Lab #2 Post-Lab Report

pciftcioglu@sabanciuniv.edu

November 2020

1 Introduction

In this lab, we tried to do several operations such as linear (Gaussian) filtering, non-linear (Median) filtering, sharpening and Sobel filtering. All of the operations have been done to gray-scale images and has been checked whether it is gray-scale before each process. To avoid any confusion, during the rest of the paper, the will-be-processed images that we will use our operations to will be referred as "Reference Image", and images we create after processes will be referred as "Resulting Image". Initially, the processes for each operation that we have done in lab will be explained with certain methods used; and accordingly, further trials with different parameters (e.g. window size) or different images will be discussed with my own results.

2 In Lab Implementations

2.1 Linear (Gaussian) Filtering

Gaussian filtering which is a local operator is used to reduce the noise in the image which can be described as "pepper and salt noise" and smoothens the image. To do so, we used a 5*5 Gaussian matrix for 2D kernel layer as follows:

$$\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} / \ 273$$

We started with initializing an empty matrix with zeros for the resulting image first and chose our window size as k=2 to apply the Gaussian kernel. For every window in the reference image, we applied matrix multiplication with the Gaussian matrix and sum all the values in the resulting matrix. Then we inserted the resulting value to the pixel value of the center of the window for the newly created matrix which is the resulting image. As it can be seen in the Figure 1, resulting image is way more smooth than the actual one and its noise is significantly less.

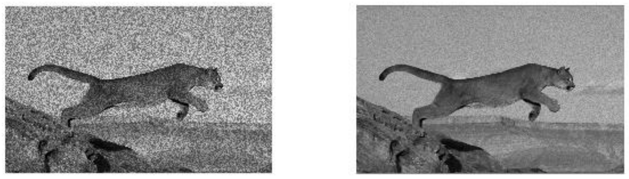

Figure 1: Linear (Gaussian) Filtered, k = 2

2.2 Non-Linear (Median) Filtering

For non-linear filtering which is also a local operator, we again initialized an empty matrix with zeros as the same size as the referenced image again. However, this time, instead of using another matrix for operation, we took the median of the window in the size 2*2 since we took k = 2 this time. For calculating the median we used a function that we implemented before which calculates the median of the pixel values in the image. As a result, median filtering resulted in a better image in terms of reduced noise compared to the Gaussian filtering as can be seen in Figure 2.

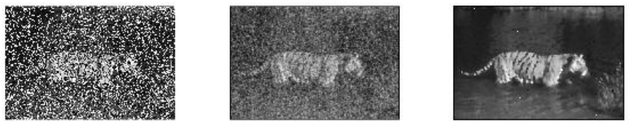

Figure 2: Linear Filtering and Non-linear filtering k = 2

2.3 Sharpening

The goal of sharpening as it has been described in the lecture is to produce an enhanced image of a reference image by increasing the contrast along the edges, without adding too much noise within homogeneous regions. For this purpose following formula has been implemented:

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

where S is the smoothed version of the given image I and $\lambda > 0$ is a scaling factor which controls the influence of the correction signal.
We used the parameter value M to determine which method will be used for smoothing the image since we need to use that version for sharpening as it can be seen in the formula. M = 1 corresponds to box filtering while M = 2 corresponds to Gaussian filtering and M = 3 to median filtering.
Since sharpening itself is a point operation, we took the referenced image and applied the formula with the lambda

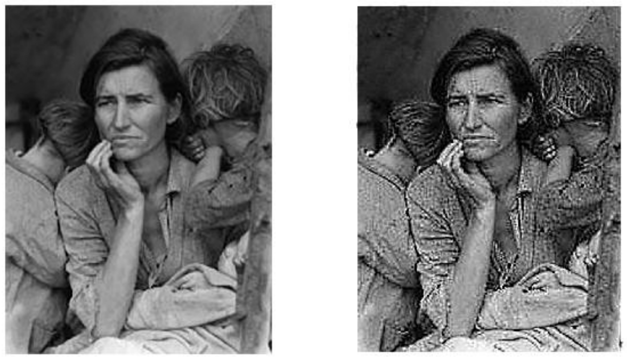

Figure 3: Sharpening M = 3 and $\lambda = 5$, k = 2

value $\lambda = 5$. And as it can be seen in the Figure 3 that the resulting image is a sharpened version of the original image. It has been used median filtering to smoothing the image and we will be comparing other methods in the following section.

