



Three dimensional visualization of medical images
by Premkumar Manangam Pandurangam

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Computer Science

Montana State University

© Copyright by Premkumar Manangam Pandurangam (1994)

Abstract:

The problem of construction of three dimensional models from two dimensional data is an old and challenging one. . This problem arises in the field of tomographic medical imaging, where it is necessary to reconstruct an organ or other objects of interest from the available two dimensional data. The two dimensional data is in the form of CAT (Computer Assisted Tomography) or MRI (Magnetic Resonance Imaging) slice images. This thesis presents a solution to this problem. The application that is used for study is the Boron Neutron Capture Therapy Project, aimed at achieving successful non-invasive treatment of brain tumors. But, it can be applied to any other similar application with minimal or no changes.

The problem can be divided into two parts. The first part is to successfully extract the necessary information and weed out the unwanted information from slice data. The second part is to build the model(s) from the extracted information. In this thesis, the first part of extracting information is done by interactive image processing tools, built using a friendly user interface. The construction of complex models is handled by a rendering tool that constructs - models and presents the models for interactive transformations.

The two criteria of performance are speed and accuracy. There is an inherent tradeoff between the two. To achieve one the other has to be sacrificed to some extent. This thesis presents a situation where the preference of one over the other is left to the user to decide, rather than presupposing anything. This gives the user the flexibility to switch from speed requirements to accuracy depending on the situation. In conclusion this thesis presents a solution to the three dimensional visualization problem with flexible and user friendly tools.

THREE DIMENSIONAL VISUALIZATION
OF MEDICAL IMAGES

by

Premkumar Manangam Pandurangam

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

May 1994

N378

P1926

ii

APPROVAL

of a thesis submitted by

Premkumar Manangam Pandurangam

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

5/2/94

Date

Ray S. Babcock

Chairperson, Graduate Committee

Approved for the Major Department

5/2/94

Date

J. D. Biggs Stanley

Head, Major Department

Approved for the College of Graduate Studies

10/27/94

Date

R. Brown

Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature M.P. Runk

Date 5/2/94

TABLE OF CONTENTS

	Page
INTRODUCTION	1
Background	1
Applications	2
BNCT PROJECT	3
The C Programming Language	4
The Unix Operating System	4
The X Window System	4
Motif User Interface	5
PEX	5
IMAGE PROCESSING AND ANALYSIS	7
Preprocessing	7
Intensity	7
Bump	7
Contrast	10
Contrast Stretch	10
Histogram Representation of Images	12
Histogram Equalization	13
Noise	15
Average Filter	15
Median Filter	16
Blur	18
Sharpening Filter	18
IMAGE SEGMENTATION	19
Edge Detection and Derivatives	19
The Sobel Operators	21
Laplacian Operator	22
INFORMATION EXTRACTION	24
Object Selection	24
MODEL CONSTRUCTION	27
Interpretation of Data	27
Ordering of points	29
Drawing	30

B-Spline Curves	31
Knot Vectors	32
Curve Order	32
Spline Surfaces	34
VIEWING	35
Viewing Reference Coordinate System	35
TRANSFORMATIONS	38
Translation	38
Scaling	38
Rotation	39
CONCLUSION	46
Significance of the work	47
Scope for improvement	47
REFERENCES CITED	49

LIST OF FIGURES

Figures	Page
3.1: Illustration of bump	9
3.2: Form of transformation function for contrast stretching	10
3.3: Illustration of Contrast Stretch	11
3.4: Histogram representations of different images	12
3.5: Illustration of the equalization process with probability density function transformation	13
3.6: Illustration of equalization process using images	14
3.7: A 3 x 3 average filter	15
3.8: Average and Median Filtering	17
3.9: A 3 x 3 sharpening Filter	18
4.1: Edge Detection and Derivatives	20
4.2: Sobel Operators	21
4.3: A 3 x 3 Laplacian mask	22
4.4: Image Segmentation	23
5.1: Selection of points in a slice image	26
6.1: Points traversal	28
6.2: Ordering of six points using the traversal algorithm	30
6.3: B-spline curves of two different orders using the same control points	33
7.1: Viewing Reference Coordinate System	36
7.2: Top View, Side view and Front View of the model	37
8.1: Translation	40
8.2: Scaling	42
8.3: Rotation	44

ABSTRACT

The problem of construction of three dimensional models from two dimensional data is an old and challenging one. This problem arises in the field of tomographic medical imaging, where it is necessary to reconstruct an organ or other objects of interest from the available two dimensional data. The two dimensional data is in the form of CAT (Computer Assisted Tomography) or MRI (Magnetic Resonance Imaging) slice images. This thesis presents a solution to this problem. The application that is used for study is the Boron Neutron Capture Therapy Project, aimed at achieving successful non-invasive treatment of brain tumors. But, it can be applied to any other similar application with minimal or no changes.

The problem can be divided into two parts. The first part is to successfully extract the necessary information and weed out the unwanted information from slice data. The second part is to build the model(s) from the extracted information. In this thesis, the first part of extracting information is done by interactive image processing tools, built using a friendly user interface. The construction of complex models is handled by a rendering tool that constructs models and presents the models for interactive transformations.

The two criteria of performance are speed and accuracy. There is an inherent tradeoff between the two. To achieve one the other has to be sacrificed to some extent. This thesis presents a situation where the preference of one over the other is left to the user to decide, rather than presupposing anything. This gives the user the flexibility to switch from speed requirements to accuracy depending on the situation. In conclusion this thesis presents a solution to the three dimensional visualization problem with flexible and user friendly tools.

INTRODUCTION

Background

To visualize non-invasively human organs in their true form and shape has intrigued mankind for centuries. The discovery of X-rays gave birth to radiology and provided a way to study internal organs in the human body. But there were limitations in terms of the type of organs that can be X-rayed and the location of the organs. Recently the invention of Computerized Tomography and Magnetic Resonance Imaging has revolutionized radiology. The limitation of X-rays have been successfully overcome by these new technologies. But still the problem of visualizing a three dimensional object in a two dimensional space is very much present.

Three dimensional imaging presents a solution to this problem. Visualization is done in a pseudo three dimensional space, the computer screen. The ability to have depth, color, shadows and motion on the computer screen makes this a good medium for visualization. With the breakthrough in computer hardware technology, it is now possible to process faster, store more information and display static and moving objects in color. In addition, mathematicians and computer scientists have developed with efficient algorithms to build complex objects. The combination of these two factors has taken the medical community by storm and has raised the hope of successful non-invasive visualization of human organs.

Applications

The field of 3-D imaging is new and fast changing and the results are stupendous. Already this technology is in widespread use in hospitals and by individual physicians. Applications range from treatment of fractures to brain surgery.

BNCT PROJECT

The Boron Neutron Capture Therapy Project is a collective effort undertaken by the federal government of the USA, through the Idaho National Engineering Laboratory (INEL) and a consortium of ten universities. The goal of this project is to successfully develop non-invasive treatment of brain tumors.

Stated in simple terms, the treatment is composed of the following steps:

- 1) Obtain images of the human brain.
- 2) Identify the tumor, its location, size and contour.
- 3) Study the feasibility and probability of success of radiation treatment.
- 4) If found feasible, inject Boron compound into the blood that will be absorbed by tumor cells.
- 5) Accurately shoot neutron beams at the tumor cells identified by the Boron compound and kill the tumor, without causing any damage to healthy brain cells.

In this process the first three steps fall into the realm of Computer Science, especially Computer Graphics and Digital Image Processing. This thesis presents a tool that will help do some of the things that are required for these steps. In the succeeding chapters we will look at the problems and how they are attacked by the tool. But first let us see what software we use to accomplish the tasks outlined and why.

The C Programming Language

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C is not tied to any particular hardware or system, and it is easy to write programs that will run without change on any machine that supports C. More importantly many software libraries and applications are written in C and that factor along with the portability feature of C are the main reasons for selecting the C programming language for this application.

The Unix Operating System

The Unix operating system is written in C and is portable just like the C language. Unix systems run on a range of computers from microprocessors to mainframe computers. The Unix programming environment is rich and productive. It supports among other things time-sharing and network programming. These qualities make Unix a good operating system and that is the reason it was chosen for this application.

The X Window System

The X Window System is a network transparent windowing system. In X, multiple applications can run simultaneously in windows, generating text and graphics in monochrome (black and white) or color. Network transparency means that application programs can run on machines scattered throughout the network. So applications need not be rewritten, recompiled or even relinked to work with new display hardware. It is

because of this quality that the X Window System is quite popular in computer graphics applications and the reason why it is favored in this application.

X library or Xlib (as it is referred to most of the time) is the C language programming interface of the X Window System. Xlib is the lowest level of programming interface to X. This library enables a programmer to write applications with an advanced user interface based on windows on the screen, with complete network transparency. The X toolkit or Xt is a toolkit written with Xlib functions providing a higher level programming environment. This application uses both these libraries and more to get the job done.

Motif User Interface

In this era of user friendly interfaces and event driven programming, it is imperative that we keep abreast of technology. Motif has a set of "widgets" which allow the programmer to do just that and easily so. It is built on top of the Xlib and Xt libraries. It provides to the application programmer an easy way of building user interfaces with a common "look and feel". That is the reason why Motif is used in this application. The figures show why Motif is a good system to use.

PEX

It is often cumbersome to write programs using the above mentioned libraries that involve extensive 3-D drawing, rendering and animation. It is possible, and has been done before, but the general opinion is that the resulting code is too long, complex and difficult to understand. That is because there are too many low level details to be considered without a good 3-D graphics library. PEXlib attempts to solve this problem.

PEXlib is a high-level programming library built on top of Xlib and Xt. PEX initially stood for "PHIGS Extension to X", where PHIGS is a three dimensional graphics standard. But PEXlib goes beyond PHIGS in functionality. With PEXlib, creating complex objects, viewing them from different angles, translating, scaling and rotating them, lighting and shading them are all made easier with high level functions and primitives. It encourages modularity and supports object oriented programming techniques.

PEXlib is new and developing and has room for improvement. It is already popular in the industry and the education community. The author feels that PEXlib will dominate 3-D graphics programming comparable to the dominance of the X Window System itself. That is the motivation behind writing this 3-D visualization application using PEXlib.

In the ensuing discussion of the application, the low level details of programming are left out and only the fundamental principles of operation of the application are discussed. But the reader is more than welcome to look at the source code of this application which should be available from Mr. Ray Babcock, Adjunct Assistant Professor, Department of Computer Science, Montana State University, Bozeman, Montana. He can be contacted via e-mail at babcock@cs.montana.edu.

IMAGE PROCESSING AND ANALYSIS

Preprocessing

At this point it is assumed that we have images available generated using Computerized Tomography or Magnetic Image Resonance techniques and stored in digital form in the computer. The problem is to extract information from the images to be used in drawing objects. This problem is compounded by irritants such as noise, blur, low/high intensities (dark/bright), poor contrast etc. It is our job to overcome these minor irritants before we go about the process of extracting information from these images. Let us look at them one at a time and see how they can be overcome.

Intensity

It is possible that an image generated by one of the techniques we mentioned is very dull, due to technical problems during image generation. But there is an easy way to correct this problem thanks to image processing techniques. Conceptually, if we look at the image as a collection of intensities ranging from absolute darkness to absolute brightness, it is easy to see that to make a dull image brighter all we need to do is to add some intensity to the whole image.

Bump

That is exactly what is done in the tool called bump in this application. The image that is used is nothing but a collection of pixels (picture elements) each one having

an intensity. So, to brighten an image an intensity is added (defined by the user interactively) and to darken a image an intensity is subtracted. This operation is done on all the pixels so that it is uniform in the image.

