function **fun\_load\_image**\_Callback()

%% Function Load Image

global Brain\_RGB\_Img inp

[f\_name, p\_name] = uigetfile('\*.\*', 'Load Image File');

if isequal(f\_name,0)||isequal(p\_name,0)

warndlg('Failed to Load Image', 'Warning');

else

% Read Image if variable are successfully loaded

Brain\_RGB\_Img = imread([p\_name, f\_name]);

Brain\_RGB\_Img = imresize (Brain\_RGB\_Img,[256,256]);

% Show Input Loaded Image on Axes 1 Location

axes(handles.axes1);

imshow(Brain\_RGB\_Img);

total\_itr = 10;

del\_t = 1/7;

kap = 15;

opt = 2;

%% Calling to Anisotropic Diffusion Function

inp = fun\_Anisotropic\_Diffusion(Brain\_RGB\_Img,total\_itr,del\_t,kap,opt);

% Converting the matrix to Image Format

inp = uint8(inp);

% Resizing to 256 x 256

inp=imresize(inp,[256,256]);

% checking if the image is in rgb, then convert to gray image

if size(inp,3)==3

inp=rgb2gray(inp);

end

end

function **fun\_High\_Pass\_Filter**\_Callback()

%% Function High Pass Filter

global Brain\_RGB\_Img inp filter\_High\_Pass\_Img

%% Possibility 1

kernel = [-1 -1 -1;-1 8 -1;-1 -1 -1];

filter\_High\_Pass\_Img = imfilter(Brain\_RGB\_Img, kernel, 'same');

%% Possibility 2

% H = fspecial('unsharp');

% filter\_High\_Pass\_Img = imfilter(Brain\_RGB\_Img, H);

% Showing on Axes

axes(handles.axes7);

imshow(filter\_High\_Pass\_Img);

function **fun\_grayscale**\_Callback(

%% Conversion

global Brain\_RGB\_Img gray\_input\_image

if(size(Brain\_RGB\_Img,3)>1)

gray\_input\_image = rgb2gray(Brain\_RGB\_Img);

else

gray\_input\_image=Brain\_RGB\_Img;

end

% Showing on Axes

axes(handles.axes8);

%%%%

imshow(gray\_input\_image);

function **fun\_Median\_Filter**\_Callback()

%% Function Mediun Filter

global Brain\_RGB\_Img img\_med\_filter binary\_input\_image

binary\_input\_image=im2bw(Brain\_RGB\_Img);

img\_med\_filter = medfilt2(binary\_input\_image);

axes(handles.axes2);

%%%%%%%%%%%%

imshow(img\_med\_filter);

function **fun\_Watershed\_Segmentation**\_Callback()

global inp Segmented\_RGB

%% Watershed Segmentation

New\_size=[256,256];

resized=imresize(inp,New\_size);

t\_0=60;

thr=t\_0+((max(inp(:))+min(inp(:)))./2);

for idx=1:1:size(inp,1)

for jdx=1:1:size(inp,2)

if inp(idx, jdx)>thr

resized(idx, jdx)=1;

else

resized(idx, jdx)=0;

end

end

end

% Applying Sobel Filter

y\_dir = fspecial('sobel');

Iy = imfilter(double(resized), y\_dir, 'replicate');

Ix = imfilter(double(resized), y\_dir', 'replicate');

% Taking Squareroot

magnitude = sqrt(Ix.^2 + Iy.^2);

L = watershed(magnitude);

Segmented\_RGB = labl2rgb(L);

% Showing Segmented Image

axes(handles.axes3);

imshow(Segmented\_RGB);

fun\_morphological\_refinments.

function **fun\_morphological\_refinments**\_Callback()

%% Function Morphological Refinments

global inp tumor\_labl labl stats dens detected\_tumor\_area sout h tumor

sout=imresize(inp,[256,256]);

t\_00=60;

th2= t\_00+((max(inp(:))+min(inp(:)))./2);

for iii=1:1:size(inp,1)

for jjj=1:1:size(inp,2)

if inp(iii,jjj)> th2

sout(iii,jjj)=1;

else

sout(iii,jjj)=0;

end

end

end

%% Morphological Operation

labl=bwlabl(sout);

%% detected\_tumor\_area Calculation using RegionProps Function

stats=regionprops(logical(sout),'Solidity','Area','BoundingBox');

dens=[stats.Solidity];

detected\_tumor\_area=[stats.Area];

high\_dense\_detected\_tumor\_area=dens>0.6;

max\_detected\_tumor\_area=max(detected\_tumor\_area(high\_dense\_detected\_tumor\_area));

tumor\_labl=find(detected\_tumor\_area==max\_detected\_tumor\_area);

tumor=ismember(labl,tumor\_labl);

if max\_detected\_tumor\_area>100

axes(handles.axes4);

imshow(tumor);

else

h = msgbox('No Tumor!!','status');

return;

end

fun\_Tumor\_detected\_tumor\_area\_Location.

function **fun\_Tumor\_Area\_Location**\_Callback()

