EE417 - Assignment 1 - Post-Lab Report

Baris Sevilmis
Lab Date: 09/10/2018
Due Date: 15/10/2018

Overview

First laboratory assignment focuses on four different types of image transformation techniques, namely: Linear scaling, conditional scaling, box filtering and lastly, local min/max filtering.

Part 1: Linear Scaling

Linear scaling is a point operation, in which intensity range of an image I, specifically $[Umin, Umax]$, is mapped to intensity range of $[0, Gmax]$.

$$Gmax = 255$$

$$a = -Umin$$

$$b = \frac{Gmax}{Umin + Umax}$$

As depicted in Figure 1, linear scaling provides a larger intensity range for images. In other words, intensity range of original image is normalized such that intensity values are expanded to their new range. This result can be directly seen from the histograms in Figure 1, as well as scaled image proves same outcome as the intensity values are clearly in a wider range.

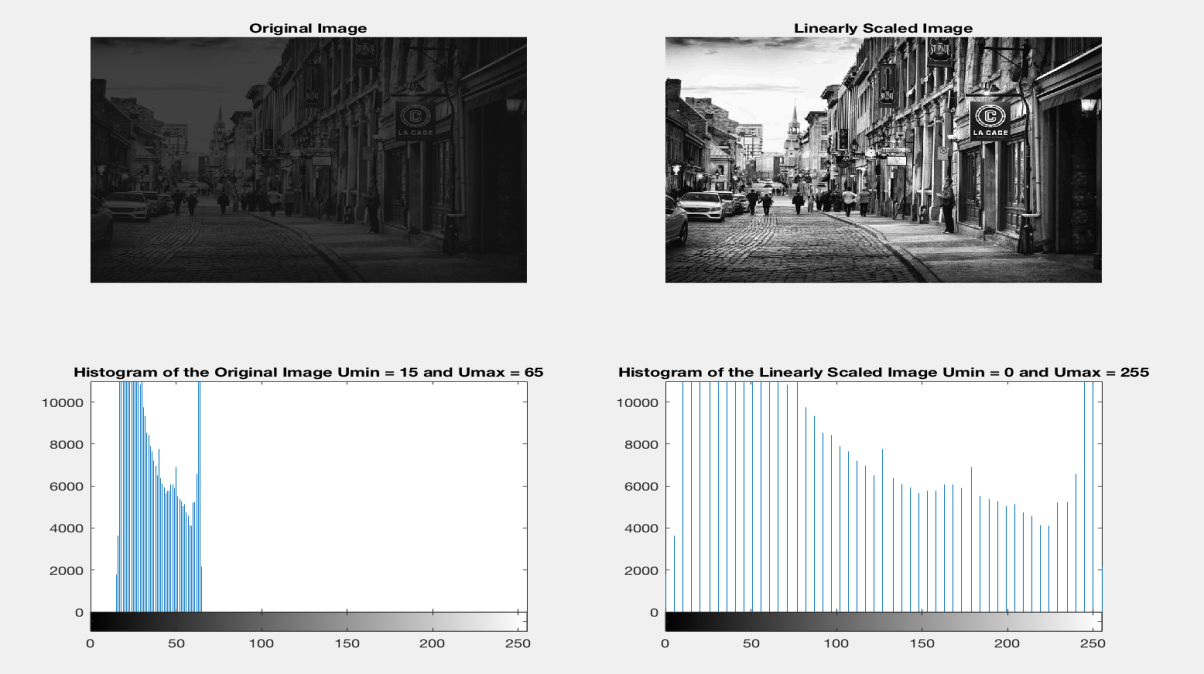

Figure 1: Linear Scaling of city image

In Figure 2, MatLab Code for Linear Scaling is provided, as it is a MatLab function. Implementation is based on the formulas provided previosly.

```
function J = lab1linscale(img)
    [row, col, ch] = size(img);

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

    umin = min(img(:));
    umax = max(img(:));
    Gmax = 255;

    a = -1 * umin;
    b = Gmax/(umax-umin);

    J = b .* (img + a);
    J = uint8(J);

    subplot(2,2,1);imshow(uint8(img));
    title('Original Image');
    subplot(2,2,2);imshow(J);
    title('Linearly Scaled Image');
    subplot(2,2,3);imhist(uint8(img));
    title('Histogram of the Original Image Umin = 15 and Umax = 65');
    subplot(2,2,4);imhist(J);
    title('Histogram of the Linearly Scaled Image Umin = 0 and Umax = 255');
end
```

