Umm Al-Qura University
College of computers
Department of CS and AI

Image processing
1st Semester 2024
Course Project

IMAGE ENHANCEMENT FOR MEDICAL RADIOLOGY IMAGES

Image processing Course

2024- 2025

GROUP MEBMERES:

NABA'A ABDUL RAHMAN
ID: 443013099
FATMA AL-ZAHRANI
ID:444006628
AMAL AL-ZAHRANI
ID: 444002268
ASAL AL-HASSEN
ID: 444001787
FIDA'A FELEMBAN
ID:444006581

INSTRCTUOR :

Dr. Manal Alghamdi

1.PROJECT IDEA AND PROJECT AIMS

This project involves developing a Medical Radiology Image Processing Tool with a MATLAB-based Graphical User Interface (GUI) to enhance the clarity of medical images and support analysis. Users can either select one of five pre-loaded images or upload their own scans, offering flexibility for different imaging needs.

The tool provides key enhancement techniques such as **Gaussian filtering** to reduce noise, **histogram equalization** to enhance contrast, and **sharpening** to highlight fine details like small lesions or micro-calcifications. These operations can be applied individually or combined to optimize the image for better analysis.

The tool also includes other features. **Edge detection** identifies boundaries, such as tumors or organ edges, while **inversion reverses** pixel intensities, offering alternative visual perspectives. **Semantic segmentation** divides the image into distinct regions, like background, tissue, or abnormal growths, helping radiologists isolate areas of interest for focused diagnostics. Color-coded segmentation enhances clarity and makes abnormalities easier to spot.

The GUI provides a side-by-side comparison of original and processed images, ensuring radiologists can evaluate transformations without losing critical information. Additionally, histograms of both images offer further insights into contrast changes. Users receive real-time feedback on their selected operations, and the program offers the flexibility to restart with a new image or exit the session.

This project aims to streamline image processing for medical professionals, enabling efficient enhancement and analysis of radiology images. By offering a range of tools in an easy-to-use interface, it supports more precise diagnosis, ultimately contributing to better patient outcomes.

2. PROJECT TECHNIQUES

1.GAUSSIAN FILTERING:

Purpose: This technique is used to reduce noise by smoothing pixel intensity variations. It ensures that subtle details remain intact while eliminating unnecessary distortions, which is crucial for medical images where small changes can be diagnostically significant.

2.HISTOGRAM EQUALIZATION:

Purpose: This technique improves image contrast by redistributing pixel intensities, making hidden features more visible. It is particularly useful when different tissues or structures have similar intensities, helping radiologists distinguish between them more effectively.

3.SHARPENING:

Purpose: Sharpening enhances edges and emphasizes fine details, such as small lesions or micro-calcifications. This feature aids in identifying abnormalities that may not be easily visible in the original scan, contributing to early diagnosis.

4.EDGE DETECTION (CANNY METHOD):

Purpose: This technique identifies boundaries by detecting abrupt changes in pixel intensity. It finds edges through gradient computation, retains the strongest ones using non-maximum suppression, and ensures continuity with hysteresis thresholding. This process reveals the shape and extent of abnormalities, aiding in precise diagnosis.

2. PROJECT TECHNIQUES

5.INVERSION (NEGATIVE TRANSFORMATION):

Purpose: This technique reverses pixel intensities, offering an alternative view of anatomical structures. The inverted perspective can provide more details, making certain features more distinguishable and aiding in better diagnosing.

6.SEMANTIC SEGMENTATION:

Purpose: This is the final step in the programThe purpose of this method is to segment an image using the active contour technique, which identifies important boundaries within the image. The image is first converted to grayscale, and a binary threshold is applied to initialize the contour. Then, the active contour algorithm runs to accurately trace the boundaries of significant regions, producing a segmented image where the desired boundaries are highlighted in red.s.

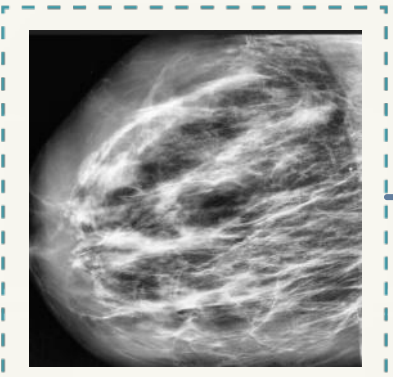
`

`

3. PROJECT FLOW

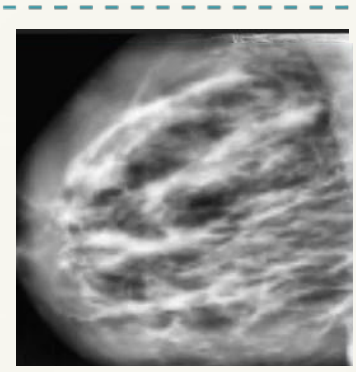
Process Summary: **The project allows users to enhance medical images by choosing from Gaussian filtering, histogram equalization, and sharpening, tailoring the process to the image's needs. For instance, noisy images benefit from Gaussian filtering, while low-contrast images require histogram equalization. These options ensure flexibility based on the input image. The program generates four possible outcomes, illustrated through diagrams showing each combination. After preprocessing, edge detection, inversion, and segmentation are automatically applied to ensure accurate boundary and region identification.**

SCENARIO1:(USER CHOSES TO PERFORM GAUSSIAN ONLY)



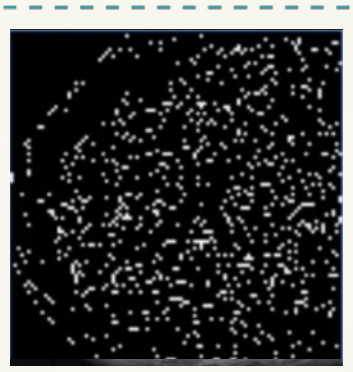
STEP 1 (INPUT)

