CHAPTER 1

INTRODUCTION

Brain tumor is abnormal growth of tissues which grow uncontrollably and unchecked by the check points which control the growth of cells normally. Brain tumor can be either primary or metastatic. Tumor that spread from other parts of body to brain is said to be metastatic. Brain tumor is one of the likely causes of mortality among children and adult. In children brain tumors are the cause of one quarter of all cancer deaths. There are about 200 different types of tumors diagnosed in UK each year. Medical resonance imaging is an advance imaging technique which is used to obtain high quality images of different parts of body.

Medical resonance imaging (MRI) is an advance imaging technique which is used to obtain high quality images of different parts of body. These MRI images are pre-processed so that further morphological operations can be performed on these images for the detection of size, shape and location of tumor. Pre- processing of MRI images is done for noise removal and image enhancement while morphological operations are performed using MATLAB algorithms to separate and detect tumor in brain.

The ultimate goal of segmentation is to extract important features from the image data. The Function of thresholding is to change an intensity image to a segmented image. MATLAB is fast algorithm used for detection of tumor from MRI images in a very short time.

CHAPTER 2

LITERATURE REVIEW

A literature survey on the research and developments in the Medical Image Processing Domain is given in this chapter. A large number of techniques for extracting and segmenting the caudate regions are reviewed and discussed.

[1]. J. C. Bezdek and L. Hall, "Review of MR image segmentation techniques using patternrecognition", Medical physics, vol. 20, no. 4, (1993), pp. 1033-1048.

[2] .W. Deng and W. Xiao, "MRI brain tumor segmentation based on improved fuzzy c-meansmethod", Sixth International Symposium on Multispectral Image Processing and Pattern Recognition, International Society for Optics and Photonics, (2009).

[3]. H. Kekre and T. Sarode, "Detection of tumor in MRI using vector quantization segmentation", International Journal of Engineering Science and Technology, vol 2, no. 8,(2010), pp. 3753-3757.

[4] .T. Logeswari and M. Karnan, "An improved implementation of brain tumor detection using segmentation based on soft computing", Journal of Cancer Research and Experimental Oncology,vol. 2, no. 1, (2009), pp. 006-014.

[5]. M. Mancas and B. Gosselin, "Segmentation using a region-growing thresholding", Electronic Imaging 2005, International Society for Optics and Photonics, (2005).

[6]. S.Murugavalli and V. Rajamani, “An Improved Implementation of Brain Tumor Detection Using Segmentation Based on Neuro Fuzzy Technique”, vol. 1, (2007).

[7].A. Mustaqeem and A. Javed, “An efficient brain tumor detection algorithm using watershed & thresholding based segmentation”, International Journal of Image, Graphics and Signal Processing, vol. 4,no. 10, (2012), p. 34.

[8]. R. C.Patil and A. Bhalchandra, “Brain tumor extraction from MRI images using MATLAB” International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE), vol. 2, no. 1, (2012), p. 1.

[9]. S. Roy and S. K. Bandyopadhyay, “Detection and Quantification of Brain Tumor from MRI of Brain and it’s Symmetric Analysis”, International Journal of Information and Communication Technology Research, vol. 2, no. 6, (2012).

[10]. M. L.Toure and Z. Beiji, “Advanced algorithm for brain segmentation using fuzzy to localize cancer and epilespy region”, Electronics and Information Engineering (ICEIE), 2010 International Conference On,IEEE, (2010).

CHAPTER 3

INTRODUCTION TO MATLAB

3.1 INTRODUCTION

MATLAB stands for Matrix Laboratory. It is a high-performance language that is used for technical computing. It was developed by Cleve Molar of the company Math Works. Inc in the year 1984.It is written in C, C++, and Java. It allows matrix manipulations, plotting of functions, and implementation ofalgorithms and creation of user interfaces.

MATLAB is a programming platform designed specifically for engineers and scientists **to analyze and design systems and products that transform our world**. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.

MATLAB® combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly. It includes the Live Editor for creating scripts that combine code, output, and Formatted text in an executable notebook.

3.2 What Can I Do With MATLAB?

- Analyze data
- Develop algorithms
- Create models and applications

MATLAB lets you take your ideas from research to production by deploying to enterprise applications andembedded devices, as well as integrating with Simulink and Model-Based Design.

3.3 Getting started with MATLAB:

It is both a programming language as well as a programming environment. It allows the Computation of statements in the command window itself.

Command Window:

In this window one must type and immediately execute the statements, as it requiresquick prototyping. These statements cannot be saved. Thus, this is can be used for small, easily executable programs.

Editor (Script):

In this window one can execute larger programs with multiple statements, and complexFunctions These can be saved and are done with the file extension '.m '.

Workspace:

In this window the values of the variables that are created in the course of the program(In the editor) are displayed.

3.4 Who Uses MATLAB?

Millions of engineers and scientists worldwide use MATLAB for a range of applications, in industry andacademia, including deep learning and machine learning, signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology.

3.5 Writing a MATLAB program

1. **Using Command Window:**
   Only one statement can be typed and executed at a time. It executes the statement when theenter key is pressed. This is mostly used for simple calculations.
   Note: ans is a default variable created by MATLAB that store the output of the givencomputation.

2. **Using Editor:**
   Multiple lines of code can be written here and only after pressing the run button (or F5) willthe code be executed. It is always a good practice to write clc, clear and close all in the beginning of the program.
   Note: Statements ending with a semicolon will not be displayed in the command window, however, their values will be displayed in the workspace. Any statement followed by % in MATLAB is considered as a comment.

3. **Vector Operations:**
   Operations such as addition, subtraction, multiplication and division can be done using a singlecommand instead of multiple loops

CHAPTER 4

HARDWARE SPECIFICATION

4.1 Hardware Specification:

Operating System	Windows 11 Windows 10 (version 20H2 or higher) Windows Server 2019 Windows Server 2022
Processor	Minimum: Any Intel or AMD x86–64 processor Recommended: Any Intel or AMD x86–64 processor Note: A future release of MATLAB will require a processorwith AVX2
RAM	Minimum: 4 GB Recommended: 8 GB
Storage	3.8 GB for just MATLAB 4-6 GB for a typical installation 23 GB for an all products installationAn SSD is strongly recommended

4.2 Software Specification

¬MATLAB Version R2013

¬Operating system: Windows 7 and above

Advantages:

- Faster Result
- Good Accuracy
- Ease to use

Disadvantages:

- Requires clear image for processing

Application:

- This system can be used in banks and other co-operate sectors

CHAPTER 5

IMAGE PROCESSSING TECHNIQUES

5.1 ANISOTROPHIC DIFFUSION

In image processing and computer vision, anisotropic diffusion, also called Perona–Malikdiffusion, is a technique aiming at reducing image noise without removing significant parts of the image content, typically edges, lines or other details that are important for theinterpretation of the image.

