EE 417 POST-LAB REPORT #2

Smoothing, Sharpening and 1st Derivative Filters

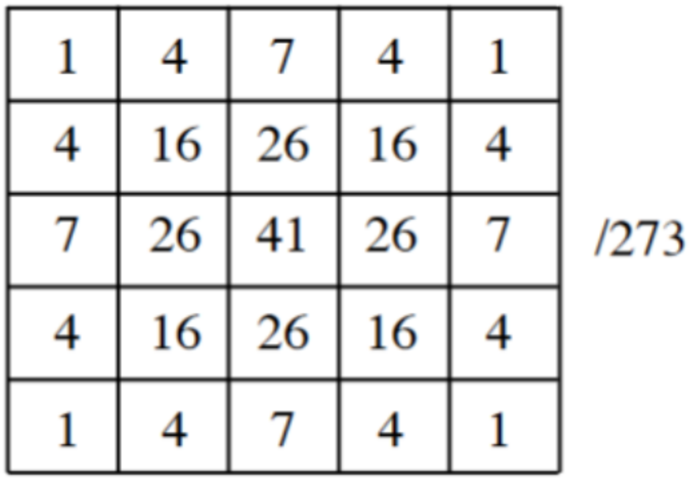
Osman Burak Çevik
osmanburak@sabanciuniv.edu
26399

Introduction

EE417 Lab #2 focuses on implementation of smoothing, sharpening and first derivative filters to grayscale images by using MATLAB software.

Linear Filtering

Gaussian filtering is a linear operation that involves the local convolution of a given image with a filter kernel of samples of the 2D Gaussian function. Gaussian filtering is used for smoothing the pictures by eliminating "outliers" in image values which are considered to be noise in a given context. In order to apply Gaussian smoothing, the kernel below is used to convolve it with the image to be filtered.



1	4	7	4	1	
4	16	26	16	4	
7	26	41	26	7	/273
4	16	26	16	4	
1	4	7	4	1	

Figure-1: 5x5 Kernel used in Gaussian filtering function

The function I wrote on MATLAB takes an image as input and returns the "Gaussian smoothed" version of the image. 5x5 kernel shown in *Figure-1* is used for convolution.

"lab2gaussfilt.m" function takes the input image "img" and calculates the row, column and channels of the image. If the channel is 3 which means the image is rgb, tha image is converted to grayscale version by using rgb2gray.

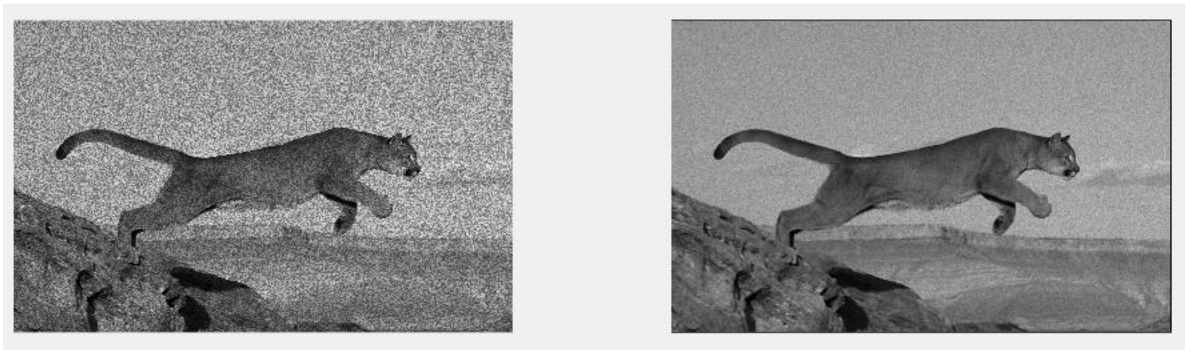
A matrix called "Gaussian_matrix" is created with the 5x5 kernel values in *figure-1* and multiplied with (1/273). "273" is the sum of the values in the kernel.

Using zeros function a matrix with the same size as the grayscale image is created, the matrix has same dimensions but the values are zero.

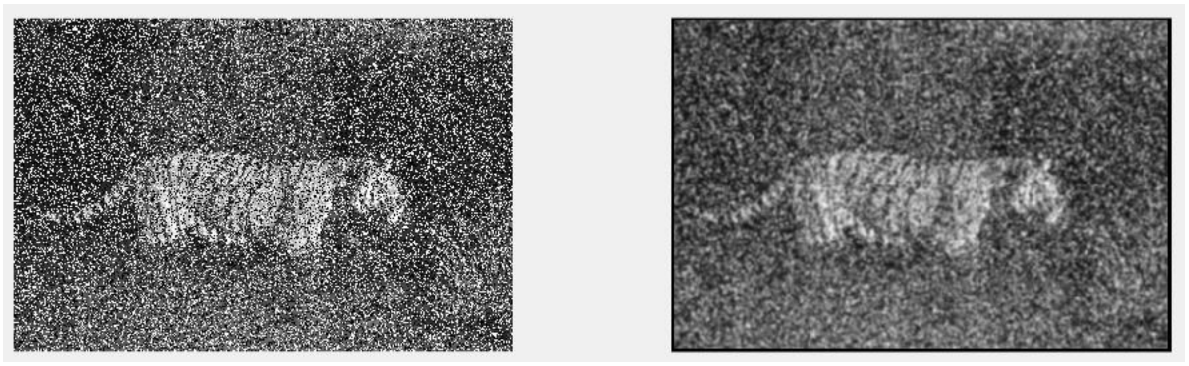
Nested for loops are created to travel around the image's pixels. "The window" misses some pixels because the processed pixel is in the center of the window and therefore the missed pixels are 0 values, which is black. The window traveling the image calculates the pixel value of the convolution and saves the value to "Filtered_img". This way the processed image is created.

