CHAPTER-1

INTRODUCTION

1.1 Introduction:

Brain tumour segmentation means segregating tumour from non-tumour tissues. In medical imaging, it is one of the crucial steps in surgical and treatment planning. There are various types of malignant tumours such as astrocytoma, and metastatic, which vary greatly in appearance, size and location. Magnetic resonance (MR) sequences such as T1weighted, T2-weighted and contrast-enhanced T1-weighted scans provide different information about tumours. On these images, brain tumours appear either hypo intense (darker than brain tissue), or isointense (same intensity as brain tissue), or hyper intense (brighter than brain tissue).

The accurate estimation of tumour size is important for clinical reasons, e.g., for treatment planning and therapy evaluation. Although maximum tumour diameter is widely used as an indication of tumour size, it may not reflect a proper assessment of this tumour attribute because of the 3D nature and irregular shape of the tumour. Tumour volume, on the other hand, may be an appropriate representation of tumour size. Extract the vessel contours. Depending on the image quality and the general image artifacts such as noise, some segmentation methods may require image preprocessing prior to the segmentation algorithm. On the other hand, some methods apply post-processing to overcome the problems arising from over segmentation.

Vessel segmentation algorithms and techniques can be divided into six main categories, pattern recognition techniques, model-based approaches, tracking-based approaches, artificial intelligence- based approaches, neural network-based approaches, and miscellaneous tube-like object detection approaches. Pattern recognition techniques are further divided into seven categories, multi-scale approaches, skeleton-based approaches, region growing approaches, ridge-based approaches, differential geometry-based approaches, matching filters approaches, and mathematical morphology schemes. Clustering analysis plays an important role in scientific research and commercial application.

This thesis provides a survey of current vessel segmentation methods using clustering approach and provides both early and recent literature related to vessel segmentation algorithms and techniques.

1.2 Motivation:

Medical image partition is a well-studied field involving the definition of cells, tissues, organs and pathological structures. Partitioning can be used to identify tumour from MRI images, the cancer could be primary or secondary. The compound brain cancer can be divided into two general categories depending on the tumour beginning, their expansion and malignancy. Primary brain cancer is a cancer that begins from cells in the brain or from the covering of the brain tissues.

A malignancy or metastatic brain cancer occurs when cancer cells spread to the brain from primary cancer in another part of the body. A secondary brain cancer is life threatening. Mostly the secondary brain tumour is malignant. We are used MRI images for tumour detection. Therefore, the brain cancer diagnosis in an appropriate time is very necessary. As a rule, the motive of segmentation is to partition the picture into non-overlapping, constituent area, clusters, and subset that are homogeneous with admire to intensity and texture.

1.3 Definition of Image Segmentation:

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

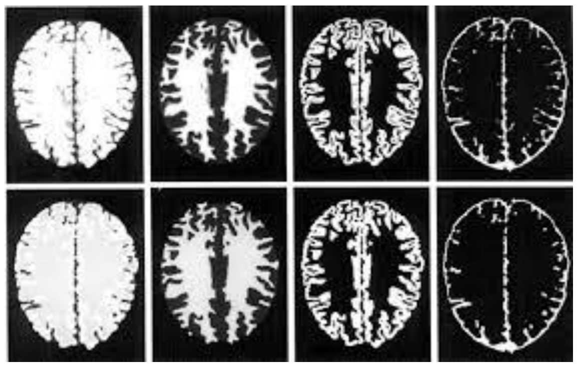


Figure 1.1 Segmented images

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (edge detection). Each of the pixels in a region are similar with respect to some characteristic or computed property, such as colour, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like Marching cubes.

1.4 Need of Image Segmentation:

Image segmentation is to locate the objects and partitioning the objects into multiple segments. It mostly assigns the pixel characteristics.

1.5 Objective and Goals:

- Enhanced information about brain tumour detection and segmentation.
- Deep Study of Techniques like performing a biopsy, performing imaging, like taking an MRI or CT scan of the brain will be done.

1.6 Software Requirements:

MATLAB

- MATLAB is a fourth-generation programming language and numerical analysis environment.
- It provides a comprehensive set of reference-standard algorithms and workflow applications for **image processing**, analysis, visualization and algorithm development.

1.7 Organizing of Thesis:

Chapter-1 explains about image segmentation, brain tumour.

Chapter-2 explains introduction to literature survey, overview of DIP, Image representation, reading and displaying images.

Chapter-3 explains about proposed methodology, block diagram, flow chart and morphological operations.

Chapter-4 explains about MATLAB implementation and tools.

Chapter-5 explains how we implemented our project, results and images.

Chapter-6 explains about conclusion and future scope of our project.

CHAPTER-2

LITERATURE SURVEY

2.1 Digital Image Processing:

Digital image processing is an area characterized by the need for extensive experimental work to establish the viability of proposed solutions to a given problem. An important characteristic underlying the design of image processing systems is the significant level of testing & experimentation that normally is required before arriving at an acceptable solution. This characteristic implies that the ability to formulate approaches quickly prototype candidate solutions generally plays a major role in reducing the cost & time required to arrive at a viable system implementation.

2.2 Overview Of DIP:

An image may be defined as a two-dimensional function f(x,y), where x & y are spatial coordinates, & the amplitude of f at any pair of coordinates (x.y) is called the intensity or gray level of the image at that point. When x,y & the amplitude values of f are all finite discrete quantities, we call the image a digital image. The field of DIP refers to processing digital image by means of digital computer Digital image is composed of a finite number of elements, each of which has a particular location value. The elements are called pixels.

Vision is the most advanced of our sensor, so it is not surprising that image play the single most important role in human perception. However, unlike humans who are limited to the visual band of the EM spectrum imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves, they can operate also on images generated by sources that humans are not accustomed associating with image.

There is no general agreement among authors regarding where image processing stops & other related areas such as image analysis computer vision start. Sometimes a distinction is made by defining image processing as a discipline in which both the input & output at a process are images. This is limiting somewhat artificial boundary. The area of image analysis (image understanding) is in between image processing & computer vision.

There are no clear-cut boundaries in the continuum from image processing at one end to complete vision at the other. However, on useful paradigm is to consider three types of computerized processes in this continuum: low-, mid- & high-level processes. Low-level process involves primitive operations such as image processing to reduce noise, contrast enhancement & image sharpening. A low-level process is characterized by the fact that both its inputs & outputs are images. Mid-level process on images involves tasks such as segmentation, description of that object to reduce them to a form suitable for computer processing & classification of individual objects. A mid-level process is characterized by the fact that its inputs generally are images but its outputs are attributes extracted from those images.

Finally, higher level processing involves "Making sense of an ensemble of recognized objects, as in image analysis & at the far end of the continuum performing the cognitive functions normally associated with human vision. Digital image processing, as already defined is used successfully in a broad range of areas of exceptional social & economic value

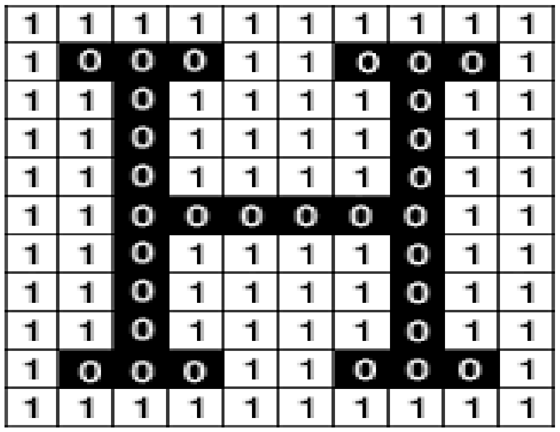
2.3 Image Representation:

An image is represented as a two dimensional function f(x,y) where x and yare spatial co-ordinates and the amplitude off at any pair of coordinates (x,y) is called the intensity of the image at that point.

A digital image is a computer file that contains graphical information instead of text or a program. Pixels are the basic building blocks of all digital images. Pixels are small adjoining squares in a matrix across the length and width of our digital image. They are so small that don't see the actual pixels when the image is on our computer monitor.

Pixels are monochromatic. Each pixel is a single solid colour that is blended from some combination of the 3 primary colours of Red, Green, and Blue. So, every pixel has a RED component, a GREEN component and BLUE component. The physical dimensions of a digital image are measured in pixels and commonly called pixel or image resolution are scalable to different physical sizes on our computer monitor or on n photo print. However, all of the pixels in any particular digital image are the same size. Pixels as represented in a printed photo become round slightly overlapping dots.

Pixel Values: As shown in this bitonal image, each pixel is assigned a tonal value, in this example 0 for black and 1 for white.



1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	1	1	0	1	1
1	1	0	1	1	1	1	0	1	1
1	1	0	1	1	1	1	0	1	1
1	1	0	0	0	0	0	0	1	1
1	1	0	1	1	1	1	0	1	1
1	1	0	1	1	1	1	0	1	1
1	1	0	1	1	1	1	0	1	1
1	0	0	0	1	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1

Figure 2.1 Bitonal Image

2.4 Pixel Dimensions:

The horizontal and vertical measurements of an image expressed in pixels. The pixel dimensions may be determined by multiplying both the width and the height by the dpi. The term "pixel dimensions" here, to me, is confusing because it sounds like we're talking about the dimensions of each individual pixel, and that's not the case. Pixel dimensions measure the total number of pixels along an image’s width and height. Resolution is the fineness of detail in a bitmap image and is measured in pixels per inch (ppi). A digital camera will also have pixel dimensions, expressed as the number of pixels horizontally and vertically that define its resolution (e.g., 2,048 by 3,072). Calculate the dpi achieved by dividing a document's dimension into the corresponding pixel dimension against which it is aligned.

Example:

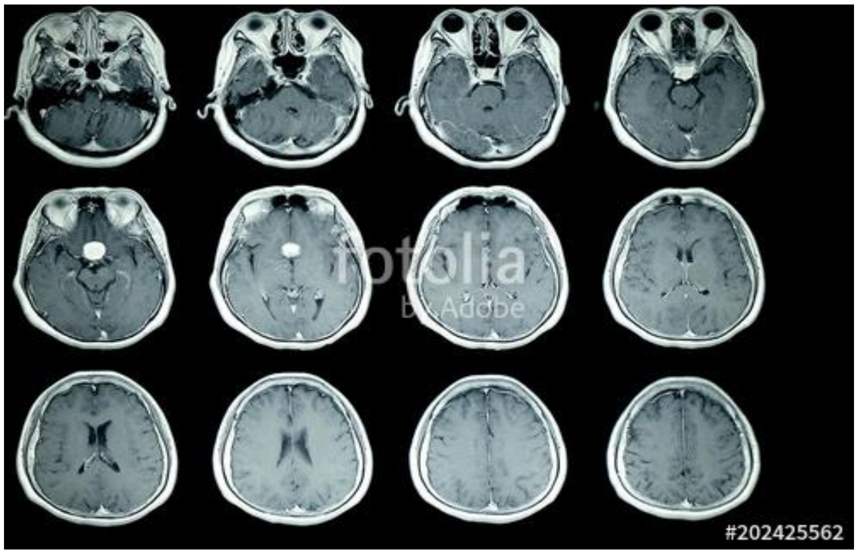


Figure 2.2 MRI Image

2.4.1 Gray Scale Image:

A grayscale image is a function 1(x,y) of the two spatial coordinates of the image plane. I(x,y) is the intensity of the image at the point (x,y) on the image plane1(x,y) takes non-negative values assume the image is bounded by a rectangle (0, a]*[0, b]I : [0, a]* [0, b]→ [0. info).

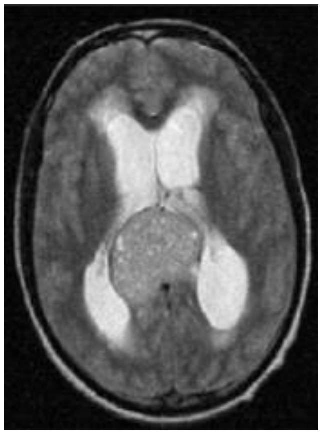


Figure 2.3 Brain tumour gray image

2.4.2 Colour Image:

It can be represented by three functions, R(x,y) for red, G (x.y) for green andB (xy) for blue. An image may be continuous with respect to the x and y coordinates and also in amplitude. Converting such an image to digital form requires that the coordinates as well as the amplitude to be digitized Digitizing the coordinate values is called sampling. Digitizing the amplitude values is called quantization.