%% Tumor detected\_tumor\_area

global inp tumor\_labl sout h tumor

labl=bwlabl(sout);

stats=regionprops(logical(sout),'Solidity','Area','BoundingBox');

dens=[stats.Solidity];

detected\_tumor\_area=[stats.Area];

high\_dense\_detected\_tumor\_area=dens>0.6;

max\_detected\_tumor\_area=max(detected\_tumor\_area(high\_dense\_detected\_tumor\_area));

tumor\_labl=find(detected\_tumor\_area==max\_detected\_tumor\_area);

tumor=ismember(labl,tumor\_labl);

if max\_detected\_tumor\_area>100

imshow(tumor);

set(handles.edit2,'String','Tumor present');

else

h = msgbox('No Tumor!!','status');

set(handles.edit2,'String','No tumor');

return;

end

bbox = stats(tumor\_labl);

estBox = bbox.BoundingBox;

axes(handles.axes5);

imshow(inp);

% Drawing Rectangle Box Around the Detected Tumor

hold on;

rectangle('Position', estBox,'EdgeColor','y');

hold off;

dil\_Amt = 5;

rad\_Amt = floor(dil\_Amt);

[rows,cols] = size(tumor);

filledimg = imfill(tumor, 'holes');

for i=1:rows

for j=1:cols

x\_1=i- rad\_Amt;

x\_2=i+ rad\_Amt;

y\_1=j- rad\_Amt;

y\_2=j+ rad\_Amt;

if x\_1<1

x\_1=1;

end

if x\_2>r

x\_2=r;

end

if y\_1<1

y\_1=1;

end

if y\_2>c

y\_2=c;

end

eroded\_Img (i,j) = min(min(filled\_img(x\_1:x\_2,y\_1:y\_2)));

end

end

tumor\_Outln =tumor – eroded\_Img;

axes(handles.axes6);

imshow(tumor\_Outln);

function **fun\_tumor\_boundary**\_Callback()

%% Boundary Plotting

global tumor Brain\_RGB\_Img

[new\_rows,new\_cols,channel]=size(Brain\_RGB\_Img);

resized\_Brain\_RGB\_Img=imresize(tumor,[new\_rows,new\_cols]);

[Bdry,L] = bwboundaries(resized\_Brain\_RGB\_Img,'noholes');

axes(handles.axes6);

imshow(Brain\_RGB\_Img);

hold on

for q = 1:length(Bdry)

boundary\_new = Bdry{q};

plot(boundary\_new(:,2), boundary\_new (:,1), 'y', 'LineWidth', 2)

end

function pushbutton6\_Callback()

function **btn\_classify\_tumor**\_Callback()

%% Function to Classify the Detected Tumor

global tumor

if(exist('tumor','var') == 1)

if(size(tumor,1)==0 & size(tumor,2)==0)

disp('Not OK');

else

disp('OK');

global detected\_tumor\_area white\_pixel\_count detected\_tumor\_perimeter1 tumor x y labledImage

global total\_pixels\_2 detected\_tumor\_perimeter total\_pixels\_1 detected\_tumor\_centroid

white\_pixel\_count = sum(tumor(:))

labledImage = bwlabl(tumor);

region\_measure = regionprops(tumor, ...

'Area', 'Centroid', 'Perimeter');

detected\_tumor\_area = region\_measure.Area

detected\_tumor\_centroid = region\_measure.Centroid

detected\_tumor\_perimeter = region\_measure.Perimeter

% Calculate the detected\_tumor\_area, in pixels

total\_pixels\_1 = sum(tumor(:));

total\_pixels\_2 = bwarea(tumor);

detected\_tumor\_area=sqrt(total\_pixels\_2);

%Conversion from pixels to mm

detected\_tumor\_area=detected\_tumor\_area\*0.26458333;

detected\_tumor\_perimeter1=detected\_tumor\_perimeter\*0.26458333;

% Get coordinates of the boundary in tumor

bw\_boundry = bwboundaries(tumor);

xy=bw\_boundry{1};

x = xy(:, 2);

y = xy(:, 1);

set(handles.text8,'String',detected\_tumor\_area);

set(handles.text9,'String',detected\_tumor\_perimeter1);

set(handles.text16,'String',detected\_tumor\_centroid);

if total\_pixels\_2 <= 100

set(handles.edit1,'String','NoTumor');

elseif (total\_pixels\_2 >= 100) && (total\_pixels\_2 <=2000)

set(handles.edit1,'String','Low');

elseif (total\_pixels\_2 >= 2000) && (total\_pixels\_2 <=4500)

set(handles.edit1,'String','Medium');

else

% % If Total Pixels of Tumor are more than 4500

set(handles.edit1,'String','High');

end

message = sprintf('Number of pixels = %d\ndetected\_tumor\_area in pixels = %.2f\ndetected\_tumor\_perimeter = %.2f\ndetected\_tumor\_centroid at (x,y) = (%.1f, %.1f)\n', ...

total\_pixels\_1, total\_pixels\_2, detected\_tumor\_perimeter, ...

detected\_tumor\_centroid(1), detected\_tumor\_centroid(2));

msgbox(message);

%% Plotting Centroid

axes(handles.axes6);

plot(detected\_tumor\_centroid(1), detected\_tumor\_centroid(2),'r.', 'markers', 10);

end

else

disp('Not OK');

end

function pushbutton9\_Callback()

function pushbutton10\_Callback()

function pushbutton11\_Callback()

function pushbutton12\_Callback()