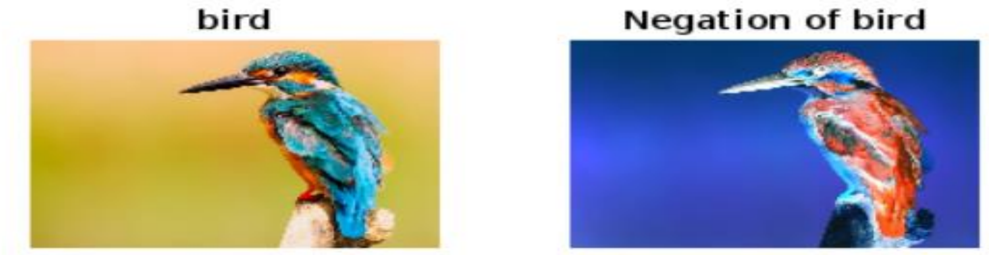
Image Processing Practical

Practical 1

1. **To implement Image negation.**

```
a = imread("bird.jpg");
subplot(1,2,1);
imshow(a);
title("bird");

b = 255 - a;
subplot(1,2,2);
imshow(b);
title('Negation of bird');
```

Output:

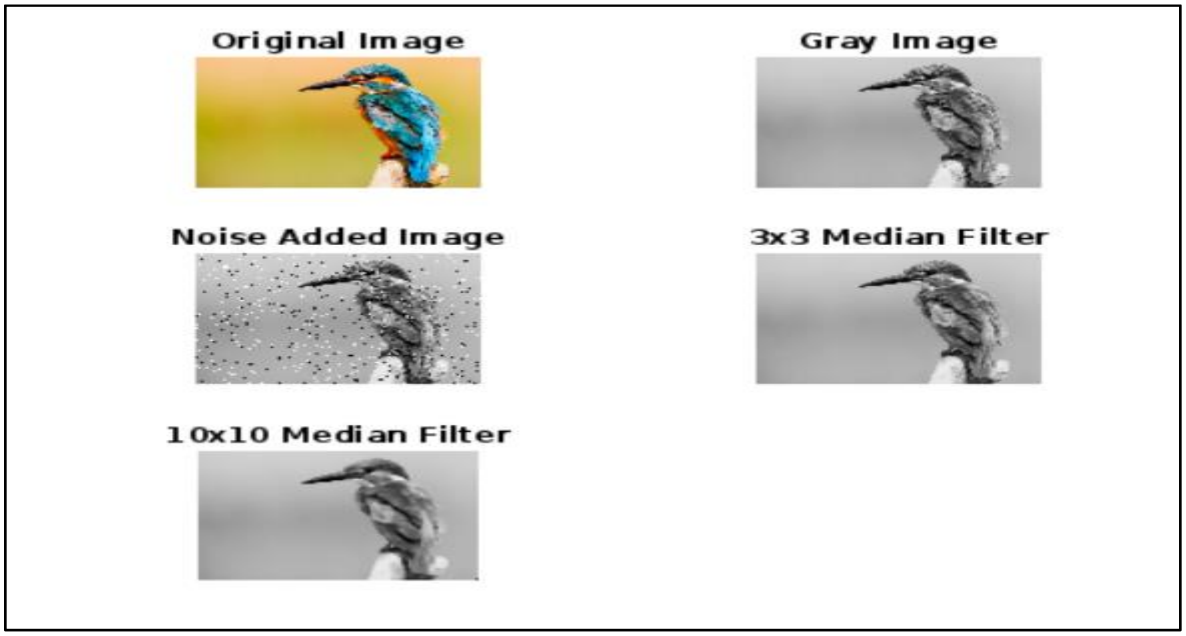


Practical 2

2. **MATLAB code to remove salt and pepper noise using median filter.**

```
K = imread('kingfisher-2046453_640.jpg');
% Convert to grayscale (since medfilt2 works only on grayscale)
grayK = rgb2gray(K);
% Add salt and pepper noise
j = imnoise(grayK, 'salt & pepper', 0.05);
% Apply median filtering
f = medfilt2(j, [3, 3]);
f1 = medfilt2(j, [10, 10]);
% Display results
subplot(3, 2, 1);
imshow(K);
title('Original Image');
subplot(3, 2, 2);
imshow(grayK);
title('Gray Image');
subplot(3, 2, 3);
imshow(j);
title('Noise Added Image');
subplot(3, 2, 4);
imshow(f);
title('3x3 Median Filter');
subplot(3, 2, 5);
imshow(f1);
title('10x10 Median Filter');
```

Output:

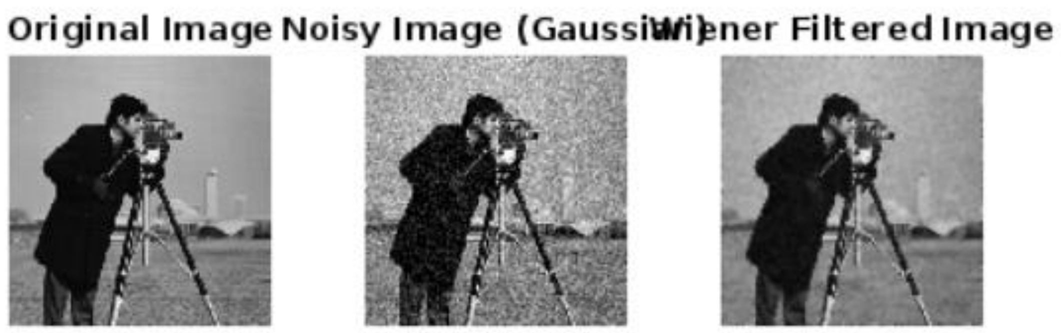


Practical 3

3. Write a MATLAB code to implement the Wiener filter.

```
% Read the original image
original_image = imread('cameraman.tif');  % Use any grayscale image
% Add Gaussian noise to the image
noisy_image = imnoise(original_image, 'gaussian', 0, 0.01);
% Apply Wiener filter with a 5x5 neighborhood
filtered_image = wiener2(noisy_image, [5 5]);
% Display the images
subplot(1, 3, 1);
imshow(original_image);
title('Original Image');
subplot(1, 3, 2);
imshow(noisy_image);
title('Noisy Image (Gaussian)');
subplot(1, 3, 3);
imshow(filtered_image);
title('Wiener Filtered Image');
```

Output:

