1. Image Resizing (Scaling): Display the color image and its Resized images

```
% Read the color image
originalImage = imread('your_image.jpg');

% Display the original image
figure; % Create a new figure window
subplot(2, 2, 1); % Create a 2x2 grid and activate the first subplot
imshow(originalImage); % Display the original image
title('Original Image'); % Add a title to the first subplot

% Resize the image to 0.5 times its original size (scaling down)
resizedImage_half = imresize(originalImage, 0.5);

% Display the resized image (scaled down)
subplot(2, 2, 2); % Activate the second subplot
imshow(resizedImage_half); % Display the resized image
title('Resized Image (0.5x)'); % Add a title to the second subplot

% Resize the image to 2 times its original size (scaling up)
resizedImage_double = imresize(originalImage, 2);

% Display the resized image (scaled up)
subplot(2, 2, 3); % Activate the third subplot
imshow(resizedImage_double); % Display the resized image
title('Resized Image (2x)'); % Add a title to the third subplot

% Resize the image to a specific size [width, height]
newSize = [300, 400]; % Example: 300 pixels height, 400 pixels width
resizedImage_custom = imresize(originalImage, newSize);

% Display the resized image (custom size)
subplot(2, 2, 4); % Activate the fourth subplot
imshow(resizedImage_custom); % Display the resized image
title('Resized Image (Custom Size)'); % Add a title to the fourth subplot

% Optional: Save the resized images
imwrite(resizedImage_half, 'resized_half.jpg');
imwrite(resizedImage_double, 'resized_double.jpg');
imwrite(resizedImage_custom, 'resized_custom.jpg');
```

2. Image Rotation: Rotate an image in different angles 30, 45, 60…

```
% Read the color image
originalImage = imread('your_image.jpg');

% Display the original image
figure;
subplot(2, 3, 1);
imshow(originalImage);
title('Original Image');

% Define the angles for rotation
angles = [30, 45, 60, 90, 120]; % Add or modify angles as needed

% Rotate and display the images at different angles
for i = 1:length(angles)
    rotatedImage = imrotate(originalImage, angles(i));
```

```
    % Display the rotated image
    subplot(2, 3, i+1);
    imshow(rotatedImage);
    title(['Rotated Image (', num2str(angles[i]), '°)']);
end

% Optional: Save the rotated images
for i = 1:length(angles)
    rotatedImage = imrotate(originalImage, angles(i));
    imwrite(rotatedImage, ['rotated_' num2str(angles(i)) '.jpg']);
end
```

3. Binary Image simulation. For example

```
% Define the size of the checkerboard
checkerboardSize = 8; % 8x8 checkerboard
blockSize = 50; % Size of each square block in pixels

% Create a checkerboard pattern
checkerboardPattern = repmat([1 0; 0 1], checkerboardSize/2, checkerboardSize/2);

% Resize the checkerboard pattern to the desired block size
binaryImage = kron(checkerboardPattern, ones(blockSize));

% Display the binary image
figure;
imshow(binaryImage);
title('Checkerboard Binary Image');

///convert to binary image

% Read the color image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale (if it is not already)
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(1, 2, 1);
imshow(grayImage);
title('Grayscale Image');

% Convert the grayscale image to a binary image using a threshold
threshold = 0.5; % You can adjust this threshold (range is 0 to 1)
binaryImage = imbinarize(grayImage, threshold);

% Display the binary image
subplot(1, 2, 2);
imshow(binaryImage);
title('Binary Image');

% Optional: Save the binary image
imwrite(binaryImage, 'binary_image.jpg');
```

