Advanced Digital Image Processing- Mini Project Report on

"Canny edge detection technique for identifying tumor in MRI Brain Image"

By

Vaishnavi C K

Under the guidance of

Dr. Shikha Tripathi

INTRODUCTION

This project focuses on the application of the canny edge detection technique in identifying brain tumor from MRI image.

The identification of the tumor in the MRI brain image will be done based on the intensity levels and on the similarities of the regions in the image, for this the canny edge detection technique is used.

This method identifies the discrepancies in the image, and is more precisely aimed at finding unrelated illumination of the image, especially along the edges where the intensity changes sharply.

ALGORITHM

Steps involved in canny edge detection:

1. Image smoothing and filtering - Gaussian function is used to smoothen the image before the edge detection process in order to reduce the noise. The Gaussian kernel/filter is applied to the input image for this.

2. Gradient Calculation - The Gaussian derivative in X direction is determined and this filter is applied on the image (Img_dx). Similarly gaussian derivative in Y direction is determined and applied on the image (Img_dy). The gradient is computed as follow,
   Img_gradient= sqrt ($(Img\_dx)^2 + (Img\_dy)^2$)

3. Non-max Suppression - The image gradient produced results in thick edges. Ideally, the final image should have thin edges. Non maximum suppression is performed to thin out the edges.
   Non maximum suppression works by finding the pixel with the maximum value in an edge. If the pixel has higher intensity than the other pixels in its gradient direction, it is kept, else set the pixel to zero (make it a black pixel).

4. Double Thresholding - The gradient magnitudes are compared with two specified threshold values. Gradients that are smaller than the low threshold

value are suppressed while those which are higher than the high threshold value are marked as strong ones.

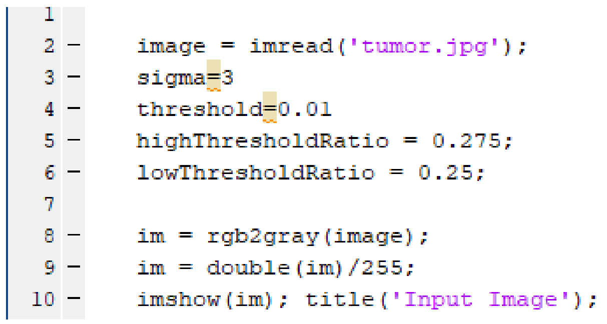
5. Edge tracking by hysteresis - To determine which weak edges are actual edges, edge tracking is done. The weak edges that are connected to strong edges will be actual/real edges. The weak edges that are not connected to strong edges will be removed.

METHODOLOGY

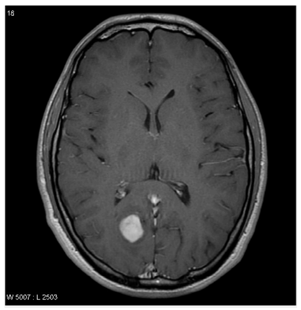
1. The input image (MRI of the brain) is read and from RGB it is converted to grayscale.
double(img)/255 converts the grayscale image to double precision and normalizes to an interval of [0, 1].

The variance (*sigma^2*) is assumed to be 9. This implies sigma value is 3.
The threshold value is taken as 0.01.

The lower threshold ratio is taken as 0.25 and upper threshold ratio is taken as 0.275.



```

image = imread('tumor.jpg');
sigma=3
threshold=0.01
highThresholdRatio = 0.275;
lowThresholdRatio = 0.25;

im = rgb2gray(image);
im = double(im)/255;
imshow(im); title('Input Image');
```



2. Meshgrid function is used to obtain the gaussian filter.
Kernel of size 25x25 and sigma of 3 is used to perform gaussian blur on the image.
Meshgrid function is used to create multi-dimensional arrays.