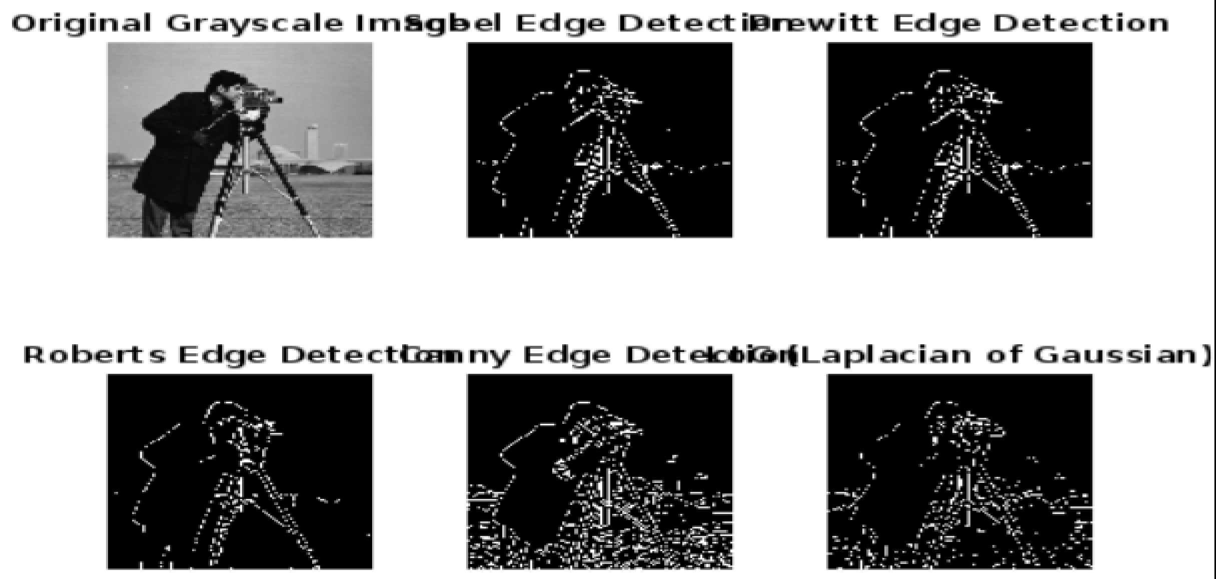


Practical 4

4. Write a program to demonstrate edge detection using different techniques.

```
% Read and convert the image to grayscale
img = imread('cameraman.tif');  % Use any suitable image
gray_img = im2gray(img);       % Use this if img is RGB
% Apply different edge detection methods
edge_sobel = edge(gray_img, 'sobel');
edge_prewitt = edge(gray_img, 'prewitt');
edge_roberts = edge(gray_img, 'roberts');
edge_canny = edge(gray_img, 'canny');
edge_log = edge(gray_img, 'log');
% Display results
figure;
subplot(2, 3, 1);
imshow(gray_img);
title('Original Grayscale Image');
subplot(2, 3, 2);
imshow(edge_sobel);
title('Sobel Edge Detection');
subplot(2, 3, 3);
imshow(edge_prewitt);
title('Prewitt Edge Detection');
subplot(2, 3, 4);
imshow(edge_roberts);
title('Roberts Edge Detection');
subplot(2, 3, 5);
imshow(edge_canny);
title('Canny Edge Detection');
subplot(2, 3, 6);
imshow(edge_log);
title('LoG (Laplacian of Gaussian)');
```

Output:

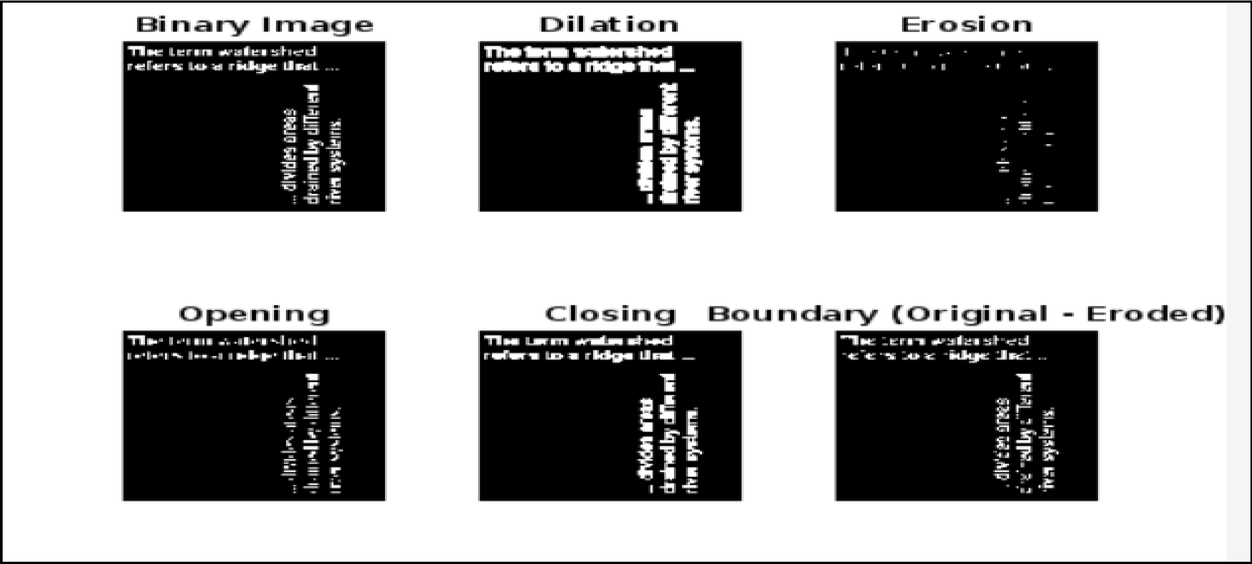


Practical 5

5. Write a program to demonstrate morphological operations on binary images.

```
% Read the input image
original = imread('text.png');  % Use a binary-style image
% Convert to grayscale if needed
if size(original, 3) == 3
  gray = rgb2gray(original);
else
  gray = original;
end
% Convert to binary after normalizing
bw = imbinarize(im2double(gray));
% Create a structuring element
se = strel('square', 3);  % You can change to 'disk', etc.
% Morphological operations
dilated = imdilate(bw, se);
eroded = imerode(bw, se);
opened = imopen(bw, se);
closed = imclose(bw, se);
% Display all results
figure;
subplot(2, 3, 1);
imshow(bw); title('Binary Image');
subplot(2, 3, 2);
imshow(dilated); title('Dilation');
subplot(2, 3, 3);
imshow(eroded); title('Erosion');
subplot(2, 3, 4);
imshow(opened); title('Opening');
subplot(2, 3, 5);
imshow(closed); title('Closing');
subplot(2, 3, 6);
imshow(bw - eroded); title('Boundary (Original - Eroded)');
```

Output:

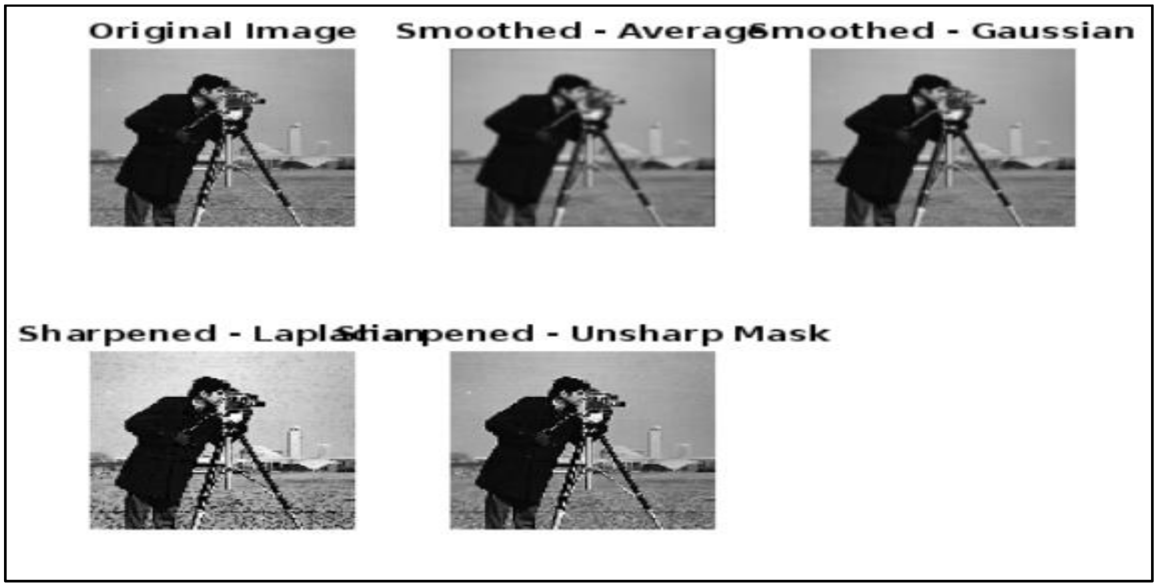


Practical 6

6. Write a program to demonstrate smoothing and sharpening of image.

```
% Read the original image
img = imread('cameraman.tif');  % Use any image of your choice
% Convert to grayscale if needed
if size(img, 3) == 3
   gray = rgb2gray(img);
else
   gray = img;
end
% -----------------------
% Smoothing operations
% -----------------------
% Average filter
avg_filter = fspecial('average', [5 5]);
smoothed_avg = imfilter(gray, avg_filter);
% Gaussian filter
gauss_filter = fspecial('gaussian', [5 5], 1);
smoothed_gauss = imfilter(gray, gauss_filter);
% -----------------------
% Sharpening operations
% -----------------------
% Laplacian sharpening
lap_filter = fspecial('laplacian', 0.2);
sharpened_lap = imadjust(gray - imfilter(gray, lap_filter));
% Unsharp masking (built-in)
sharpened_unsharp = imsharpen(gray);
% -----------------------
% Display all results
% -----------------------

figure;

subplot(2, 3, 1);

imshow(gray);

title('Original Image');

subplot(2, 3, 2);

imshow(smoothed_avg);

title('Smoothed - Average');

subplot(2, 3, 3);

imshow(smoothed_gauss);

title('Smoothed - Gaussian');

subplot(2, 3, 4);

imshow(sharpened_lap);

title('Sharpened - Laplacian');

subplot(2, 3, 5);

imshow(sharpened_unsharp);

title('Sharpened - Unsharp Mask');
```

OutPut:

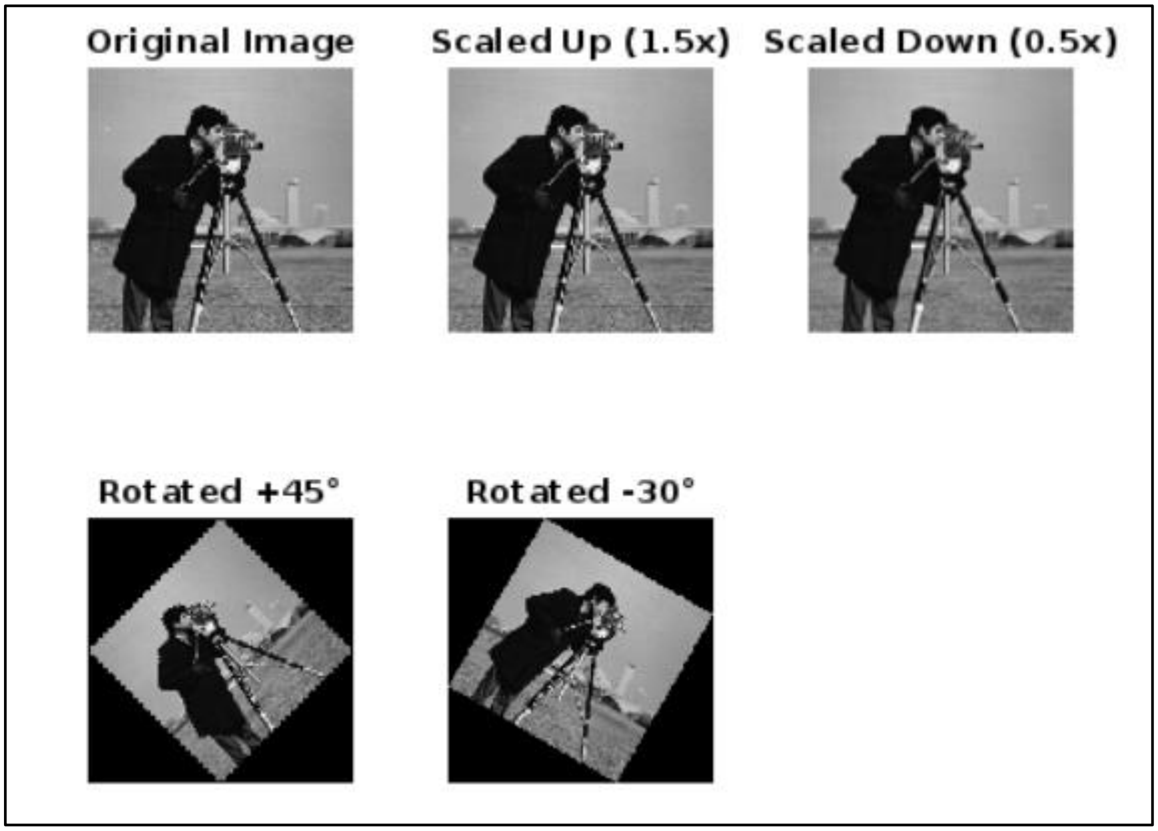


Practical 7

7. Write a MATLAB code for scaling and rotation of image.

```
% Read a different image (grayscale)
original = imread('cameraman.tif');  % Built-in MATLAB image
% ------------------
% Image Scaling
% ------------------
% Scale up by 1.5 times
scaled_up = imresize(original, 1.5);
% Scale down by 0.5 times
scaled_down = imresize(original, 0.5);
% ------------------
% Image Rotation
% ------------------
% Rotate the image 45 degrees counterclockwise
rotated_45 = imrotate(original, 45);
% Rotate the image -30 degrees (clockwise)
rotated_neg30 = imrotate(original, -30);
% ------------------
% Display Results
% ------------------
figure;
subplot(2,3,1);
imshow(original);
title('Original Image');
subplot(2,3,2);
imshow(scaled_up);
title('Scaled Up (1.5x)');
subplot(2,3,3);
imshow(scaled_down);
title('Scaled Down (0.5x)');
subplot(2,3,4);
imshow(rotated_45);
title('Rotated +45°');
subplot(2,3,5);
imshow(rotated_neg30);
title('Rotated -30°');
```

Output:

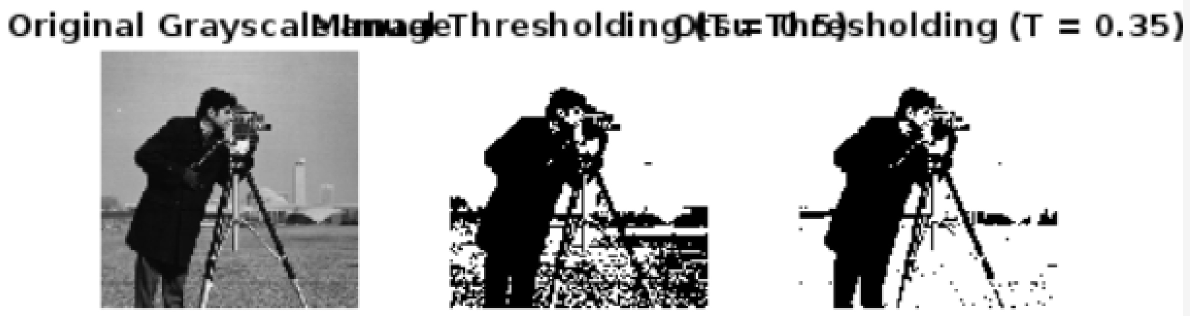


Practical 8

8. Program to implement Thresholding of an Image.

```
% Read the input image
img = imread('cameraman.tif');  % You can use your own grayscale image
% Convert to grayscale if it's RGB
if size(img, 3) == 3
  gray = rgb2gray(img);
else
  gray = img;
end
% Convert image to double for processing
gray_double = im2double(gray);
% ----------------------------
% Manual Thresholding
% ----------------------------
threshold_value = 0.5;  % Set threshold between 0 and 1
bw_manual = gray_double > threshold_value;
% ----------------------------
% Otsu's Automatic Thresholding
% ----------------------------
level = graythresh(gray);         % Otsu's threshold (returns 0–1)
bw_otsu = imbinarize(gray, level);
% ----------------------------
% Display Results
% ----------------------------
figure;
subplot(1, 3, 1);
imshow(gray);
title('Original Grayscale Image');
subplot(1, 3, 2);
imshow(bw_manual);
title(['Manual Thresholding (T = ' num2str(threshold_value) ')']);
subplot(1, 3, 3);
imshow(bw_otsu);
title(['Otsu Thresholding (T = ' num2str(level, '%.2f') ')']);
```

