicable], as well as the writing of the manuscript to take responsibility for it. The undersigned believe the manuscript represents valid work and have reviewed the final version of the manuscript and approve it for publication.

Data availability

The undersigned attest that they shall produce the data upon which the manuscript is based for the evaluation by the editors or the assignees should it be requested.

Disclaimer

The undersigned warrant that the manuscript contains no libellous or unlawful statements and does not infringe on the rights of others including a person's rights to privacy and confidentiality. If excerpts [e.g. text and artwork] from all types of copyrighted works are included in this manuscript, written permission will be secured by the authors and proper credit will be shown in the manuscript.

TRANSFER OF COPYRIGHT

Author/s' own work

In consideration of the reviewing and editing done by the editors of *The South African Radiographer* of the above named manuscript, the author/s hereby transfer, assign, or otherwise convey all copyright ownership world-wide, in all languages, to the Society of Radiographers of South Africa in the event that this manuscript is accepted for publication.

Commissioned work

If the manuscript has been commissioned by another person or organisation, or if it has been written as part of the duties of an employee, an authorised representative of the commissioning organisation or employer must also sign this form stating his/her title in the organisation

....

AUTHOR/S

SURNAME/FAMILY NAME & INITIALS [please use block letters]

____

____

Signature : ________________________________ **Date:**______________

Signature : ________________________________ **Date:**______________

Tick appropriate block ✓ **Author/s own work** ☐ **Commissioned work** ☐

Commissioned work

Name of the commissioning organisation or employer ______________________

Signature of employer or authorised representative ______________ **Date** __________

Fax : 27 (0) 21 4212566

POST ORIGINAL TO

SOCIETY OF RADIOGRAPHERS OF SA
PO BOX 6014
ROGGEBAAI SOUTH AFRICA 8012

12.1: Paper for Publication

Theoretical method for detecting the 3D point in space with one X-ray exposure

R Nell Nat Dip Rad (D), B Tech (Elec. Eng) (CPUT)

Abstract

To detect the position of an artifact in space with an X-ray, we normally do two exposures. In these cases we either use two X-ray exposures by moving the tube or we move a patient to obtain a 90-degree view from the previous one. In the movement of the patient or the X-ray tube, we are moving the position of the artifact from the Anterior Posterior (AP) to Lateral (Lat) position. This will expose the radiographer and patient to radiation and to limit the amount of radiation for detection of an artifact, the method to detect the position in space with only one X-ray exposure was developed.

The work done on detecting the point in 3D is in the beginning and although planning was done and seeking new methods, improvements can be made to deliver better quality of position detection in space.

Keywords: Stereoscope, horopter, pseudoscopic, point focus, finite focus, penumbra, Z co-ordinate

Introduction:

In review of the literature, the following write up about X-Ray holograms were mentioned where it is stated that if such holograms become possible, one exposure could provide a radiographic image which the radiologist or clinician could examine from different angles presenting a different radiographic presentation to the observer. [1] Although this aspect is in the beginning stage, the techniques developed can be used for position detection of artifacts.

Stereoscopic vision

Humans are capable of stereoscopic vision which is fundamental to depth perception, because the are able to focus both eyes on a single object which (see figure 1 and figure 2).Stereoscopic vision can be described in terms of the vision process involved in the use of a stereoscope, which presents an image from two slightly different angles so that the eyes can merge them into a single image in three dimensions. A horopter is the projection of the points in the visual field corresponding to the aggregate of points registering on the two retinas.

The figure below shows L and R representing the two eyes and SS a line (the "horopter") [5] drawn through the point A where the optic axes LA and RA intersect, and parallel to a line joining the two eyes L and R. The point A is seen in corresponding points of the two eyes, axially situated. Two points r and l, however, may be so placed-either in the plane of the horopter or outside it-that the two eyes together perceive the points r and l as one point, B (in Fig. 1 point B is nearer to the eye, and in Fig. 2 farther from the eye than the horopter SS itself.) Now, in Fig. 1, a diagram is made representing l and A, and another representing r and A. Then suppose the former is laid before the left eye and the latter before the right eye. The two optic axes are thus made to converge, so that the image of A is formed in corresponding points in the two eyes. As a result, the points l and r will appear to blend into one, situated either nearer the eye than A or further from it. This explains the action of the stereoscope and also the "pseudoscopic" effect produced when the pictures are reversed.

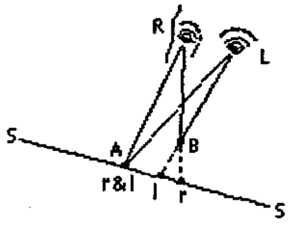


Figure 1 [7]

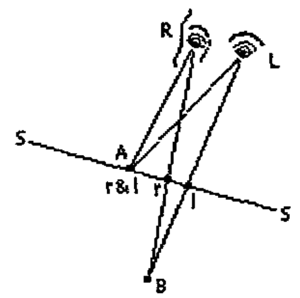


Figure 2 [7]

Image of X-ray beam entering patient

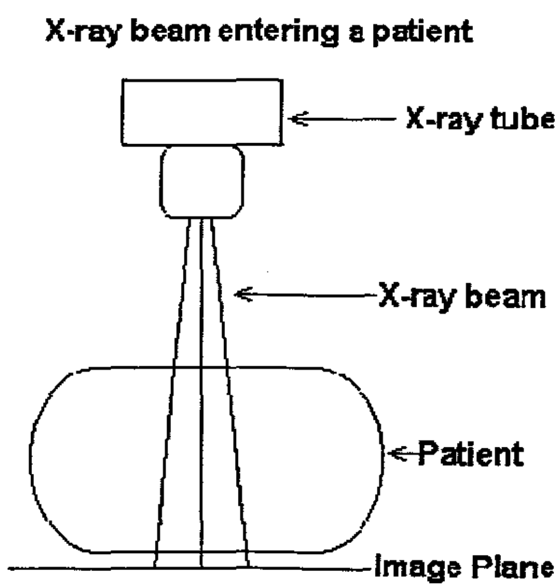


Figure 3: X-ray set up for patient examination [1]

Examination Anterior Posterior (AP) and Lateral (Lat) position using two exposures

To detect the position in space we normally do two exposures. [4]

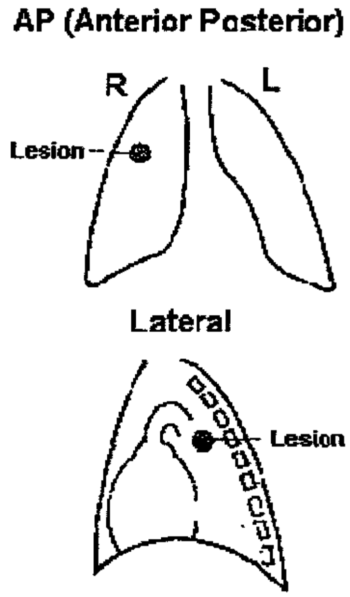


Figure 4: Example of localising a lesion in an X-ray examination of the chest [1]

Method to detect the position in space with only one exposure
Two techniques have been developed to detect the position of an artifact in 3D space.

1) Graphical method using the enlargement technique and the penumbra of the image.
2) Theoretical method using the characteristics of the X-ray image system.

Definitions
1) Point focus
This is the ideal situation in which the source of the X-rays is a point focus.
See Figure (Point source of X-ray)

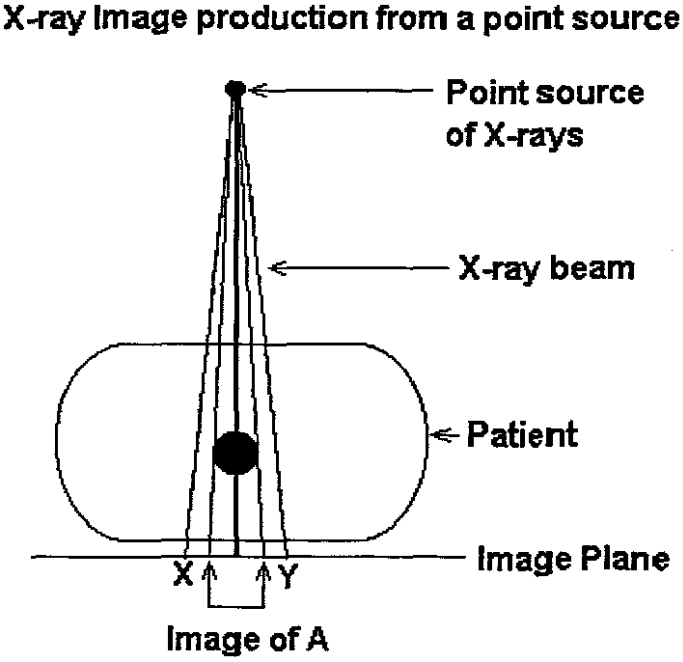


Figure 5: X-ray source from a point focus [1]

2) Finite focus
The X-ray source has a finite area.
See Figure (Finite source of X-ray)

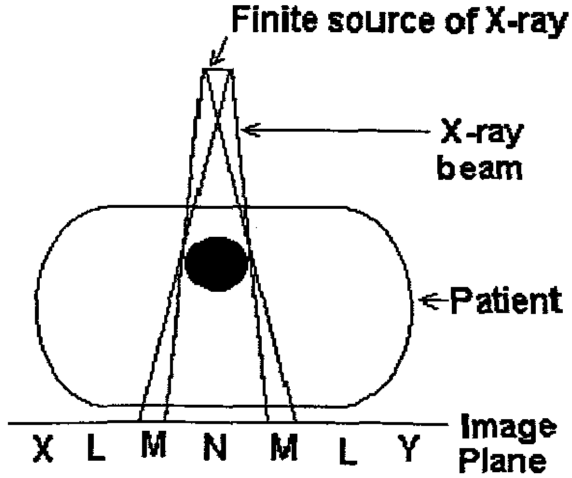


Figure 6: X-ray source from a finite area [1]

Image from Finite area
In figure 6 consider the image production from a source of finite area. Rays traced from the edges of the source divide the image plane into regions L, M and N were M is the penumbra.

3) Penumbra
This is a partial shadow when the image is created and is called the penumbra.
See Figure (Geometry of image formation)

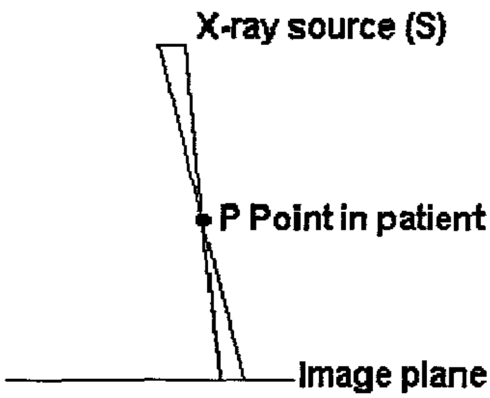


Figure 7: Penumbra produced from a point
[1]

Technique 1): Graphical method using the enlargement technique.
In this case we use two X-ray image planes as positioned in figure below.

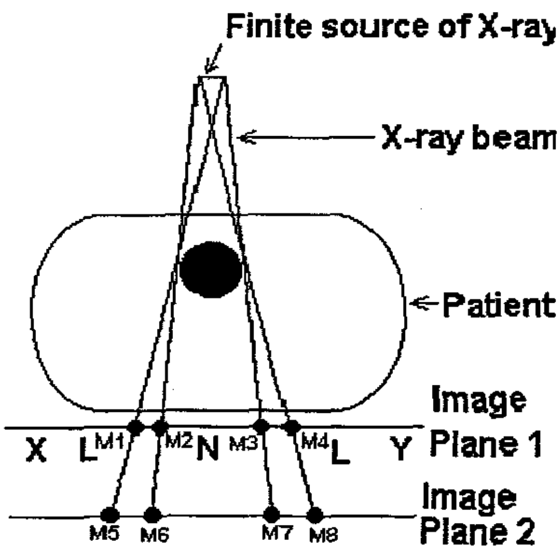


Figure 8: Detecting the penumbra of an artefact

As can see only one exposure will produce two different images in size and using this enlargement technique the position of the artifact can be determined.

The image on image plane 1 can be projected at another position as from image two with a different setup.

If viewed from the sides, two different images are obtained with slightly different co-ordinants.

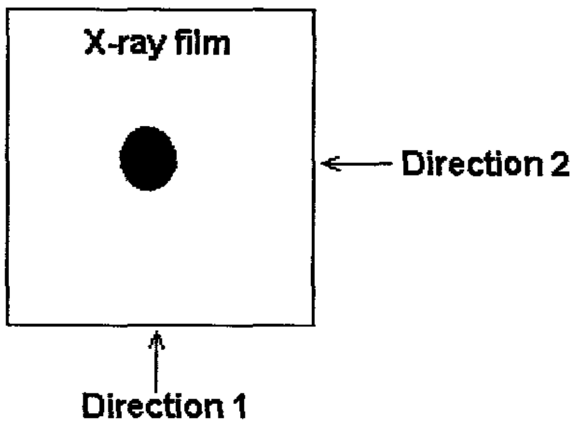


Figure 9: X-ray image view from the above

The image as projected from the side can be viewed in two directions, namely direction 1 (see on figure 9) and direction 2 (see on figure 9). The penumbra on image plane 1 and image plane 2 are projected with different co-ordinates.

Lets examine the images as viewed from direction 1

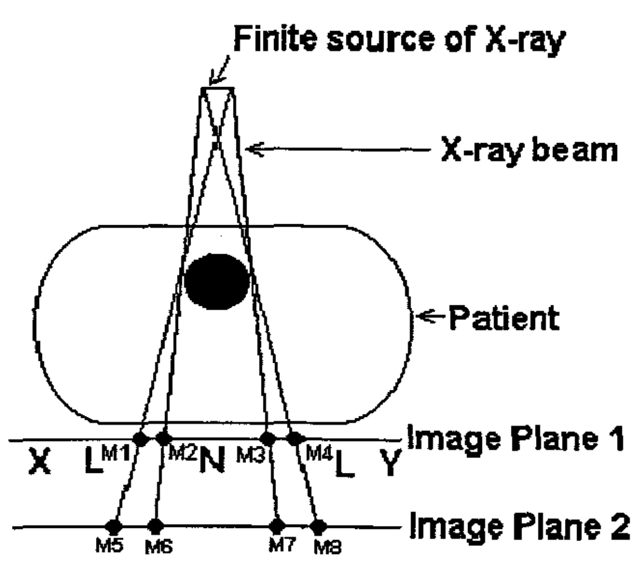


Figure 10: Using the penumbra to detect the position

The distance between Image Plane 1 and Image Plane 2 is a set value. With the known distance and the points M1, M2, M3, M4, M5, M6, M7, M8 marked and determined, there are eight co-ordinants (four per image plane) and this information is used to determine the position.

Now lets extend line M5 through M1 and line M2 through M6. Where the intersection of these line M7 M3 and line M8 M4 is on figure 10, it will give a position in space. This will give the left point. By doing the same with M3 and M7 and M4 and M8 the right hand side is determined.

The same process can be followed when viewing the image from direction 2.

Certainly a practical consideration must be considered and this aspect gives an explination for 3D position detection with one exposure.

Detecting a single point in space

Lets consider figure 11 below, which shows a finite source of X-ray (S), a point (P) in an object, and the two image planes [I1] and [I2].

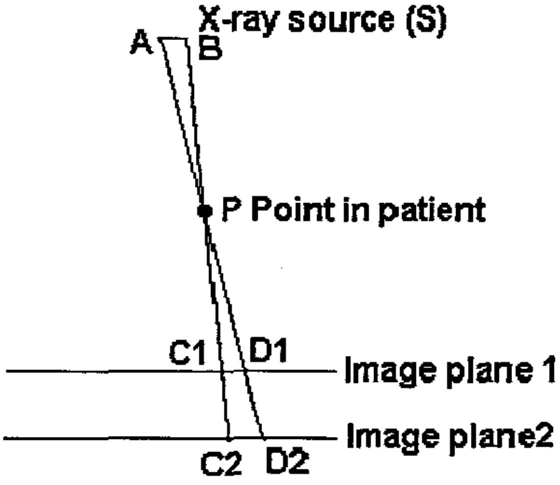


Figure 11

At the image planes the images of P is represented by C1, D1 and C2, D2. The image P should be a point, but because of the finite size of the X-ray source it is imaged as a disk whose diameter is C1, D1 and C2, D2.