Images, both the original and filtered, are turned to uint8 and the figures are called in the main.m part.

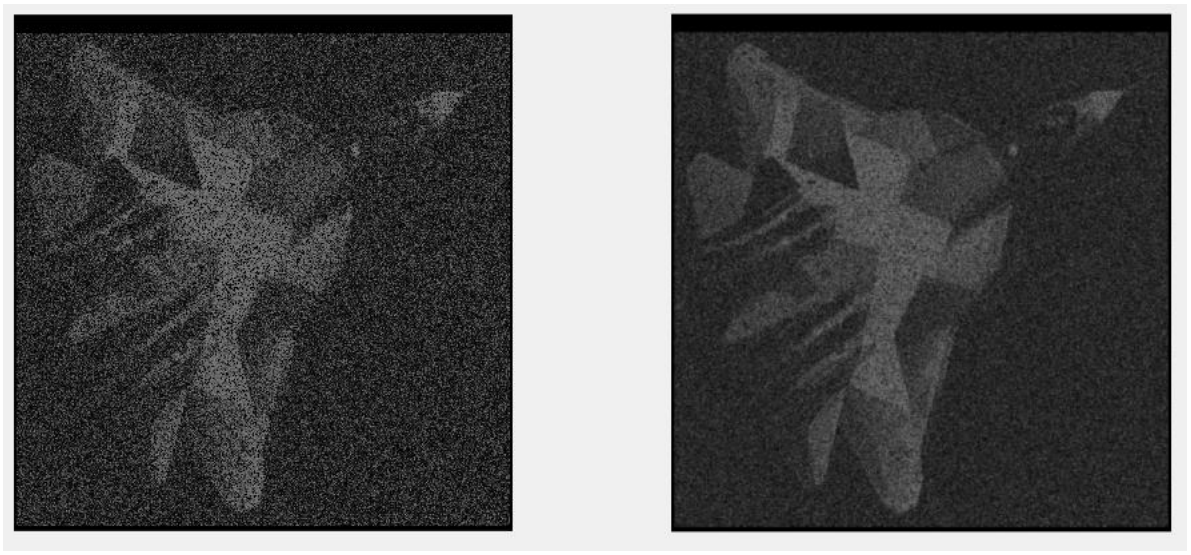
Results for lab2gaussfilt.m with jump.png(1,577x986):



Results for lab2gaussfilt.m with tiger.png(384x256):



Results for lab2gaussfilt.m with gaussNoise.png(675x698):



Observations and Conclusion for Linear Filtering(Gaussian Smoothed):

The noise of the images is reduced. The transition between the pixels are smoothen.

Nonlinear Filtering

Median filtering is a nonlinear operation since it involves sorting and picking the median element. Due to nonlinearity, median filter can not be implemented as a convolution.

The function takes an image “img” and a constant k as inputs and returns the “median filtered” version of the image img, “ResImg”.

The function first calculates the row, column and channels of the image. If the channel is 3 which means the image is rgb, the image is converted to grayscale version by using rgb2gray.

The grayscale “img” is converted to double for implementation of the operations. Using zeros function a matrix with the same size as the grayscale image is created, the matrix has same dimensions but the values are zero.

The gaussian smoothed image “GaussianFiltered” is created by using the lab2gaussfilt() function.

Nested for loops are created to travel around the image’s pixels. “The window” misses some pixels because the processed pixel is in the center of the window and therefore the missed pixels are 0 values, which is black. The window traveling the image calculates the median of the pixels by using myMedian() function.

myMedian() function first calculates the row, column and channels of the image. After the calculation cardinality of the function is calculates by multiplying rows and columns. If the channel is 3 which means the image is rgb, the image is converted to grayscale version by using rgb2gray.

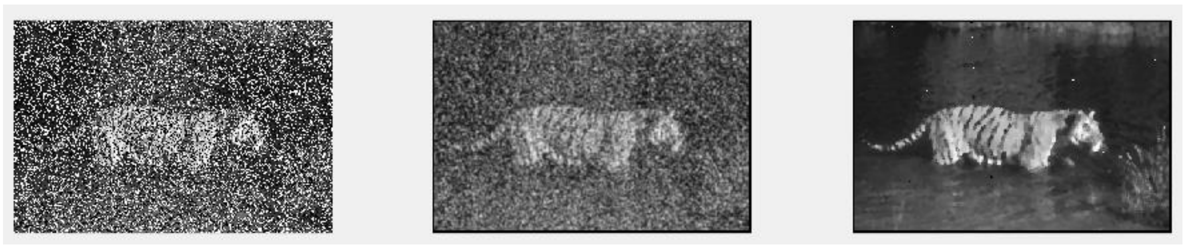
Then the image is converted to double for implementation of the operations. “Data” array is created with the size of 1*cardinality. Then the sort() function is used for placing the pixel in order according to their values.