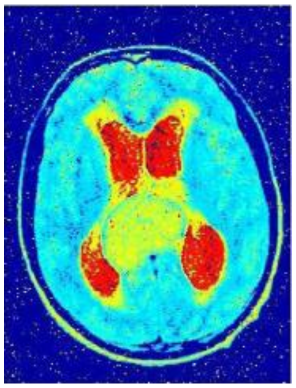


Figure 2.4 Brain colour image

2.4.3 Coordinate Convention:

The result of sampling and quantization is a matrix of real numbers. We use two principal ways to represent digital images. Assume that an image fx.) is sampled so that the resulting image has M rows and N columns. We say that the image is of size M X N. The values of the coordinates (xy) are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates. In many image processing books, the image origin is defined to be at (xy)-(0,0). The next coordinate values along the first row of the image are (xy) (0,1). It is important to keep in mind that the notation (0,1) is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Following figure shows the coordinate convention. Note that x ranges from 0 to M-1 and y from 0 to N.1 in integer increment. The coordinate convention used in the toolbox to denote arrays is different from the preceding paragraph in two minor ways. First, instead of using (x,y) the toolbox uses the notation (race) to indicate rows and columns Note, however, that the order of coordinates is the same as the order discussed in the previous paragraph, in the sense that the first element of a coordinate topples, (alb), refers to a row and the second to a column. The other difference is that the origin of the coordinate system is at (r.) (1.1); thus, ranges from I to M and c from 1 to N in integer increments IPT documentation refers to the coordinates. Less frequently the toolbox also employs another coordinate convention called spatial coordinates which uses x to refer to columns and y to refers to rows. This is the opposite of our use of variables x and y.

2.5 Image as Matrices

The preceding discussion leads to the following representation for a digitized image function:

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(o,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \qquad \rightarrow \text{Eq (1)}$$

The right side of this equation is a digital image by definition. Each element of this array is called an image element, picture element, and pixel. The terms image and pixel are used throughout the rest of our discussions to denote a digital image and its elements.

Where f (1,1) =f (0,0) (mote the use of a monoscope font to denote MATLAB quantities). Clearly the two representations are identical, except for the shift in origin. The notation f(p,q) denotes the element located in row p and the column q. For example, 1(6,2) is the element in the sixth row and second column of the matrix Typically we use the letters M and N respectively to denote the number of rows and columns in a matrix. A 1 X N matrix is called a row vector whereas an M X 1 matrix is called a column vector. A l X l matrix is a scalar.

Matrices in MATLAB are stored in variables with names such as A, a, RGB, real array and so on. Variables must begin with a letter and contain only letters. numerals and underscores. As noted in the previous paragraph. all MATLAB quantities are written using mono-scope characters. We use conventional Roman italic notation such as fix y), for mathematical expressions.

2.6 Image Coordinate Systems:

2.6.1 Pixel Coordinates:

Generally, the most convenient method for expressing locations in an image to use pixel coordinates. In this coordinate system, the image is treated as a grid of discrete elements, ordered from top to bottom and left to right, as illustrated by the following figure.

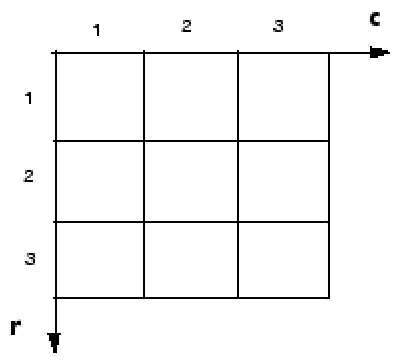


Figure 2.5 The Pixel Coordinate System.

For pixel coordinates, the first component (the row) increases downward while the second component c (the column) increases to the right. Pixel coordinates are integer values and range between 1 and the length of the row or column There is a one-to-one correspondence between pixel coordinates and the coordinates MATLAB uses for matrix subscripting. This correspondence makes the relationship between an image's data matrix and the way the image is displayed easy to understand. For example, the data for the pixel in the fifth row, second column is stored in the matrix element (5,2). We use normal MATLAB matrix subscripting to access values of individual pixels. For example, the MATLAB code 1 (2, 15), returns the value of the pixel at row 2 column 15 of the image I.

2.6.2 Spatial Coordinates:

In the pixel coordinate system, a pixel is treated as a discrete unit, uniquely identified by a single coordinate pair, such as (5,2). From this perspective, a location such as (5.3.2.2) is not meaningful. At times, however, it is useful to think of a pixel as a square patch. From this perspective, a location such as (5.3, 2.2) is meaningful, and is distinct from (5, 2). In this spatial coordinate system, locations in an image are positions on a plane. And they are described in terms of x and y (not r and c as in the pixel coordinate system)**.** The following figure illustrates the spatial coordinate system med for images. Notice that y increases downward.

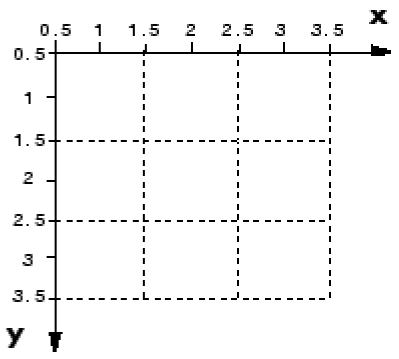


Figure 2.6 The Spatial Coordinate System

This spatial coordinate system corresponds closely to the pixel coordinate system in many ways. For example, the spatial coordinates of the center point of any pixel are identical to the pixel coordinates for that pixel. There are some important differences, however. In pixel coordinates, the upper left corner of an image is (1,1), while in spatial coordinates, this location by default is (0.5,0.5). This difference is due to the pixel coordinate system's being discrete, while the spatial coordinate system is continuous. Also, the upper left corner is always (1.1) in pixel coordinates, but we can specify a non-default origin for the spatial coordinate system. Another potentially confusing difference is largely a matter of convention: the order of the horizontal and vertical components is reversed in the notation for these two systems. As mentioned earlier, pixel coordinates are expressed as it c), while spatial coordinates are expressed as (x.y). In the reference pages, when the syntax for a function uses and c, it refers to the pixel coordinate system. When the syntax uses x and y, it refers to the spatial coordinate system.

2.7 Reading Images:

Images are read into the MATLAB environment using function imread whose syntax is →**imread('filename');**

Table 2.1 Image Formats

Format	**Filename Extension**
Tagged Image File Format (TIFF)	**.tiff, .tif**
Joint Photographic Experts Group (JPEG)	**.jpeg, .jpg**
Graphic Interchange Format (GIF)	**.gif**
Portable Network Graphic (PNG)	**.png**
Windows Bitmap Format (DIB)	**.bmp, .DMPF**
Windows Icon Format	**.ioo**
Windows Cursor	**.our**

Digital Image File Types:

The 5 most common digital image file types are as follows:

- **JPEG** is a compressed file format that supports 24-bit colour (millions of colours) This is the best format for photographs to be shown on the web or as email attachments. This is because the colour informational bits in the computer file are compressed (reduced) and download times are minimized.
- **GIF** is an uncompressed file format that supports only 256 distinct colours. Best used with web clip art and logo type images. GIF is not suitable for photographs because of its limited colour support.
- **TIFF** is an uncompressed file format with 24- or 48-bit colour support Uncompressed means that all of the colour information from our scanner or digital camera for each individual pixel is preserved when we save as TIFF. TIFF is the best format for saving digital images that we will want to print. Tiff supports embedded file information, including exact colour space, output profile information

and EXIF data. There is a lossless compression for TIFF called LZW. LZW is much like zipping the image file because there is no quality loss. An LZW TIFF decompresses (opens) with all of the original pixel information unaltered.

- **BMP** is a Windows (only) operating system uncompressed file format that supports 24-bit colour. BMP does not support embedded information like EXIF. calibrated colour space and output profiles. Avoid using BMP for photographs because it produces approximately the same file sizes as TIFF without any of the advantages of TIFF.
- **Camera RAW** is a lossless compressed file format that is proprietary for each digital camera manufacturer and model. A camera RAW file contains the 'raw data from the camera's imaging sensor. Some image editing programs have their own version of RAW too. However, camera RAW is the most common type of RAW file the advantage of camera RAW is that it contains the full range of colour information from the sensor. This means the RAW file contains 12 to 14 bits of colour information for each pixel If we shoot JPEG, we only get 8 bits of colour for each pixel. These extra colour bits make shooting camera RAW much like shooting negative film. We have a little more latitude in setting our exposure and a slightly wider dynamic range.

Here filename is a spring containing the complete of the image file including any applicable extension for example the command line **→f-imread (‘leaf jpg');**

Reads the JPEG (above table) image leaf into image array. Note the use of single quotes () to delimit the string filename. The semicolon at the end of a command line is used by MATLAB for suppressing output. If a semicolon is not included. MATLAB displays the results of the operation) specified in that line. The prompt symbol >> designates the beginning of a command line, as it appears in the MATLAB command window.

When as in the preceding command line no path is included in filename. imread reads the file from the current directory and if that fails it tries to find the file in the MATLAB search path. The simplest way to read an image from a specified directory is to include a full or relative path to that directory in filename.

For example:

→ f imread (‘D:\myimages\leaf jpg’);

Reads the image from a folder called my images on the D: drive, whereas

→f imread(‘.\myimages\leaf jpg’);

Reads the image from the myimages subdirectory of the current of the current working directory. The current directory window on the MATLAB desktop toolbar displays MATLAB's current working directory and provides a simple, manual way to change it. Above table lists some of the most of the popular image/graphics formats supported by imread and imwrite.

Function size gives the row and column dimensions of an image:

→size (f) ans = 1024*1024

This function is particularly useful in programming when used in the following

form to determine automatically the size of an image

→[M, N]-size ()

This syntax returns the number of rows(M) and columns () in the image.

The whole function displays additional information about an array. For

instance, the statement

Whose F gives

Table 2.2 Image ‘Brain.png’ format

Name	Size	Bytes	Class
F	1024*1024	1048576	Unit8 array

Grand total is 1048576 elements using 1048576 bytes. The unit's entry shown refers to one of several MATLAB data classes. A semicolon at the end of a whose line has no effect so normally one is not used.

2.8 Displaying Images:

Images are displayed on the MATLAB desktop using function imshow. Which has the basic syntax**:→imshow(f.g)**. Where is an image array, and g is the number of intensity levels used to display it. If g is omitted it defaults to 256 levels using the syntax **→imshow(f. {low high})**. Displays as black all values less than or equal to low and as white all values greater than or equal to high. The values in between are displayed as intermediate intensity values using the default number of levels Finally the syntax**→imshow(f,[ ])**. Sets variable low to the minimum value of array and high to its maximum value. This form of imshow is useful for displaying images that have a low dynamic range or that have positive and negative values. Function pixel is used frequently to display the intensity values of individual pixels interactively. This function displays a cursor overlaid on an image. As the cursor is moved over the image with the mouse the coordinates of the cursor position and the corresponding intensity values are shown on a display that appears below the figure window When working with colour images, the coordinates as well as the red, green and blue components are displayed. If the left button on the mouse is clicked and then held pressed, pixel displays the Euclidean distance between the initial and current cursor locations, the syntax form of interest here is Pixel which shows the cursor on the last image displayed. Clicking the X button on the cursor window turns it off. The following statements read from disk image called rose 512.tif extract basic information about the image and display it using imshow:

→f=imread('brain _512.tif');

whose f is

Table 2.3 ‘imagebrain_512.tif’ format

Name	Size	Bytes	Class
F	512*512	262144	Unit8 array

Grand total is 262144 elements using 262144 bytes

→imshow (F)

A semicolon at the end of an imshow line has no effect, so normally one is not used. If another image g, is displayed using imshow, MATLAB replaces the image in the screen with the new image. To keep the first image and output a second image, we use function figure as follows:

→figure, imshow(g)

Using the statement

→imshow(F), figure ,imshow(g) displays both images.

Note that more than one command can be written on a line as long as different commands are properly delimited by commas or semicolons. As mentioned earlier, a semicolon is used whenever it is desired to suppress screen outputs from a command line. Suppose that we have just read an image h and find that using imshow produces the image. It is clear that this image has a low dynamic range, which can be remedied for display purposes by using the statement.

→imshow (h, [])

2.9 Writing Images:

Images are written to disk using function imwrite, which has the following basic syntax:**→Imwrite (‘filename');**

With this syntax, the string contained in filename must include a recognized file format extension Alternatively: the desired format can be specified explicitly with third input argument. inwrite ('patient_10_run.tif)**.**

Or alternatively

For example, the following command writes f to a TIFF file named patient 10_run1:

→imwrite (f,'patient_10_runl.tif');

If filename contains no path information, then imwrite saves the file in the current working directory. The imwrite function in have other parameters depending on c file format. Selected. Most of the work in the following deals either with JPEG or TIFF images so we focus attention here en these two formats More general imwrite syntax applicable only to JPEG images is→imwrite (f,'filename.jpg', 'quality' q); where is an integer between 0 and 100(the lower number the higher the degradation due to JPEG compression). For example, for q-25 the applicable syntax is

→imwrite (f, 'bubbles25 jpg', quality',25)

The image for q 15 has false contouring that is barely visible, but this effect becomes quite pronounced for 5 and q 0. Thus, an expectable solution with some margin for error is to compress the images with q 25. In order to get an idea of the compression achieved and to obtain other image file details.

