DETECTION OF BRAIN TUMOR USING MRI IMAGES

```
fontSize = 20;
grayImage = rgb2gray(imread('03.jpg'));
grayImage1 = grayImage(:, : );
% Get the dimensions of the image.  numberOfColorBands should be = 1.
[rows columns numberOfColorBands] = size(grayImage);
% Display the original gray scale image.
subplot(3, 4, 1);
imshow(grayImage);
title('Original Grayscale Image', 'FontSize', fontSize);
% Enlarge figure to full screen.
set(gcf, 'Position', get(0,'Screensize'));
set(gcf,'name','Image Viewer','numbertitle','off')

% HPF
[m, n]=size(grayImage);
f_transform=fft2(grayImage);
f_shift=fftshift(f_transform);
p=m/2;
q=n/2;
d0=30;
for i=1:m
   for j=1:n
distance=sqrt((i-p)^2+(j-q)^2);
low_filter(i,j)=1-exp(-(distance)^2/(2*(d0^2)));
   end
end
filter_apply=f_shift.*low_filter;
image_orignal=ifftshift(filter_apply);
image_filter_apply=abs(ifft2(image_orignal));
% Display the image.
subplot(3, 4, 2);
imshow(image_filter_apply);
title('High Pass Filtered Image', 'FontSize', fontSize);

% MF
imflt=medfilt2(image_filter_apply, [11 11]);
% Display the image.
subplot(3, 4, 3);
imshow(imflt);
title('Median Filtered Image', 'FontSize', fontSize);

% Threshold segmentation 1
% Threshold the image to make a binary image.
```

```
thresholdValue = 100;
TS = grayImage > thresholdValue;
% Display the image.
subplot(3, 4, 4);
imshow(TS);
title('Binary image threshold value 100', 'FontSize', fontSize);


% Extract the outer blob, which is the skull.
% The outermost blob will have a label number of 1.
labeledImage = bwlabel(TS);                % Assign label ID numbers to all blobs.
TS = ismember(labeledImage, 1);% Use ismember() to extract blob #1.
% Thicken it a little with imdilate().
TS = imdilate(TS, true(0));
% Display the final binary image.
%subplot(3, 4, 10);
%imshow(TS);
%title('Binary image of Skull Alone', 'FontSize', fontSize);


% Mask out the skull from the original gray scale image.
skullFreeImage = grayImage; % Initialize
skullFreeImage(TS) = 0; % Mask out.
% Display the image.
subplot(3, 4, 5);
imshow(skullFreeImage);
title('Skull Free Image', 'FontSize', fontSize);


% Now threshold to find the tumor
thresholdValue = 150;
TS = skullFreeImage > thresholdValue;
% Display the image.
subplot(3, 4, 6);
imshow(TS);
title('Binary image threshold value 150', 'FontSize', fontSize);

% Assume the tumor is the largest blob, so extract it
binaryTumorImage = bwareafilt(TS, 1);
% Display the image.
subplot(3, 4, 7);
imshow(binaryTumorImage);
title('Tumor Alone', 'FontSize', fontSize);

% Find tumor boundaries.
```

```
% bwboundaries() returns a cell array, where each cell contains the row/column coordinates for an object in the image.
% Plot the borders of the tumor over the original grayscale image using the coordinates returned by bwboundaries.
subplot(3, 4, 8);
imshow(grayImage, []);
axis on;
caption = sprintf('Tumor\nOutlined in red in the overlay');
title(caption, 'FontSize', fontSize, 'Color', 'r');
axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
hold on;
boundaries = bwboundaries(binaryTumorImage);
numberOfBoundaries = size(boundaries, 1);
for k = 1 : numberOfBoundaries
	thisBoundary = boundaries{k};
	% Note: since array is row, column not x,y to get the x you need to use the second column of thisBoundary.
	plot(thisBoundary(:,2), thisBoundary(:,1), 'r', 'LineWidth', 2);
end
hold off;

% Now indicate the tumor a different way, with a red tinted overlay instead of outlines.
subplot(3, 4, 9);
imshow(grayImage, []);
caption = sprintf('Tumor\nSolid & tinted red in overlay');
title(caption, 'FontSize', fontSize, 'Color', 'r');
axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
hold on;
% Display the tumor in the same axes.
% Make a truecolor all-red RGB image.  Red plane has the tumor and the green and blue planes are black.
redOverlay = cat(3, ones(size(binaryTumorImage)), zeros(size(binaryTumorImage)), zeros(size(binaryTumorImage)));
hRedImage = imshow(redOverlay); % Save the handle; we'll need it later.
hold off;
axis on;
% Now the tumor image "covers up" the gray scale image.