Since the distance between image plane 1 and image plane 2 is a set distance, the co-ordinates of C1, D1, C2, D2 can be determined. Using the graphical method lines C1 and C2 can be extended and also lines D1 and D2. This will provide two lines that will cross at point P.

For the above methods two image planes are required, but with the next method only one-image plane is needed.

Technique 2): Theoretical method using the characteristics of the X-ray System

Lets consider the image below with only on image plane. Here certain characteristics of the system must be known, namely the focus size as well position of the focus in the system.

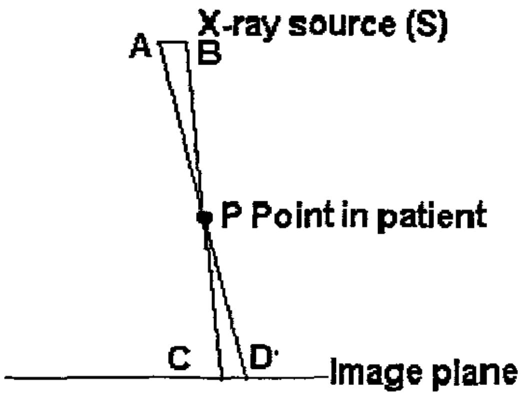


Figure 12

The penumbra is also used, but the position in space is calculated using the characteristics of the image system. The formation of the penumbra is a consequence of the finite size of the X-ray source. Consider the figure below which shows a finite source of X-ray (S), a point P, in an object and the image plane (I) which would normally be accompanied by a film, screen cassette or image intensifier input phosphor. At the image plane, CD represents the image of P. The image P should be a point, but because of the finite size of the X-ray source, it is imaged as a disk whose diameter is CD. Triangles ABP and CDP are geometrically similar. Therefore AB/CD=BP/CP. Re-arranging this gives CD=(AB X CP)/BP.

Lets get the co-ordinants of the positions of A, B, C and D. Since the position of C and D is projected on the image plane, their co-ordinates can be determined. If the size of the focus (S) is known and it's position the co-ordinates of A, and B must be determined. Now we have the co-ordinates of A, B, C and D and we form a line from A to D and from B to C. By getting the mathematical formula of these lines and at their crossing point, will give the position of point P.

By using the same method as previously explained, the position of point P can be calculated with only on the image plane and only one X-ray exposure by using the characteristics of the system and references.

Point Spread Function

The tissues displayed by X-ray imaging consist of a number of minute points of anatomical detail. A point structure in an object should produce a point image. In practice the image of a point will be unsharp and will and will appear as a small blurred disk. This will make the recognizing of the penumbra very difficult.

Limitations to consider

One of the major limitations is the measurement of the penumbra. In many cases the penumbra will be very small due to the long source to object distance and small size of the focal spot. [1] The size of the penumbra is dependent on the position of the object from the X-ray source. Another limitation is the fact than most of the objects blocks X-ray partially only (Radio lucent material). The point-spread function will also contribute to the unsharpened of the point in space.

Concluding Remarks

This is a theoretical method of determining the height of a point in space. Here are two methods of determining the Z co-ordinate by using only one X-ray exposure. Certainly the technique and methods can be refined to obtain better results. By having these two methods, they can be used as a checking method for each other.

Acknowledgements

I wish to thank all those for their input and support.
Mr. Aladdin Speelman: School of Radiography
Groote Schuur Hospital
Cape Peninsula University of Technology.
Mr. Miles Jacobs: Department of Electrical Engineering
Cape Peninsula University of Technology.

References:

1. Ball J & Price T. Chesney's Radiographic Imaging. 5th edition, Blackwell Scientific Publications, Oxford, 1989,25-28, 335-343.
2. MacDonald S & Burns D. Physics For The life and Health Sciences, Addison-Wesley Publishing Company, Amsterdam, 1979, 677-684
3. Webster J, Medical Instrumentation: Application and Design. 3rd edition, 1998, 532-554
4. McMullan J.T., Physical Techniques in Medicine. Volume 1, John Wiley & Sons, 1979, 68-71
5. S. Delorme, Y. Petit, J. A. de Guise, H. Labelle, C. E. Aubin and J. Dansereau,"Assesment of 3-D Reconstruction and High-Resolution Geometrical Mdeling of the Human Skeletal Trunk From 2-D Radiographic Images". IEEE Trans. Biomed. Eng., vol 50, pp.989-998, Aug. 2003.
6. A. Berenstein, J. Hartman, "Three Dimensional Arteriography", Simens Electro Medica Special Issue Cardiology,vol 68, pp27-30 1-31
7. Microsoft® Encarta® Encyclopedia 2003. 1993-2002 Microsoft Corporation.

Raymond is a Radiographer and is involved in research in 3D reconstruction of anatomical structures. He has a qualification in Radiography, Degree in Electrical Engineering and also does his Master's Degree in Electrical Engineering. Raymond is involved in Rapid Prototyping and has produced an artifact of the human scull using methods and algorithms developed by him. A paper on Design and Analysis of a System for the fabrication of anatomical structures was presented at the University of Cape Town (UCT) in November 2004.His knowledge in Radiography, Electrical Engineering, micro controller and computer programming, laid a firm basis for recognizing new techniques and algorithms for imaging of the human organs.

Contact Details:
Tel:(021) 633 7774 (H)/
(021) 954 2237 (W)
email: raymondnell@webmail.co.za