IMAGE IS CHOSEN



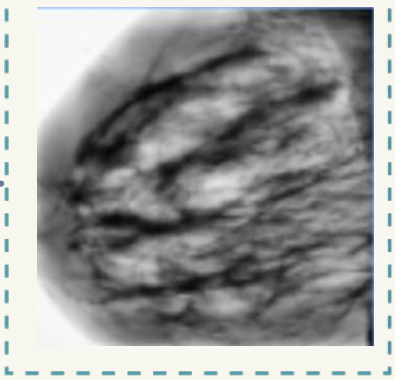
STEP 2

GAUSSIAN FILTER IS APPLIED



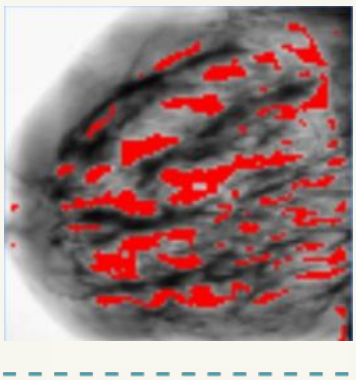
STEP 3

EDGE DETECTION (CANNY) IS APPLIED



STEP 4

INEVTRING METHOD APPLIED

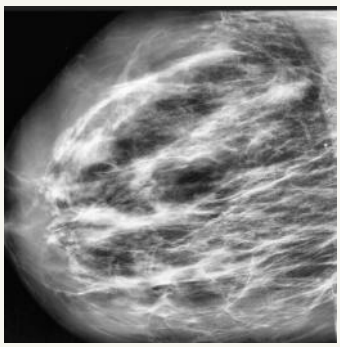


STEP 5 (OUTPUT)

SEGMENTAION METHOD APPLIED

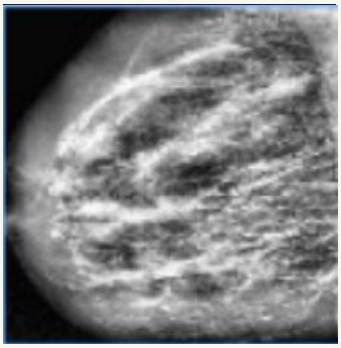
3. PROJECT FLOW

SCENARIO2: (USER CHOSES TO PERFORM SHARPENING ONLY)



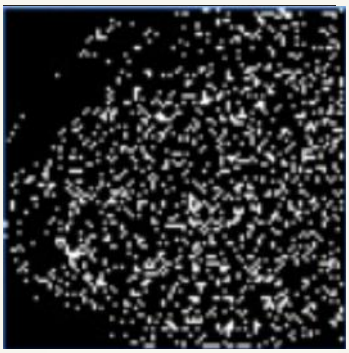
STEP 1 (INPUT)

IMAGE IS CHOSEN



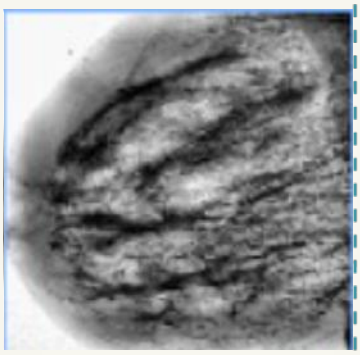
STEP 2

SHARPPING IS APPLIED



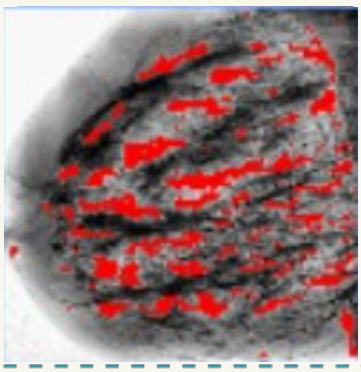
STEP 3

EDGE DETECTION (CANNY)
IS APPLIED



STEP 4

INEVTRING METHOD APPLIED

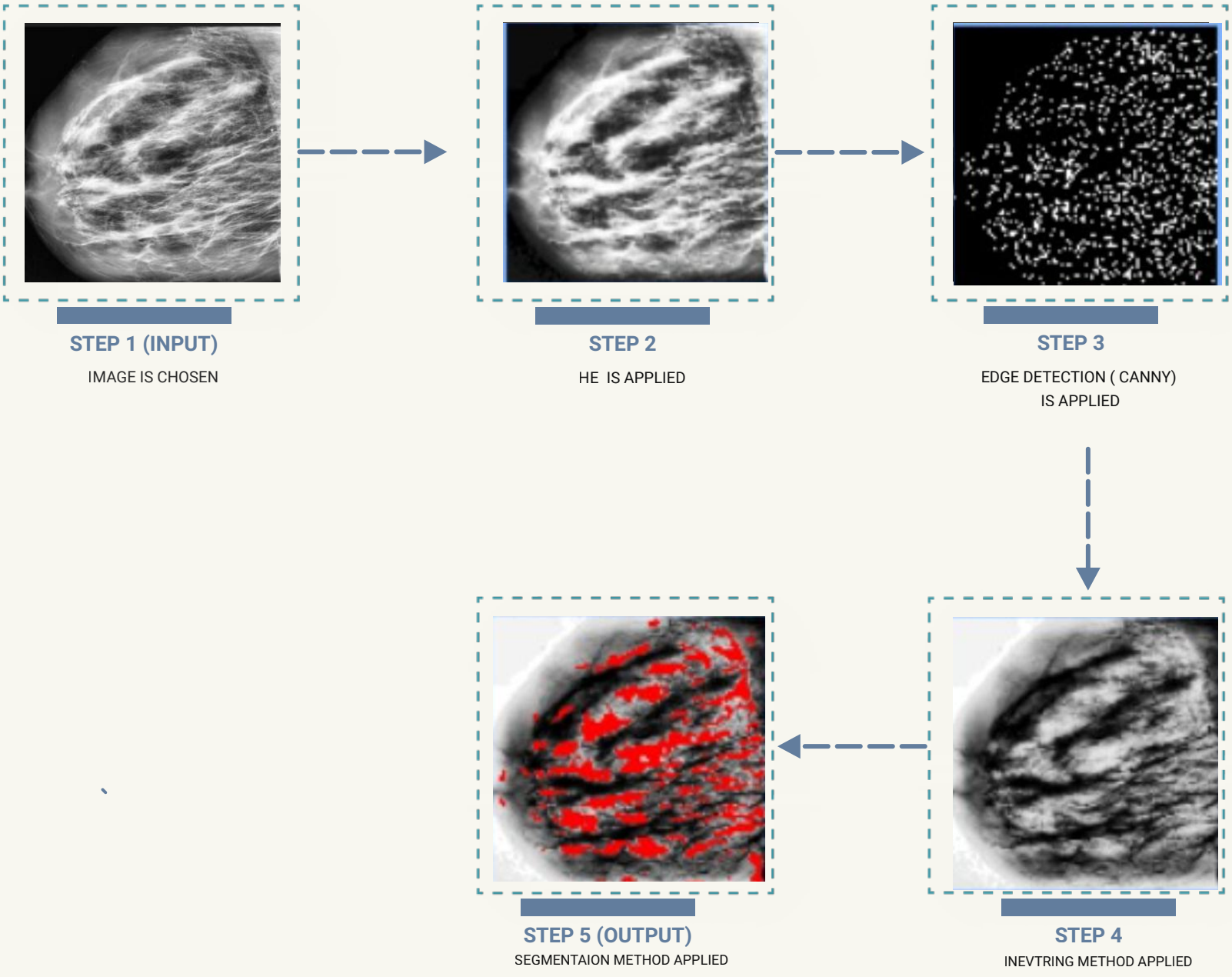


STEP 5 (OUTPUT)