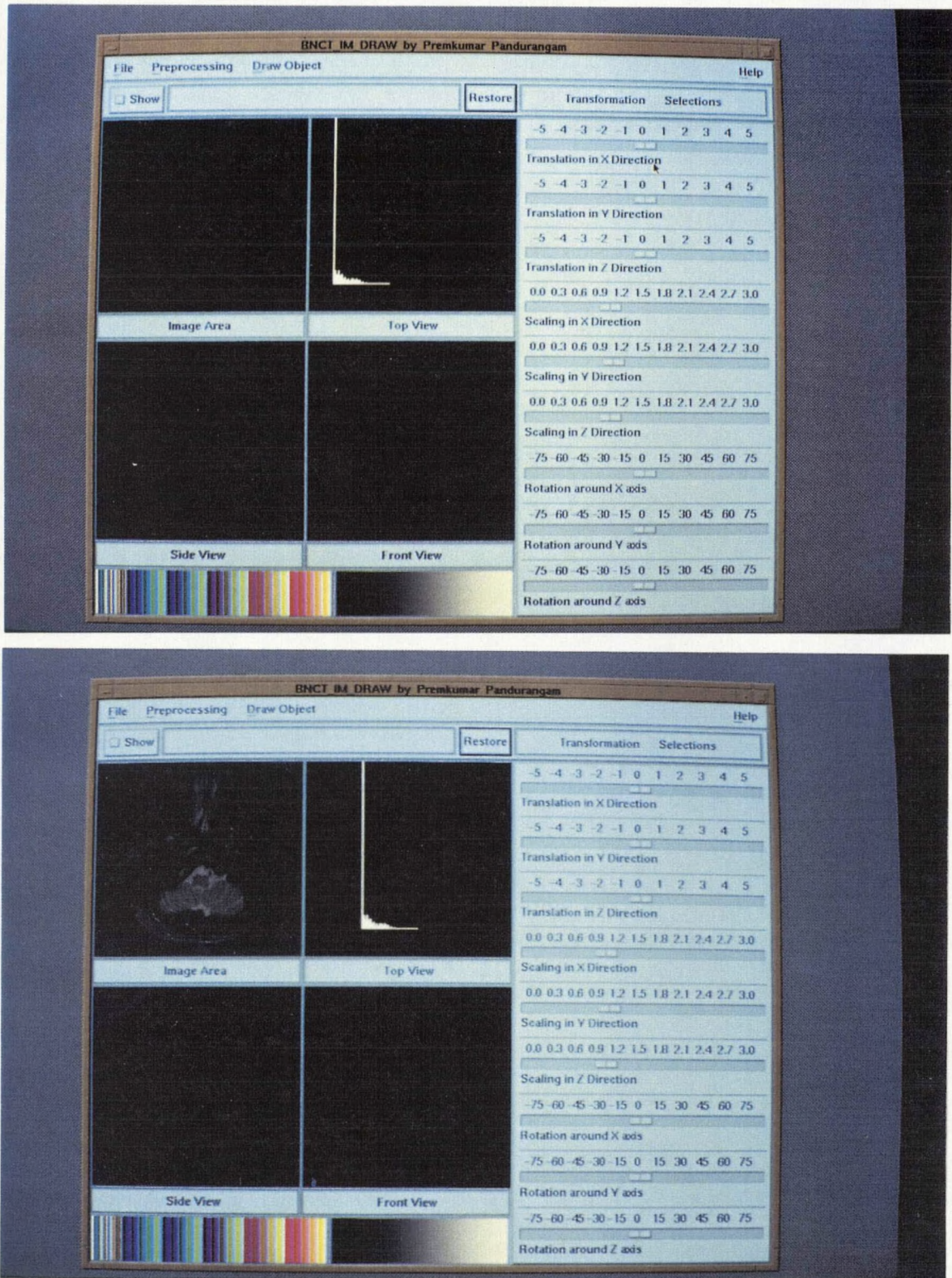


Figure 3.1: Illustration of bump (a) an image with low intensities (b) a "bumped" image. Note that the image as a whole is brightened rather than a portion of the image.

Contrast

Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor or even wrong setting of a lens aperture during image acquisition.

Contrast Stretch

The idea behind contrast stretching is to increase the dynamic range of the intensities in the image being processed. Say, we have an image with intensities ranging from 0 to 255, but most of the intensity variations are in the range from 100 to 200. It makes sense to exaggerate these changes in this particular range so that we get a better representation of the image. Figure 3.2 shows one such transformation function used for contrast stretching. Intermediate values of (r_1, s_1) and (r_2, s_2) produce various degrees of spread in the intensity levels of the output image, thus affecting its contrast. In this application the user is prompted to select the intermediate values and the stretch factor to effect the contrast stretch.

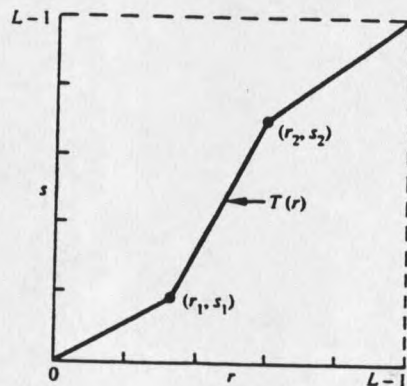


Figure 3.2: Form of transformation function for contrast stretching. (Courtesy of Digital Image Processing, by Gonzalez and Woods, Addison-Wesley 1992)

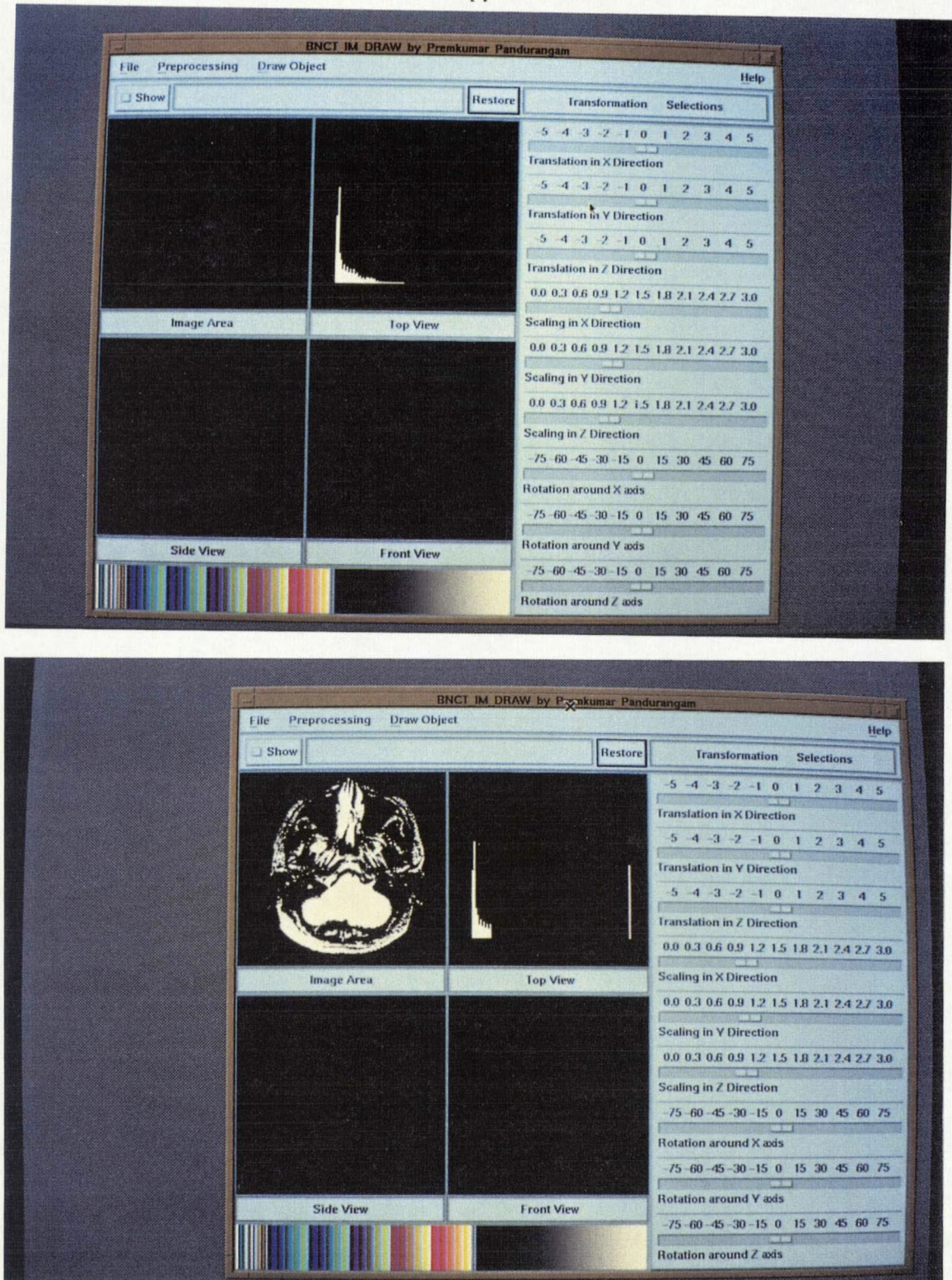


Figure 3.3: Illustration of Contrast Stretch (a) a low contrast image (b) result of contrast stretching

Histogram Representation of Images

One way of describing an image is histogram representation. The histogram of a digital image with intensity levels in the range $[0 \text{ to } L-1]$ is a discrete function $p(r_k) = n_k/n$, where r_k is the k th intensity level, n_k is the number of pixels in the image with that intensity level, n is the total number of pixels in the image, and $k = 0 \text{ to } L-1$. In other words, $p(r_k)$ gives an estimate of the probability of occurrence of intensity level r_k . To better understand the concept here are histogram representations of some images.

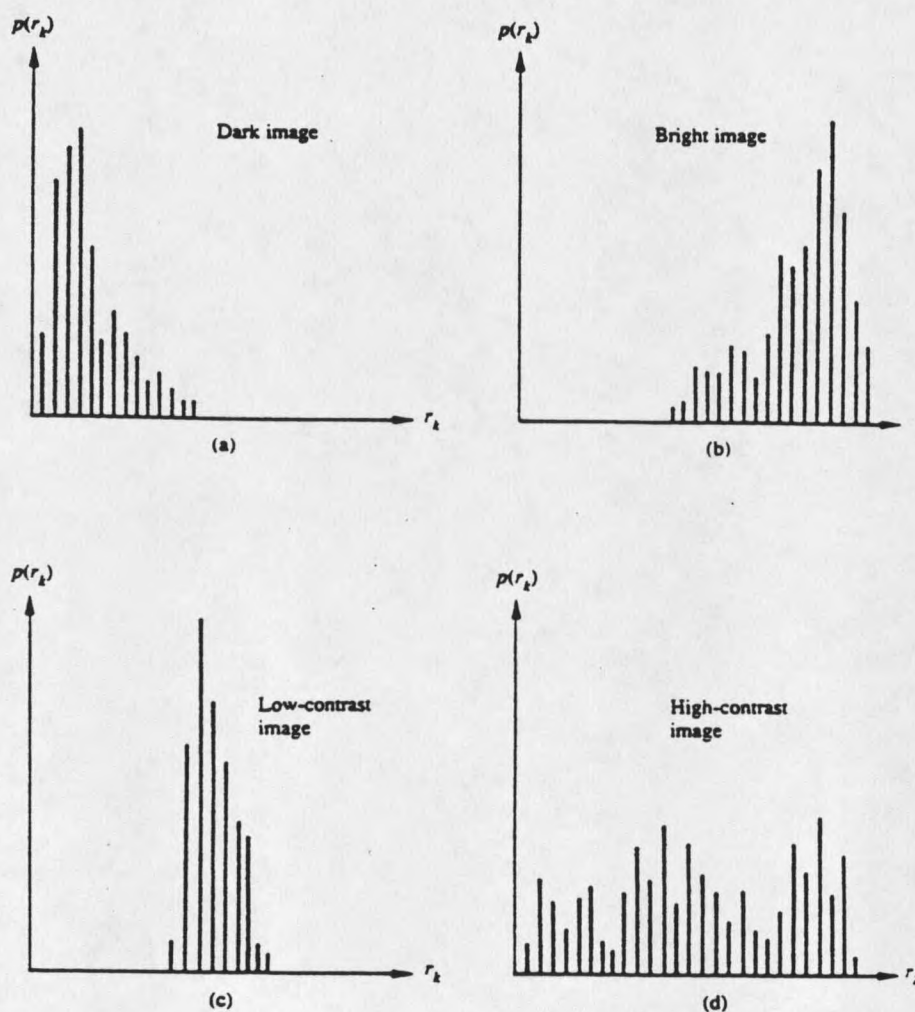


Figure 3.4: Histogram representations of different images (Courtesy of Digital Image Processing, by Gonzalez and Woods, Addison-Wesley, 1992)

Histogram Equalization

A low contrast image has a histogram which is narrow with little dynamic range. On the other hand a high contrast image has a histogram which has significant spread. Intuitively, to get a high contrast image from a low contrast image we need to "stretch" the histogram of the low contrast image. This is accomplished by a process called "Histogram Equalization". The mathematical content of the histogram equalization process is left out for brevity and an illustration of the process is given below to understand the process. The application provides tools to generate, display and equalize histograms of the images.