4. Write a program to remove noise from an image using median filter and Mean filter.

```
% Read the noisy image
noisyImage = imread('your_noisy_image.jpg');

% Convert to grayscale if the image is in color
if size(noisyImage, 3) == 3
   grayImage = rgb2gray(noisyImage);
else
   grayImage = noisyImage;
end

% Display the original noisy image
figure;
subplot(1, 3, 1);
imshow(grayImage);
title('Original Noisy Image');

% Apply median filter
medianFiltered = medfilt2(grayImage);

% Display the median filtered image
subplot(1, 3, 2);
imshow(medianFiltered);
title('Median Filtered Image');

% Apply mean filter (using convolution with averaging kernel)
kernelSize = 3; % Define the size of the kernel
meanKernel = fspecial('average', kernelSize);
meanFiltered = imfilter(grayImage, meanKernel, 'replicate');

% Display the mean filtered image
subplot(1, 3, 3);
imshow(meanFiltered);
title('Mean Filtered Image');

% Optional: Save the filtered images
imwrite(medianFiltered, 'median_filtered.jpg');
imwrite(meanFiltered, 'mean_filtered.jpg');


// to preserv color different program dnt copy continously.

% Read the noisy image
noisyImage = imread('your_noisy_image.jpg');

% Display the original noisy image
figure;
subplot(2, 3, 1);
imshow(noisyImage);
title('Original Noisy Image');

% Initialize filtered images for each channel
medianFiltered = noisyImage;
meanFiltered = noisyImage;

% Apply median filter to each channel
medianFiltered(:,:,1) = medfilt2(noisyImage(:,:,1)); % Red channel
medianFiltered(:,:,2) = medfilt2(noisyImage(:,:,2)); % Green channel
```

```
medianFiltered(:,:,3) = medfilt2(noisyImage(:,:,3)); % Blue channel

% Display the median filtered image
subplot(2, 3, 2);
imshow(medianFiltered);
title('Median Filtered Image');

% Apply mean filter to each channel
kernelSize = 3; % Define the size of the kernel
meanKernel = fspecial('average', kernelSize);

meanFiltered(:,:,1) = imfilter(noisyImage(:,:,1), meanKernel, 'replicate'); % Red channel
meanFiltered(:,:,2) = imfilter(noisyImage(:,:,2), meanKernel, 'replicate'); % Green channel
meanFiltered(:,:,3) = imfilter(noisyImage(:,:,3), meanKernel, 'replicate'); % Blue channel

% Display the mean filtered image
subplot(2, 3, 3);
imshow(meanFiltered);
title('Mean Filtered Image');

% Optional: Display grayscale images for comparison
grayImage = rgb2gray(noisyImage);
subplot(2, 3, 4);
imshow(grayImage);
title('Grayscale Noisy Image');

medianGrayFiltered = medfilt2(grayImage);
subplot(2, 3, 5);
imshow(medianGrayFiltered);
title('Median Filtered Grayscale Image');

meanGrayFiltered = imfilter(grayImage, meanKernel, 'replicate');
subplot(2, 3, 6);
imshow(meanGrayFiltered);
title('Mean Filtered Grayscale Image');

% Optional: Save the filtered images
imwrite(medianFiltered, 'median_filtered_color.jpg');
imwrite(meanFiltered, 'mean_filtered_color.jpg');
```

5. Write a program to detect edge of an image using canny edge detection Algorithm.

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(1, 2, 1);
imshow(grayImage);
title('Original Grayscale Image');
```

```
% Apply Canny edge detection
edgeDetectedImage = edge(grayImage, 'Canny');

% Display the edge-detected image
subplot(1, 2, 2);
imshow(edgeDetectedImage);
title('Canny Edge Detected Image');

% Optional: Save the edge-detected image
imwrite(edgeDetectedImage, 'canny_edge_detected.jpg');
```

6. **Write a program to detect edge of an image using Laplacian of Gaussian (LoG) edge detection technique.**

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(1, 2, 1);
imshow(grayImage);
title('Original Grayscale Image');

% Apply Gaussian filter to smooth the image
smoothedImage = imgaussfilt(grayImage, 2); % Adjust the sigma as needed

% Apply Laplacian filter
laplacianFilter = fspecial('log', [5, 5], 0.5); % Adjust the size and sigma as needed
logImage = imfilter(smoothedImage, laplacianFilter, 'replicate');

% Threshold the image to get binary edges
threshold = 0.05; % Adjust the threshold as needed
edgeDetectedImage = logImage > threshold;

% Display the edge-detected image
subplot(1, 2, 2);
imshow(edgeDetectedImage);
title('LoG Edge Detected Image');

% Optional: Save the edge-detected image
imwrite(edgeDetectedImage, 'log_edge_detected.jpg');
```

7. Histogram Sliding

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(2, 2, 1);
imshow(grayImage);
title('Original Grayscale Image');

% Compute and display the original histogram
subplot(2, 2, 2);
imhist(grayImage);
title('Original Histogram');

% Define the shift value
shiftValue = 50; % Adjust this value to control the brightness increase

% Perform histogram sliding by adding the shift value to the image
slidImage = imadd(grayImage, shiftValue); % imadd automatically handles overflow

% Display the shifted image
subplot(2, 2, 3);
imshow(slidImage);
title('Shifted Image');

% Compute and display the shifted histogram
subplot(2, 2, 4);
imhist(slidImage);
title('Shifted Histogram');

% Optional: Save the shifted image
imwrite(slidImage, 'shifted_image.jpg');
```

8. Histogram Stretching

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(2, 2, 1);
```

```
imshow(grayImage);
title('Original Grayscale Image');

% Compute and display the original histogram
subplot(2, 2, 2);
imhist(grayImage);
title('Original Histogram');

% Perform histogram stretching
minValue = double(min(grayImage(:)));
maxValue = double(max(grayImage(:)));
stretchedImage = imadjust(grayImage, [minValue/255; maxValue/255], []);

% Display the stretched image
subplot(2, 2, 3);
imshow(stretchedImage);
title('Stretched Image');

% Compute and display the stretched histogram
subplot(2, 2, 4);
imhist(stretchedImage);
title('Stretched Histogram');

% Optional: Save the stretched image
imwrite(stretchedImage, 'stretched_image.jpg');
```

9. Histogram Equalization: Write a program to apply histogram equalization to enhance the contrast of an image

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(2, 2, 1);
imshow(grayImage);
title('Original Grayscale Image');

% Compute and display the original histogram
subplot(2, 2, 2);
imhist(grayImage);
title('Original Histogram');