Anisotropic diffusion resembles the process that creates a scale space, where an image generates a parameterized family of successively more and more blurred images based ona diffusion process.

This diffusion process is a linear and space-invariant transformation of the original image.Anisotropic diffusion is a generalization of this diffusion process: it produces a family of parameterized images, but each resulting image is a combination between the original image and a filter that depends on the local content of the original image.

As a consequence, anisotropic diffusion is a non-linear and space-variant transformationof the original image.

In its original formulation, presented by Perona and Malik in 1987, the space-variant filteris in fact isotropic but depends on the image content such that it approximates an impulse function close to edges and other structures that should be preserved in the image over thedifferent levels of the resulting scale space.

This formulation was referred to as anisotropic diffusion by Perona and Malik eventhough the locally adapted filter is isotropic, but it has also been referred to as inhomogeneous and nonlinear diffusion or Perona–Malik diffusion by other authors.

Although the resulting family of images can be described as a combination between the original image and space-variant filters, the locally adapted filter and its combination with the image do not have to berealized in practice.

Anisotropic diffusion is normally implemented by means of an approximation of the generalized diffusion equation: each new image in the family is computed by applying this equation to the previousimage.

Imdiffusefilt:

Anisotropic diffusion filtering of images.

Syntax:

```
J = imdiffusefilt (I)
J = imdiffusefilt (I, Name,Value)
```

Perform Edge-Preserving Smoothing Using Anisotropic Diffusion

Read an image into the workspace and display it:

```
I = imread('brain.jpg');
imshow (I)title ('Original
Image')
```

Figure 5.1 Edge-Preserving Smoothing Using Anisotropic Diffusion

Smooth the image using anisotropic diffusion. For comparison, also smooth the image using Gaussian blurring. Adjust the standard deviation sigma of the Gaussian smoothing kernel so thattextured regions, such as the grass, are smoothed a similar amount for both methods.

```
I diffusion = imdiffusefilt (I);

sigma = 1.2;
I gaussian = imgaussfilt(I, sigma);
```

Display the results.

```
montage({Idiffusion,Igaussian},'ThumbnailSize',[])

title('Smoothing Using Anisotropic Diffusion (Left) vs. Gaussian
Blurring (Right)')
```

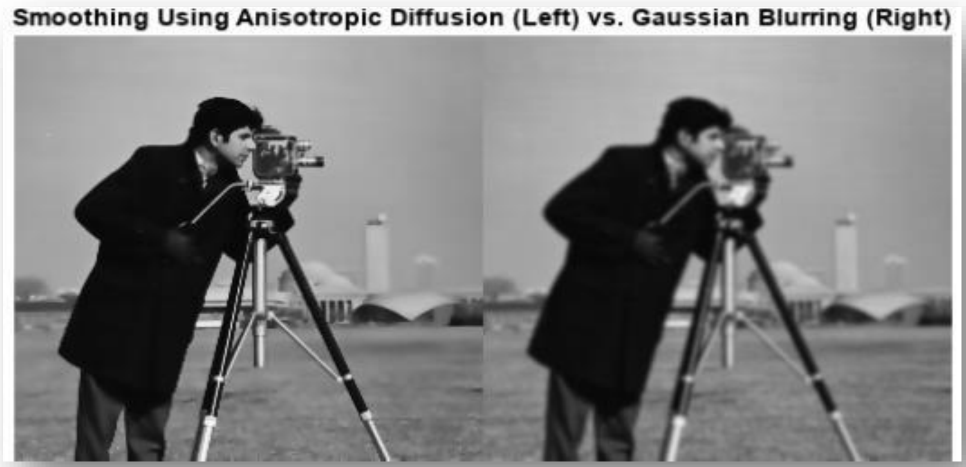

Smoothing Using Anisotropic Diffusion (Left) vs. Gaussian Blurring (Right)

Figure 5.2 Edge-Aware original and noise sample image

Anisotropic diffusion preserves the sharpness of edges better than Gaussian blurring.Perform Edge-Aware Noise Reduction Using Anisotropic Diffusion

Display the noisy image:

```
I = imread ('pout.tif');
NoisyImage = imnoise (I,'gaussian', 0, 0.005);
Imshow (noisyImage)
title ('Noisy Image')
```

Output:

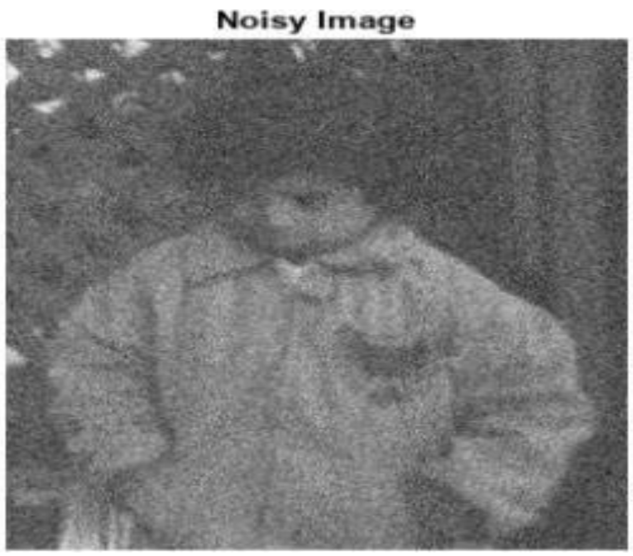


Figure 5.3 Edge-Aware Noise Reduction Using Anisotropic Diffusion

Compute the structural similarity index (SSIM) to measure the quality of the noisy image.the closer the SSIM value is to 1, the better the image agrees with the noiseless reference image.

```
n = ssim(I, noisyImage);
Disp (['The SSIM value of the noisy image is ', num2str (n),'.'])
```

The SSIM value of the noisy image is 0.26556.