2.10 Image Types:

The toolbox supports four types of images

- 1 Intensity images
- 2. Binary images
- 3. Indexed images
- 4. RGB images

Most monochrome image processing operations are carried out using binary or intensity images, so our initial focus is on these two image types. Indexed and RGB colour images.

2.10.1 Intensity Images:

An intensity image is a data matrix whose values have been scaled to represent intentions. When the elements of an intensity image are of class units or class unit16, they have integer values in the range (0,255) and [0, 65535), respectively. If the image is of class double, the values are floating point numbers Values of scaled, double intensity images are in the range [0, 1] by convention**.** The matrix can be of class unit8, uint16, int16, single, or double. While grayscale images are rarely saved with a colour map, MATLAB uses a colour map to display them.For a matrix of class single or double, using the default grayscale colour map, the intensity 0 represents black and the intensity represents white. For a matrix of type uint8, um 16, or int16, the intensity intmin (class (I) represents black and the intensity intmax (class (T)) represents white.

The figure below depicts a grayscale image of class double.

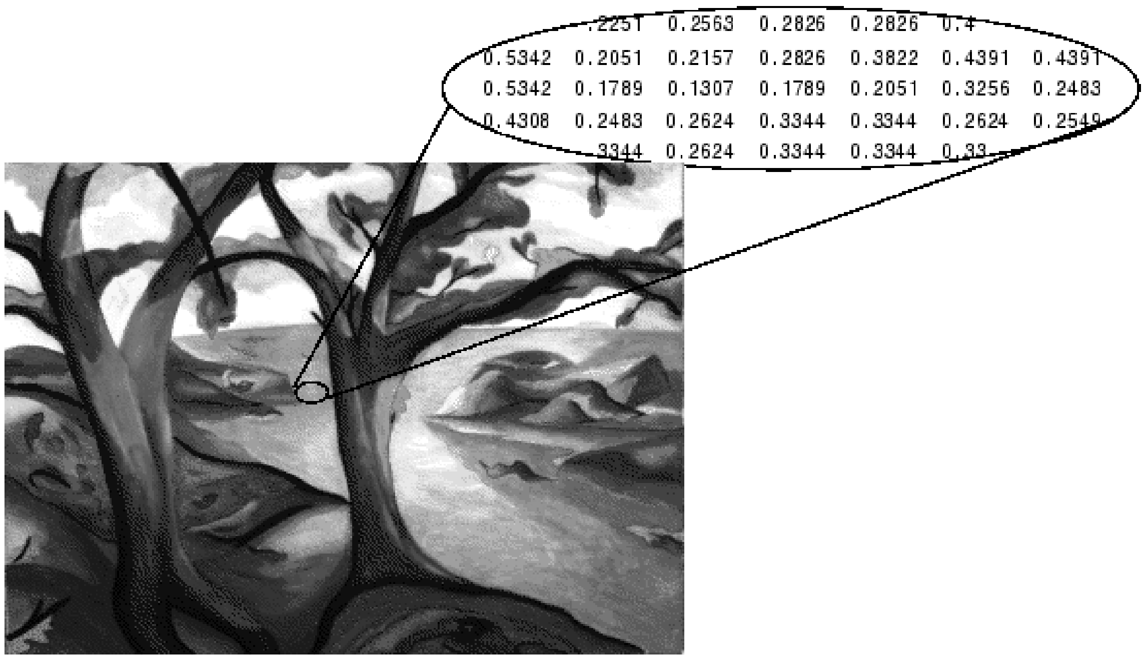


Figure 2.7 Pixel Values in Grayscale, Image Define Gray Levels.

2.10.2 Binary Images:

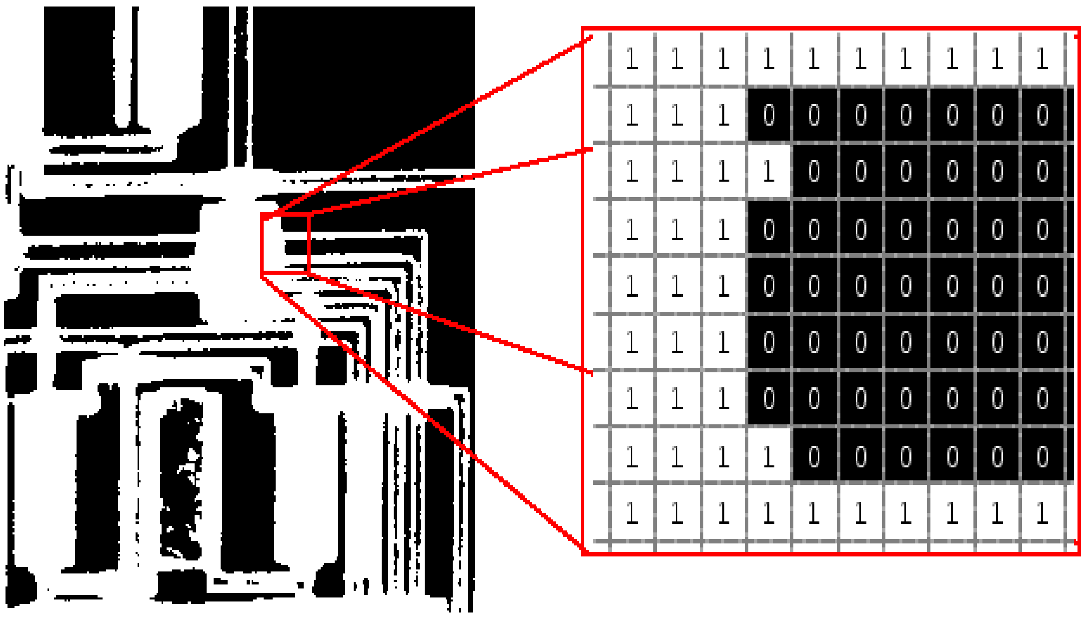
Binary images have a very specific meaning in MATLAB binary image is a logical array 0s and. Thus, an array of 0s and Is whose values are of data class, say units, is not considered as a binary image in MATLAB A numeric array is converted to binary using function logical. Thus, if A is a numeric array consisting of 0s and 1 s**, we** create an array B using the statement

B=logical (A)

If A contains elements other than O s and 1 s Use of the logical function converts all nonzero quantities to logical Is and all entries with value 0 to logical 0s. Using relational and logical operators also creates logical arrays. To test if an array is logical, we use the 1 logical function:

islogical(c)

If c is a logical array, this function returns a 1 Otherwise returns a 0. Logical array can be converted to numeric arrays using the data class conversion functions. In a binary image, each pixel assumes one of only two discrete values: 1 or 0. A binary image is stored as a logical array. By convention, this documentation uses the variable name BW to refer to binary images. The following figure shows a binary image with a close-up view of some of the pixel values.



1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1

Figure 2.8 Pixel Values in a Binary Image.

2.10.3 Indexed Images:

An indexed image has two components:

- A data matrix integer, x.
- A colour map matrix, map.

Matrix map is an m 3 arrays of class double containing floating point values in the range [0, 1]. The length of the map is equal to the number of colours it defines. Each row of map specifies the red, green and blue components of a single-colour. Indexed images use direct mapping of pixel intensity values colour map values. The colour of each pixel is determined by using the corresponding value the integer matrix as a pointer in to map. If x is of class double then all of its components with values less than or equal to point to the first row in map all components with value 2 point to the second row and so on. The following figure illustrates the structure of an indexed image to the figure, the image matrix is of class double, so the value 5 points to the throw of the colour map.

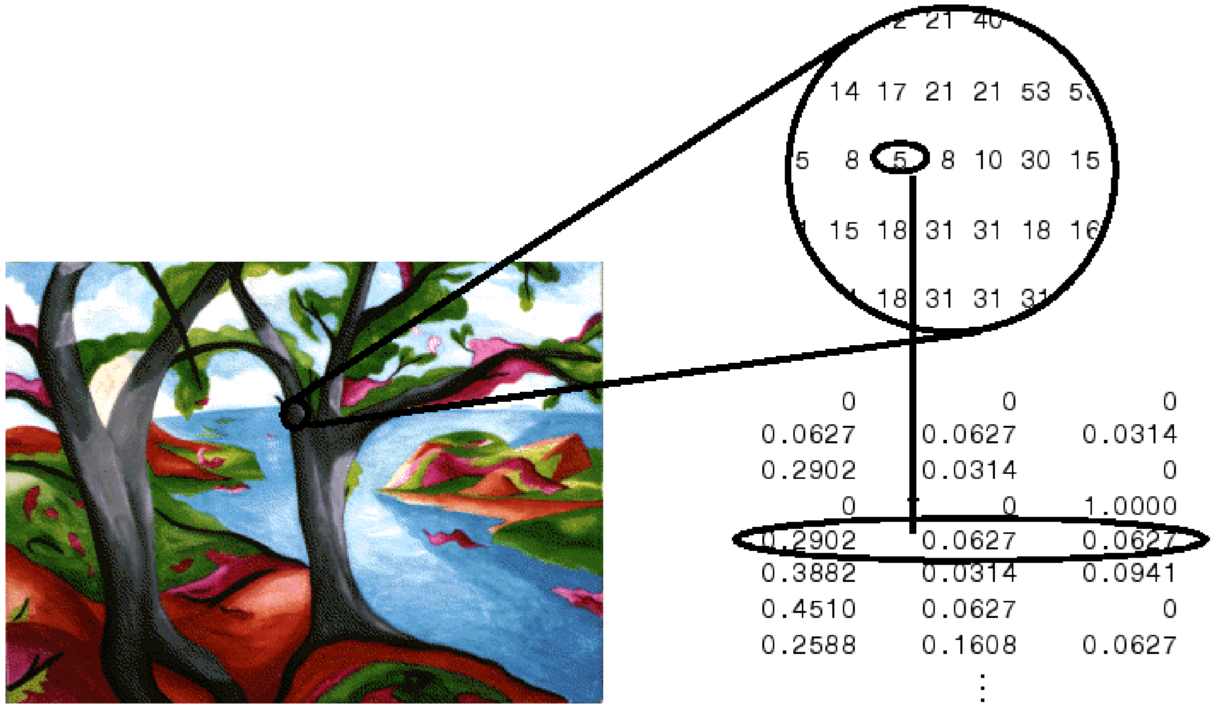


Figure 2.9 Pixel value to Colourmap Entries in Indexed Images

2.10.4 RGB Image:

An RGB colour image is an M*N 3 any of colour pixels where each colour pixel is triplet corresponding to the red, green and blue components of an RGB image, at a specific spatial location An RGB image may be viewed as stack of three gray scale images that when fed in to the red, green and blue inputs of a colour monitor**.**

Produce a colour image on the screen. Convention the three images forming an RGB colour image are referred to as the red, green and blue components images. The data class of the components images determines their range of values. If an RGB image is of class double the range of values is [0, 1].

Similarly, the range of values is [0,255] or [0, 65535) For RGB images of class units or unit 16 respectively. The number of bits use to represents the pixel values of the component images determines the bit depth of an RGB image. For example, if each component image is an 8-bit image, the corresponding RGB image is said to be 24 bits deep. Generally, the number of bits in all component images is the same. In this case the number of possible colours in RGB image is b) where is a number of bits in each component image. For the Batcave the number is 16777216 colours.

A colour array can be of class uint8, uint16, single, or double in a colour array of class single or double, each colour component is a value between 0 and 1. The three-colour components for each pixel are stored along the third dimension of the data array. For example, the red, green, and blue colour components of the pixel (10,5) are stored in RGB (10,5,1), RGB (10,5.2), and RGB (10,5,3), respectively**.**

The following figure depicts a colour image of class double.

