Electrical & Electronics Engineering Department
EE434 – Biomedical Signal Processing - Final Takehome Exam
Deadline: January 11th, 2024 23:59
Instructor: M. Zübeyir Ünlü

Name: Harun Durmuş Student No: 270206025

Honor statement: *I pledge that I have not used any notes, text, or any other reference materials during this exam. I pledge that I have neither given nor received any aid from any other person during this examination, and that the work presented here is entirely my own.*

Signature:

Q1. About Image Segmentation and Related Subjects.

a) About applying the first order derivative (gradient) to find edges.

i. Plot the |gx| (magnitude of the gradient image in x direction) of the image.

ii. Plot the |gy| (magnitude of the gradient image in y direction) of the image.

iii. Plot the gradient image, |gx| + |gy|.

iv. Apply a 5x5 smoothing filter and after that apply the steps (i), (ii), and (iii) again. Plot the resulting images. Compare them with the previous ones.

b) Apply Marr-Hildreth algorithm to find edges. Plot the resulting image.

c) Apply Canny edge detection algorithm to find edges. Plot the resulting image.

d) Compare all the results.

A1. MATLAB code[1] is used for this question.

a) The Sobel filter is used in part (i) to determine the input image's gradient in the x-direction (horizontal). It represents fluctuations in intensity along the horizontal direction. The phase in (ii), like the previous one, uses the Sobel filter to determine the gradient of the input image in the y-direction (vertical). It represents intensity fluctuations in the vertical direction. Step (iii) combines the magnitudes of the gradient images computed in the previous two steps to produce a composite gradient image that includes both horizontal and vertical edges.

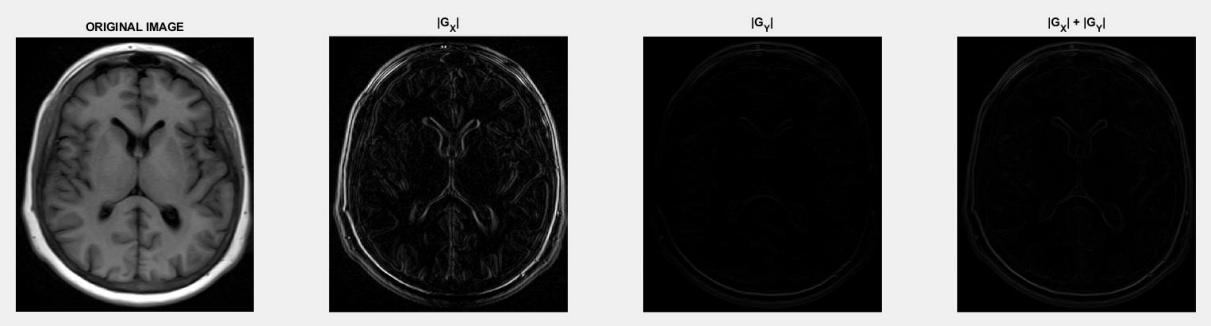

Figure 1: *Applying first order derivative (gradient) to find edges*

To reduce noise, the input image is smoothed using a 5x5 filter. After smoothing, steps (i), (ii), and (iii) are repeated to compute the gradient images of the smoothed image. This helps to reduce noise and improve edges.