2.4 Sobel Filtering

This filter results in the first derivative of the given axis and we will be implementing it for both x and y axes. To do so, we use two matrices sized 3*3 which is as follows:

$$\text{x Filter} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \text{ y Filter} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel filtering is a local operator so we also take parameter k = 1 for window size. And again as we did in the median filtering we operate matrix multiplication with x filter and window of size k and sum all the values in the resulting matrix and insert the resulting values into a new matrix which also corresponds to the vertical edgels of the referenced image. We also did the same operation for every window with y filter matrix and this corresponds to the horizontal edgels of the referenced image.

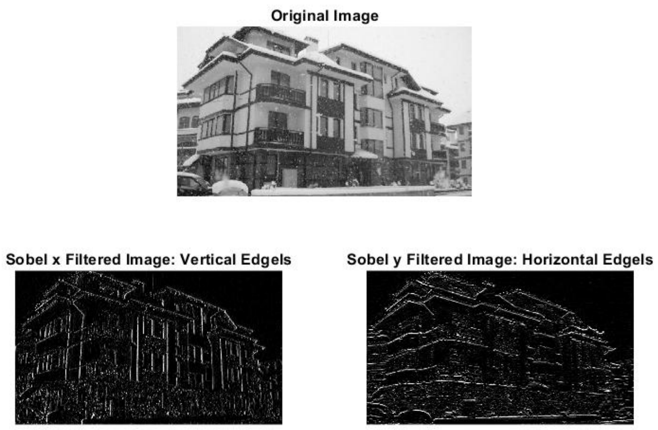
Original Image
Sobel x Filtered Image: Vertical Edgels
Sobel y Filtered Image: Horizontal Edgels

Figure 4: Local Max and Local Min Filtering

As it can be seen in the Figure 4 that, when applied x filter, resulted image shows the horizontal edges more defined and as a skeleton whereas y filters clearly shows the vertical edges.

3 Post-Lab Experiments

3.1 Linear Filtering

I used two more images to try this experiment with Gaussian filters to see whether it is going to be successful or not. And as it can be seen from the Figures 5 and 6, images are way easier to see and smoother. However, they are still not clear enough such that even though the "salt and pepper" noise is less, it is still looking like a pixelled image. But this might be because of the lower resolution of the images which is 338*338 for the image at Figure 5 and 256*384 for the image in Figure 6 which is smaller resolution compared to the image in the Figure 1, so we will keep experimenting.

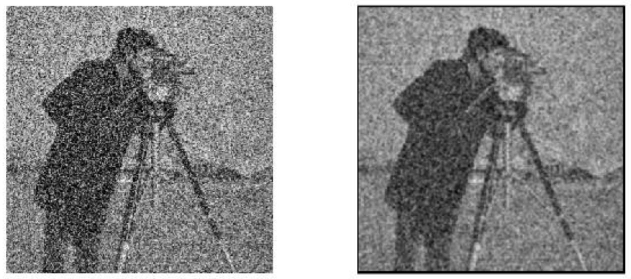

Figure 5: Linear Filtering k = 2

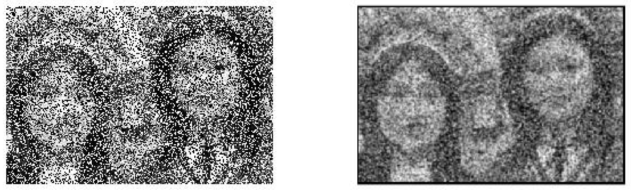

Figure 6: Linear Filtering k = 2

3.2 Median Filtering

First, I tried a smaller and a larger k value to see what is going to happen to the image and as it can be observed from the Figure 1 and Figure 8 larger k resulted in a smoother image and it even caused the image ending up way too blurry for large k. And since we also observe that non-linear filtering does a better job than linear filtering in terms of reducing the salt and pepper noise, I will try to smoothen the previous images with this method too.