The algorithm finds the index of median element of the even sized array by mod calculation, odd sized array by basic mathematical calculation. Using the index, the median element is found.

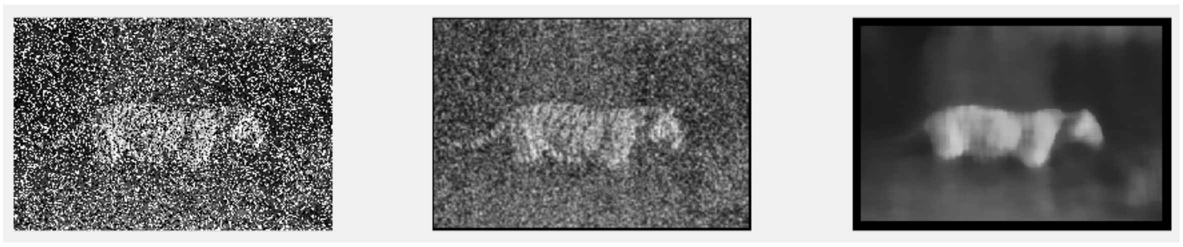
The pixel’s median values are found according to the size of the sliding window which is proportional with k.

Filtered image is returned and plotted. The image on the left is the original image, middle one is the Gaussian smoothed and the image on the right is median filtered.

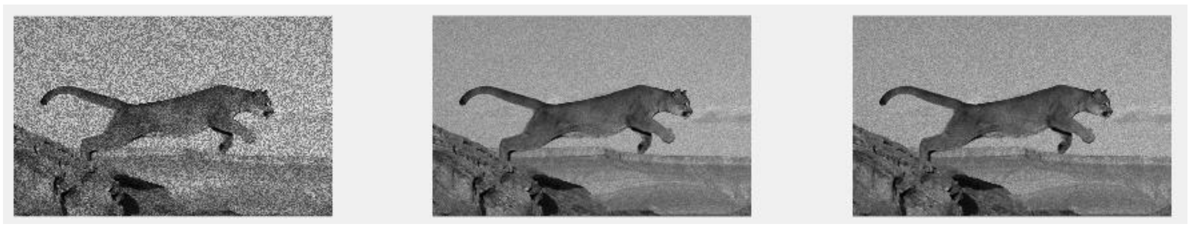
Results for lab2medfilt.m with Tiger.png(384x256), k=2:



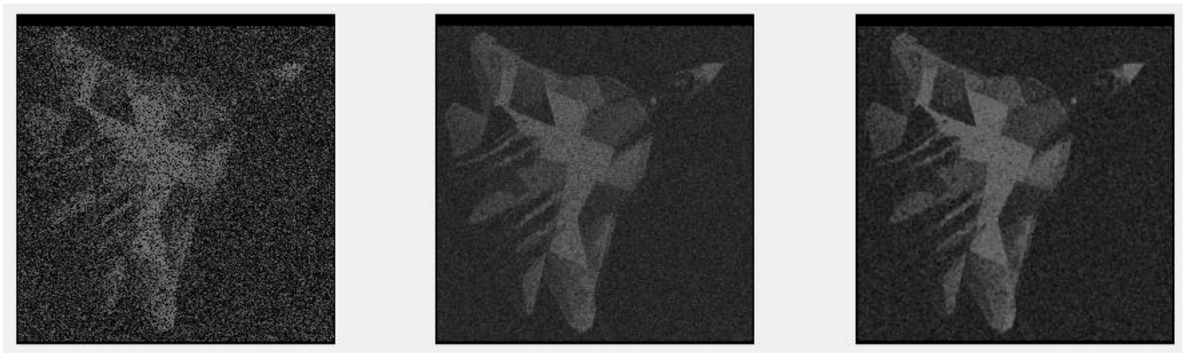
Results for lab2medfilt.m with Tiger.png(384x256), k=10:



Results for lab2medfilt.m with jump.png(1,577x986), k=2:



Results for lab2medfilt.m with gaussNoise.png(675x698), k=2:



Observations and Conclusion for Noninear Filtering(Median Filtered):

The function creates a window with the size of (2k+1)x(2k+1). The filter then maps the median of the window to the refernce pixel. The smoothening removes the outliers and changes the contrast.

Increasing k makes the image smoothed and blurred and the frame(0 valued pixels, left from unprocessed pixels due to k) get bigger.

Sharpening

The aim of the sharpening operation is to produce an enhanced image J by increasing the contrast of the given image I along edges, without adding too much noise within homogenous regions in the image. Sharpening can be implemented as shown in the figure below:

$$J(p) = I(p) + \lambda \Big[I(p) - S(p) \Big]$$

Figure-2

where S is the smoothed version of the given image I and limbda>0 is a scaling factor which controls the influence of the correction signal.

The function takes an image, a constant limbda and an integer M as inputs and returns the “sharpened” version of the image. M is used for smoothing method selection. Options are: box filter, Gaussian filter and median filter.