Reduce the noise using anisotropic diffusion :

.

```
B = imdiffusefilt (noisy Image);Imshow (B)
t.title ('Anisotropic Diffusion with Default Parameters'nB = ssim (I, B);
```

Disp (['The SSIM value using default anisotropic diffusion is ', num2str (nB),'.'])The SSIM value using default anisotropic diffusion is 0.65665.

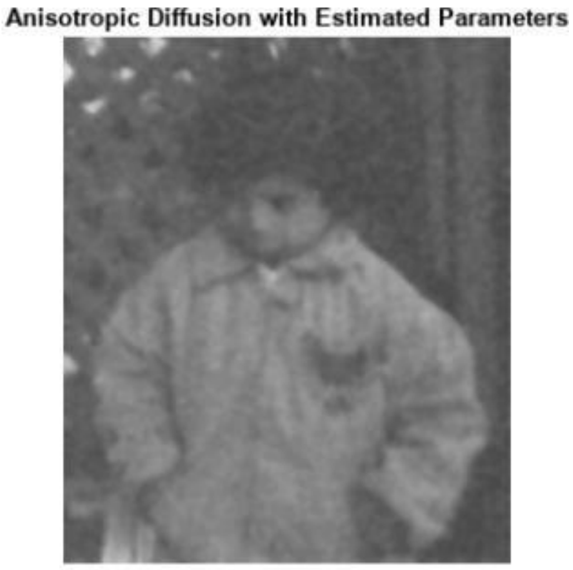


Figure 5.4 Anisotropic Diffusion with Estimated Parameters

```
nC = ssim(I,C);
Disp (['The SSIM value using quadratic anisotropic diffusion is ', num2str (nC),'.'])
```

The SSIM value using quadratic anisotropic diffusion n is 0.88135.

5.1.1 Perform 3-D Edge-Aware Noise ReductionLoad mristack

Perform edge-aware noise reduction on the volume using anisotropic diffusion.

The tradeoffs' is that less noise is removed.

```
diffused Image = imdiffusefilt(mristack,'NumberOfIterations',3);
```

To compare the noisy image and the filtered image in detail, display the tenth slice of both.

```
Imshowpair (mristack (:,:,10),diffused Image(:,:,10),'montage')
title ('Noisy Image (Left) vs. Anisotropic-Diffusion-Filtered Image (Right)
```

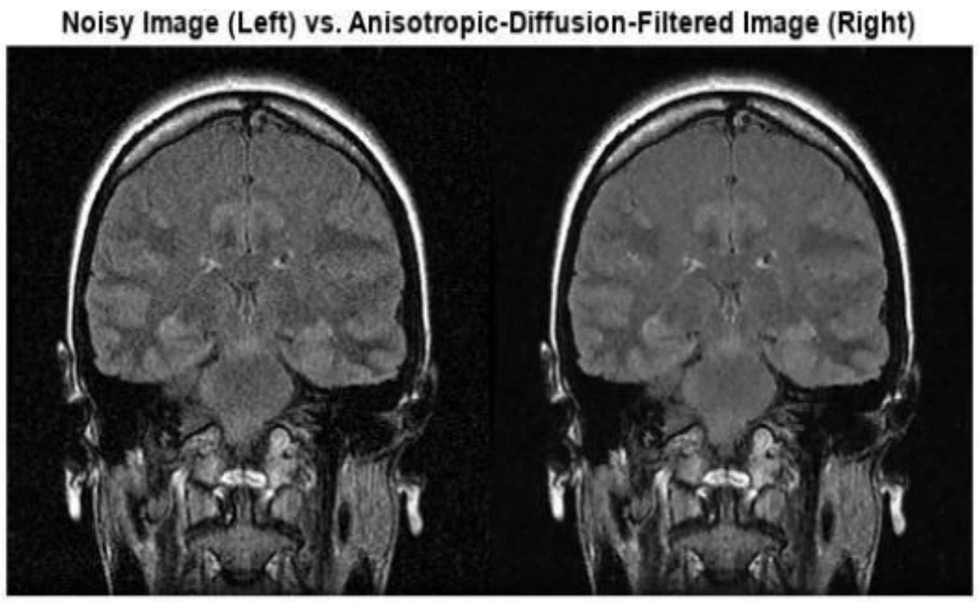


Figure 5.6 3-D Edge-Aware Noise Reduction

CHAPTER 6

METHODOLOGY

This section describes the details of the proposed method. The flowchart of the proposed method is shown in Figure 6.1. The detail of each step will be given in the following subsections.

6.1 Methodology

The proposed method contains main seven phases and the schematic diagram is shown in Figure 6.1.

- ✓ Getting Input image
- ✓ Preprocessing
- ✓ Thresholding
- ✓ Morphological operation
- ✓ Regionprops
- ✓ Bounding
- ✓ Getting outline of tumor