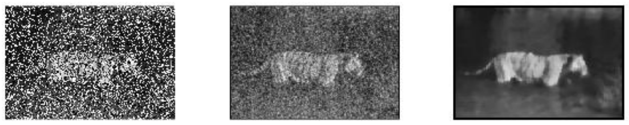

Figure 7: Median Filtering with k = 5

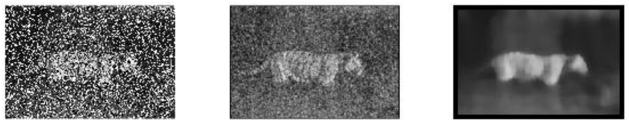

Figure 8: Median Filtering with k = 10

I tried couple of different k values for both of the image from the previous experiment and observed which k values are optimal for them so that the image does not look like it has that much noise anymore but also not that blurry to actually confuse what is in the image.

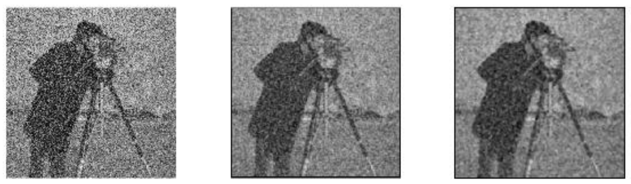

Figure 9: Conditional Scaling

For the image in the Figure 9 I used k = 2 and as it can be seen from the comparison with the Gaussian filtering is that the resulting image is blurrier than the second image. However, even though it is less noisy it became harder to see so it might be possible to conclude that the resolution of the image affected the performance of the filtering. Thus I tried another image for this again as follows:

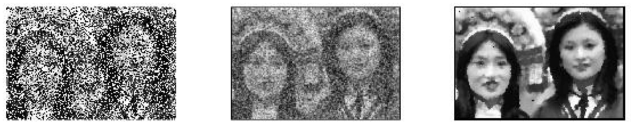

Figure 10: Median Filtering with k = 3

I picked the window size value as k = 3 in this case since when it was k = 2 the noise was still clear to see and there were still some inconsistent pixels present with their background noises and when k ¿ 3 it started becoming way too blurry and the image started seeming a bit scary because of the shapes of the human faces got into as can be seen from Figure 11. On the other hand, it is possible to state that the median filtering did a much better job for reducing the noise in the image in this example such that the girls' faces are way more easy to see and the image is way clearer in terms of brightness and contrast spread.