CHAPTER 13

13) Software code

13.1) Software code written in C++ Builder

This chapter gives the sours code of the software developed to read and display medical images.

```
//---------------------------------------------------------------------
-----------
#include <vcl.h>
#include <math.h>
#pragma hdrstop

#include "Main.h"

#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<io.h>
#include<fcntl.h>
#include<sys\stat.h>
#include<time.h>

//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"


unsigned char Mask[520][520]={0},
              ImageData1[520][520]={0};
//---------------------------------------------------------------------
-----
void delayms(float time_in_ms)
{
   clock_t start, end;
   start = clock();

   while(1)
   {
   end = clock();
   if(
        (((end - start) / CLK_TCK)*1000.0)>=time_in_ms
      )
  break;

   }
}
//---------------------------------------------------------------------
-----
int STLheader(char *description,unsigned long
number_of_trangles,FILE *stream)
{
int n;

for(n=0;n<=79;n++)
if(fwrite(&description[n], sizeof(description[n]), 1, stream)!=1)
return(0);
```

```
if(fwrite(&number_of_trangles, sizeof(number_of_trangles), 1,
stream)!=1) return(0);

return(1);
}
//--------------------------------------------------------------
-----
int STLtriangle(
           float *point,
           float *normal,
           unsigned char *colour,
           FILE *stream
          )
{
short int STL_RGB_colour,
          n;

for(n=0;n<=2;n++)
if(fwrite(&normal[n], sizeof(normal[n]), 1, stream)!=1) return(0);

for(n=0;n<=8;n++)
  if(fwrite(&point[n], sizeof(point[n]), 1, stream)!=1) return(0);

//STL_RGB_colour=0x8000|(colour[0]&0x1F)<<10|(colour[1]&0x1F)<<5|(
colour[2]&0x1F);
STL_RGB_colour=0x0000|(colour[0]&0x1F)<<10|(colour[1]&0x1F)<<5|(co
lour[2]&0x1F);
if(fwrite(&STL_RGB_colour, sizeof(STL_RGB_colour), 1, stream)!=1)
return(0);

return(1);

}
//--------------------------------------------------------------
-----
int file_copy(char *oldname1,char *oldname2, char *newname, long
byte_number)
{
FILE *fold1,
     *fold2,
     *fnew;
int c;
long file2_byte_number=0;

//Open the first source file for reading in binary mode
if(
   (fold1=fopen(oldname1,"rb"))==NULL
  )
  return(-1);

//Open the second source file for reading in binary mode
if(
   (fold2=fopen(oldname2,"rb"))==NULL
  )
  {
   fclose(fold1);
  return(-1);
  }

//Open the destination file for writing in binary mode
```

```
if(
   (fnew=fopen(newname,"wb"))==NULL
  )
  {
   fclose(fold1);
   fclose(fold2);
   return(-1);
  }


//Read one byte at a time from first source file
//if end of file has not been reached
//write the byte to the destination file
while(1)
  {
   c=fgetc(fold1);
   if( !feof(fold1) )
     fputc(c,fnew);
   else
    break;
  }


//Read one byte at a time from second source file
//if end of file has not been reached
//write the byte to the destination file
while(1)
  {
   c=fgetc(fold2);
   if( !feof(fold2) )
   {
   file2_byte_number++;
   if(file2_byte_number>=byte_number)
     fputc(c,fnew);
   }
   else
    break;
  }


fclose(fnew);
fclose(fold1);
fclose(fold2);

return(0);

}
//---------------------------------------------------------------------
----------
int scan_rectangle(int left,int top,int right,int bottom,int
pixelvalue)
{
int x,
    y;

for(y=top;y<=bottom;y++)
for(x=left;x<=right;x++)
  if(ImageData1[x][y]!=(BYTE)pixelvalue)
  //if(getpixel(x,y)!=pixelvalue)
  return(0);
return(1);
```

```
}
//--------------------------------------------------------------------
----------

void get_blocks_of_value(int value)
{
   int x,
       y,
       xx,
       yy,
       _fillcolor,
       width=200,//511,
       height=200,//511,
       scan_width,
       scan_height,
       step=1,
       step1=1,
       y_start=0,
       x_start=0,
       y_end=height,
       x_end=width,
       number_of_squares=0,
       pic_width,
       pic_height,
       pixel_color,
       image_type;

   //char str[100];
for(scan_height=height;scan_height>=0;scan_height=scan_height-
step)
for(scan_width=width;scan_width>=0;scan_width=scan_width-step)
for(y=y_start;y<=y_end;y=y+step1)
for(x=x_start;x<=x_end;x=x+step1)
   {
    //if(scan_height==height)
    scan_height=scan_width;
    //if(scan_width==width)
    //scan_width=scan_height;
     if(
      ((x+scan_width)<=width)&&
      ((y+scan_height)<=height)
       )
      if(scan_rectangle(x,y,x+scan_width,y+scan_height,value))
        {
           if(scan_width<=1)
         _fillcolor=3;//(BYTE)value+1;//random(13)+1;
           else
         _fillcolor=1;//(BYTE)value+1;//random(13)+1;
         for(yy=y;yy<=y+scan_height;yy++)
          for(xx=x;xx<=x+scan_width;xx++)
            {
             if(
              (yy==y)||(yy==(y+scan_height))||
              (xx==x)||(xx==(x+scan_width))
             )
                 //;
            //putpixel(110+xx,yy,_fillcolor);
            ImageData1[xx][yy]=_fillcolor;
              else
            ImageData1[xx][yy]=_fillcolor+1;
            }
```

```
          //number_of_squares++;
         }
        /*
        if(
         ((x+scan_height)<=width)&&
         ((y+scan_width)<=height)
        )
         if(scan_rectangle(x,y,x+scan_height,y+scan_width,value))
           {

          //_fillcolor=255;//random(13)+1;
          _fillcolor=100;//(BYTE)value+1;//random(13)+1;
          for(yy=y;yy<=y+scan_width;yy++)
           for(xx=x;xx<=x+scan_height;xx++)
             {
              if(
               (yy==y)||(yy==(y+scan_width))||
               (xx==x)||(xx==(x+scan_height))
              )
                  ;
             //putpixel(110+xx,yy,_fillcolor);
             ImageData1[xx][yy]=_fillcolor;
             }
            number_of_squares++;
           }
             */
}
}
//-----------------------------------------------------------------------
----------
void get_lines(void)
{
int x,y,xx,yy,n,xxx,yyy,lenght1,lenght2,lenght,
    p1x,p1y,p2x,p2y,p3x,p3y,p4x,p4y;
float s1x,s1y,s2x,s2y,s3x,s3y,xpixel_lenght=1,ypixel_lenght=1;

for(y=0;y<511;y++)
 for(x=0;x<511;x++)
  if(ImageData1[x][y]==14)
    {
     s1x=(x-1)*xpixel_lenght;s1y=(y-1)*ypixel_lenght;
     s2x=x*xpixel_lenght;s2y=(y-1)*ypixel_lenght;
     s3x=x*xpixel_lenght;s3y=y*ypixel_lenght;
     //line(310+s1x,s1y,310+s2x,s2y);
     //line(310+s2x,s2y,310+s3x,s3y);
     //line(310+s3x,s3y,310+s1x,s1y);
     s1x=(x-1)*xpixel_lenght;s1y=(y-1)*ypixel_lenght;
     s2x=x*xpixel_lenght;s2y=y*ypixel_lenght;
     s3x=(x-1)*xpixel_lenght;s3y=y*ypixel_lenght;
     //line(310+s1x,s1y,310+s2x,s2y);
     //line(310+s2x,s2y,310+s3x,s3y);
     //line(310+s3x,s3y,310+s1x,s1y);
    }
//getch();
for(y=0;y<511;y++)
 for(x=0;x<511;x++)
  if(
     (ImageData1[x][y]==15)&&
     (ImageData1[x][y+1]==15)&&
     (ImageData1[x+1][y]==15)&&
     (ImageData1[x+1][y+1]==12)
```

```
)
{
 lenght1=x;
 xx=x;
 yy=y;
 p1x=xx;p1y=yy;
 while(1)
 {
  if(
   (ImageData1[xx][yy]==15)&&
   (ImageData1[xx-1][yy]==15)&&
   (ImageData1[xx][yy+1]==15)&&
   (ImageData1[xx-1][yy+1]==12)
   )
   break;
   else
   xx++;
 }
 lenght2=xx;
 lenght=lenght2-lenght1;
 p2x=p1x+lenght;p2y=p1y;
 p3x=p1x+lenght;p3y=p1y+lenght;
 p4x=p1x;p4y=p1y+lenght;

 s2x=(p1x-1)*xpixel_lenght;s2y=(p1y-1)*ypixel_lenght;

s1x=s2x+(lenght*xpixel_lenght/2);s1y=s2y+(lenght*ypixel_lenght/2);

//s1x=s2x+(lenght*xpixel_lenght/4);s1y=s2y+(lenght*ypixel_lenght/4
);

 for(xxx=p1x;xxx<=p2x;xxx++)
 {
  if
   (
    (ImageData1[xxx][yy]==15)&&
    (ImageData1[xxx][yy-1]==14)
   )
   {
    s3x=(xxx-1)*xpixel_lenght;s3y=yy*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
  else if
   (
    (ImageData1[xxx][yy]==15)&&
    (ImageData1[xxx-1][yy]==15)&&
    (ImageData1[xxx][yy+1]==15)&&
    (ImageData1[xxx-1][yy+1]==12)
   )
   {
    s3x=xxx*xpixel_lenght;s3y=(p1y-1)*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
  else if
   (
```

```
    (ImageData1[xxx][yy-1]==15)&&
    (ImageData1[xxx-1][yy-1]==15)&&
    (ImageData1[xxx][yy-2]==15)&&
    (ImageData1[xxx-1][yy-2]==12)&&
    (ImageData1[xxx+1][yy-1]!=15)&&
    (ImageData1[xxx+1][yy-2]!=15)
   )
   {
    s3x=xxx*xpixel_lenght;s3y=(p1y-1)*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
  else if
   (
    (ImageData1[xxx][yy-1]==15)&&
    (ImageData1[xxx+1][yy-1]==15)&&
    (ImageData1[xxx][yy-2]==15)&&
    (ImageData1[xxx+1][yy-2]==12)&&
    (ImageData1[xxx-1][yy-1]!=15)&&
    (ImageData1[xxx-1][yy-2]!=15)
   )
   {
    s3x=(xxx-1)*xpixel_lenght;s3y=(p1y-1)*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
  else if
   (
    (ImageData1[xxx][yy-1]==15)&&
    (ImageData1[xxx+1][yy-1]==15)&&
    (ImageData1[xxx][yy-2]==15)&&
    (ImageData1[xxx+1][yy-2]==12)
   )
   {
    s3x=xxx*xpixel_lenght;s3y=(p1y-1)*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
 }

 s2x=p2x*xpixel_lenght;s2y=(p2y-1)*ypixel_lenght;
 for(yyy=p2y;yyy<=p3y;yyy++)
 {
  if
   (
    (ImageData1[xx][yyy]==15)&&
    (ImageData1[xx+1][yyy]==14)
   )
   {
    s3x=xx*xpixel_lenght;s3y=yyy*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
```

```
else if
 (
  (ImageData1[xx][yyy]==15)&&
  (ImageData1[xx][yyy-1]==15)&&
  (ImageData1[xx-1][yyy]==15)&&
  (ImageData1[xx-1][yyy-1]==12)
 )
 {
  s3x=xx*xpixel_lenght;s3y=yyy*ypixel_lenght;
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
 }
else if
 (
  (ImageData1[xx+1][yyy]==15)&&
  (ImageData1[xx+2][yyy]==15)&&
  (ImageData1[xx+1][yyy-1]==15)&&
  (ImageData1[xx+2][yyy-1]==12)&&
  (ImageData1[xx+1][yyy+1]!=15)&&
  (ImageData1[xx+2][yyy+1]!=15)
 )
 {
  s3x=xx*xpixel_lenght;s3y=yyy*ypixel_lenght;
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
 }
else if
 (
  (ImageData1[xx+1][yyy]==15)&&
  (ImageData1[xx+2][yyy]==15)&&
  (ImageData1[xx+1][yyy+1]==15)&&
  (ImageData1[xx+2][yyy+1]==12)&&
  (ImageData1[xx+1][yyy-1]!=15)&&
  (ImageData1[xx+2][yyy-1]!=15)
 )
 {
  s3x=xx*xpixel_lenght;s3y=yyy*ypixel_lenght;
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
 }
else if
 (
  (ImageData1[xx+1][yyy]==15)&&
  (ImageData1[xx+2][yyy+1]==15)&&
  (ImageData1[xx+1][yyy+1]==15)&&
  (ImageData1[xx+2][yyy+1]==15)
 )
 {
  s3x=xx*xpixel_lenght;s3y=yyy*ypixel_lenght;
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
 }
}
```

```
//--------------------------------------------------------------
---------
s2x=p3x*xpixel_lenght;s2y=p3y*ypixel_lenght;
yy=p3y;
for(xxx=p3x;xxx>=p4x;xxx--)
{
 if
  (
   (ImageData1[xxx][yy]==15)&&
   (ImageData1[xxx][yy+1]==14)
  )
  {
   s3x=(xxx-1)*xpixel_lenght;s3y=yy*ypixel_lenght;
   //line(310+s1x,s1y,310+s2x,s2y);
   //line(310+s2x,s2y,310+s3x,s3y);
   //line(310+s3x,s3y,310+s1x,s1y);
   s2x=s3x;s2y=s3y;
  }
 else if
  (
   (ImageData1[xxx][yy]==15)&&
   (ImageData1[xxx+1][yy]==15)&&
   (ImageData1[xxx][yy-1]==15)&&
   (ImageData1[xxx+1][yy-1]==12)
  )
  {
   s3x=(xxx-1)*xpixel_lenght;s3y=yy*ypixel_lenght;
   //setcolor(9);
   //line(310+s1x,s1y,310+s2x,s2y);
   //line(310+s2x,s2y,310+s3x,s3y);
   //line(310+s3x,s3y,310+s1x,s1y);
   s2x=s3x;s2y=s3y;
   //setcolor(15);
  }
 else if
  (
   (ImageData1[xxx][yy]==15)&&
   (ImageData1[xxx-1][yy]==15)&&
   (ImageData1[xxx][yy-1]==15)&&
   (ImageData1[xxx-1][yy-1]==12)
  )
  {
   //setcolor(9);
   s3x=(xxx-1)*xpixel_lenght;s3y=yy*ypixel_lenght;
   //line(310+s1x,s1y,310+s2x,s2y);
   //line(310+s2x,s2y,310+s3x,s3y);
   //line(310+s3x,s3y,310+s1x,s1y);
   s2x=s3x;s2y=s3y;
   //setcolor(15);
  }
 else if
  (
   (ImageData1[xxx][yy+1]==15)&&
   (ImageData1[xxx+1][yy+1]==15)&&
   (ImageData1[xxx][yy+2]==15)&&
   (ImageData1[xxx+1][yy+2]==12)&&
   (ImageData1[xxx-1][yy+1]!=15)&&
   (ImageData1[xxx-1][yy+2]!=15)
  )
  {
   //setcolor(9);
```

```
     s3x=(xxx-1)*xpixel_lenght;s3y=yy*ypixel_lenght;
     //line(310+s1x,s1y,310+s2x,s2y);
     //line(310+s2x,s2y,310+s3x,s3y);
     //line(310+s3x,s3y,310+s1x,s1y);
     s2x=s3x;s2y=s3y;
     //setcolor(15);
    }
  else if
   (
     (ImageData1[xxx][yy+1]==15)&&
     (ImageData1[xxx-1][yy+1]==15)&&
     (ImageData1[xxx][yy+2]==15)&&
     (ImageData1[xxx-1][yy+2]==12)&&
     (ImageData1[xxx+1][yy+1]!=15)&&
     (ImageData1[xxx+1][yy+2]!=15)
    )
    {
     //setcolor(9);
     s3x=xxx*xpixel_lenght;s3y=yy*ypixel_lenght;
     //line(310+s1x,s1y,310+s2x,s2y);
     //line(310+s2x,s2y,310+s3x,s3y);
     //line(310+s3x,s3y,310+s1x,s1y);
     s2x=s3x;s2y=s3y;
     //setcolor(15);
    }
  else if
   (
     (ImageData1[xxx][yy+1]==15)&&
     (ImageData1[xxx-1][yy+1]==15)&&
     (ImageData1[xxx][yy+2]==15)&&
     (ImageData1[xxx-1][yy+2]==15)
    )
    {
     //setcolor(9);
     s3x=(xxx-1)*xpixel_lenght;s3y=yy*ypixel_lenght;
     //line(310+s1x,s1y,310+s2x,s2y);
     //line(310+s2x,s2y,310+s3x,s3y);
     //line(310+s3x,s3y,310+s1x,s1y);
     s2x=s3x;s2y=s3y;
     //setcolor(15);
    }
   }
 //-------------------------------------------------------------
---------
 s2x=(p4x-1)*xpixel_lenght;s2y=p4y*ypixel_lenght;
 xx=p4x;
 //for(xxx=p3x;xxx>=p4x;xxx--)
 for(yyy=p4y;yyy>=p1y;yyy--)
 {
  if
   (
    (ImageData1[xx][yyy]==15)&&
    (ImageData1[xx-1][yyy]==14)
   )
   {
    s3x=(xx-1)*xpixel_lenght;s3y=yyy*ypixel_lenght;
    //line(310+s1x,s1y,310+s2x,s2y);
    //line(310+s2x,s2y,310+s3x,s3y);
    //line(310+s3x,s3y,310+s1x,s1y);
    s2x=s3x;s2y=s3y;
   }
```

```
else if
 (
  (ImageData1[xx][yyy]==15)&&
  (ImageData1[xx+1][yyy]==15)&&
  (ImageData1[xx][yyy-1]==15)&&
  (ImageData1[xx+1][yyy-1]==12)
 )
 {
  s3x=(xx-1)*xpixel_lenght;s3y=yyy*ypixel_lenght;
  //setcolor(8);
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
  //setcolor(15);
 }
else if
 (
  (ImageData1[xx][yyy]==15)&&
  (ImageData1[xx+1][yyy]==15)&&
  (ImageData1[xx][yyy+1]==15)&&
  (ImageData1[xx+1][yyy+1]==12)
 )
 {
  s3x=(xx-1)*xpixel_lenght;s3y=(yyy-1)*ypixel_lenght;
  //setcolor(8);
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
  //setcolor(15);
 }
else if
 (
  (ImageData1[xx-1][yyy]==15)&&
  (ImageData1[xx-1][yyy+1]==15)&&
  (ImageData1[xx-2][yyy]==15)&&
  (ImageData1[xx-2][yyy+1]==12)&&
  (ImageData1[xx-1][yyy-1]!=15)&&
  (ImageData1[xx-2][yyy-1]!=15)
 )
 {
  s3x=(xx-1)*xpixel_lenght;s3y=(yyy-1)*ypixel_lenght;
  //setcolor(8);
  //line(310+s1x,s1y,310+s2x,s2y);
  //line(310+s2x,s2y,310+s3x,s3y);
  //line(310+s3x,s3y,310+s1x,s1y);
  s2x=s3x;s2y=s3y;
  //setcolor(15);
 }
else if
 (
  (ImageData1[xx-1][yyy]==15)&&
  (ImageData1[xx-2][yyy]==15)&&
  (ImageData1[xx-1][yyy-1]==15)&&
  (ImageData1[xx-2][yyy-1]==12)&&
  (ImageData1[xx-1][yyy+1]!=15)&&
  (ImageData1[xx-2][yyy+1]!=15)
 )
 {
  s3x=(xx-1)*xpixel_lenght;s3y=yyy*ypixel_lenght;
```

```
          //setcolor(8);
          //line(310+s1x,s1y,310+s2x,s2y);
          //line(310+s2x,s2y,310+s3x,s3y);
          //line(310+s3x,s3y,310+s1x,s1y);
          //s2x=s3x;s2y=s3y;
          //setcolor(15);
         }
        else if
         (
          (ImageData1[xx-1][yyy]==15)&&
          (ImageData1[xx-2][yyy]==15)&&
          (ImageData1[xx-1][yyy-1]==15)&&
          (ImageData1[xx-2][yyy-1]==15)
         )
         {
          s3x=(xx-1)*xpixel_lenght;s3y=(yyy-1)*ypixel_lenght;
          //setcolor(8);
          //line(310+s1x,s1y,310+s2x,s2y);
          //line(310+s2x,s2y,310+s3x,s3y);
          //line(310+s3x,s3y,310+s1x,s1y);
          s2x=s3x;s2y=s3y;
          //setcolor(15);
         }
       }


     }
}
//-------------------------------------------------------------------
----------
int DICOM_3D_point_in_STL(char *source_file, BYTE min_value, BYTE
max_value)
{
        char _source_file[400];
        FILE *source_file_handle;

        char byte_read,
             byte1=NULL,byte2=NULL,byte3=NULL,
             byte4=NULL,byte5=NULL,byte6=NULL,
             message[5],
             Rows[5]={0x28,0x00,0x10,0x00,NULL},
             Columns[5]={0x28,0x00,0x11,0x00,NULL},
             Bits_Allocated[5]={0x28,0x00,0x00,0x01,NULL},
             Bits_Stored[5]={0x28,0x00,0x01,0x01,NULL},
             High_Bit[5]={0x28,0x00,0x02,0x01,NULL},
             Pixel_Data[5]={0xE0,0x7F,0x10,0x00,NULL};

        int  _width=0,
             _height=0,
             _bits_allocated=0,
             _high_bit=0,
             bits_stored=0,
             max_width=512,
             max_height=512;

        long bytenumber=0,
             message_number=0,
             Pixel_Data_tag_found=0;

//        OpenDialog1->Filter = "Dicom files (*.dcm)|*.DCM|All
files (*.*)|*.*";
```

```
//        if (OpenDialog1->Execute())
        {
//            strcpy(source_file,OpenDialog1->FileName.c_str());
          strcpy(_source_file,source_file);
          if((source_file_handle=fopen(_source_file,"rb"))==NULL)
          return(0);
              //ShowMessage("Error Opening Source file");
           else
           {
             while(1)
                  {
                   byte_read=fgetc(source_file_handle);
                   message[message_number]=byte_read;

                   if (bytenumber>=4)
                        {

                        if(
                           (message[0]==Rows[0])&&
                           (message[1]==Rows[1])&&
                           (message[2]==Rows[2])&&
                           (message[3]==Rows[3])&&
                           (Pixel_Data_tag_found==0)
                          )
                          {
                          //ShowMessage("Rows Tag found");
                          byte1=fgetc(source_file_handle);
                          byte2=fgetc(source_file_handle);
                          byte3=fgetc(source_file_handle);
                          byte4=fgetc(source_file_handle);
                          byte5=fgetc(source_file_handle);
                          byte6=fgetc(source_file_handle);
                          _width=((byte6 << 8) | byte5);
                          //Edit2->Text = _width;
                          //ShowMessage("Width value read");
                          }

                       if(
                          (message[0]==Columns[0])&&
                          (message[1]==Columns[1])&&
                          (message[2]==Columns[2])&&
                          (message[3]==Columns[3])&&
                          (Pixel_Data_tag_found==0)
                         )
                         {
                          //ShowMessage("Columns Tag found");
                          byte1=fgetc(source_file_handle);
                          byte2=fgetc(source_file_handle);
                          byte3=fgetc(source_file_handle);
                          byte4=fgetc(source_file_handle);
                          byte5=fgetc(source_file_handle);
                          byte6=fgetc(source_file_handle);
                          _height=((byte6 << 8) | byte5);
                          //Edit3->Text = _height;
                          //ShowMessage("Height value read");
                         }

                       if(
                          (message[0]==Bits_Stored[0])&&
                          (message[1]==Bits_Stored[1])&&
```

```
    (message[2]==Bits_Stored[2])&&
    (message[3]==Bits_Stored[3])&&
    (Pixel_Data_tag_found==0)
   )
   {
   //Edit2->Text = bytenumber;
    //ShowMessage("Bits Stored Tag found");
    byte1=fgetc(source_file_handle);
    byte2=fgetc(source_file_handle);
    byte3=fgetc(source_file_handle);
    byte4=fgetc(source_file_handle);
    byte5=fgetc(source_file_handle);
    byte6=fgetc(source_file_handle);
    bits_stored=((byte6 << 8) | byte5);
    //Edit7->Text = bits_stored;
    //ShowMessage("Bits Stored value read");
   }


 if(
    (message[0]==Bits_Allocated[0])&&
    (message[1]==Bits_Allocated[1])&&
    (message[2]==Bits_Allocated[2])&&
    (message[3]==Bits_Allocated[3])&&
    (Pixel_Data_tag_found==0)
   )
   {
    //ShowMessage("Bits Allocated Tag
found");

    byte1=fgetc(source_file_handle);
    byte2=fgetc(source_file_handle);
    byte3=fgetc(source_file_handle);
    byte4=fgetc(source_file_handle);
    byte5=fgetc(source_file_handle);
    byte6=fgetc(source_file_handle);
    _bits_allocated=((byte6 << 8) | byte5);
    //Edit7->Text = _bits_allocated;
    //ShowMessage("Bits Allocated value

read");

   }


 if(
    (message[0]==High_Bit[0])&&
    (message[1]==High_Bit[1])&&
    (message[2]==High_Bit[2])&&
    (message[3]==High_Bit[3])&&
    (Pixel_Data_tag_found==0)
   )
   {
    //ShowMessage("High Bit Tag found");
    byte1=fgetc(source_file_handle);
    byte2=fgetc(source_file_handle);
    byte3=fgetc(source_file_handle);
    byte4=fgetc(source_file_handle);
    byte5=fgetc(source_file_handle);
    byte6=fgetc(source_file_handle);
    _high_bit=((byte6 << 8) | byte5);
    //Edit7->Text = _high_bit;
    //ShowMessage("High Bit value read");
   }
```

```
//PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPP
                        if(
                           (message[0]==Pixel_Data[0])&&
                           (message[1]==Pixel_Data[1])&&
                           (message[2]==Pixel_Data[2])&&
                           (message[3]==Pixel_Data[3])
                          )
                          {
                           Pixel_Data_tag_found=1;
                           //Edit2->Text = bytenumber;
                           //ShowMessage("Pixel Data Tag found");

                           //-------------
                           if (
                              //((_width>0)&&(_width<=max_width))&&

//((_height>0)&&(_height<=max_height))
