EAST WEST UNIVERSITY

Department of Computer Science and Engineering

Lab-report:04

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: 12-3-24

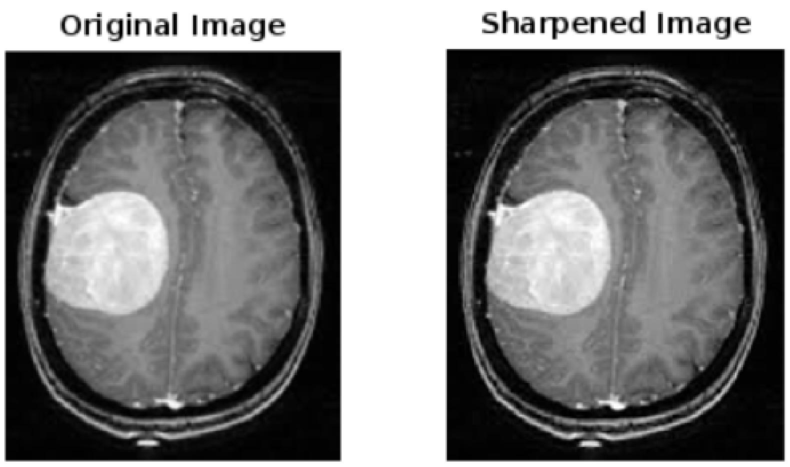
Problem 1: Sharpen the following image by applying the following and find out which one is better:

a) Unsharp Masking

Code:

```
gray_img = imread('img 4.1.png');

sharpened_img = imsharpen(gray_img);

% Display the original and sharpened images side by side
figure;
subplot(1, 2, 1);
imshow(gray_img);
title('Original Image');

subplot(1, 2, 2);
imshow(sharpened_img);
title('Sharpened Image');
```

Output:



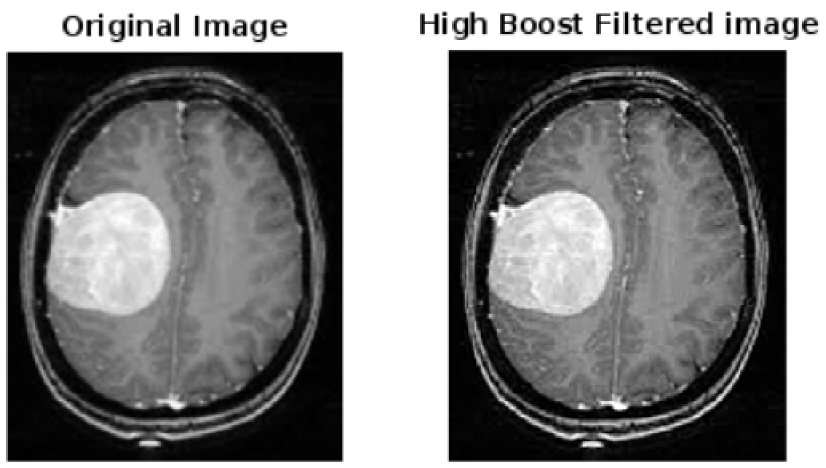
b) High Boost Filtering

Code:

```
gray_img = imread('img 4.1.png');

gray_img = im2double(gray_img);

% high boost filter parameters
k = 2.5;
```

```
mask_size = 5;


sigma = 1;
h = fspecial('gaussian', mask_size, sigma);

low_pass_img = imfilter(gray_img, h, 'replicate');

high_pass_img = gray_img - low_pass_img;

sharpened_img = gray_img + k * high_pass_img;

sharpened_img = max(0, min(sharpened_img, 1));

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

subplot(1, 2, 2);
imshow(sharpened_img);
title('High Boost Filtered image');
```

Output:



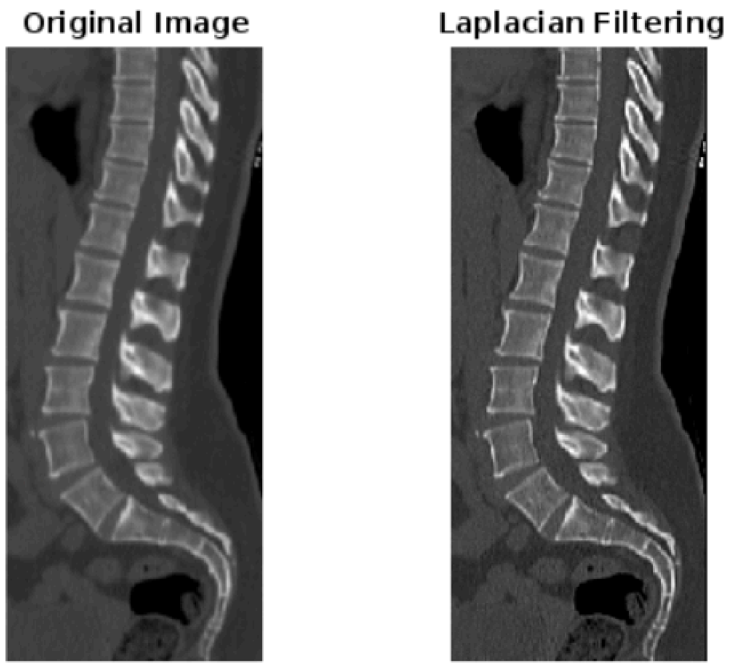
After comparing the output, it can be said that the High Boost filtering is better than the unsharp masking.