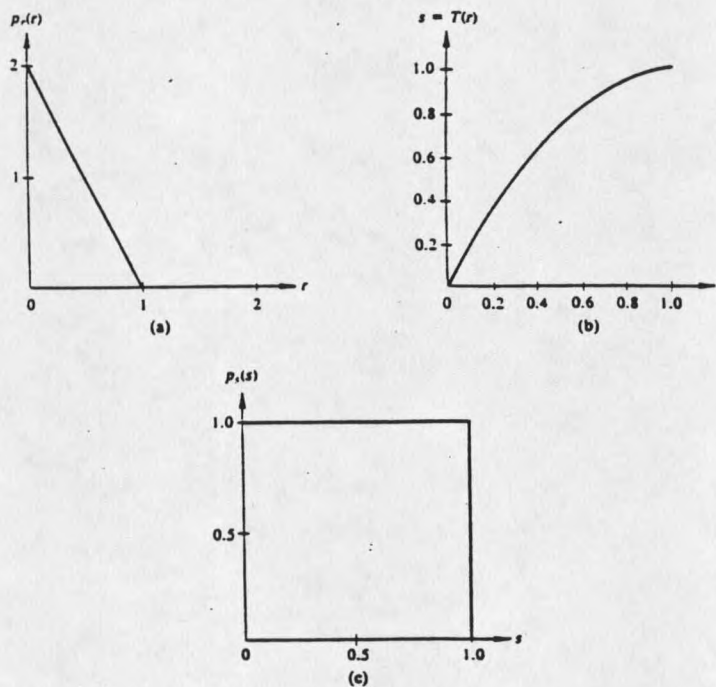


Figure 3.5: Illustration of the equalization process with probability density function transformation (a) original probability density function (b) transformation function (c) resulting uniform density. (Courtesy of Digital Image Processing, by Gonzalez and Woods, Addison-Wesley, 1992)

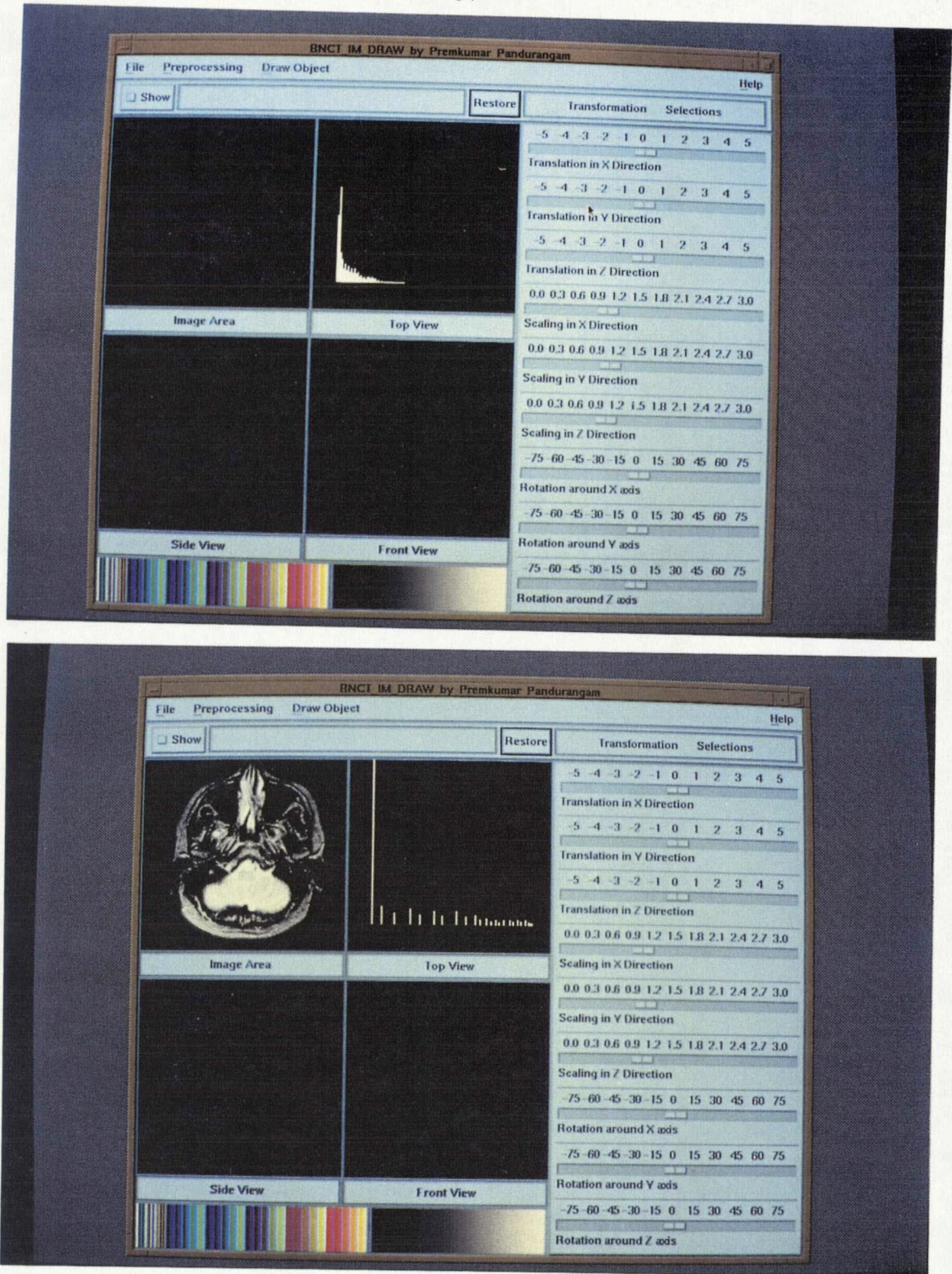


Figure 3.6: Illustration of equalization process using images (a) Original image (b) its histogram (c) image subjected to histogram equalization and (d) its histogram.

Noise

Noise is an irritant present in images which aggravates the problem of extracting useful information from images. It is very common to encounter "spots" and/or "streaks" in images. But there are some simple ways to minimize the noise in images. Spatial filtering by using spatial masks is an easy and effective method for minimizing noise in digital images.

Average Filter

The basic approach is to sum products between the mask coefficients and the intensities of the pixels under the mask at a specific location in the image. Denoting the intensity levels of pixels under the mask at any location by $z_1, z_2, z_3, \dots, z_9$, and the mask coefficients by $w_1, w_2, w_3, \dots, w_9$, the response of a linear mask is

$$R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

A typical spatial filter would be a mask in which all coefficients have a value of 1. However, from the above equation, the response would then be a sum of intensity levels for nine pixels, which would cause R to be out of the valid intensity range. The solution is to scale the sum by dividing R by 9. Basically what is done in this process is averaging of neighboring pixels' intensity values for a given pixel. For this reason this method is called neighborhood averaging and the filter used is called average filter.

1/9 x	1	1	1
	1	1	1
	1	1	1

Figure 3.7: A 3 x 3 average filter

Median Filter

Neighborhood averaging or average filtering both blurs edges and sharp details. It can be used where there is a need for smoothing. If we need to achieve noise reduction and avoid blurring, an alternative approach is to use a median filter. In this method, the intensity of each pixel is replaced by the median of the intensity levels in the neighborhood of that pixel. In order to perform median filtering in a neighborhood of a pixel, we first sort the values of the pixel and its neighbors, determine the median (middle value), and assign this value to the pixel.



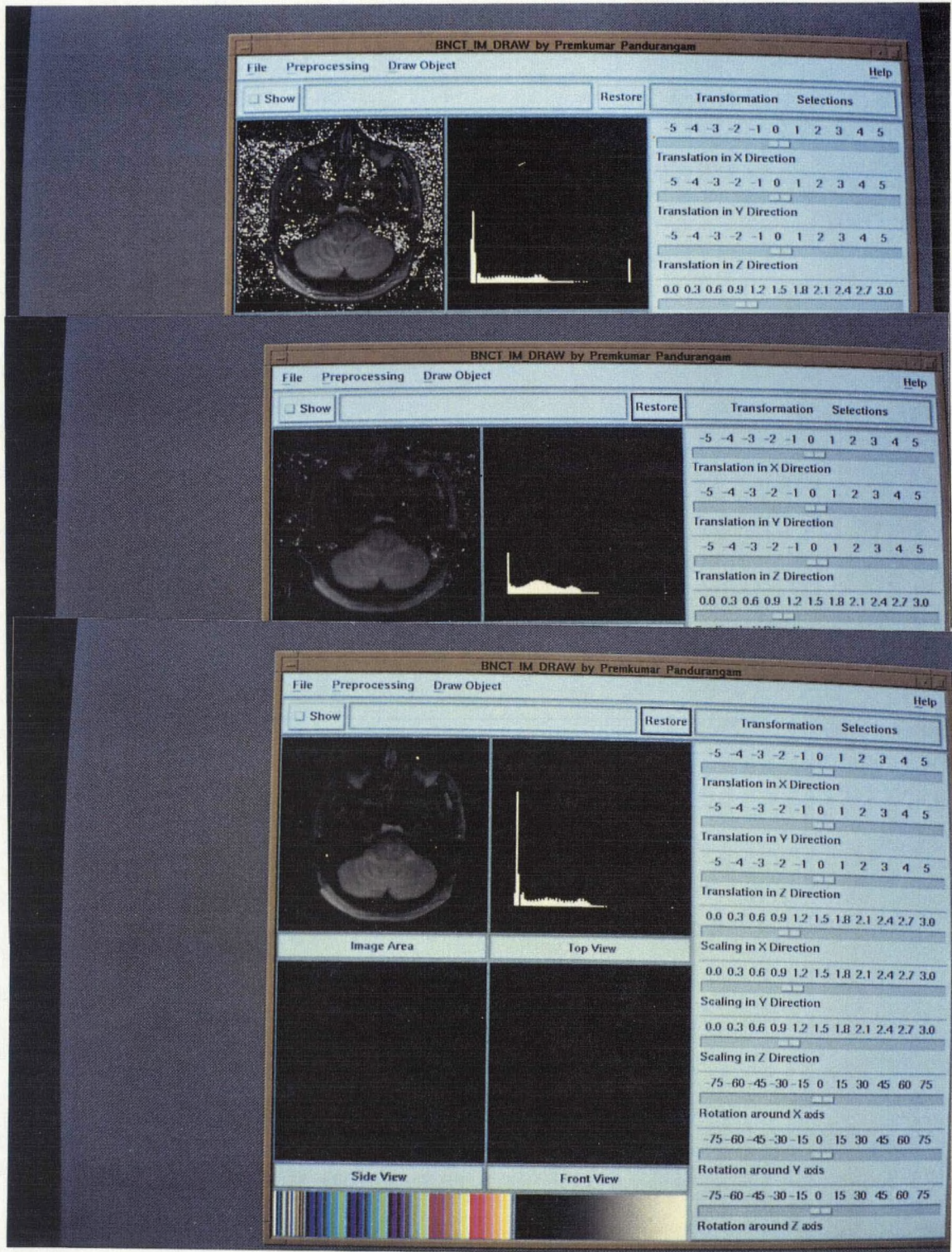


Figure 3.8: Average and Median Filtering (a) Original image with noise (b) result of 3 x 3 average filtering (c) result of 3 x 3 median filtering

Blur

There might be occasions where we might have a blurred image due to a natural effect of a particular method of image acquisition. Sharpening the image by highlighting fine details in an image is accomplished in the same way as average filtering was done, but different filter values.

Sharpening Filter

The sharpening filter (see Figure 3.9) is similar to the average filter, except that the coefficients at the borders are negative and the sum of the coefficients is zero. This mask was used for the sharpening filter tool for this application.

1/9 x	-1	-1	-1
	-1	8	-1
	-1	-1	-1

Figure 3.9: A 3 x 3 sharpening filter

IMAGE SEGMENTATION

Once the images are improved using the techniques described above, then we must think of segmenting the images. Segmentation subdivides an image into its constituent parts or objects. The level to which this subdivision is carried depends on the problem being solved. That is, segmentation would stop when the objects of interest in an application have been isolated. For example, in the BNCT medical imaging application, interest lies in identifying tumors in the brain. The first step is to segment the skull from the image and then to segment the brain and tumor from other bodies.