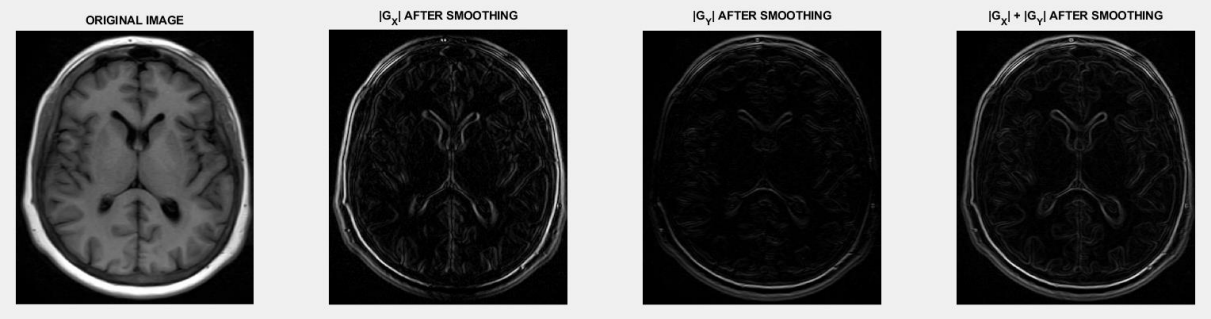

Figure 2: *Applying 5x5 smoothing filter*

b) The Marr-Hildreth algorithm is used on the original image. The image is convolved with a Laplacian of Gaussian (LoG) filter to detect edges. The zero-crossings in the filtered image are utilized to determine edges.

- **Smoothing (Gaussian) Filtering:** The first step of the Marr-Hildreth algorithm is to apply Gaussian smoothing to the input image. This reduces noise in the image and prepares it for edge detection. The image is smoothed by applying a Gaussian filter.
- **Laplacian of Gaussian (LoG) Filtering**: After smoothing, the algorithm applies the Laplacian operator on the resulting image. The Laplacian operator computes the image's second derivative, which identifies places with rapid intensity fluctuations. Applying the Laplacian operator to the smoothed image yields the Laplacian of Gaussian (LoG) image.
- **Zero-Crossing Detection:** The LoG image is analyzed for zero-crossings, which occur when the gradient's sign changes. A zero-crossing represents the presence of an edge. Moving from a positive to a negative value (or vice versa) in the LoG image represents an edge. These zero-crossings are probable edge sites.
- **Thresholding:** After detecting zero-crossings, the LoG image is thresholded to separate edges from noise. Any zero-crossing with a magnitude greater than a predetermined threshold is termed an edge pixel.
- **Edge Localization:** To refine the edges, the algorithm can conduct edge localization, which locates the edge precisely inside a neighborhood around each detected zero-crossing.

c) The grayscale image is processed using the Canny edge detection algorithm. Edge detection is accomplished using a multi-stage procedure that includes Gaussian smoothing, gradient calculation, non-maximum suppression, and edge tracking via hysteresis.

- **Gaussian Smoothing:** The Canny edge detection approach, like the Marr-Hildreth algorithm, starts with Gaussian smoothing of the input image. This process reduces noise and prepares the image for edge detection.
- **Gradient Calculation:** Canny computes the picture gradient utilizing methods such as the Sobel and Prewitt operators. This stage determines the gradient magnitude and direction for each pixel in the image.

- **Non-Maximum Suppression:** After calculating the gradient, non-maximum suppression is applied. This entails assessing each pixel's gradient magnitude and comparing it to adjacent pixels along the gradient direction. Only the pixels with the largest gradient magnitude in their local neighborhood are kept; the rest are suppressed.
- **Edge Tracking by Hysteresis:** Canny employs hysteresis to join edge pixels into continuous curves. It entails establishing two thresholds: a high and low threshold. Pixels with gradient magnitudes greater than the high threshold are designated strong edge pixels. Pixels that fall between the high and low thresholds are considered weak edge pixels. To create continuous edges, the algorithm tracks and joins weak edge pixels with strong ones.
- **Edge Thin-Out:** Finally, the algorithm can use edge thinning to further refine the discovered edges. This stage removes isolated or errant edge pixels, resulting in cleaner and more accurate edge maps.

To summarize, the Marr-Hildreth algorithm uses the image's second derivative to discover edges, but the Canny approach uses gradient information and hysteresis to detect edges with greater resilience and noise tolerance. Both techniques have advantages and disadvantages, making them appropriate for various edge detection circumstances.