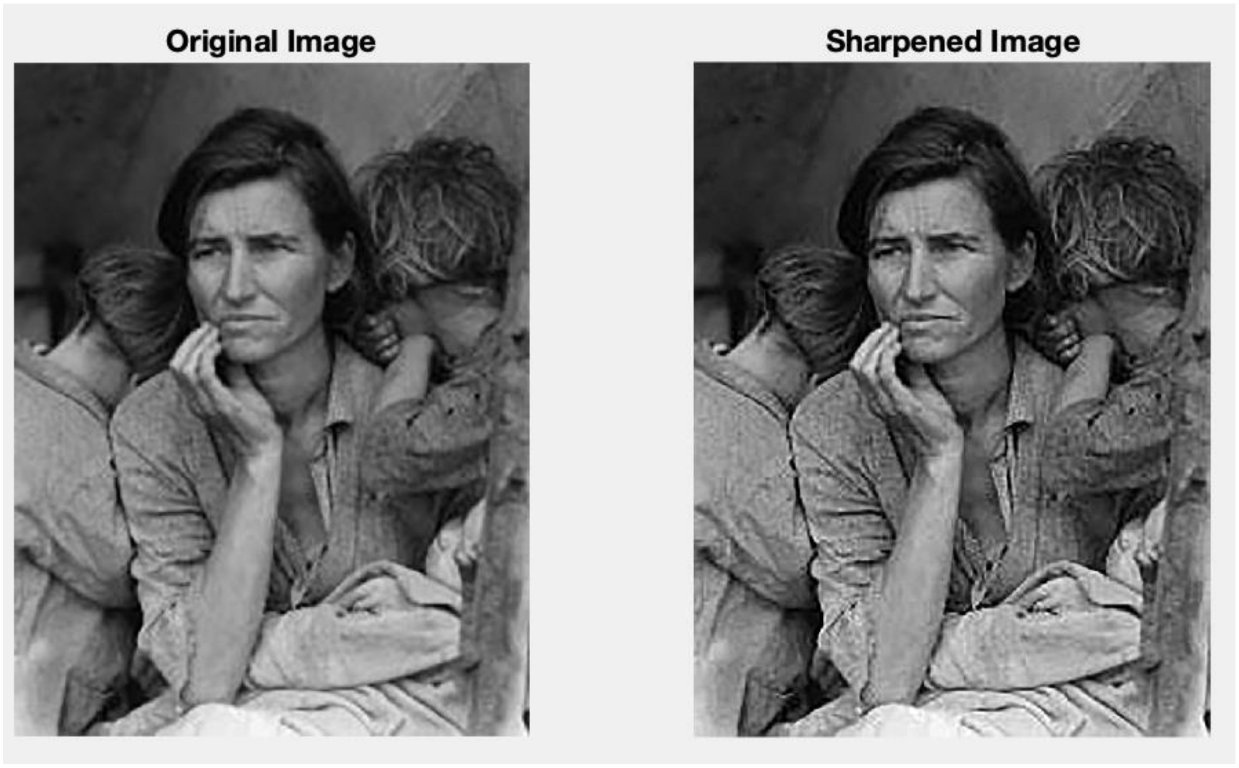
The function takes the input image “img” and calculates the row, column and channels of the image. If the channel is 3 which means the image is rgb, tha image is converted to grayscale version by using rgb2gray.

Then the M(mode) selection part comes. 1,2 and 3 are the only accepted values for mode. Locbox, gaussfilt and medfilt are the corresponding values to 1,2 and 3. The fsmoothed image is called as “Smoothed”.

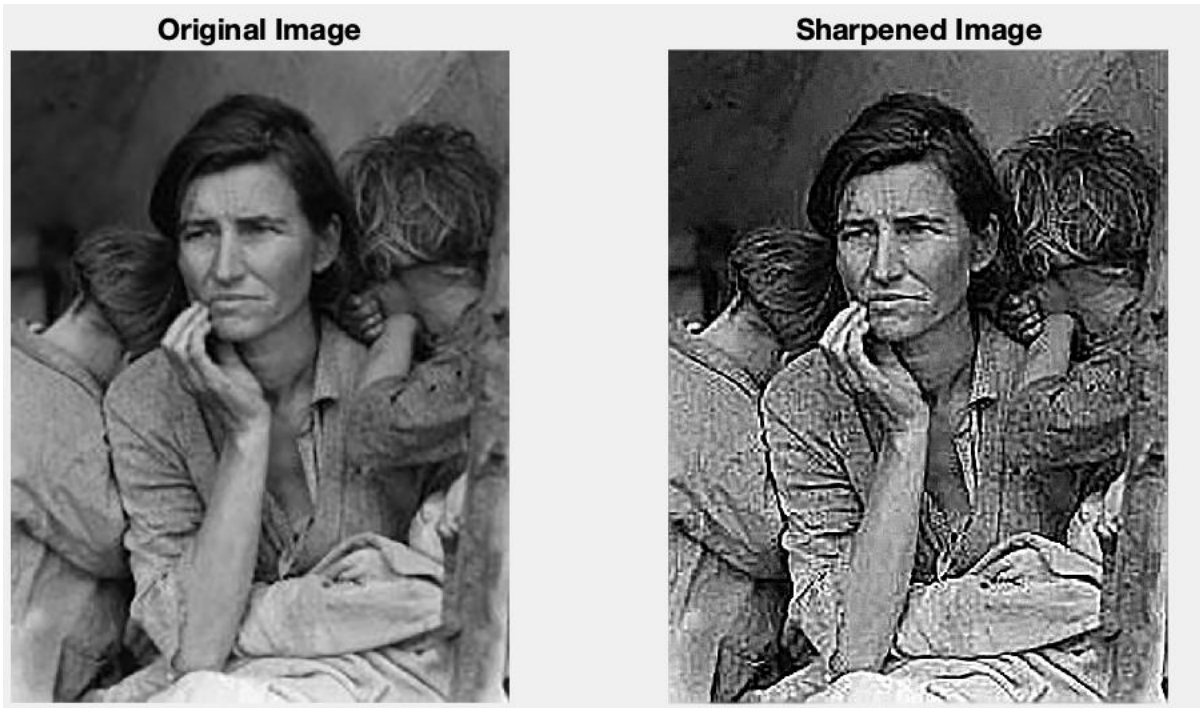
The grayscale “img” and “smoothed” are converted to double for implementation of the operations.