SEGMENTAION METHOD APPLIED

3. PROJECT FLOW

SCENARIO3: (USER CHOSES HISTOGRAM EQUALIZATION ONLY)



STEP 1 (INPUT)

IMAGE IS CHOSEN

STEP 2

HE IS APPLIED

STEP 3

EDGE DETECTION (CANNY) IS APPLIED

STEP 4

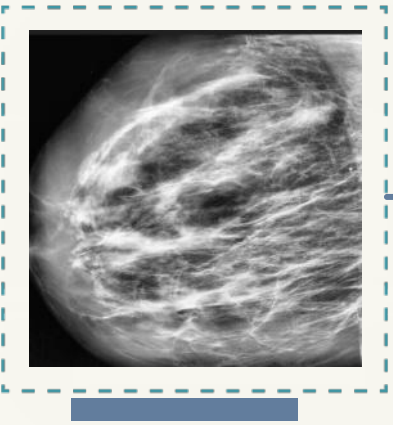
INEVTRING METHOD APPLIED

STEP 5 (OUTPUT)

SEGMENTAION METHOD APPLIED

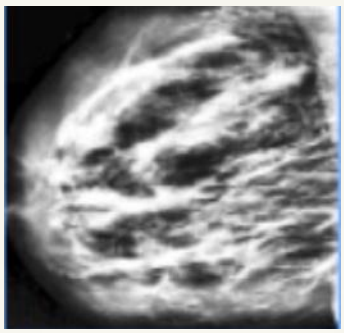
3. PROJECT FLOW

SCINARIO4: (USER CHOSES TO PERFORM ALL METHOEDS)



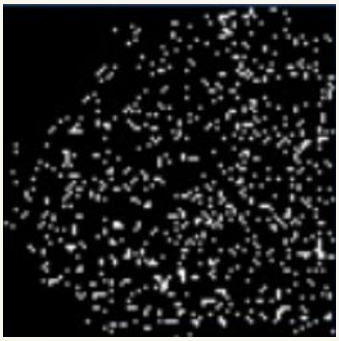
STEP 1 (INPUT)

IMAGE IS CHOSEN



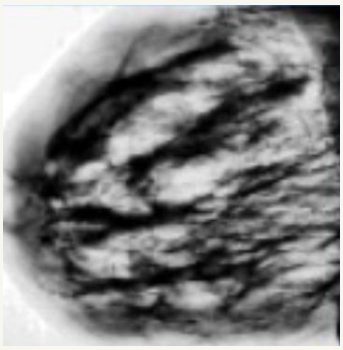
STEP 2

ALL 3 METHODS APLLIED



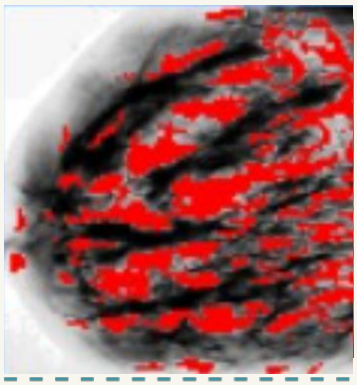
STEP 3

EDGE DETECTION (CANNY)
IS APPLIED



STEP 4

INEVTRING METHOD APPLIED



STEP 5 (OUTPUT)

SEGMENTAION METHOD APPLIED

3. PROJECT FLOW

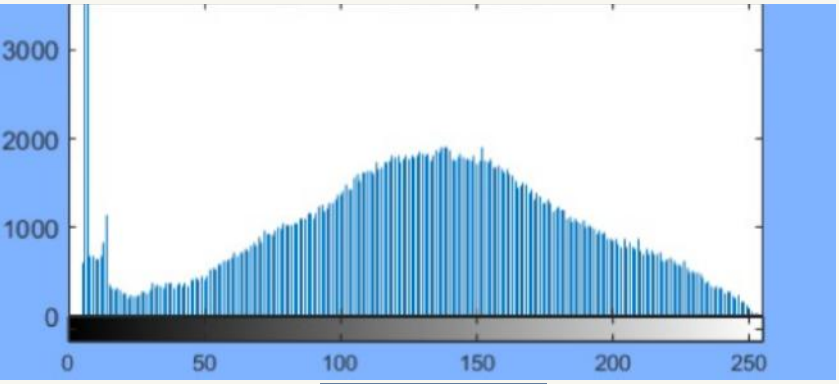
CONCLUSION FOR THE PERVIOUS 4 SCENARIOS:

- **The results CAN SHOW A SMALL VARIATION based on the selected enhancement techniques, highlighting the importance of user control to adapt the processing to different image qualities and diagnostic needs.**
- **Gaussian filtering smooths the image but may reduce detail visibility.**
- **Sharpening emphasizes small features but may introduce noise if applied alone.**
- **Histogram equalization improves contrast, making subtle differences more visible.**
- **The combination of all three techniques offers the most comprehensive enhancement, but in some cases, a more focused approach may be preferable based on the diagnostic context.**