Edge Detection and Derivatives

Although point and line detection certainly are elements of any discussion on segmentation, edge detection is by far the most common approach for detecting discontinuities in intensity levels. The reason is that isolated points and thin lines are not frequent occurrences in most practical applications.

An edge is the boundary between two regions with relatively distinct intensity level properties. We assume that the regions in question are sufficiently homogeneous so that the transformation between two regions can be determined on the basis of intensity level discontinuities alone. The idea underlying most edge detection techniques is the computation of a local derivative operator. Figure 4.1 illustrates this concept.

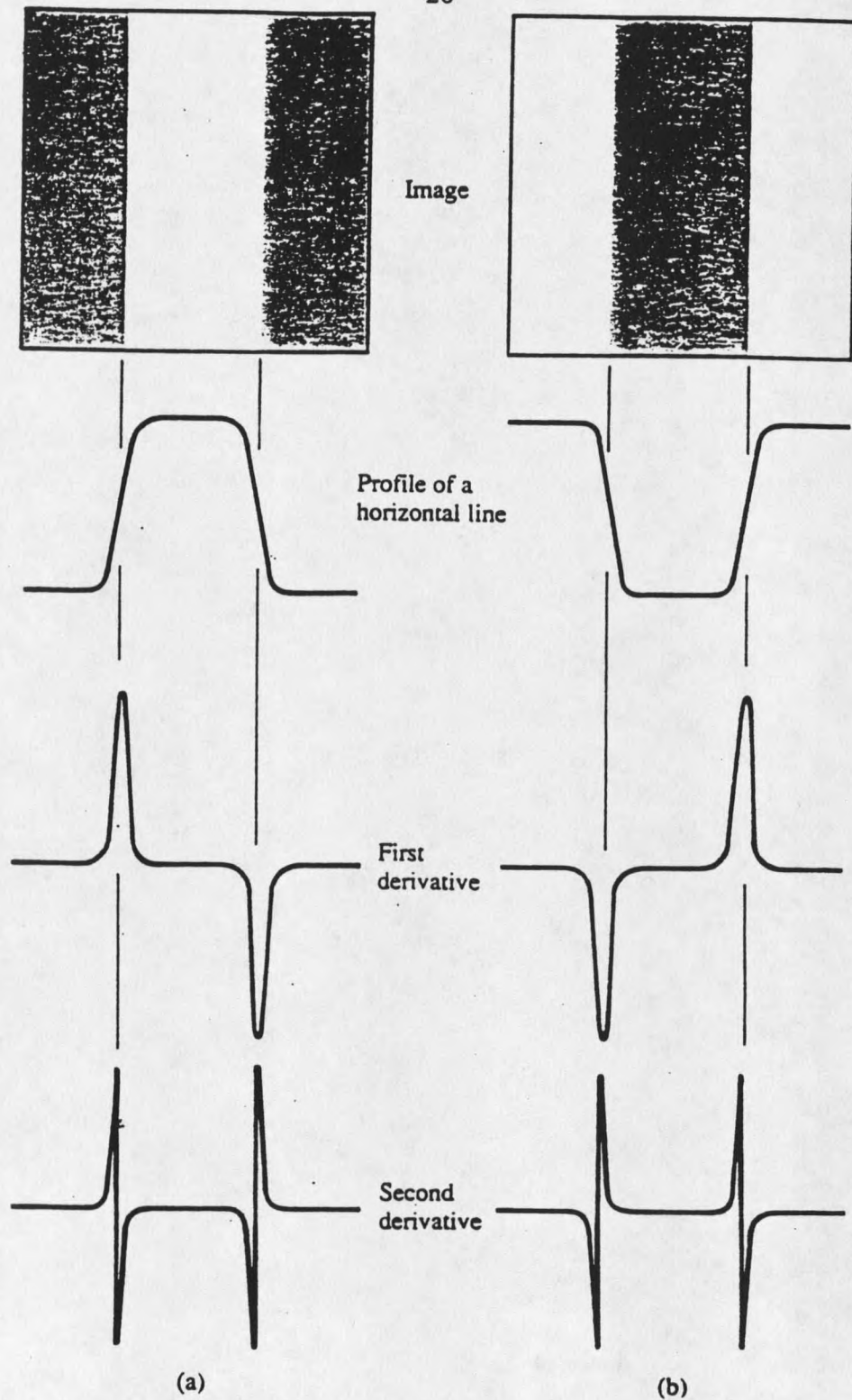


Figure 4.1: Edge Detection and Derivatives (a) light stripe on a dark background (b) dark stripe on a light background (Courtesy of Digital Image Processing, by Gonzalez and Woods, Addison-Wesley, 1992)

The Sobel Operators

The first derivative at any point in an image is obtained by using the magnitude of the gradient at that point. The magnitude of the gradient vector is given by

$$\text{mag}(df) = [G_x^2 + G_y^2]^{1/2}$$

where, gradient of an image = $f(x, y)$ at location (x, y)

and $G_x = df/dx$, $G_y = df/dy$

This is approximated with absolute values:

$$\text{mag}(df) = |G_x| + |G_y|$$

The derivatives may be implemented in digital form in several ways. However, the Sobel operators have the advantage of providing both a differencing and a smoothing effect. Because derivatives enhance noise, the smoothing effect is a particularly attractive feature of the Sobel operators.

-1	-2	-1
0	0	0
1	2	1

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

Figure 4.2: Sobel Operators (a) The mask used to compute G_x at center point of the 3 x 3 region (b) The mask used to compute G_y at that point

From Figure 4.2 the derivatives based on the Sobel operator masks are

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

where, as before, the z 's are the intensity levels of the pixels overlapped by the masks at any location in an image.

Laplacian Operator

The Laplacian of a function $f(x, y)$ is a second order derivative defined as

$$\nabla^2 f = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}$$

As in the case of the gradient, this also may be implemented in digital form. For a 3 x 3 region, the Laplacian operator can be defined as

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

0	-1	0
-1	4	-1
0	-1	0

Figure 4.3: A 3 x 3 Laplacian mask

The Sobel and Laplacian operators were implemented using the masks shown in the above figures.

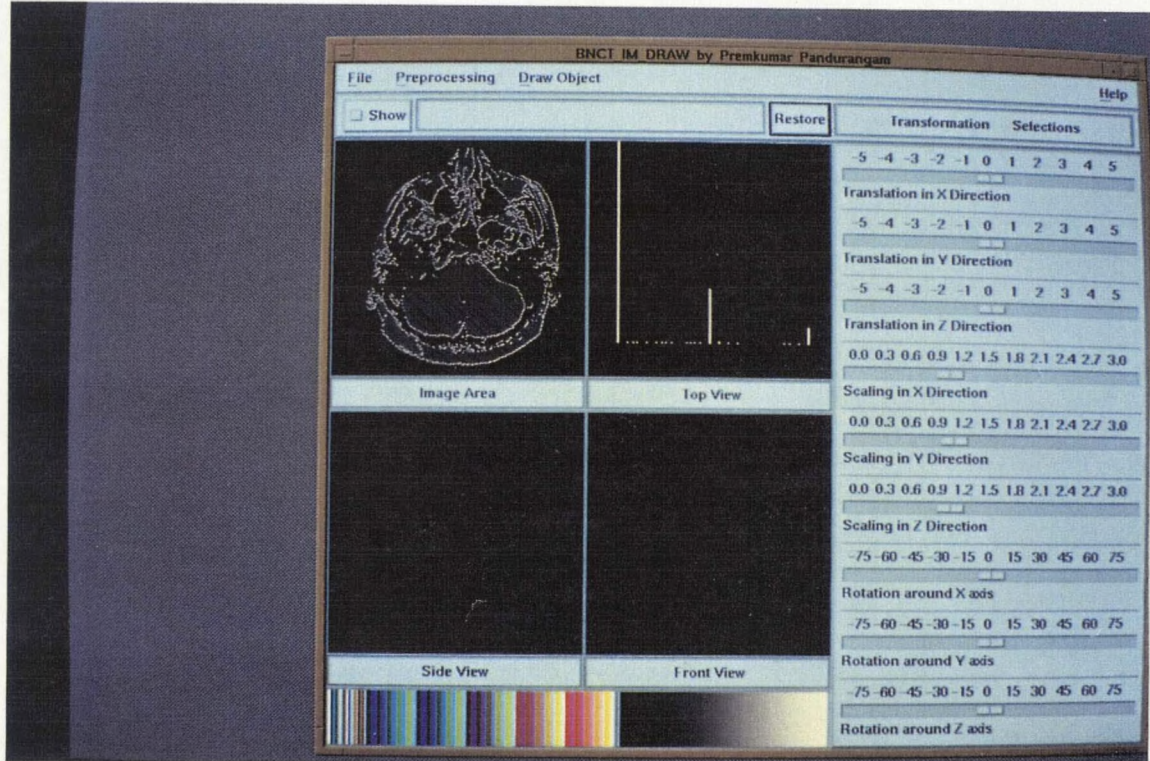
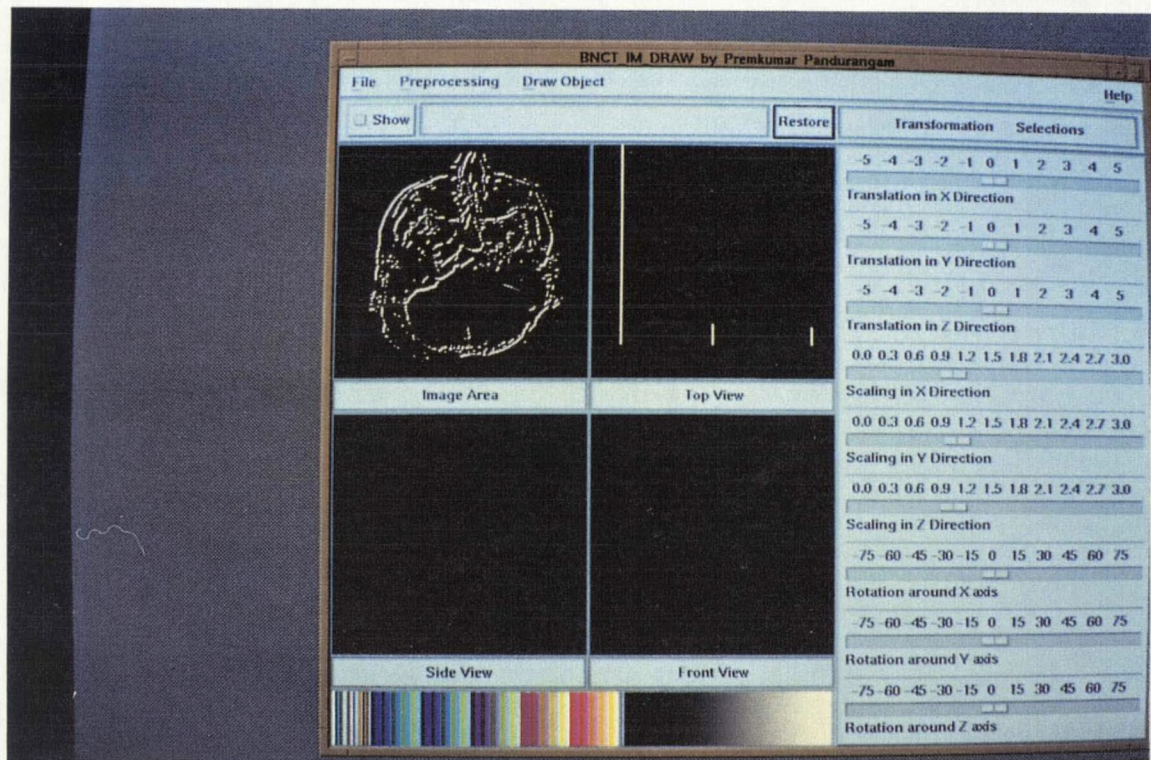


Figure 4.4: Image Segmentation (a) using Sobel operators (b) using Laplacian operator

INFORMATION EXTRACTION

Once the images are improved and segmented, then the next task is to extract information from these images. This is done in a simple and straight-forward way. The idea is to load one image at a time from the available image set and select one object at a time. Let us suppose we have a set of slice images of a human head. These images will have several objects including the eye balls, ears, brain, tumor etc. We will extract information about each object one by one.

