

**Digital Image Processing (CSE-438)**

**Sec:03**

**Lab: 01**

**Submitted to:**

Md. Asif Khan Rifat

Lecturer  
Department of Computer Science & Engineering

East West University

**Submitted by:**

Prinom Mojumder

2021-2-60-098

1. Determine the perimeter of an object by using 4 connected neighborhoods and 8 connected neighborhoods.

Code:

I = imread('Picture1.png');

imshow(I);

BW = imbinarize(rgb2gray(I));

P4 = bwperim(BW, 4);

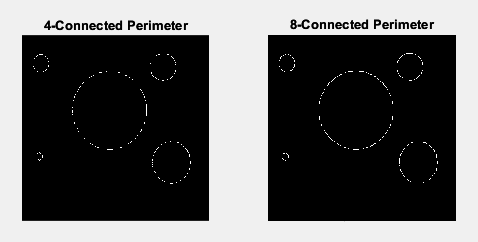
P8 = bwperim(BW, 8);

figure;

subplot(1,2,1), imshow(P4), title('4-Connected Perimeter');

subplot(1,2,2), imshow(P8), title('8-Connected Perimeter');

Output:



1. Create a binary image using a threshold.

Code:

img = imread('Picture2.png');

gray\_img = rgb2gray(img);

imshow(gray\_img);

binary\_img1 = imbinarize(gray\_img, 0.5); % Convert to binary with threshold 0.5

imshow(binary\_img1);

binary\_img2 = imbinarize(gray\_img, 0.3); % Convert to binary with threshold 0.5

imshow(binary\_img2);

binary\_img3 = imbinarize(gray\_img, 0.7); % Convert to binary with threshold 0.5

imshow(binary\_img3);

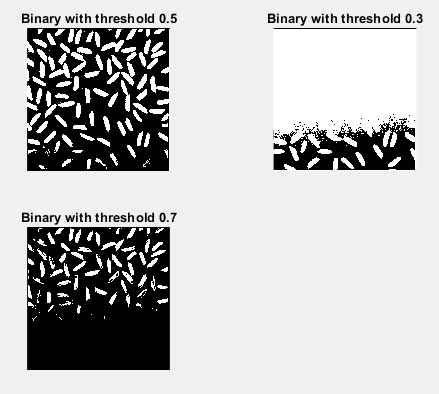
figure;

subplot(2,2,1), imshow(binary\_img1), title('Binary with threshold 0.5');

subplot(2,2,2), imshow(binary\_img2), title('Binary with threshold 0.3');

subplot(2,2,3), imshow(binary\_img3), title('Binary with threshold 0.7');

Output:



1. Determine the number of objects in the binary image generated in Question 2 using the

concept of connectivity.

Code:

img = imread('Picture2.png'); % Read image

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

binary\_img1 = imbinarize(gray\_img, 0.5); % Convert to binary with threshold 0.5

imshow(binary\_img1);

title('Binary Image');

cc4 = bwconncomp(binary\_img1, 4); % 4-connectivity finds objects considering diagonal connections

num\_objects4 = cc4.NumObjects; % Get number of objects

disp(['Number of objects in the binary image using 4-connectivity: ', num2str(num\_objects4)]);

cc8 = bwconncomp(binary\_img1, 8); % 8-connectivity finds objects considering diagonal connections

num\_objects8 = cc8.NumObjects; % Get number of objects

disp(['Number of objects in the binary image using 8-connectivity: ', num2str(num\_objects8)]);

Output:



1. Find the Euclidean distance between two points of the image.

Code:

img = imread('Picture2.png'); % Read image

imshow(img);

title('Select Two Points');

[x, y] = ginput(2); % Click two points in the image

distance = sqrt((x(2) - x(1))^2 + (y(2) - y(1))^2);

disp(['Euclidean Distance between selected points: ', num2str(distance)]);

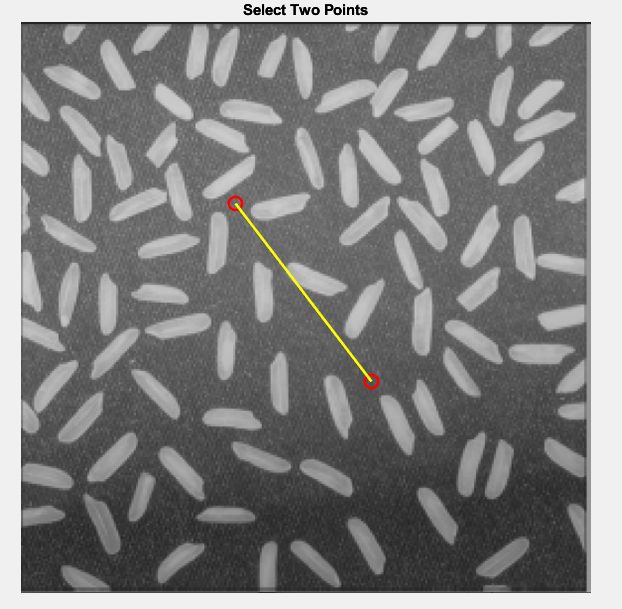
hold on;

plot(x, y, 'ro', 'MarkerSize', 10, 'LineWidth', 2);

line(x, y, 'Color', 'yellow', 'LineWidth', 2);

hold off;