`

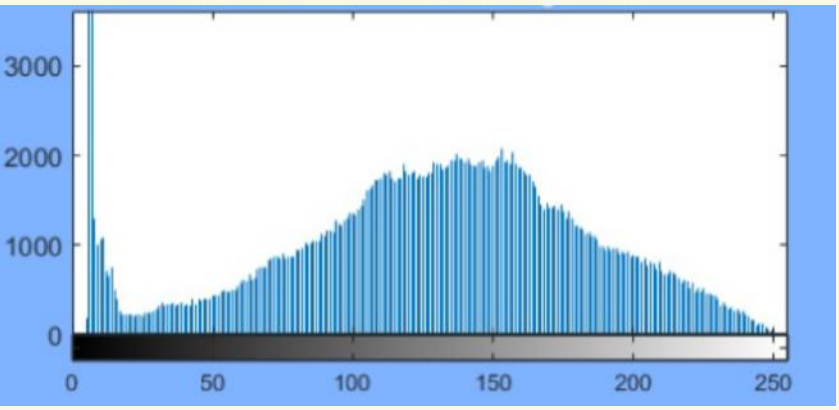
`

4. HISTOGRAM ANALYZATION



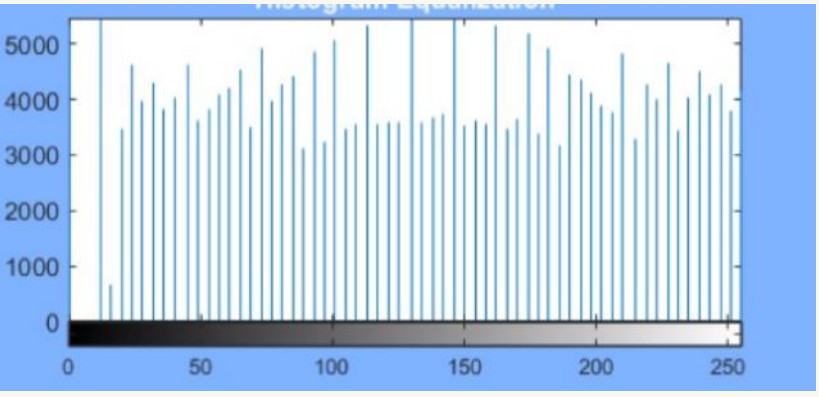
HISTOGRAM 1

ORIGINAL IMAGE



HISTOGRAM 2

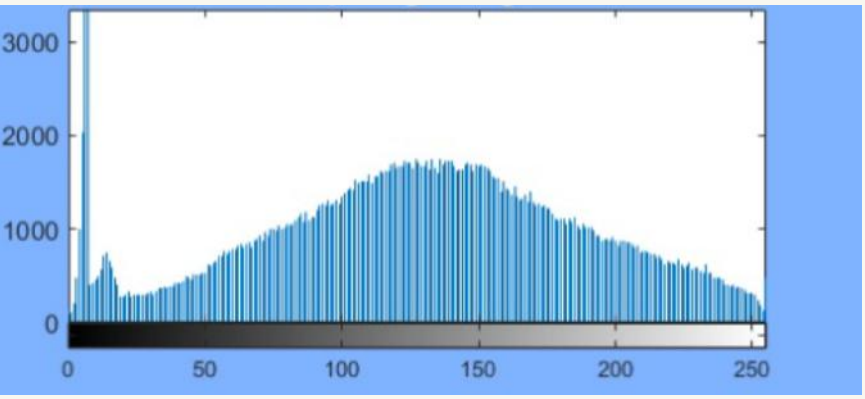
IMAGE AFTER APPLYING GAUSSIAN



HISTOGRAM 3

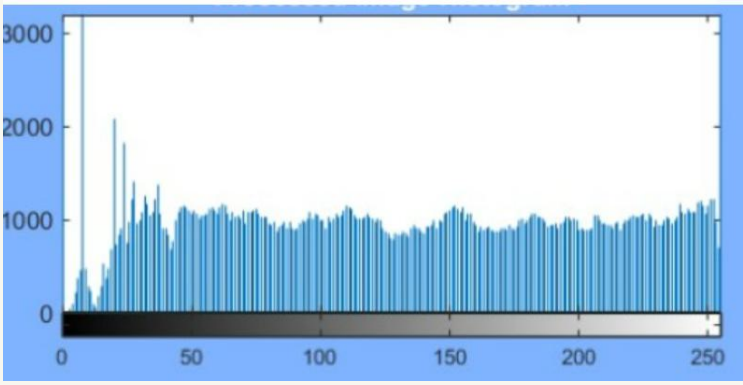
IMAGE AFTER APPLYING HISTOGRAM EQUALIZTION

4. HISTOGRAM ANALYZATION



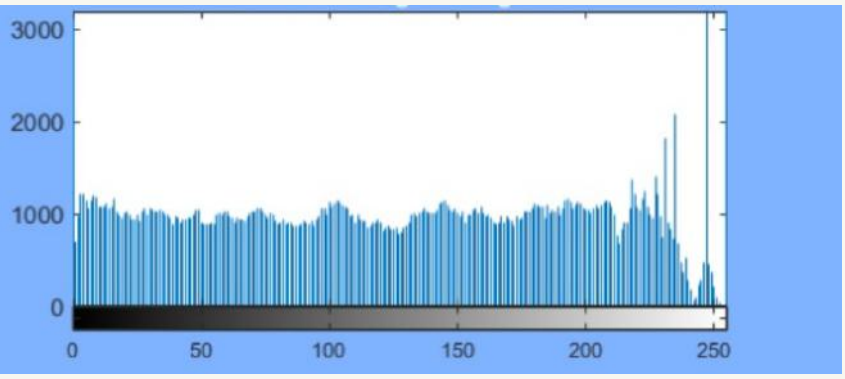
HISTOGRAM 4

SHARPENED IMAGE



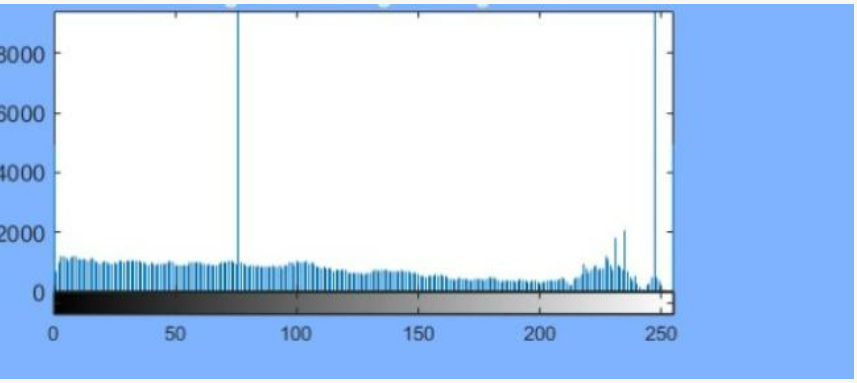
HISTOGRAM 7

IMAGE AFTER APPLYING:
HISTOGRAM EQUALIZTION +
GAUSSIAN + SHARPINING



HISTOGRAM 5

INVERTED IMAGE



HISTOGRAM 6

Segmented Image

4. HISTOGRAM ANALYZATION

HISTOGRSAM 1

This histogram presents the pixel intensity distribution of the unprocessed image. The x-axis represents pixel intensity values (ranging from 0 to 255 for an 8-bit grayscale image), while the y-axis displays the frequency of pixels at each intensity level.

