Lab #1 Post-Lab Report

pciftcioglu@sabanciuniv.edu

October 2020

1 Introduction

As it is mentioned in the lab assignment document, recorded image quality can be degraded from various reasons and it might result in ”noisy” images. In this lab, we tried to do several operations such as linear scaling, conditional scaling, box filtering, local minimum and maximum scaling to make them more suitable for the further processes on them. All of the operations have been done to gray-scale images. 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 images will be discussed with my own results.

2 In Lab Implementations

2.1 Linear Scaling

This function takes a gray-scale image and applies linear transformation which is a point operation as we discussed in the lecture. It uses a gradation function which maps every value used in the image to a whole scale. The scale starts from 0 and goes until a certain maximum value which used to be 255 (G_{max}) in our case. More precisely, the function takes all the positive histogram values for each pixel (as in the form of column matrix) in the image in interval $[u_{min}, u_{max}]$ and scales them to 0 from G_{max}. All the pixels with the histogram value u_{min} corresponds to 0 while all the pixels with value umax maps onto G_{max} in the new filtered image. As it can be seen in the Figure 1 the resulting image is significantly clearer to see than the original darker version. And it can also be seen from the histograms that, the pixel values are distributed more evenly across the whole scale in contrast to the unprocessed image and this helps us to see a better quality of the image.

$$g(u) = b(u + a)$$

where,

$$a = -u_{min}, b = \frac{G_{max}}{(u_max - u_min)}$$

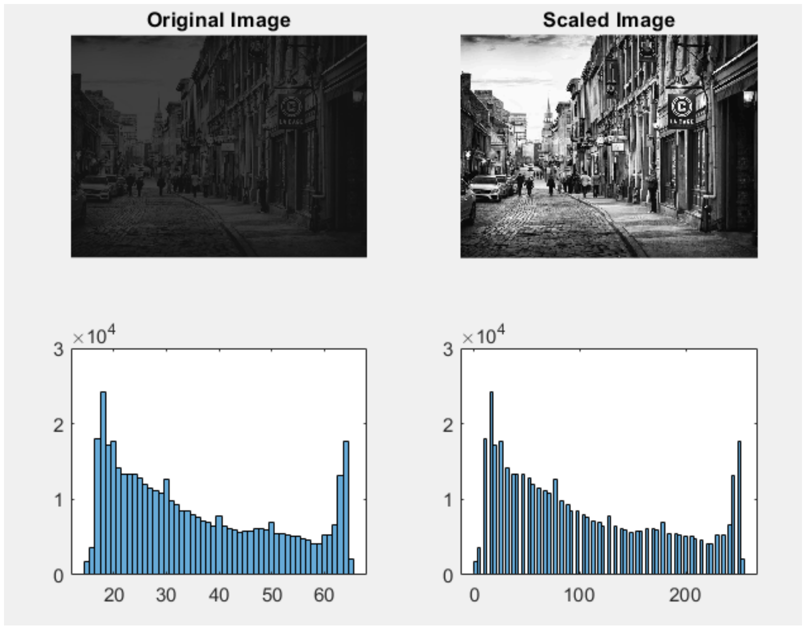

Figure 1: Linear Scaling

2.2 Conditional Scaling

This function is another processing method in point operations which also uses a gradation function to map the pixels to new values. However, this time the image that will be processed uses mean and variance values from another image. With more detail, the function takes two images where one of the is the reference image with the parameters that will be used and the other one is processed. For both images, it has been calculated their mean and standard deviation within their pixel values and used in the formula as below:

$$g(u) = b(u + a)$$

where,

$$a = \mu_I \frac{\sigma_J}{\sigma_I} - \mu_J \text{ and } b = \frac{\mu_I}{\mu_J}$$

And accordingly, new image has been created with the values that have been mapped by the function. It can be seen from the Figure 2 that the new image looks similar with the reference image in terms of the brightness and contrast. And it provides a better vision compared to the old version since it has been created with values from an image that we were satisfied with its brightness quality and clearness.
Additionally, the standard deviation and mean values has been displayed and returned the values:

- For the reference image mean and standard deviation values: 116.5072, 49.1187
- For the will be processed image mean and standard deviation values: 35.7225, 15.0345
- For the processed image after the scaling mean and standard deviation values: 116.5072, 49.1187 (same as the referenced image)

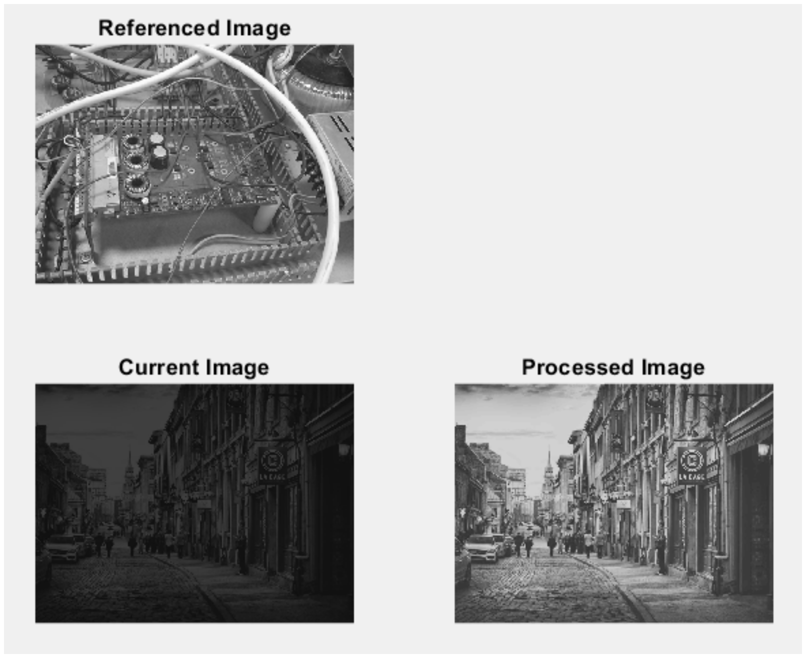

Figure 2: Conditional Scaling

2.3 Box Filtering

This process is an instance of local operators which is different than the first two operations. Furthermore, this function uses the help of a concept called ”window” with a size k to k, which covers (2k+1)*(2k+1) pixels with the center pixel p. And the related operations are calculated based on the local information of the pixels within that window frame. In other words, the window with a certain size slides upon the whole image and inserts the new pixel value for the center p with calculating the average of values within the window.

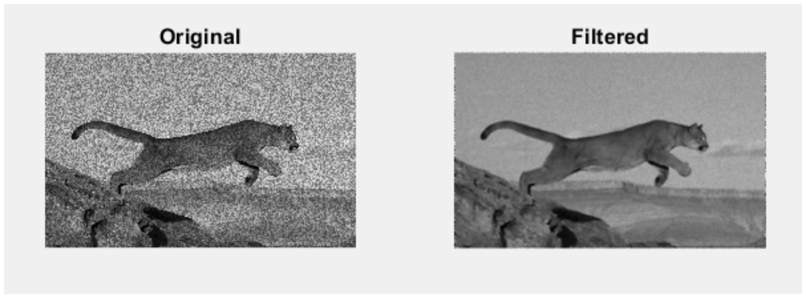

Figure 3: Local Mean (Box) Filtering

In our case, the local mean (i.e. box filter) operator used to reduce the noise in the original image as it can be seen from the Figure 3. It can be clearly seen that the new image has a better view in terms of view clarity since it has a blurring and smoothing effect.

2.4 Local Max and Local Min Filters

This is another example from local operators and it is used for dilation and erosion of pixels. It again uses a sliding window with the same definition and size (2k+1)*(2k+1) where k=3 in our case. The difference from the previous operation is that the maximum and the minimum pixel values are calculated for each window and the center pixel p is changed with the new value (maximum for local maximum filter and minimum for local min filter).

As it can be seen in the Figure 4 local max filtered image is brighter than the original image in the places where its darker in the original image whereas local min filter ended up in a darker image.