Object Selection

The selection of describing points of an object is done using the mouse (a pointer device widely used in workstations and personal computers). After doing the necessary preprocessing operations on the image, the cursor is moved over the object of interest and clicked repeatedly along the boundary of the object. The clicks made are recorded and the coordinates of the points clicked are stored in a file. All the while the user sees the coordinate values and intensity values of the clicked point in the screen.

The user has to predefine the number of points he/she wishes to use to represent the object in one image slice, the number of slice images he/she wishes to use and the object he/she is describing. All this information is stored in the data file even before the coordinate values are recorded.

As you might have noticed already, we have only a two dimensional screen and can get data only in two dimensions (x and y) from the image. The third dimension z is taken from the image slice number information. Suppose, we have a set of slice images taken from front of the head to back of the head and the slice image number starts from 1 to whatever the number of slices used. This slice number is read and used for the third dimension. The image set that was used was a set of axial images (going from bottom to top of the head). So, the screen coordinate information was used for x and z dimensions and the slice image number for the y dimension. Also, it would be easy for drawing purposes if we had the same number of points defining an object in a slice image. Also it is required in this BNCT project application that the objects be closed and represented by same number of points in each slice. So, it is designed such that the user is required to select the same number of points in each slice image.

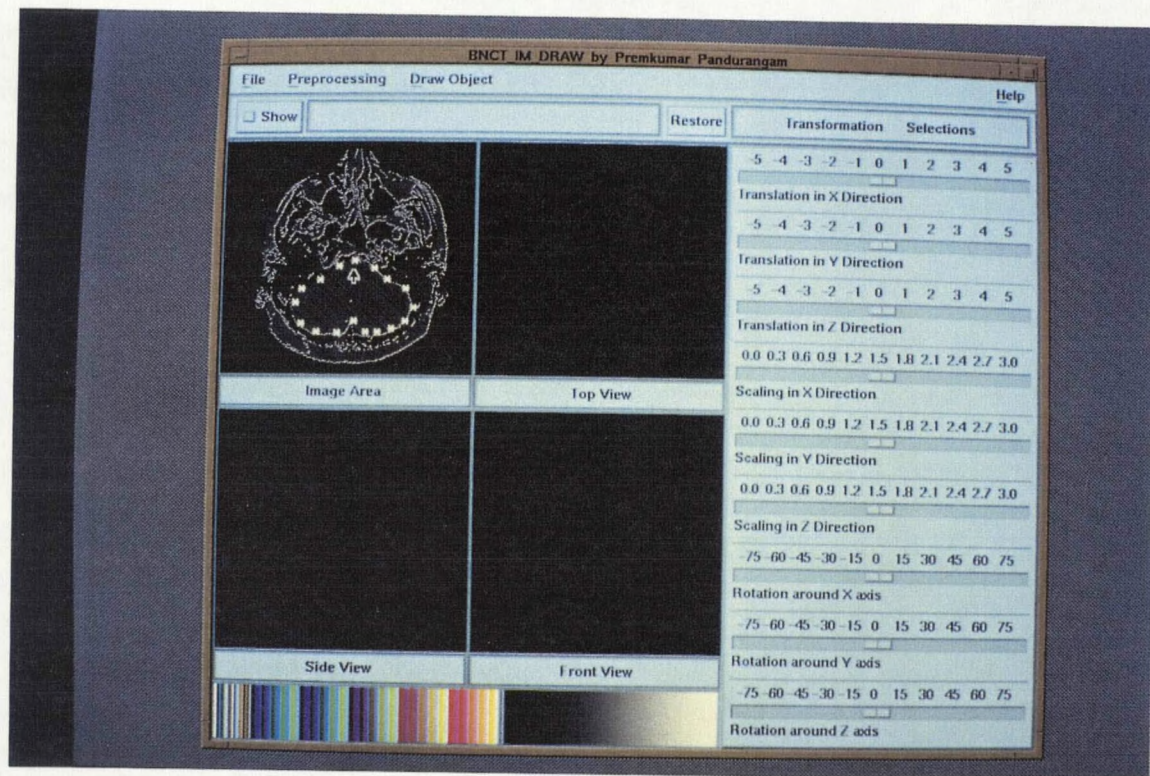


Figure 5.1: Selection of points from a slice image

MODEL CONSTRUCTION

Assuming that we have the coordinates of the objects of interest in a file let us try to build models of these objects using the information we have. Once again the philosophy is to divide and conquer the problem of multiple objects. The information about each object is stored in a separate data file. These files will be read in an orderly fashion and acted upon one by one. Since we had stored the number of slice images used for each object in the data file, we can read this information and act accordingly. Also, since the number of points selected also is stored we know in advance how many points totally we need to process for building an object.

Interpretation of Data

After reading the initial contents of the data file of interest, we are left with the data points of the object of interest. These points are read one by one and stored in an array. Then, just to make sure that these points are in proper slice order, they are sorted based on their z values. Now the points are arranged slice by slice. In each slice i.e. a common z value, the points have to be arranged so that a closed curve can be drawn using the points. For example, if you had a number of points scattered on a X-Y plane. There are arbitrarily many ways you can join them all. But only a clockwise (or counter-clockwise) traversal will produce a closed object. Figure 6.1 illustrates this.

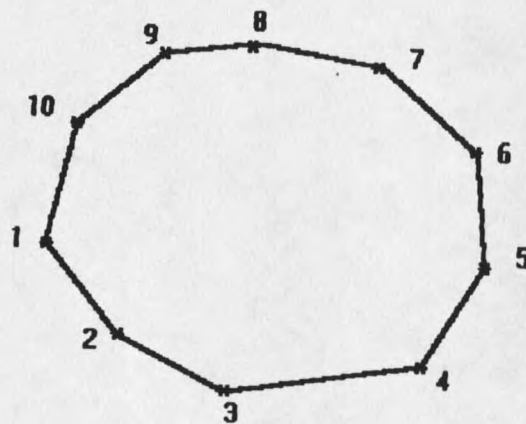
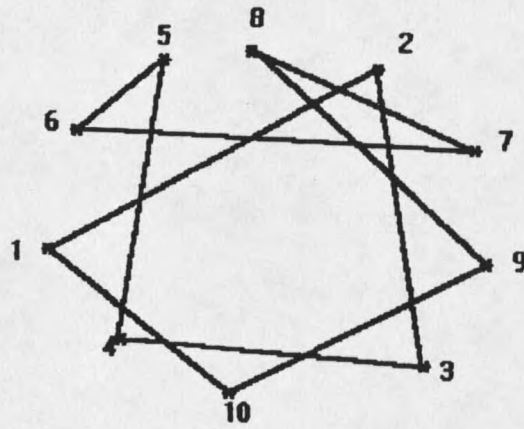


Figure 6.1: Points traversal (a) 10 points in a X-Y plane (b) arbitrary traversal (c) counter-clockwise traversal

Ordering of points

If there are several points in an arbitrary order, how do we order them so that a counter-clockwise traversal is possible? The algorithm is simple and here is how it works:

- 1) For a given slice, find the centroid $[(x_{\min} + x_{\max})/2, (y_{\min} + y_{\max})/2]$
- 2) Treat the centroid as the origin and $x = (x_{\min} + x_{\max})/2$ as the X axis and $y = (y_{\min} + y_{\max})/2$ as the Y axis.
- 3) Then, calculate the difference in X direction and the difference in Y direction between each point and the centroid.
- 4) Sort the points based on these values using the following conditions:

when comparing two points,

if one lies in the positive Y quadrant (i.e. y difference is positive) and the other lies in negative Y quadrant then put the point in the positive Y quadrant first.

if both lie in the positive Y quadrant then put the point with the higher x difference first

if both lie in the negative Y quadrant then put the point with the lower x difference first

Thus the extracted points were sorted in counter-clockwise direction in all the slices and then presented for drawing of curves (or surfaces) in this application.

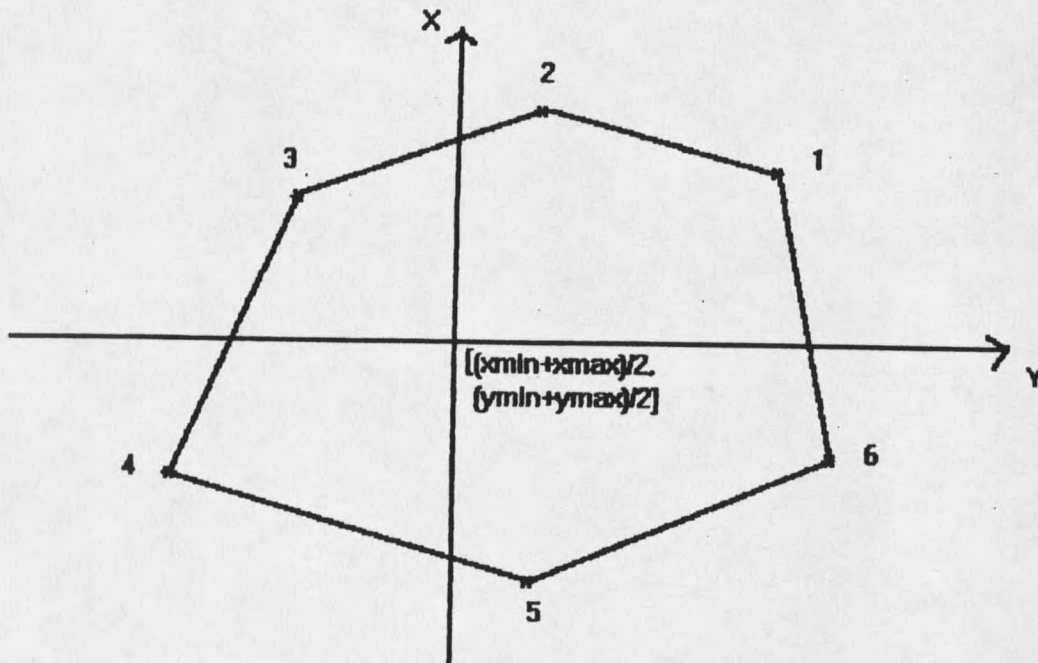


Figure 6.2: Ordering of six points using the traversal algorithm

The working of the algorithm should be obvious from the illustrated example.

Drawing

This algorithm is applied to all the slice points. The next step is to draw lines or curve(s) traversing the points in the order they are sorted. There are a couple of choices here. One way would be to join the points slice by slice by curves and join corresponding points in all the slices. Another way would be to take the entire data set and join them by a surface. This application allows the user to choose which method to

use. In either case B-splines are used for drawing. B-splines are chosen because they are smoother and easy to work with on the computer.

B-Spline Curves

B-spline curves can represent a wide variety of shapes like free-form curves, curves with sharp bends, conic sections etc. A B-spline curve is defined as a list of control points, much like the vertices of a polygon. Each control point influences the shape of the curve over a limited portion of the curve's length. Transforming the curve is a matter of transforming the control points. Control points can be added to give more control over the shape of the curve. We use the points generated for objects as the control points for the B-spline curves and surfaces.

The points along a B-spline curve are computed by a function:

$$P = C(t)$$

where t is the independent variable or the parameter as it is referred to usually. The function $C(t)$ is a combination of polynomials in the variable t . A B-spline curve can be completely specified by the three factors defining $C(t)$:

- 1) The curve's control points
- 2) The curve's order
- 3) The curve's knot vectors

The B-spline they define is given by:

$$C(t) = \sum_{i=1}^n B_{i,k}(t)P_i$$

where, n is the number of control points, $B_{i,k}$ are the B-spline basis functions of the curve's order k .

Knot Vectors

The knots are values of t that relate the control points to the parameter t . The list of knots is called a knot vector. If adjacent knots have the same value, then that value is said to have a multiplicity equal to the number of adjacent knots. For example, the knot values 0 and 9 have a multiplicity of two in the following knot vector:

(0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9)

Uniform curves have the knots at the ends of the knot vector with multiplicity equal to their order and their interior knots equally spaced. For example, the knot vector given above is a uniform knot vector for a second order curve. In this application we use uniform curves. The number of knots in the knot vector must be equal to the number of control points plus the order.

Curve Order

A curve's order defines the degree of the highest-order term in the B-spline basis function. The order is one more than the degree. That is, a third-order spline contains second-degree terms. The order indicates the number of adjacent control points that influence the position of any given point on the curve. In this application the user is asked to specify the order of the curves.