The histogram reveals a bell-shaped curve with a moderately distributed spread of intensities. The distribution leans slightly toward mid-to-dark values, suggesting a balanced yet non-uniform intensity range. The uneven peaks and valleys indicate potential variations in brightness and contrast across different

HISTOGRAM 2

This histogram presents the pixel intensity distribution after applying a Gaussian filter. The distribution has smoothed out compared to the original, as the Gaussian filter blurs the image by reducing high-frequency noise and detail. Consequently, the histogram appears less jagged, with more gradual transitions between intensity levels. This smoothing effect highlights the filter's ability to reduce noise, resulting in a softer image with a slightly narrower range of intensity variations.

The transformation in the histogram reveals a narrower peak, confirming the Gaussian filter's role in reducing contrast and averaging the intensity distribution. This adjustment helps minimize unwanted noise, which is

`

4. HISTOGRAM ANALYZATION

HISTOGRSAM 3

This histogram presents the pixel intensity distribution after applying histogram equalization, revealing a distinct transformation. Histogram equalization redistributes intensity values to enhance contrast by spreading them more evenly across the available range.

The histogram now exhibits a more uniform appearance, with pixel intensities distributed more evenly from 0 to 255. This even distribution enhances contrast by shifting pixel values into underrepresented intensity regions, making dark areas darker and bright areas brighter. Consequently, this transformation improves the visibility of details, particularly in images with poor initial contrast.

HISTOGRAM 4

This histogram presents the pixel intensity distribution after applying a sharpening filter. Sharpening enhances edges and increases contrast at boundaries within the image.

The histogram closely resembles the original distribution, with a more pronounced peak in the mid-tones. However, sharper transitions at certain intensity levels indicate that edges and fine details are now more distinct. The sharpening filter amplifies the contrast between adjacent pixels with varying intensities, making edges more noticeable without significantly altering the overall distribution. This enhancement is particularly useful for revealing details

4. HISTOGRAM ANALYZATION

HISTOGRSAM 5

This histogram presents the pixel intensity distribution after inverting the image. In an inversion transformation, each pixel value is replaced by its complementary value within the available range, causing bright regions to become dark and vice versa.

The inversion flips the histogram horizontally, shifting intensity values that were previously concentrated in the darker to mid-range levels toward the higher end. The sharp peak on the right side of the histogram reflects the transformation of darker pixels into brighter values. This technique is often used for stylistic purposes or to enhance visibility in specialized applications, such as medical imaging

HISTOGRSAM 6

This histogram represents the pixel intensity distribution after applying a segmentation process to the image. Image segmentation typically involves dividing the image into distinct regions based on certain characteristics, often separating foreground objects from the background or isolating specific intensity ranges.

In this histogram, we see a large concentration of pixel values around the lower intensity range (0-100), followed by a sharp drop in the mid-range (100-200), and then a few isolated peaks toward the higher intensities (200-255). This pattern suggests that the segmentation process has effectively categorized the pixels into specific intensity groups, likely distinguishing between foreground and background areas.

The distribution shows a sparse presence of intensity values in the mid-range, meaning most pixels have been grouped into darker or lighter intensities, with little in-between. This is characteristic of segmentation techniques that aim to isolate specific regions based on contrast differences. The peaks near the darker values may represent background elements, while the high-intensity peaks could correspond to segmented regions of interest, such as bright objects or features within the image.

4. HISTOGRAM ANALYZATION

HISTOGRSAM 7

This histogram shows the pixel intensity distribution of an image processed with histogram equalization, Gaussian filtering, and sharpening. The intensities span the full range from 0 to 255, indicating a balanced spread of brightness and contrast.

The darker intensity range (0 to 50) has a noticeable concentration of values, showing that some dark areas remain distinct. In the mid-range (50 to 150), the histogram displays smooth fluctuations, reflecting the Gaussian filter's effect in reducing noise while preserving tonal transitions. Toward the brighter end (200 to 255), pixel values remain well-distributed, indicating bright areas are represented without over-saturation.

The relatively smooth and uniform appearance of the histogram reflects the impact of histogram equalization, which enhances contrast by spreading intensities more evenly. The Gaussian filter further smooths the distribution by reducing noise, while sharpening enhances edges, making transitions more distinct without disturbing the overall balance. The result is an image with improved clarity, balanced contrast, and minimal noise

WHY THE EDGE DETECTION HISTOGRAM WAS NOT ANALYZED

The histogram for the Canny edge detection showed no meaningful results because the output is grayscale and binary or near-binary. Pixel values are limited to either low intensity (non-edges) or high intensity (edges), typically 0 for black and 255 for white. This narrow intensity range makes the histogram sparse, offering little insight compared to histograms from other processing techniques that involve a broader spread of pixel values.

5. CODE EXPLAINED

1.

```
function medical_image_processing_gui()
    % Create a dialog box for the user to decide whether to proceed
    choice = questdlg('Welcome to the Medical Radiology Images Processing Tool! This program allows you to select a radiology image and apply various e
        'Medical Image Processing', ...
        'Yes', 'Cancel', 'Yes');

    % Handle the user's choice
    switch choice
        case 'Yes'
            runProgram();  % Proceed to the main program
        case 'Cancel'
            return;  % Exit the program
    end
end
```

This function initializes the program by displaying a dialog box using the questdlg() function. It prompts the user to decide whether to proceed with the tool or exit.

Dialog Box Creation: Prompts the user with the choice to continue or cancel.

User Choice Handling: If "Yes" is selected, the main GUI is launched via runProgram(). If "Cancel" is chosen, the program exits.

2.

```
function runProgram()
    % Create the main figure window
    fig = figure('Name', 'Medical Image Selection', 'Position', [100 100 800 600], 'Color', [0.5, 0.7, 1]);  % Light blue background

    % Display 5 sample images for selection in a 1x5 grid
    for i = 1:5
        subplot(1,5,i);  % Create a row of 5 images
        imgPath = fullfile('MATLAB33', ['image', num2str(i), '.jpg']);  % Path to the image
        imgHandle = imshow(imread(imgPath));  % Display the image and get the image handle
        title(['Image ', num2str(i)], 'Color', 'white');  % Set title text to white

        % Set a callback directly on the image handle
        set(imgHandle, 'ButtonDownFcn', @(src, event)processImageSelection(i));
    end
```

This function creates the main GUI window and displays a row of sample images for the user to select.

Creating the Main Window:

The figure() function creates a GUI window with the title "Medical Image Selection". It sets the size and position of the window and applies a light blue background.

Displaying Sample Images:

A for-loop displays five images in a 1x5 grid using subplot(). Each image is loaded from the MATLAB33 folder and shown using imshow(). A title is added to each image with white text.