Figure 3: *Comparison of Marr-Hildreth and Canny Algorithm*

d) The performance and accuracy of various edge detection algorithms are determined by the image's specific properties as well as its noise level. There is no one-size-fits-all solution, and the method used should be determined by the application's requirements. The first-order derivative (gradient) approach is susceptible to noise but gives an easy way to find edges. Marr-Hildreth produces smoother results, but it is still sensitive to noise and requires careful parameter tweaking. Canny edge detection is a reliable and frequently used method that balances noise reduction and accurate edge recognition, making it ideal for a variety of scenarios. In the supplied visual comparison, Canny edge detection looks to produce cleaner and more linked edges, making it a better choice for many edge detection jobs. However, the decision should ultimately be dependent on the application's specific requirements and the qualities of the input image.

Q2. About Image Segmentation and Related Subjects.

Using Otsu's method find the optimum thresholds to segment out this image into different subregions. To find the optimum number of subregions according to the image given, try different numbers. After that show each subregion in a separate image.

A2. MATLAB code[2] is used for this question.

Otsu's approach is a methodology for automatic picture thresholding that divides an image into two or more classes (foreground and background) based on pixel intensities. Otsu's method seeks to identify an ideal threshold that maximizes the separation between these classifications.

Figure 4: *Segmentation of the image by Otsu's method*

The method uses a histogram of pixel intensities in the image, with each bin representing the number of pixels with a specific intensity value. The algorithm operates as follows:

- **Computing Histogram:** Histogram of pixel intensities in the grayscale image is created. This histogram depicts the distribution of pixel values throughout the image.
- **Normalizing Histogram:** Histogram values are normalized so they add up to one. This step is required to ensure that the approach is consistent throughout image size and intensity range.

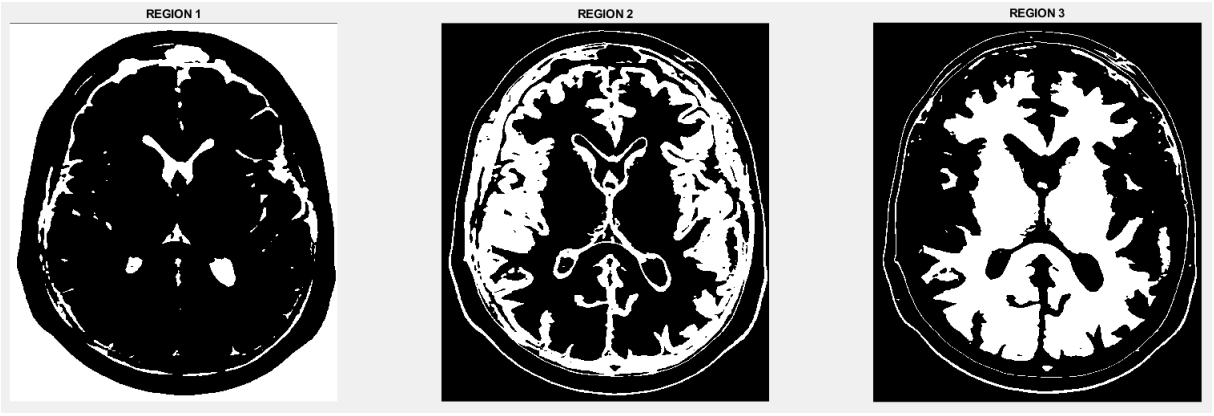

Figure 5: *Each subregion that separates the image*

- **Threshold Search:** All of the possible threshold values in the histogram are iterated, from the lowest to the highest. Each threshold value separates the image into two classes: pixels with intensities less than the threshold (Class A) and pixels with intensities equal to or greater than the threshold (Class B). Choosing the threshold that produces the greatest separation between the classes and using the chosen threshold to binarize the original image is the final stage of the method. Pixels with intensities less than the threshold are assigned to one class, while pixels with intensities equal to or greater than the threshold are assigned to another class.

