EAST WEST UNIVERSITY

Department of Computer Science and Engineering

Lab-report:03

Course Name: Digital Image Processing
Course Code: CSE438
Section No: 03

Submitted To:

Prof. Dr. Ahmed Wasif Reza

Department of Computer Science and Engineering

East West University

Submitted by:

Student's ID: 2020-3-60-012

Student's Name: Sadia Islam Prova

Date of submission: 5-3-24

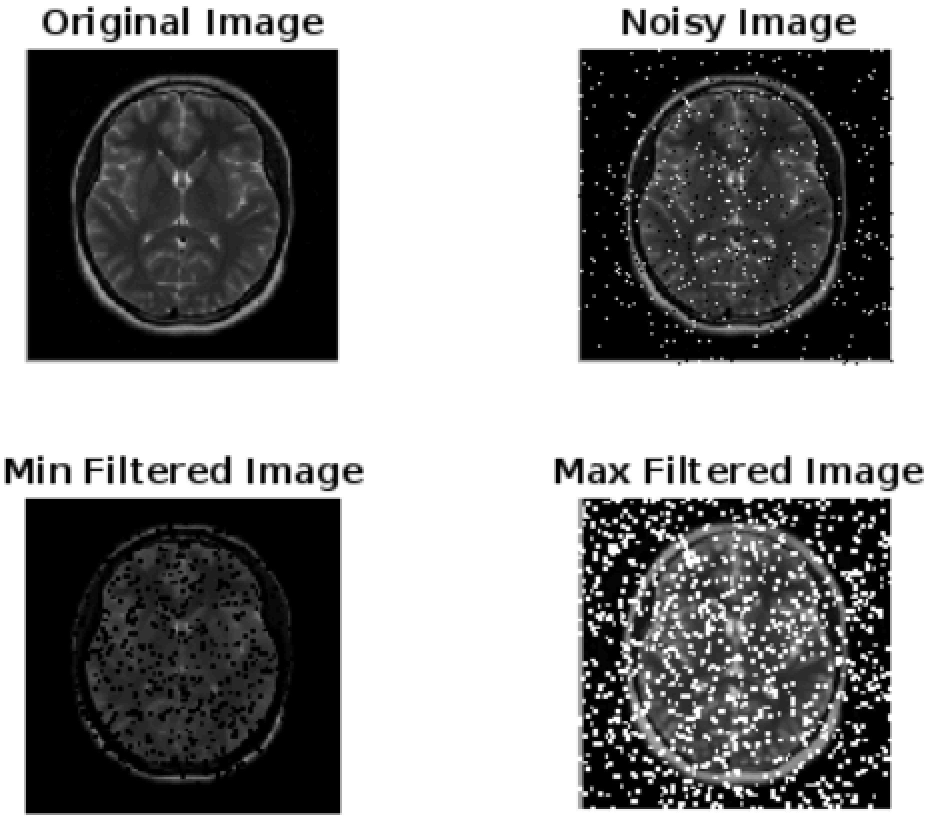
Problem 1:

Apply salt and pepper noise to the following image and remove the noise using min and max filtering technique. Show input and output side by side.

Code:

```
original_img = imread('img1.PNG');


gray_img = rgb2gray(original_img);


gray_img = im2double(gray_img);


noisy_img = imnoise(gray_img, 'salt & pepper', 0.05);


min_filtered_img = ordfilt2(noisy_img, 1, true(3));
max_filtered_img = ordfilt2(noisy_img, 9, true(3));


figure;

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

subplot(2, 2, 2);
imshow(noisy_img);
title('Noisy Image');

subplot(2, 2, 3);
imshow(min_filtered_img);
title('Min Filtered Image');

subplot(2, 2, 4);
imshow(max_filtered_img);
title('Max Filtered Image');
```

Output:



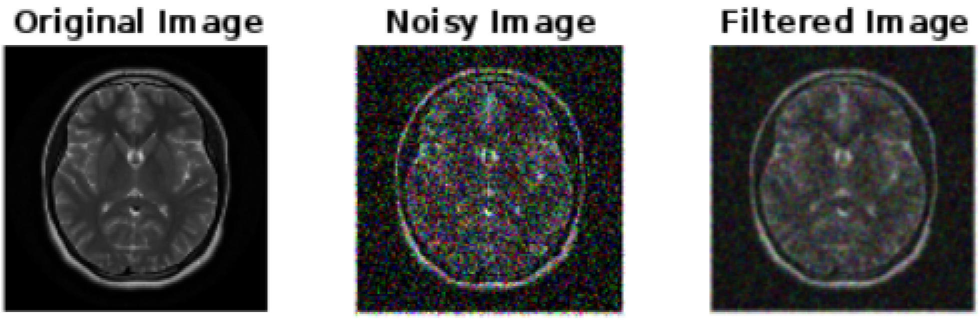
Problem 2:
Apply Gaussian noise to the following image and remove the noise using Gaussian filtering. Show input and output side by side.

Code:

```
original_img = imread('img1.PNG');


gray_img = im2double(original_img);


sigma = 0.2;
noisy_img = imnoise(gray_img, 'gaussian', 0, sigma^2);


filtered_img = imgaussfilt(noisy_img, 5*sigma);
```

```
figure;

subplot(1, 3, 1);
imshow(gray_img);
title('Original Image');

subplot(1, 3, 2);
imshow(noisy_img);
title('Noisy Image');

subplot(1, 3, 3);
imshow(filtered_img);
title('Filtered Image');
```

Output:



Problem 3:

Apply any noise to the following image and restore it using:

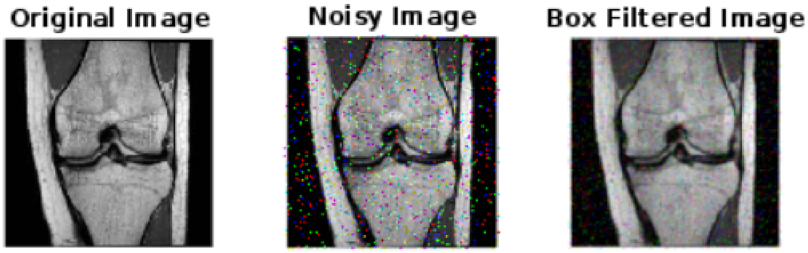
a) Box filtering
b) Average filtering
c) Median filtering
Show input and output side by side. Also show the comparison between the 3 techniques.
Mention which method works better than others.

Box filtering:

Code:

```
original_img = imread('img2.PNG');


gray_img = im2double(original_img);


noisy_img = imnoise(gray_img, 'salt & pepper', 0.05);

box_filtered_img = imboxfilt(noisy_img, 3);

figure;

subplot(1, 3, 1);
imshow(gray_img);
title('Original Image');

subplot(1, 3, 2);
imshow(noisy_img);
title('Noisy Image');

subplot(1, 3, 3);
imshow(box_filtered_img);
title('Box Filtered Image');
```

Output:



Original Image

Noisy Image

Box Filtered Image

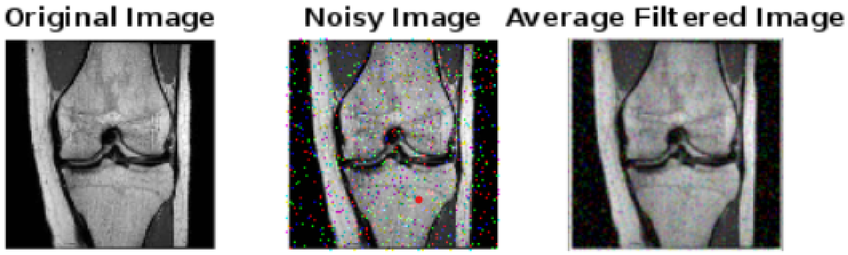
Average filtering:

Code:

```
original_img = imread('img2.PNG');

gray_img = im2double(original_img);
```

```
noisy_img = imnoise(gray_img, 'salt & pepper', 0.05);

filter_size = 3;
average_filtered_img = imfilter(noisy_img, fspecial('average', [filter_size
filter_size]));

figure;

subplot(1, 3, 1);
imshow(gray_img);
title('Original Image');

subplot(1, 3, 2);
imshow(noisy_img);
title('Noisy Image');

subplot(1, 3, 3);
imshow(average_filtered_img);
title('Average Filtered Image');
```