X=floor(25/2) = 12 will be the range for determining x values of meshgrid (ie -12 to 12 along x-coordinate)
Y=floor(25/2) = 12 will be the range for determining y values of meshgrid (ie -12 to 12 along y-coordinate)

Generating Xmtx and Ymtx using meshgrid:
[Xmtx, Ymtx] = meshgrid(-X:X, -Y:Y) = meshgrid(-12:12, -12:12)
Here Xmtx is a matrix of order 25x25 and its values are:

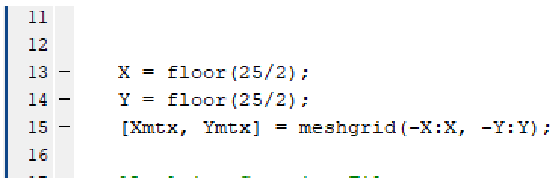
$$\begin{bmatrix} -12 & -11 & \dots & \dots & 0 & \dots & \dots & 11 & 12 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & ; \\ -12 & -11 & \dots & \dots & 0 & \dots & \dots & 11 & 12 \end{bmatrix}$$ **25x25**

Ymtx is also of order 25x25 and its values are:

$$\begin{bmatrix} -12 & .. & .. & .. & -12 \\ -11 & .. & .. & .. & -11 \\ \vdots & .. & .. & .. & \vdots \\ \vdots & .. & .. & .. & \vdots \\ 0 & .. & .. & .. & 0 \\ \vdots & .. & .. & .. & \vdots \\ \vdots & .. & .. & .. & \vdots \\ 11 & .. & .. & .. & 11 \\ 12 & .. & .. & .. & 12 \end{bmatrix}$$ **25x25**

Ymtx is the transpose of Xmtx.

Xmtx and Ymtx are used in the gaussian expression so as to compute the gaussian filter.



```
X = floor(25/2);
Y = floor(25/2);
[Xmtx, Ymtx] = meshgrid(-X:X, -Y:Y);
```

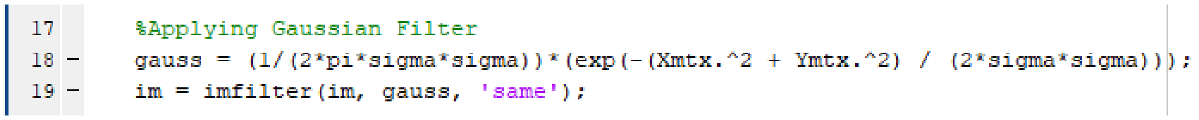
3. The equation for the gaussian filter kernel is given by

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

The Gaussian filter is determined using this equation by taking x as Xmtx and y as Ymtx.
It is then applied on the input image array using imfilter.

im = imfilter(im,gauss,'same')

This operation filters the 2-D input array with the gaussian filter (gauss) according to the condition ‘same’ (i.e. output array dimensions will be same as that of input array) specified.



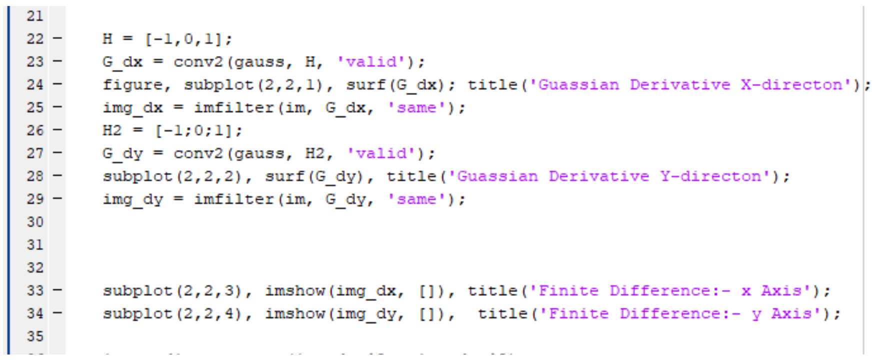
```
%Applying Gaussian Filter
gauss = (1/(2*pi*sigma*sigma))*(exp(-(Xmtx.^2 + Ymtx.^2) / (2*sigma*sigma)));
im = imfilter(im, gauss, 'same');
```

4. Gaussian derivatives in the x and y direction are determined.
To determine gaussian derivative along x axis, the gaussian filter is convolved with [-1,0,1]

G_dx = conv2(gauss, H, 'valid')
This operation computes the 2-D convolution of the gaussian filter with H=[-1,0,1] and returns only those parts of the convolution that are computed without the zero-padded edges.

img_dx = imfilter(im, G_dx, 'same')
G_dx is then applied on the image (on which gaussian filter is already applied) using imfilter and the output of this is the finite difference in x axis (img_dx)