Output:


Original GrayscaleManual Thresholding (T = 0.5)
Otsu Thresholding (T = 0.35)

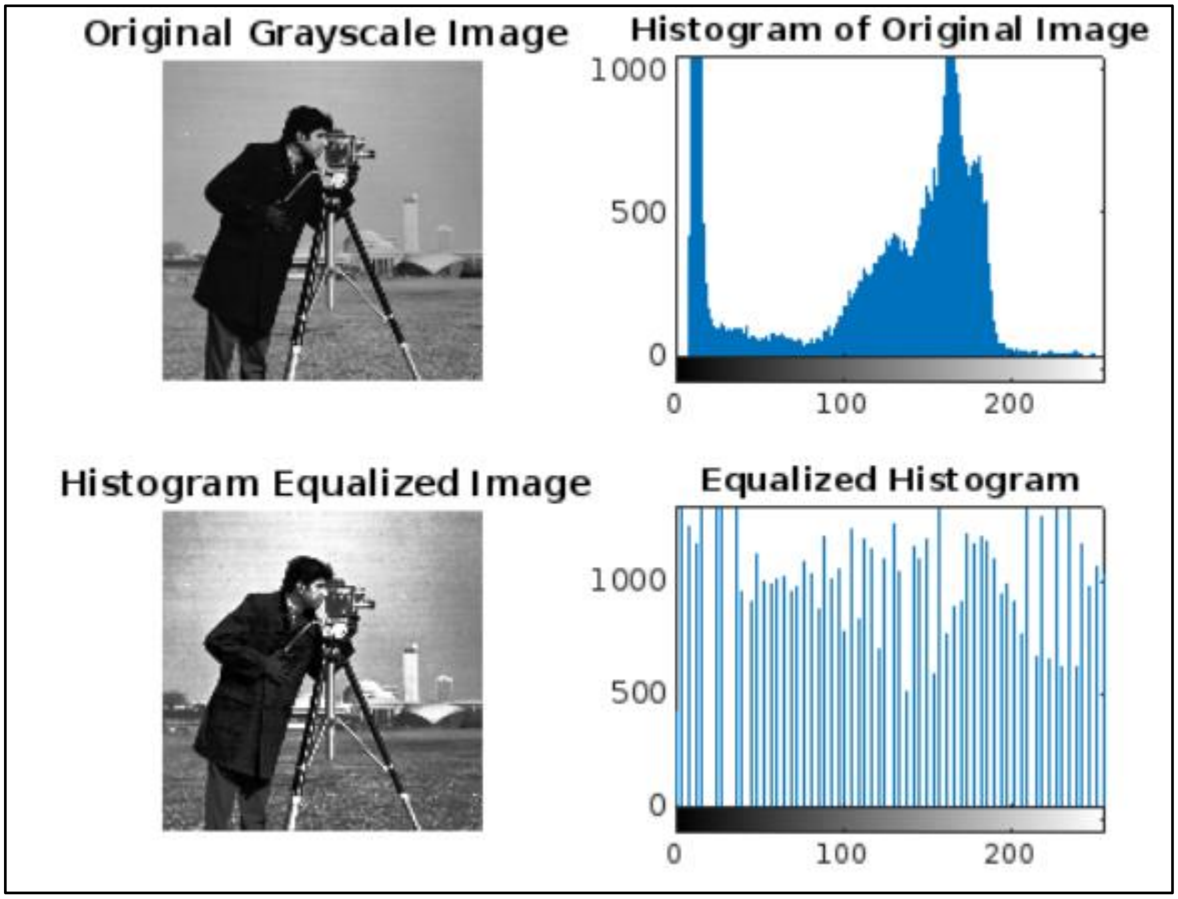
Practical 9

9. **Program to produce the Histogram, Equalized Histogram, and Equalized image of an input image.**

```
% Read the input image
img = imread('cameraman.tif');  % Replace with your own image if needed
% Convert to grayscale if image is RGB
if size(img, 3) == 3
  gray = rgb2gray(img);
else
  gray = img;
end
% ----------------------------
% Original Histogram
% ----------------------------
figure;
subplot(2, 2, 1);
imshow(gray);
title('Original Grayscale Image');
subplot(2, 2, 2);
imhist(gray);
```

```
title('Histogram of Original Image');
% ----------------------------
% Histogram Equalization
% ----------------------------
equalized_img = histeq(gray);
