**QUESTIONS AND ANSWERS FOR IMAGE PROCESSING – NAAN MUDHALVAN**

1) **How to Load and Display an Image in MATLAB?**

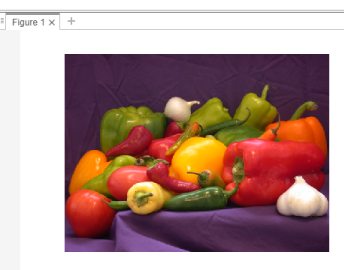
clc; clear all; close all;

% Load an image

img = imread('peppers.png');

% Display the image

imshow(img);



2)**How to Blur an Image in MATLAB?**

clc; clear all; close all;

% Load the image

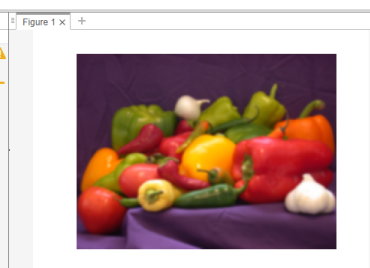
img = imread('peppers.png');

% Apply Gaussian blur

blurred\_img = imgaussfilt(img, 2);

% Display the blurred image

imshow(blurred\_img);

3)**How to Enhance Image Contrast using Histogram Equalization in MATLAB?**

clc; clear all; close all;

img = imread('peppers.png');

gray\_img = rgb2gray(img); % Convert to grayscale

filtered\_img = medfilt2(gray\_img); % Apply median filtering enhanced\_img = histeq(filtered\_img); % Perform histogram equalization subplot(1, 2, 1);

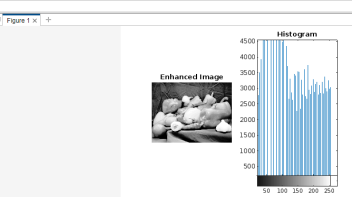
imshow(enhanced\_img);

title('Enhanced Image');

subplot(1, 2, 2);

imhist(enhanced\_img);

title('Histogram');



4) **How to Apply a Sobel Filter for Edge Detection in MATLAB?** clc; clear all; close all;

% Load the image

img = imread('peppers.png');

% Convert to grayscale

img\_gray = rgb2gray(img);

% Apply Sobel filter

edge\_img = edge(img\_gray,'sobel' );

% Display all three images

subplot(1,3,1);

imshow(img);

title('Original Image');

subplot(1,3,2);

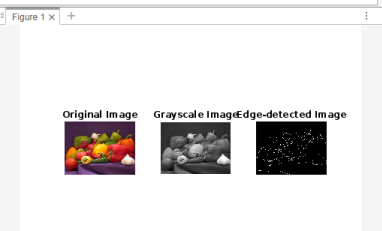
imshow(img\_gray);

title('Grayscale Image');

subplot(1,3,3);

imshow(edge\_img);

title('Edge-detected Image');



5)**How to Apply Fourier Transform to an Image in MATLAB?** clc; clear all; close all;

% Load the image

img = imread('peppers.png');

% Convert to grayscale

img\_gray = rgb2gray(img);

% Compute Fourier Transform

fft\_img = fft2(img\_gray);

% Display the magnitude spectrum

magnitude\_spectrum = log(1 + abs(fftshift(fft\_img))); imshow(magnitude\_spectrum, []);

% Print results

disp('Max value in magnitude spectrum:');

disp(max(magnitude\_spectrum(:)));

disp('Min value in magnitude spectrum:');

disp(min(magnitude\_spectrum(:)));



6)**How can you import an image, apply a Gaussian filter to it, print the mean value and standard deviation of the filtered image in MATLAB?**

clc; clear all; close all;

% Import the image

img = imread('peppers.png');

% Convert the image to grayscale

img\_gray = rgb2gray(img);

% Define Gaussian parameters

kernel\_size = 5; % Size of the kernel (odd number)

sigma = 2; % Standard deviation of the Gaussian distribution % Create Gaussian kernel

gaussian\_kernel = fspecial('gaussian', kernel\_size, sigma); % Apply Gaussian filter

filtered\_img = imfilter(img\_gray, gaussian\_kernel, 'conv', 'replicate'); % Display the filtered image

imshow(filtered\_img);

title('Filtered Image');

% Compute some features (e.g., mean and standard deviation) mean\_val = mean(filtered\_img(:));

std\_dev = std(double(filtered\_img(:)));

% Print the computed features

fprintf('Mean value of filtered image: %f\n', mean\_val);

fprintf('Standard deviation of filtered image: %f\n', std\_dev);

7)**How to Identify Lane Boundaries in an Image using MATLAB?**

clc; clear all; close all;

% Load the image

img = imread('peppers.png');

% Convert to grayscale

img\_gray = rgb2gray(img);

% Apply edge detection

edge\_img = edge(img\_gray, 'Canny');

