**TO CHECK FOR A SINGLE IMAGE**

% Specify the path to the image file

imagePath = "<Path>";

% Read the image

image = imread(imagePath);

% Convert the image to grayscale

gray\_image = im2gray(image);

% Convert the grayscale image to binary using a threshold

bw = im2bw(image, 0.6);

% Label connected components in the binary image

label = bwlabel(bw);

% Apply Wiener2 denoising to the labeled image

denoised\_img = wiener2(label, [5, 5]);

% Compute region properties (area and bounding box) for the labeled image

stats = regionprops(label, 'Area', 'BoundingBox');

area = [stats.Area];

bounding\_boxes = {stats.BoundingBox};

% Compute aspect ratios for each bounding box

aspect\_ratios = zeros(size(bounding\_boxes));

for i = 1:numel(bounding\_boxes)

aspect\_ratios(i) = bounding\_boxes{i}(3) / bounding\_boxes{i}(4);

end

% Define criteria for potential tumor regions

high\_area\_region = area > 500;

reasonable\_aspect\_ratio = aspect\_ratios > 0.5 & aspect\_ratios < 2;

potential\_tumor\_regions = high\_area\_region & reasonable\_aspect\_ratio;

% Find the maximum area among potential tumor regions

max\_area = max(area(potential\_tumor\_regions));

% Set a threshold for tumor detection

area\_threshold = 2000;

% Check if a tumor is detected based on the maximum area

if max\_area > area\_threshold

tumor\_label = find(area == max\_area);

fprintf('Tumor detected in region %d with area %d.\n', tumor\_label, max\_area);

tumor = ismember(label, tumor\_label);

else

fprintf('No tumor detected.\n');

tumor = zeros(size(label));

end

% Create a structuring element for dilation

se = strel('square', 5);

% Dilate the detected tumor region

tumor = imdilate(tumor, se);

% Display the original image, the isolated tumor region, and the detected tumor boundaries

figure;

subplot(1, 3, 1);

imshow(image, []);

title('Brain');

subplot(1, 3, 2);

imshow(tumor, []);

title('Tumor Alone');

% Find and plot boundaries of the detected tumor region on the original image

[B, L] = bwboundaries(tumor, 'noholes');

subplot(1, 3, 3);

imshow(image, []);

hold on

for i = 1:length(B)

plot(B{i}(:, 2), B{i}(:, 1), 'y', 'linewidth', 1.45);

end

title('Detected Tumor');

**TO CHECK THE TOTAL FILES OF IMAGES**

% Brain Tumor Image Processing GUI

function integratedImageProcessingGUI

% Create the main figure

mainFig = figure('Name', 'BRAIN TUMOR Scan Image Processing', 'Position', [100, 100, 1000, 600]);

% Create a button to select an image

uicontrol('Style', 'pushbutton', 'String', 'Select MRI Scan Image', 'Position', [50, 550, 200, 30], 'Callback', @selectImage);

% Display result text

resultText = uicontrol('Style', 'text', 'String', 'Result: ', 'Position', [300, 550, 200, 30]);

% Display original image axis

originalAxis = subplot(2, 3, [1, 2]);

title(originalAxis, 'Original Image', 'FontSize', 12);

% Display tumor alone axis

tumorAloneAxis = subplot(2, 3, 4);

title(tumorAloneAxis, 'Tumor Alone', 'FontSize', 12);

% Display detected tumor axis

detectedTumorAxis = subplot(2, 3, 5);

title(detectedTumorAxis, 'Detected Tumor', 'FontSize', 12);

% Callback function to select an image

function selectImage(~, ~)

% Open a dialog to select an image file

[filename, pathname] = uigetfile({'\*.png;\*.jpg;\*.jpeg;\*.tif;\*.tiff', 'Image Files (\*.png, \*.jpg, \*.jpeg, \*.tif, \*.tiff)'; '\*.\*', 'All Files'}, 'Select CT Scan Image');

% Check if the user selected a file

if isequal(filename, 0) || isequal(pathname, 0)

disp('User canceled image selection');

else

% Read the selected image

imagePath = fullfile(pathname, filename);

ctScanImage = imread(imagePath);

% Perform processing directly in the callback function

[result, tumorImage] = yourProcessingLogic(ctScanImage);

% Display the result

set(resultText, 'String', ['Result: ' num2str(result)]);

% Display the original image

imshow(ctScanImage, 'Parent', originalAxis);

% Display the tumor alone image

imshow(tumorImage, 'Parent', tumorAloneAxis);

% Display the detected tumor boundaries on the original image

[B, ~] = bwboundaries(tumorImage, 'noholes');

imshow(ctScanImage, 'Parent', detectedTumorAxis);

hold(detectedTumorAxis, 'on');

for i = 1:length(B)

plot(detectedTumorAxis, B{i}(:, 2), B{i}(:, 1), 'y', 'linewidth', 1.45);

end

hold(detectedTumorAxis, 'off');

title(detectedTumorAxis, 'Detected Tumor', 'FontSize', 12);

end

end

% Your processing logic goes here

function [result, tumorImage] = yourProcessingLogic(inputImage)

% Replace the following with your actual processing logic

% For example, you might want to calculate some statistic on the image

% Convert the input image to grayscale

grayImage = im2gray(inputImage);

% Convert the grayscale image to a binary image using a threshold of 0.6

bwImage = im2bw(inputImage, 0.6);

% Label connected components in the binary image

label = bwlabel(bwImage);

% Apply Wiener2 denoising to the labeled image with a neighborhood size of [5, 5]

denoisedImg = wiener2(label, [5, 5]);

% Extract region properties (Area) and bounding boxes

stats = regionprops(label, 'Area', 'BoundingBox');

area = [stats.Area];

% Identify regions with high area (potential tumor regions)

highAreaRegion = area > 500;

% Calculate reasonable aspect ratios based on bounding boxes

reasonableAspectRatio = [stats.BoundingBox] ./ 4;

aspectRatios = reasonableAspectRatio(3:4:end) ./ reasonableAspectRatio(4:4:end);

reasonableAspectRatio = aspectRatios > 0.5 & aspectRatios < 2;

% Identify potential tumor regions based on both high area and reasonable aspect ratio

potentialTumorRegions = highAreaRegion & reasonableAspectRatio;

% Find the region with the maximum area among potential tumor regions

maxArea = max(area(potentialTumorRegions));

% Set an area threshold for tumor detection

areaThreshold = 2000;

% Check if the maximum area exceeds the threshold for tumor detection

if maxArea > areaThreshold

% Identify the label of the detected tumor region

tumorLabel = find(area == maxArea);

% Create a binary image with only the detected tumor region

tumorImage = ismember(label, tumorLabel);

fprintf('Tumor detected in region %d with area %d.\n', tumorLabel, maxArea);

else

fprintf('No tumor detected.\n');

% Set the tumor mask to zeros if no tumor is detected

tumorImage = zeros(size(label));

end

% Apply morphological dilation to the tumor image using a square structuring element of size 5

se = strel('square', 5);

tumorImage = imdilate(tumorImage, se);

% Replace with your actual result calculation (e.g., mean intensity of the original image)

result = mean(inputImage(:));

end

end