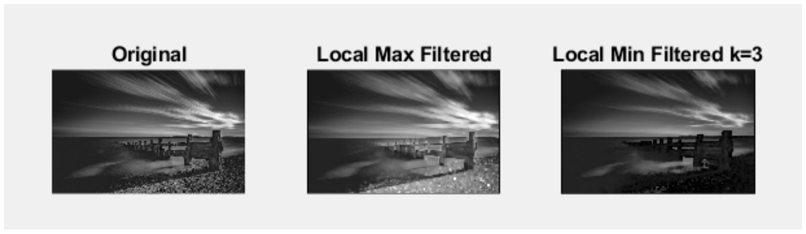

Figure 4: Local Max and Local Min Filtering

3 Post-Lab Experiments

3.1 Linear Scaling

After the trials in class I tried another picture to see its effect. Here Figure 5 as it can be seen from the original picture it is a monochromatic image. I assumed that it will be more saturated when linearly scaled but it didn't work. The algorithm couldn't scale the image to a brighter version. Maybe its because the darkness was not enough.

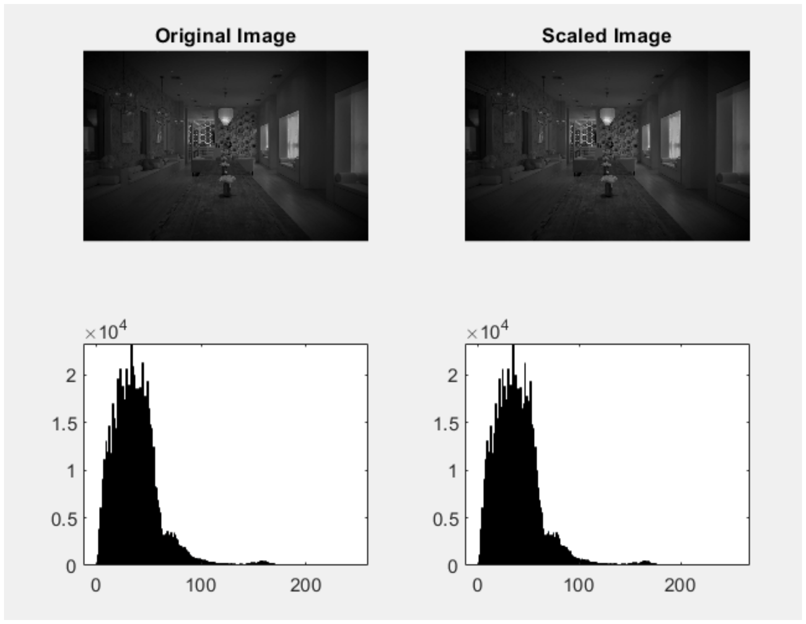

Figure 5: Linear Scaling

3.2 Conditional Scaling

Then I thought maybe conditional scaling would work on the previous image if I pick a brighter image to get it brighter. So I found a very bright image and used that in the function.

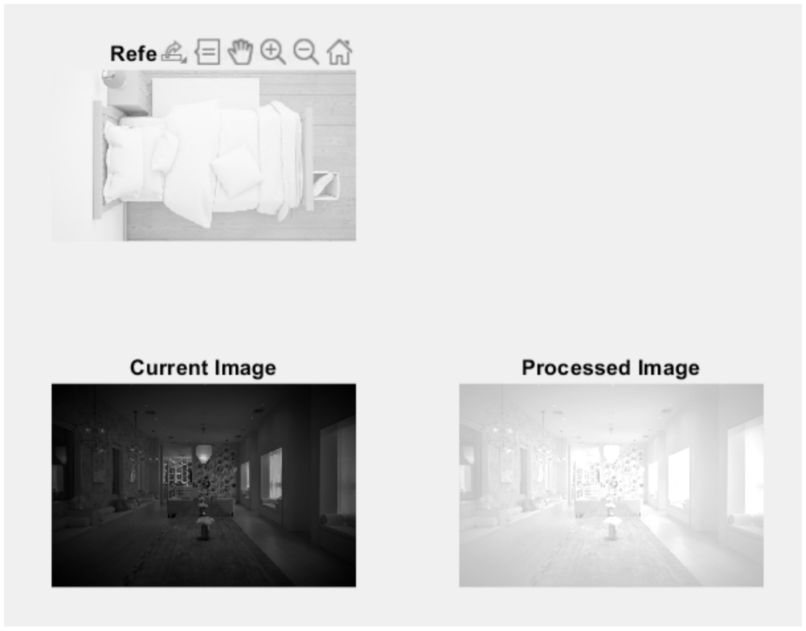

Figure 6: Conditional Scaling

Although, it worked to make the image brighter than how it normally is, this time it became so bright that it passed where it is clear enough. So I tried another bright image but a less white one.