% Perform histogram equalization
equalizedImage = histeq(grayImage);

% Display the equalized image
subplot(2, 2, 3);
imshow(equalizedImage);
title('Equalized Image');

% Compute and display the equalized histogram
```

```
subplot(2, 2, 4);
imhist(equalizedImage);
title('Equalized Histogram');

% Optional: Save the equalized image
imwrite(equalizedImage, 'equalized_image.jpg');
```

10.Write a program to apply Apply Gaussian smoothing to blur an image. Apply Gaussian smoothing with a specified standard deviation (sigma)

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(1, 2, 1);
imshow(grayImage);
title('Original Grayscale Image');

% Specify the standard deviation (sigma) for Gaussian smoothing
sigma = 2; % Adjust this value as needed

% Apply Gaussian smoothing
smoothedImage = imgaussfilt(grayImage, sigma);

% Display the smoothed image
subplot(1, 2, 2);
imshow(smoothedImage);
title(['Gaussian Smoothed Image (sigma = ', num2str(sigma), ')']);

% Optional: Save the smoothed image
imwrite(smoothedImage, 'gaussian_smoothed_image.jpg');
```

11.Write a program to add random noise to an image and display the noisy image.

```
% Read the image
originalImage = imread('your_image.jpg');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
    grayImage = rgb2gray(originalImage);
else
    grayImage = originalImage;
end

% Display the original grayscale image
figure;
subplot(2, 2, 1);
imshow(grayImage);
title('Original Grayscale Image');
```

```
% Add Gaussian noise
noisyImageGaussian = imnoise(grayImage, 'gaussian', 0, 0.01); % mean=0, variance=0.01

% Display the image with Gaussian noise
subplot(2, 2, 2);
imshow(noisyImageGaussian);
title('Image with Gaussian Noise');

% Add salt & pepper noise
noisyImageSaltPepper = imnoise(grayImage, 'salt & pepper', 0.02); % noise density=0.02

% Display the image with salt & pepper noise
subplot(2, 2, 3);
imshow(noisyImageSaltPepper);
title('Image with Salt & Pepper Noise');

% Add speckle noise
noisyImageSpeckle = imnoise(grayImage, 'speckle', 0.04); % variance=0.04

% Display the image with speckle noise
subplot(2, 2, 4);
imshow(noisyImageSpeckle);
title('Image with Speckle Noise');

% Optional: Save the noisy images
imwrite(noisyImageGaussian, 'noisy_image_gaussian.jpg');
imwrite(noisyImageSaltPepper, 'noisy_image_salt_pepper.jpg');
imwrite(noisyImageSpeckle, 'noisy_image_speckle.jpg');
```

12.Write a program to crop a region of interest (ROI) from an image.

16.Write a program to apply Ideal Lowpass Filter (ILPF) and Ideal Highpass Filter (IHPF) for image smoothing

```
% Read the image
originalImage = imread('./image.png');

% Convert the image to grayscale if it is not already
if size(originalImage, 3) == 3
grayImage = rgb2gray(originalImage);
else
grayImage = originalImage;
end
% Display the original grayscale image
figure;
subplot(2, 3, 1);
imshow(grayImage);
title('Original Grayscale Image');
% Compute the Fourier Transform of the image
F = fft2(double(grayImage));
F_shifted = fftshift(F);
% Compute the magnitude spectrum of the Fourier Transform
magnitude_spectrum = log(1 + abs(F_shifted));
% Display the magnitude spectrum
subplot(2, 3, 2);
imshow(magnitude_spectrum, []);
title('Magnitude Spectrum');
```

```
% Define the cutoff frequency for the filters
cutoff_frequency = 50; % Adjust this value as needed
% Create ideal low pass filter (ILPF)
[rows, cols] = size(grayImage);
[x, y] = meshgrid(1:cols, 1:rows);
center_x = floor(cols / 2) + 1;
center_y = floor(rows / 2) + 1;
ILPF = double(sqrt((x - center_x).^2 + (y - center_y).^2) <= cutoff_frequency);
% Apply ILPF to the magnitude spectrum
filtered_image_ILPF = ifftshift(F_shifted .* ILPF);
filtered_image_ILPF = real(ifft2(filtered_image_ILPF));
% Display the image after applying ILPF
subplot(2, 3, 4);
imshow(filtered_image_ILPF, []);
title('Image after ILPF');
% Create ideal high pass filter (IHPF)
IHPF = 1 - ILPF;
% Apply IHPF to the magnitude spectrum
filtered_image_IHPF = ifftshift(F_shifted .* IHPF);
filtered_image_IHPF = real(ifft2(filtered_image_IHPF));
% Display the image after applying IHPF
subplot(2, 3, 5);
imshow(filtered_image_IHPF, []);
title('Image after IHPF');
% Optional: Save the filtered images
imwrite(uint8(filtered_image_ILPF), 'filtered_image_ILPF.jpg');
imwrite(uint8(filtered_image_IHPF), 'filtered_image_IHPF.jpg');
```