Setting Image Callbacks:

Each image is clickable. When clicked, it triggers the processImageSelection() function with the corresponding image index as input.

5. CODE EXPLAINED

3.

```
    % Add a button below the images to upload an image from the user's device
    uicontrol('Style', 'pushbutton', 'String', 'Upload Image', ...
        'Position', [350 50 100 90], 'BackgroundColor', [0, 0, 0.5], ...
        'ForegroundColor', 'white', ...
        'Callback', @(src, event)uploadImageFromDevice());
end

% Function to handle image selection from the pre-loaded images
function processImageSelection(imageIndex)
    disp(['Selected Image: ', num2str(imageIndex)]);

    % Load the selected image
    selectedImage = imread(['image', num2str(imageIndex), '.jpg']);

    % Continue with the operation selection process
    proceedWithSelectedImage(selectedImage);
end
```

This section of the code adds a button to the GUI for uploading images and defines a function to handle pre-loaded image selection.

Upload Image Button:

The uicontrol() function creates a push button labeled "Upload Image". It is positioned below the images and styled with a dark blue background and white text. When clicked, the button triggers the uploadImageFromDevice() function to allow users to upload an image.

Image Selection Handling:

The processImageSelection() function is called when a user clicks on one of the pre-loaded images. It displays the selected image's index using disp() and loads the corresponding image with imread(). After loading, the program calls proceedWithSelectedImage() to continue with further operations

5. CODE EXPLAINED

4.

```
% Function to handle image upload from the user's device
function uploadImageFromDevice()
[file, path] = uigetfile({'.jpg;.jpeg;*.png', 'Image Files (.jpg, *.jpeg, *.png)'}, 'Select an Image');  % Open file dialog
    if isequal(file, 0)
        disp('User canceled image upload.');  % If no file is selected, cancel
    else
        selectedImage = imread(fullfile(path, file));  % Read the selected image
        disp(['User selected: ', fullfile(path, file)]);

        % Continue with the operation selection process
        proceedWithSelectedImage(selectedImage);
    end
end
```

This function allows the user to upload an image from their device using a file dialog.

Opening File Dialog: Uses uigetfile() to open a dialog for selecting an image file.

Handling File Selection: If no file is selected, it displays a cancellation message. Otherwise, it reads the selected image using imread() and displays the file path.

Proceeding with Operations: Calls proceedWithSelectedImage() to continue the image processing flow

`

`

5. CODE EXPLAINED

5.

```
Function to handle the selected image (from either pre-loaded or uploaded)
nction proceedWithSelectedImage(selectedImage)
  % Create a figure divided into input and output sections
  fig = figure('Name', 'Select Operations', 'Position', [300, 300, 800, 400], 'Color', [0.5, 0.7, 1]);  % Light blue background

  % Input section for selected image and operations checkboxes
  inputPanel = uipanel('Title', 'Input', 'Position', [0.05 0.05 0.4 0.9], 'BackgroundColor', [0.5, 0.7, 1], 'ForegroundColor', 'white');  % Light bl
  axInput = axes('Parent', inputPanel, 'Position', [0.1 0.5 0.8 0.4]);  % Display selected image
  imshow(selectedImage, 'Parent', axInput);
  title(axInput, 'Selected Image', 'Color', 'white');  % Set title text to white

  % Checkboxes for selecting the operations
  cb1 = uicontrol('Parent', inputPanel, 'Style', 'checkbox', 'String', 'Gaussian Filter', ...
      'Position', [100 150 150 20], 'Value', 0, 'BackgroundColor', [0, 0, 0.5], 'ForegroundColor', 'white');
  cb2 = uicontrol('Parent', inputPanel, 'Style', 'checkbox', 'String', 'Histogram Equalization', ...
      'Position', [100 120 150 20], 'Value', 0, 'BackgroundColor', [0, 0, 0.5], 'ForegroundColor', 'white');
  cb3 = uicontrol('Parent', inputPanel, 'Style', 'checkbox', 'String', 'Sharpening', ...
      'Position', [100 90 150 20], 'Value', 0, 'BackgroundColor', [0, 0, 0.5], 'ForegroundColor', 'white');

  % Output section for processed image
  outputPanel = uipanel('Title', 'Output', 'Position', [0.55 0.05 0.4 0.9], 'BackgroundColor', [0.5, 0.7, 1], 'ForegroundColor', 'white');  % Light
  axOutput = axes('Parent', outputPanel, 'Position', [0.1 0.5 0.8 0.4]);
```

This function creates a GUI layout to display the selected image and provide options for applying image processing techniques.

Figure Creation: Creates a GUI window titled "Select Operations" with a light blue background.

Input Section: Displays the selected image inside a panel labeled "Input".

Operation Checkboxes: Provides three checkboxes to select Gaussian Filter, Histogram Equalization, and Sharpening operations.

Output Section: Prepares an output panel to display the processed image.

5. CODE EXPLAINED

6.

```
    % Button to process the image with selected operations
    uicontrol('Style', 'pushbutton', 'String', 'Process', ...
        'Position', [150 40 100 30], 'BackgroundColor', [0, 0, 0.5], ...  % Dark blue background for button
        'ForegroundColor', 'white', ...  % White text
        'Callback', @(src, event)updateProcessedImage(selectedImage, cb1, cb2, cb3, axOutput));

    % Confirm button to finalize selection and proceed
    uicontrol('Style', 'pushbutton', 'String', 'Confirm', ...
        'Position', [550 40 100 30], 'BackgroundColor', [0, 0, 0.5], ...  % Dark blue background for button
        'ForegroundColor', 'white', ...  % White text
        'Callback', @(src, event)confirmProcessing(selectedImage, cb1, cb2, cb3));
end

% Function to update the processed image in the output section
function updateProcessedImage(selectedImage, cb1, cb2, cb3, axOutput)
    % Process the image with selected techniques
    processedImage = applyOperations(selectedImage, cb1, cb2, cb3);
    % Display the processed image in the output section
    imshow(processedImage, 'Parent', axOutput);
    title(axOutput, 'Processed Image', 'Color', 'white');  % Set title text to white
end
```

This section defines buttons for processing the selected image and confirming the user's choice, along with a function to update the processed image in the output section.

Process Button:Triggers the updateProcessedImage() function to apply selected operations to the image.Styled with a dark blue background and white text.