                              ((_width>0)&&(_width<=512))&&
                              ((_height>0)&&(_height<=512))
                              )
                              {
                               //ShowMessage("Image size in
range");
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++

        int xp, yp;
        short Grayp,ch=0;
        //TColor RGBp;

        //Image1->Picture->Bitmap = new Graphics::TBitmap;
        //Image1->Picture->Bitmap->PixelFormat = pf24bit;
        //Image1->Picture->Bitmap->Width = _width;
        //Image1->Picture->Bitmap->Height = _height;
        for (yp=0; yp<=_height-1; yp++)
                for (xp=0; xp<=_width-1; xp++)
                {
                        ch =
((fgetc(source_file_handle))|(fgetc(source_file_handle)<<8));
                        Grayp =ch>>3;
                        //RGBp = (Graphics::TColor)((Grayp << 16)
| (Grayp << 8) | Grayp);
                        //Image1->Canvas->Pixels[xp][yp] = RGBp;
                }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
                              }
                              else
                              return(0);
                               //ShowMessage("Image size in out of
range.\nMax(512x512)");

                           //-------------
                          }

                        message[0]=message[1];
                        message[1]=message[2];
                        message[2]=message[3];
```

```
                        message_number=3;
                       }
                     else
                        message_number++;

                     if(!feof(source_file_handle))
                        bytenumber++;
                     //else if (Pixel_Data_tag_found==1)
                          //break;
                     else
                        break;
                    }
                fclose(source_file_handle);
                //Edit2->Text = bytenumber; strcatstrcat
                //ShowMessage("Finished Opened Source file ");
            }

        }

}
//------------------------------------------------------------------
----------
int get_start_point(int *start)
{
int x,y;
BYTE value=255;
 //for(x=0;x<=400;x++)
for(y=0;y<511;y++)
 for(x=0;x<511;x++)
  if(ImageData1[x][y]==value)
  {
  start[0]=x;
  start[1]=y;
  return(1);
  }
  return(0);
}
//------------------------------------------------------------------
----------
void get_art_pixels(int x,int y,int *pix)
{
 BYTE value=255;
 if(ImageData1[x-1][y-1]==value)
   pix[1]=1;
 else
   pix[1]=0;
 if(ImageData1[x][y-1]==value)
   pix[2]=1;
 else
   pix[2]=0;
 if(ImageData1[x+1][y-1]==value)
   pix[3]=1;
 else
   pix[3]=0;
 if(ImageData1[x-1][y]==value)
   pix[4]=1;
 else
   pix[4]=0;
 if(ImageData1[x+1][y]==value)
   pix[6]=1;
 else
```

```
    pix[6]=0;
  if(ImageData1[x-1][y+1]==value)
    pix[7]=1;
  else
    pix[7]=0;
  if(ImageData1[x][y+1]==value)
    pix[8]=1;
  else
    pix[8]=0;
  if(ImageData1[x+1][y+1]==value)
    pix[9]=1;
  else
    pix[9]=0;
}
//-----------------------------------------------------------------
-----------
struct facet
{
  float normal_x;
  float normal_y;
  float normal_z;

  float vertex1_x;
  float vertex1_y;
  float vertex1_z;

  float vertex2_x;
  float vertex2_y;
  float vertex2_z;

  float vertex3_x;
  float vertex3_y;
  float vertex3_z;

  int color;
};
//-----------------------------------------------------------------
-----------
//-----------------------------------------------------------------
-----------
TForm1 *Form1;
//-----------------------------------------------------------------
-----------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//-----------------------------------------------------------------
-----------
void __fastcall TForm1::Button1Click(TObject *Sender)
//Load D-Image
{
        char source_file[400];
        int x, y,handle;
        int Gray;
        int ch,
            filename_lenght,
            file_length;
        TColor RGB;
```

```
OpenDialog1->Filter = "Dicom files (*.dcm)|*.DCM|Img files
(*.img)|*.IMG|Bmp files (*.bmp)|*.BMP|All files (*.*)|*.*";
        if (OpenDialog1->Execute())
            {
        strcpy(source_file,OpenDialog1->FileName.c_str());
        filename_lenght=strlen(source_file);


        if(
           ((source_file[filename_lenght-
1]=='p')||(source_file[filename_lenght-1]=='p'))&&
           ((source_file[filename_lenght-
2]=='m')||(source_file[filename_lenght-2]=='m'))&&
           ((source_file[filename_lenght-
3]=='b')||(source_file[filename_lenght-3]=='b'))&&
           ((source_file[filename_lenght-
4]=='.')||(source_file[filename_lenght-4]=='.'))
          )
        {
                Image1->Picture->Bitmap->LoadFromFile(OpenDialog1-
>FileName);
                Edit1->Text = OpenDialog1->FileName;
                Edit2->Text = Image1->Width;
                Edit3->Text = Image1->Height;
                Edit4->Text = Image1->Picture->Bitmap-
>PixelFormat;
        }
        else if(
           ((source_file[filename_lenght-
1]=='m')||(source_file[filename_lenght-1]=='M'))&&
           ((source_file[filename_lenght-
2]=='c')||(source_file[filename_lenght-2]=='C'))&&
           ((source_file[filename_lenght-
3]=='d')||(source_file[filename_lenght-3]=='D'))&&
           ((source_file[filename_lenght-
4]=='.')||(source_file[filename_lenght-4]=='.'))
          )
//...
{
        char _source_file[400];
        FILE *source_file_handle;

        char byte_read,
             byte1=NULL,byte2=NULL,byte3=NULL,
             byte4=NULL,byte5=NULL,byte6=NULL,
             message[5],
             Rows[5]={0x28,0x00,0x10,0x00,NULL},
             Columns[5]={0x28,0x00,0x11,0x00,NULL},
             Bits_Allocated[5]={0x28,0x00,0x00,0x01,NULL},
             Bits_Stored[5]={0x28,0x00,0x01,0x01,NULL},
             High_Bit[5]={0x28,0x00,0x02,0x01,NULL},
             Pixel_Data[5]={0xE0,0x7F,0x10,0x00,NULL};

        int  _width=0,
             _height=0,
             _bits_allocated=0,
             _high_bit=0,
             bits_stored=0,
             max_width=512,
             max_height=512;
```

```
        long bytenumber=0,
             message_number=0,
             Pixel_Data_tag_found=0;

//          OpenDialog1->Filter = "Dicom files (*.dcm)|*.DCM|All
files (*.*)|*.*";

//          if (OpenDialog1->Execute())
        {
//              strcpy(source_file,OpenDialog1->FileName.c_str());
            strcpy(_source_file,source_file);
            if((source_file_handle=fopen(_source_file,"rb"))==NULL)
                ShowMessage("Error Opening Source file");
             else
             {
               while(1)
                    {
                     byte_read=fgetc(source_file_handle);
                     message[message_number]=byte_read;

                     if (bytenumber>=4)
                          {

                          if(
                              (message[0]==Rows[0])&&
                              (message[1]==Rows[1])&&
                              (message[2]==Rows[2])&&
                              (message[3]==Rows[3])&&
                              (Pixel_Data_tag_found==0)
                            )
                            {
                            //ShowMessage("Rows Tag found");
                            byte1=fgetc(source_file_handle);
                            byte2=fgetc(source_file_handle);
                            byte3=fgetc(source_file_handle);
                            byte4=fgetc(source_file_handle);
                            byte5=fgetc(source_file_handle);
                            byte6=fgetc(source_file_handle);
                            _width=((byte6 << 8) | byte5);
                            Edit2->Text = _width;
                            //ShowMessage("Width value read");
                            }

                         if(
                             (message[0]==Columns[0])&&
                             (message[1]==Columns[1])&&
                             (message[2]==Columns[2])&&
                             (message[3]==Columns[3])&&
                             (Pixel_Data_tag_found==0)
                           )
                           {
                            //ShowMessage("Columns Tag found");
                            byte1=fgetc(source_file_handle);
                            byte2=fgetc(source_file_handle);
                            byte3=fgetc(source_file_handle);
                            byte4=fgetc(source_file_handle);
                            byte5=fgetc(source_file_handle);
                            byte6=fgetc(source_file_handle);
                            _height=((byte6 << 8) | byte5);
                            Edit3->Text = _height;
                            //ShowMessage("Height value read");
```

```
    }

  if(
     (message[0]==Bits_Stored[0])&&
     (message[1]==Bits_Stored[1])&&
     (message[2]==Bits_Stored[2])&&
     (message[3]==Bits_Stored[3])&&
     (Pixel_Data_tag_found==0)
    )
    {
    Edit2->Text = bytenumber;
     //ShowMessage("Bits Stored Tag found");
     byte1=fgetc(source_file_handle);
     byte2=fgetc(source_file_handle);
     byte3=fgetc(source_file_handle);
     byte4=fgetc(source_file_handle);
     byte5=fgetc(source_file_handle);
     byte6=fgetc(source_file_handle);
     bits_stored=((byte6 << 8) | byte5);
     Edit7->Text = bits_stored;
     //ShowMessage("Bits Stored value read");
    }


  if(
     (message[0]==Bits_Allocated[0])&&
     (message[1]==Bits_Allocated[1])&&
     (message[2]==Bits_Allocated[2])&&
     (message[3]==Bits_Allocated[3])&&
     (Pixel_Data_tag_found==0)
    )
    {
     //ShowMessage("Bits Allocated Tag
found");

     byte1=fgetc(source_file_handle);
     byte2=fgetc(source_file_handle);
     byte3=fgetc(source_file_handle);
     byte4=fgetc(source_file_handle);
     byte5=fgetc(source_file_handle);
     byte6=fgetc(source_file_handle);
     _bits_allocated=((byte6 << 8) | byte5);
     Edit7->Text = _bits_allocated;
     //ShowMessage("Bits Allocated value
read");
    }


  if(
     (message[0]==High_Bit[0])&&
     (message[1]==High_Bit[1])&&
     (message[2]==High_Bit[2])&&
     (message[3]==High_Bit[3])&&
     (Pixel_Data_tag_found==0)
    )
    {
     //ShowMessage("High Bit Tag found");
     byte1=fgetc(source_file_handle);
     byte2=fgetc(source_file_handle);
     byte3=fgetc(source_file_handle);
     byte4=fgetc(source_file_handle);
     byte5=fgetc(source_file_handle);
```

```
                              byte6=fgetc(source_file_handle);
                              _high_bit=((byte6 << 8) | byte5);
                              Edit7->Text = _high_bit;
                              //ShowMessage("High Bit value read");
                             }

//PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPP
                           if(
                              (message[0]==Pixel_Data[0])&&
                              (message[1]==Pixel_Data[1])&&
                              (message[2]==Pixel_Data[2])&&
                              (message[3]==Pixel_Data[3])
                             )
                             {
                              Pixel_Data_tag_found=1;
                              //Edit2->Text = bytenumber;
                              //ShowMessage("Pixel Data Tag found");

                              //-------------
                              if (
                                 //((_width>0)&&(_width<=max_width))&&
//((_height>0)&&(_height<=max_height))
                                 ((_width>0)&&(_width<=512))&&
                                 ((_height>0)&&(_height<=512))
                                 )
                                 {
                                  //ShowMessage("Image size in
range");
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++

        int xp, yp;
        short Grayp,ch=0;
        TColor RGBp;

        Image1->Picture->Bitmap = new Graphics::TBitmap;
        Image1->Picture->Bitmap->PixelFormat = pf24bit;
        Image1->Picture->Bitmap->Width = _width;
        Image1->Picture->Bitmap->Height = _height;
        for (yp=0; yp<=_height-1; yp++)
                for (xp=0; xp<=_width-1; xp++)
                {
                        ch =
((fgetc(source_file_handle))|(fgetc(source_file_handle)<<8));
                        Grayp =ch>>3;
                        RGBp = (Graphics::TColor)((Grayp << 16) |
(Grayp << 8) | Grayp);
                        Image1->Canvas->Pixels[xp][yp] = RGBp;
                }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++
                                 }
                                 else
                                  ShowMessage("Image size in out of
range.\nMax(512x512)");

                              //-------------
                             }
```

```
                        message[0]=message[1];
                        message[1]=message[2];
                        message[2]=message[3];

                        message_number=3;
                       }
                    else
                        message_number++;

                    if(!feof(source_file_handle))
                        bytenumber++;
                    //else if (Pixel_Data_tag_found==1)
                         //break;
                    else
                        break;
                   }
                fclose(source_file_handle);
                //Edit2->Text = bytenumber; strcatstrcat
                //ShowMessage("Finished Opened Source file ");
            }

        }

}
          else
{
        char _source_file[400];
        FILE *source_file_handle;

        char byte_read,
             byte1=NULL,byte2=NULL,byte3=NULL,
             byte4=NULL,byte5=NULL,byte6=NULL,
             message[5],
             Rows[5]={0x28,0x00,0x10,0x00,NULL},
             Columns[5]={0x28,0x00,0x11,0x00,NULL},
             Bits_Allocated[5]={0x28,0x00,0x00,0x01,NULL},
             Bits_Stored[5]={0x28,0x00,0x01,0x01,NULL},
             High_Bit[5]={0x28,0x00,0x02,0x01,NULL},
             Pixel_Data[5]={0xE0,0x7F,0x10,0x00,NULL};

        int  _width=0,
             _height=0,
             _bits_allocated=0,
             _high_bit=0,
             bits_stored=0,
             max_width=512,
             max_height=512;

        long bytenumber=0,
             message_number=0,
             Pixel_Data_tag_found=0;

//        OpenDialog1->Filter = "Dicom files (*.dcm)|*.DCM|All
files (*.*)|*.*";

//        if (OpenDialog1->Execute())
        {
//             strcpy(source_file,OpenDialog1->FileName.c_str());
           strcpy(_source_file,source_file);
           if((source_file_handle=fopen(_source_file,"rb"))==NULL)
               ShowMessage("Error Opening Source file");
```

```
else
{
  while(1)
       {
        byte_read=fgetc(source_file_handle);
        message[message_number]=byte_read;

        if (bytenumber>=4)
             {

             if(
                 (message[0]==Rows[0])&&
                 (message[1]==Rows[1])&&
                 (message[2]==Rows[2])&&
                 (message[3]==Rows[3])&&
                 (Pixel_Data_tag_found==0)
               )
               {
               //ShowMessage("Rows Tag found");
               byte1=fgetc(source_file_handle);
               byte2=fgetc(source_file_handle);
               byte3=fgetc(source_file_handle);
               byte4=fgetc(source_file_handle);
               byte5=fgetc(source_file_handle);
               byte6=fgetc(source_file_handle);
               _width=((byte6 << 8) | byte5);
               Edit2->Text = _width;
               //ShowMessage("Width value read");
               }

            if(
                (message[0]==Columns[0])&&
                (message[1]==Columns[1])&&
                (message[2]==Columns[2])&&
                (message[3]==Columns[3])&&
                (Pixel_Data_tag_found==0)
              )
              {
               //ShowMessage("Columns Tag found");
               byte1=fgetc(source_file_handle);
               byte2=fgetc(source_file_handle);
               byte3=fgetc(source_file_handle);
               byte4=fgetc(source_file_handle);
               byte5=fgetc(source_file_handle);
               byte6=fgetc(source_file_handle);
               _height=((byte6 << 8) | byte5);
               Edit3->Text = _height;
               //ShowMessage("Height value read");
              }

            if(
                (message[0]==Bits_Stored[0])&&
                (message[1]==Bits_Stored[1])&&
                (message[2]==Bits_Stored[2])&&
                (message[3]==Bits_Stored[3])&&
                (Pixel_Data_tag_found==0)
              )
              {
              Edit2->Text = bytenumber;
               //ShowMessage("Bits Stored Tag found");
               byte1=fgetc(source_file_handle);
```

```
                  byte2=fgetc(source_file_handle);
                  byte3=fgetc(source_file_handle);
                  byte4=fgetc(source_file_handle);
                  byte5=fgetc(source_file_handle);
                  byte6=fgetc(source_file_handle);
                  bits_stored=((byte6 << 8) | byte5);
                  Edit7->Text = bits_stored;
                  //ShowMessage("Bits Stored value read");
                 }


                if(
                  (message[0]==Bits_Allocated[0])&&
                  (message[1]==Bits_Allocated[1])&&
                  (message[2]==Bits_Allocated[2])&&
                  (message[3]==Bits_Allocated[3])&&
                  (Pixel_Data_tag_found==0)
                 )
                 {
                  //ShowMessage("Bits Allocated Tag
found");

                  byte1=fgetc(source_file_handle);
                  byte2=fgetc(source_file_handle);
                  byte3=fgetc(source_file_handle);
                  byte4=fgetc(source_file_handle);
                  byte5=fgetc(source_file_handle);
                  byte6=fgetc(source_file_handle);
                  _bits_allocated=((byte6 << 8) | byte5);
                  Edit7->Text = _bits_allocated;
                  //ShowMessage("Bits Allocated value
read");
                 }


                if(
                  (message[0]==High_Bit[0])&&
                  (message[1]==High_Bit[1])&&
                  (message[2]==High_Bit[2])&&
                  (message[3]==High_Bit[3])&&
                  (Pixel_Data_tag_found==0)
                 )
                 {
                  //ShowMessage("High Bit Tag found");
                  byte1=fgetc(source_file_handle);
                  byte2=fgetc(source_file_handle);
                  byte3=fgetc(source_file_handle);
                  byte4=fgetc(source_file_handle);
                  byte5=fgetc(source_file_handle);
                  byte6=fgetc(source_file_handle);
                  _high_bit=((byte6 << 8) | byte5);
                  Edit7->Text = _high_bit;
                  //ShowMessage("High Bit value read");
                 }

//PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPP
                if(
                  (message[0]==Pixel_Data[0])&&
                  (message[1]==Pixel_Data[1])&&
                  (message[2]==Pixel_Data[2])&&
                  (message[3]==Pixel_Data[3])
```

```
                              )
                              {
                               Pixel_Data_tag_found=1;
                               //Edit2->Text = bytenumber;
                               //ShowMessage("Pixel Data Tag found");

                               //-------------
                               if (
                                  //((_width>0)&&(_width<=max_width))&&
//((_height>0)&&(_height<=max_height))
                                  ((_width>0)&&(_width<=512))&&
                                  ((_height>0)&&(_height<=512))
                                  )
                                  {
                                   //ShowMessage("Image size in
range");
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++

        int xp, yp;
        short Grayp,ch=0;
        TColor RGBp;

        Image1->Picture->Bitmap = new Graphics::TBitmap;
        Image1->Picture->Bitmap->PixelFormat = pf24bit;
        Image1->Picture->Bitmap->Width = _width;
        Image1->Picture->Bitmap->Height = _height;
        for (yp=0; yp<=_height-1; yp++)
                for (xp=0; xp<=_width-1; xp++)
                {
                        ch =
((fgetc(source_file_handle))|(fgetc(source_file_handle)<<8));
                        Grayp =ch>>3;
                        RGBp = (Graphics::TColor)((Grayp << 16) |
(Grayp << 8) | Grayp);
                        Image1->Canvas->Pixels[xp][yp] = RGBp;
                }
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
                                  }
                                  else
                                   ShowMessage("Image size in out of
range.\nMax(512x512)");

                               //-------------
                              }

                            message[0]=message[1];
                            message[1]=message[2];
                            message[2]=message[3];

                            message_number=3;
                          }