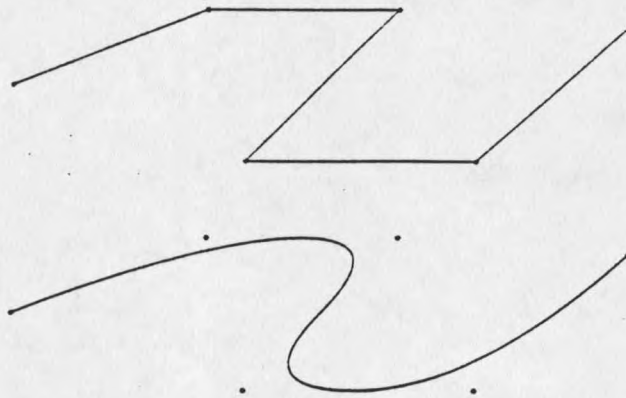


Figure 6.3: B-spline curves of two different orders using the same control points. (a) second order (b) third order (Courtesy of PEXlib Programming Manual, by Gaskins, O'Reilly and Associates, Sebastapool, CA, 1992)

Spline Surfaces

B-spline surfaces share all the advantages of the B-spline curves and are defined by the same basis functions. Therefore they have similar mathematical properties. B-spline surfaces are defined by two independent variables u and v as:

$$P = S(u, v)$$

where u and v are the independent variables or the parameters. The B-spline surface is defined by:

$$S(u, v) = \sum_{i=1}^n \sum_{j=1}^m B_{i,k}(u) B_{j,l}(v) P_{i,j}$$

where $B_{i,k}(u)$ is the basis function of order k , and $B_{j,l}(v)$ is the basis function of order j . $P_{i,j}$ are the control points (a grid of points). The number of control points in the u dimension is n and the number of control points in the v dimension is m . B-spline surfaces need two knot vectors, one for each dimension.

In this application the points generated are used as the grid points of the surface and the orders are specified by the user.

VIEWING

After the models are built for the objects they need to be displayed such that the user can get a good idea of the shape and orientation of the objects. This can be accomplished by presenting the viewer with different views of the objects. Three standard views namely top view, side view and front view are used in this application. These are the common views used when a three dimensional object is viewed in a two dimensional medium. Good examples are floor plans, machine drawings etc.

To understand how this is done we have to look at some of the underlying concepts in three dimensional viewing. Viewing involves projection and defining a view volume against which the 3-D world is clipped. These two operations make it possible to show an object on a 2-D screen. We will look at the Viewing Reference Coordinate (VRC) system which is used to define and display different views of objects.

Viewing Reference Coordinate System

The Viewer Reference Point (VRP) is the origin of the VRC system. One of the axes is defined called the View Plane Normal (VPN). These two define the projection plane (or view plane). The VPN defines the direction of projection. The other axis is defined by the projection of the View Up Vector (VUP) on the view plane. The VUP vector defines the orientation of the view plane. The center of projection, called Projection Reference Point (PRP) is defined in the VRC system. These are the four vital parameters used for defining different views. Changing the VRP moves the view plane.

Changing the VPN changes the direction of the view plane. And changing the VUP vector changes the orientation of the view plane. Finally changing PRP changes the viewer position. Figure 7.1 illustrates the use of these parameters in defining different views of the same object. These parameters were used by the application to produce three different views of the object(s) and display them simultaneously.

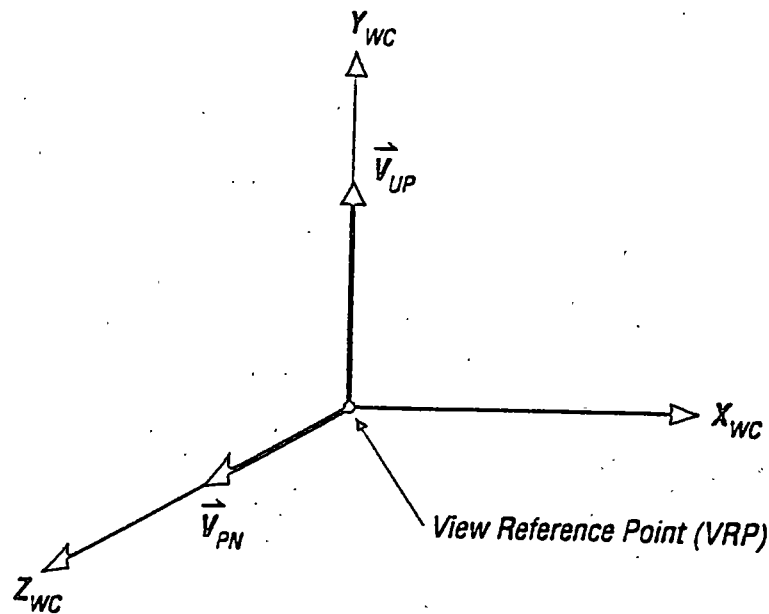


Figure 7.1: Viewing Reference Coordinate System (Courtesy of PEXlib Programming Manual, by Gaskins, O'Reilly and Associates, Sebastapool, CA, 1992)



Figure 7.2: Top View, Side view and Front View of the model

TRANSFORMATIONS

It is not enough if different views of 3-D objects are presented for the user. This gives the user a representation that is no better than one on paper. On the other hand if the objects can be shown in motion the user can better understand the details of the 3-D objects. Common transformations that are used in 3-D visualization are translation, scaling and rotation.

Translation

We can translate a point or a set of points by adding translation amounts to their coordinates of the points. Any object can be moved in X or Y or Z directions by adding translation amounts in the respective directions. The user is asked to select the direction of the transformations and the translation amount. This input from the user is used for transformation and display of the objects.

Scaling

Often the size of the objects displayed might not be satisfactory and may need to be resized to suit the need. This can be done by scaling. Points can be scaled (stretched) along any of the three axes by multiplying them by a scaling factor. Any object can be scaled in any direction by specifying the scaling factors in that particular direction. The user is asked to select the direction of scaling and the scaling factor. This information is used for transformation and display the objects.

Rotation

Three dimensional visualization is less effective if it is not possible to rotate the objects and display them. This transformation gives the illusion of 3-D on a 2-D screen. Points can be rotated about the origin through an angle. Rotation is also specified distinctly for the three axes and it is imperative that we define the axis of rotation along with the angle, when we want to rotate an object. Once again the user is asked for input to effect this transformation.

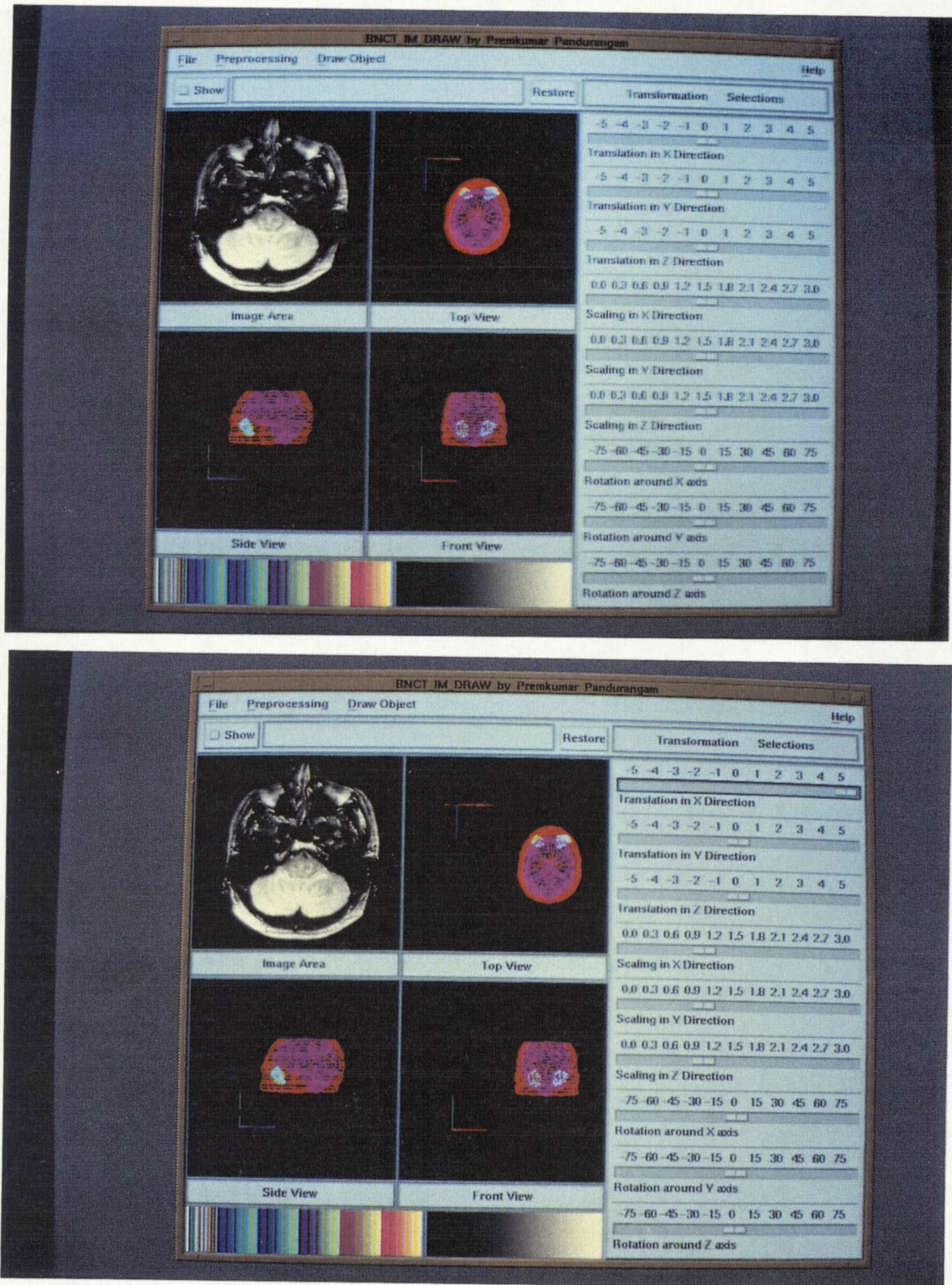


Figure 8.1: Translation (a) original object (b) effect of translation in X direction

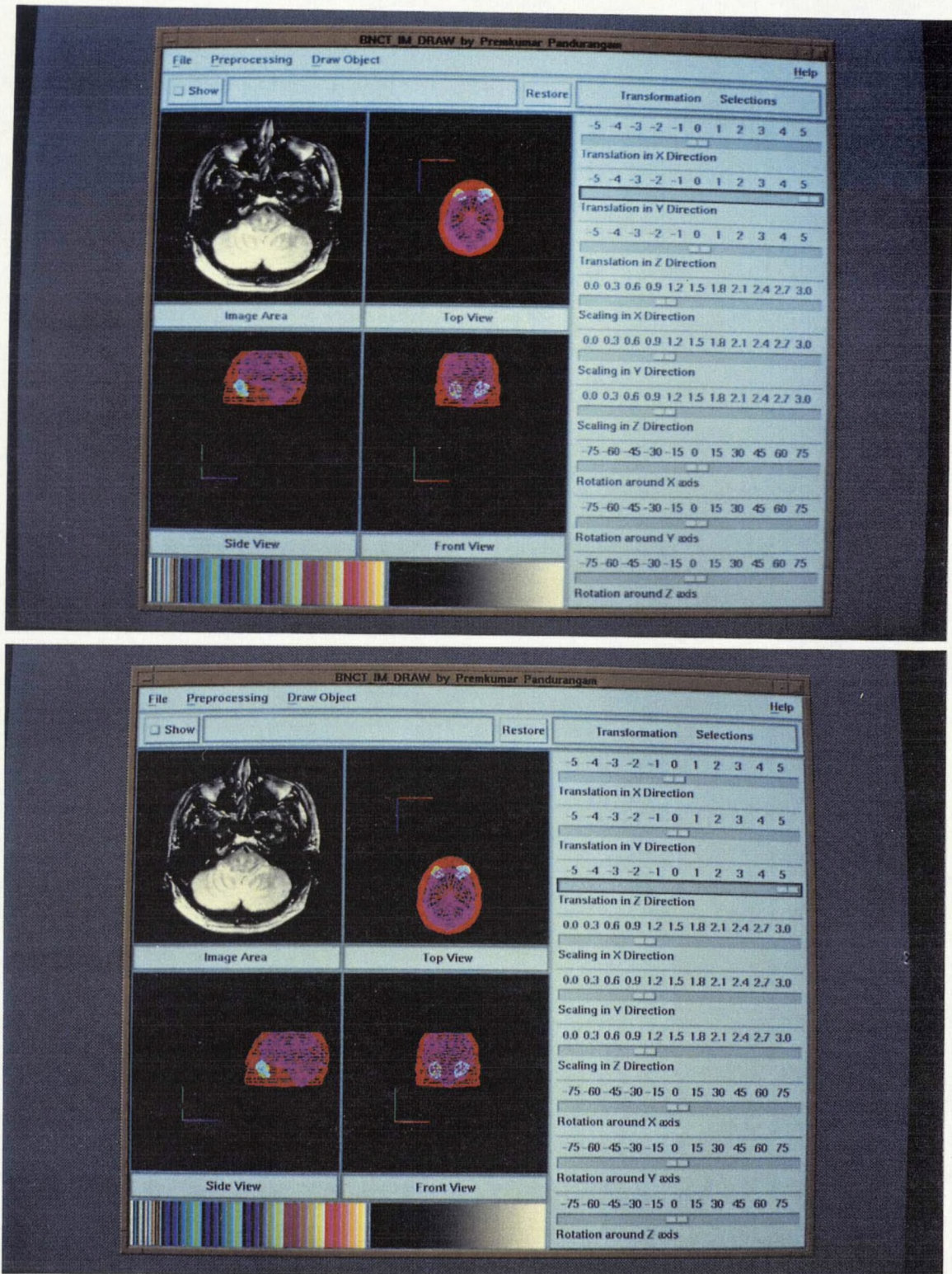


Figure 8.1: Translation (c) effect of translation in Y direction (d) effect of translation in Z direction