% Identify lane boundaries using Hough transform [H,theta,rho] = hough(edge\_img);

peaks = houghpeaks(H,5);

lines = houghlines(edge\_img,theta,rho,peaks); % Draw the detected lines on the image

imshow(img), hold on

for k = 1:length(lines)

xy = [lines(k).point1; lines(k).point2];

plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green'); end

8)**How to Perform Image Filtering and defiltering in MATLAB?**

clc; clear all; close all;

% Load the image

img = imread('peppers.png');

% Define a filter

filter = fspecial('average', [5 5]);

% Apply the filter

filtered\_img = imfilter(img, filter);

% Display the filtered image

subplot(1, 2, 1);

imshow(filtered\_img);

title('Filtered Image');

% Define the inverse filter (deconvolution)

inv\_filter = fspecial('unsharp', 0.5);

% Apply the inverse filter (deconvolution)

deconvolved\_img = deconvwnr(filtered\_img, inv\_filter, 0.01); % Display the deconvolved image

subplot(1, 2, 2);

imshow(deconvolved\_img);

title('Deconvolved Image');



9) **How to Perform Image Thresholding in MATLAB?**

clc; clear all; close all;

% Read the image

img = imread('peppers.png');

% Convert to grayscale if necessary

if size(img, 3) == 3

img = rgb2gray(img);

end

% Define the threshold value

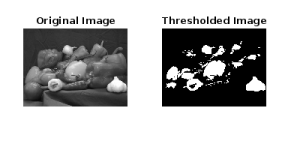
threshold\_value = 128;

% Perform thresholding

binary\_img = img > threshold\_value;

% Display the original and thresholded images

subplot(1,2,1), imshow(img), title('Original Image'); subplot(1,2,2), imshow(binary\_img), title('Thresholded Image');

10)**How to Perform Image Restoration ?**

clc; clear all; close all;

% Read the noisy image

noisy\_img = imread('peppers.png');

% Convert to grayscale if necessary

if size(noisy\_img, 3) == 3

noisy\_img = rgb2gray(noisy\_img);

end

% Define the size of the Gaussian filter

filter\_size = 5;

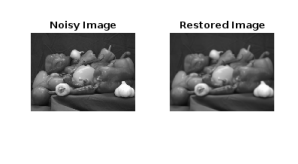
% Define the standard deviation of the Gaussian filter sigma = 2;

% Apply Gaussian filter for image restoration

restored\_img = imgaussfilt(noisy\_img, sigma);

% Display the original and restored images

subplot(1,2,1), imshow(noisy\_img), title('Noisy Image'); subplot(1,2,2), imshow(restored\_img), title('Restored Image');



11)**How to Import and Manipulate Excel Data in MATLAB?**

clc; clear all; close all;

% Import Excel data

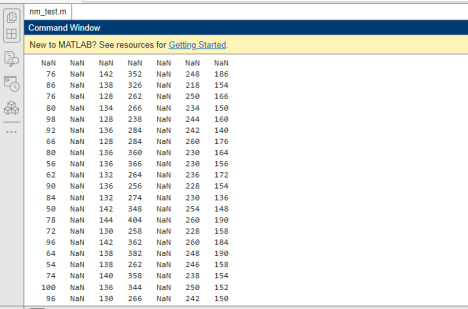
data = xlsread('patients.xls');

% Perform manipulation

manipulated\_data = data \* 2;

% Display the manipulated data

disp(manipulated\_data);



12)**How to Convert an Image from RGB to Grayscale in MATLAB?** clc; clear all; close all;

% Load the RGB image

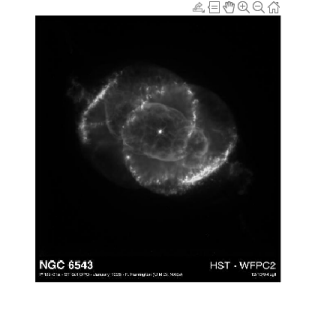
rgb\_img = imread('example.tif');

% Convert to grayscale

gray\_img = rgb2gray(rgb\_img);

% Display the grayscale image

imshow(gray\_img);



13)**How to Perform Image Segmentation using Otsu's Method in MATLAB?** clc; clear all; close all;

% Load the grayscale image

img = imread('peppers.png');

% Convert to grayscale if necessary

if size(img, 3) == 3

img = rgb2gray(img);

end

% Perform segmentation using Otsu's method

threshold = graythresh(img);

binary\_img = imbinarize(img, threshold);

% Display the segmented image

imshow(binary\_img);

14)**How to Perform Image Segmentation using Watershed Algorithm in MATLAB?** clc; clear all; close all;

% Load the grayscale image

img = imread('peppers.png');

% Convert to grayscale if necessary

if size(img, 3) == 3

img = rgb2gray(img);

end

% Compute gradient magnitude

gradient\_mag = imgradient(img);