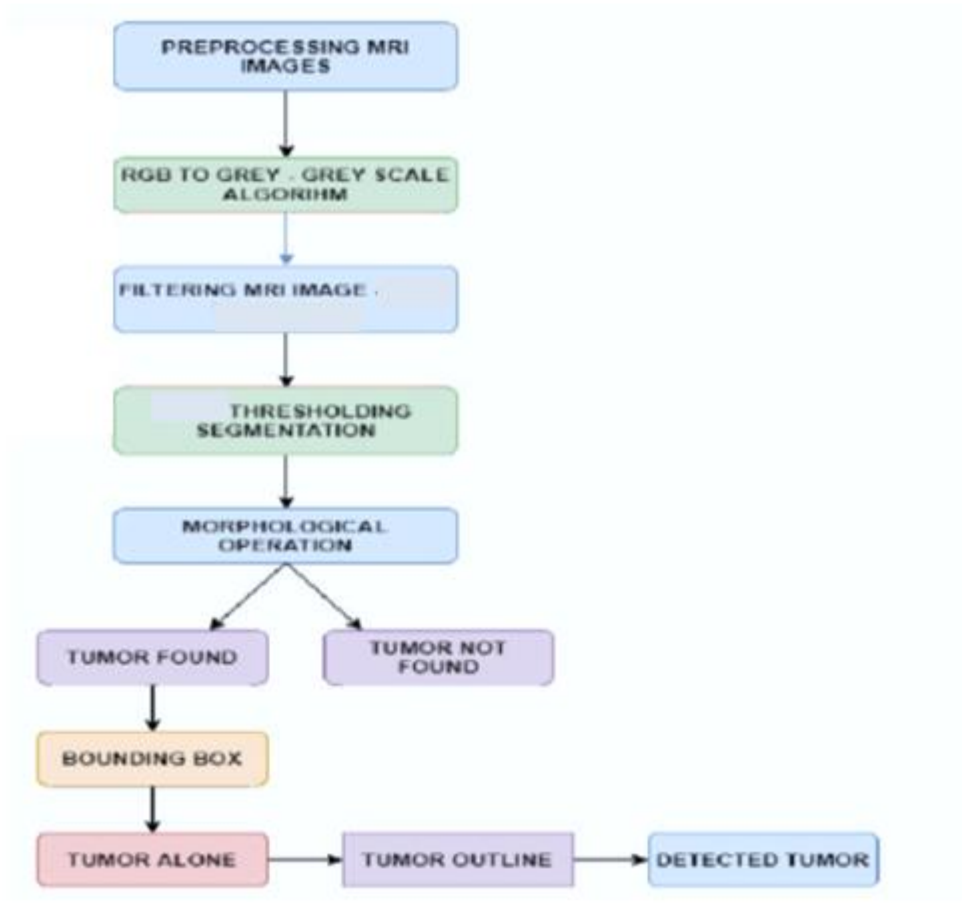


Figure 6.1 Schematic Diagrams of the Project

6.1.1 Getting MRI image

We can get MRI images of brain tumor data from datasets .We can able to download MRI images of brain tumor dataset from **Kaggle.**

After downloading the datset we need to implement it in the MATLAB code. The below coding describes how to input the MRI image in the MATLAB code.

CODING:

```
s=imread('C:\Users\PC\Downloads\project images\d.jpg');

figure;

imshow(s);

title('Input image','FontSize',20);
```

OUTPUT:

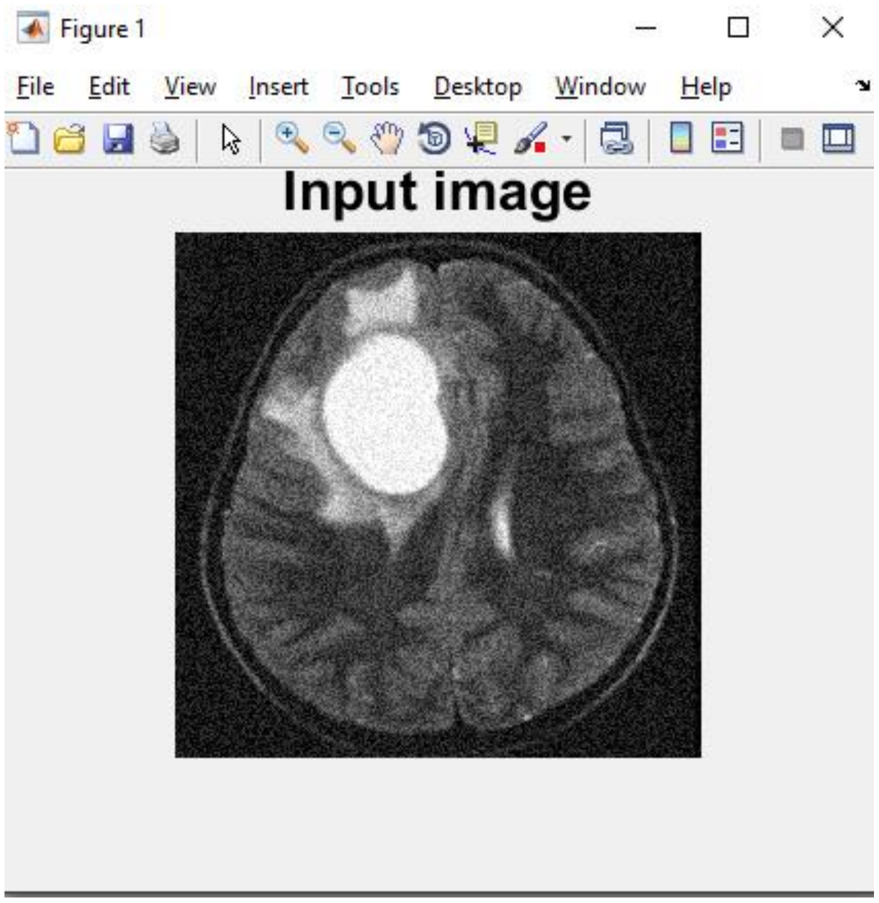


Figure 6.2 Input Image

6.1.2 IMAGE PREPROCESSING

In this process we filterering, smoothing and enhancing the MRI image using image preprocessing technique ie.),**Anisotropic Diffusion.**.

6.1.2.1 PREPROCESSING

- Data preprocessing is the concept of changing the raw data into a clean data set. The dataset is preprocessed in order to check missing values, noisy data, and other inconsistencies before executing it to the algorithm.
- **Data cleaning, smoothing, grouping**. **Data** can require preprocessing techniques to ensure accurate, efficient, or meaningful analysis.
- There are so many preprocessing techniques or methods are there ,the preprocessing methods used in here are,
- anisotropic diffusion filtering,
- thresholding, and
- region properties
- Here we applies anisotropic diffusion filtering to the image using the anisodiff function. This function is used to remove noise and enhance edges in the image.

6.1.2.2 FILTERING

Filtering is a technique for modifying or enhancing an image.

ANISOTROPIC DIFFUSION

Anisotropic Diffusion Filter is a technique used to remove noise from an image. **Anisotropic diffusion**, is also called **Perona–Malik diffusion**, .It is a non-linear filter that smooths the image while preserving edges. It is particularly useful in image processing applications.

Using anisodiff() funcion we can define this filtering

After filtering the MRI image we got the following output.