Problem 2: Sharpen the following image using the concept of Laplacian Filtering.

Code:

```
gray_img = imread('img 4.2.png');

gray_img = im2double(gray_img);

laplacian_kernel = [0 -1 0; -1 4 -1; 0 -1 0];

laplacian_filtered_img = imfilter(gray_img, laplacian_kernel, 'replicate');

sharpened_img = gray_img + laplacian_filtered_img;

sharpened_img = max(0, min(sharpened_img, 1));

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

subplot(1, 2, 2);
imshow(sharpened_img);
title('Laplacian Filtering');
```

Output:

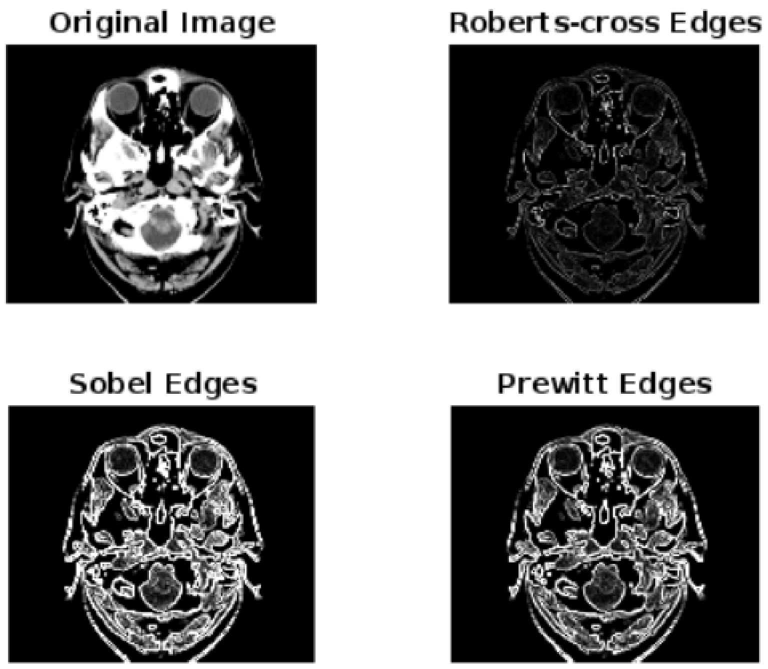


Problem 3: Use Roberts-cross, Sobel, and Prewitt operators to detect the edge of the following image.

Code:

```
gray_img = imread('img 4.3.png');

gray_img = im2double(gray_img);

% Roberts-cross operator
roberts_x = [1 0; 0 -1];
roberts_y = [0 1; -1 0];
roberts_x_img = abs(imfilter(gray_img, roberts_x, 'replicate'));
roberts_y_img = abs(imfilter(gray_img, roberts_y, 'replicate'));
roberts_edges = sqrt(roberts_x_img.^2 + roberts_y_img.^2);

% Sobel operator
sobel_x = [-1 0 1; -2 0 2; -1 0 1];
sobel_y = [-1 -2 -1; 0 0 0; 1 2 1];
sobel_x_img = abs(imfilter(gray_img, sobel_x, 'replicate'));
sobel_y_img = abs(imfilter(gray_img, sobel_y, 'replicate'));
sobel_edges = sqrt(sobel_x_img.^2 + sobel_y_img.^2);

% Prewitt operator
prewitt_x = [-1 0 1; -1 0 1; -1 0 1];
prewitt_y = [-1 -1 -1; 0 0 0; 1 1 1];
prewitt_x_img = abs(imfilter(gray_img, prewitt_x, 'replicate'));
prewitt_y_img = abs(imfilter(gray_img, prewitt_y, 'replicate'));
prewitt_edges = sqrt(prewitt_x_img.^2 + prewitt_y_img.^2);

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

subplot(2, 2, 2);
imshow(roberts_edges, []);
title('Roberts-cross Edges');

subplot(2, 2, 3);
imshow(sobel_edges, []);
title('Sobel Edges');

subplot(2, 2, 4);
imshow(prewitt_edges, []);
title('Prewitt Edges');
```