Output:



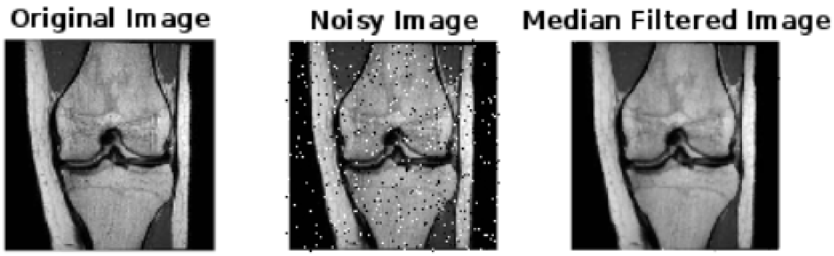
<u>Median filtering:</u>

Code:

```
original_img = imread('img2.PNG');

gray_img = rgb2gray(original_img);


gray_img = im2double(gray_img);

noisy_img = imnoise(gray_img, 'salt & pepper', 0.05);

filter_size = 3;
median_filtered_img = medfilt2(noisy_img, [filter_size filter_size]);
```

```
figure;

subplot(1, 3, 1);
imshow(gray_img);
title('Original Image');

subplot(1, 3, 2);
imshow(noisy_img);
title('Noisy Image');

subplot(1, 3, 3);
imshow(median_filtered_img);
title('Median Filtered Image');
```

Output:



Median filtering is more effective in removing noise while preserving image details, especially in the presence of salt-and-pepper noise.

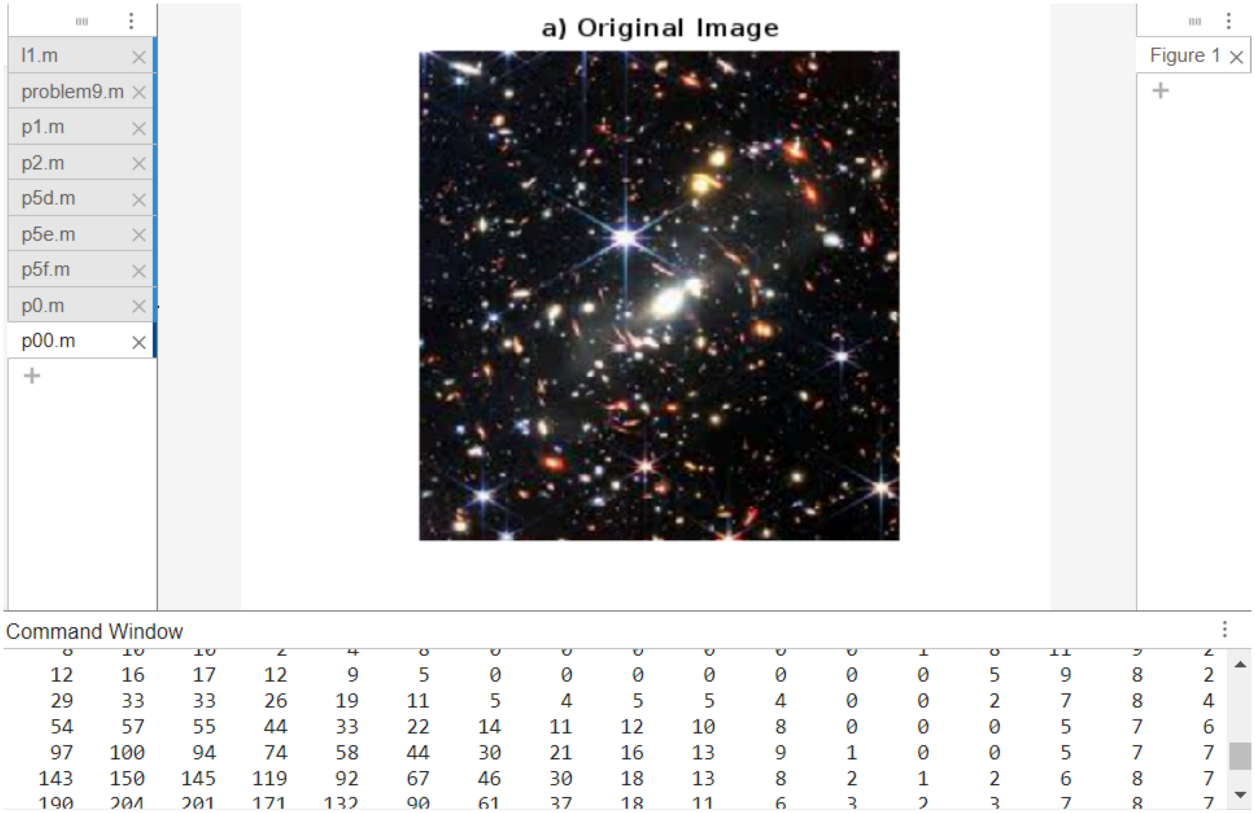
Problem 4:

Code:

```
% a) Read and show the image
image = imread('3.png');
imshow(image);
title('a) Original Image');
% b) Show the matrix form of the image
disp('b) Matrix form of the image:');
disp(image);
% c) Show the pixel information by hovering the cursor on the image
% was not able to solve this
% d) Find the value of the pixel (10, 78)
```

```
pixel_value = image(10, 78);
disp(['d) Pixel value at (10, 78): ', num2str(pixel_value)]);
% e) Show the size of the image
image_size = size(image);
disp(['e) Size of the image: ', num2str(image_size(1)), ' x ', num2str(image_size(2))]);
% f) Show all the information of the image
disp('f) All infor,mation of the image');
whos image;
```

Output:



```
     8     10     10      2      4      8      0      0      0      0      0      0      1      8     11      9      2
    12     16     17     12      9      5      0      0      0      0      0      0      0      5      9      8      2
    29     33     33     26     19     11      5      4      5      5      4      0      0      2      7      8      4
    54     57     55     44     33     22     14     11     12     10      8      0      0      0      5      7      6
    97    100     94     74     58     44     30     21     16     13      9      1      0      0      5      7      7
   143    150    145    119     92     67     46     30     18     13      8      2      1      2      6      8      7
   190    204    201    171    132     90     61     37     18     11      6      3      2      3      7      8      7
```

Problem 5:

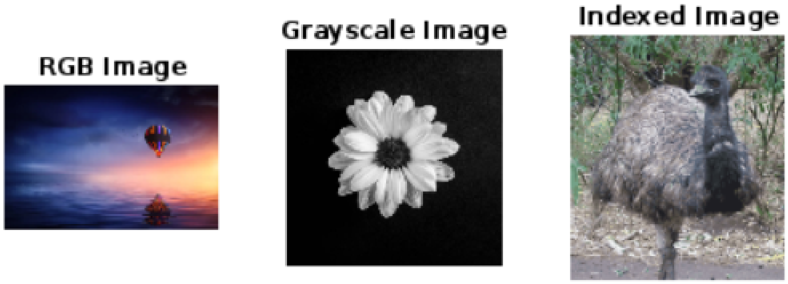
a)Read and show all three types of images (RGB, Grayscale, and Indexed).

Code:

```
rgb_img = imread('rgb.png');

gray_img = imread('gray.png');
```

```
indexed_img = imread('indexed.png');

figure;
subplot(1, 3, 1);
imshow(rgb_img);
title('RGB Image');


subplot(1, 3, 2);
imshow(gray_img);
title('Grayscale Image');

subplot(1, 3, 3);
imshow(indexed_img);
title('Indexed Image');
```

Output:

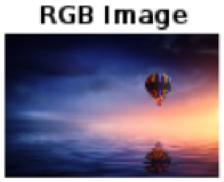


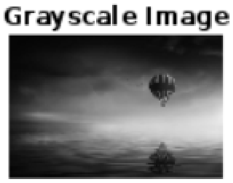
b) Turn the RGB image to Grayscale image.