Figure 2: Linear Scaling function in MatLab

Part 2: Conditional Scaling

Conditional Scaling is likewise a point operator, in which image J is being transformed to image Jnew, in which Jnew has the same mean and standard deviation as a reference image I. Conditional scaling is linear operator such as linear scaling, therefore same method will be used with different formulas.

$$a = \mu_i \frac{\sigma_j}{\sigma_i} - \mu_j$$

$$b = \frac{\sigma_i}{\sigma_j}$$

It is clearly shown in Figure 3 that the city image was scaled conditionally with respect to board image. As a consequence, conditionally scaled city image has the same mean and standard deviation as reference board image. Respective mean and standard deviation values are below the images in Figure 3.

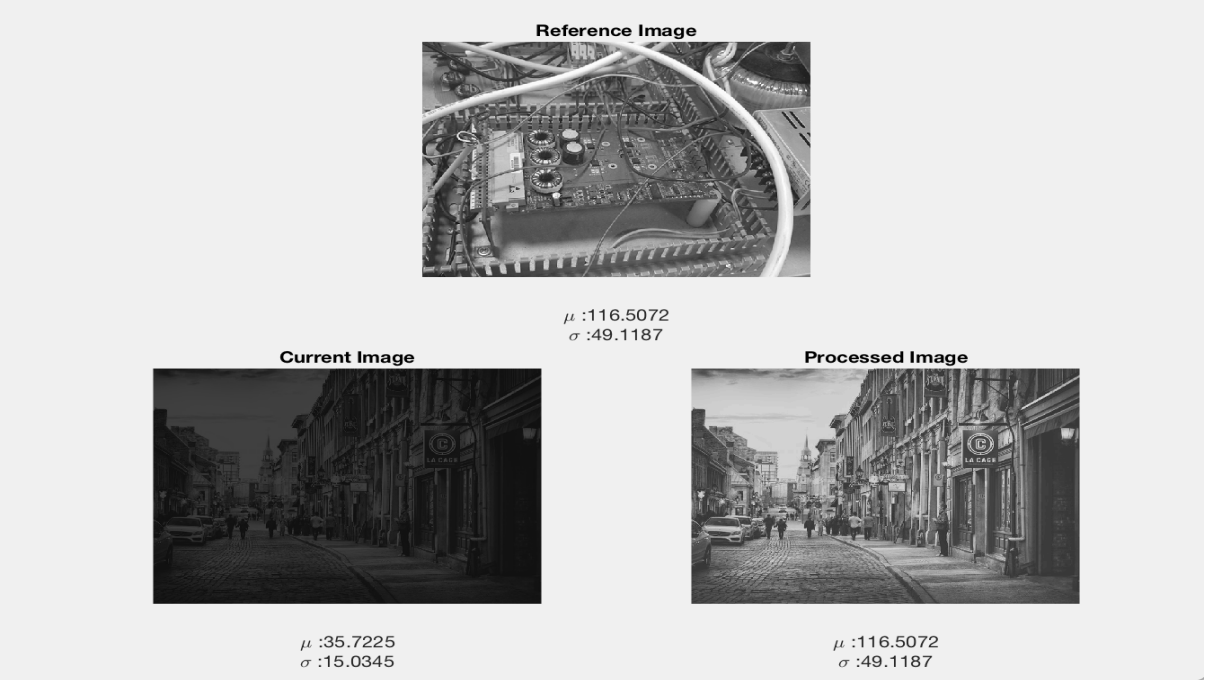

Figure 3: Conditional Scaling of city image by referencing board image

Figure 4 depicts the respective MatLab code for the conditional scaling function. For the implementation, given mathematical formulas were used such that image was transformed into its new form without any problem.

```
function J = lab1condscale(img_i, img_j)
    [row1, col1, ch1] = size(img_j);
    [row2, col2, ch2] = size(img_i);
    if(ch1 == 3)
       img_j = rgb2gray(img_j);
    end
    if(ch2 == 3)
       img_i = rgb2gray(img_i);
    end
    img_j = double(img_j);
    img_i = double(img_i);

    mu_j = mean(img_j(:));
    mu_i = mean(img_i(:));
    std_j = std(img_j(:),0 , 'all');
    std_i = std(img_i(:),0 , 'all');

    b = std_j / std_i;
    a = mu_i * (std_j / std_i) - mu_j;

    J = b .* (img_i + a);
    J = uint8(J);
```

Figure 4: Conditional Scaling function in MatLab (1)