To conclude everything that have been stated so far, Otsu's approach is used in the code to automatically calculate threshold values for image segmentation. The code displays binary segmentation as well as multilevel thresholding, which allows us to divide the image into multiple subregions based on intensity levels. This method can be beneficial in a variety of image processing applications, including object detection and ROI extraction.

Figure 6: *Smoothed and segmented image with Otsu's method*

Figure 7: *Each smoothed subregion that separates the image*

The comparison of picture segmentation using Otsu's method with and without a smoothing filter demonstrates that applying a smoothing filter before segmentation improves noise reduction and produces cleaner results, potentially leading to more consistent and coherent

segmentation outcomes. However, this method may slightly blur or smooth out finer visual features, reducing sensitivity to high-frequency structures. In contrast, segmentation without a smoothing filter preserves finer picture details but is subject to noise in the source image, potentially resulting in fragmented or noisy segmentation, particularly in noise-prone regions. The decision between these approaches is determined by the specific image properties and the necessary balance of noise reduction and retention of essential image features for the current image processing task.

Q3. About Image Segmentation and Related Subjects.

Apply k-means clustering to the image. Try at least three different k's. Which one is appropriate for the given image? Show the resulting images.

A3. MATLAB code[3] is used for this question.

K-means clustering is a popular unsupervised machine learning approach that divides data into K unique, non-overlapping clusters or groupings. In image segmentation, K-means clustering can be used to group pixels based on their similarity in intensity or color. The algorithm operates as follows:

- **Initialization:** K initial cluster centers (centroids) are chosen at random and reflect the cluster centers' initial predictions.
- **Assignment:** Each data point (pixel in the image) is assigned to the nearest centroid using a similarity metric, commonly Euclidean distance.
- **Update Centroids:** The centroids are computed using the average of all data points allocated to each cluster.
- **Iteration:** The assignment and centroid update processes are repeated until convergence or a predetermined number of iterations is reached.
- **Result:** Following convergence, the algorithm generates K clusters and assigns each pixel to one of them, thus segmenting the image into K sections.

The attached MATLAB code performs K-means clustering on an input image with three distinct K values (k = 3, 9, and 15). Here's a brief comparison of the findings:

- **k = 3:** When K is set to 3, the image is divided into three clusters that may indicate important regions or structures within the image, such as the background, soft tissues, or bones.
- **k = 9:** Increasing K to 9 creates more clusters, resulting in sharper image segmentation. This can catch more details and different pixel intensities.
- **k = 15:** Setting K to 15 further separates the image into smaller clusters, resulting in finer-grained segmentation. This may record very tiny variations in pixel intensity.