The operation in figure-2 is made by: img+limbda*(img-smoothed) and saved to Sharpened_img. After turning the img, Smoothed and Sharpened_img to uint8 img and Sharpened_img is plotted.

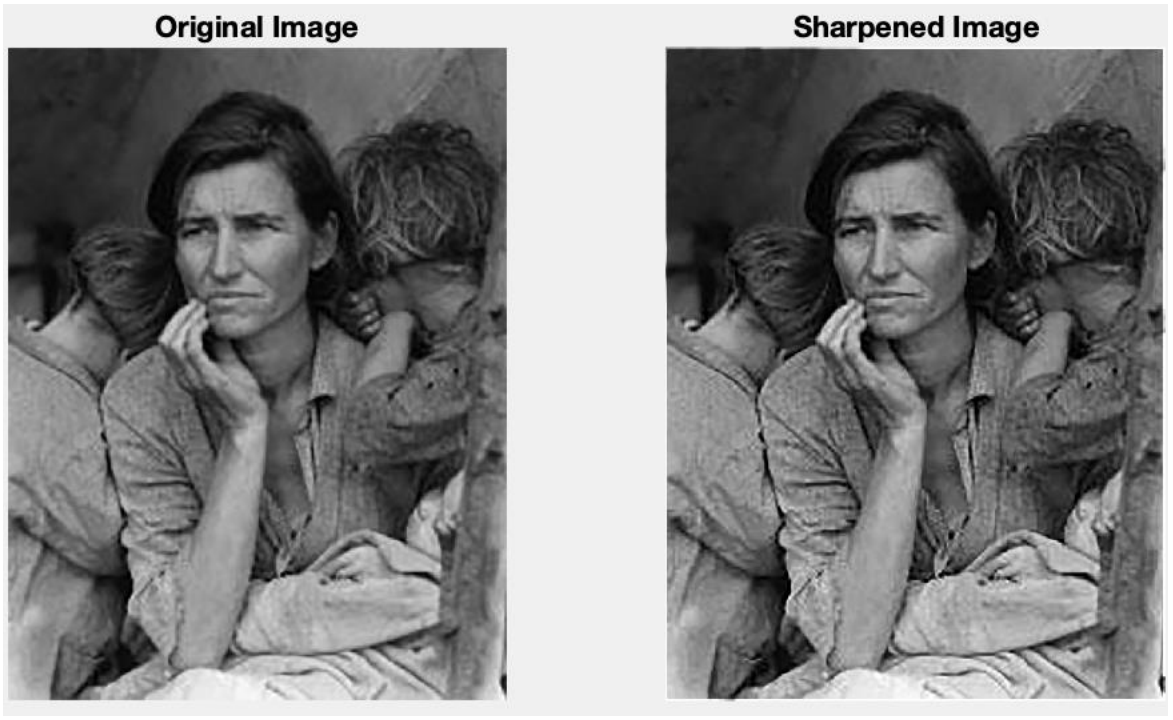
Results for lab2sharpen.m with mother.png(600x784), M=1, limbda=2:



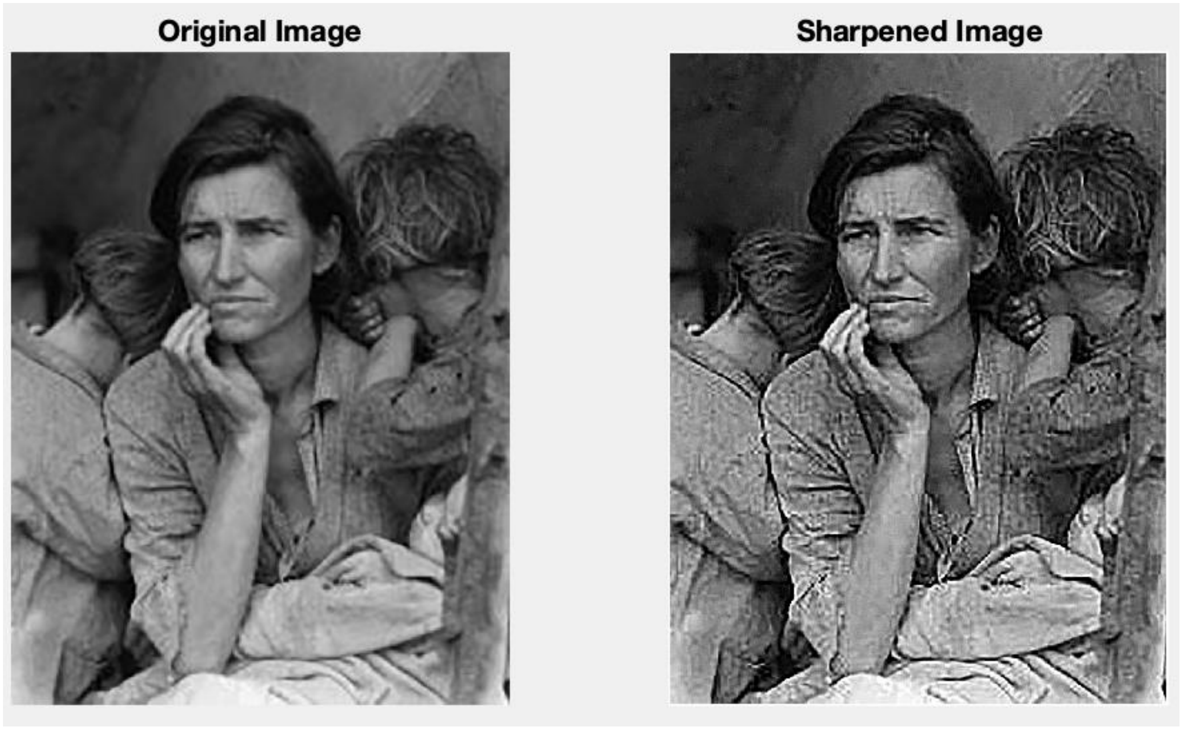
Results for lab2sharpen.m with mother.png(600x784), M=1, limbda=10:



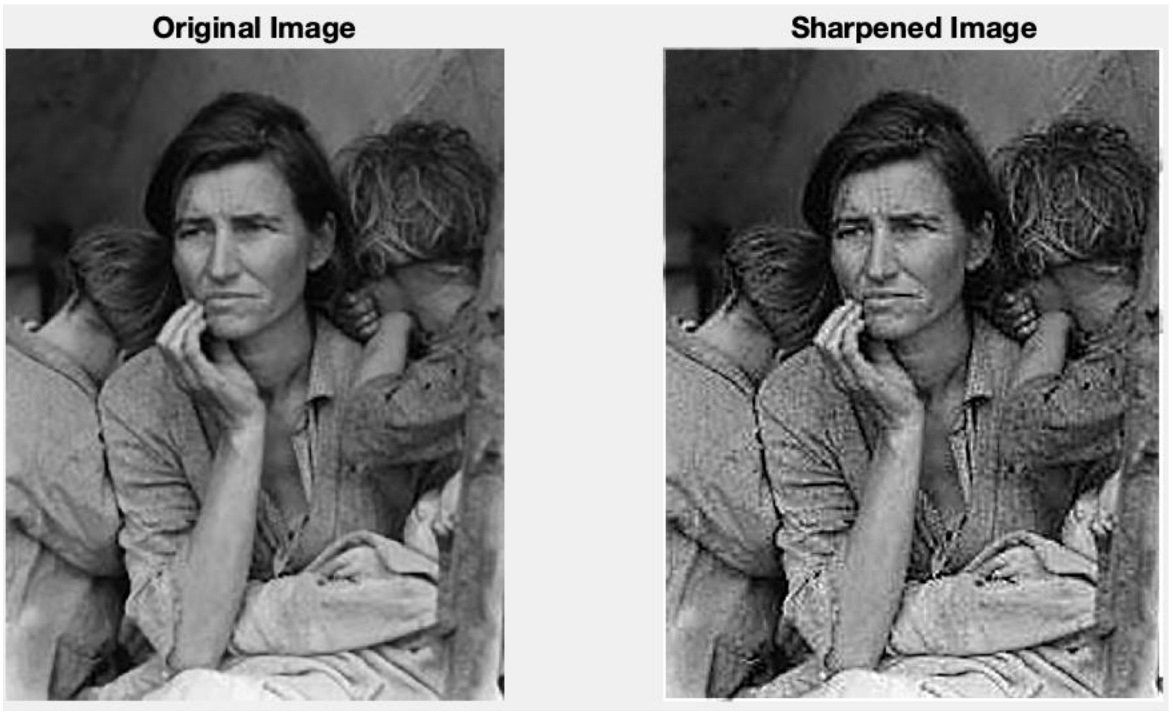
Results for lab2sharpen.m with mother.png(600x784), M=2, limbda=2:



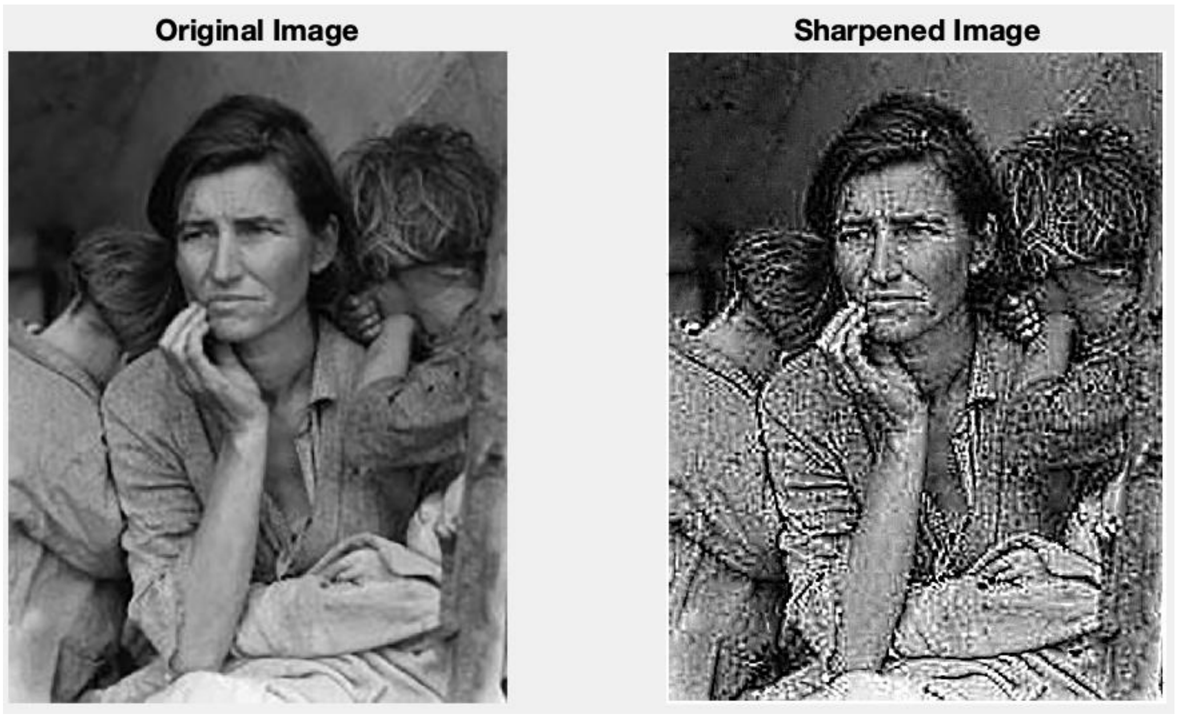
Results for lab2sharpen.m with mother.png(600x784), M=2, limbda=10:



Results for lab2sharpen.m with mother.png(600x784), M=3, limbda=2:



Results for lab2sharpen.m with mother.png(600x784), M=3, limbda=10:

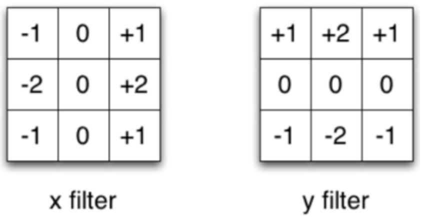


Observations and Conclusion for Sharpening:

Increasing limbda value increase the density of sharpening. The filter works as creating a residual. Different modes(filters) were taken into account when returning the filtered image.

First Derivative

Sobel filtering is a discrete 2D derivative operation which can be applied with the following kernels:



-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

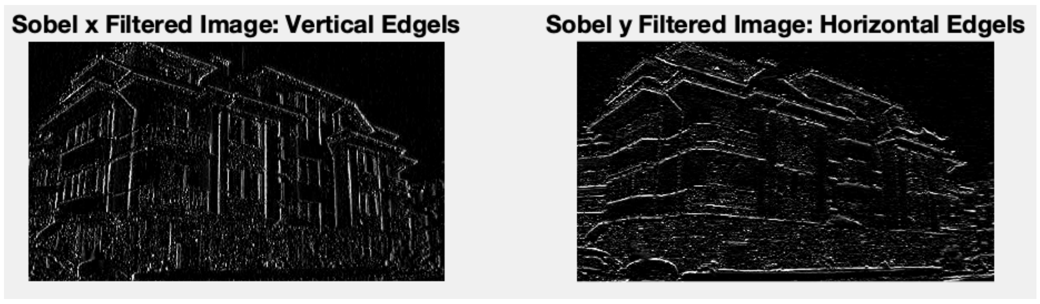
y filter

Figure-3

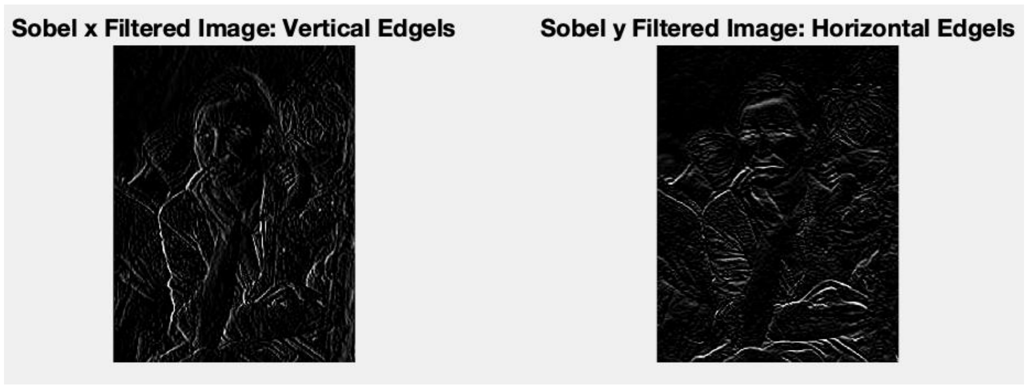
The function takes an image as input and utilizes "Sobel filter" to return two images which are the first derivatives of the image along x and y directions.

After rgb input check is made in the same way with the other filters, the image creates x-filtered and y-filtered versions of the original image by convolving the pixels with the kernels in figure-3. Then the results are plotted.

