ENS 417 Computer Vision

Berkant Deniz Aktaş

19515

Post-Laboratory Report

October 14, 2018

Sabancı University

Faculty of Engineering and Natural Sciences

Computer Science and Engineering

QUESTION 1 – LINEAR SCALING

In this question aim was using a gradient function g(u) = b(u+a) which will convert u_{min} to 0 and u_{max} to 255, which is G_{max}, intensity values. Variable a is defined as: a = -u_{min} and b is defined as: b = G_{max} / (u_{max} –u_{min}).

```
function img2 = 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 = -umin;
b = Gmax/(umax-umin);

img2 = b.*(img+a);

img2 = uint8(img2);
img = uint8(img);

figure;
subplot(2,2,1);
imshow(img);
subplot(2,2,2);
imshow(img2);
subplot(2,2,3);
imhist(img)
subplot(2,2,4);
imhist(img2)
end
```

Function lab1linscale takes img and outputs img2 as a scaled image.In order to do computations, Image is converted to grayscale if it is not on ch 3. After finding minimum and maximum values, and inserting corresponding values to a and b variables, img2 can be computed using gradient function *b.*(img+a).*

In order to plot the images and compare results, images should be turned to uint8.

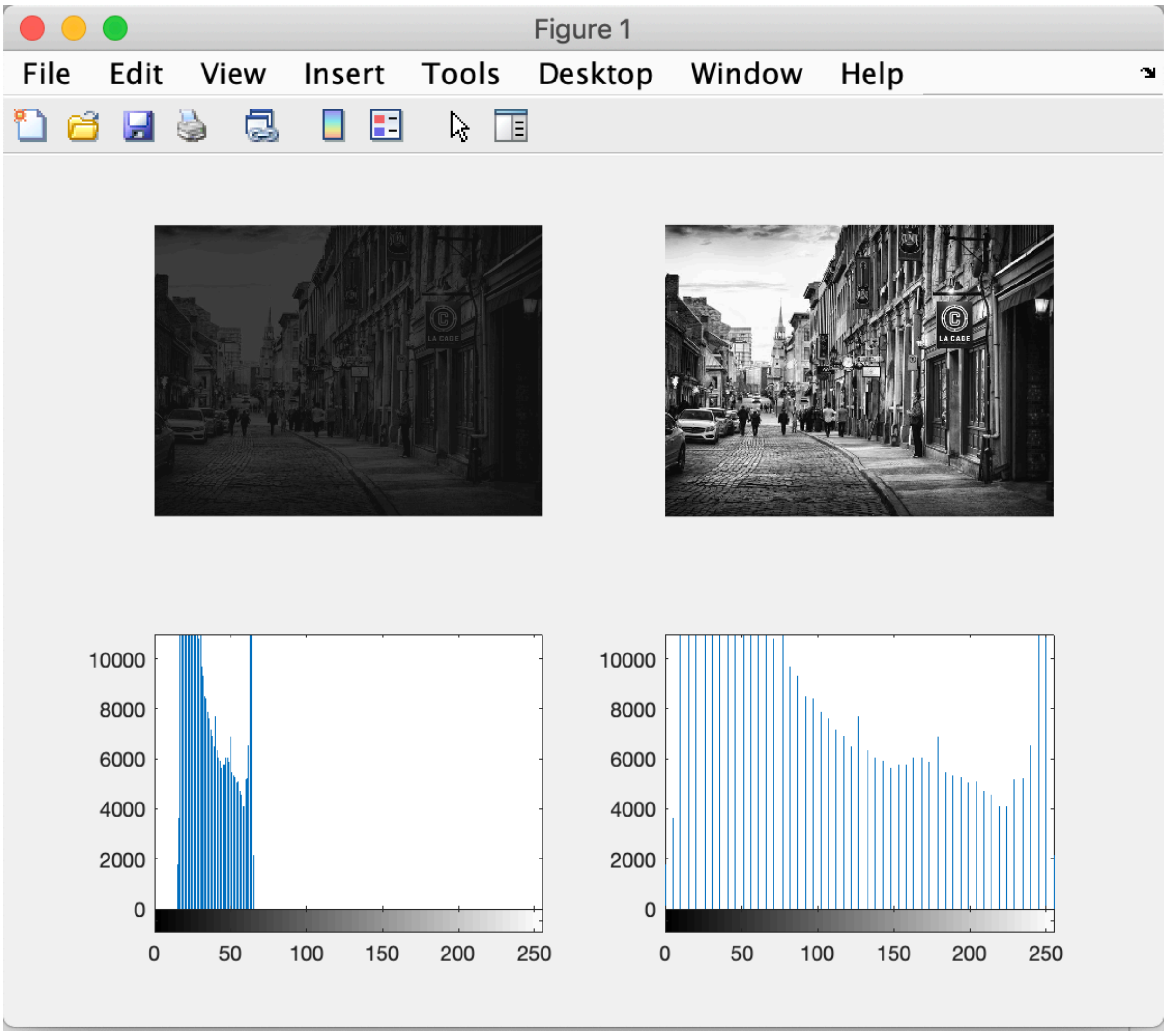

Figure 1. Linear Scaled Version of City.png; img on left, img2 on right

After running the script, images and histograms are shown on the screen. Img has its intensity values between 50 and some small number greater than 0. Which leds a darker image of city. After using linear scaling method, one can see more bright values on the screen. Thanks to linear scaling, img2 has its intensity values distributed between 0 and 255 which makes image more clear and visible.