                        else
                           message_number++;

                        if(!feof(source_file_handle))
                           bytenumber++;
                        //else if (Pixel_Data_tag_found==1)
                             //break;
```

```
                    else
                       break;
                   }
               fclose(source_file_handle);
               //Edit2->Text = bytenumber; strcatstrcat
               //ShowMessage("Finished Opened Source file ");
            }

        }

}
      }
}
//---------------------------------------------------------------------
-----------


void __fastcall TForm1::Image1MouseMove(TObject *Sender,
TShiftState Shift,
      int X, int Y)
{
        Edit5->Text = X;
        Edit6->Text = Y;
        Edit7->Text = ((Image1->Picture->Bitmap->Canvas-
>Pixels[X][Y]));
        Edit10->Text = ((Image1->Picture->Bitmap->Canvas-
>Pixels[X][Y])/16777215.0)*255;
//         Image1->Canvas->Pixels[X][Y] = clWhite;
}
//---------------------------------------------------------------------
-----------




void __fastcall TForm1::Button3Click(TObject *Sender)
{//Darker
if((Image1->Height)!=249)
{
        char min_str[20],
             max_str[20];
        int x, y;
        BYTE Gray;
        TColor RGB;
        BYTE* LinePtr;


        strcpy(min_str,Edit8->Text.c_str());
        strcpy(max_str,Edit9->Text.c_str());

   for (y=0; y<=((Image1->Height)-1); y++)
   {
         //LinePtr=(BYTE*)Image1->Picture->Bitmap->ScanLine[y];
         for (x=0; x<=((Image1->Width)-1); x++)
         {
           Gray=(BYTE)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
                         //if((Gray >=0x4A)&&(Gray <=0x68))
                         //if((Gray >=atoi(min_str))&&(Gray
<=atoi(max_str)))
                            //Gray=Gray;
```

```
                        //else
                        if(Gray>=(10))
                           Gray=Gray-10;
                         RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                         Image1->Canvas->Pixels[x][y] = RGB;
          }
   }
}
else
               ShowMessage("No Image");

}
//------------------------------------------------------------------
-----------


void __fastcall TForm1::Button4Click(TObject *Sender)
{//Invert
if((Image1->Height)!=249)
{
        char min_str[20],
             max_str[20];
        int x, y;
        BYTE Gray;
        TColor RGB;
        BYTE* LinePtr;


        strcpy(min_str,Edit8->Text.c_str());
        strcpy(max_str,Edit9->Text.c_str());

   for (y=0; y<=((Image1->Height)-1); y++)
   {
         //LinePtr=(BYTE*)Image1->Picture->Bitmap->ScanLine[y];
         for (x=0; x<=((Image1->Width)-1); x++)
         {
           Gray=(BYTE)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
                        //if((Gray >=0x4A)&&(Gray <=0x68))
                        //if((Gray >=atoi(min_str))&&(Gray
<=atoi(max_str)))
                           //Gray=Gray;
                        //else
                           Gray=255-Gray;
                         RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                         Image1->Canvas->Pixels[x][y] = RGB;
         }
   }
}
else
               ShowMessage("No Image");

}
//------------------------------------------------------------------
-----------

void __fastcall TForm1::Button5Click(TObject *Sender)
{
if ((Image1->Height)!=249)
```

```
{
SaveDialog1->Filter = "BMP files (*.bmp)|*.BMP|All files
(*.*)|*.*";
        if (SaveDialog1->Execute())
        {

        Image1->Picture->Bitmap->SaveToFile(SaveDialog1-
>FileName.c_str());
                //ShowMessage("Image Saved as:\nc:Test.bmp");
        }
}
else
                ShowMessage("No Image");

}
//------------------------------------------------------------------
-----------


void __fastcall TForm1::Button7Click(TObject *Sender)
{//Filter to RED
if((Image1->Height)!=249)
{
        //char min_str[20],
             //max_str[20];
        int x, y,color_number;
        BYTE Gray;
        TColor RGB;
        BYTE* LinePtr;


        //strcpy(min_str,Edit8->Text.c_str());
        //strcpy(max_str,Edit9->Text.c_str());

        color_number=Edit18->Text.ToIntDef(0);
   for (y=0; y<=((Image1->Height)-1); y++)
   {
         for (x=0; x<=((Image1->Width)-1); x++)
         {
           //Gray=(BYTE)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
          Gray=(BYTE)(Mask[x][y]);
                       if(Gray==255)
                       {
                         if(color_number==0)
                        Image1->Canvas->Pixels[x][y] = clAqua;
                         else if(color_number==1)
                        Image1->Canvas->Pixels[x][y] = clBlue;
                         else if(color_number==3)
                        Image1->Canvas->Pixels[x][y] = clBlue;
                         else if(color_number==4)
                        Image1->Canvas->Pixels[x][y] = clFuchsia;
                         else if(color_number==5)
                        Image1->Canvas->Pixels[x][y] = clGreen;
                         else if(color_number==6)
                        Image1->Canvas->Pixels[x][y] = clLime;
                         else if(color_number==7)
                        Image1->Canvas->Pixels[x][y] = clMaroon;
                         else if(color_number==8)
                        Image1->Canvas->Pixels[x][y] = clNavy;
                         else if(color_number==9)
```

```
                              Image1->Canvas->Pixels[x][y] = clOlive;
                               else if(color_number==10)
                              Image1->Canvas->Pixels[x][y] = clPurple;
                               else if(color_number==11)
                              Image1->Canvas->Pixels[x][y] = clRed;
                               else if(color_number==12)
                              Image1->Canvas->Pixels[x][y] = clSilver;
                               else if(color_number==13)
                              Image1->Canvas->Pixels[x][y] = clTeal;
                               else if(color_number==14)
                              Image1->Canvas->Pixels[x][y] = clYellow;
                               else
                              Image1->Canvas->Pixels[x][y] = clRed;

                             }
                                //Gray=Gray;
                             //else
                                //Gray=0;
                              //RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                              //Image1->Canvas->Pixels[x][y] = RGB;
                 }
        }
}
else
                ShowMessage("No Image");

}
//-----------------------------------------------------------------
-----------


void __fastcall TForm1::Button10Click(TObject *Sender)
{//Lighter
if((Image1->Height)!=249)
{
        char min_str[20],
             max_str[20];
        int x, y;
        BYTE Gray;
        TColor RGB;
        BYTE* LinePtr;


        strcpy(min_str,Edit8->Text.c_str());
        strcpy(max_str,Edit9->Text.c_str());

   for (y=0; y<=((Image1->Height)-1); y++)
   {
         //LinePtr=(BYTE*)Image1->Picture->Bitmap->ScanLine[y];
         for (x=0; x<=((Image1->Width)-1); x++)
         {
           Gray=(BYTE)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
                             //if((Gray >=0x4A)&&(Gray <=0x68))
                             //if((Gray >=atoi(min_str))&&(Gray
<=atoi(max_str)))
                                //Gray=Gray;
                             //else
                             if(Gray<=(255-10))
                                Gray=Gray+10;
```

```
                        RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                        Image1->Canvas->Pixels[x][y] = RGB;
         }
   }
}
else
               ShowMessage("No Image");

}
//-----------------------------------------------------------------
-----------

void __fastcall TForm1::Button11Click(TObject *Sender)
{//Filter to RED
if((Image1->Height)!=249)
{
        char min_str[20],
             max_str[20];
        int x, y;
        BYTE Gray;
        TColor RGB;
        BYTE* LinePtr;


        strcpy(min_str,Edit8->Text.c_str());
        strcpy(max_str,Edit9->Text.c_str());

   for (y=0; y<=((Image1->Height)-1); y++)
   {
         for (x=0; x<=((Image1->Width)-1); x++)
         {
           Gray=(BYTE)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
                       if((Gray >=atoi(min_str))&&(Gray
<=atoi(max_str)))
                        Image1->Canvas->Pixels[x][y] = clRed;
                          //Gray=Gray;
                       //else
                          //Gray=0;
                        //RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                        //Image1->Canvas->Pixels[x][y] = RGB;
         }
   }
}
else
               ShowMessage("No Image");

}
//-----------------------------------------------------------------
-----------




void __fastcall TForm1::Button17Click(TObject *Sender)
{
```

```
//SaveDialog1->Filter = "Dicom files (*.dcm)|*.DCM|Img files
(*.img)|*.IMG|Bmp files (*.bmp)|*.BMP|All files (*.*)|*.*";
SaveDialog1->Filter = "STL files (*.stl)|*.STL|All files
(*.*)|*.*";
        if (SaveDialog1->Execute())
        {
//;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;
FILE *stream;
char stlfile[255],//="stlbin.stl",
     header[]={"STL file Created by RD NELL Pentech"};
float point[9],
      side[3],
      normal[3];
unsigned char colour[2],Gray_Mask;//RGB
TColor RGB;

int x,
    y,
    z;

strcpy(stlfile,SaveDialog1->FileName.c_str());
stream = fopen(stlfile, "wb+");
STLheader(header,2,stream);
//;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;
   for (y=Edit13->Text.ToIntDef(0); y<=Edit14->Text.ToIntDef(100);
y++)
         for (x=Edit11->Text.ToIntDef(0); x<=Edit12-
>Text.ToIntDef(100); x++)
         {
colour[0]=(unsigned char)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
colour[1]=(unsigned char)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>8);
colour[2]=(unsigned char)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>0);

normal[0]=0;
normal[1]=0;
normal[2]=0;

side[0]=0.390625;//xlenght of each side
side[1]=0.390625;//y
side[2]=0;//atof(Edit2->Text.c_str());//z

point[0]=x*side[0];
point[1]=y*side[1];//f = atof(str);
point[2]=Edit17->Text.ToIntDef(0);
//point[2]=atof(Edit3->Text.c_str());
//point[2]=Edit3->Text.c_str();

point[3]=point[0]+side[0];
point[4]=point[1];
point[5]=point[2];

point[6]=point[0]+side[0];;
point[7]=point[1]+side[1];
point[8]=point[2];

if(
```

```
   (colour[0]>=Edit8->Text.ToIntDef(0))&&
   (colour[0]<=Edit9->Text.ToIntDef(255))&&
   (colour[1]>=Edit8->Text.ToIntDef(0))&&
   (colour[1]<=Edit9->Text.ToIntDef(255))&&
   (colour[2]>=Edit8->Text.ToIntDef(0))&&
   (colour[2]<=Edit9->Text.ToIntDef(255))
  )
{
           Gray_Mask=(BYTE)(Mask[x][y]);
                     if(Gray_Mask==255)
STLtriangle(point,normal,colour,stream);
}
point[0]=point[0];
point[1]=point[1];
point[2]=point[2];

point[3]=point[0]+side[0];;
point[4]=point[1]+side[1];
point[5]=point[2];

point[6]=point[0];
point[7]=point[1]+side[1];
point[8]=point[2];

//colour[0]=31;colour[1]=0;colour[2]=0;
if(
   (colour[0]>=Edit8->Text.ToIntDef(0))&&
   (colour[0]<=Edit9->Text.ToIntDef(255))&&
   (colour[1]>=Edit8->Text.ToIntDef(0))&&
   (colour[1]<=Edit9->Text.ToIntDef(255))&&
   (colour[2]>=Edit8->Text.ToIntDef(0))&&
   (colour[2]<=Edit9->Text.ToIntDef(255))
  )
{
           Gray_Mask=(BYTE)(Mask[x][y]);
                     if(Gray_Mask==255)
STLtriangle(point,normal,colour,stream);
}
           colour[0]=255-colour[0];
           colour[1]=255-colour[1];
           colour[2]=255-colour[2];
        RGB = (Graphics::TColor)((colour[0] << 16) | (colour[1] <<
8) | colour[0]);
           Image1->Canvas->Pixels[x][y] = RGB;
         }

fclose(stream);
//;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;
        }

}
//----------------------------------------------------------------
-----------


void __fastcall TForm1::Button19Click(TObject *Sender)
{
char source1[256],
     source2[256],
     destination[256];
```

```
    //mfile2_byte_number_str[80];

long mfile2_byte_number=85;

ShowMessage("Add two STL files together\nSelect first source
file");
OpenDialog1->Filter = "STL files (*.stl)|*.STL|Dicom files
(*.dcm)|*.DCM|Img files (*.img)|*.IMG|Bmp files (*.bmp)|*.BMP|All
files (*.*)|*.*";
        if (OpenDialog1->Execute())
            {
             strcpy(source1,OpenDialog1->FileName.c_str());
             ShowMessage(source1);
            }

ShowMessage("Add two STL files together\nSelect second source
file");
OpenDialog1->Filter = "STL files (*.stl)|*.STL|Dicom files
(*.dcm)|*.DCM|Img files (*.img)|*.IMG|Bmp files (*.bmp)|*.BMP|All
files (*.*)|*.*";
        if (OpenDialog1->Execute())
            {
             strcpy(source2,OpenDialog1->FileName.c_str());
             ShowMessage(source2);
            }


ShowMessage("Add two STL files together\nEnter Destination file");
OpenDialog1->Filter = "STL files (*.stl)|*.STL|Dicom files
(*.dcm)|*.DCM|Img files (*.img)|*.IMG|Bmp files (*.bmp)|*.BMP|All
files (*.*)|*.*";
        if (OpenDialog1->Execute())
            {
             strcpy(destination,OpenDialog1->FileName.c_str());
             ShowMessage(destination);
            }


if(
   file_copy(source1,source2,destination,mfile2_byte_number)==0
  )
  ShowMessage("STL files successfully added");
else
ShowMessage("Error Addind STL files");


}
//-----------------------------------------------------------------
-----------

void __fastcall TForm1::Button20Click(TObject *Sender)
{
        SaveDialog1->Filter = "STL files (*.stl)|*.STL|All files
(*.*)|*.*";
        if (SaveDialog1->Execute())
        {
                FILE *stream;
                char stlfile[255],
                header[]={"STL file Created by RD NELL Pentech"};
                float point[9],
                side[3],
```

```
                normal[3];
                unsigned char colour[2],
                              Centre,
                              Left,
                              Right,
                              Top,
                              Bottom,//RGB
                              Mask_Centre,
                              Mask_Left,
                              Mask_Right,
                              Mask_Top,
                              Mask_Bottom;//RGB
                TColor RGB;

                int x,
                    y,
                    z;

                strcpy(stlfile,SaveDialog1->FileName.c_str());
                stream = fopen(stlfile, "wb+");
                STLheader(header,2,stream);

                //for (y=Edit13->Text.ToIntDef(0)-1; y<=Edit14-
>Text.ToIntDef(100)-1; y++)
                //for (x=Edit11->Text.ToIntDef(0)-1; x<=Edit12-
>Text.ToIntDef(100)-1; x++)
                for (y=Edit13->Text.ToIntDef(0); y<=Edit14-
>Text.ToIntDef(100)-1; y++)
                for (x=Edit11->Text.ToIntDef(0); x<=Edit12-
>Text.ToIntDef(100)-1; x++)
                {
                        Centre   =(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x][y])>>16);
                        Left     =(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x-1][y])>>16);
                        Right    =(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x+1][y])>>16);
                        Top      =(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x][y-1])>>16);
                        Bottom   =(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x][y+1])>>16);

                        Mask_Centre=(BYTE)(Mask[x][y]);
                        Mask_Left=(BYTE)(Mask[x-1][y]);
                        Mask_Right=(BYTE)(Mask[x+1][y]);
                        Mask_Top=(BYTE)(Mask[x][y-1]);
                        Mask_Bottom=(BYTE)(Mask[x][y+1]);


                        colour[0]=(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x][y])>>16);
                        colour[1]=(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x][y])>>8);
                        colour[2]=(unsigned char)((Image1-
>Picture->Bitmap->Canvas->Pixels[x][y])>>0);

                        normal[0]=0;
                        normal[1]=0;
                        normal[2]=0;

                        side[0]=0.390625;//x lenght of each side
```

```
                              side[1]=0.390625;//y lenght of each side
                              side[2]=5;//atof(Edit2->Text.c_str());//z
lenght of each side


                              if(
                                 (colour[0]>=Edit8->Text.ToIntDef(0))&&
                                 (colour[0]<=Edit9-
>Text.ToIntDef(255))&&
                                 (colour[1]>=Edit8->Text.ToIntDef(0))&&
                                 (colour[1]<=Edit9-
>Text.ToIntDef(255))&&
                                 (colour[2]>=Edit8->Text.ToIntDef(0))&&
                                 (colour[2]<=Edit9->Text.ToIntDef(255))
                                 )
                                 {

//EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
                                       if(
                                          (Mask_Centre>=200)&&
                                          (Mask_Left<=100)
                                          )
                                          {
                                                point[0]=x*side[0];
                                                point[1]=y*side[1];//f =
atof(str);
                                                point[2]=Edit17-
>Text.ToIntDef(0);


                                                point[3]=point[0];
                                                point[4]=point[1]+side[1];
                                                point[5]=point[2];

                                                point[6]=point[0];
                                                point[7]=point[1]+side[1];
                                                point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                                point[0]=point[0];
                                                point[1]=point[1];//f =
atof(str);
                                                point[2]=point[2];//Edit3-
>Text.ToIntDef(0);


                                                point[3]=point[0];
                                                point[4]=point[1]+side[1];
                                                point[5]=point[2]+side[2];

                                                point[6]=point[0];
                                                point[7]=point[1];
                                                point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                                colour[0]=255-colour[0];
                                                colour[1]=255-colour[1];
                                                colour[2]=255-colour[2];
                                                //RGB =
(Graphics::TColor)((colour[0] << 16) | (colour[1] << 8) |
colour[0]);
```

```
                                        //Image1->Canvas-
>Pixels[x][y] = RGB;
                                    }
                                if(
                                    (Mask_Centre>=200)&&
                                    (Mask_Right<=100)
                                    )
                                    {
point[0]=(x*side[0])+side[0];
                                        point[1]=y*side[1];//f =
atof(str);
                                        point[2]=Edit17-
>Text.ToIntDef(0);

                                        point[3]=point[0];
                                        point[4]=point[1]+side[1];
                                        point[5]=point[2];

                                        point[6]=point[0];
                                        point[7]=point[1]+side[1];
                                        point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                        point[0]=point[0];
                                        point[1]=point[1];//f =
atof(str);
                                        point[2]=point[2];//Edit3-
>Text.ToIntDef(0);

                                        point[3]=point[0];
                                        point[4]=point[1]+side[1];
                                        point[5]=point[2]+side[2];

                                        point[6]=point[0];
                                        point[7]=point[1];
                                        point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                        colour[0]=255-colour[0];
                                        colour[1]=255-colour[1];
                                        colour[2]=255-colour[2];
                                        //RGB =
(Graphics::TColor)((colour[0] << 16) | (colour[1] << 8) |
colour[0]);
                                        //Image1->Canvas-
>Pixels[x][y] = RGB;
                                    }
                                if(
                                    (Mask_Centre>=200)&&
                                    (Mask_Top<=100)
                                    )
                                    {