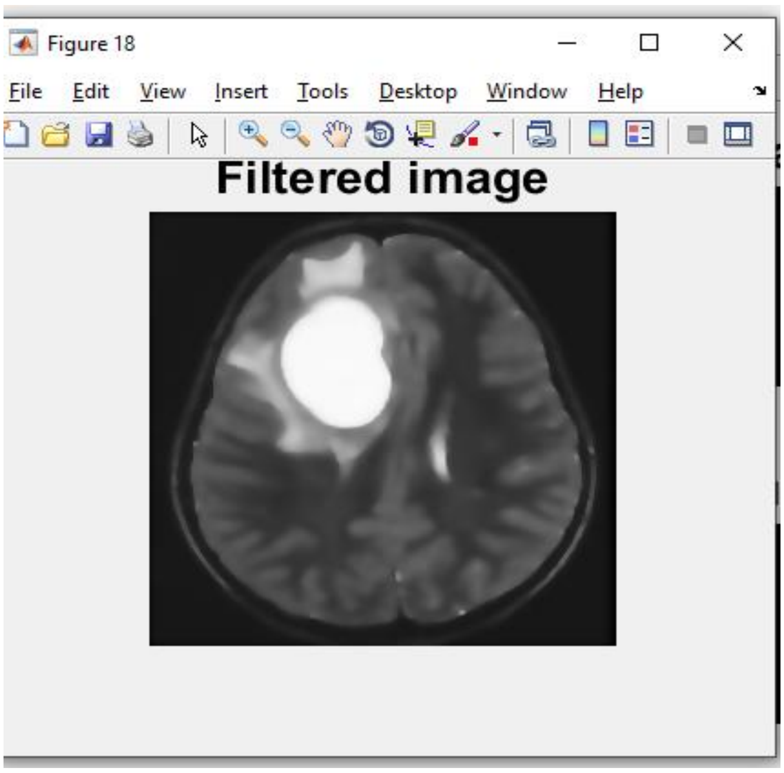


Figure 6.3 Preprocessed Image

6.1.2.3 THRESHOLDING

Applying threshold to the image to separate the tumor from the background.

Threshold is the separation of an image region into two regions. An area corresponds to the foreground area; in which it contains the objects of interest to us. The remaining area is the background, corresponding to unnecessary objects. This provides image segmentation based on different visual intensities and discontinuous intensities in the foreground and background regions.

Threshold based Image segmentation is a simple but powerful approach based on the characteristics of image. Thresholding operation converts a multilevel image into a binary image by means of a Threshold value T. Any pixel (x, y) is considered as a part of object if its intensity is greater than or equal to threshold value i.e., $f(x, y) \geqslant T$, else pixel belongs to the background. As per the selection of thresholding value, two types of thresholding methods are in existence [24], global and local thresholding.

Then the threshold value is then calculated using the mean intensity of the filtered image and a fixed value of 60. The thresholded image is obtained by setting all pixels with intensity greater than the threshold value to 1 and all other pixels to 0.

Thresholding is a type of image segmentation, where we change the pixels of an image to make the image easier to analyze.

The thresholded image is then labeled using the bwlabel function to identify the connected components in the image.

After thresholding the image we need to do morphological operations in it.

6.1.2.4 MORPHOLOGICAL OPERATION

Morphology is used to extract image components useful in representing region shape, boundaries, etc.

Morphology is a broad set of image processing operations that process images based on shapes.

once the segmentation is complete, morphological operations can be used to remove imperfections in the segmented image and deliver information on the shape and structure of the image as shown in next image.

After morphological operations regionprops function will happen.

6.1.2.5 REGION PROPS

The regionprops function is used to calculate the solidity and area of each connected component.

Then it selects the connected component with the highest area and creates a binary image of the tumor using the ismember function.

If the area of the selected connected component is greater than 100, the code displays the tumor alone.

Otherwise, the code displays a message box indicating that no tumor was found.

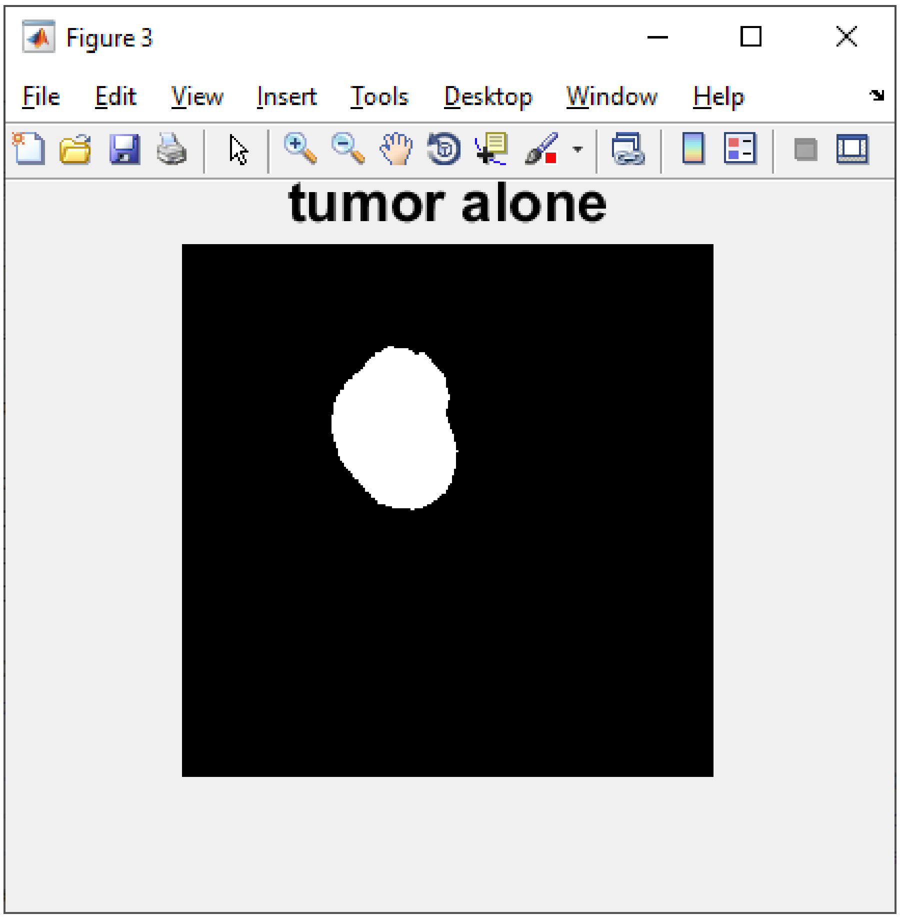
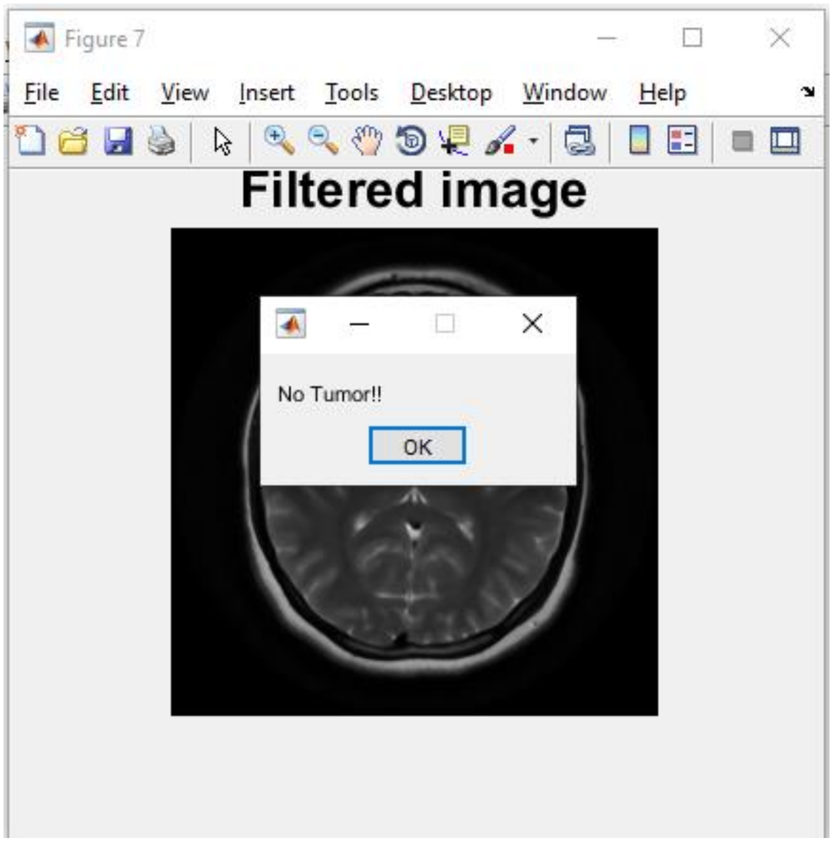