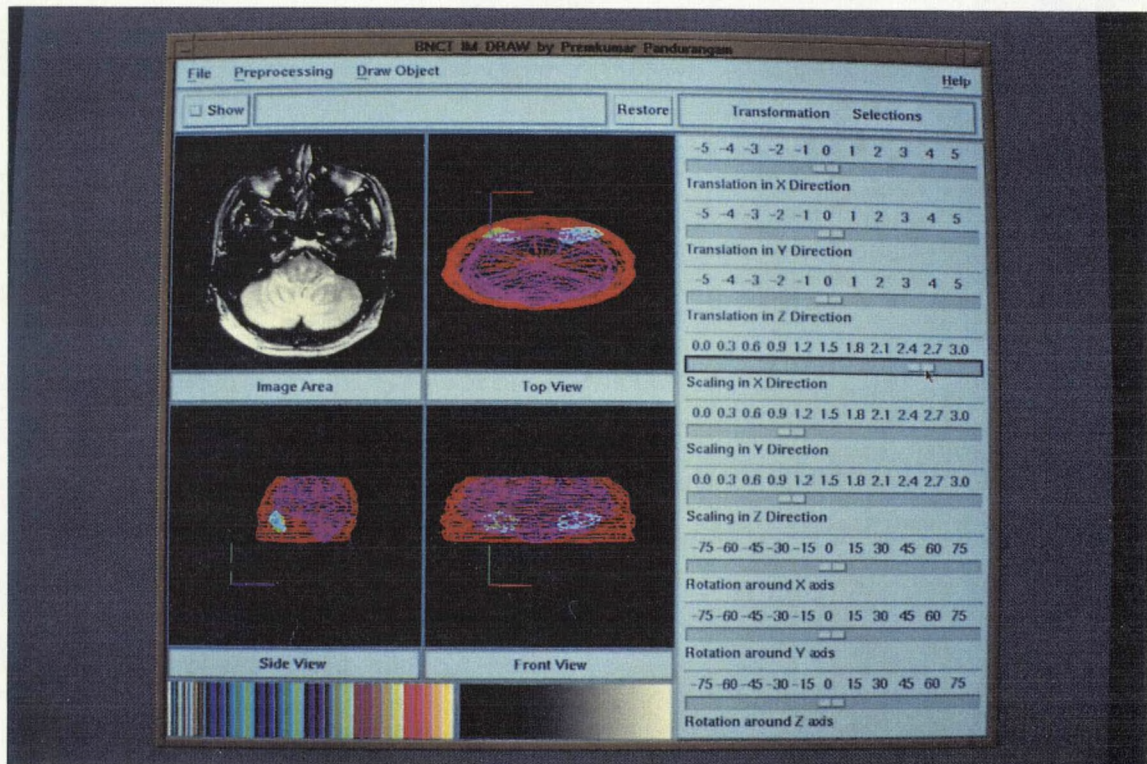


Figure 8.2 Scaling (a) original object (b) effect of scaling in X direction

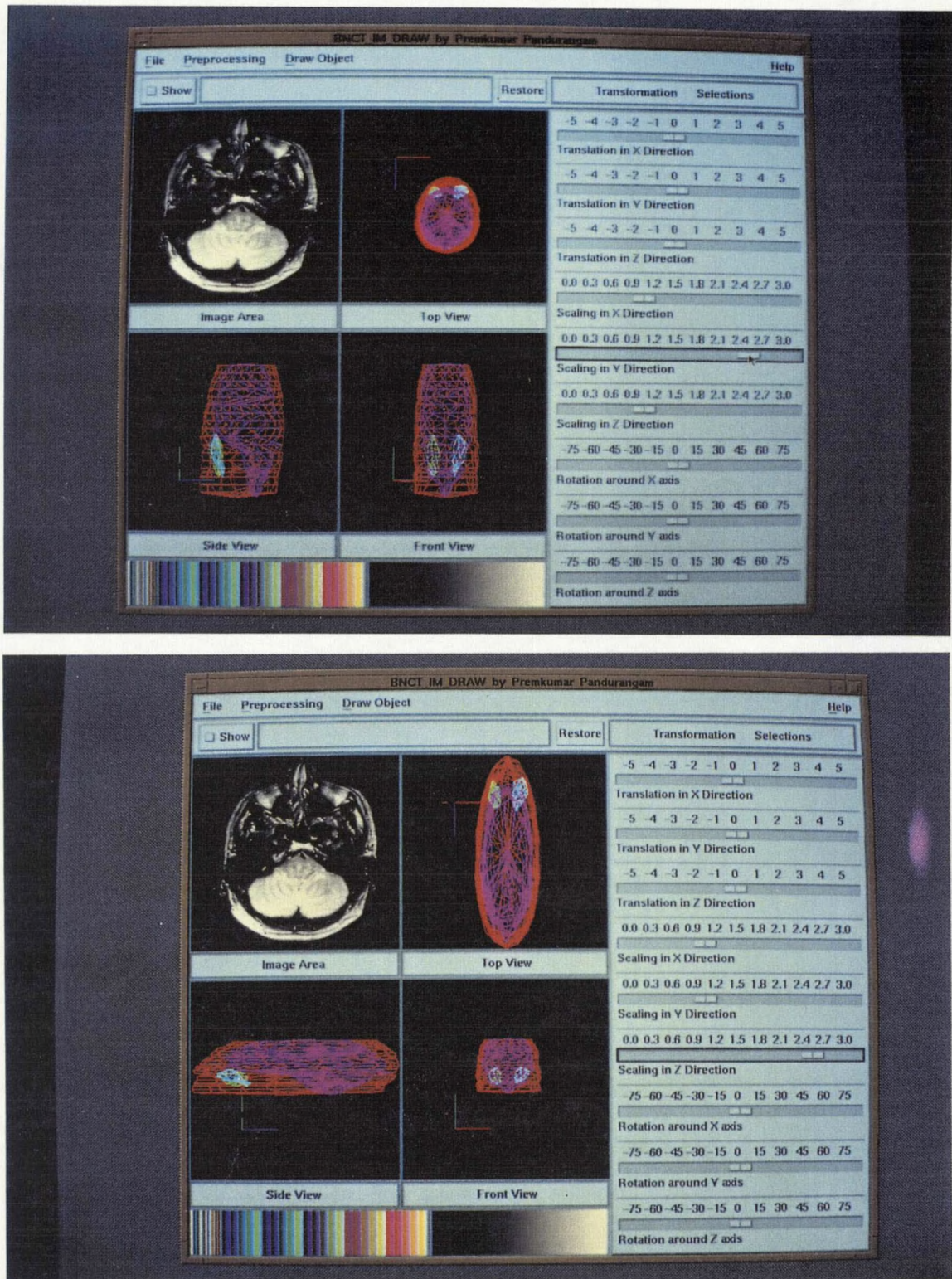


Figure 8.2: Scaling (c) effect of scaling in Y direction (d) effect of scaling in Z direction