                                        point[0]=x*side[0];
                                        point[1]=y*side[1];//f =
atof(str);
                                        point[2]=Edit17-
>Text.ToIntDef(0);
```

```
                                          point[3]=point[0]+side[0];
                                          point[4]=point[1];
                                          point[5]=point[2];

                                          point[6]=point[0]+side[0];
                                          point[7]=point[1];
                                          point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                          point[0]=point[0];
                                          point[1]=point[1];//f =
atof(str);
                                          point[2]=point[2];//Edit3-
>Text.ToIntDef(0);

                                          point[3]=point[0]+side[0];
                                          point[4]=point[1];
                                          point[5]=point[2]+side[2];

                                          point[6]=point[0];
                                          point[7]=point[1];
                                          point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                          //colour[0]=255-colour[0];
                                          //colour[1]=255-colour[1];
                                          //colour[2]=255-colour[2];
                                          //RGB =
(Graphics::TColor)((colour[0] << 16) | (colour[1] << 8) |
colour[0]);
                                          //Image1->Canvas-
>Pixels[x][y] = RGB;
                                     }
                                 if(
                                     (Mask_Centre>=200)&&
                                     (Mask_Bottom<=100)
                                     )
                                     {
                                          point[0]=x*side[0];

point[1]=(y*side[1])+side[1];//f = atof(str);
                                          point[2]=Edit17-
>Text.ToIntDef(0);

                                          point[3]=point[0]+side[0];
                                          point[4]=point[1];
                                          point[5]=point[2];

                                          point[6]=point[0]+side[0];
                                          point[7]=point[1];
                                          point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                          point[0]=point[0];
                                          point[1]=point[1];//f =
atof(str);
                                          point[2]=point[2];//Edit3-
>Text.ToIntDef(0);
```

```
                                        point[3]=point[0]+side[0];
                                        point[4]=point[1];
                                        point[5]=point[2]+side[2];

                                        point[6]=point[0];
                                        point[7]=point[1];
                                        point[8]=point[2]+side[2];

STLtriangle(point,normal,colour,stream);

                                        //colour[0]=255-colour[0];
                                        //colour[1]=255-colour[1];
                                        //colour[2]=255-colour[2];
                                        //RGB =
(Graphics::TColor)((colour[0] << 16) | (colour[1] << 8) |
colour[0]);
                                        //Image1->Canvas-
>Pixels[x][y] = RGB;
                                   }

//EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
                              }
                        //colour[0]=255-colour[0];
                        //colour[1]=255-colour[1];
                        //colour[2]=255-colour[2];
                        //RGB = (Graphics::TColor)((colour[0] <<
16) | (colour[1] << 8) | colour[0]);
                        //Image1->Canvas->Pixels[x][y] = RGB;
                  }
        fclose(stream);
        }
}
//---------------------------------------------------------------
--


void __fastcall TForm1::Button21Click(TObject *Sender)
{//Mask
        if((Image1->Height)!=249)
          {
                int x, y;

                for (y=0; y<=((Image1->Height)-1); y++)
                        for (x=0; x<=((Image1->Width)-1); x++)
                                Mask[x][y]   =(unsigned
char)((Image1->Picture->Bitmap->Canvas->Pixels[x][y])>>16);
                                //Image1->Canvas->Pixels[x][y] =
clRed;                         }
          }
        else
               ShowMessage("No Image");

}
//---------------------------------------------------------------
--


void __fastcall TForm1::Button8Click(TObject *Sender)
{
        BYTE ImageData[700][700]={0}, *LinePtr;
```

```
        BYTE Output[700][700]={255};
        //BYTE Gray;
        int x,
            y;
        BYTE Gray,
             value,
            //filter_value,
            //min_filter_value,
            //max_filter_value,
            difference=1,
            TOPLEFT,
            TOP,
            TOPRIGHT,
            LEFT,
            CENTER,
            RIGHT,
            BOTTOMLEFT,
            BOTTOM,
            BOTTOMRIGHT;

        TColor RGB;
//difference=Edit10->Text.ToIntDef(1);
//colour[1]<=Edit9->Text.ToIntDef(255)
            //min_filter_value=80;
            //max_filter_value=180;
            //min_filter_value=(BYTE)Edit8->Text.ToIntDef(80);
            //max_filter_value=(BYTE)Edit9->Text.ToIntDef(180);;
//.................................................................
..
        // Copy the image to ImageData
        for (y=0; y<=((Image1->Height)-1); y++)
                for (x=0; x<=((Image1->Width)-1); x++)
                {
                        Gray=(BYTE)((Image1->Picture->Bitmap-
>Canvas->Pixels[x][y])>>16);
                        //if(
                      //        (Gray>=min_filter_value)&&
                      //        (Gray<=max_filter_value)
                        //    )
                                //ImageData[x][y] =
min_filter_value;
                                ImageData[x][y] =
Gray;//min_filter_value;
                }
//.................................................................
..
   for (y=1; y<=((Image1->Height)-2); y++)
         for (x=1; x<=((Image1->Width)-2); x++)
      {
       {
        TOPLEFT=ImageData[x-1][y-1];
        TOP=ImageData[x][y-1];
        TOPRIGHT=ImageData[x+1][y-1];
        LEFT=ImageData[x-1][y];
        CENTER=ImageData[x][y];
        RIGHT=ImageData[x+1][y];
        BOTTOMLEFT=ImageData[x-1][y+1];
        BOTTOM=ImageData[x][y+1];
        BOTTOMRIGHT=ImageData[x+1][y+1];
        if(
           (abs(TOP-CENTER)>=difference)
```

```
        )
        Output[x][y]=CENTER;
      if(
         (abs(LEFT-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
      if(
         (abs(RIGHT-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
      if(
         (abs(BOTTOM-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
      if(
         (abs(TOPLEFT-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
      if(
         (abs(TOPRIGHT-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
      if(
         (abs(BOTTOMLEFT-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
      if(
         (abs(BOTTOMRIGHT-CENTER)>=difference)
        )
        Output[x][y]=CENTER;
     }
    //}
}
//...................................................................
..
        //Output the Image
   for (y=0; y<=((Image1->Height)-1); y++)
        for (x=0; x<=((Image1->Width)-1); x++)
        {
                        Gray=Output[x][y];
                        //if(Gray!=0)
                      RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                      //else
                      //RGB=clGreen;//clGreen
                      Image1->Canvas->Pixels[x][y] = RGB;
        }
//...................................................................
..
}
//-------------------------------------------------------------------
--


void __fastcall TForm1::Button15Click(TObject *Sender)
{//Change the RED filtered Image to Black and White
if((Image1->Height)!=249)
{
        char min_str[20],
             max_str[20];
        int x, y;
```

```
        BYTE Gray;
        TColor RGB,RGB1;
        BYTE* LinePtr;


        strcpy(min_str,Edit8->Text.c_str());
        strcpy(max_str,Edit9->Text.c_str());

   for (y=0; y<=((Image1->Height)-1); y++)
   {
         for (x=0; x<=((Image1->Width)-1); x++)
         {
           //Gray=(BYTE)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
                          RGB1=Image1->Canvas->Pixels[x][y];
                         //if((Gray >=atoi(min_str))&&(Gray
<=atoi(max_str)))
                         if(RGB1==clRed)
                          Image1->Canvas->Pixels[x][y] = clWhite;
                         else
                          Image1->Canvas->Pixels[x][y] = clBlack;
                            //Gray=Gray;
                         //else
                            //Gray=0;
                          //RGB = (Graphics::TColor)((Gray << 16) |
(Gray << 8) | Gray);
                          //Image1->Canvas->Pixels[x][y] = RGB;
         }
   }
}
else
               ShowMessage("No Image");

}
//-----------------------------------------------------------------
--

void __fastcall TForm1::Button14Click(TObject *Sender)
{
//SaveDialog1->Filter = "Dicom files (*.dcm)|*.DCM|Img files
(*.img)|*.IMG|Bmp files (*.bmp)|*.BMP|All files (*.*)|*.*";
SaveDialog1->Filter = "STL files (*.stl)|*.STL|All files
(*.*)|*.*";
        if (SaveDialog1->Execute())
        {
//;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
FILE *stream;
char stlfile[255],//="stlbin.stl",
     header[]={"STL file Created by RD NELL Pentech"};
float point[9],yzpoint[9],
      side[3],
      normal[3],
      temp1;
unsigned char colour[2],Gray_Mask;//RGB
TColor RGB;

int x,
    y,
    z;
```

```
strcpy(stlfile,SaveDialog1->FileName.c_str());
stream = fopen(stlfile, "wb+");
STLheader(header,2,stream);
//;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;
   for (y=Edit13->Text.ToIntDef(0); y<=Edit14->Text.ToIntDef(100);
y++)
         for (x=Edit11->Text.ToIntDef(0); x<=Edit12-
>Text.ToIntDef(100); x++)
         {
colour[0]=(unsigned char)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>16);
colour[1]=(unsigned char)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>8);
colour[2]=(unsigned char)((Image1->Picture->Bitmap->Canvas-
>Pixels[x][y])>>0);

normal[0]=0;
normal[1]=0;
normal[2]=0;

side[0]=0.390625;//xlenght of each side
side[1]=0.390625;//y
side[2]=0;//atof(Edit2->Text.c_str());//z

point[0]=x*side[0];
point[1]=y*side[1];//f = atof(str);
point[2]=Edit17->Text.ToIntDef(0);
//point[2]=atof(Edit3->Text.c_str());
//point[2]=Edit3->Text.c_str();

point[3]=point[0]+side[0];
point[4]=point[1];
point[5]=point[2];

point[6]=point[0]+side[0];;
point[7]=point[1]+side[1];
point[8]=point[2];

yzpoint[0]=point[6];
yzpoint[1]=point[7];
yzpoint[2]=point[8];

yzpoint[3]=point[3];
yzpoint[4]=point[4];
yzpoint[5]=point[5];

yzpoint[6]=point[0];
yzpoint[7]=point[1];
yzpoint[8]=point[2];


if(
   (colour[0]>=Edit8->Text.ToIntDef(0))&&
   (colour[0]<=Edit9->Text.ToIntDef(255))&&
   (colour[1]>=Edit8->Text.ToIntDef(0))&&
   (colour[1]<=Edit9->Text.ToIntDef(255))&&
   (colour[2]>=Edit8->Text.ToIntDef(0))&&
   (colour[2]<=Edit9->Text.ToIntDef(255))
  )
{
```

```
            Gray_Mask=(BYTE)(Mask[x][y]);
                          if(Gray_Mask==255)
STLtriangle(yzpoint,normal,colour,stream);
}
yzpoint[0]=point[6];
yzpoint[1]=point[7];
yzpoint[2]=point[8];

yzpoint[3]=point[3];
yzpoint[4]=point[4];
yzpoint[5]=point[5];

yzpoint[6]=point[0];
yzpoint[7]=point[1];
yzpoint[8]=point[2];


//colour[0]=31;colour[1]=0;colour[2]=0;
if(
   (colour[0]>=Edit8->Text.ToIntDef(0))&&
   (colour[0]<=Edit9->Text.ToIntDef(255))&&
   (colour[1]>=Edit8->Text.ToIntDef(0))&&
   (colour[1]<=Edit9->Text.ToIntDef(255))&&
   (colour[2]>=Edit8->Text.ToIntDef(0))&&
   (colour[2]<=Edit9->Text.ToIntDef(255))
  )
{
            Gray_Mask=(BYTE)(Mask[x][y]);
                          if(Gray_Mask==255)
STLtriangle(yzpoint,normal,colour,stream);
}
            colour[0]=255-colour[0];
            colour[1]=255-colour[1];
            colour[2]=255-colour[2];
          //RGB = (Graphics::TColor)((colour[0] << 16) | (colour[1]
<< 8) | colour[0]);
            //Image1->Canvas->Pixels[x][y] = RGB;
          }

fclose(stream);
//;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
         }

}
//-------------------------------------------------------------------
--

void __fastcall TForm1::Button18Click(TObject *Sender)
{
int start[2],
    x,y,
    xp,yp,
    pix[10]={0};

long pixels=0l,n;

BYTE Gray;

TColor edge_tracker_RGB=clFuchsia;
```

```
for(y=0;y<511;y++)
 for(x=0;x<511;x++)
   ImageData1[x][y] = 0;

//.................................................................
..
        // Copy the image to ImageData
        for (y=1; y<=((Image1->Height)-1); y++)
                for (x=1; x<=((Image1->Width)-1); x++)
                {
                        Gray=(BYTE)((Image1->Picture->Bitmap-
>Canvas->Pixels[x][y])>>16);
                        if(Gray==255)
                           ImageData1[x][y] = Gray;
                        else
                           ImageData1[x][y] = 0;
                }
//.................................................................
..
get_start_point(start);
xp=start[0];
yp=start[1];
Image1->Canvas->Pixels[xp][yp] = edge_tracker_RGB;
//.................................................................
..
        // Count the white pixels
        for (y=0; y<=((Image1->Height)-1); y++)
                for (x=0; x<=((Image1->Width)-1); x++)
                {
                        //Gray=(BYTE)((Image1->Picture->Bitmap-
>Canvas->Pixels[x][y])>>16);
                        //if(Gray==255)
                        if(ImageData1[x][y]==255)
                        pixels++;
                }
//.................................................................
..
//get_art_pixels(xp,yp,pix);
//.................................................................
..
for(n=pixels;n>01;n--)
{
        //delayms(200);
        get_art_pixels(xp,yp,pix);
        if(pix[1]==1)
        {
                ImageData1[xp-1][yp-1]=14;
                //putpixel(xp-1,yp-1,6);
                Image1->Canvas->Pixels[xp-1][yp-1] =
edge_tracker_RGB;
                xp=xp-1;
                yp=yp-1;
        }
        else if(pix[2]==1)
        {
                ImageData1[xp][yp-1]=14;
                //putpixel(xp,yp-1,6);
                Image1->Canvas->Pixels[xp][yp-1] =
edge_tracker_RGB;
                xp=xp;
                yp=yp-1;
```

```
        }
        else if(pix[3]==1)
        {
                ImageData1[xp+1][yp-1]=14;
                //putpixel(xp+1,yp-1,6);
                Image1->Canvas->Pixels[xp+1][yp-1] =
edge_tracker_RGB;
                xp=xp+1;
                yp=yp-1;
        }
        else if(pix[4]==1)
        {
                ImageData1[xp-1][yp]=14;
                //putpixel(xp-1,yp,6);
                Image1->Canvas->Pixels[xp-1][yp] =
edge_tracker_RGB;
                xp=xp-1;
                yp=yp;
        }
        else if(pix[6]==1)
        {
                ImageData1[xp+1][yp]=14;
                //putpixel(xp+1,yp,6);
                Image1->Canvas->Pixels[xp+1][yp] =
edge_tracker_RGB;
                xp=xp+1;
                yp=yp;
        }
        else if(pix[7]==1)
        {
                ImageData1[xp-1][yp+1]=14;
                //putpixel(xp-1,yp+1,6);
                Image1->Canvas->Pixels[xp-1][yp+1] =
edge_tracker_RGB;
                xp=xp-1;
                yp=yp+1;
        }
        else if(pix[8]==1)
        {
                ImageData1[xp][yp+1]=14;
                //putpixel(xp,yp+1,6);
                Image1->Canvas->Pixels[xp][yp+1] =
edge_tracker_RGB;
                xp=xp;
                yp=yp+1;
        }
        else if(pix[9]==1)
        {
                ImageData1[xp+1][yp+1]=14;
                //putpixel(xp+1,yp+1,6);
                Image1->Canvas->Pixels[xp+1][yp+1] =
edge_tracker_RGB;
                xp=xp+1;
                yp=yp+1;
        }
        //get_art_pixels(xp,yp,pix);
}
//.................................................................
..
}
```

```
//------------------------------------------------------------------
--

void __fastcall TForm1::Button12Click(TObject *Sender)
{
ShowMessage("Open Source STL file");
OpenDialog1->Filter = "STL files (*.stl)|*.STL|All files
(*.*)|*.*";
if (OpenDialog1->Execute())
    ShowMessage(OpenDialog1->FileName.c_str());
///*
//char oldname[100],
//      newname[100];
float x0=Edit15->Text.ToIntDef(0),
      y0=Edit16->Text.ToIntDef(0),
      z0=Edit17->Text.ToIntDef(0);
int newcolor=0,
    change_color=0;
FILE *fold, *fnew;
unsigned char buf[100];
float value;
struct facet facet_change;
//------------------------------------------------------------------
--
//strcpy(oldname,"MAZEX.STL");
//strcpy(newname,"MAZEX1.STL");
//------------------------------------------------------------------
--
//Opening of files
//Open source file
if ((fold=fopen(OpenDialog1->FileName.c_str(),"rb"))==NULL)
ShowMessage("Error opening source STL file");
else
   {
ShowMessage("Open Destination STL file");
SaveDialog1->Filter = "STL files (*.stl)|*.STL|All files
(*.*)|*.*";
  if (SaveDialog1->Execute())
ShowMessage(SaveDialog1->FileName.c_str());
    //Open destination file
    if ((fnew=fopen(SaveDialog1->FileName.c_str(),"wb+"))==NULL)
    {
     fclose(fold);
     ShowMessage("Error opening destination STL file");
    }
    else
       {
      //------------------------------------------------------------
--
      //Copy the header from oldfile
      fread(buf, 84, 1, fold);
      //Write the haderto newfile
      fwrite(buf, 84, 1, fnew);
      //------------------------------------------------------------
--
      //Read the facets
      while(!feof(fold))
           {
            //----------------------------------------------------
--
            //if(feof(fold))
```

```
//break;
//else
{
 fread(&facet_change.normal_x, 4, 1, fold);
 facet_change.normal_x=facet_change.normal_x+x0;
 fwrite(&facet_change.normal_x, 4, 1, fnew);
}
//-----------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.normal_y, 4, 1, fold);
 facet_change.normal_y=facet_change.normal_y+y0;
 fwrite(&facet_change.normal_y, 4, 1, fnew);
}
//-----------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.normal_z, 4, 1, fold);
 facet_change.normal_z=facet_change.normal_z+z0;
 fwrite(&facet_change.normal_z, 4, 1, fnew);
}
//-----------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex1_x, 4, 1, fold);
 facet_change.vertex1_x=facet_change.vertex1_x+x0;
 fwrite(&facet_change.vertex1_x, 4, 1, fnew);
}
//-----------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex1_y, 4, 1, fold);
 facet_change.vertex1_y=facet_change.vertex1_y+y0;
 fwrite(&facet_change.vertex1_y, 4, 1, fnew);
}
//-----------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex1_z, 4, 1, fold);
 facet_change.vertex1_z=facet_change.vertex1_z+z0;
 fwrite(&facet_change.vertex1_z, 4, 1, fnew);
}
//-----------------------------------------------------
--
//if(feof(fold))
//break;
```

```
//else
{
 fread(&facet_change.vertex2_x, 4, 1, fold);
 facet_change.vertex2_x=facet_change.vertex2_x+x0;
 fwrite(&facet_change.vertex2_x, 4, 1, fnew);
}
//------------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex2_y, 4, 1, fold);
 facet_change.vertex2_y=facet_change.vertex2_y+y0;
 fwrite(&facet_change.vertex2_y, 4, 1, fnew);
}
//------------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex2_z, 4, 1, fold);
 facet_change.vertex2_z=facet_change.vertex2_z+z0;
 fwrite(&facet_change.vertex2_z, 4, 1, fnew);
}
//------------------------------------------------------
--
//------------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex3_x, 4, 1, fold);
 facet_change.vertex3_x=facet_change.vertex3_x+x0;
 fwrite(&facet_change.vertex3_x, 4, 1, fnew);
}
//------------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex3_y, 4, 1, fold);
 facet_change.vertex3_y=facet_change.vertex3_y+y0;
 fwrite(&facet_change.vertex3_y, 4, 1, fnew);
}
//------------------------------------------------------
--
//if(feof(fold))
//break;
//else
{
 fread(&facet_change.vertex3_z, 4, 1, fold);
 facet_change.vertex3_z=facet_change.vertex3_z+z0;
 fwrite(&facet_change.vertex3_z, 4, 1, fnew);
}
//------------------------------------------------------
--
//if(feof(fold))
```

```
            //break;
            //else
            {
             fread(&facet_change.color, 2, 1, fold);
             if(change_color==1)
               facet_change.color=newcolor;
             fwrite(&facet_change.color, 2, 1, fnew);
            }
            //-----------------------------------------------------
--
           }
      //-----------------------------------------------------------
--
      fclose(fnew);
      fclose(fold);
       }
   }
//*/
}
//-----------------------------------------------------------------
--
```