Similarly gaussian derivative along y (i.e G_dy) is computed by convolving with $[-1,0,1]^T$ and G_dy is then applied on the image (on which gaussian filter is already applied) using imfilter and the output of this is the finite difference in y axis (img_dy)



```

H = [-1,0,1];
G_dx = conv2(gauss, H, 'valid');
figure, subplot(2,2,1), surf(G_dx); title('Guassian Derivative X-directon');
img_dx = imfilter(im, G_dx, 'same');
H2 = [-1;0;1];
G_dy = conv2(gauss, H2, 'valid');
subplot(2,2,2), surf(G_dy), title('Guassian Derivative Y-directon');
img_dy = imfilter(im, G_dy, 'same');



subplot(2,2,3), imshow(img_dx, []), title('Finite Difference:- x Axis');
subplot(2,2,4), imshow(img_dy, []),  title('Finite Difference:- y Axis');

```

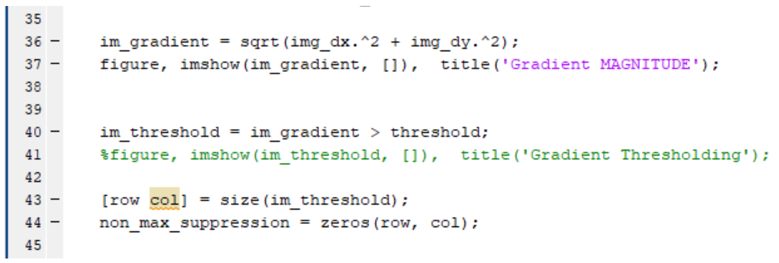
5. Image gradient magnitude is computed using the expression:
Gradient= $sqrt((img\_dx)^2 + (img\_dy)^2)$

Having obtained the gradient, thresholding is applied.
im_threshold = im_gradient > threshold

Threshold value is 0.01

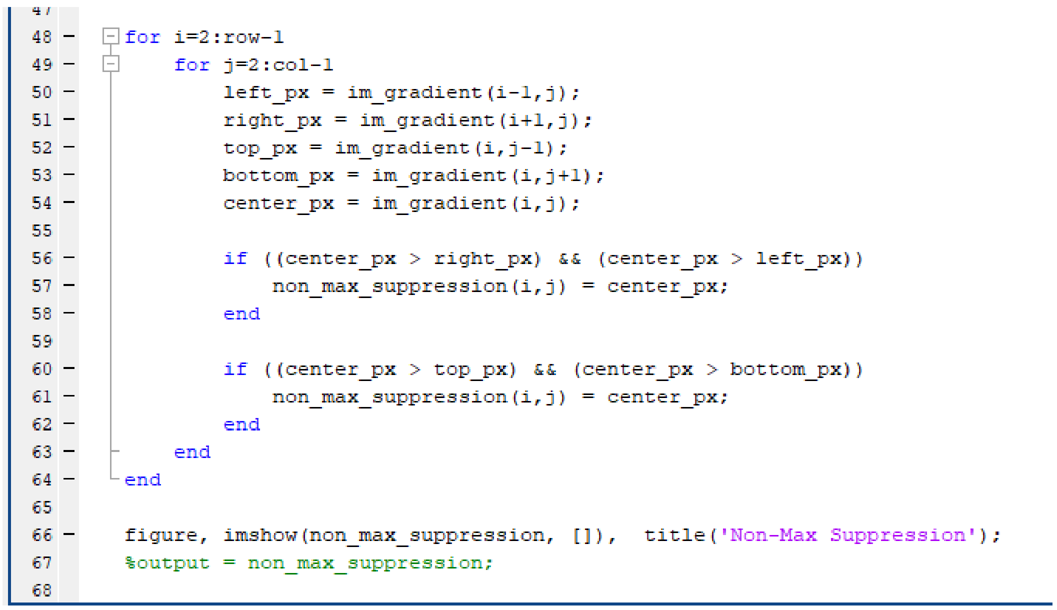
Values which satisfy gradient>threshold are allocated the pixel value 1 and pixel value 0 is allocated for values that don't satisfy the condition gradient >threshold. Hence im_threshold is a black and white image.