Confirm Button: Finalizes the user's operation selection and proceeds with processing.
Also styled with dark blue background and white text.

updateProcessedImage() Function: Applies the chosen operations using applyOperations().
Displays the processed image in the output panel with a white title text

5. CODE EXPLAINED

7.

```
% Function to process the image with selected techniques
function processedImage = applyOperations(image, cb1, cb2, cb3)
    % Step 1: Noise Reduction (Gaussian Filtering) if selected
    if cb1.Value == 1
        disp('Applying Gaussian Filter...');
        image = imgaussfilt(image, 2);  % Gaussian filter with size 2
    end

    % Step 2: Histogram Equalization if selected
    if cb2.Value == 1
        disp('Applying Histogram Equalization...');
        if size(image, 3) == 3  % If it's RGB
            % Apply histeq to each channel
            image = cat(3, histeq(image(:, :, 1)), histeq(image(:, :, 2)), histeq(image(:, :, 3)));
        else
            image = histeq(image);  % Grayscale image
        end
    end

    % Step 3: Sharpening if selected
    if cb3.Value == 1
        disp('Applying Sharpening...');
        image = imsharpen(image, 'Radius', 2, 'Amount', 1);
    end

    processedImage = image;
end
```

This function applies selected image processing techniques (Gaussian filtering, histogram equalization, and sharpening) based on the user's choices.

Gaussian Filtering: If selected, reduces noise using a Gaussian filter of size 2.

Histogram Equalization:If selected, enhances contrast.
For RGB images, applies histeq() to each channel individually.
For grayscale images, applies histeq() directly.

Sharpening: If selected, applies sharpening with a radius of 2 and an amount of 1 to enhance edges.

5. CODE EXPLAINED

8.

```
% Function to confirm processing and continue with other operations
% Function to confirm processing and continue with other operations
function confirmProcessing(selectedImage, cb1, cb2, cb3)
    % Process the image with selected techniques
    processedImage = applyOperations(selectedImage, cb1, cb2, cb3);

    % Step 4: Edge Detection using Canny
    disp('Performing Edge Detection...');
    edgeDetectedImage = edge(im2gray(processedImage), 'Canny');

    % Step 5: Invert the Image (Negative Transformation)
    disp('Inverting Image...');
    invertedImage = imcomplement(processedImage);

    % Step 6: Active Contour (Snake) Segmentation
    disp('Performing Active Contour Segmentation...');
    grayImage = im2gray(invertedImage); % Convert to grayscale

    % Pre-process with a binary threshold to initialize mask
    binaryMask = imbinarize(grayImage, 'adaptive', 'Sensitivity', 0.5);

    % Initialize the contour with the binary mask
    iterations = 300;  % Number of iterations for the active contour
    activeContourMask = activecontour(grayImage, binaryMask, iterations, 'edge');

    % Overlay the contour on the original image
    segmentedActiveContourImage = imoverlay(grayImage, activeContourMask, [1 0 0]);  % Red contour

    % Display the results in subplots
    figure('Color', [0.5, 0.7, 1]);  % Light blue background
    subplot(2, 3, 1); imshow(selectedImage); title('Original Image', 'Color', 'white');
    subplot(2, 3, 2); imshow(processedImage); title('Processed Image', 'Color', 'white');
    subplot(2, 3, 3); imshow(edgeDetectedImage); title('Edge Detection', 'Color', 'white');
```

This function applies advanced image processing operations to the selected image and displays the results.

Edge Detection (Canny): Converts the image to grayscale and applies Canny edge detection.

Inversion`(Negative Transformation): Inverts the pixel values to create a negative image.

Active Contour (Snake) Segmentation:

- Converts the inverted image to grayscale and applies adaptive thresholding.

- Uses the active contour method to detect edges with a red overlay.

Displaying Results:

Uses subplot() to display the original image, processed image, edge detection, inverted image, and segmentation results in a 2x3 grid with white titles

5. CODE EXPLAINED

9.

```
    % Ask user if they want to see histograms
    choice_hist = questdlg('Do you want to see the histograms of the original and processed images?', ...
        'View Histograms', 'Yes', 'No', 'No');

    switch choice_hist
        case 'Yes'
            displayHistograms(selectedImage, processedImage, invertedImage, segmentedActiveContourImage, cb1, cb2, cb3);  % Display histograms
    end

    % Ask user if they want to restart or exit
    choice = questdlg('Do you want to process another image or exit?', ...
        'Continue or Exit', 'Restart', 'Exit', 'Restart');

    switch choice
        case 'Restart'
            runProgram();  % Restart the program
        case 'Exit'
            close all;  % Close all windows and exit
    end
nd
```

This section prompts the user to display histograms and provides options to restart or exit the program.

Histogram Display Prompt: Uses questdlg() to ask if the user wants to view histograms.
If "Yes" is selected, it calls displayHistograms() to show the histograms for the original, processed, inverted, and segmented images.

Program Control Prompt:

Asks the user if they want to restart the program or exit.
If "Restart" is chosen, the program starts again by calling runProgram().
If "Exit" is selected, all windows are closed using close all

This logic ensures smooth interaction by giving users control over viewing histograms and continuing or exiting the program.

5. CODE EXPLAINED

10.

```
% Function to display histograms of the original and processed images
function displayHistograms(originalImage, processedImage, invertedImage, segmentedImage, cb1, cb2, cb3)
    figure('Name', 'Image Histograms', 'Color', [0.5, 0.7, 1]);  % Light blue background

    % Convert to grayscale if the images are RGB
    if size(originalImage, 3) == 3
        originalImageGray = rgb2gray(originalImage);
    else
        originalImageGray = originalImage;
    end

    if size(processedImage, 3) == 3
        processedImageGray = rgb2gray(processedImage);
    else
        processedImageGray = processedImage;
    end

    if size(invertedImage, 3) == 3
        invertedImageGray = rgb2gray(invertedImage);
    else
        invertedImageGray = invertedImage;
    end

    if size(segmentedImage, 3) == 3
        segmentedImageGray = rgb2gray(segmentedImage);
    else
        segmentedImageGray = segmentedImage;
    end
```

This function displays histograms for the original, processed, inverted, and segmented images by first converting them to grayscale if necessary.

Figure Creation:

A new figure window is created with the title "Image Histograms" and a light blue background.

Grayscale Conversion:

Each image (original, processed, inverted, segmented) is checked for its number of channels using size().

If the image is RGB, it is converted to grayscale using rgb2gray().