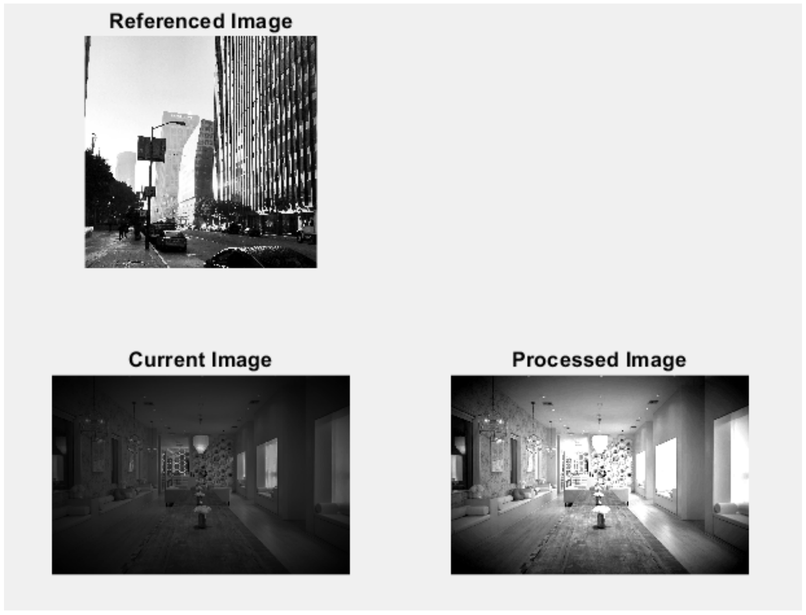

Figure 7: Conditional Scaling

Now, as the selected image to be taken as a reference for processing was brighter and more balanced as in terms of colors and brightness it seems like it helped to fix the color of our image.

3.3 Box Filtering

Now, after we reduced the noise on the image, I wondered what is going to happen after bigger k value. And as it can be seen in the Figure 8 that bigger k made the image blurry. Thus, I tried with a smaller value to see if k = 5 is the only option or optimum.

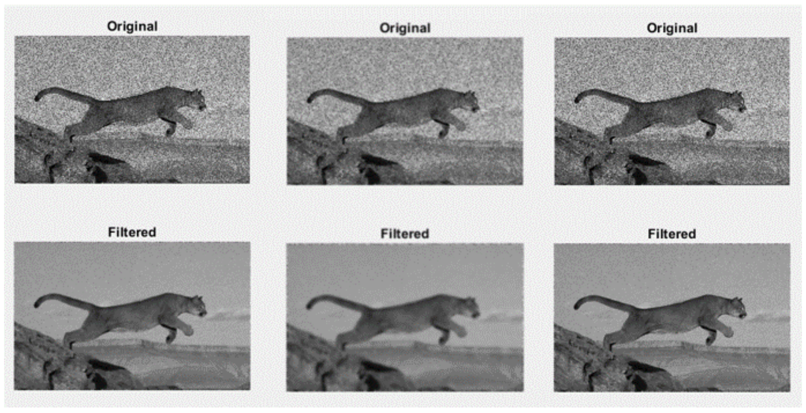

Figure 8: Box Filtering with k = 5, 10 and 3

The Figure 8 shows the k = 5, k = 10 and k = 3 respectively. As I checked if the k = 3 changed the result in a better way, I realized that it was still a bit more noisy. So based on the preference we want to achieve, k value can be adjusted too see which value suits better since it might be different k in every example.

3.4 Local Max and Min Filters

As we did local maximum and local minimum filters with k = 3, I want to try a bigger k value such that k = 10. As it can be also seen in Figure 9, it resulted in image getting low quality and the image got broken in the local min operation since the pixel look increased. However, the local max operation made the image darker (compared to both the original image and k = 3) but not any better than k = 3.