QUESTION 2 – CONDITIONAL SCALING

Questions asks for implementing a conditional scaling. Idea is using a similar gradient function in the previous question as g(u) = b(u+a) but inserting different values to a and b variables. Aim is using an imgI, which has its own mean and std values, in order to convert imgJ to imgJnew where imgJnew has same mean and std values with imgI.

```
function imgJnew = lab1condscale(imgJ,imgI)

[row, col, ch] = size(imgJ);
[row2, col2, ch2] = size(imgI);
if(ch==3)
    img = rgb2gray(imgJ);
end
if(ch2==3)
    img2 = rgb2gray(imgI);
end

imgJ = double(imgJ);
imgI = double(imgI);

meanJ = mean(imgJ(:));
meanI = mean(imgI(:));
stdJ = std(imgJ(:));
stdI = std(imgI(:));

a = (meanI*(stdJ/stdI))-meanJ;
b = (stdI/stdJ);

imgJnew = b.*(imgJ +a);

imgJ = uint8(imgJ);
imgJnew = uint8(imgJnew);
imgI = uint8(imgI);

figure;
subplot(2,3,1);
imshow(imgJ);
subplot(2,3,2);
imshow(imgJnew);
subplot(2,3,3);
imshow(imgI)
subplot(2,3,4);
imhist(imgJ)
subplot(2,3,5);
imhist(imgJnew)
subplot(2,3,6);
imhist(imgI)
end
```

Using a method similar with previous question, imgJ and imgI checked if they are greyscaled or not. After that they are converted to double in order to do computations. After finding mean and std of imgJ and imgI, one can compute a as: a = meanI* (stdJ/stdI) – meanJ and b as: b = stdI/stdJ. After computing the g(u) = b(u+a) and creating imgJnew as imgJnew = b.*(imgj+a) one can plot the result with converting images to uint8.