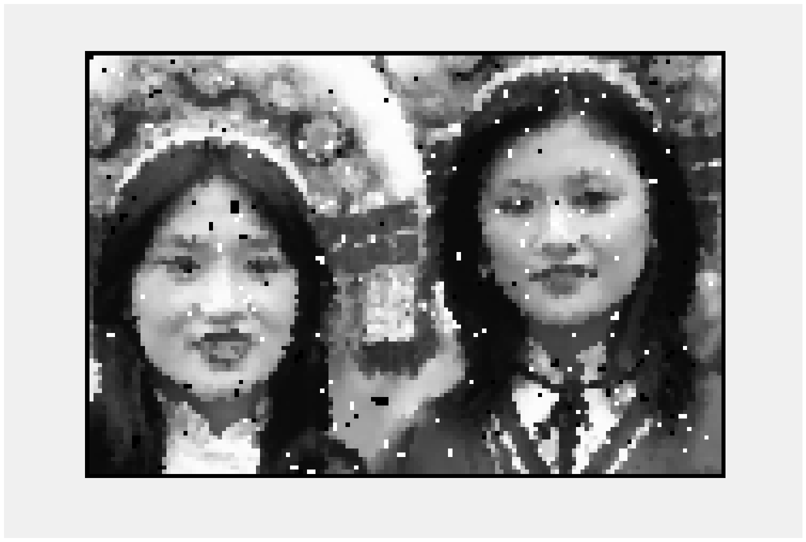

Figure 11: Median Filtering with k = 2

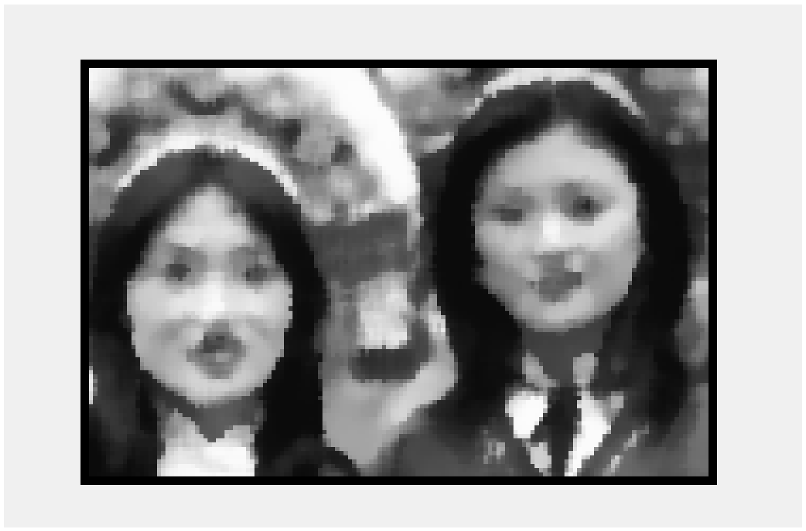

Figure 12: Median Filtering with k = 5

3.3 Sharpening

Since it was getting harder to see the image because it was getting blurrier after reducing the noise of the image in the Figured 9, this time I tried sharpening on the smoothed image to see if I can make it better. And I used median filtering to smoothing the image since it was the most successful one compared to linear filtering and local box filtering.

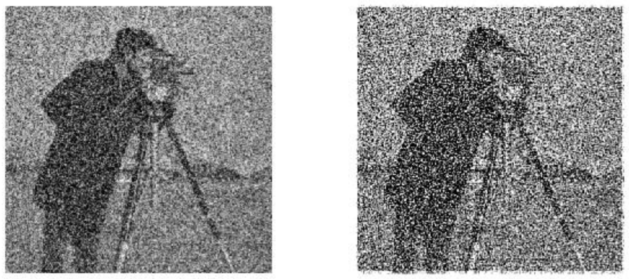

Figure 13: Sharpening with M = 3 and k = 2

However, sharpening the image did not resulted in a better (in terms of clearness) image but rather brought the noise back to the image. And bigger lambda value actually made the image worse than it actually was so I took lambda = 1 but still this image has failed to be better in terms of quality with this experiment. So I wanted to try one another experiment to see whether larger λ makes the image sharper or not.

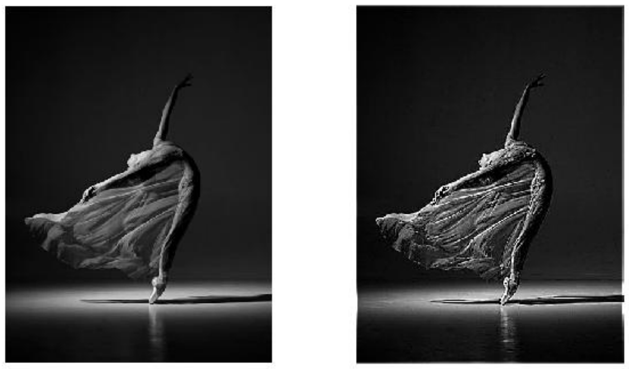

Figure 14: Sharpening with M = 3, k = 2 and $\lambda = 3$

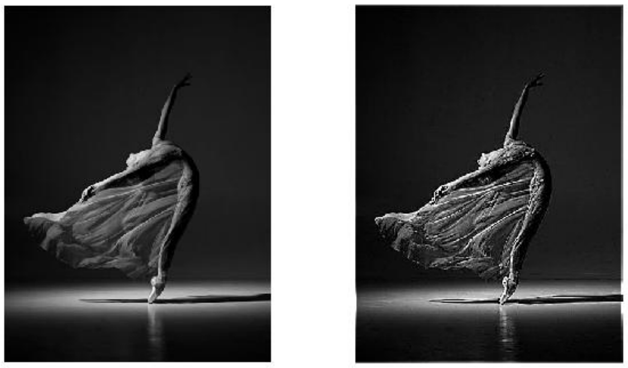

Figure 15: Sharpening with M = 3, k = 2 and $\lambda = 10$

I again chose the mode 3 with k = 2 since it was the best smoothing method according to the previous results. And as can be seen from the Figures 14 and 15 larger λ resulted in a more defined and sharper image both it might start to be way more defined that can bother our vision after some larger k.

