

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

**Sec:03**

**Lab: 04**

**Submitted to:**

Md. Asif Khan Rifat

Lecturer  
Department of Computer Science & Engineering

East West University

**Submitted by:**

Prinom Mojumder

2021-2-60-098

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

a) Unsharp Masking

Code:

original = imread('Picture1.jpg');

original = im2double(original); % Convert to double

if size(original, 3) == 3

original\_gray = rgb2gray(original);

else

original\_gray = original;

end

h = fspecial('gaussian', [5 5], 1); % Gaussian blur kernel

blurred = imfilter(original\_gray, h, 'replicate');

mask = original\_gray - blurred;

unsharp\_img = original\_gray + mask;

figure('Name','Unsharp Masking Result','NumberTitle','off');

subplot(1, 2, 1);

imshow(original\_gray);

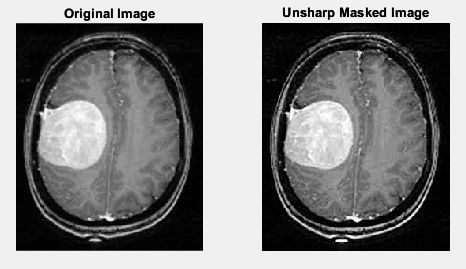
title('Original Image');

subplot(1, 2, 2);

imshow(unsharp\_img);

title('Unsharp Masked Image');

Output:



b) High Boost Filtering

Code:

original = imread('Picture1.jpg');

original = im2double(original);

if size(original, 3) == 3

original\_gray = rgb2gray(original);

else

original\_gray = original;

end

h = fspecial('gaussian', [5 5], 1);

blurred = imfilter(original\_gray, h, 'replicate');

mask = original\_gray - blurred;

A = 1.5;

high\_boost\_img = original\_gray + A \* mask;

figure('Name','High Boost Filtering','NumberTitle','off');

subplot(1, 2, 1);

imshow(original\_gray);

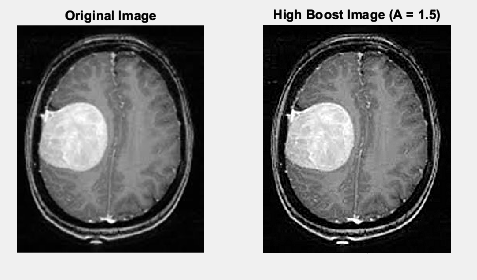
title('Original Image');

subplot(1, 2, 2);

imshow(high\_boost\_img);

title(['High Boost Image (A = ', num2str(A), ')']);

Output:



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

Code:

original = imread('Picture2.jpg');

original = im2double(original);

if size(original, 3) == 3

original\_gray = rgb2gray(original);

else

original\_gray = original;

end

laplacian\_filter = fspecial('laplacian', 0.2);

laplacian\_img = imfilter(original\_gray, laplacian\_filter, 'replicate');

sharpened\_img = original\_gray - laplacian\_img;

figure('Name','Laplacian Filtering','NumberTitle','off');

subplot(1, 2, 1);

imshow(original\_gray);

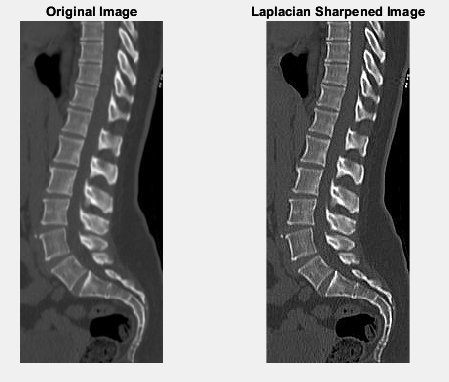
title('Original Image');

subplot(1, 2, 2);

imshow(sharpened\_img);

title('Laplacian Sharpened Image');

Output:



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

image.

Code:

original = imread('Picture3.jpg');

original = im2double(original);

if size(original, 3) == 3

original\_gray = rgb2gray(original);

else

original\_gray = original;

end

edge\_roberts = edge(original\_gray, 'roberts');

edge\_sobel = edge(original\_gray, 'sobel');

edge\_prewitt = edge(original\_gray, 'prewitt');

figure('Name','Edge Detection using Different Operators','NumberTitle','off');

subplot(2, 2, 1);

imshow(original\_gray);

title('Original Image');

subplot(2, 2, 2);

imshow(edge\_roberts);

title('Roberts Operator');

subplot(2, 2, 3);

imshow(edge\_sobel);