Output:





1. Apply the following operations using Fig.1 and Fig.2:

a. Addition

b. Subtraction

c. Multiplication

d. Division

Code:

img1 = imread('Picture1.png');

img2 = imread('Picture2.png');

img1 = imresize(img1, [size(img2,1), size(img2,2)]);

addition = imadd(img1, img2); % Image addition

subtraction = imsubtract(img1, img2); % Image subtraction

multiplication = immultiply(img1, img2); % Image multiplication

division = imdivide(img1, img2); % Image division (avoid division by zero)

figure;

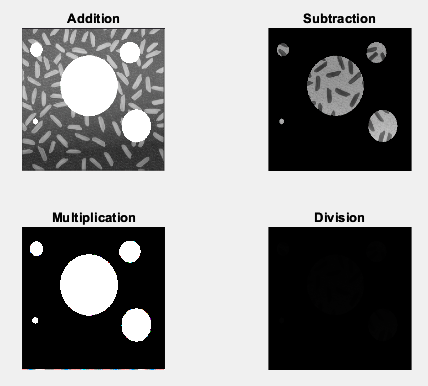
subplot(2,2,1), imshow(addition), title('Addition');

subplot(2,2,2), imshow(subtraction), title('Subtraction');

subplot(2,2,3), imshow(multiplication), title('Multiplication');

subplot(2,2,4), imshow(division, []), title('Division');

Output:



1. Apply the following operations using Fig.1 and Fig.2:

a. AND

b. OR

c. NOT

Code:

img1 = imread('Picture1.png');

img2 = imread('Picture2.png');

img1 = imresize(img1, [size(img2,1), size(img2,2)]);

if size(img1, 3) == 3

img1 = rgb2gray(img1);

end

if size(img2, 3) == 3

img2 = rgb2gray(img2);

end

bw1 = imbinarize(img1);

bw2 = imbinarize(img2);

and\_result = bw1 & bw2; % AND operation

or\_result = bw1 | bw2; % OR operation

not\_result = ~bw1; % NOT operation (on Picture1)

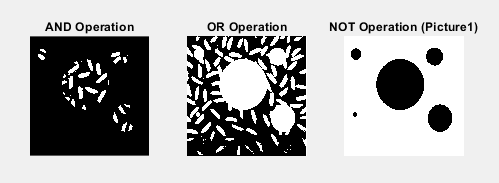
figure;

subplot(1,3,1), imshow(and\_result), title('AND Operation');

subplot(1,3,2), imshow(or\_result), title('OR Operation');

subplot(1,3,3), imshow(not\_result), title('NOT Operation (Picture1)');

Output:



1. Adjust the contrast of the following image.

Code:

img = imread('Picture3.jpg');

if size(img, 3) == 3

img = rgb2gray(img);

end

adjusted\_img = imadjust(img, stretchlim(img), []);

figure;

subplot(1,2,1), imshow(img), title('Original Image');

subplot(1,2,2), imshow(adjusted\_img), title('Contrast Adjusted Image');

Output:

A comparison of a landscape

AI-generated content may be incorrect.

1. Brighten the following image

Code:

img = imread('Picture4.jpg');

brightened\_img = imadjust(img, [], [0.3 1]);

figure;

subplot(1,2,1), imshow(img), title('Original Image');

subplot(1,2,2), imshow(brightened\_img), title('Brightened Image');

Output:

A comparison of a picture of a forest

AI-generated content may be incorrect.

1. Quantize the Grayscale image by 8 levels.

Code:

img = imread('Picture5.jpg');

% Number of quantization levels

num\_levels = 8;

% Quantize the image

quantized\_img = floor(double(img) / (256 / num\_levels)) \* (256 / num\_levels);

% Display the original and quantized images

figure;

subplot(1,2,1), imshow(img), title('Original Grayscale Image');

subplot(1,2,2), imshow(uint8(quantized\_img)), title('Quantized Image (8 Levels)');

Output:

A comparison of coins with a scale

AI-generated content may be incorrect.

1. Find the digital negative of the image.

Code:

img = imread('Picture6.jpg');

negative\_img = 255 - img;

figure;

subplot(1,2,1), imshow(img), title('Original Grayscale Image');

subplot(1,2,2), imshow(negative\_img), title('Digital Negative Image');

Output:

A close-up of a brain

AI-generated content may be incorrect.