% Perform watershed segmentation

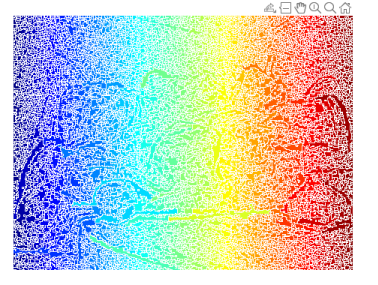
marker = imextendedmin(gradient\_mag, 0.2);

marker = imimposemin(-gradient\_mag, marker);

labels = watershed(marker);

% Display the segmented image

imshow(label2rgb(labels));



15)**How to Perform Template Matching in MATLAB?**

**clc; clear all; close all;**

**input = imread('peppers.png');**

**mask = imread('Capture.PNG');**

**gleam = @(I) (1/3) \* ((double(I(:,:,1))/255).^(1/2.2) + ... (double(I(:,:,2))/255).^(1/2.2) + ...**

**(double(I(:,:,3))/255).^(1/2.2));**

**normalize = @(I) (I-min(min(I)))/(max(max(I))-min(min(I))); g\_input = gleam(input);**

**g\_mask = gleam(mask);**

**gd\_input = conv2(g\_input, [-1 0 1; -1 0 1; -1 0 1]/3, 'same') + ... conv2(g\_input, [-1 -1 -1; 0 0 0; 1 1 1]/3, 'same');**

**gd\_mask = conv2(g\_mask, [-1 0 1; -1 0 1; -1 0 1]/3, 'same') + ... conv2(g\_mask, [-1 -1 -1; 0 0 0; 1 1 1]/3, 'same');**

**gd\_mask = rot90(gd\_mask, 2);**

**convolved = conv2(gd\_input, gd\_mask, 'valid');**

**[min\_value, min\_index] = min(convolved(:));**

**[min\_row, min\_col] = ind2sub(size(convolved), min\_index); figure; imshow(normalize(convolved)); hold on;**

**rectangle('Position', [min\_col-32+2 min\_row-32+2 64 64], 'EdgeColor', [1 0 0]);**

**figure; imshow(input\*0.75+64); hold on;**

**rectangle('Position', [min\_col+2 min\_row+2 64 64], ...**

**'EdgeColor', [0.5 0 0], ...**

**'LineWidth', 3, ...**

**'LineStyle', ':');**

****

****

16)**How to Perform Image Segmentation using Region Growing in MATLAB?** clc;

clear all;

close all;

gray1=imread('peppers.png'); %to read img 1

gray=rgb2gray(gray1)

figure();

imshow(gray)

title('Original Image');

% gray=rgb2gray(var);

so=edge(gray,'sobel'); %type -1 sobel edge detection ca=edge(gray,'canny'); %type -2 canny edge detection % subplot(3,1,2)

figure();

imshow(so);

title('SO Edge Detection');

% subplot(3,1,3)

figure();

imshow(ca);

title('ca Edge Detection');

17)**Create Simple Neural Network in MATLAB?** clc;

clear all;

close all;

% Define the input data

X = [0 0; 0 1; 1 0; 1 1]; % Input features

Y = [0; 1; 1; 0]; % Target outputs

% Create a feedforward neural network

net = feedforwardnet(10); % 10 neurons in the hidden layer % Train the neural network

net = train(net, X', Y');

% Test the neural network with new data

output = net(X');

% Display the output

disp('Predicted Output:');

disp(output);







18)**How to Perform Image Segmentation using K-Means Clustering in MATLAB?** clc;

clear all;

close all;

% Load the image

img = imread('peppers.png');

% Reshape the image into a 2D array

[rows, cols, ~] = size(img);

X = reshape(img, rows \* cols, []);

% Perform K-Means clustering

K = 3; % Number of clusters

[idx, centers] = kmeans(double(X), K);

% Reshape the clustered image

segmented\_img = reshape(idx, rows, cols);

% Display the segmented image

imshow(segmented\_img, []); colormap(gca, 'parula');

19)**How to Extract Features using Harris Corner Detection in MATLAB?**

clc;

clear all;

close all;

% Load the grayscale image

img = imread('peppers.png');

% Convert to grayscale

img\_gray = rgb2gray(img);

% Perform Harris corner detection

corners = detectHarrisFeatures(img\_gray);

% Display the detected corners

imshow(img);

hold on;

plot(corners);

20) **How to plot boundary box on object present in the image using MATLAB?** clc;

clear all;

close all;

% Read the image

image = imread('peppers.png');

% Convert the image to grayscale

image\_gray = rgb2gray(image);

% Perform edge detection

edges = edge(image\_gray, 'Canny');

% Perform connected component analysis

cc = bwconncomp(edges);

% Get bounding boxes of connected components

bounding\_boxes = regionprops(cc, 'BoundingBox');

% Draw bounding boxes on the original image

imshow(image);

hold on;

for i = 1:numel(bounding\_boxes)

rectangle('Position', bounding\_boxes(i).BoundingBox, 'EdgeColor', 'r', 'LineWidth', 2);

end

title('Bounding Box Detection');

hold off;