Output:



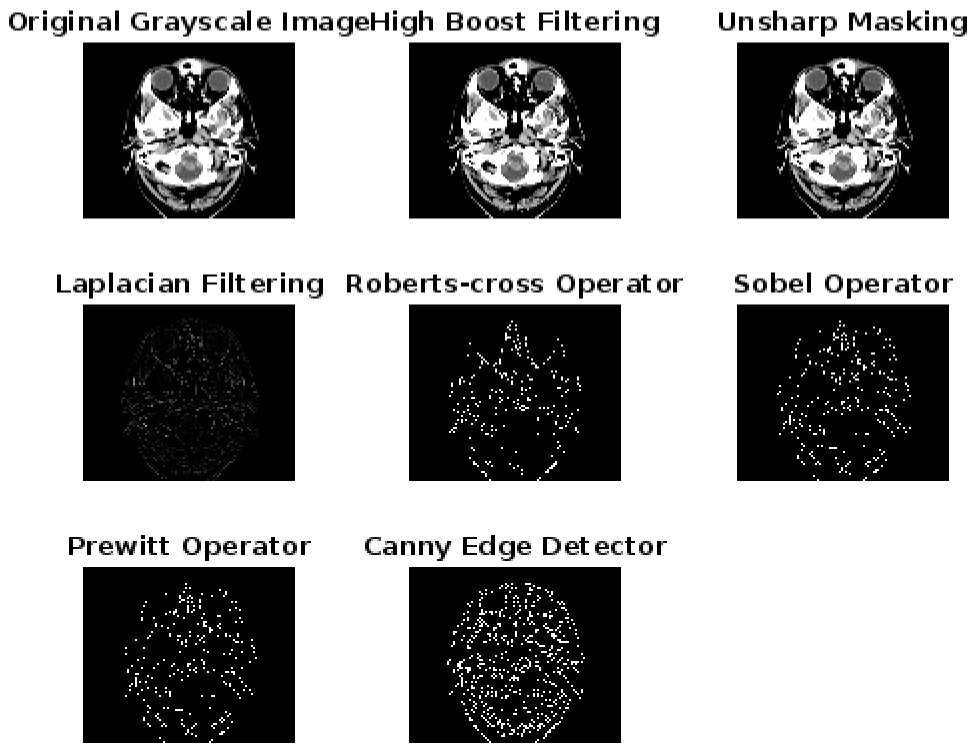
Problem 4: Show performance comparison among High Boost, Unsharp, Laplacian Roberts-cross, Sobel, Prewitt and Canny filtering for edge detection – find out which one is better for the given image.

Code:

```
gray_img = imread('img 4.3.png');

if size(gray_img, 3) == 3
    gray_img = rgb2gray(gray_img);
end

% Convert the image to double precision for calculations
gray_img = im2double(gray_img);

% High Boost filtering
k_high_boost = 1.5; % Boosting factor
sharpened_high_boost = gray_img + k_high_boost * (gray_img - imfilter(gray_img,
fspecial('average', 3), 'replicate'));

%Unsharp masking
sharpened_unsharp = imsharpen(gray_img);

%Laplacian filtering
laplacian_filtered_img = imfilter(gray_img, fspecial('laplacian', 0.5));
```

```
%Roberts-cross operator
edges_roberts = edge(gray_img, 'Roberts');

%Sobel operator
edges_sobel = edge(gray_img, 'Sobel');

%Prewitt operator
edges_prewitt = edge(gray_img, 'Prewitt');

%Canny edge detector
edges_canny = edge(gray_img, 'Canny');


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

subplot(3, 3, 2);
imshow(sharpened_high_boost);
title('High Boost Filtering');

subplot(3, 3, 3);
imshow(sharpened_unsharp);
title('Unsharp Masking');

subplot(3, 3, 4);
imshow(laplacian_filtered_img);
title('Laplacian Filtering');

subplot(3, 3, 5);
imshow(edges_roberts);
title('Roberts-cross Operator');

subplot(3, 3, 6);
imshow(edges_sobel);
title('Sobel Operator');

subplot(3, 3, 7);
imshow(edges_prewitt);
title('Prewitt Operator');

subplot(3, 3, 8);
imshow(edges_canny);
title('Canny Edge Detector');
```

Output:



After comparing the output, it can be said that the Canny Edge Detector is better than the other techniques for edge detection.