On the other hand, Figure 5 shows the necessary coding in order to visualize the MatLab code in Figure 4.

```
23
24 -        subplot(2,2,[1,2]);imshow(uint8(img_j));
25 -        title('Reference Image');xlabel({['\mu :',num2str(mu_j)],['\sigma :',num2str(std_j)]})
26 -        subplot(2,2,3);imshow(uint8(img_i));
27 -        title('Current Image');xlabel({['\mu :',num2str(mu_i)],['\sigma :',num2str(std_i)]});
28 -        subplot(2,2,4);imshow(J);
29 -        title('Processed Image');xlabel({['\mu :',num2str(mu_j)],['\sigma :',num2str(std_j)]})
30 -    end
```

Figure 5: Conditional Scaling function in MatLab(2)

Part 3: Local Mean/Box Filter

Local Mean Filtering method is a method in which average of the elements within the window are taken. Result of this operation, is the new intensity value for the transformed image. Every pixel value is replaced by the window average value. Technically, average operation can be expressed as the following:

$$\mu_{Wp} = \frac{1}{(2k+1)^2} \sum_{i=-k}^{+k} \sum_{j=-k}^{+k} I(x+i, y+j)$$

Three different experiments were produced in order to understand results better. First experiment can be seen in Figure 6, as window size is fixed to 3. Since window size is small, averaging operation remains relatively local comparing to large window sizes. Therefore, result is much more clear, because only very close pixels are averaged.

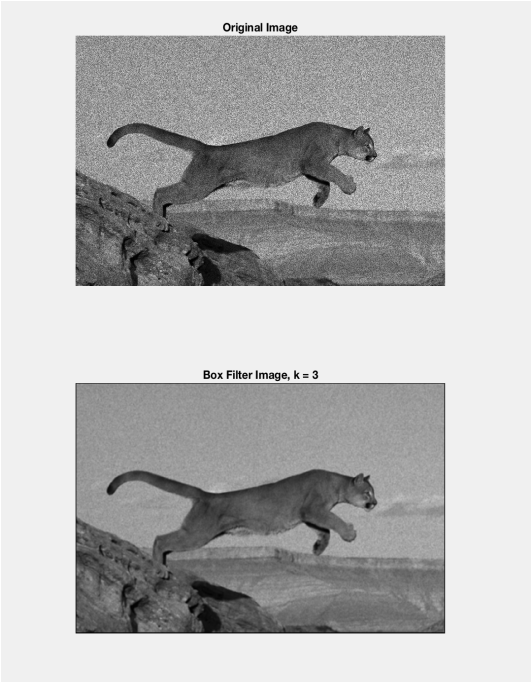

Figure 6: Local Mean Filtering with k = 3

As in Figure 7, second experiment for local mean operator is with window size =5. At a certain distance, differences of window sizes are almost invisible. However, if looked with attention, k = 5 results in a more blurry vision because of the larger window size.

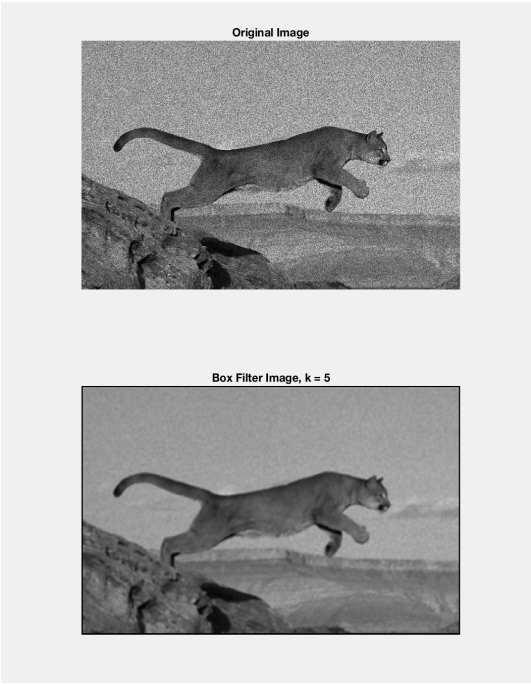

Figure 7: Local Mean Filtering with k = 5

Finally, Figure 8 depicts same experiment with window size = 11, and it is clear that new image is a lot blurrier than previous images. Since the window size is greater now, averaging operation happens over a larger matrix of pixels.