13.2) Software code written in C

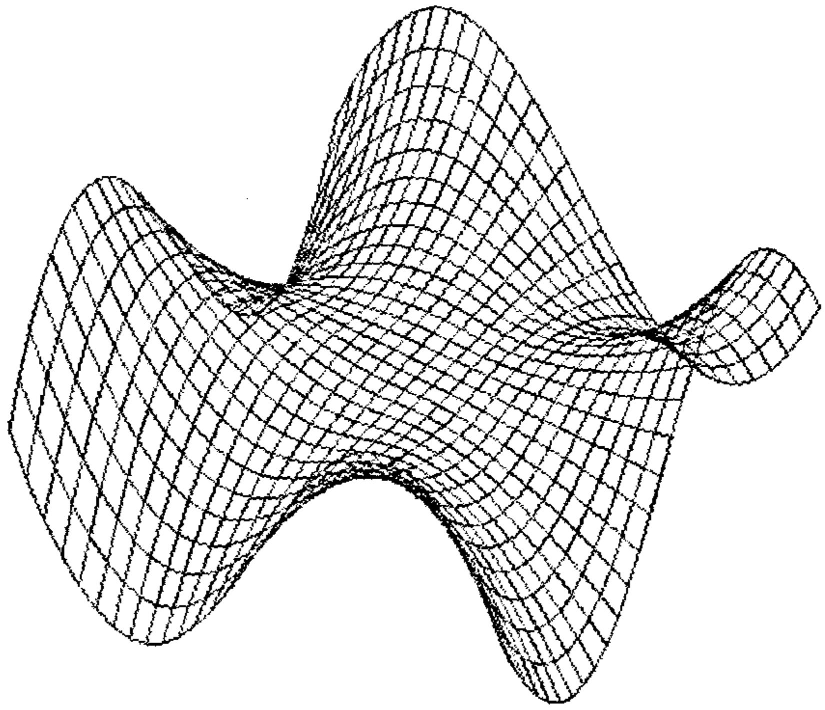
```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <io.h>
#include <alloc.h>
#include <fcntl.h>
#include <process.h>
#include <sys\stat.h>
#include <math.h>
#include <graphics.h>
```

```
void setup_graphics(void)
{
// request auto detection
int gdriver = DETECT, gmode, errorcode;

// initialize graphics and local variables
initgraph(&gdriver, &gmode, "C:\\lang\\tc\\bin");

// read result of initialization
errorcode = graphresult();
if (errorcode != grOk)  // an error occurred
 {
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch();
  exit(1); // terminate with an error code
 }
}
//-----------------
void stl_line(float x1,float y1,float z1,float x2,float y2,float z2,FILE* stream)
{
float normal[3],
    vertex1[3],
    vertex2[3],
    vertex3[3];
    //line_thickness=0;
    //z1=0,
    //z2=0;//,x1=1,x2=2,x3=3,y1=0,y2=1,y3=1,z1=1,z2=0,z3=2;

unsigned int c,colour;

normal[0]=0;
normal[1]=0;
normal[2]=0;

vertex1[0]=x1;
vertex1[1]=y1;
vertex1[2]=z1;

vertex2[0]=x2;
vertex2[1]=y2;
vertex2[2]=z2;

vertex3[0]=x1;
vertex3[1]=y1;//+line_thickness;
vertex3[2]=z1;

c=random(3);
```

```
if(c==0)
colour=0xFC00;
else if(c==1)
colour=0x83E0;
else
colour=0x801F;

fwrite(normal, sizeof(normal), 1, stream);
fwrite(vertex1, sizeof(vertex1), 1, stream);
fwrite(vertex2, sizeof(vertex2), 1, stream);
fwrite(vertex3, sizeof(vertex3), 1, stream);
fwrite(&colour, sizeof(colour), 1, stream);
}

//-----------------
void main(void)
{
float x,y,z,x1,y1,z1,x2,y2,z2,start,end,inc;

FILE *stream;
char stlfile[]="stlbin.stl",
   header[80]={0x20};
unsigned long  number_of_trangles,point=1;

//--------------------
setup_graphics();
//--------------------
// open a file for update
stream = fopen(stlfile, "wb+");
//-------------------------------------------------------------
fwrite(header, sizeof(header), 1, stream);
//-------------------------------------------------------------
number_of_trangles=2;
fwrite(&number_of_trangles, sizeof(number_of_trangles), 1, stream);
//----------------
start=-1.5;
end=1.5,
inc=0.1;

for(y=start;y<=end;y=y+inc)
{
for(x=start;x<=end;x=x+inc)
  {
  z=x*y*(x-y)*(x+y)/sqrt(x*x+y*y);
  x1=x2;
  y1=y2;
  z1=z2;
  x2=x;
  y2=y;
  z2=z;
```

```
    point++;
    if(point>=3)
    //stl_line(x1,y1,0,x2,y2,0,stream);
    stl_line(x1,y1,z1,x2,y2,z2,stream);
   }
point=1;
}

//for(y=-1;y<=1;y=y+0.1)
for(x=start;x<=end;x=x+inc)
{
//for(x=-1;x<=1;x=x+0.1)
for(y=start;y<=end;y=y+inc)
   {
    z=x*y*(x-y)*(x+y)/sqrt(x*x+y*y);
    x1=x2;
    y1=y2;
    z1=z2;
    x2=x;
    y2=y;
    z2=z;
    point++;
    if(point>=3)
    //stl_line(x1,y1,0,x2,y2,0,stream);
    stl_line(x1,y1,z1,x2,y2,z2,stream);
   }
point=1;
}

//stl_line(0,0,0,1,1,1,stream);
//stl_line(10,5,9,6.8,9,12,stream);
//------------------------
// close STL file
fclose(stream);
//------------------------
// clean up
    printf("\nPress any key...");
getch();
closegraph();
}
```

Monkey saddle image produced



References:

Accardo A., Candido G., Jellus V., Toffanin R., Vittur F., " Ex Vivo Assessment of Trabecular Bone Structure from Three-Dimentioanal Projection Reconstruction MR Micro-Images," IEEE Trans. Eng., vol. 50, pp. 970-976, August 2003.