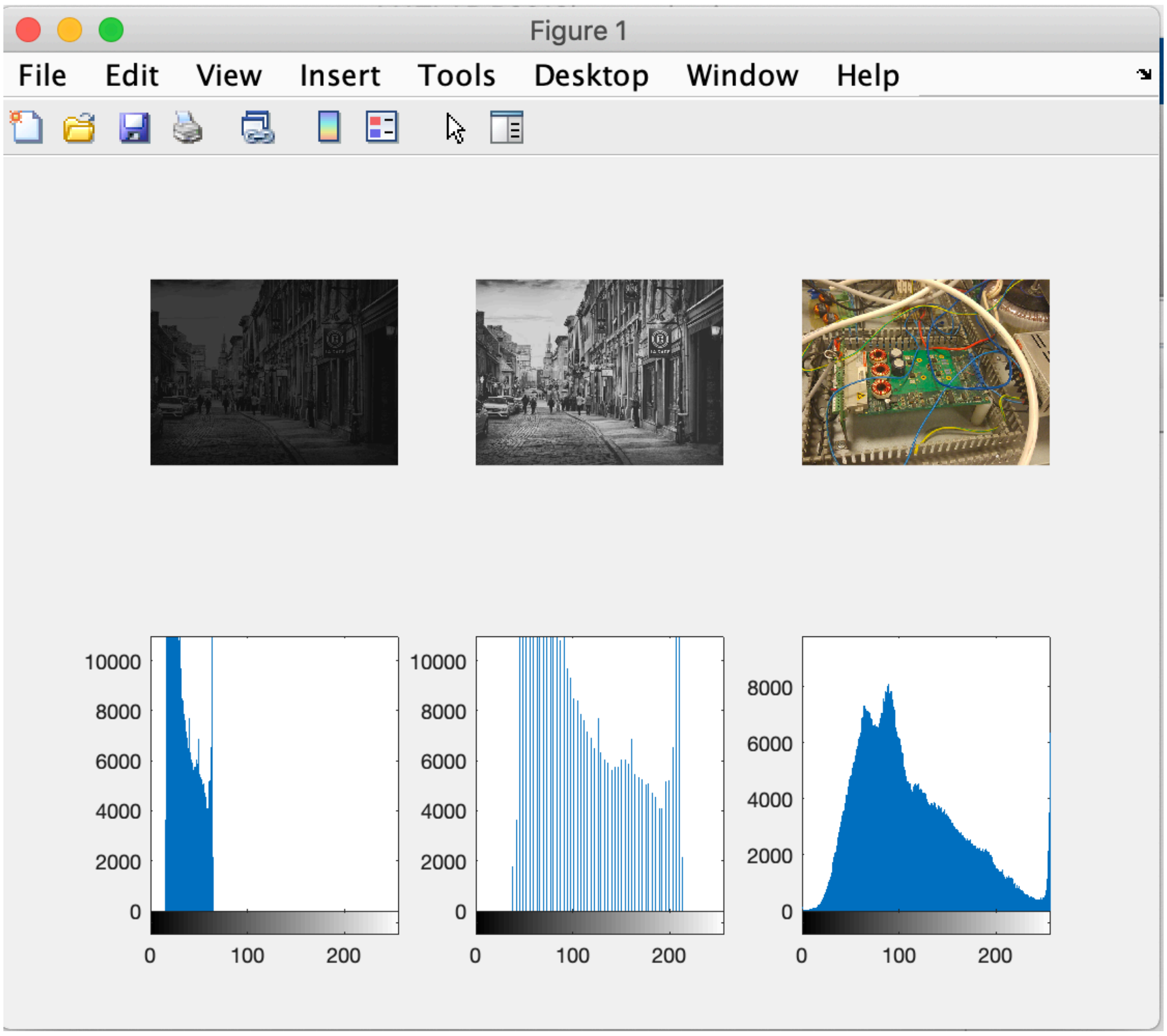

Figure 2. imgJ on left, imgJnew on middle, imgI on right.

Comparing the histograms of imgJnew and imgI will result similar means and std values.

Thus, imgJ conditionally scaled to ImgJnew where mean and std values of imgI and imgJ new are equal.

QUESTION 3 – BOX FILTER

In this question aim is applying KxK window to image and changing the value of the pixel in the center of the window. In order to traverse all pixels, used double for loop and get the mean intensity value of the each window. This operation will increase the blur of the image. Choosing bigger window size K will results a more blurry image.

```
function img2 = lab1locbox(img,K)

[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end

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);
       meanValue=mean(subImg(:));
       img2(i,j)= meanValue;
    end
end

img = uint8(img);
img2 = uint8(img2);
figure;
subplot(2,2,1);
imshow(img);
subplot(2,2,2);
imhist(img);
subplot(2,2,3);
imshow(img2)
subplot(2,2,4);
imhist(img2)

end
```

