

East West University

**Course title:** Digital Image Processing

**Course code:** CSE438

Section: 02

**Semester:** Summer 23

**Submitted to**

**Instructor:**

Dr. Ahmed Wasif Reza

Professor, Department of CSE

Additional Director, Institutional Quality Assurance Cell (IQAC)

Submitted by:

**Name:** Apurba Roy Ajay

**Id:** 2018-3-60-063

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

**Codes:**

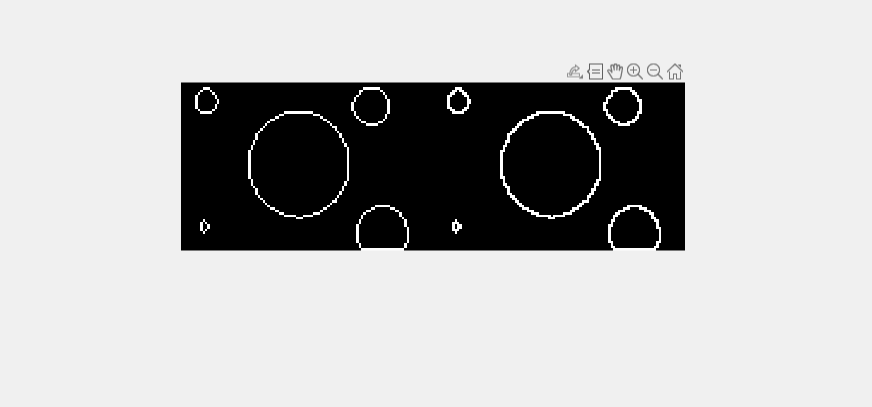
BW=imread('img1.png');

BW2=bwperim(x,4);

BW3=bwperim(x,8);

imshowpair(BW2,BW3,'montage')

**Output:**



**2.** **Create a binary image using threshold.**

**Code:**

I = imread('img2.png');

BW = imbinarize(I, 'adaptive');

figure

imshowpair(I,BW,'montage')

**Output:**

A picture containing pattern, art, sketch, design

Description automatically generated

**3.** **Determine the number of objects in the binary image generated in Question 2 using the concept of connectivity.**

**Codes:**

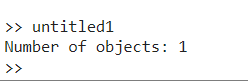
binaryImage = imread('img2.png');

connComp = bwconncomp(binaryImage);

numObjects = connComp.NumObjects;

fprintf('Number of objects: %d\n', numObjects);

**Output:**

****

**4. Find the Euclidean distance between two points of the image.**

**Code:**

I = imread('img1.png');

I = rgb2gray(I);

h = imhist(I); % this will have default bins 256

% now second image

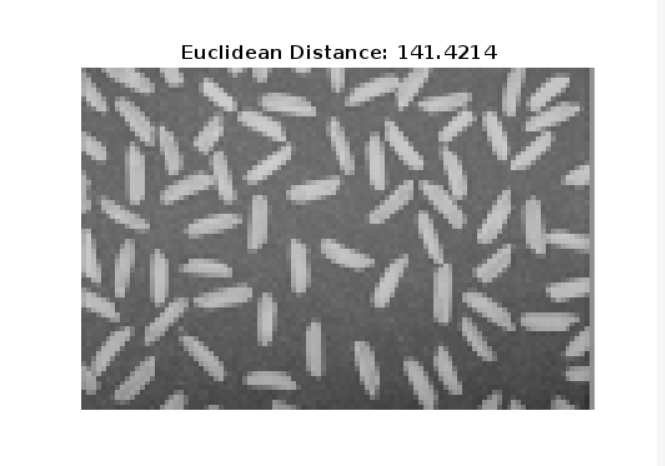
J = imread('img2.png');

J = rgb2gray(J);

h1 = imhist(J); % this will have default bins 256

E\_distance = sqrt(sum((h-h1).^2));