Figure 6.4 Tumor alone image

After applying thresholding and morphological operation into the MRI image , If we input an MRI image that doesn’t have any tumor it display that the image has no tumor in a message box.

The output is below,



No Tumor!!

Figure 6.5 No tumor image

6.1.2.6 BOUNDING BOX

A bounding box is drawn around the tumor using the rectangle function. A bounding box is an abstract rectangle that acts as a reference point for object detection and produces a collision box for that object.

Using the bounding box for Object Detection. The computer wants to know what an object is and where it is to detect it in an image.

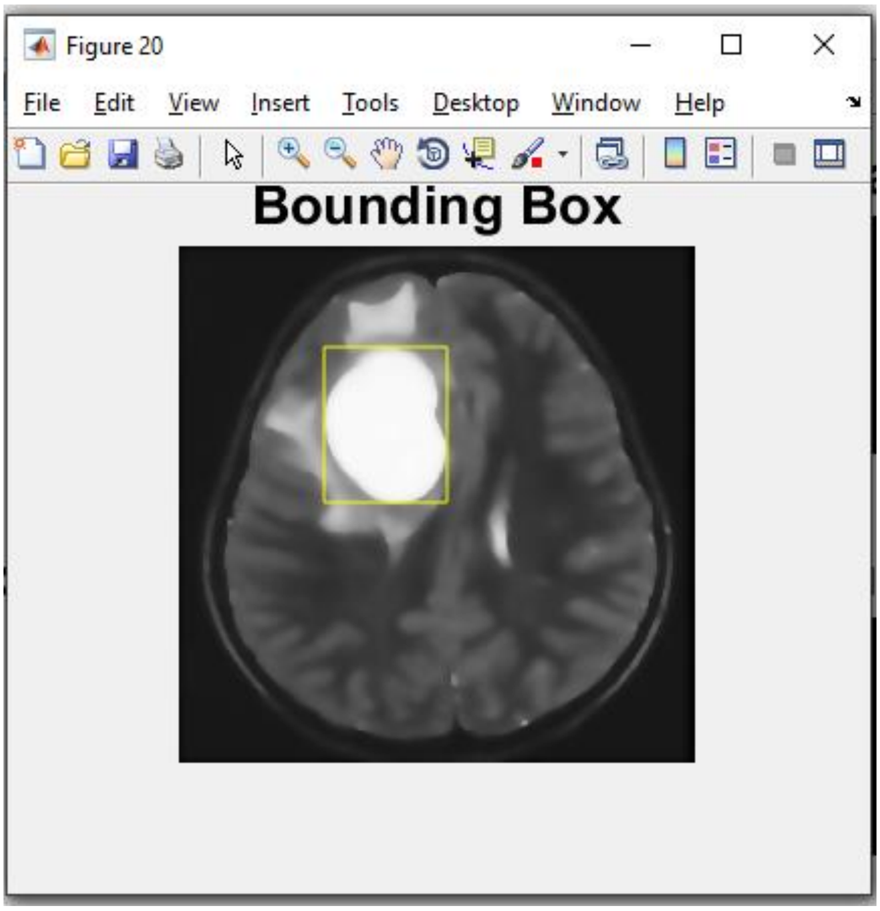


Figure 6.6 Bounded Tumor

6.1.2.7 GETTING OUTLINE OF TUMOR

To detect the tumor outline we need to do dilation, erosion, subtracting process on the MRI Image.

Dilation:

Dilation adds pixels to the boundaries of objects in an image.Here, it fills the holes in the connected component.

Erosion:

erosion removes pixels on object boundaries. Here the dilated part will remove.

Subtracting:

Then we subtracts the eroded image from the binary image of the tumor to obtain the tumor outline.

Here we subtracting eroded image from original BW image.

After this process we got an RGB image of the original image with the tumor outline overlaid on it.