If the image is already grayscale, it is used as-is

5. CODE EXPLAINED

11.

```
% Display the histogram of the original image
subplot(3, 3, 1);
imhist(originalImageGray);
title('Original Image Histogram', 'Color', 'white');  % Set title text to white

% Display the histogram of the processed image
subplot(3, 3, 2);
imhist(processedImageGray);
title('Processed Image Histogram', 'Color', 'white');  % Set title text to white

% Conditionally display histograms for each selected operation
if cb1.Value == 1
    subplot(3, 3, 4);
    imhist(imgaussfilt(originalImageGray, 2));
    title('Gaussian Filter Histogram', 'Color', 'white');  % Set title text to white
end
if cb2.Value == 1
    subplot(3, 3, 5);
    imhist(histeq(originalImageGray));
    title('Histogram Equalization', 'Color', 'white');  % Set title text to white
end
if cb3.Value == 1
    subplot(3, 3, 6);
    imhist(imsharpen(originalImageGray, 'Radius', 2, 'Amount', 1));
    title('Sharpening Histogram', 'Color', 'white');  % Set title text to white
end
```

This section of the code displays the histograms for the original, processed, and selectively applied operations using subplots.

Displaying Original and Processed Image Histograms:

Uses subplot() to place the histograms for the original and processed images in a 3x3 grid.
Titles are set with white text for better visibility

Conditional Display of Operation Histograms:

If Gaussian filtering is selected, the corresponding histogram is displayed in subplot (3, 3, 4).
If Histogram equalization is selected, its histogram appears in subplot (3, 3, 5).
If Sharpening is selected, its histogram is shown in subplot (3, 3, 6).
This setup allows the user to visually compare the intensity distributions before and after processing, as well as for each individual operation, providing meaningful insights into the image transformations

5. CODE EXPLAINED

12.

```
    % Display the histogram of the inverted image
    subplot(3, 3, 7);
    imhist(invertedImageGray);
    title('Inverted Image Histogram', 'Color', 'white');  % Set title text to white

    % Display the histogram of the segmented image
    subplot(3, 3, 8);
    imhist(segmentedImageGray);
    title('Segmented Image Histogram', 'Color', 'white');  % Set title text to white

    sgtitle('Histogram Results', 'Color', 'white');  % Set sgtitle text to white
end
```

This section of the code displays the histograms for the inverted and segmented images and sets an overall title for the histogram results.

Inverted Image Histogram:
Displayed in subplot (3, 3, 7).
The histogram shows the pixel intensity distribution after inversion, with the title in white.

Segmented Image Histogram:
Displayed in subplot (3, 3, 8).
Shows the intensity distribution for the segmented image, also with white text for the title.

Overall Histogram Title:
The sgtitle() function sets a general title for all histograms as "Histogram Results" with white text for clarity.

This layout ensures that all relevant histograms, including those for inverted and segmented images, are clearly organized and displayed for easy comparison

6. WHAT WE LEARNED

1.WHAT HAVE YOU LEARNED FROM THE PROJECT?

We gained practical experience in image processing techniques such as filtering, contrast enhancement, and segmentation. Additionally, we learned how these techniques can improve the quality of medical images, aiding in more accurate diagnoses.

2.WHAT ADVANTAGES DID YOU GAIN FROM DESIGNING THIS PROJECT?

This project allowed us to apply theoretical knowledge in a practical context, enhance our problem-solving skills, and understand the importance of precision in medical image processing.

3.DID YOU FIND TEAMWORK USEFUL IN THIS PROGRAMMING PROJECT?

Yes, teamwork was essential. It allowed us to divide tasks efficiently, leverage each member's strengths, and complete the project on time while improving our collaboration skills.

`

4.ARE YOU SATISFIED WITH YOUR APPLICATION?

Yes, the application performs as expected, meeting the project's goals. We are proud of the outcome and its effectiveness in enhancing image quality.

`

5.DO YOU FEEL READY TO WORK ON A LARGE-SCALE PROJECT IN THE FUTURE?

Yes, this project has given us the confidence to take on more complex projects. We feel well-prepared to manage larger tasks and collaborate effectively in the future.

7. PROJECT DIFFICULTIES

WHAT HAVE YOU LEARNED FROM THE PROJECT?

During the project, we encountered technical difficulties in achieving optimal results with segmentation. We also explored the use of line detection techniques to enhance the image processing workflow, but it proved ineffective for our specific needs and did not contribute meaningfully to the analysis.

In terms of resources, the availability of suitable medical images was limited, which restricted the scope of our testing. However, through effective teamwork and collaboration, we were able to overcome these challenges and successfully complete the project.

`

`

8. THE DISTIBUTION OF THE WORK

NAME	TASK
NABA'A ABDUL RAHMAN	writing code to Apply (edge detection , segmention and inverting) on the images + histograms analysis + project documentation
FATIMA AL-ZAHRANI	GUI to provide a clear layout for users to select and preview image processing operations before finalizing their choice with the Confirm button+ project documentation
AMAL AL-ZAHRANI	A user interface program for medical image processing that allows displaying sample images or uploading an external image and applying various enhancement+ data collection
ASAL AL-HASSAN	writing code to Displaying Histograms for Original and Processed Images + draw a diagram for project flow
FIDA'A FELEMBAN	writing code to Presenting Options for the User to Execute (Gaussian filter , Sharpening, Histogram Equalization, or Apply All at Once) + data collection

9. REFRENCEC USED TO MAKE THIS PROJECT

change colors of GUI components

https://www.mathworks.com/matlabcentral/answers/514285-how-do-i-change-matlab-gui-components-color

GUI text boxes

https://www.mathworks.com/help/matlab/ref/questdlg.html

code for select boxes for image enhancement and other buttons

https://www.mathworks.com/help/matlab/ref/uicontrol.html

code responsible of allowing user to upload images :

https://www.mathworks.com/help/matlab/ref/uigetfile.html

code to display and select images:

https://www.mathworks.com/help/matlab/ref/matlab.ui.control.uicontrol-properties.html

Data references:

https://www.kaggle.com/datasets?search=X-ray+images&page=12https://

https://nihcc.app.box.com/v/ChestXray-NIHCC/file/221185642661

https://data.mendeley.com/datasets/8fztxggjnc/1

REPHRASING THE CONTENT AND HELPING WITH ERRORS FOUND IN THE CODE AS WELL AS HELLPING WITH FINDING MEDICAL DATA WEBSITES
CHATGPT

10- PROJECT DECLARATION

We confirm that the work of this project was solely undertaken by ourselves and that no help was provided from other sources as those allowed. As well as we confirm that we completely aware of the violation consequences of the academic integrity.

NABA'A ABDUL RAHMAN
ID: 443013099

FATMA ALZAHARNI
ID: 444006628

ASAL- ALHASSAN
ID:444001787

FIDA'A FELEMBAN
ID: 444006581

AMAL AL-ZAHRANI
ID: 444002268

`