Code:

```
rgb_img = imread('rgb.png');


gray_img = rgb2gray(rgb_img);

imshow(gray_img);
title('Grayscale Image');
```

Output:



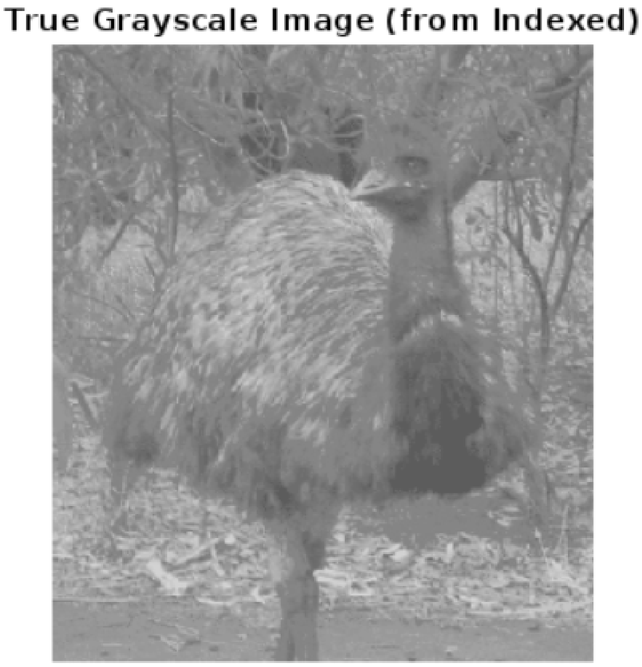


c) Turn the Indexed image to Grayscale image.

Code:

```
indexed_img = imread('indexed.png');


figure;
imshow(indexed_img);
title('Indexed Image');


gray_img = ind2gray(indexed_img, colormap);


true_gray_img = rgb2gray(gray_img);

figure;
imshow(true_gray_img);
title('True Grayscale Image (from Indexed)');
```

Output:



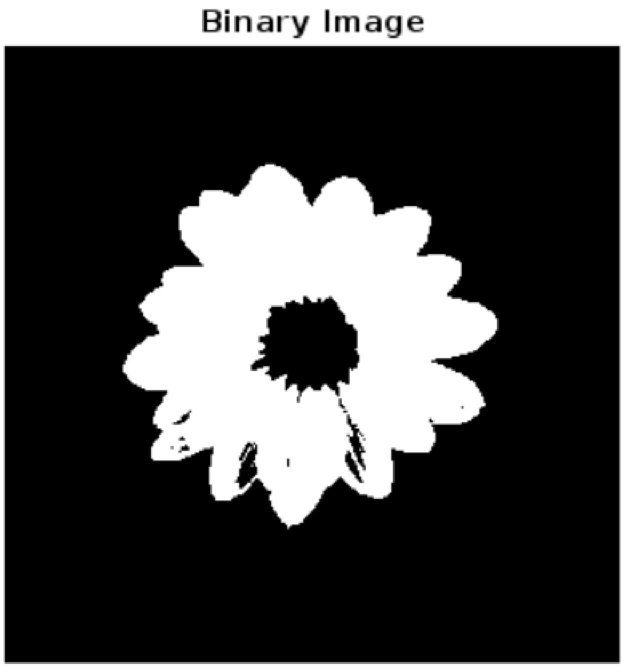
True Grayscale Image (from Indexed)

e) Convert the Grayscale image to a Binary image.

Code:

```
gray_img = imread('gray.png');


threshold = graythresh(gray_img);
binary_img = imbinarize(gray_img, threshold);

binary_img_uint8 = uint8(binary_img) * 255;

imshow(binary_img_uint8);
title('Binary Image');
```

Output:



f) Show the inverted form of that Binary image.

Code:

```
% Read the grayscale image
gray_img = imread('gray.png');

% Convert the grayscale image to binary using thresholding
threshold = graythresh(gray_img); % Determine threshold automatically using Otsu's
method
binary_img = imbinarize(gray_img, threshold);

% Invert the binary image
inverted_binary_img = ~binary_img;

% Convert the inverted binary image to uint8 format for display
inverted_binary_img_uint8 = uint8(inverted_binary_img) * 255;

% Display the inverted binary image
imshow(inverted_binary_img_uint8);
title('Inverted Binary Image');
```

Output:

Inverted Binary Image