Results for lab2sobelfilt.m with house.png(1,060x610:


Sobel x Filtered Image: Vertical Edgels
Sobel y Filtered Image: Horizontal Edgels

Results for lab2sobelfilt.m with mother.png(600x784):


Sobel x Filtered Image: Vertical Edgels
Sobel y Filtered Image: Horizontal Edgels

Observations and Conclusion for First Derivative:

The function eliminates the axis it filters(X axis gets greater pixel values when y filter is applied, Y axis gets greater values when x filter is applied). The function makes edges easier to detect.

APPENDIX

```
%% Part 1: Linear Filtering (Gaussian)
clear all; close all; clc;
img = imread('gaussNoise.png');
ResImg = lab2gaussfilt(img);
figure
    subplot(1,2,1)
        imshow(img)
    subplot(1,2,2)
        imshow(ResImg)
%% Part 2: Non-Linear Filtering (Median)
clear all; close all; clc;
img = imread('Tiger.png');
k=10;
ResImg = lab2medfilt(img,k);
%% Part 3: Sharpening
clear all; close all; clc;
img = imread('mother.png');
limbda = 10;
Mode = 3;
ResImg = lab2sharpen(img,limbda,Mode);
%% Part 4: First Derivative
clear all; close all; clc;
img = imread('mother.png');
[x_filtered,y_filtered] = lab2sobelfilt(img);
```

```
function [Filtered_img] = lab2gaussfilt(img)

    [row,col,ch]=size(img);

    if (ch==3)
        img = rgb2gray(img);
    end

    Gaussian_matrix = (1/273)*[    1   4   7   4    1  ;
                                   4   16  26  16   4  ;
                                   7   26  41  26   7  ;
                                   4   16  26  16   4  ;
                                   1   4   7   4    1  ];

    Filtered_img = zeros(size(img));
    img = double(img);
    k = 2;

    for i = k+1:1:row-k-1

       for j = k+1:1:col-k-1

           Window = img(i-k:i+k,j-k:j+k);

           value = sum(sum(Window.*Gaussian_matrix));

           Filtered_img(i,j) = value;

       end

    end

    %method 2
%     Filtered_img = conv2(img,Gaussian_matrix,'full');
%
    img = uint8(img);
    Filtered_img = uint8(Filtered_img);
end
```

```
function [Sharpened_img] = lab2sharpen(img,limbda,M)

    [row,col,ch]=size(img);

    if (ch==3)

        img = rgb2gray(img);

    end


    if (M==1)

        Smoothed = lab1locbox(img,2);

    elseif (M==2)

        Smoothed = lab2gaussfilt(img);

    elseif (M==3)

        Smoothed = lab2medfilt(img,3);

    end


    img = double(img);

    Smoothed = double(Smoothed);


    Sharpened_img = img + limbda*(img - Smoothed);


    img = uint8(img);

    Smoothed = uint8(Smoothed);

    Sharpened_img = uint8(Sharpened_img);


    figure('NumberTitle', 'off', 'Name', 'Sharpening');
        subplot(1,2,1)
            imshow(img)
            title('Original Image');
        subplot(1,2,2)
            imshow(Sharpened_img)
            title('Sharpened Image');

end
```

```
function [x_filtered,y_filtered] = lab2sobelfilt(img)

    [row,col,ch]=size(img);

    if (ch==3)

        img = rgb2gray(img);

    end


    img = double(img);


    x_Filter = [-1 0 1;-2 0 2; -1 0 1];

    y_Filter = [-1 -2 -1; 0 0 0 ; 1 2 1];


    k=1;

    for i = k+1:1:row-k-1

       for j = k+1:1:col-k-1

           window = img(i-k:i+k,j-k:j+k);

           Xvalue = sum(sum(window.*x_Filter));

           Yvalue = sum(sum(window.*y_Filter));

           x_filtered(i,j) = Xvalue;

           y_filtered(i,j) = Yvalue;
       end
    end
    img = uint8(img);

    x_filtered = uint8(x_filtered);

    y_filtered = uint8(y_filtered);

    figure;
    subplot(2,2,[1 2])
        imshow(img)
        title('Original Image');
    subplot(2,2,3)
        imshow(x_filtered)
        title('Sobel x Filtered Image: Vertical Edgels');
    subplot(2,2,4)
        imshow(y_filtered)
        title('Sobel y Filtered Image: Horizontal Edgels');

end
function [Filtered_img] = lab2medfilt(img,k)

    [row,col,ch]=size(img);

    if (ch==3)

        img = rgb2gray(img);

    end

    img = double(img);

    Filtered_img = zeros(size(img));

    GaussianFiltered = lab2gaussfilt(img);

    for i = k+1:1:row-k-1

        for j = k+1:1:col-k-1

            Window = img(i-k:i+k,j-k:j+k);
            value = myMedian(Window);
            Filtered_img(i,j) = value;

        end

    end

    img = uint8(img);

    Filtered_img = uint8(Filtered_img);


    figure
    subplot(1,3,1)
        imshow(img)
    subplot(1,3,2)
        imshow(GaussianFiltered)
    subplot(1,3,3)
        imshow(Filtered_img)

end
```

```
function median = myMedian(img)

[row,col,ch] = size(img);


Card = row*col;

    if (ch==3)

        img = rgb2gray(img);

    end


    img = double(img);

    Data = reshape(img, [1 Card]);

    Data = sort(Data);


    if (mod(Card,2) == 0)

        ind = Card/2;

        median = (Data(ind) + Data(ind+1))/2;

    else

        ind = (Card+1)/2;

        median = Data(ind);

    end
end
```