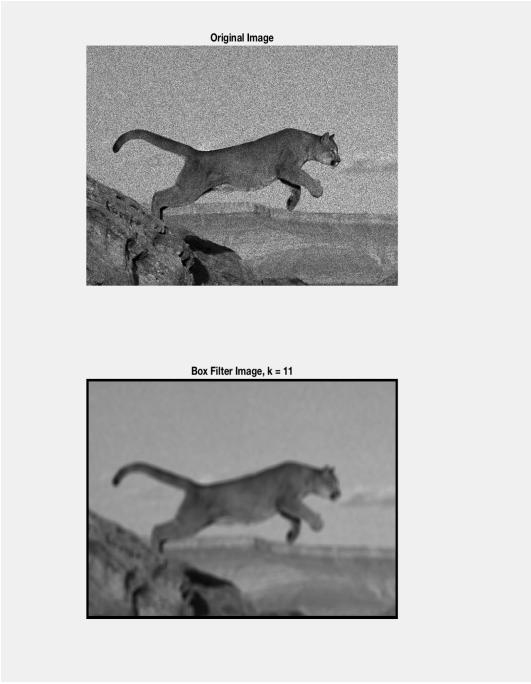

Figure 8: Local Mean Filtering with k = 11

Figure 9 contains the respective function in MatLab for Local Mean/Box Filtering additionally with the respective plotting.

```
function J = lab1locbox(img, k)
    [row, col, ch] = size(img);
    if(ch == 3)
       img = rgb2gray(img);
    end
    J = zeros(size(img));
    img = double(img);

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

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

            subimg = img(i-k: i+k, j-k:j+k) ;
            mu = mean(subimg(:));
            J(i, j) = mu;

        end
    end
    J = uint8(J);

    subplot(2,1,1);imshow(uint8(img));
    title('Original Image');
    subplot(2,1,2);imshow(J);
    title(['Box Filter Image, k = ', num2str(k)]);
end
```

Figure 9: Local Mean Filtering code in MatLab

Part 4: Local Max and Min Filter

Local Max and Local Min Filter operations are considered as local operators with the functionality of maximizing/minimizing the intensity value of a pixel by taking maximum/minimum value within the window. For technical precision, local max filtering indicates dilation and local min filtering refers to erosion of an image. To summarize, maximum/minimum values within a window are assigned to pixel in focus. Mathematical explanation can be explained as the following:

$$J(p)_{max} = max(I(x+i, y+j) : -k < i < +k \wedge -k < j < +k)$$
$$J(p)_{min} = min(I(x+i, y+j) : -k < i < +k \wedge -k < j < +k)$$

Figure 10: Local Max/Min Filtering with k = 3

Figure 10 depicts local maximum/minimum operations on the given image using a window of size 3. It is clear that minimum operation results in a darker image, as the max operation image has brighter regions that may seem as blurry. However, those areas are maximized in terms of intensity values because of neighbor bright pixels.In Figure 11, same local max/min operations are used with window size = 5. Therefore, local min filter results in a darker image as the local max image loses its smoothness because of its increased brightness.

Figure 11: Local Max/Min Filtering with k = 5

Lastly, figure 12 demonstrates the results of local max/min with window size = 11. Effects of local max/min filters can be clearly seen, as the local min image is much darker. On the other hand, bright regions in local max image are much more distributed. In short, increasing window size increases the effect of operations on image as the window involves more pixels.

Figure 12: Local Max/Min Filtering with k = 11

MatLab code for Local Max/Min including visualization of these filters are demonstrated in Figure 13. This code is almost same with the Local Mean code, though operations are different. However window structure is always the same considering windowing direction is always the same, from left to right. Therefore same code is used for both Local Mean and Local Min/Max, as only operation is changed.

```
function [Jmax, Jmin] = lab1locmaxmin(img, k)
    [row, col, ch] = size(img);
    if(ch == 3)
       img = rgb2gray(img);
    end
    Jmax = zeros(size(img));
    Jmin = zeros(size(img));
    img = double(img);

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

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

            subImg = img(i-k: i+k, j-k:j+k);
            Jmax(i,j) = max(subImg(:));
            Jmin(i,j) = min(subImg(:));

        end
    end
    Jmax = uint8(Jmax);
    Jmin = uint8(Jmin);
    subplot(1,3,1);imshow(uint8(img));
    title('Original Image');
    subplot(1,3,2);imshow(Jmax);
    title(['Local Max Filter Image, k = ', num2str(k)]);
    subplot(1,3,3);imshow(Jmin);
    title(['Local Min Filter Image, k = ', num2str(k)]);
end
```

Figure 13: Local Max/Min Filter Code in MatLab