Aissaoui R., Lacoste M., Dansereau J., " Analysis of sliding distribution during a repositioning of persons in a similar chair," IEEE Trans. Neural Syst. Rehab. Eng., vol 9., pp. 215-224, June 2001.

Alshamali A., Al-Fahoum A., "An Efficient Coding Algorithm for the Compression of ECG Using the Wavelet transform," IEEE Trans. Eng., vol. 50, pp. 1034-1037, August 2003.

Arfken, G. Mathematical Methods for Physicists, 3rd edition, Orlando, FL: Academic Press, 1985.

Armstrong P., Wastie M., Diagnostic Imaging. 3rd edition, Blackwell Science, Oxford, 8-14.

Bai Q., Wise K. D., Anderson D.J., "A high-yield microassembly structure for three-dimensional microelectrode arrays," IEEE Trans. Eng., vol. 47, pp. 281-289, March 2000.

Ball J., Price T., Chesney's Radiographic Imaging. 5th edition, Blackwell Scientific Publications, Oxford, 1989, 25-28, 335-343.

Barr R. C., Plonsey R., "Electrode Systems for Measuring Cardiac Impedances Using Optical Transmembrane Potential Sensors and Instititial Electrodes," IEEE Trans. Eng., vol. 50, pp. 931-933, August 2003.

Beetner D. G., Kapoor S., Manjunath S., Zhou X., Stoecker W. V., "Differentiation Among Basal Cell Carcinoma, Benign Lesions, and Normal Skin

Using Electric Impedance," IEEE Trans. Eng., vol. 50, pp. 1021-1024, August 2003.

Berenstein A., Hartman J., "Three Dimensional Arteriography", Simens Electro Medica Special Issue Cardiology,vol 68, pp27-30 1-31

Bester E.A, Ham J, Loots K., Stark A., Study & Master Mathematics, National Book Printers, Goodwood, 1998, 55-69, 193-209,228-261,327-346.

Cappozzo A., Catani F., Leardini A, Benedetti M. G., Croce U. D., "Positioning and orientation in space of bone sduring movement: Experimental artifacts," Clin. Biomed., vol 11, no. 2, pp 90-100, 1996

Czes Kosniowski, Fun Mathematics on your Microcomputer, Cambridge University Press, New York, 1983 – 1984, 5-20, 129-144.

Dellorme S., Petit Y., Labelle H., Aubin C.E., Dansereau J., "Assessment of the 3-D Reconstruction and High Resolution Geometrical Modelling of the Human Skeletal Trunk From 2-D Radiographic Images," IEEE Trans. Eng., vol. 50, pp. 989-996, August 2003.

Dipole Modeling and localization from spatio-temporal MEG data," IEEE Trans. Eng., vol. 39, pp. 541-557, June 1992.

Ekoule, A., Peyrin, F. and Odet, C. A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours. ACM Transactions onGraphics 10, 2 (April, 1991), pp. 182-191.

Fernando K.L., Mathews V. J., Varner M. W., Clark E. B., "Robust Estimation of Fetal Rate Variability Using Doppler Sound," IEEE Trans. Eng., vol. 50, pp. 950-956, August 2003.

http://home.att.net/~castleisland/

http://mathworld.wolfram.com/Quaternion.html

http://mathworld.wolfram.com/RotationMatrix.html http://mdchoice.com/xray/ctscan/Ct.asp

http://medical.nema.org/

http://medical.nema.org/dicom/2003.html

http://www.3dsystems.com/

http://www.cc.utah.edu/~asn8200/rapid.html

http://www.cis.rit.edu/htbooks/mri/

http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=Published/EmeraldFullTextArticle/Articles/1560060304.html

http://www.enseignement.polytechnique.fr/profs/informatique/Francois.Sillion/Majeure/Projets/kong/projet.html (Interpolation of 3D Shapes)

http://www.materialise.com/medical-rpmodels/casestudies_ENG.html

http://www.materialise.com/mimics/stl_ENG.html

http://www.med.harvard.edu/AANLIB/cases/caseNA/pb9.htm

http://www.med.harvard.edu/AANLIB/home.html

http://www.med.uni-giessen.de/ipl/dicomvl.html

http://www.medicinenet.com/mri_scan/article.htm

http://www.medrecinst.com/

http://www.netmedicine.com/xray/ctscan/ct27.htm

http://www.radiologyinfo.org/content/ct_of_the_body.htm

http://www.radiologyinfo.org/content/mr_of_the_body.htm

http://www.rapidprototyping.net/

http://www.sph.sc.edu/comd/rorden/dicom.html

http://www-pet.umds.ac.uk/~eds/dicom.html

Hwang S. N., Wehrli F. W., Williams J. Lord J.C., Haddad J. G., "Cancellous bone volume structure in the forearm: Noninvasive assessment with MR microimaging and image processing," Radiology vol. 206, pp. 347-357, Feb. 1997.

James R. Kent, Richard E. Parent and Wayne E. Carlson, Establishing Correspondences by Topological Merging: A New Approach to 3-D Shape Transformation. Graphics Interface '91, pp. 271-278.

James R. Kent, Wayne E. Carlson and Richard E. Parent, Shape Transformation for Polyhedral Objects. Computer ames Graphics (July, 1992), pp. 47-54.

Kabel J., Odgaard A., Van Rietergen B., Huiskes R., " Connectivity and the elastic properties of cancellous bone," Bone, vol. 24 pp.115-120, Feb. 1999.

Kauffmann C., Gravel P., Godbout B., Gravel A.,Beaudoin G., Raynauld J., Martell-Pelletier J., Pelletier J., de Guise J.A., "Computer-Aided Method for Quantification of Cartilage Thickness Volume Changes Using MRI: Validation Study Using a Synthetic Model," IEEE Trans. Eng., vol. 50, pp. 979-986, August 2003.

Kim K. H., Kim S. J., "A Wavelet-Based Method for Action Potential Detection From Extracellular Neural Signal Recording With Low Signal-to-Noise Ratio," IEEE Trans. Eng., vol. 50, pp. 999-1004, August 2003.

Kingsley S. B., Smith W. M., "Experimental techniques for investigating cardiac electrical activity and response to electrical stimuli," Proc. IEEE Trans. Eng., vol. 84, pp. 417-427, 1996.

Kumar D., Singh M., "Characterization and Imaging of Compositional Variation in Tissues," IEEE Trans. Eng., vol. 50, pp. 1015-1018, August 2003.

Kwan R.K.S., Evans A.C., Pike G.B., "MRI simulation-based evaluation of image-processing and classification methods," IEEE Trans. Med. Imag., vol 11, pp. 1085-1097. Nov. 1999

Lalonde N., Dansereau J., Rachid Aissaoui R., Pauget P., Cinquin P., "Differences Between Pelvic Skin Bone Landmark Identification in Different seated Positions on Spinal-Cord Injured subjects," IEEE Trans. Eng., vol. 50, pp. 958-962, August 2003.

MacDonald S & Burns D. Physics For The life and Health Sciences, Addison-Wesley
Publishing Company, Amsterdam, 1979, 677-684.

Maris E., "A Resampling Method for Estimating The signal Substance of Spatio-Temporal EEG/MEG Data," IEEE Trans. Eng., vol. 50, pp. 936-941, August 2003.

Mason J.C., Basic Numerical Mathematics, Brittish Library Cataloguing in Publication Data, Butterworth & Co Publications, London, 1983, 54-59, 94-97.
McMullan J.T., Physical Techniques in Medicine. Volume 1, John Wiley & Sons, 1979, 68-71.

McRae R., Practical Fracture Treatment. 3rd edition, Churchill Livingstone, London, 1999, 191-195.

Microsoft Encarta Encyclopedia 2003. 1993-2002 Microsoft Corporation.

Plonsey R., Barr R. C., "The Four-electrode resistivity technique as applied in cariac mussle," IEEE Trans. Eng., vol. BME-29, pp. 541-546, 1982.

Roth B.J., "Electrical stimulation of cardiac tissue: A bidomain model with active membrane properties," IEEE Trans. Eng., vol. 41, pp. 232-240, March 1994.

Shvartsman L. D., Fine I., "Optical Transmission of Blood: Effect of Erythrocyte Aggregation," IEEE Trans. Eng., vol. 50, pp. 1026-1033, August 2003.

Stephen L. Mosher, Methods and Programs Mathematical functions, Ellis Horwood limited Publications, Chichester,360-362.

Webster J, Medical Instrumentation: Application and Design. 3rd edition, 1998,
532-554

Letter from journal acknowledge receipt of journal paper

Reply Reply To All Forward Next Unread Delete & Next **Back to INBOX** Delete Set Flag Clos

From: "Gail" <sorsa.admin@iafrica.com>
Subject: Re: Propose Paper for pablication
Date: Thu, 2 Jun 2005 13:01:26 +0200
To: "Raymond Nell" <raymondnell@webmail.co.za>

Hello Raymond
I acknowledge receipt of your proposed paper for publication. I have forwarded it to the editor of the journal, Mrs Leonie Munro.
Thank you for submitting your paper for consideration.
Regards

Gail McLellan
Admin Secretary
Society of Radiographers of South Africa
Tel + 27 21 419 4857
Fax + 27 21 421 2566
Email sorsa.admin@iafrica.com

----- Original Message -----
From: "Raymond Nell" <raymondnell@webmail.co.za>
To: <sorsa.admin@iafrica.com>
Sent: Thursday, June 02, 2005 12:03 PM
Subject: Propose Paper for pablication

> To : Society of Radiographers
>
> From : Raymond Nell
> Cape Peninsula University of Technology
>
> Dear Sir/Madam
>
> I would like to forward he following paper about
> Theoretical method for detecting the 3D point in space with
> one X-ray exposure for publication. I am a Radiographer at
> the Community Health Centre and currently doing my Master's
> degree in Electrical Engineering.
>
> The proposed paper is part of my research in Design and
> Analysis of system for the fabrication of synthetic
> anatomical structures where we developed physical models of
> 3D anathomical structures.
>
> I have forwarded the paper to Alladin Speelman at Grootte
> Schuur Hospital to read through and he provided me with
> your contact details
>
> Please do take a look at the paper and can you please
> provide me with confirmation of receiving the paper.
>
> Thank you
> Raymond Nell
>

Copy to... Move to... Redirect to...

Take Address

Reply | Reply To All | Forward | Next Unread | Delete & Next | **Back to INBOX** | Delete | Set Flag | Close as Unrea

From: Lorna Martin <lornaj@curie.uct.ac.za>
Subject: Re: Feedback from Radio Program
Date: Fri, 11 Mar 2005 12:24:10 +0200
To: Raymond Nell <raymondnell@webmail.co.za>

Dear Raymond, thanks for the feedback & interest in the Stascan machine.
Glad to hear the program went off well.
Lorna Martin

Raymond Nell wrote:

> Dear Prof. Martin
>
> From Raymond
>
> I just want to thank you for you blessings for the Radio
> Program on Voice of the Cape.
>
> Myself, Greg, Allison Davison (from the environmental
> health) were in the studio and Prof. Martin and Dr. Barday
> join in via telephone.
>
> Both Prof. Knobel and Dr. Barday discussed the topic in
> detail over the air with Greg and myself.
>
> Prof. Knobel explained the benefits of having the LODOX
> system and we fill in where is needed. Certainly the topic
> of doing more research was also discussed and all the
> participants contributed their views.
>
> In general the program went well to explain the benefits of
> the LODOX system and other and hope it reached a huge
> community.
>
> Thank you
> Raymond
>
> http://www.webmail.co.za the South African FREE email service

Prof. Lorna J Martin

ADR

Falmouth Building, Level 1, Entrance 3
Faculty of Health Sciences Anzio Road
Observatory
Cape Town

EMAIL	lornaj@curie	
N	Martin Lorna J	
ORG	University of Cape Town Division of Forensic Medicine & Toxicology	
TEL	082 600 6350	cell
	021 448 1249	fax
	021 406 6412	work
TITLE	Professor	
X-MOZILLA-HTML	FALSE	

Copy to... Move to... Redirect to...

Take Address

Reply Reply To All Forward Next Unread Delete & Next **Back to INBOX** Delete Set Flag Close as Unrea

From: "Greg Flash" <greg.flash@lodox.com>
Subject: RE: Message of thanks for participating in Radio Program
Date: Fri, 11 Mar 2005 08:19:16 +0200
To: "'Raymond Nell'" <raymondnell@webmail.co.za>

Hi Raymond

It was only a pleasure. I'm happy to go on the show any other time
or to do
some educational promotions in schools or wherever as you
suggested.

Thanks

Greg

-----Original Message-----
From: Raymond Nell [mailto:raymondnell@webmail.co.za]

Sent: 10 March 2005 17:52 PM
To: greg.flash@lodox.com
Subject: Message of thanks for participating in Radio Program

Message of thanks for participating in Radio Program
==

Hi Greg

From Raymond

I just want to thank you for your contribution and taking part in the Radio Program on Voice of the Cape.

I've already sent a message of thanks to Dr. Barday, Alison Davison and Prof. Knobel.

As can see this topic is very debatable and the contributions of Prof. Knobel is well appriciated.

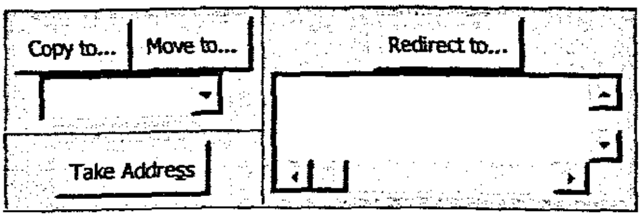
Thanks again
Raymond

---

http://www.webmail.co.za the South African FREE email service

--
No virus found in this incoming message.
Checked by AVG Anti-Virus.
Version: 7.0.308 / Virus Database: 266.7.1 - Release Date: 09/03/2005

--
No virus found in this outgoing message.
Checked by AVG Anti-Virus.
Version: 7.0.308 / Virus Database: 266.7.1 - Release Date: 09/03/2005



University of Cape Town

Application Status for the First Semester

Applicant: **NELL, RAYMOND D MR**

Applicant Number: **NLLRAY001**

You may have applied for more than one programme of study (i.e. a first and second choice). The information given here refers to the choice that is currently under consideration. The information about your accommodation and funding application is only valid for the current application.

Academic Application

Your active application for study is for the:
MEDB14 - BACHELOR OF MEDICINE AND BACHELOR OF SURGERY

You have been placed on a waiting list.

Accommodation Application

You did not request a place in UCT Student Accommodation.

Funding Application

You did not request funding based on financial need from UCT on your UCT Application Form.

Housing application data may not show the latest status - for more up to date information please contact the Student Housing office on 021 650 2102

Please direct any application enquiries to: admissions@bremner.uct.ac.za