Non- max suppression will have the same dimension as of im_threshold and it is initialized with values 0.



```

im_gradient = sqrt(img_dx.^2 + img_dy.^2);
figure, imshow(im_gradient, []),  title('Gradient MAGNITUDE');


im_threshold = im_gradient > threshold;
%figure, imshow(im_threshold, []),  title('Gradient Thresholding');

[row col] = size(im_threshold);
non_max_suppression = zeros(row, col);

```

6. Non max suppression is implemented by comparing the pixel values with its 4 neighbours.
Pixel values(i,j) in the image gradient whose intensity is greater than its left pixel (i-1,j), right pixel (i+1,j) , top pixel (i,j-1) and bottom pixel (i,j+1) only are retained. This gives the non max suppression output.



```
for i=2:row-1
    for j=2:col-1
        left_px = im_gradient(i-1,j);
        right_px = im_gradient(i+1,j);
        top_px = im_gradient(i,j-1);
        bottom_px = im_gradient(i,j+1);
        center_px = im_gradient(i,j);

        if ((center_px > right_px) && (center_px > left_px))
            non_max_suppression(i,j) = center_px;
        end

        if ((center_px > top_px) && (center_px > bottom_px))
            non_max_suppression(i,j) = center_px;
        end
    end
end

figure, imshow(non_max_suppression, []),  title('Non-Max Suppression');
%output = non_max_suppression;

```

7. Double thresholding

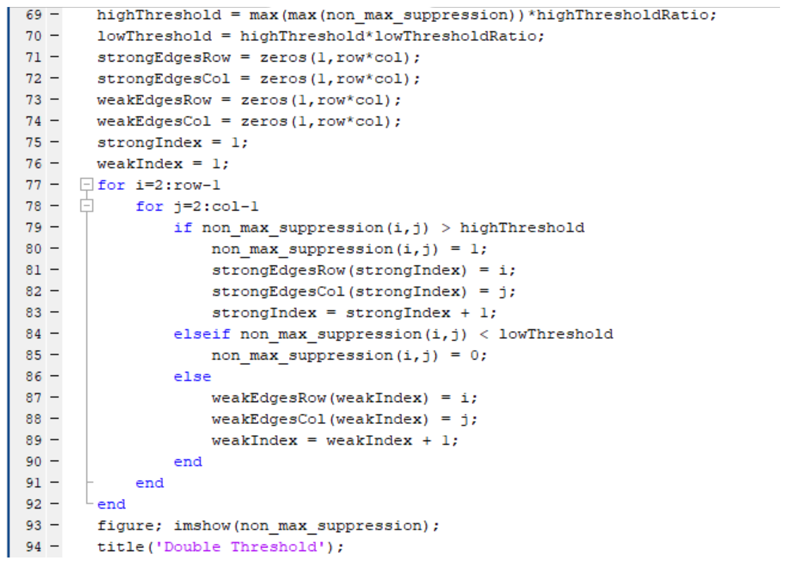
Double thresholding is done so as to identify the strong, weak and non-relevant pixels.
Instead of defining specific threshold values, threshold ratios are defined (arbitrarily). The upper limit on the threshold (ie high_threshold) is obtained by multiplying the high threshold ratio with the maximum pixel value in the image. The lower limit on the threshold (ie low_threshold) is obtained by multiplying the low threshold ratio with the high_threshold value.

Pixel values (intensity) more than the high_threshold value are identified as strong pixels. Pixel values less than the low_threshold value are identified as non-relevant pixels and are discarded. Pixel values that lie in between low_threshold and high_threshold are identified as weak pixels.

Strong and weak pixels contribute to edge detection. Row matrices are defined so as to keep track of the row index and column index of the strong and weak edges.

On iterating through the pixel values in the non max suppression output, each pixel value is compared with the threshold values and correspondingly assigned as strong edge and given value 1 (white) and non-relevant pixels are discarded by assigning it value 0 (black).

These values are updated in the non max suppression output matrix. The corresponding coordinates of the strong and weak edge pixels are identified and stored in the respective row matrices.