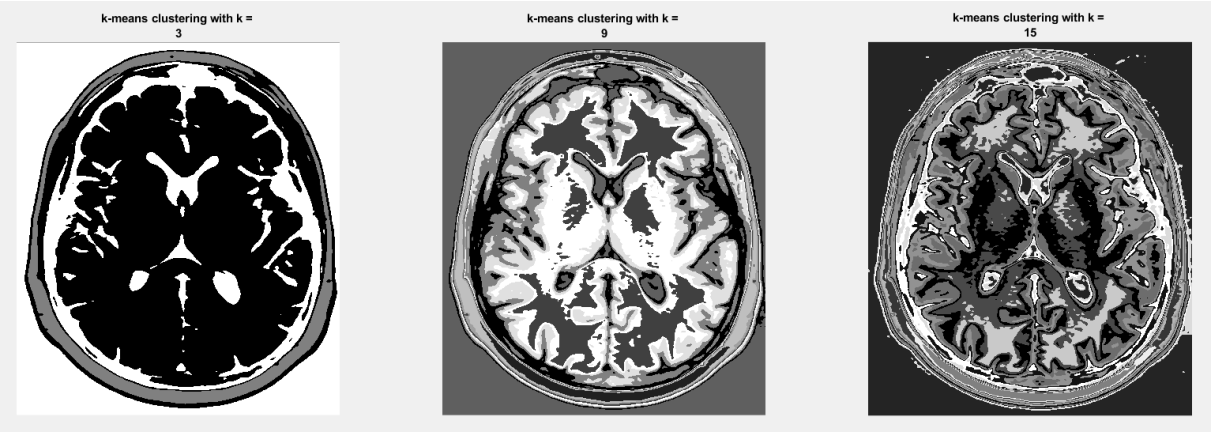

Figure 8: *K-means clustering with different k values*

The suitable K for a given image is determined by its complexity and the level of detail required for segmentation. In practice, choosing the appropriate K frequently necessitates a trade-off between over-segmentation (too many regions) and under-segmentation. The selection of K should be based on the specific analytic objectives and the required level of granularity in the segmentation outcomes.

K-means clustering is a versatile picture segmentation approach that allows you to experiment with multiple K values to achieve the required amount of detail in segmented regions. The appropriateness of K is determined by the image's complexity and the specific goals of the segmentation operation.

Q4. About Medical Image Registration.

Using only the rotation and translations in x and y please register Head-CT-Transformed.png to Head-CT.png. At the end please show the resulting image (call it as Head-CT-Registered.png) and the difference image (call it as Head-CT-Difference.png). Please also give the values for rotation in degree, and translations in x and y to register Head-CT-Transformed.png to Head-CT.png. (To find the optimum ones you should try different values. It is not necessary for all of you to have the same values. There may be slight differences.)

A4. MATLAB code[4] is used for this question.

In the realm of biomedical imaging, image modification and registration are essential. Medical practitioners and researchers frequently use a variety of imaging modalities, each of which offers unique insights into a patient's condition. To fully realize the potential of these modalities and acquire a holistic knowledge, it is necessary to align and fuse data from these many sources. Here's where picture registration comes in. It serves as a bridge, ensuring that images from many modalities use the same coordinate system, allowing for meaningful comparisons and complete analyses. Furthermore, in medical research and clinical practice, longitudinal studies spanning numerous time points are prevalent, making it critical to synchronize images acquired at different times. Moreover, during surgical procedures, image registration allows for exact navigation by overlaying preoperative images on real-time patient anatomy, guiding surgeons' movements. Furthermore, proper picture alignment is critical in interventional radiology and radiation treatment to ensure accurate instrument placement and radiation administration. In essence, image registration is the cornerstone that allows medical practitioners and researchers to make informed decisions and give the best treatment to patients.