**Output:**

****

**5. Apply the following operations using img1 and img2:**

**a. Addition**

**b. Subtraction**

**c. Multiplication**

**d. Division**

**Code:**

img1 = imread('img1.png');

img2 = imread('img2.png');

img1 = double(img1);

img2 = double(img2);

addition\_result = img1 + img2;

subtraction\_result = img1 - img2;

multiplication\_result = img1 .\* img2;

division\_result = img1 ./ img2;

figure;

subplot(2, 2, 1), imshow(uint8(addition\_result)), title('Addition');

subplot(2, 2, 2), imshow(uint8(subtraction\_result)), title('Subtraction');

subplot(2, 2, 3), imshow(uint8(multiplication\_result)), title('Multiplication');

subplot(2, 2, 4), imshow(uint8(division\_result)), title('Division');

**Output:**

A screenshot of a computer

Description automatically generated

**6. Apply the following operations using img1 and img2:**

**a. AND**

**b. OR**

**c. NOT**

**Codes:**

image1 = imread('img1.png');

image2 = imread('img2.png');

result\_and = logical(image1) & logical(image2);

result\_and\_gray = uint8(result\_and) \* 255;

subplot(1, 3, 1);

imshow(result\_and\_gray);

title('Logical AND');

result\_or = logical(image1) | logical(image2);

result\_or\_gray = uint8(result\_or) \* 255;

subplot(1, 3, 2);

imshow(result\_or\_gray);

title('Logical OR');

result\_not = ~logical(image1);

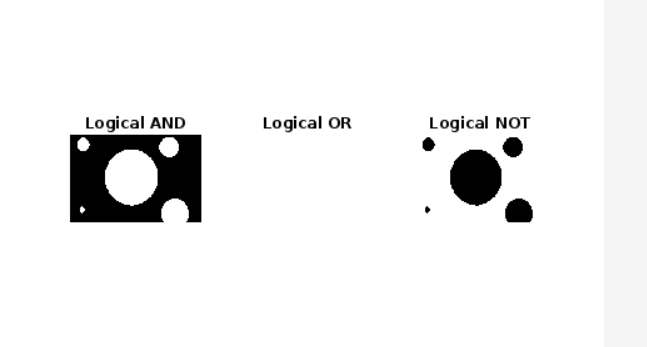
result\_not\_gray = uint8(result\_not) \* 255;

subplot(1, 3, 3);

imshow(result\_not\_gray);

title('Logical NOT');

**Output:**

****

**7.Find the digital negative of the image.**

**Codes:**

skItn = imread("img3.png");

subplot(1, 2, 1),

imshow(skItn);

title("Original image");

Level = 2 ^ 8;

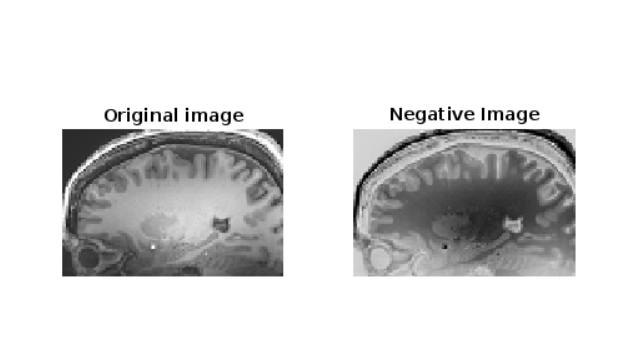
neg = (Level - 1) - skItn;

subplot(1, 2, 2),

imshow(neg);

title("Negative Image")

**Output:**

****

**8.Apply bit plane slicing on the image.**

**Codes:**

c = imread('img4.jpg');

cd = double(c);

c1 = mod(cd, 2);

c2 = mod(floor(cd/2), 2);

c3 = mod(floor(cd/4), 2);

c4 = mod(floor(cd/8), 2);

c5 = mod(floor(cd/16), 2);

c6 = mod(floor(cd/32), 2);

c7 = mod(floor(cd/64), 2);

c8 = mod(floor(cd/128), 2);

cc = (2 \* (2 \* (2 \* (2 \* (2 \* (2 \* (2 \* c8 + c7) + c6) + c5) + c4) + c3) + c2) + c1);

subplot(2, 5, 1);

imshow(c);

title('Original Image');

subplot(2, 5, 2);

imshow(c1);

title('Bit Plane 1');

subplot(2, 5, 3);

imshow(c2);

title('Bit Plane 2');

subplot(2, 5, 4);

imshow(c3);

title('Bit Plane 3');

subplot(2, 5, 5);

imshow(c4);

title('Bit Plane 4');

subplot(2, 5, 6);

imshow(c5);

title('Bit Plane 5');

subplot(2, 5, 7);

imshow(c6);

title('Bit Plane 6');

subplot(2, 5, 8);

imshow(c7);

title('Bit Plane 7');

subplot(2, 5, 9);

imshow(c8);