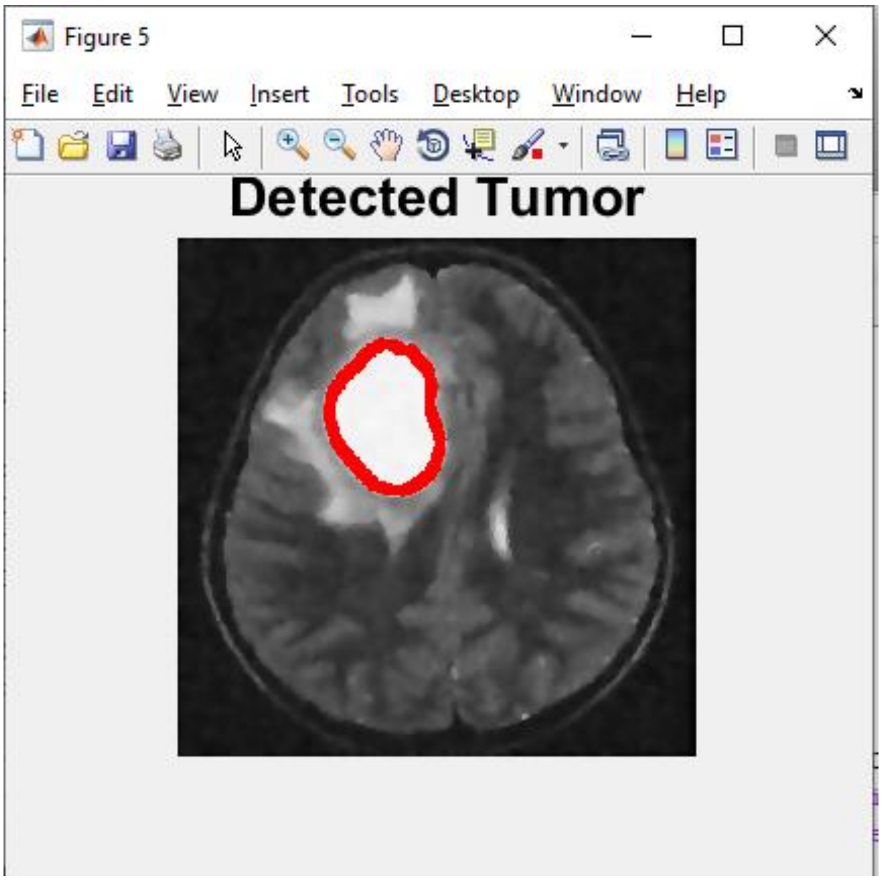


Figure 6.7 Detected Tumor

CHAPTER 7

CODING AND RESULT

7.1 CODING

7.1.1 FINAL.M

```
s=imread('C:\Users\PC\Downloads\project dataset\c1.jpg');
figure;
imshow(s);
title('Input image','FontSize',20);
num_iter = 10;
    delta_t = 1/7;
    kappa = 15;
    option = 2;
    disp('Preprocessing image please wait . . .');
    inp = anisodiff(s,num_iter,delta_t,kappa,option);
    inp = uint8(inp);

inp=imresize(inp,[256,256]);
if size(inp,3)>1
    inp=rgb2gray(inp);
end
figure;
imshow(inp);
title('Filtered image','FontSize',20);
sout=imresize(inp,[256,256]);
t0=60;
th=t0+((max(inp(:))+min(inp(:)))./2);
```

```
for i=1:1:size(inp,1)
    for j=1:1:size(inp,2)
        if inp(i,j)>th
            sout(i,j)=1;
        else
            sout(i,j)=0;
        end
    end
end
label=bwlabel(sout);
stats=regionprops(logical(sout),'Solidity','Area','BoundingBox');
density=[stats.Solidity];
area=[stats.Area];
high_dense_area=density>0.6;
max_area=max(area(high_dense_area));
tumor_label=find(area==max_area);
tumor=ismember(label,tumor_label);
if max_area>100
   figure;
   imshow(tumor)
   title('tumor alone','FontSize',20);
else
    h = msgbox('No Tumor!!','status');
    %disp('no tumor');
    return;
end
```

```
box = stats(tumor_label);
wantedBox = box.BoundingBox;
figure
imshow(inp);
title('Bounding Box','FontSize',20);
hold on;
rectangle('Position',wantedBox,'EdgeColor','y');
hold off;
dilationAmount = 5;
rad = floor(dilationAmount);
[r,c] = size(tumor);
filledImage = imfill(tumor, 'holes');
for i=1:r
   for j=1:c
       x1=i-rad;
       x2=i+rad;
       y1=j-rad;
       y2=j+rad;
       if x1<1
           x1=1;
       end
       if x2>r
           x2=r;
       end
       if y1<1
           y1=1;
```

```
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end
tumorOutline=tumor;
tumorOutline(erodedImage)=0;

rgb = inp(:,:,[1 1 1]);
red = rgb(:,:,1);
red(tumorOutline)=255;
green = rgb(:,:,2);
green(tumorOutline)=0;
blue = rgb(:,:,3);
blue(tumorOutline)=0;
tumorOutlineInserted(:,:,1) = red;
tumorOutlineInserted(:,:,2) = green;
tumorOutlineInserted(:,:,3) = blue;
figure
imshow(tumorOutlineInserted);
title('Detected Tumor','FontSize',20);
figure
subplot(231);imshow(s);title('Input image            ','FontSize',20);
subplot(232);imshow(inp);title('     Filtered image        ','FontSize',20);
subplot(233);imshow(inp);title('        Bounding Box   ','FontSize',20);

hold on;rectangle('Position',wantedBox,'EdgeColor','y');hold off;

subplot(235);imshow(tumor);title('tumor alone       ','FontSize',20);

subplot(236);imshow(tumorOutlineInserted);title('Detected Tumor','FontSize',20);
```