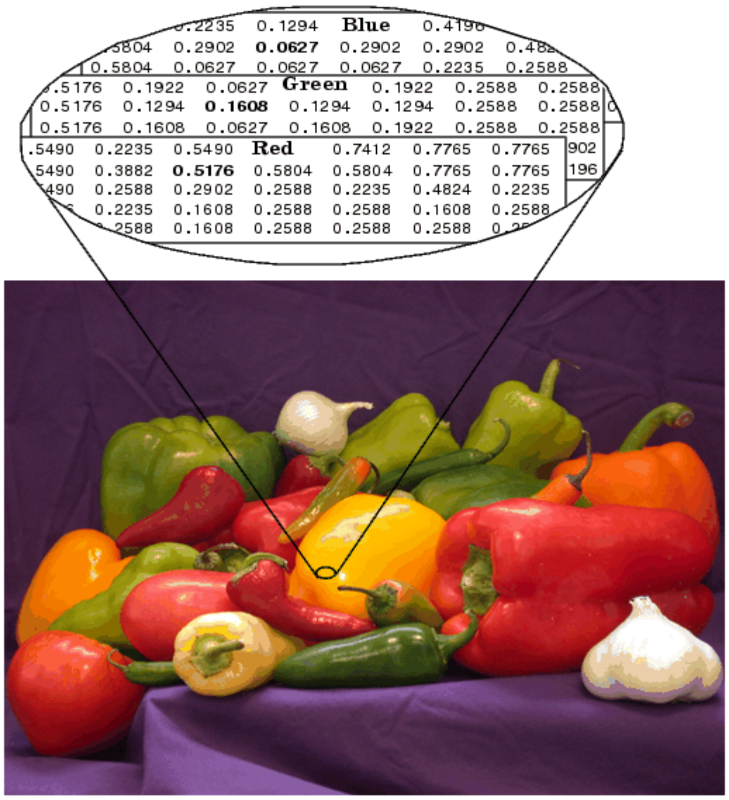


Figure 2.10 Colour Planes of True Colour Image

CHAPTER-3

PROPOSED METHODOLOGY

MRI image of the brain is processed for the detection of the tumour using MATLAB. The proposed methodology employed here comprises of three stages. Initially pre-processing of given MRI image is done then edge detection of brain is conducted and finally, segmentation displays the tumour region vividly. K-means clustering algorithm has also been implemented as an alternative method of segmentation. K-means clustering displays other important tissues and edges along with the tumour region.

3.1 Block Diagram:

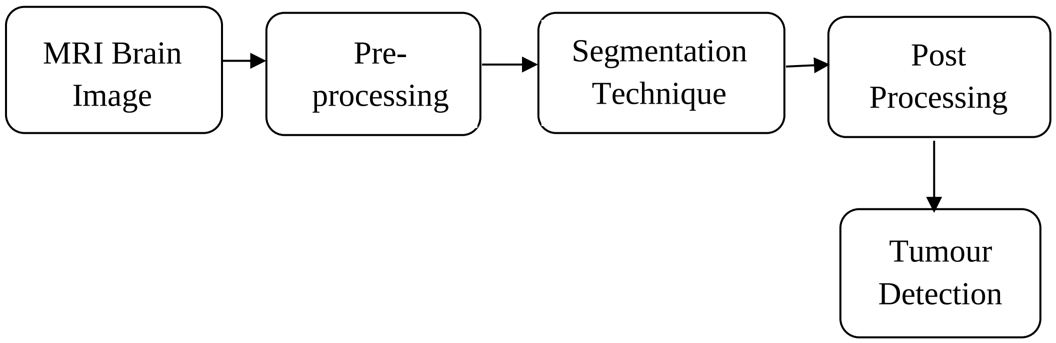


Figure 3.1 Block Diagram of Brain Tumour Detection

The steps of the algorithm are discussed as follows.

Proposed Algorithm for Detection of Brain Tumour

Step 1: Take MRI image of the brain as an input.

Step 2: Convert it into equivalent grayscale image.

Step 3: Apply filtering methods for removing noise.

Step 4: Apply image enhancement techniques.

Step 5: Implement segmentation technique and clustering algorithm for proper detection of tumour region.

The above-mentioned steps are explained below in detail.

3.2 Pre-processing Stage:

Image pre-processing aims in noise removal and to improve the clarity of image or altering the quality of image to suit a purpose. The functions performed at the pre-processing stage are described as follows.

- RGB to Grayscale Conversion: As the name indicates, the image may consist of shades of grey. A ‘gray’ colour is one in which the red, green and blue elements have similar intensity in RGB space. A grayscale image contains the grayscale values but some MRI images consist of primary (RGB) content. These images need to be converted into grayscale image which range from 0 to 255pixel values where range 0 defines the pure black colour and range 255 defines pure white colour.
- Noise Removal using Median Filtering: Filtering is a technique used for eliminating the noise present within an image. During the conversion of an image from RGB to gray some sort of noise creeps into the image. Thus, this noise needs to get removed using filtering. It is applied to eradicate the noises such as salt and pepper from the converted grayscale image. It exchanges the value of the pixel in the centre with the median of the intensity values in the neighbouring pixels.
- Image Enhancement: Acquired image may have defects such as poor contrast. These defects have huge impact on the contrast of an image. It is the process of adjusting digital images so that the results are more suitable for display or further image analysis. When contrast is poor, the contrast enhancement technique comes into play. In this case, the gray level of each pixel is scaled for improving the contrast. The visualization of the MRI image is improved through contrast enhancement technique.

3.3 Clustering

Clustering is a process of representing the image into various divisions for easier analysis and detailed study of the meaningful portions of the image. The image is divided in such a manner that each segment of the image shares certain similar characteristics such as intensity, texture or colour. The collected set of segmented image builds up the entire image. In the proposed method, the k-means algorithm is implemented for accurate prediction of the brain tumour regions.

Steps involved in the algorithm are discussed below in details.

Step 1: Set k different points where k indicates total number of regions of the image.

Step 2: Assign each pixel to the kth point that has the closest centroid as per their Euclidean distance.

Step 3: When each pixel of the image is assigned to a cluster then the k’s position is recalculated.

Step 4: Repeat steps 2 and 3 until the centroid “k” does not change.

Step 5: Display each divided cluster separately to view k number of clusters

K-means clustering is the most widely used and studied method among the clustering formulations that are based on minimizing a formal objective function. Modifications to the K-means clustering method that makes it faster and more efficient are proposed. K-means clustering is a key technique in pixel-based methods. In which pixel-based methods based on K-means clustering are simple and the computational complexity is relatively low compared with other region-based or edge-based methods, the application is more practicable. The main argument of the proposed modifications is on the reduction of intensive distance computation that takes place at each run(iteration) of K-means algorithm between each data point and all cluster centres. To reduce the intensive distance computation, a simple mechanism by which, at each iteration, the distance between each data point and the cluster nearest to it computed and recorded in a data structure is suggested. Thus, on the following iterations the distance between each data point and its previous nearest cluster is recomputed in the proposed method, we combine segmentation and the K-means clustering. A brain Image consists of four regions i.e. gray matter (GM), white matter (WM), cerebra spinal fluid (CSF) and background. Therefore, an input image needs to be divided into these four classes. In order to avoid the chances of misclassification, the outer elliptical shaped object should be removed. After the enhancement of image morphological process is carried out to extract the required region. The Next step is by implementing K-means with clusters exact result is produced.

3.4 Flowchart:

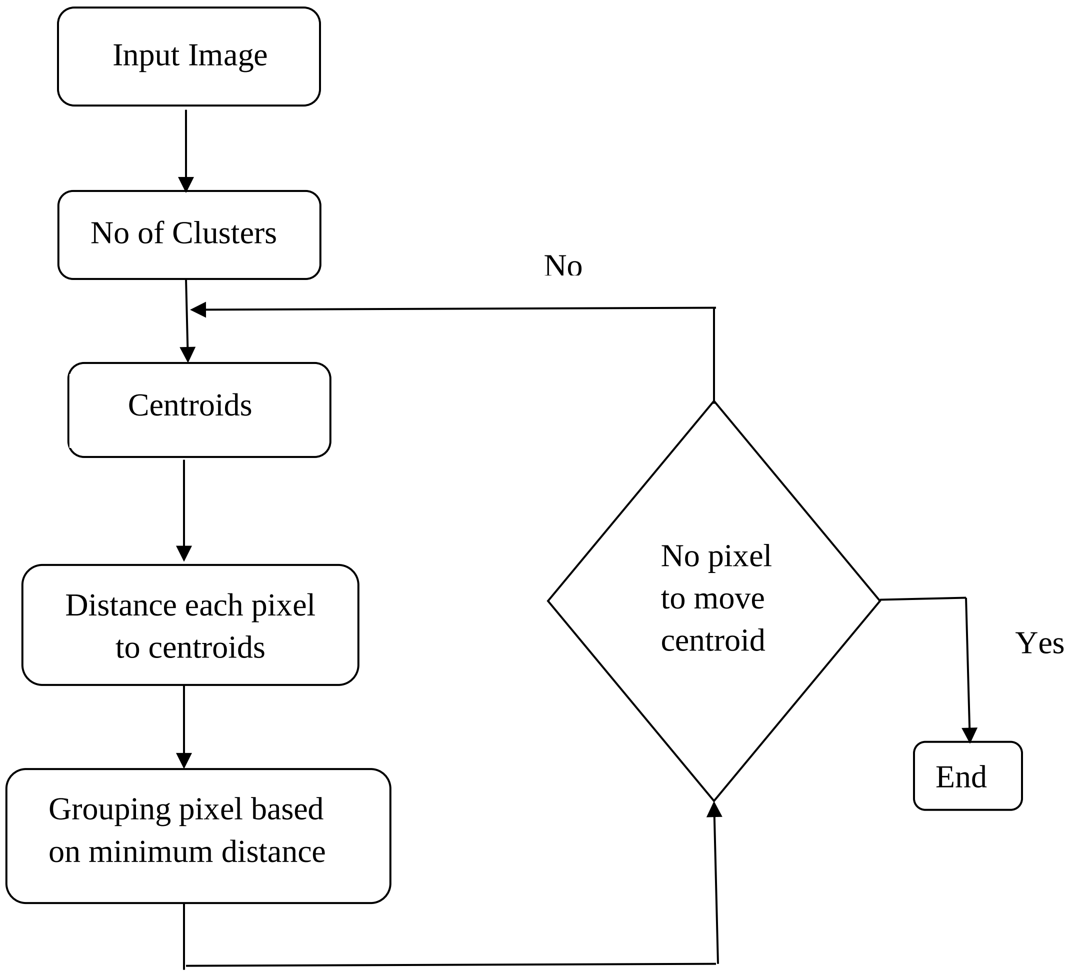

Input Image
No of Clusters
No
Centroids
Distance each pixel to centroids
Grouping pixel based on minimum distance
No pixel to move centroid
Yes
End

Figure 3.2 K-Means Clustering Flow Chart

In above figure we are showing K-means segmentation algorithm work flow. The first block shows the input image then it will convert that image into numbers of clusters, then it will find out the cluster centroid and find out the distance of each pixel from the centroid, then it will continue to grouping of pixels till it will reached the last pixel and stop.

Then k-cluster center are chosen randomly. The distance between each pixel to each cluster centers are calculated. The distance may be of simple Euclidean function. Single pixel is compared to all cluster centers using the distance formula. The pixel is moved to particular cluster which has shortest distance among all. Then the centroid is re-estimated.

Again, each pixel is compared to all centroids. The process continuous until the center converges.

$$W(C) = \frac{1}{2}\sum_{k=1}^{\kappa}\sum_{C(i)=k}\sum_{C(j)=x}\left|x_i - x_j\right|^2 \qquad \rightarrow \text{Eq (2)}$$

For a given cluster assignment *C* of the data points, compute the cluster means for a current set of cluster means, assign each observation.

3.5 Morphological Operations

The term morphology refers to the description of the properties of shape and structure of any objects. In the context of computer vision, this term refers to the description of the properties of shapes of areas on the image. Operations of mathematical morphology were originally defined as operations on sets, but it soon became clear that they are also useful in the processing tasks of the set of points in the two-dimensional space. Sets in mathematical morphology represent objects in the image. It is easy to see that the set of all background pixels of binary image is one of the options for a full description. In the first-place mathematical morphology is used to extract some properties of the image, useful for its presentation and descriptions. For example, contours, skeletons and convex hulls. Also, morphological methods are used in the preliminary and final image processing. For example, morphological filtering, thickening or thinning. The input data for the mathematical morphology are the two images: processed and special, depending on the type of operations and solve problems. Such a special image called primitive or structural element. Typically, a structural element is much smaller than the processed image. Structural element can be regarded as a description of the area with some form. It is clear that the shape can be arbitrary, as long as it can be represented as a binary image of a given size.

The structuring element is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one:

- The matrix dimensions specify the size of the structuring element.
- The pattern of ones and zeros specifies the shape of the structuring element.
- An origin of the structuring element is usually one of its pixels, although generally the origin can be outside the structuring element.

The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion. This table lists the rules for both dilation and erosion.

3.5.1 Dilation:

Dilation operation consists of convoluting an image with some kernel, which can have any shape or size, usually a square or circle. The kernel has a defined anchor point, usually being the center of the kernel. As the kernel is scanned over the image, we compute the maximal pixel value overlapped by and replace the image pixel in the anchor point position with that maximal value. As you can deduce, this maximizing operation causes bright regions within an image to “grow” (therefore the name dilation).

To compute the dilation of a binary input image by this structuring element, we consider each of the background pixels in the input image in turn. For each background pixel (which we will call the input pixel) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position. If at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value. If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.

For our example 3×3 structuring element, the effect of this operation is to set to the foreground color any background pixels that have a neighboring foreground pixel. Such pixels must lie at the edges of white regions, and so the practical upshot is that foreground regions grow (and holes inside a region shrink).

