Matlab Programming

1. write a program for image enhancement . for matlab

% Read the input image

input\_image = imread('input\_image.jpg');

% Convert the image to grayscale if it's not already in grayscale if size(input\_image, 3) == 3

input\_image\_gray = rgb2gray(input\_image); else

input\_image\_gray = input\_image; end

% Perform histogram equalization output\_image = histeq(input\_image\_gray);

% Display the original and enhanced images

subplot(1, 2, 1), imshow(input\_image\_gray), title('Original Image');

subplot(1, 2, 2), imshow(output\_image), title('Enhanced Image');

% Save the enhanced image imwrite(output\_image, 'enhanced\_image.jpg');

1. write a program for image compression . for matlab

% Read the input image

input\_image = imread('input\_image.jpg');

% Convert the image to grayscale if it's not already in grayscale

if size(input\_image, 3) == 3

input\_image\_gray = rgb2gray(input\_image); else

input\_image\_gray = input\_image; end

% Set the compression quality (adjust as needed) compression\_quality = 80;

% Compress the image using JPEG compression

compressed\_image = imresize(input\_image\_gray, 0.5); % Resize for faster processing imwrite(compressed\_image, 'compressed\_image.jpg', 'Quality', compression\_quality);

% Display the original and compressed images

subplot(1, 2, 1), imshow(input\_image\_gray), title('Original Image');

subplot(1, 2, 2), imshow(compressed\_image), title('Compressed Image');

% Calculate compression ratio input\_image\_info = imfinfo('input\_image.jpg');

compressed\_image\_info = imfinfo('compressed\_image.jpg');

compression\_ratio = input\_image\_info.FileSize / compressed\_image\_info.FileSize; fprintf('Compression ratio: %.2f\n', compression\_ratio);

1. Write a program for color image processing (in MATLAB)

% Read the input color image

input\_image = imread('input\_color\_image.jpg');

% Display the original color image

subplot(2, 2, 1), imshow(input\_image), title('Original Color Image');

% Convert the color image to grayscale gray\_image = rgb2gray(input\_image);

% Display the grayscale image

subplot(2, 2, 2), imshow(gray\_image), title('Grayscale Image');

% Apply a median filter to reduce noise filtered\_image = medfilt2(gray\_image, [3, 3]);

% Display the filtered image

subplot(2, 2, 3), imshow(filtered\_image), title('Filtered Image');

% Perform edge detection using the Sobel operator edge\_image = edge(filtered\_image, 'Sobel');

% Display the edge-detected image

subplot(2, 2, 4), imshow(edge\_image), title('Edge-Detected Image');

% Save the processed images imwrite(gray\_image, 'grayscale\_image.jpg'); imwrite(filtered\_image, 'filtered\_image.jpg'); imwrite(edge\_image, 'edge\_detected\_image.jpg');

1. Write a program for image segmentation (in MATLAB)

% Read the input grayscale image input\_image = imread('input\_image.jpg');

% Convert the image to grayscale if it's not already in grayscale if size(input\_image, 3) == 3

input\_image\_gray = rgb2gray(input\_image); else

input\_image\_gray = input\_image; end

% Display the original grayscale image

subplot(1, 2, 1), imshow(input\_image\_gray), title('Original Grayscale Image');

% Apply thresholding to segment the image threshold\_value = 100; % Adjust threshold value as needed binary\_image = input\_image\_gray > threshold\_value;

% Display the segmented image

subplot(1, 2, 2), imshow(binary\_image), title('Segmented Image');

% Save the segmented image imwrite(binary\_image, 'segmented\_image.jpg');

1. Write a program for image morphology.(in MATLAB)

% Read the input binary image binary\_image = imread('binary\_image.jpg');

% Display the original binary image

subplot(2, 3, 1), imshow(binary\_image), title('Original Binary Image');

% Perform erosion operation

se = strel('disk', 5); % Define a disk-shaped structuring element eroded\_image = imerode(binary\_image, se);

% Display the eroded image

subplot(2, 3, 2), imshow(eroded\_image), title('Eroded Image');

% Perform dilation operation

dilated\_image = imdilate(binary\_image, se);

% Display the dilated image

subplot(2, 3, 3), imshow(dilated\_image), title('Dilated Image');

% Perform opening operation opened\_image = imopen(binary\_image, se);

% Display the opened image

subplot(2, 3, 4), imshow(opened\_image), title('Opened Image');

% Perform closing operation

closed\_image = imclose(binary\_image, se);

% Display the closed image

subplot(2, 3, 5), imshow(closed\_image), title('Closed Image');

% Perform boundary extraction

boundary\_image = binary\_image - eroded\_image;

% Display the boundary image

subplot(2, 3, 6), imshow(boundary\_image), title('Boundary Image');

% Save the processed images imwrite(eroded\_image, 'eroded\_image.jpg'); imwrite(dilated\_image, 'dilated\_image.jpg'); imwrite(opened\_image, 'opened\_image.jpg'); imwrite(closed\_image, 'closed\_image.jpg'); imwrite(boundary\_image, 'boundary\_image.jpg');