7.1.2 ANISODIFF.M

```
functiondiff_im = anisodiff(im, num_iter, delta_t, kappa, option)

fprintf('Removing noise\n');

fprintf('Filtering Completed !!');

% Convert input image to double.im = double(im);

% PDE (partial differential equation) initial condition.diff_im = im;

% Center pixel distances.dx = 1;

dy = 1;

dd = sqrt(2);

% 2D convolution masks - finite differences.hN = [0 1 0; 0 -1 0; 0 0 0];

hS = [0 0 0; 0 -1 0; 0 1 0];

hE = [0 0 0; 0 -1 1; 0 0 0];

hW = [0 0 0; 1 -1 0; 0 0 0];

hNE = [0 0 1; 0 -1 0; 0 0 0];

hSE = [0 0 0; 0 -1 0; 0 0 1];

hSW = [0 0 0; 0 -1 0; 1 0 0];

hNW = [1 0 0; 0 -1 0; 0 0 0];

for t = 1:num_iter

% Finite differences. [imfilter(.,.,'conv') can be replaced by conv2(.,.,'same')]

nablaN = imfilter(diff_im,hN,'conv');

nablaS = imfilter(diff_im,hS,'conv'); nablaW = imfilter(diff_im,hW,'conv');

 nablaE = imfilter(diff_im,hE,'conv'); nablaNE = imfilter(diff_im,hNE,'conv');

nablaSE = imfilter(diff_im,hSE,'conv'); nablaSW = imfilter(diff_im,hSW,'conv');

nablaNW = imfilter(diff_im,hNW,'conv');
```

```
% Diffusion function.if option == 1

cN =  exp(-(nablaN/kappa).^2);

cS  = exp(-(nablaS/kappa).^2);

cW = exp(-(nablaW/kappa).^2);

cE = exp(-(nablaE/kappa).^2);

cNE = exp(-(nablaNE/kappa).^2);

cSE = exp(-(nablaSE/kappa).^2);

cSW = exp(-(nablaSW/kappa).^2);

cNW = exp(-(nablaNW/kappa).^2);

elseif option == 2

cN = 1./(1 + (nablaN/kappa).^2);

 cS = 1./(1 + (nablaS/kappa).^2);

cW = 1./(1 + (nablaW/kappa).^2);

cE = 1./(1 + (nablaE/kappa).^2);

cNE = 1./(1 +(nablaNE/kappa).^2);



cSE = 1./(1 + (nablaSE/kappa).^2);

cSW = 1./(1+ (nablaSW/kappa).^2);

cNW = 1./(1 + (nablaNW/kappa).^2);

end

diff_im =

diff_im + ...
```

```
delta_t*(...

(1/(dy^2))*cN.*nablaN + (1/(dy^2))*cS.*nablaS + ...

(1/(dx^2))*cW.*nablaW + (1/(dx^2))*cE.*nablaE + ...

(1/(dd^2))*cNE.*nablaNE + (1/(dd^2))*cSE.*nablaSE + ...

(1/(dd^2))*cSW.*nablaSW + (1/(dd^2))*cNW.*nablaNW );


end
```

7.2 FINAL OUTPUT

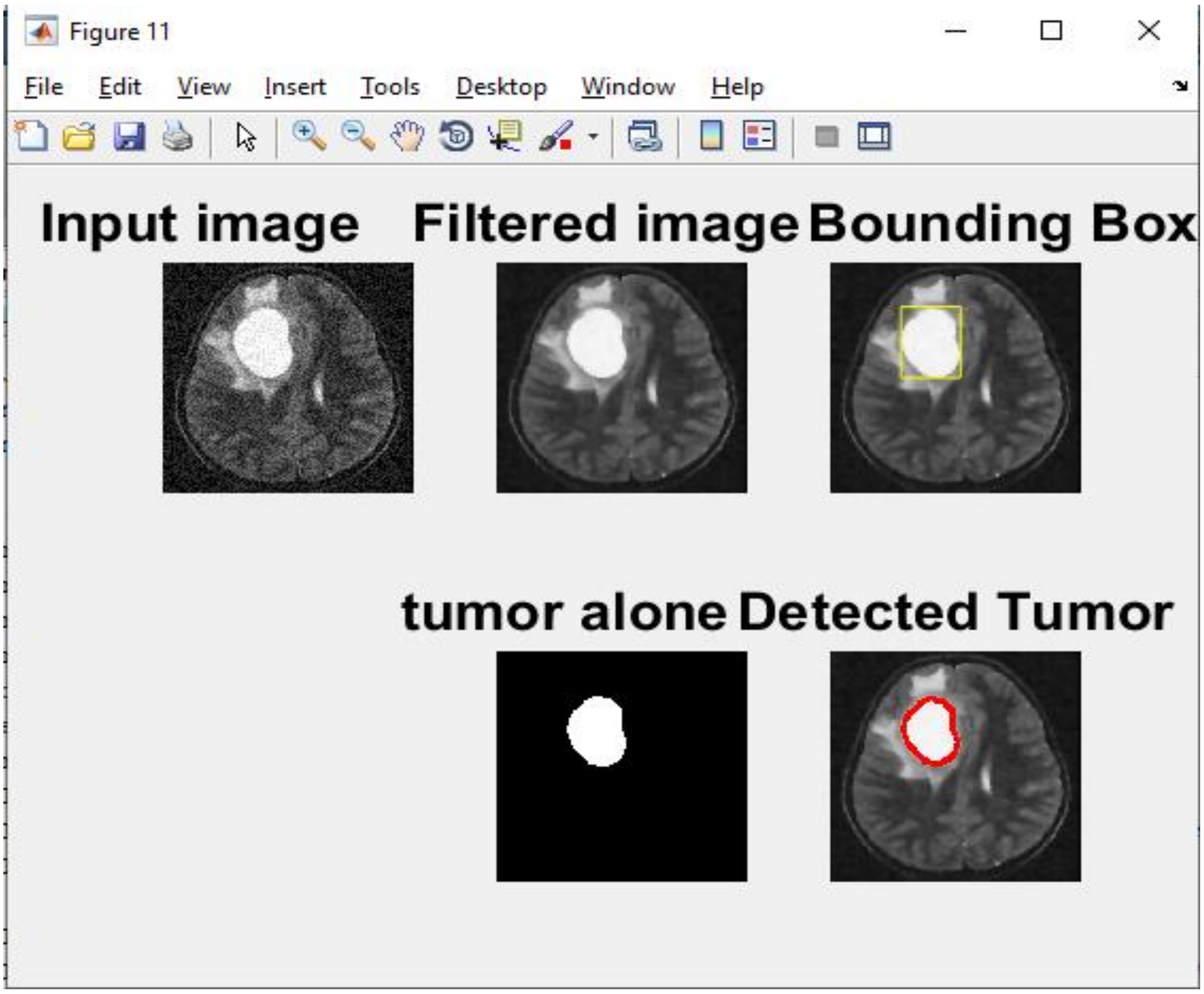


Figure 7.1 Final Output

CHAPTER-8

CONCLUSION

Anisotrophic diffusion filetring technique have obtained good results in recent years to preprocess image in the medical image analysis field because It is better than bilinear and trilinear filtering.

In this model, we implemented the necessary phases such as image preprocessing, thresholding segmentation, bounding and morphological operation.

The implementation produced excellent results, as shown in the above output results figures.

In brain tumor detection, the proposed scheme achieved an accuracy of 95.0%