subplot(2, 2, 3);
imshow(equalized_img);
title('Histogram Equalized Image');
subplot(2, 2, 4);
imhist(equalized_img);
title('Equalized Histogram');
```

Output:

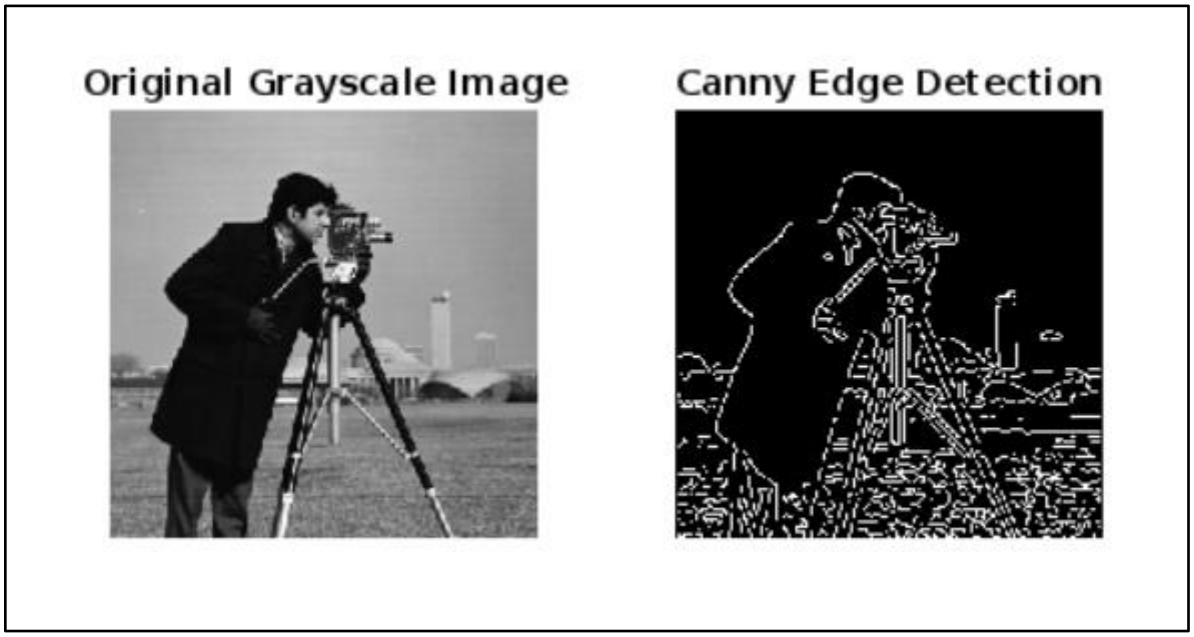


Practical 10

10. Matlab Code to find edges using canny Edge Detection.

```
% Read the input image
img = imread('cameraman.tif');  % Replace with your own image if needed
% Convert to grayscale if image is RGB
if size(img, 3) == 3
  gray = rgb2gray(img);
else
  gray = img;
end
% Apply Canny edge detection
edges_canny = edge(gray, 'Canny');
% Display the original and edge-detected images
figure;
subplot(1, 2, 1);
imshow(gray);
title('Original Grayscale Image');
subplot(1, 2, 2);
imshow(edges_canny);
title('Canny Edge Detection');
```

Output:


Original Grayscale Image
Canny Edge Detection