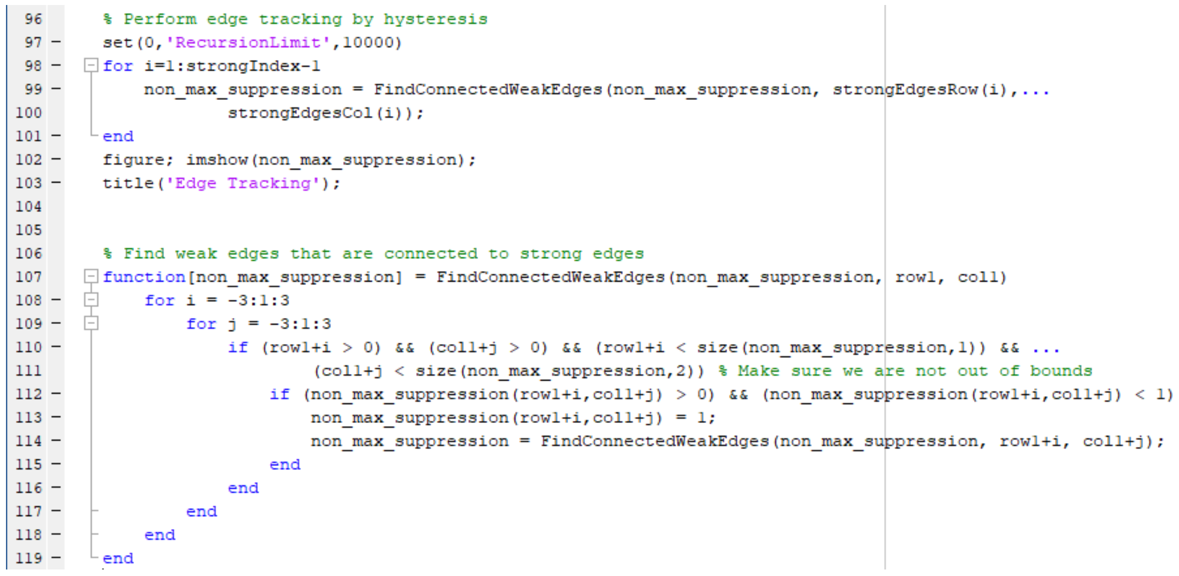


```
highThreshold = max(max(non_max_suppression))*highThresholdRatio;
lowThreshold = highThreshold*lowThresholdRatio;
strongEdgesRow = zeros(1,row*col);
strongEdgesCol = zeros(1,row*col);
weakEdgesRow = zeros(1,row*col);
weakEdgesCol = zeros(1,row*col);
strongIndex = 1;
weakIndex = 1;
for i=2:row-1
    for j=2:col-1
        if non_max_suppression(i,j) > highThreshold
            non_max_suppression(i,j) = 1;
            strongEdgesRow(strongIndex) = i;
            strongEdgesCol(strongIndex) = j;
            strongIndex = strongIndex + 1;
        elseif non_max_suppression(i,j) < lowThreshold
            non_max_suppression(i,j) = 0;
        else
            weakEdgesRow(weakIndex) = i;
            weakEdgesCol(weakIndex) = j;
            weakIndex = weakIndex + 1;
        end
    end
end
figure; imshow(non_max_suppression);
title('Double Threshold');
```

8. Edge tracking
Edge tracking is done so as to determine the weak edges which are actual edges. Weak edges that are connected to the strong edges are actual edges and those not connected to strong edges are discarded.

Edge tracking is done by recursively iterating through the strong edges.

The pixel values near the coordinates of the strong edge coordinates, that lie between 0 and 1 are identified as the weak edges connected to the strong edges and are set to 1 (white), since these contribute to the final edge.