% We need to set the transparency of the red overlay image to be 30% opaque (70% transparent).
alpha_data = 0.3 * double(binaryTumorImage);
set(hRedImage, 'AlphaData', alpha_data);
```

DETECTION OF WHETHER THERE IS A TUMOR OR NO TUMOR (UI.m)

```
function varargout = ui(varargin)
% UI MATLAB code for ui.fig
%      UI, by itself, creates a new UI or raises the existing
%      singleton*.
%
%      H = UI returns the handle to a new UI or the handle to
%      the existing singleton*.
%
%      UI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UI.M with the given input arguments.
%
%      UI('Property','Value',...) creates a new UI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ui

% Last Modified by GUIDE v2.5 28-Nov-2022 22:45:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ui_OpeningFcn, ...
                   'gui_OutputFcn',  @ui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before ui is made visible.
function ui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ui (see VARARGIN)

% Choose default command line output for ui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


global pcb_model;
global img_path;
% UIWAIT makes ui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in btn_select.
function btn_select_Callback(hObject, eventdata, handles)
% hObject    handle to btn_select (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% axes(handles.axes1);
% image = imread('1.jpg');
% imshow(image);

axes(handles.axes1);
```

```
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.png;*.jpeg;*.tif'},'Select a file','F:\test');
str=[pathname filename];
global img_path;
img_path=str;
% ≈–∂œŒƒº¸ «∑ÒŒ™ø'£¨"≤ø…"'≤ª"√',∏ˆ≤Ÿ◊˜£°÷±Ω"∂¡»ÎÕºΔ¨"≤ø…"'µƒ
% im = imread(str);
% imshow(im)

global pcb_model;
if isequal(filename,0)||isequal(pathname,0)
    warndlg('please select a picture first!','warning');
    return;
else
    pcb_model = imread(str);
    imshow(imresize(pcb_model, 0.1));
end


% --- Executes on button press in btn_detect.
function btn_detect_Callback(hObject, eventdata, handles)
% hObject    handle to btn_detect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% axes(handles.axes2);
global pcb_model;
global img_path;
size(pcb_model);
% if isequal(filename,0)||isequal(pathname,0)
%     warndlg('please select a picture first!','warning');
%     return;
% else
I = imread(img_path);
% gray = rgb2gray(I);
% % figure, imshow(gray);
%
% I = imread('01.jpg');
[rows, columns, numberOfColorChannels] = size(I);
if numberOfColorChannels > 1
        % It's not really gray scale like we expected - it's color.
        grayImage = rgb2gray(I); % Take green channel.
else
    grayImage=I;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
adjust=imadjust(grayImage)
% subplot(2,3,2);imshow(adjust);
% title('Adjust Contrast')
% MF
imflt=medfilt2(adjust,[1 1]);
% subplot(2,3,3);imshow(imflt);
% title('Median Pass Filter')
axes(handles.axes2);
imshow(imflt);

% Threshold the image to make a binary image.
n=imhist(imflt); % Compute the histogram
N=sum(n); % sum the values of all the histogram values
max=0; %initialize maximum to zero
for i=1:256
   P(i)=n(i)/N; %Computing the probability of each intensity level
end
for T=2:255      % step through all thresholds from 2 to 255
   w0=sum(P(1:T)); % Probability of class 1 (separated by threshold)
   w1=sum(P(T+1:256)); %probability of class2 (separated by threshold)
   u0=dot([0:T-1],P(1:T))/w0; % class mean u0
   u1=dot([T:255],P(T+1:256))/w1; % class mean u1
   sigma=w0*w1*((u1-u0)^2); % compute sigma i.e variance(between class)
   if sigma>max % compare sigma with maximum
      max=sigma; % update the value of max i.e max=sigma
      threshold=T-1; % desired threshold corresponds to maximum variance of between
class
   end
end
bw=im2bw(imflt,threshold/255); % Convert to Binary Image
%==========================111111111111111111
============================
% Extract the outer blob, which is the skull.
% The outermost blob will have a label number of 1.
labeledImage = bwlabel(bw);                % Assign label ID numbers to all blobs.
bw = ismember(labeledImage, 1);% Use ismember() to extract blob #1.
% Thicken it a little with imdilate().
bw = imdilate(bw, true(15));%%%%%%%%%%%%%%
axes(handles.axes3);
imshow(bw);
%==========================================================================
========
% Mask out the skull from the original gray scale image.
skullFreeImage = imflt; % Initialize
skullFreeImage(bw) = 0; % Mask out.
% Display the image.