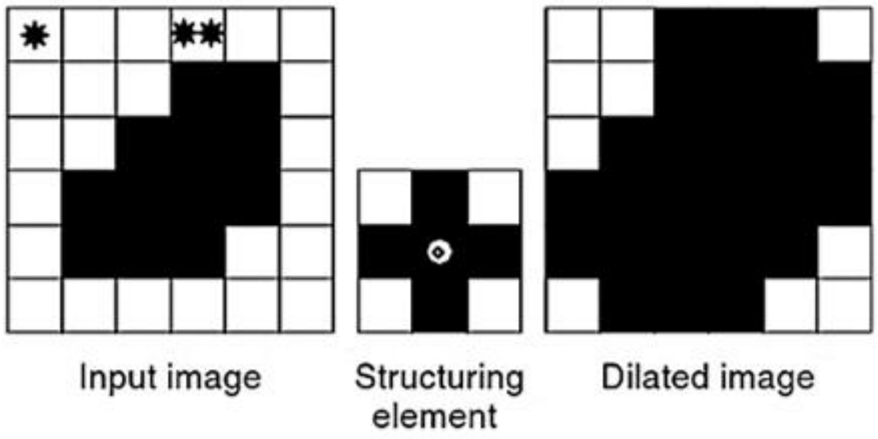


Figure 3.3 Morphological Dilation Process

3.5.2 Erosion:

Erodes away the boundaries of foreground object used to diminish the features of an image. A kernel (a matrix of odd size (3,5,7) is convolved with the image. A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). Thus, all the pixels near boundary will be discarded depending upon the size of kernel. So, the thickness or size of the foreground object decreases or simply white region decreases in the image.

To compute the erosion of a binary input image by this structuring element, we consider each of the foreground pixels in the input image in turn. For each foreground pixel we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel coordinates. If for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.

For our example 3×3 structuring element, the effect of this operation is to remove any foreground pixel that is not completely surrounded by other white pixels. Such pixels must lie at the edges of white regions, and so the practical upshot is that foreground regions shrink.

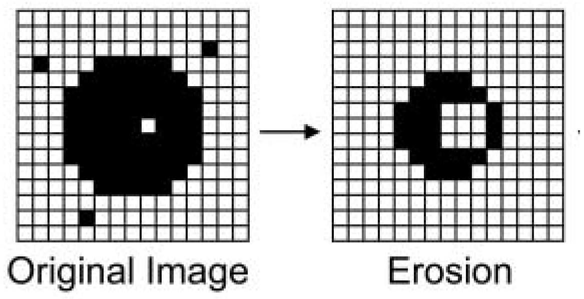


Figure 3.4 Morphological Erosion Process

3.5.3 Opening:

Opening is defined as an erosion followed by a dilation. Figure shows the opposite operation of closing, defined as a dilation followed by an erosion. As illustrated by these examples, opening removes small islands and thin filaments of object pixels. Likewise, closing removes islands and thin filaments of background pixels. These techniques are useful for handling noisy images where some pixels have the wrong binary value. For instance, it might be known that an object cannot contain a "hole", or that the object's border must be smooth. While erosion can be used to eliminate small clumps of undesirable foreground pixels, *e.g.* Salt noise, quite effectively, it has the big disadvantage that it will affect all regions of foreground pixels indiscriminately.

Opening gets around this by performing both an erosion and a dilation on the image. The effect of opening can be quite easily visualized. Imagine taking the structuring element and sliding it around inside each foreground region, without changing its orientation. All pixels which can be covered by the structuring element with the structuring element being entirely within the foreground region will be preserved. However, all foreground pixels which cannot be reached by the structuring element without parts of it moving out of the foreground region will be eroded away. After the opening has been carried out, the new boundaries of foreground regions will all be such that the structuring element fits inside them, and so further openings with the same element have no effect.

Opening is process where the image is eroded and then dilated.

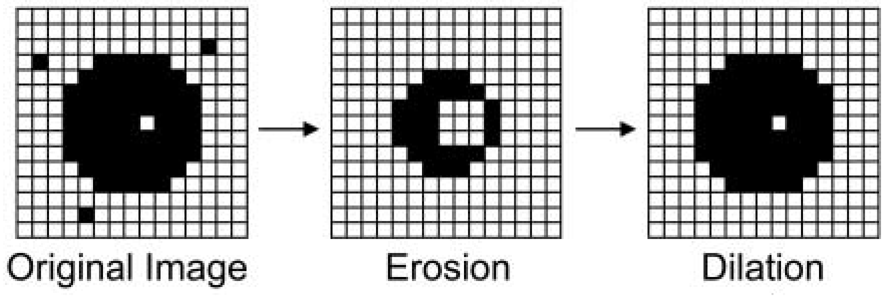


Figure 3.5 Opening Process

3.5.4 Closing:

Closing is reverse of Opening, dilation followed by erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object. Closing is opening performed in reverse. It is defined simply as a dilation followed by an erosion using the same structuring element for both operations. See the sections on erosion and dilation for details of the individual steps. The closing operator therefore requires two inputs: an image to be closed and a structuring element. Gray level closing consists straightforwardly of a gray level dilation followed by a gray level erosion.

Closing is closing the foreground pixels with a particular structuring element, is equivalent to closing the background with the same element. One of the uses of dilation is to fill in small background color holes in images, *e.g.* Pepper Noise. One of the problems with doing this, however, is that the dilation will also distort all regions of pixels indiscriminately. By performing an erosion on the image after the dilation, *i.e.* a closing, we reduce some of this effect. The effect of closing can be quite easily visualized. Imagine taking the structuring element and sliding it around outside each foreground region, without changing its orientation. For any background boundary point, if the structuring element can be made to touch that point, without any part of the element being inside a foreground region, then that point remains background. If this is not possible, then the pixel is set to foreground. After the closing has been carried out the background region will be such that the structuring element can be made to cover any point in the background without any part of it also covering a foreground point, and so further closings will have no effect.

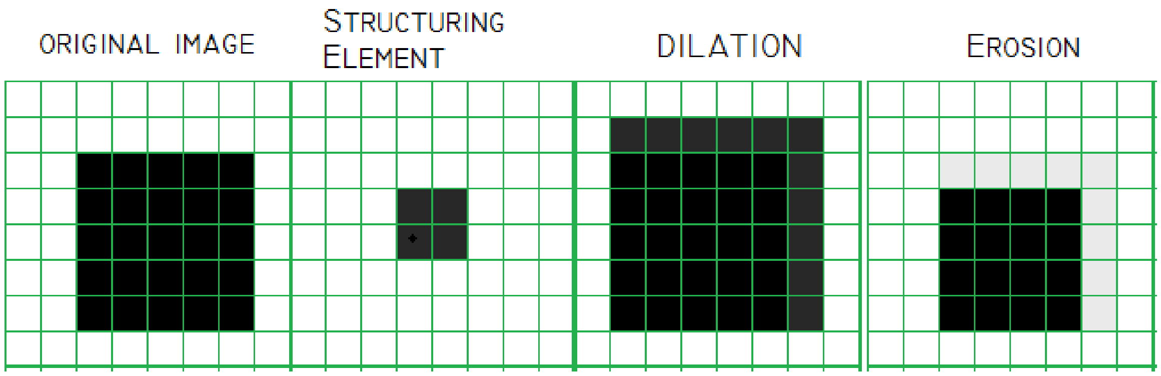


Figure 3.6 Closing Process

CHAPTER-4

MATLAB IMPLEMENTATION

4.1 Implementation

4.1.1 Basics of MATLAB:

MATLAB is a high-performance language for technical computing It integrates computation visualization and programming in an easy to use environment.

MATLAB stands for matrix laboratory. It was written originally to provide easy access to matrix software developed by LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is therefore built on a foundation of sophisticated matrix software which the basic element is matrix that does not require pre dimensioning

4.1.2 Typical Uses of MATLAB:

- Math and computation
- Algorithm development
- Data acquisition
- Data analysis exploration and visualization
- Scientific and engineering graphics

4.1.3 The Main Features of MATLAB:

- Advance algorithm for high performance numerical computation especially in the Field matrix algebra.
- A large collection of predefined mathematical functions and the ability to define one's own functions
- Two-and three-dimensional graphics for plotting and displaying data.
- A complete online help system.

- Powerful, matrix or vector oriented high-level programming language for individual applications
- Toolboxes available for solving advanced problems in several application areas

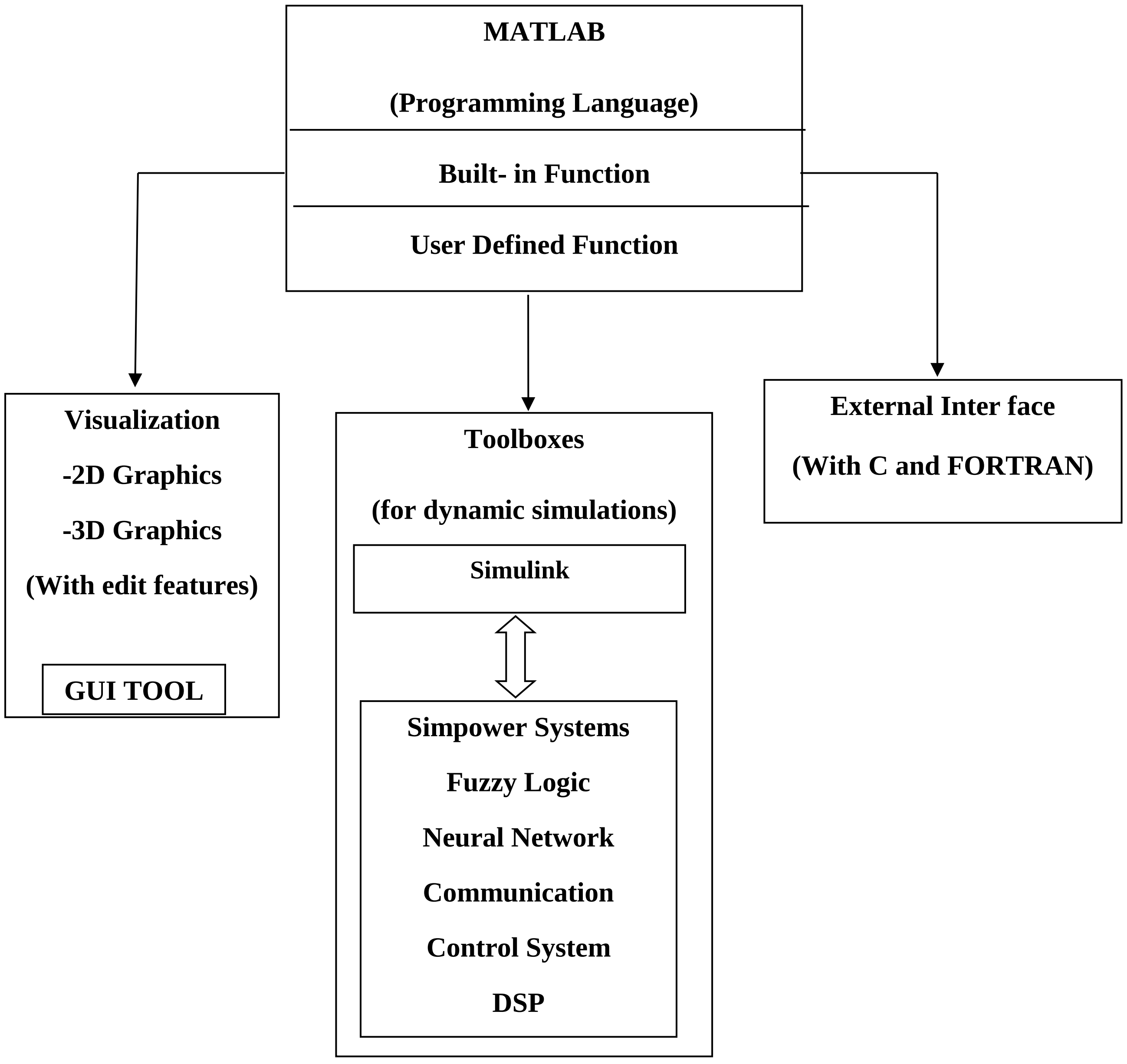


Figure 4.1 Features and capabilities of the MATLAB System

4.1.4 Development Environment:

This is the set of tools and facilities that help we use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

4.1.5 The MATLAB Mathematical Function Library

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms. Like libraries, model references allow you to define a set of blocks once and use it repeatedly. Model references provide several advantages that are unavailable with subsystems and libraries. Several of these advantages result from referenced models compiling independent of the context of the Model block, including:

4.2 The MATLAB Language:

This is a high-level matrix array language with control flow statements functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small to rapidly create quick and dirty throw-away programs, and "programming in the large to create large and complex application programs.

4.2.1 Graphics:

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow we to fully customize the appearance of graphics as well as to build complete graphical user interfaces on our MATLAB applications.