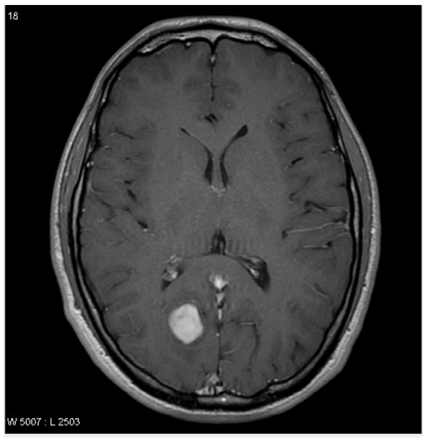
```
% Perform edge tracking by hysteresis
set(0,'RecursionLimit',10000)
for i=1:strongIndex-1
    non_max_suppression = FindConnectedWeakEdges(non_max_suppression, strongEdgesRow(i),...
            strongEdgesCol(i));
end
figure; imshow(non_max_suppression);
title('Edge Tracking');


% Find weak edges that are connected to strong edges
function[non_max_suppression] = FindConnectedWeakEdges(non_max_suppression, row1, col1)
    for i = -3:1:3
        for j = -3:1:3
            if (row1+i > 0) && (col1+j > 0) && (row1+i < size(non_max_suppression,1)) && ...
                    (col1+j < size(non_max_suppression,2)) % Make sure we are not out of bounds
                if (non_max_suppression(row1+i,col1+j) > 0) && (non_max_suppression(row1+i,col1+j) < 1)
                    non_max_suppression(row1+i,col1+j) = 1;
                    non_max_suppression = FindConnectedWeakEdges(non_max_suppression, row1+i, col1+j);
                end
            end
        end
    end
end
```