3.4 Sobel Filtering

I wanted to try Sobel filtering on an image that includes human inside to observe whether is it going to be easy to understand what the image represent or will it make a difference if its horizontal or vertical edgels.

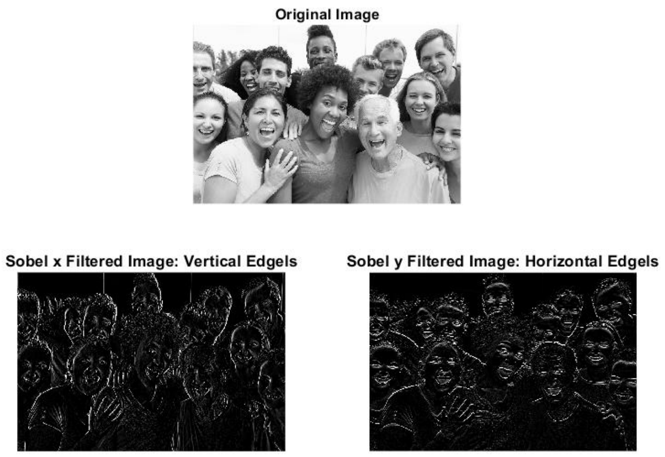
Original Image
Sobel x Filtered Image: Vertical Edgels
Sobel y Filtered Image: Horizontal Edgels

Figure 16: Sobel Filtering with k = 1

As it can be seen from the Figure 16 that in both of the directions it is still possible to understand what the image represents but the way that the objects have been observed from our eyes is different. For instance, if we zoom in to both of the images as in represented at Figure 17, we can see that we are both observing the teeth and mouth but the way the shadows are is different. And it possible to pay attention to the vertical details in the x filtered and vice-versa.

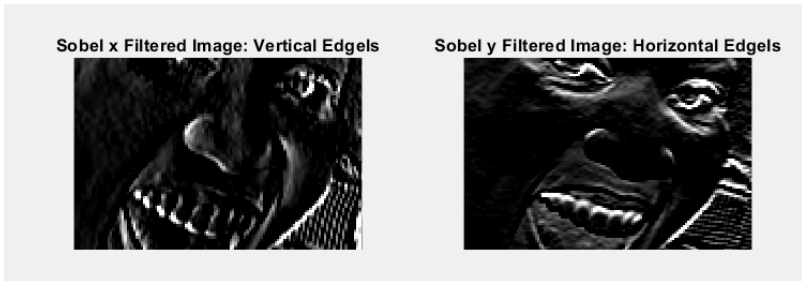
Sobel x Filtered Image: Vertical Edgels
Sobel y Filtered Image: Horizontal Edgels

Figure 17: Sobel Filtering with k = 1 Zoomed In

4 Appendix

4.1 Main

```
%% Part 1: Linear Filtering (Gaussian)

clear all; close all; clc;

img = imread('man.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('man.png');
k = 2;
% larger k is smoother image
ResImg = lab2medfilt(img, k);

%% Part3: Sharpening

clear all; close all; clc;

img = imread('ballet.jpg');
lambda = 10;
Mode = 3;
ResImg = lab2sharpen(img, lambda, Mode);

%% Part4: Sobel Filtering

clear all; close all; clc;

img = imread('people.jpeg');
[x_filtered, y_filtered] = lab2sobelfilt(img);
```

4.2 Linear Filtering

```
function [ResImg] = 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 ];

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

    % method 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);
            value = sum(sum(Window.*Gaussian_Matrix));
            ResImg(i,j) = value;
        end

    end

%     % Method 2
%     ResImg = conv2(img, Gaussian_Matrix, "full");


    img = uint8(img);
    ResImg = uint8(ResImg);

end
```

4.3 Non-Linear Filtering

```
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
```

4.4 Sharpening

```
function [Simg] = lab2sharpen(img, lambda, M)
% Sharpened Image as output

    [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,2);
       % k is 3
   end

   img = double(img);
   Smoothed = double(Smoothed);

   Simg = img + lambda*(img - Smoothed); %this is the formula

   img = uint8 (img);
   Smoothed = uint8(Smoothed);
   Simg = uint8(Simg);

   figure
subplot(1,2,1)
    imshow(img)
subplot(1,2,2)
    imshow(Simg)

end
```

4.5 Sobel Filtering

```
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
```