```

```
% subplot(2, 3, 6);
% imshow(skullFreeImage, []);
% axis on;
% caption = sprintf('Gray Scale Image\nwith Skull Stripped Away');
% title(caption, 'FontSize', 15, 'Interpreter', 'None');
axes(handles.axes4);
imshow(skullFreeImage);
%=================================================================
=======================================
% Now threshold to find the tumor
thresholdValue = 150;
% % Threshold the image to make a binary image.
n1=imhist(skullFreeImage); % Compute the histogram
N1=sum(n1(2:256)) % sum the values of all the histogram values

pixels_nums = 0;
index = 30;
mean_gray=50;
for h=2:256
    pixels_nums=pixels_nums+n1(h);
    mean_gray = mean_gray+n1(h)*h;
    if pixels_nums > N1*0.8
        index = h;
        break;
    end
end
mean_gray=mean_gray/sum(n1(2:index))

thresholdValue = mean_gray+50

% for T=256:-1:2
%     nn1=sum(n1(T:256));
%     nn2=sum(n1(T-1:256));
%     if nn1/N1 < 0.03 && nn2/N1 > 0.03
%         thresholdValue = T;
%         break;
%     end
% end

thresholdValue

binaryImage = skullFreeImage >
thresholdValue;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% % Display the image.
% hFig2 = figure();
```

```
% subplot(2, 2, 1);
% imshow(binaryImage, []);
% axis on;
% caption = sprintf('Initial Binary Image\nThresholded at %d Gray Levels',
thresholdValue);
% title(caption, 'FontSize', 15, 'Interpreter', 'None');
% % Set up figure properties:
% % Enlarge figure.
% set(gcf, 'Units', 'Normalized', 'OuterPosition', [0.25 0.15 .5 0.7]);
% % Get rid of tool bar and pulldown menus that are along top of figure.
% % set(gcf, 'Toolbar', 'none', 'Menu', 'none');
% % Give a name to the title bar.
% set(gcf, 'Name', 'Demo by ImageAnalyst', 'NumberTitle', 'Off')
% drawnow;
axes(handles.axes5);
imshow(binaryImage);
%=================================================================
=======================================
% Assume the tumor is the largest blob, so extract it
binaryTumor = bwareafilt(binaryImage,1);
% % Display the image.
% subplot(2, 2, 2);
% imshow(binaryTumor, []);
% axis on;
% caption = sprintf('Tumor Alone');
% title(caption, 'FontSize', 15, 'Interpreter', 'None');

%=================================================================
=======================================
% Find tumor boundaries.
% bwboundaries() returns a cell array, where each cell contains the row/column
coordinates for an object in the image.
% Plot the borders of the tumor over the original grayscale image using the coordinates
returned by bwboundaries.
% subplot(2, 2, 3);
% imshow(grayImage, []);
% axis on;
% caption = sprintf('Tumor\nOutlined in red in the overlay');
% title(caption, 'FontSize', 15, 'Color', 'r');
% axis image; % Make sure image is not artificially stretched because of screen's
aspect ratio.
% hold on;
boundaries = bwboundaries(binaryTumor);
numberOfBoundaries = size(boundaries, 1);

% for k = 1 : numberOfBoundaries
```

```
%      thisBoundary = boundaries{k};
%      % Note: since array is row, column not x,y to get the x you need to use the second
column of thisBoundary.
%      plot(thisBoundary(:,2), thisBoundary(:,1), 'r', 'LineWidth', 2);
% end
% hold off;

top= 100000;
bot = 0;
left = 10000;
right = 0;

res = zeros(size(grayImage));
for k = 1 : numberOfBoundaries
        thisBoundary = boundaries{k};
    res(thisBoundary(:,1), thisBoundary(:,2)) = 1;
end
a1 = sum(res);%(1,297)

for s = 1:size(a1,2)
    if a1(s) > 1
        left = s;
        break;
    end
end
for t = size(a1,2):-1:1
    if a1(t) > 1
        right = t;
        break;
    end
end

a1 = sum(res,2);%(1,297)

for s = 1:size(a1,1)
    if a1(s) > 1
        top = s;
        break;
    end
end
for t = size(a1,1):-1:1
    if a1(t) > 1
        bot = t;
        break;
    end
end
```

```
% [row,col,channel]=size(I)
P=I;
for i = top:bot
    P(i, left,1) = 255;
    P(i, right,1) = 255;
    P(i, left,2) = 0;
    P(i, right,2) = 0;
    P(i, left,3) = 0;
    P(i, right,3) = 0;
end
for j = left:right
    P(top, j,1) = 255;
    P(bot, j,1) = 255;
    P(top, j,2) = 0;
    P(bot, j,2) = 0;
    P(top, j,3) = 0;
    P(bot, j,3) = 0;
    P(bot-1, j,2) = 0;
    P(bot-1, j,3) = 0;
    P(bot-1, j,3) = 0;
end
% imshow(I);

% imshow(P);%imresize(P,0.1));
axes(handles.axes6);
imshow(P);

if numberOfBoundaries > 0
    msgbox('have tumor');
    set(handles.edit1, 'string', 'have tumor');
else
    msgbox('no tumor');
    set(handles.edit1, 'string', 'no tumor');
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,
'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```