4.2.2 The MATLAB Application Program Interface:

This is a library that allows we to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files. Use Matrix API and MEX API functions in gateway and computational routines to interact with data in the MATLAB® workspace. These APIs are part of the MATLAB C/C++ and Fortran API Reference library.

To use these functions, include the mex header, which declares the entry point and interface routines. Put this statement in your source file:

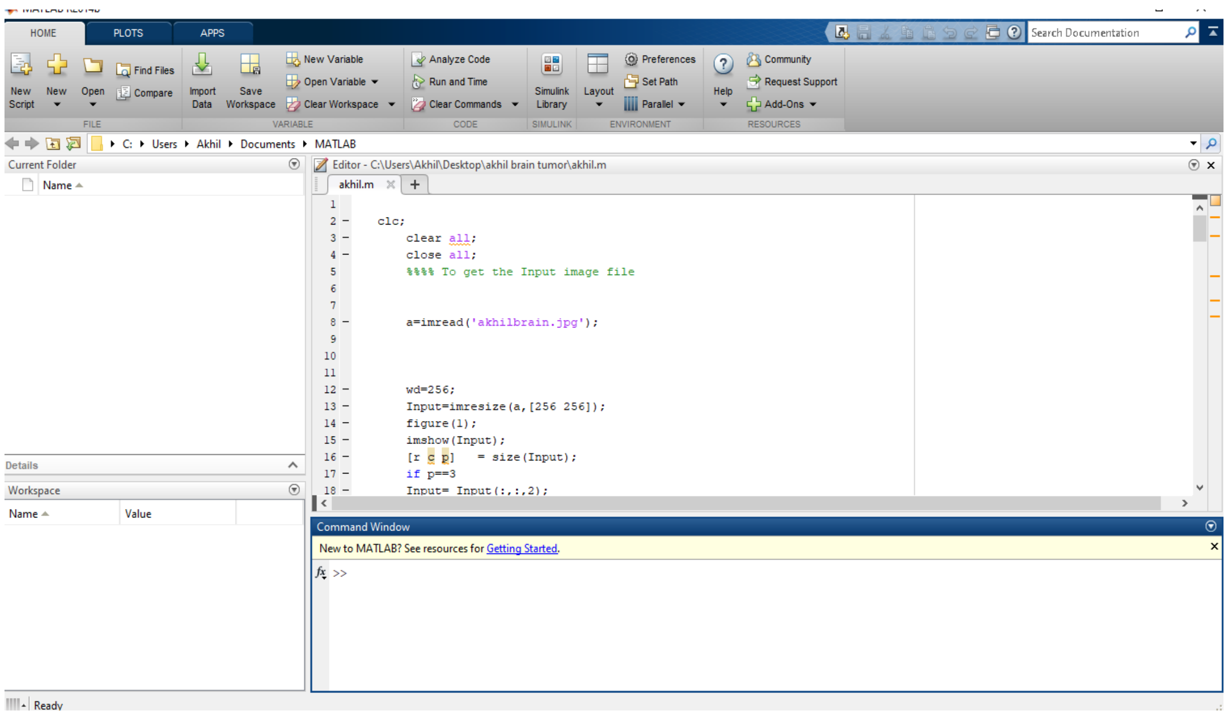
```
#include "mex.h"
```

4.2.3 Starting MATLAB:

On Windows platforms, start MATLAB by double-clicking the MATLAB shortcut icon on our Windows desktop. On UNIX platforms, start MATLAB by typing MATLAB at the operating system prompt. We can customize MATLAB startup. For example, we can change the directory in which MATLAB starts or automatically execute MATLAB statements in a script file named startup.

4.3 MATLAB Desktop:

When we start MATLAB, the MATLAB desktop appears, containing tools graphical user interfaces for managing files, variables, and applications associated with MATLAB. The following illustration shows the default desktop. We can customize the arrangement of tools and documents to suit our needs. The MATLAB desktop environment helps you run commands, manage files, and view results. You can change the desktop layout and set preferences, such as fonts, keyboard shortcuts, and initial working folder.



Editor - C:\Users\Akhil\Desktop\akhil brain tumor\akhil.m

```
clc;
    clear all;
    close all;
    %%%% To get the Input image file


    a=imread('akhilbrain.jpg');



    wd=256;
    Input=imresize(a,[256 256]);
    figure(1);
    imshow(Input);
    [r c p]    = size(Input);
    if p==3
    Input= Input(:,:,2);
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>>
```

Figure 4.2 MATLAB Window

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or FORTRAN.

The name MATLAB stands for ***matrix laboratory.***

MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as computational engine, and for reading and writing MAT-files.

4.4 Using MATLAB Editor to Create M-Files

MATLAB editor is both a text editor specialized for creating M-files and a graphical MATLAB debugger. The editor can appear in a window by itself, or it can be a sub window in the desktop. M-files are denoted by the extension .m, as in pixelup.m. The MATLAB editor window has numerous pull-down menus for tasks such as saving, viewing, and debugging files. Because it performs some simple checks and also uses colour to differentiate between various elements of code, this text editor is recommended as the tool of choice for writing and editing M-functions.

To open the editor, type edit at the prompt opens the M-file filename min an editor window, ready for editing. As noted earlier, the file must be in the current directory, or in a directory in the search path.

4.5 Getting Help

The principal way to get help online is to use the MATLAB help browser, opened as a separate window either by clicking on the question mark symbol (?) on the desktop toolbar, or by typing help browser at the prompt in the command window. The help Browser is a web browser integrated into the MATLAB desktop that displays a Hypertext Mark-up Language (HTML) documents. The Help Browser consists of two panes, the help navigator pane, used to find information, and the display pane, used to view the information. Self-explanatory tabs other than navigator pane are used to perform a search.

For example, help on a specific function is obtained by selecting the search tab, selecting Function Name as the Search Type, and then typing in the function name in the search for field. It is good practice to open the Help Browser at the beginning of a MATLAB session to have helped readily available during code development or another MATLAB task.

Another way to obtain for a specific function is by typing doc followed by the function name at the command prompt. For example, typing doc format displays documentation for the function called format in the display pane of the Help Browser. This command opens the browser if it is not already open.

M-functions have two types of information that can be displayed by the user. The first is called the H1 line, which contains the function name and alone line description. The second is a block of explanation called the Help text block. Typing help at the prompt followed by a function name displays both the H1 line and the Help text for that function in their command window. Occasionally, this information can be more up to date than the documentation of the M-function in question. Typically look for followed by a keyword displays all the H1 lines that contain that keyword. This function is useful when looking for a particular topic without knowing the names of applicable functions.

For example, typing look for edge at the prompt displays the H1 lines containing that keyword. Because the H1 line contains the function name, it then becomes possible to look at specific functions using the other help methods. Typing look for edge-all at the prompt displays theH1 line of all functions that contain the word edge in either the H1 line or the Help text block. Words that contain the characters edge also are detected. For example, the H1 line of a function containing the word poly-edge in the H1 line or Help text would also be displayed.

4.6 Saving and Retrieving A Work Session

There are several ways to save and load an entire work session or selected workspace variables in MATLAB. The simplest is as follows.

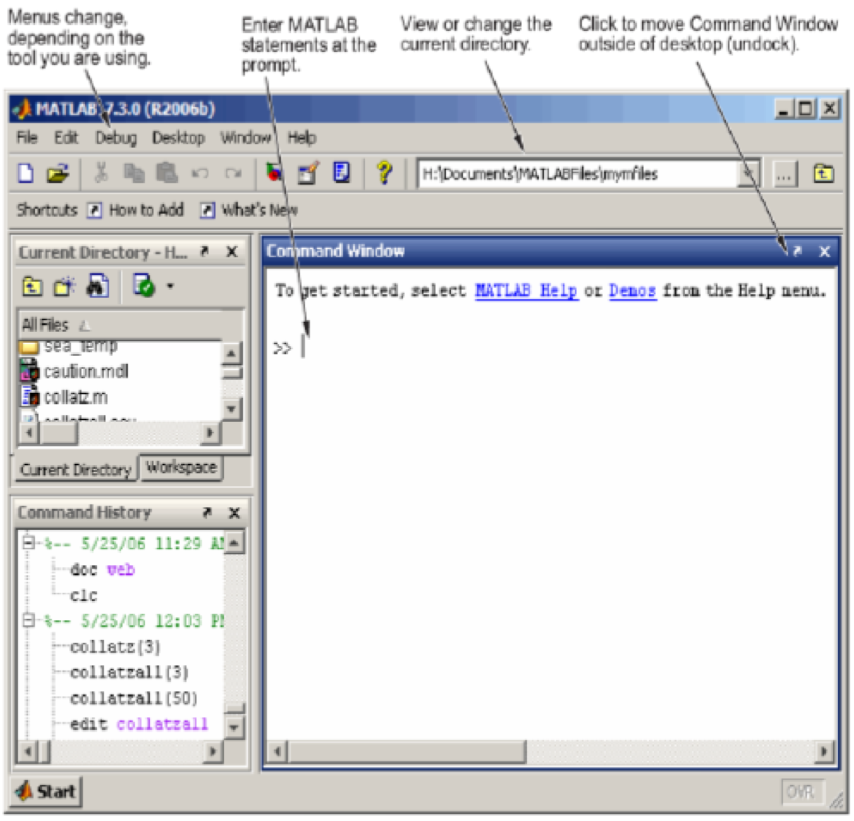
To save the entire workspace, simply right-click on any blank space in the workspace Browser window and select Save Workspace As from the menu that appears. This opens a directory window that allows naming the file and selecting any folder in the system in which to save it. Then simply click Save. To save a selected variable from the workspace, select the variable with a left click and then right-click on the highlighted area.

Then select save selection as from the menu that appears. This again opens a window from which a folder can be selected to save the variable.

To select multiple variables, use shift click or control click in the familiar manner, and then use the procedure just described for a single variable. All files are saved in the double-precision, binary format with the extension. Mat. These saved files commonly are referred to as MAT-files.

To load saved workspaces or variables, left click on the folder icon on the toolbar of the workspace browser window. This causes a window to open from which a folder containing MAT-file or selecting open causes the contents of the file to be restored in the workspace Browser window.

It is possible to achieve the same results described in the preceding paragraphs by typing save and load at the prompt, with the appropriate file names and path information.


Menus change, depending on the tool you are using.
Enter MATLAB statements at the prompt.
View or change the current directory.
Click to move Command Window outside of desktop (undock).
MATLAB 7.3.0 (R2006b)
File Edit Debug Desktop Window Help
H:\Documents\MATLABFiles\mymfiles
Shortcuts How to Add What's New
Current Directory - H...
All Files
sea_temp
caution.mdl
collatz.m
Current Directory
Workspace
Command History
Command Window
Start
OVR

```
To get started, select MATLAB Help or Demos from the Help menu.

>>
```

```
%-- 5/25/06 11:29 AM
doc web
clc
%-- 5/25/06 12:03 PM
collatz(3)
collatzall(3)
collatzall(50)
edit collatzall
```

Figure 4.3 MATLAB command window

4.7 Plotting Tools

Plotting tools are attached to figures and create an environment for creating Graphs. These tools enable you to do the following:

- Select from a wide variety of graph types
- Change the type of graph that represents a variable
- See and set the properties of graphics objects
- Annotate graphs with text, arrows, etc.
- Create and arrange subplots in the figure
- Drag and drop data into graphs

Display the plotting tools from the **View** menu or by clicking the plotting tools icon in the figure toolbar, as shown in the following picture

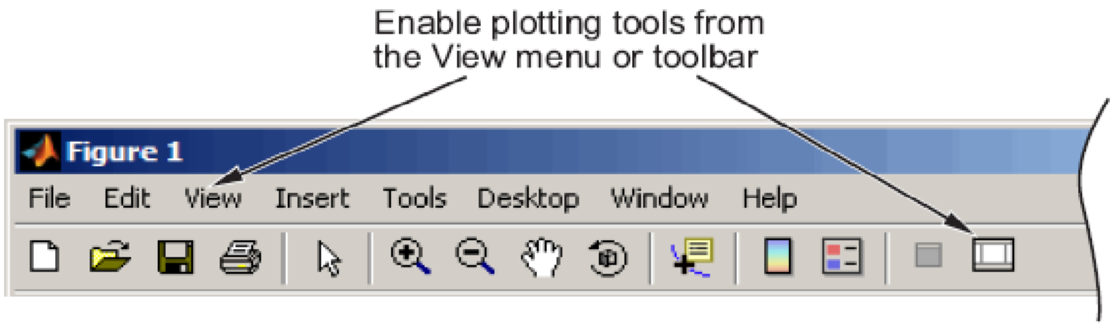
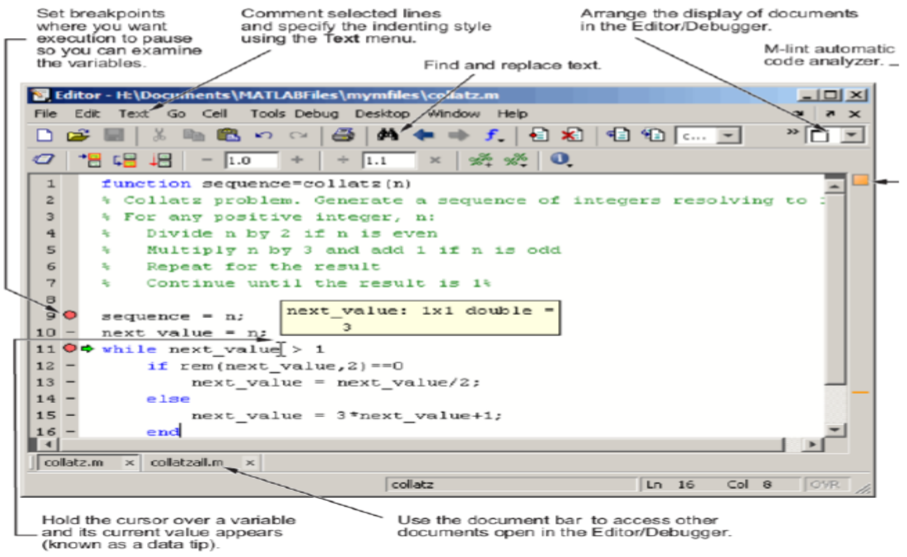


Figure 4.4 Plotting window

4.8 Editor/Debugger

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for text editing, as well as for M-file debugging. To create or edit an M-file use **File > New** or **File >Open**, or use the edit function.