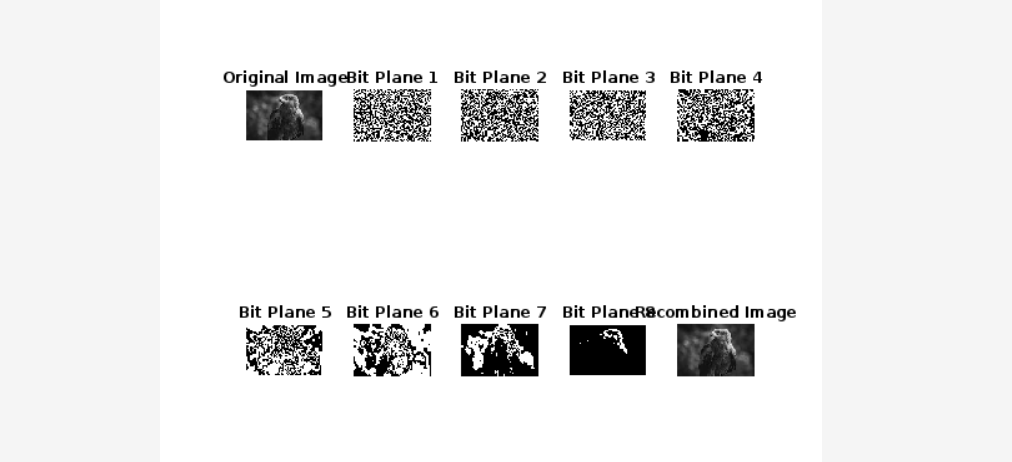
title('Bit Plane 8');

subplot(2, 5, 10);

imshow(uint8(cc));

title('Recombined Image');

**Output:**

****

**9. Use contrast stretching on the image.**

**Codes:**

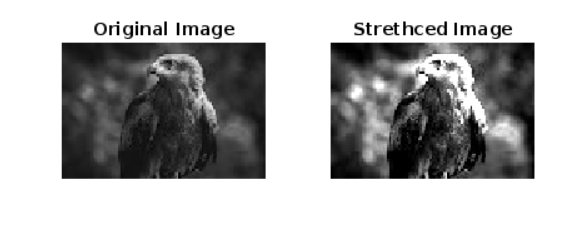
image = imread('img4.jpg');

stretched\_Image = imadjust(image, stretchlim(image, [0.05, 0.95]),[]);

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

subplot(2,2,2), imshow(stretched\_Image), title('Strethced Image');

**Output:**

****

**10.** **Change the contrast of the image using Logarithmic Transformation and Power-law Transformation.**

**Codes:**

input\_image = imread('img2.png');

input\_image = im2double(input\_image);

c = 1;

gamma = 0.5;

output\_image = c \* input\_image.^gamma;

output\_image = mat2gray(output\_image);

figure;

subplot(1, 2, 1);

imshow(input\_image);

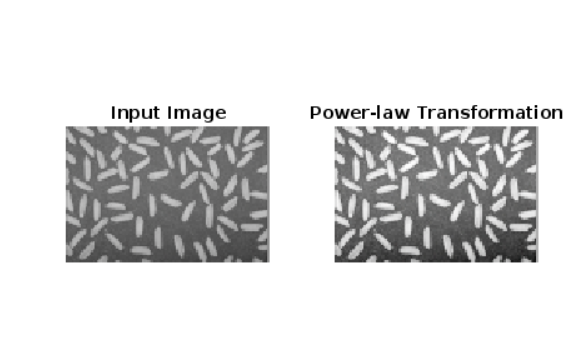
title('Input Image');

subplot(1, 2, 2);

imshow(output\_image);

title('Power-law Transformation');

imwrite(output\_image, 'output\_image\_power.jpg');

**Output: **