SOFTWARE

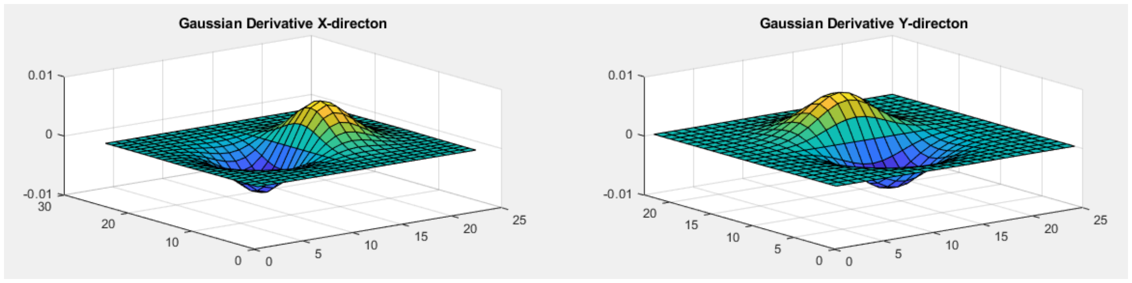
MATLAB

RESULTS

Input Image (MRI image of brain with tumor) -



Gaussian Derivatives (G_dx and G_dy) -



Finite Difference (Img_dx and Img_dy) -

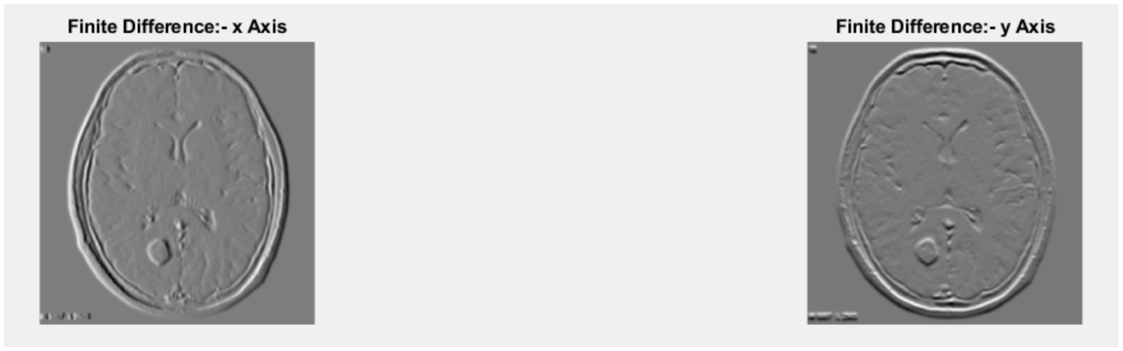
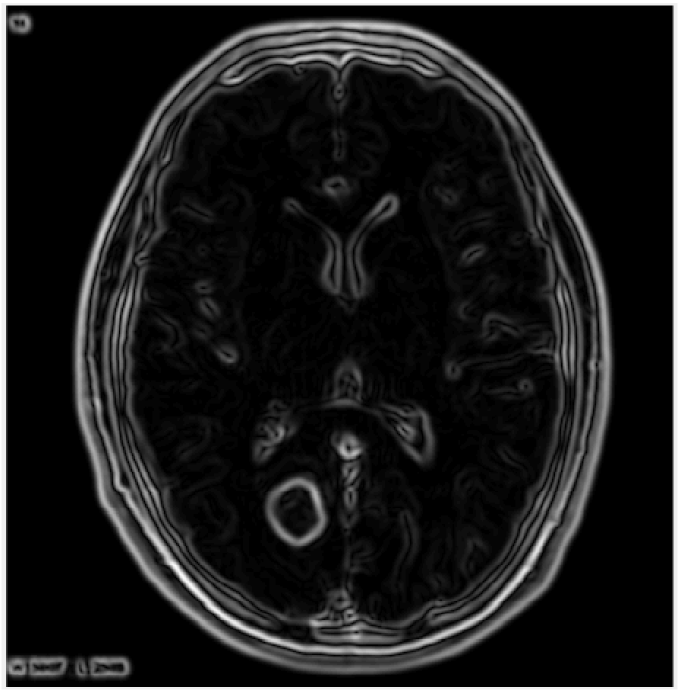
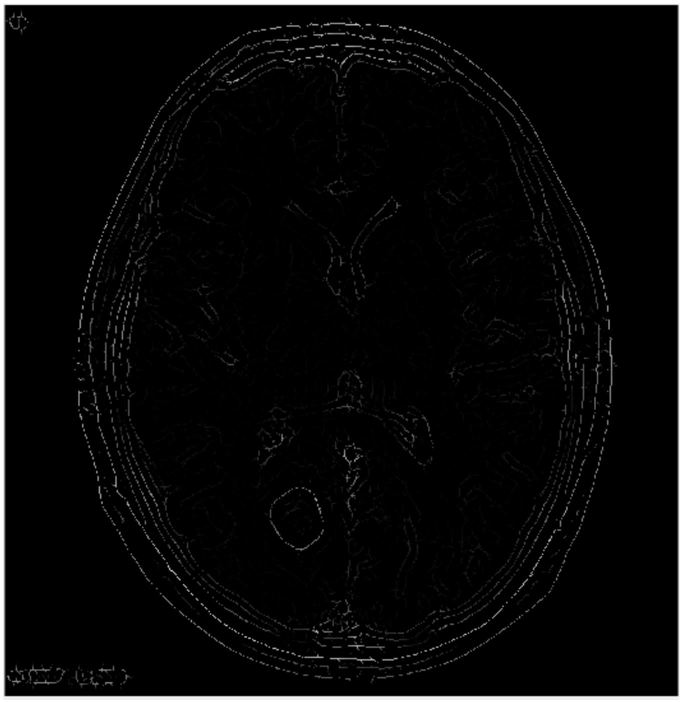


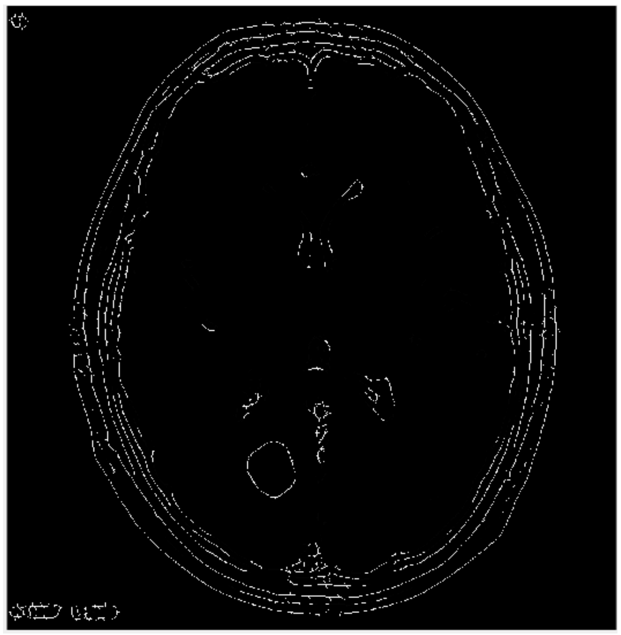
Image Gradient -



Non Max Suppression Output -



Double Thresholding -



Edge Tracking by Hysteresis -