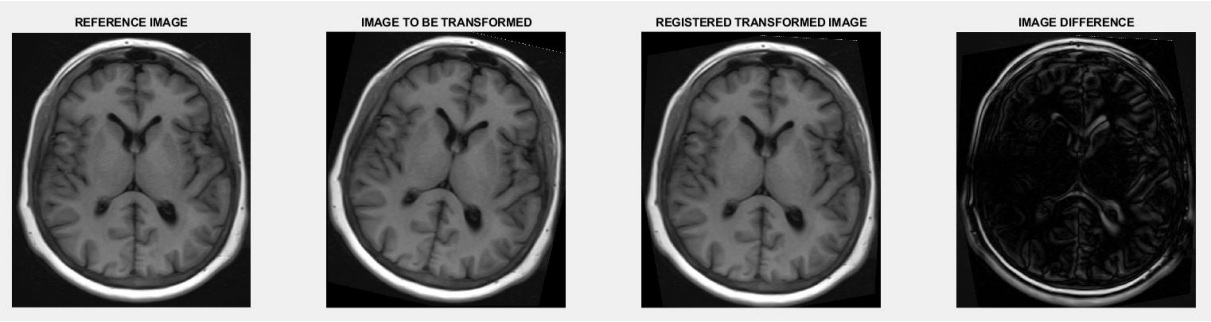

Figure 9: *Transformation with test parameters*

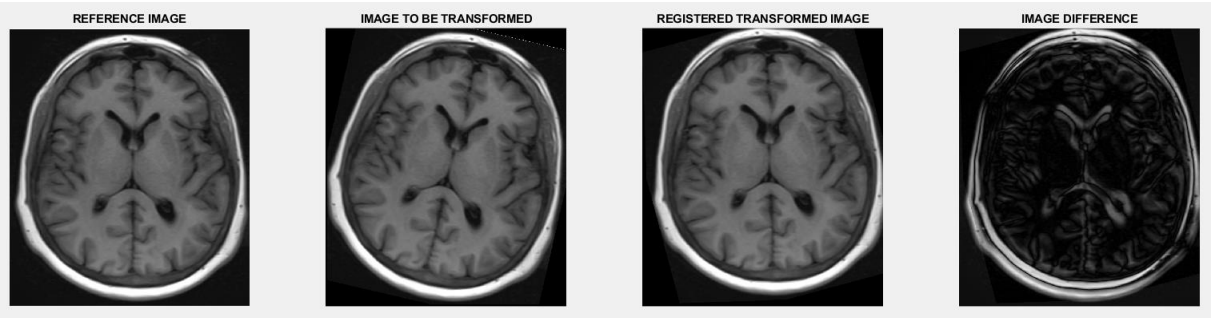

Figure 10: *Trying for more accuracy*

After testing different parameters, it is observed that more accurate results have been achieved with the optimum parameters that is obtained by using the algorithm in the code.

```
Optimum parameters for accurate transformation:
Rotation: 13.01 degrees
Translation X: -100.08 pixels
Translation Y: 113.36 pixels
```

Figure 11: *Obtaining parameters for accurate transformation*

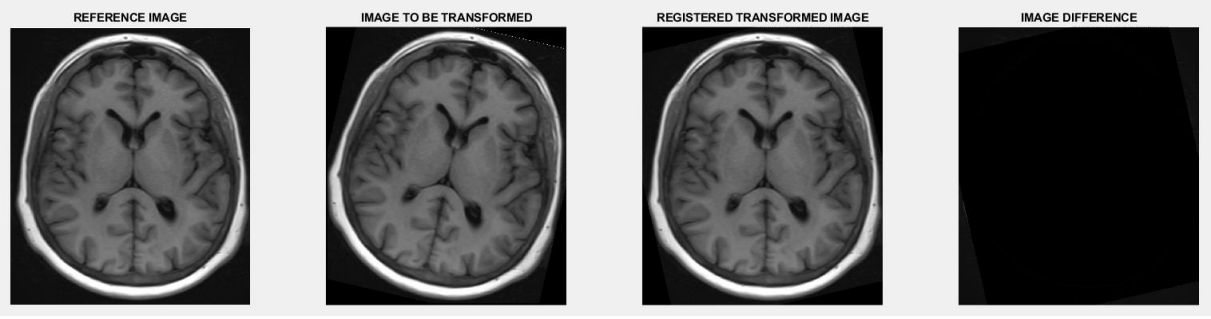

Figure 12: *Accurate Transformation*

References

- Lecture Notes (9 and 10)