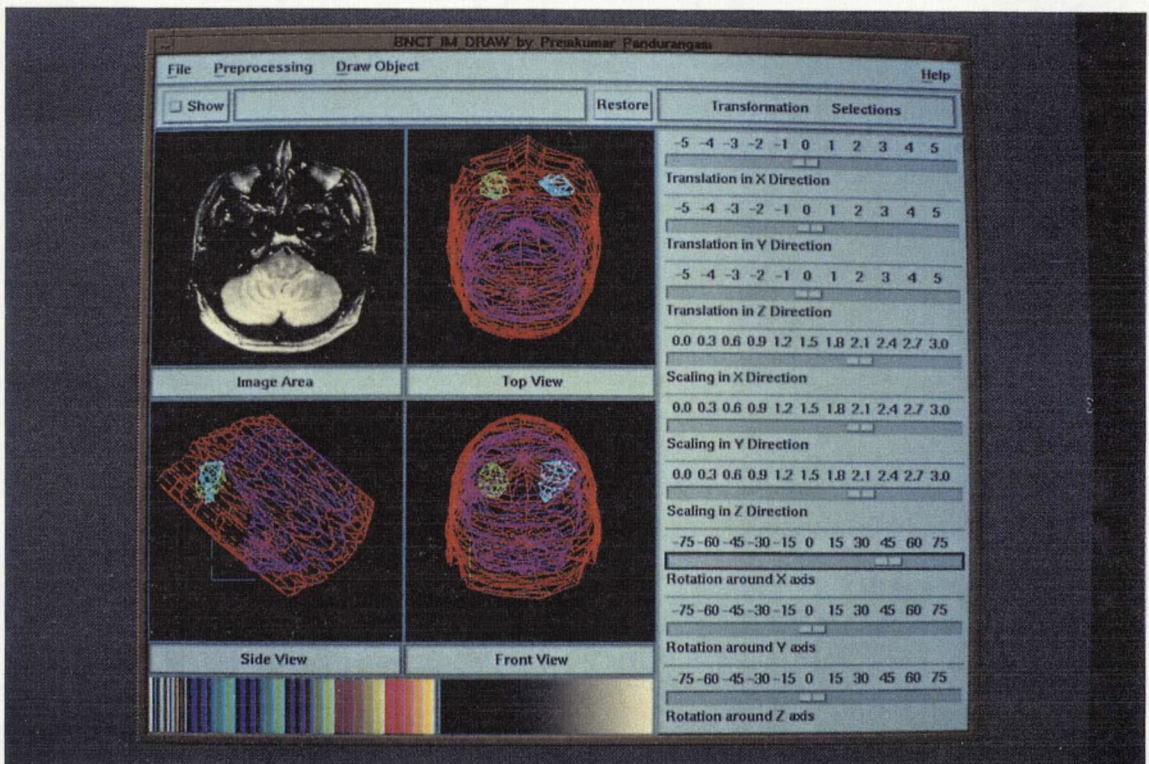
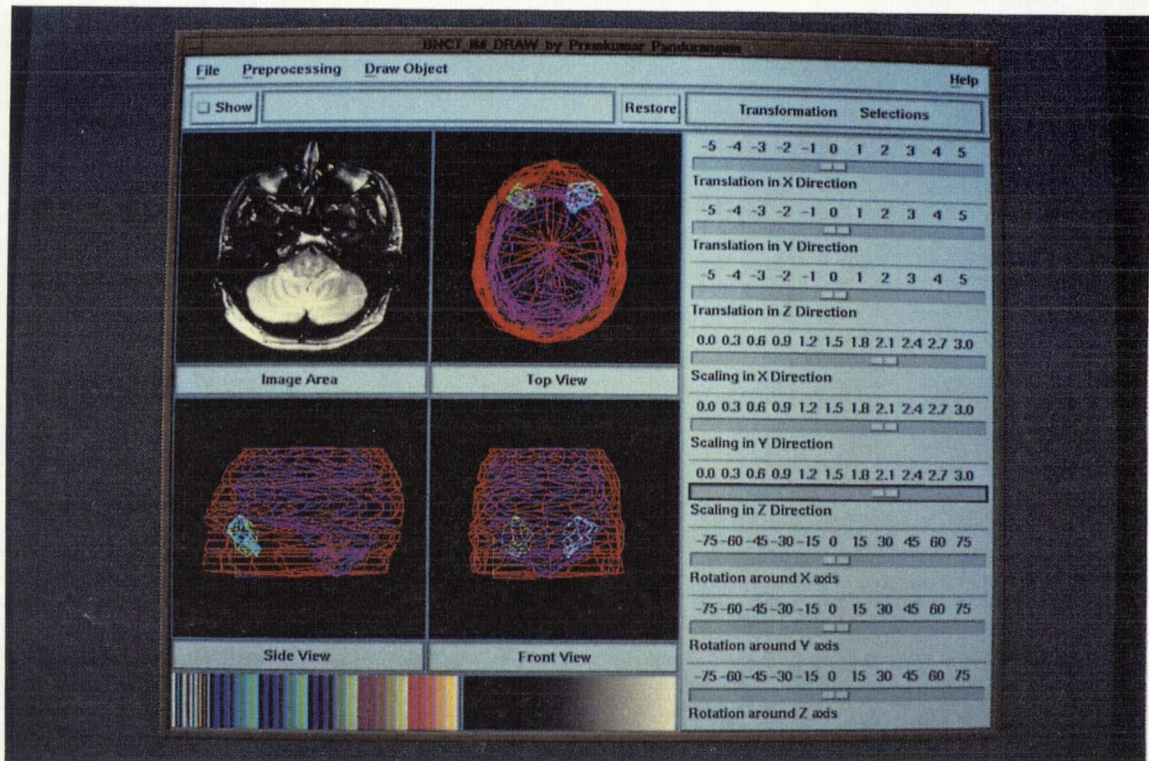
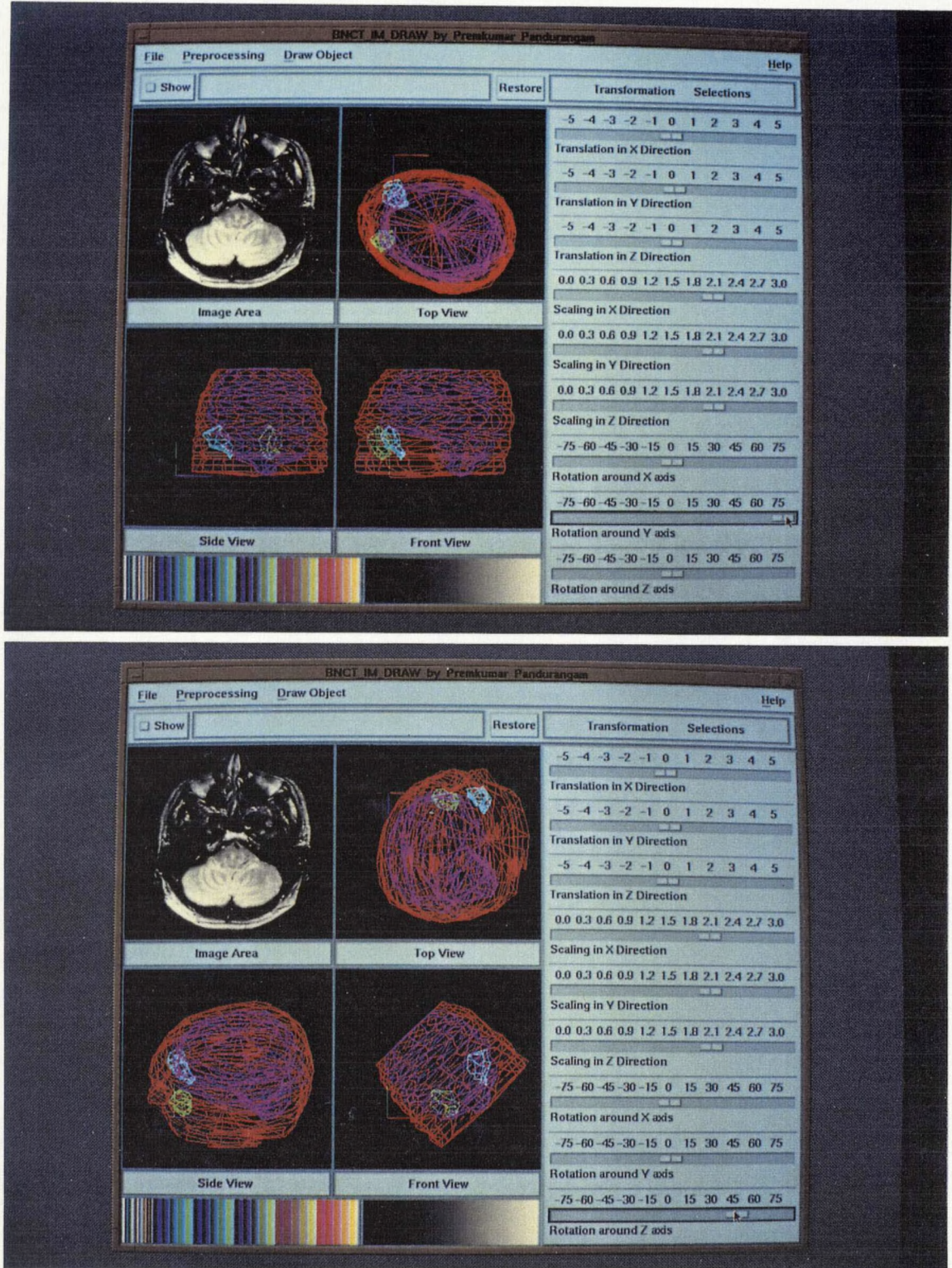


Figure 8.3: Rotation (a) original object (b) effect of rotation around X axis



CONCLUSION

We have seen how from a set of slice images of intensity levels we have been able to construct three dimensional models. We have also been able to view from different angles and transform the models by translating, scaling and rotating them.

We need to check on the promise of trading speed for accuracy and vice versa. This tradeoff might not be obvious at a first look. But a closer look would make this tradeoff obvious. To get an accurate representation, we would need an infinite number of slices and an infinite number of points in each slice. Stated in realistic terms we would need a large number of slices and a large number of points in each slice to get a good representation of the objects of interest. This would mean longer computation time for sorting the points and drawing the models. But if we choose to lower our accuracy expectations by settling for lesser number of slices and/or lesser number of points per slice, we can end up with an "acceptable" representation.

Of course, both these situations are quite possible and justifiably so. Imagine that a physician wants to take a quick look using the available M.R.I images of a patient whether the patient has a tumor or not and if present where. Here the accuracy requirement is not so high and the physician would rather not wait for long. Whereas if the physician wants to operate on the patient and wants to see if radiation therapy will be successful, then accuracy should be at its peak. And the physician wouldn't mind waiting for a long time. These are very different real world scenarios which dictate

different requirements. And a robust application should be able to cater to both scenarios. This was the logic behind letting the user select the number of slices and number of points in each slice. This way the user is left to decide if he/she would rather have speed or accuracy.

Significance of the work

The significance of this thesis is that image processing techniques are combined with rendering and visualization techniques to achieve the goal of three dimensional visualization of medical slice images. Equally significant, if not more significant is the usage and blending of emerging software technologies to achieve this. Particularly using PEXlib, a new software library which eases the pain of programming in 3-D graphics, along with Motif, a proven user interface building software library, is considered significant. This thesis shows the possibility and the advantage of combining these software libraries.

Scope for improvement

There are several ways this application can be made better. One area of improvement would be preprocessing. The user is asked to process one slice at a time and select points for each object of interest. This is cumbersome and time consuming. A good way of improving this process of selecting points would be automatic selection of points. It could work like this: step one could be identifying an object in some intensity level range, step two could be segmentation of this object in all the slices, step three could be thresholding in all the slices so that the object is totally isolated and step four could be selection of a certain number of points from all the slices. This process

could be repeated for all the objects of interest. A still better way could be that several objects could be selected at once and points generated simultaneously. The problem in doing these customized operations is the difference in intensities of the images. For example, the tumor can be brighter in one slice than it is in another. One way to get around this problem is to look at the intensity differences rather than the intensities of the objects. And of course this would be doomed if the contrasts are not equal in the images.

Other areas for enhancement are lighting and shading. These concepts are easy to implement and effective tools to give the user a more realistic view of the objects. Color is used in this application to distinguish between different objects. It would be interesting to look at different lighting and shading models and observe the quality of the visualization. These tools can give a better understanding of the objects rendered. As before this process can be made interactive and user friendly.

Another area to look at may be "picking". Every so often the need may arise to concentrate on one object (say, the tumor) more closely among the available objects. It is possible to "pick" an object and render it separately. This operation is also simple and can be implemented in this application with little effort.

REFERENCES CITED

Bartels, R.H., Beatty, J.C. and Barsky, B.A., "An Introduction to Splines for Use in Computer Graphics and Computer Modeling", Morgan Kaufmann, 1987.

Berlage, T., "OSF/Motif: Concepts and Programming", Addison-Wesley, 1991.

Brain, M., "Motif Programming: The Essentials and More", Digital Press, Burlington, MA, 1992.

Chui, C.K., "Multivariate Splines", Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

Earnshaw, R.A. and Wiseman, N., "An Introductory Guide to Scientific Visualization", Springer-Verlag, Berlin Heidelberg, 1992.

Ferguson, P.M., "Motif Reference Manual", O'Reilly and Associates, Sebastapool, CA, 1993.

Flanagan, D., "Programmer's Supplement for Release 5", O'Reilly and Associates, Sebastapool, CA, 1991.

Foley, J.D., van Dam, A., Feiner, S.K. and Hughes, J.F., "Computer Graphics: Principles and Practice", 2nd edition, Addison-Wesley, 1990.

Gaskins, T., "PEXlib Programming Manual", O'Reilly and Associates, Sebastapool, CA, 1992.

Gonzalez, R.C. and Woods, R.E., "Digital Image Processing", Addison-Wesley, 1992.

Herman, G.T, Zeng, J., and Bucholtz, C.A., "Shape-Based Interpolation", IEEE Computer Graphics and Applications, pp. 69-79, May 1990.

Jones, O., "Introduction to the X Window System", Prentice Hall, Englewood Cliffs, NJ, 1989.

Kernighan, B.W. and Ritchie, D.M., "The C Programming Language (ANSI-C Version)", 2nd edition, Prentice Hall, Englewood Cliffs, NJ, 1989.

Macovski, A., "Medical Imaging", Prentice Hall, Englewood Cliffs, NJ, 1983.

Mansfield, N., "The X Window System: A User's Guide", Addison-Wesley, 1991.

Mikes, S., "X Window System Technical Reference", Addison-Wesley, 1990.

Mikes, S., "X Window System Program Design and Development", Addison-Wesley, 1992.

Nye, A. and O'Reilly, T., "X Toolkit Intrinsic Programming Manual", O'Reilly and Associates, Sebastapool, CA, 1990.

Nye, A. and O'Reilly, T., "X Toolkit Intrinsic Reference Manual", O'Reilly and Associates, Sebastapool, CA, 1990.

Nye, A., "Xlib Programming Manual", O'Reilly and Associates, Sebastapool, CA, 1990.

Open Software Foundation, "OSF/Motif Programmer's Guide", Prentice Hall, Englewood Cliffs, NJ, 1990.

Open Software Foundation, "OSF/Motif Programmer's Reference", Prentice Hall, Englewood Cliffs, NJ, 1990.

Scheifler, R.W. and Gettys, J., "X Window System: The Complete Reference to Xlib; Protocol, ICCCM, XLFD", 2nd edition, Digital Press, Bedford, MA, 1990.

Schumaker, L.L., "Spline Functions: Basic Theory", John Wiley and Sons, NY 1981.

Talbott, S., "PEXlib Reference Manual", O'Reilly and Associates, Sebastapool, CA, 1992.

Udupa, J.K. and Herman, G.T., "3-D Imaging in Medicine", CRC Press, Boca Raton, FL, 1991.

MONTANA STATE UNIVERSITY LIBRARIES



3 1762 10261116 5