Set breakpoints where you want execution to pause so you can examine the variables.
Comment selected lines and specify the indenting style using the Text menu.
Find and replace text.
Arrange the display of documents in the Editor/Debugger.
M-lint automatic code analyzer.
Editor - H:\Documents\MATLABFiles\mymfiles\collatz.m
File Edit Text Go Cell Tools Debug Desktop Window Help
next_value: 1x1 double = 3
collatz.m
collatzall.m
collatz
Ln 16 Col 8 OVR
Hold the cursor over a variable and its current value appears (known as a data tip).
Use the document bar to access other documents open in the Editor/Debugger.

```
function sequence=collatz(n)
% Collatz problem. Generate a sequence of integers resolving to
% For any positive integer, n:
%   Divide n by 2 if n is even
%   Multiply n by 3 and add 1 if n is odd
%   Repeat for the result
%   Continue until the result is 1%

sequence = n;
next_value = n;
while next_value > 1
    if rem(next_value,2)==0
        next_value = next_value/2;
    else
        next_value = 3*next_value+1;
    end
```

Figure 4.5 Details of Editor Window

MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and FORTRAN.

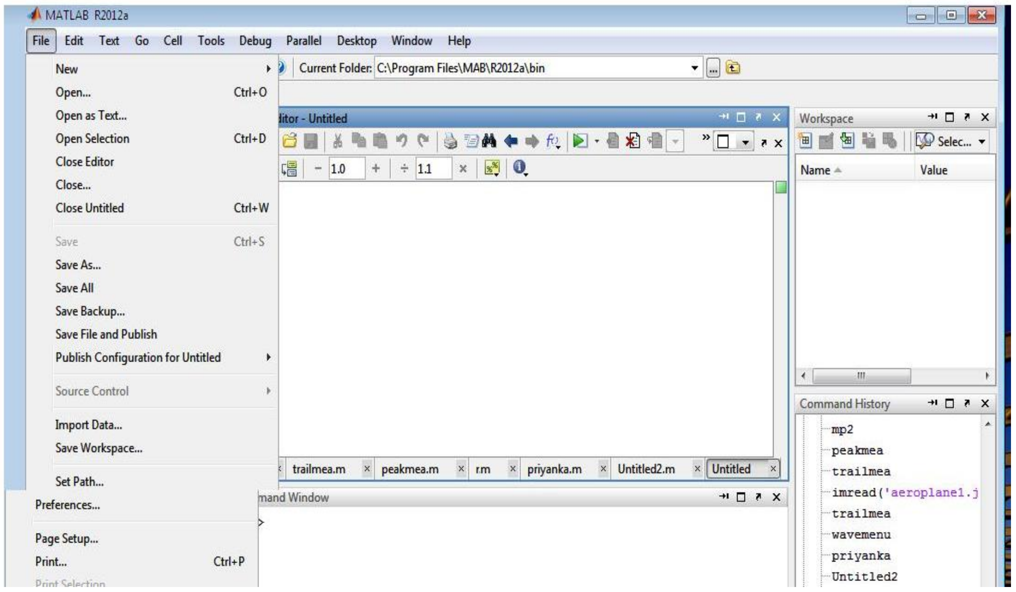
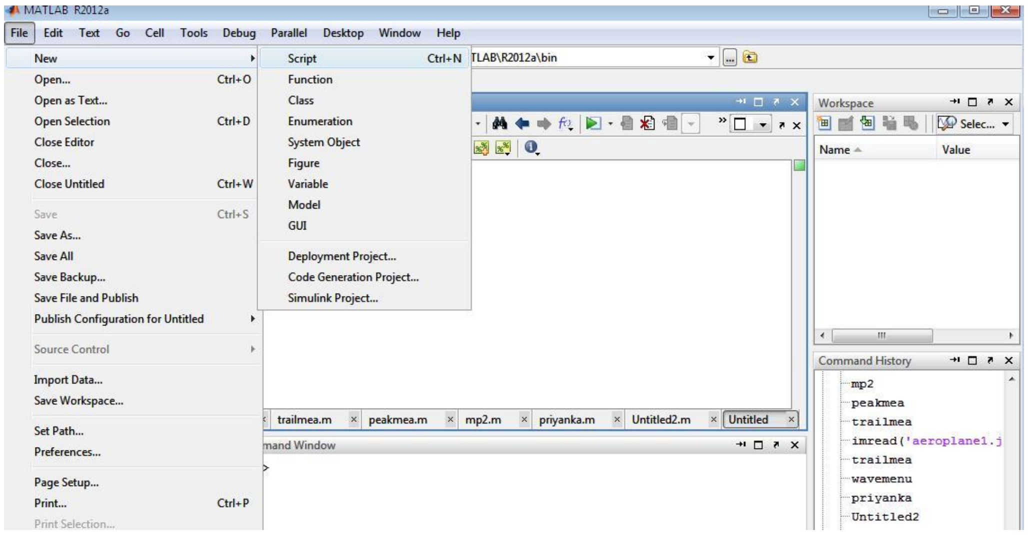


Figure 4.6 New file

- Open the new script



```
mp2
peakmea
trailmea
imread('aeroplane1.j
trailmea
wavemenu
priyanka
Untitled2
```

Figure 4.7 Opening New script

- New script

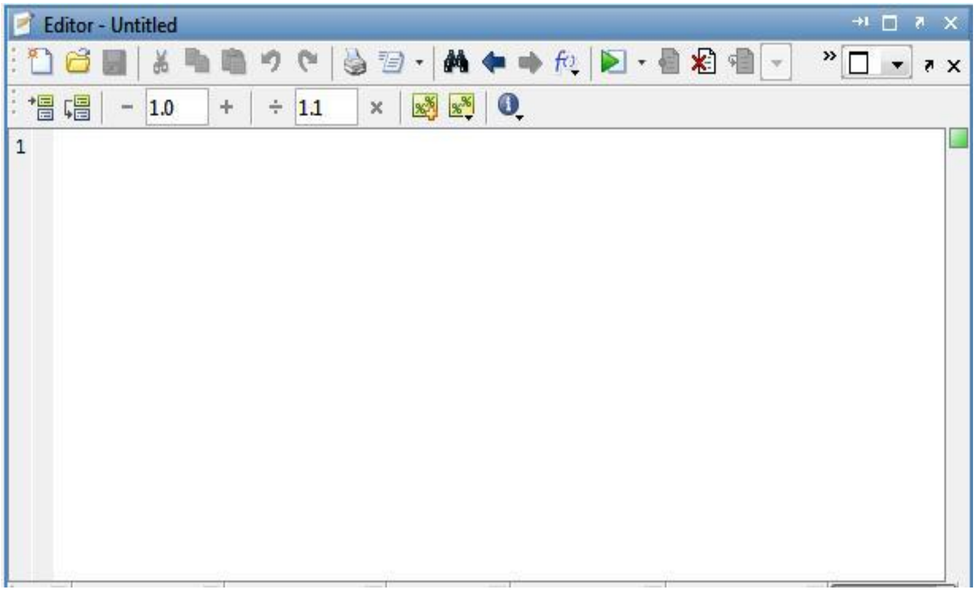
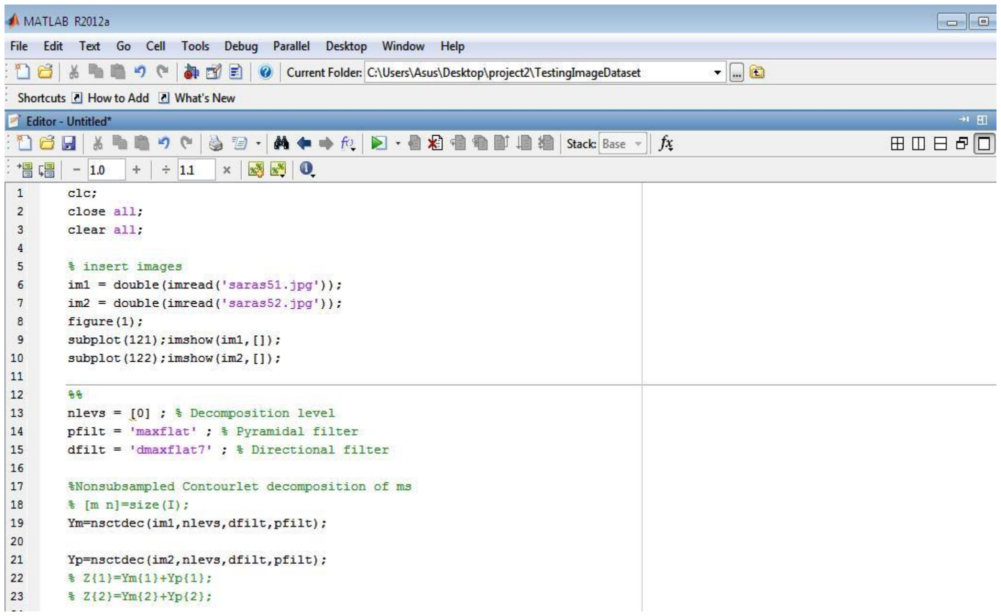


Figure 4.8 New Script

- Write the code

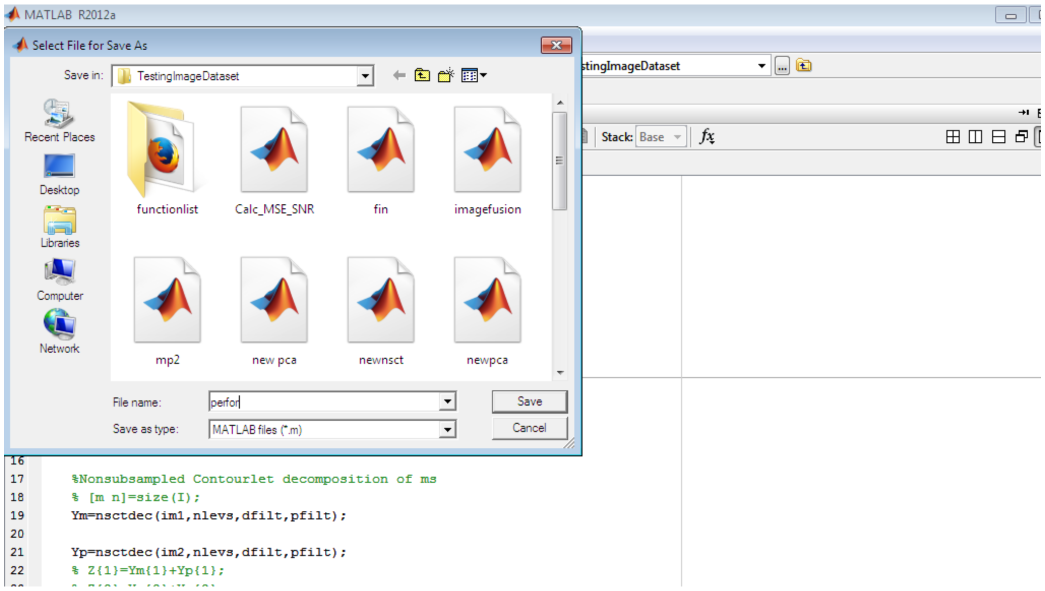

MATLAB R2012a
File Edit Text Go Cell Tools Debug Parallel Desktop Window Help
Shortcuts How to Add What's New
Editor - Untitled*
Stack: Base