Appendix

[1] MATLAB code "F1.m"

```
clc;
clear all;
close all;
%% QUESTION 1
%%
I = imread("Head-CT.png"); % reading image file
if size(I,3) == 3
    graycoded_I = rgb2gray(I); % converting rgb to gray code
else
    graycoded_I = I;
end
%% step a
%% substep i, ii, iii
[G_x, G_y] = gradient(double(graycoded_I)); % obtaining gradient_x and gradient_y
abs_Gx = abs(G_x);
abs_Gy = abs(G_y);
gradient_I = abs_Gx + abs_Gy;
%% substep iv
filter_smooth = fspecial("average", [5 5]); % smooth filter
I_smooth = imfilter(graycoded_I, filter_smooth, "replicate");
[Gx_smooth, Gy_smooth] = gradient(double(I_smooth));
abs_Gx_smooth = abs(Gx_smooth);
abs_Gy_smooth = abs(Gy_smooth);
gradient_I_smooth = abs_Gx_smooth + abs_Gy_smooth;
%% step b
%%
I_marr = im2double(I);
% smoothening the image with a filter
filter_marr = [0 0 1 0 0; 0 1 2 1 0; 1 2 -16 2 1; 0 1 2 1 0; 0 0 1 0 0];
filtered_I = conv2(I_marr,filter_marr);
% finding the zero crossings
[row,column] = size(filtered_I);
zerocross = zeros([row,column]);
for i=2:row-1
    for j=2:column-1
        if (filtered_I(i,j)>0)
             if (filtered_I(i,j+1)>=0 && filtered_I(i,j-1)<0) ||
(filtered_I(i,j+1)<0 && filtered_I(i,j-1)>=0)

                zerocross(i,j)= filtered_I(i,j+1);

            elseif (filtered_I(i+1,j)>=0 && filtered_I(i-1,j)<0) ||
(filtered_I(i+1,j)<0 && filtered_I(i-1,j)>=0)
                    zerocross(i,j)= filtered_I(i,j+1);
            elseif (filtered_I(i+1,j+1)>=0 && filtered_I(i-1,j-1)<0) ||
(filtered_I(i+1,j+1)<0 && filtered_I(i-1,j-1)>=0)
                  zerocross(i,j)= filtered_I(i,j+1);
            elseif (filtered_I(i-1,j+1)>=0 && filtered_I(i+1,j-1)<0) ||
(filtered_I(i-1,j+1)<0 && filtered_I(i+1,j-1)>=0)
                  zerocross(i,j)=filtered_I(i,j+1);
            end
```

```
        end

    end
end
output = im2uint8(zerocross);
% thresholding
output_thresholded = output > 10;
%% step c
%%
canny_edge = edge(graycoded_I, "canny");
%% figuring the results
%%
figure (1)

subplot(141)
imshow(graycoded_I, []);
title("ORIGINAL IMAGE");

subplot(142)
imshow(abs_Gx, []);
title("|G_X|");

subplot(143)
imshow(abs_Gy, []);
title("|G_Y|");

subplot(144)
imshow(gradient_I, []);
title("|G_X| + |G_Y|");

figure(2)

subplot(141)
imshow(graycoded_I, []);
title("ORIGINAL IMAGE");

subplot(142)
imshow(abs_Gx_smooth, []);
title("|G_X| AFTER SMOOTHING");

subplot(143)
imshow(abs_Gy_smooth, []);
title("|G_Y| AFTER SMOOTHING");

subplot(144)
imshow(gradient_I_smooth, []);
title("|G_X| + |G_Y| AFTER SMOOTHING");

figure(3)

subplot(131)
imshow(graycoded_I, []);
title("ORIGINAL IMAGE");

subplot(132)
imshow(output_thresholded, []);
title("MARR-HILDRETH");
```

```
subplot(133)
imshow(canny_edge, []);
title("CANNY");
```