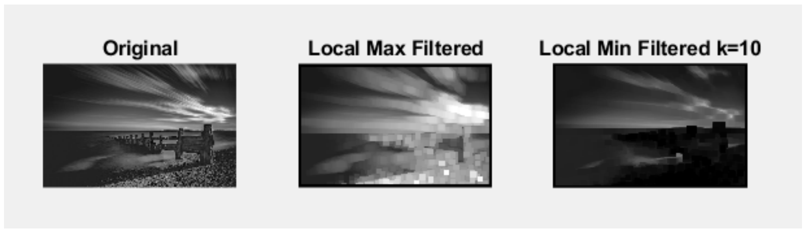

Figure 9: Local Min and Max Filters with k = 10

4 Appendix

4.1 Main

```
%% Linear Scaling
clc; clear; close all

Im = imread("sceneroom2.png");

[Inew] = lab1linscale(Im);

%% Conditional Scaling
clc; clear; close all

Im = imread("sceneroom2.png");
Iref = imread("city3.jpg");

[Jnew] = lab1condscale(Im,Iref);

%% Box Filter
clc; clear; close all

Im = imread("jump.png");
k = 5;
[Inew] = lab1locbox(Im,k);


%% Local Min/Max Filter
clc; clear; close all

Im = imread("currentImage.png");
k = 3;
[Imax, Imin] = lab1locmaxmin(Im,k);
```

4.2 Linear Scaling

```
function [Inew] = lab1linscale(Im)

[h, w, c] = size(Im);

if c == 3
    Im = rgb2gray(Im);
end

I = double(Im);

a = - min(I(:));

Gmax = 256;
b = Gmax / ( max(I(:)) - min(I(:)) );

Inew = b * (I + a);

Inew = uint8(Inew);

subplot(2,2,1), imshow(Im);
title("Original Image")
subplot(2,2,2), imshow(Inew);
title("Scaled Image")
subplot(2,2,3), histogram(Im);
subplot(2,2,4), histogram(Inew);

end
```

4.3 Conditional Scaling

```
function [Jnew] = lab1condscale(Im,Iref)

[h, w, c] = size(Im);
if c == 3
    Im = rgb2gray(Im);
end

[h1, w1, c1] = size(Iref);
if c1 == 3
    Iref = rgb2gray(Iref);
end

I = double(Iref);
J = double(Im);

b = std(I(:)) / std(J(:));
a = mean(I(:))/b - mean(J(:));

Jnew = b * (J + a);
disp([ mean(I(:)), mean(J(:)), mean(Jnew(:)) ]);
disp([ std(I(:)), std(J(:)), std(Jnew(:)) ]);

Jnew = uint8(Jnew);

subplot(2,2,1), imshow(Iref);
title("Referenced Image")
subplot(2,2,3), imshow(Im);
title("Current Image")
subplot(2,2,4), imshow(Jnew);
title("Processed Image")

end
```

4.4 Box Filtering

```
function [Inew] = lab1locbox(Im,k)
I = double(Im);

[h, w, c] = size(I);

% Inew = zeros(h,w);
Inew = I;

for i= k+1:h-k
   for j= k+1:w-k
       wp = I(i-k:i+k, j-k:j+k);
       Inew(i,j) = mean(wp(:));
   end
end

Inew = uint8(Inew);
subplot(2,1,1), imshow(Im);
title("Original")
subplot(2,1,2), imshow(Inew);
title("Filtered")
end
```

4.5 Local Min/Max Filters

```
function [Imax,Imin] = lab1locmaxmin(Im,k)
I = double(Im);

```

```
[h, w, c] = size(I);

Imax = zeros(h,w);
Imin = zeros(h,w);

for i= k+1:h-k
   for j= k+1:w-k
       wp = I(i-k:i+k, j-k:j+k);
       Imax(i,j) = max(wp(:));
       Imin(i,j) = min(wp(:));

   end
end

Imax = uint8(Imax);
Imin = uint8(Imin);


subplot(1,3,1), imshow(Im);
title("Original")
subplot(1,3,2), imshow(Imax);
title("Local Max Filtered")
subplot(1,3,3), imshow(Imin);
title("Local Min Filtered " + ['k=' num2str(k)])

end
```