Current Folder: C:\Users\Asus\Desktop\project2\TestingImageDataset

```
clc;
close all;
clear all;

% insert images
im1 = double(imread('saras51.jpg'));
im2 = double(imread('saras52.jpg'));
figure(1);
subplot(121);imshow(im1,[]);
subplot(122);imshow(im2,[]);

%%
nlevs = [0] ; % Decomposition level
pfilt = 'maxflat' ; % Pyramidal filter
dfilt = 'dmaxflat7' ; % Directional filter

%Nonsubsampled Contourlet decomposition of ms
% [m n]=size(I);
Ym=nsctdec(im1,nlevs,dfilt,pfilt);

Yp=nsctdec(im2,nlevs,dfilt,pfilt);
% Z{1}=Ym{1}+Yp{1};
% Z{2}=Ym{2}+Yp{2};
```

Figure 4.9 Editor Window

- Save the code



Save in: TestingImageDataset

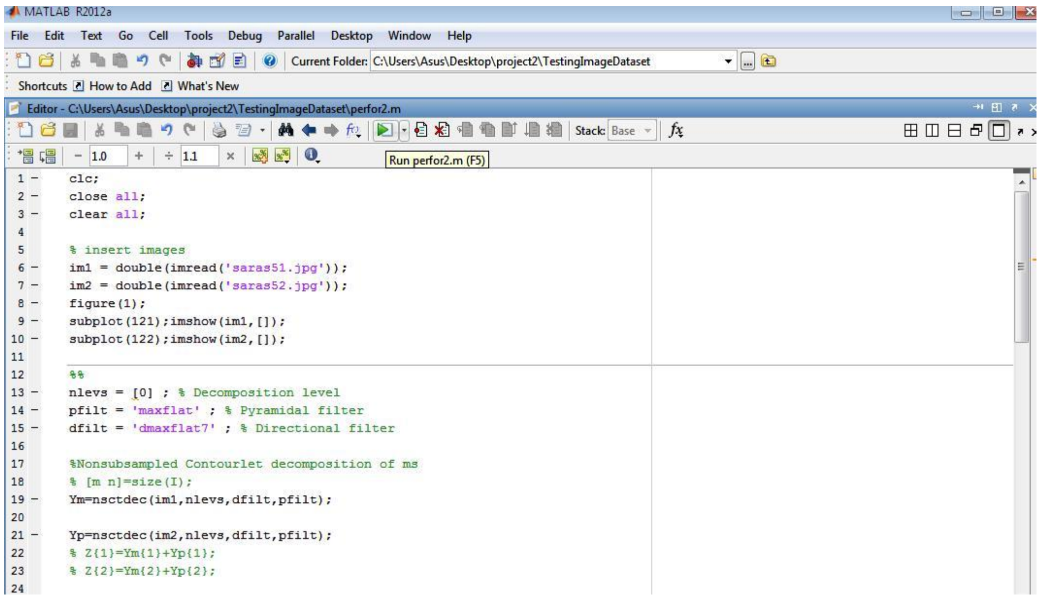
File name: perfor

Save as type: MATLAB files (*.m)

```
%Nonsubsampled Contourlet decomposition of ms
% [m n]=size(I);
Ym=nsctdec(im1,nlevs,dfilt,pfilt);

Yp=nsctdec(im2,nlevs,dfilt,pfilt);
% Z{1}=Ym{1}+Yp{1};
```

Figure 4.10 Saving the MATLAB File

- Run the code


MATLAB R2012a
File Edit Text Go Cell Tools Debug Parallel Desktop Window Help
Shortcuts How to Add What's New
Stack: Base
Run perfor2.m (F5)

Current Folder: C:\Users\Asus\Desktop\project2\TestingImageDataset

Editor - C:\Users\Asus\Desktop\project2\TestingImageDataset\perfor2.m

```
clc;
close all;
clear all;

% insert images
im1 = double(imread('saras51.jpg'));
im2 = double(imread('saras52.jpg'));
figure(1);
subplot(121);imshow(im1,[]);
subplot(122);imshow(im2,[]);

%%
nlevs = [0] ; % Decomposition level
pfilt = 'maxflat' ; % Pyramidal filter
dfilt = 'dmaxflat7' ; % Directional filter

%Nonsubsampled Contourlet decomposition of ms
% [m n]=size(I);
Ym=nsctdec(im1,nlevs,dfilt,pfilt);

Yp=nsctdec(im2,nlevs,dfilt,pfilt);
% Z{1}=Ym{1}+Yp{1};
% Z{2}=Ym{2}+Yp{2};
```

Figure 4.11 Run Code

CHAPTER-5

RESULTS

MRI images are converted into grayscale images. Normally the MRI image is filled with noise. To remove this, we have to increase the contrast of the image by Median Filter. The first step is to input the image and then the image has to be denoised for the next steps.

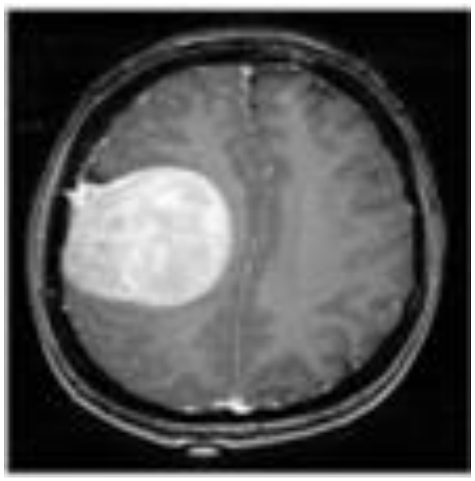


Figure 5.1 Input Image

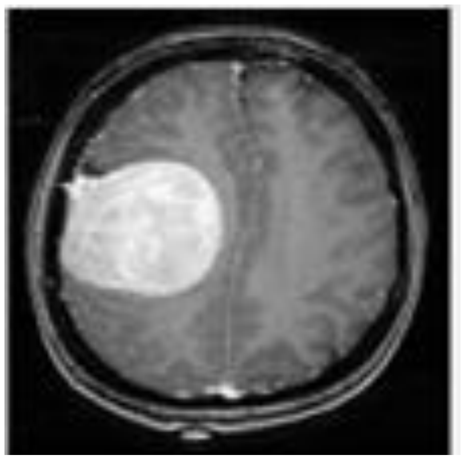


Figure 5.2 Denoised Image

The segmented image properly displays a noise free image which clearly shows the affected region. The segmentation is done by the K-Means Clustering Algorithm. The image is segmented based on the cluster value given (i.e. K=5).

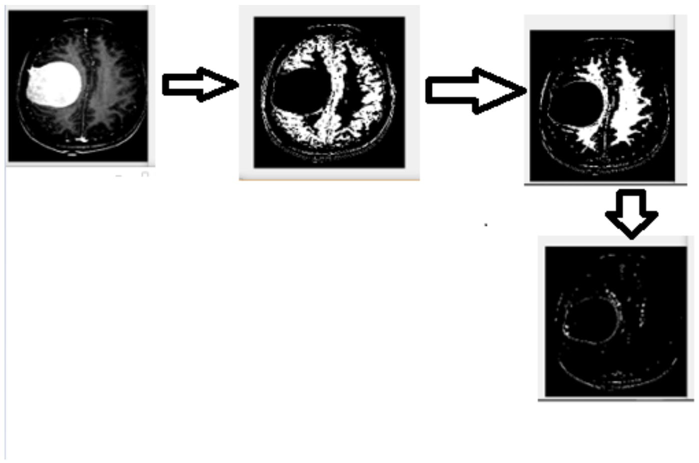


Figure 5.3 Image Processing in K-Means Clustering

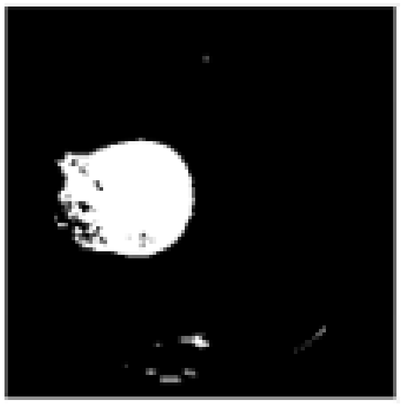
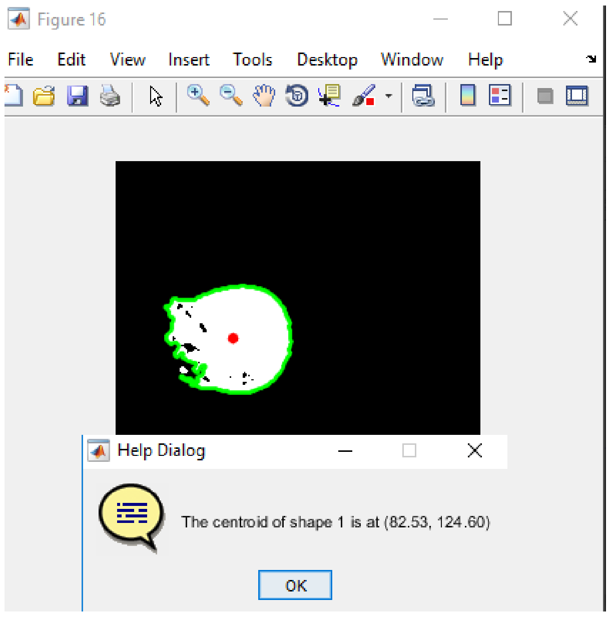


Figure 5.4 K-Means Clustered Output



The centroid of shape 1 is at (82.53, 124.60)

Figure 5.5 Detected Brain Tumour Output

After clustering, the different cluster images are appeared. In those images, the high intensity cluster value is picked for the brain tumour output. Then the image may have some noise after clustering for eliminating the noises we have to go for image Morphological processing. For this image, we chose morphological opening. After the opening process the affected brain tumour output can be outed. For recognising the brain tumour ease, boundary is plotted around the tumour and also the centroid is obtained.

The Brain Tumour Output has its centroid with the x, y plot representation in the picture with respect to the pixels.

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion:

Random selection of seed points does not yield accurate segmentation. Accuracy of the segmentation process is essential to achieve more precise and repeatable radiological diagnostic systems. Accuracy can be improved by incorporating a prior information on vessel anatomy and let high level knowledge guide the segmentation algorithm. K means algorithm is a popular clustering algorithm applied widely, but the standard algorithm which selects k objects randomly from population as initial centroids cannot always give a good and stable clustering.

Experimental results show that selecting centroids by our algorithm can lead to a better clustering. Along with the fast development of database and network, the data scale clustering tasks involved in which becomes more and more large. K-means algorithm is a popular partition algorithm in cluster analysis.

6.2 Future Scope:

Along with the fast development of database and network, the data scale clustering tasks involved in which becomes more and more large. K means algorithm is a popular clustering algorithm applied widely, but the standard algorithm which selects k objects randomly from population as initial centroids cannot always give a good and stable clustering.

On considering for 3D Technology scan images, we can find the Depth of the tumour and exact shape of the tumour. This will make easy for the doctors to estimate the patient’s stage and helps to determine the steps to cure it.

REFERENCES

[1] A.R.Kavitha, Dr.C.Chellamuthu, Ms.Kavin Rupa, “An Efficient Approach for Brain Tumour Detection Based on Modified Region Growing and Network in MRI Images,”IEEE, 2012.

[2] Wen-Liange, De-Hua Chen, Mii-shen Yang, “Suppressed fuzzy-soft learning vector quantization for MRI segmentation” Elsevier ltd, 2011.

[3] Vida Harati, Rasoul Khayati, Abdolreza Farzan, “Fully automated tumor segmentation based on improved fuzzy connectedness algorithm in brain MR images” Elsevier ltd, 2011.

[4] O. K. Firke, and Hemangi S. Phalak, “Brain Tumour Detection using CT Scan Images”, IJESC, Vol. 6, No. 8, pp. 2568-2570, August 2016.

[5] Suneetha Bobbillapati, and A. Jhansi Rani, “Automatic Detection of Brain Tumor through MRI”, International Journal of Scientific and Research Publication, Vol. 3, Issue 11, pp. 1-5, November 2013.

[6] D. Dilip Kumar, S Vandana, K. Sakhti Priya and S. Jeneeth Subhashini, “Brain Tumor Image Segmentation using MATLAB”, IJIRST, Vol. 1, Issue 12, pp. 447-451, May 2015.

[7] J. Shi and J. Malik, ―Normalized cuts and image segmentation, <IEEETrans.PatternAnal. Mach.Intell., vol.22, n

[8] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy,” Sonar image segmentation using a hierarchical MRF model” IEEE Trans. Image Process., vol.9, no.7, pp.1216–1231, Jul.2000.

[9] F. Destrempes, J.-F. Angers, and M. Mignotte, “Fusion of hidden Markov random field models and its Bayesian estimation,” IEEE Trans. Image Process., vol.15, no.10, pp.2920–2935, Oct.2006.