[2] MATLAB code “F2.m”

```
clc;
clear all;
close all;
%% QUESTION 2
%%
I = imread('Head-CT.png');
graycoded_I = I;
level = graythresh(graycoded_I); % finding optimum threshold by Otsu's method
blacknwhite = imbinarize(graycoded_I, level);

figure(1)
imshow(blacknwhite);
title("SEGMENTATION OF THE IMAGE BY OTSU'S METHOD");

levels = multithresh(graycoded_I, 5);
segmented_I = imquantize(graycoded_I, levels);

figure(2)

subplot(131)
imshow(segmented_I == 1)
title("REGION 1");

subplot(132)
imshow(segmented_I == 2)
title("REGION 2");

subplot(133)
imshow(segmented_I == 3)
title("REGION 3");

I_smooth = imgaussfilt(graycoded_I, 5); % Gaussian filter with sigma value 5

level_smooth = graythresh(I_smooth);
blacknwhite_smooth = imbinarize(I_smooth, level_smooth);

figure(3)
imshow(blacknwhite_smooth);
title("SMOOTHED AND SEGMENTED IMAGE WITH OTSU'S METHOD");

levels_smooth = multithresh(I_smooth, 2);
segmented_I_smooth = imquantize(I_smooth, levels_smooth);

figure(4)

subplot(131)
imshow(segmented_I_smooth == 1)
title("SMOOTHED REGION 1");

subplot(132)
imshow(segmented_I_smooth == 2)
title("SMOOTHED REGION 2");
```

```
subplot(133)
imshow(segmented_I_smooth == 3)
title("SMOOTHED REGION 3");
```

[3] MATLAB code “F3.m”

```
clc;
clear all;
close all;
%% QUESTION 3
%%
I = imread("Head-CT.png");
if size(I,3) == 3
    graycoded_I = rgb2gray(I); % converting rgb to gray code
else
    graycoded_I = I;
end
reshaped_I = reshape(graycoded_I, [], 1);
k = [3, 9, 15]; % k values to be tried
figure;
for i = 1:length(k)
    [cluster_I_dx, ~] = kmeans(double(reshaped_I), k(i), "Distance",
"sqEuclidean", "Replicates", 3);
    clustered_I = reshape(cluster_I_dx, size(graycoded_I));
    max_value = max(clustered_I(:));
    clustered_I_scaled = uint8(clustered_I * (255 / max_value));

    subplot(1,3,i)
    imshow(clustered_I_scaled, []), title(["k-means clustering with k = ",
num2str(k(i))]);
end
```

[4] MATLAB code “F4.m”

```
clc;
clear all;
close all;
%% QUESTION 4
%%
reference = imread('Head-CT.png'); % original image
changing = imread('Head-CT-Transformed.png'); % image to be transformed
%% registration
[optimizer, metric] = imregconfig('monomodal');
transform = imregtform(changing, reference, 'rigid', optimizer, metric);
registered = imwarp(changing, transform, 'OutputView', imref2d(size(reference)));
%% difference calculation
difference = imabsdiff(reference, registered);
%% saving images
imwrite(registered, 'Head-CT-Registered.png');
imwrite(difference, 'Head-CT-Difference.png');
%% extracting the transformation parameters
rotation = rad2deg(asin(transform.T(2,1))); % Convert radians to degrees
translation_x = transform.T(3,1);
translation_y = transform.T(3,2);
fprintf('Optimum parameters for accurate transformation:\nRotation: %.2f
degrees\n', rotation);
fprintf('Translation X: %.2f pixels\n', translation_x);
fprintf('Translation Y: %.2f pixels\n', translation_y);
%% applying transformation
```

```
%% figure 9
% rotation = 10;
% translation_x = -80;
% translation_y = 100;
%% figure 10
% rotation = 15;
% translation_x = -100;
% translation_y = 110;
%%
transform = affine2d([cosd(rotation) -sind(rotation) 0; sind(rotation)
cosd(rotation) 0; translation_x translation_y 1]);
changing_registered = imwarp(changing, transform, 'OutputView',
imref2d(size(reference)), 'SmoothEdges', true);
difference = imabsdiff(reference, changing_registered);
%% figuring results
figure(1)

subplot(141);
imshow(reference);
title('REFERENCE IMAGE');

subplot(142);
imshow(changing);
title('IMAGE TO BE TRANSFORMED');

subplot(143);
imshow(changing_registered);
title('REGISTERED TRANSFORMED IMAGE');

subplot(144)
imshow(difference);
title('IMAGE DIFFERENCE');
```