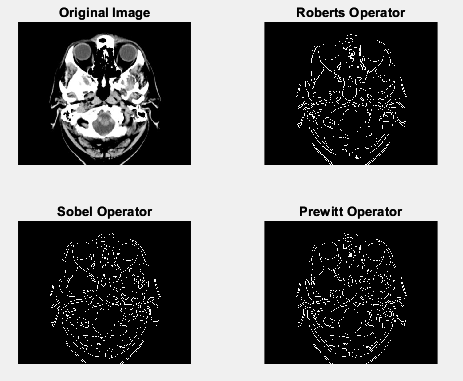
title('Sobel Operator');

subplot(2, 2, 4);

imshow(edge\_prewitt);

title('Prewitt Operator');

Output:



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:

original = imread('Picture3.jpg');

original = im2double(original);

if size(original, 3) == 3

original\_gray = rgb2gray(original);

else

original\_gray = original;

end

h = fspecial('gaussian', [5 5], 1);

blurred = imfilter(original\_gray, h, 'replicate');

mask = original\_gray - blurred;

unsharp\_img = original\_gray + mask;

unsharp\_sobel = edge(unsharp\_img, 'sobel');

A = 1.5;

high\_boost\_img = original\_gray + A \* mask;

highboost\_sobel = edge(high\_boost\_img, 'sobel');

laplacian\_filter = fspecial('laplacian', 0.2);

laplacian\_img = original\_gray - imfilter(original\_gray, laplacian\_filter, 'replicate');

laplacian\_sobel = edge(laplacian\_img, 'sobel');

edge\_roberts = edge(original\_gray, 'roberts');

edge\_sobel = edge(original\_gray, 'sobel');

edge\_prewitt = edge(original\_gray, 'prewitt');

edge\_canny = edge(original\_gray, 'canny');

figure;

subplot(3, 3, 1); imshow(original\_gray); title('Original');

subplot(3, 3, 2); imshow(unsharp\_sobel); title('Unsharp + Sobel');

subplot(3, 3, 3); imshow(highboost\_sobel); title('High Boost + Sobel');

subplot(3, 3, 4); imshow(laplacian\_sobel); title('Laplacian + Sobel');

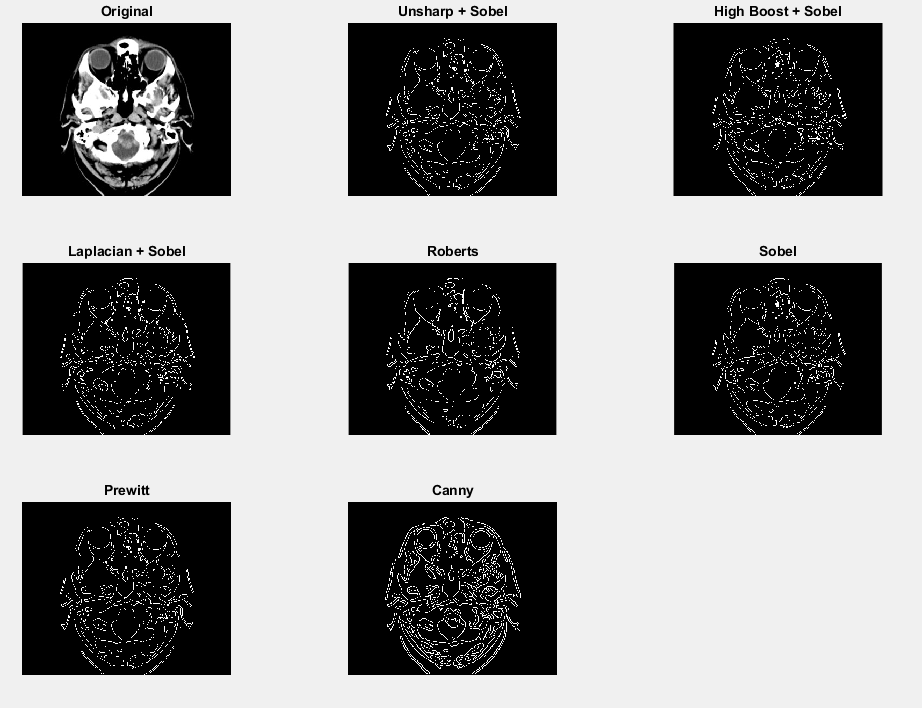
subplot(3, 3, 5); imshow(edge\_roberts); title('Roberts');

subplot(3, 3, 6); imshow(edge\_sobel); title('Sobel');

subplot(3, 3, 7); imshow(edge\_prewitt); title('Prewitt');

subplot(3, 3, 8); imshow(edge\_canny); title('Canny');

Output:



**Observation:**  
Canny

* Best overall: clean, accurate, continuous edges.
* Includes Gaussian smoothing → low noise.
* High performance for object boundaries.