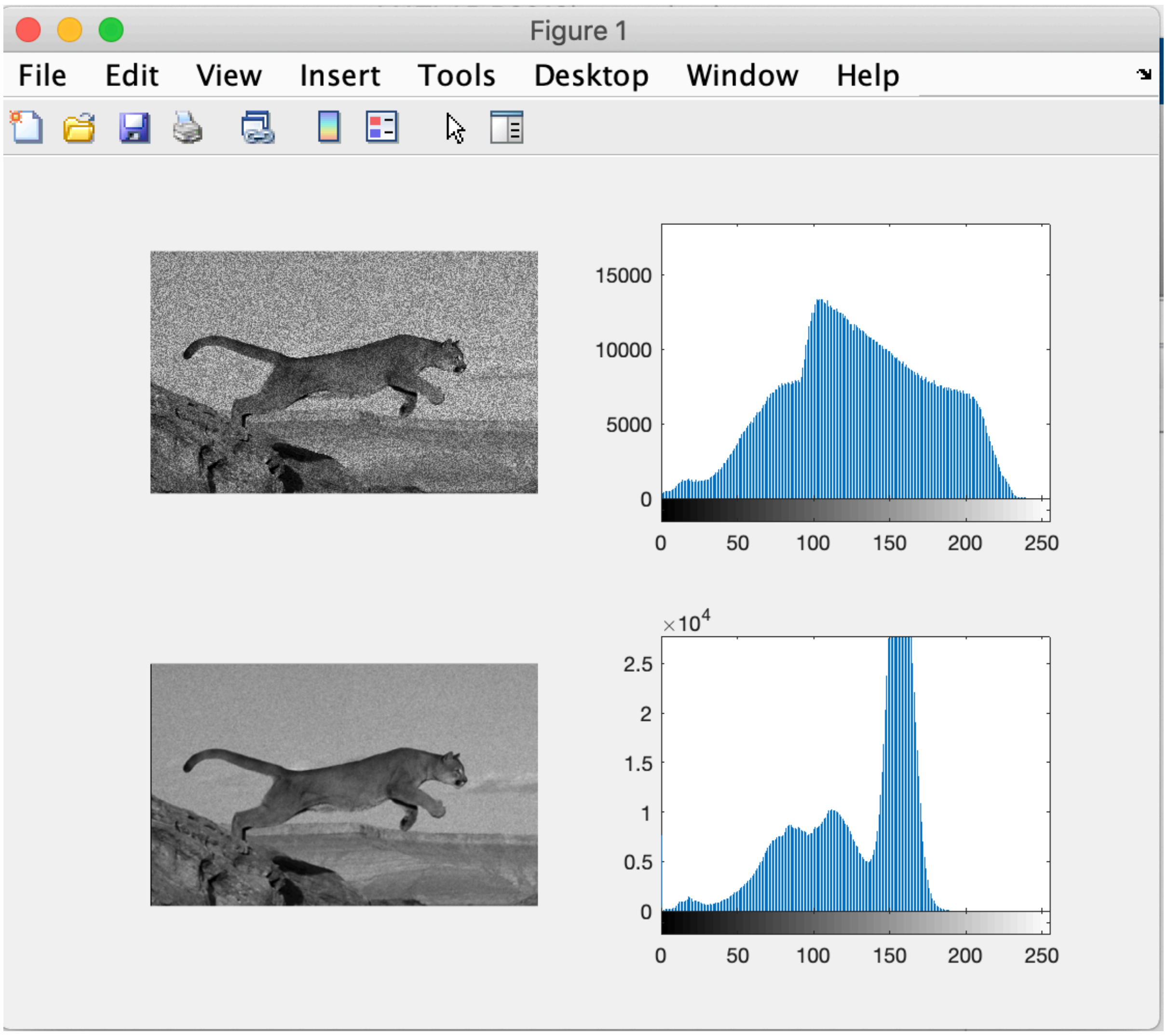

Figure 3. Box filtered image with K = 3

As its shown in the figure 3, second image has more blur and the noise is reduced in this case. If one increase the window size to K = 20, image will contain more blur and has a different kind of noise which should be reduced.

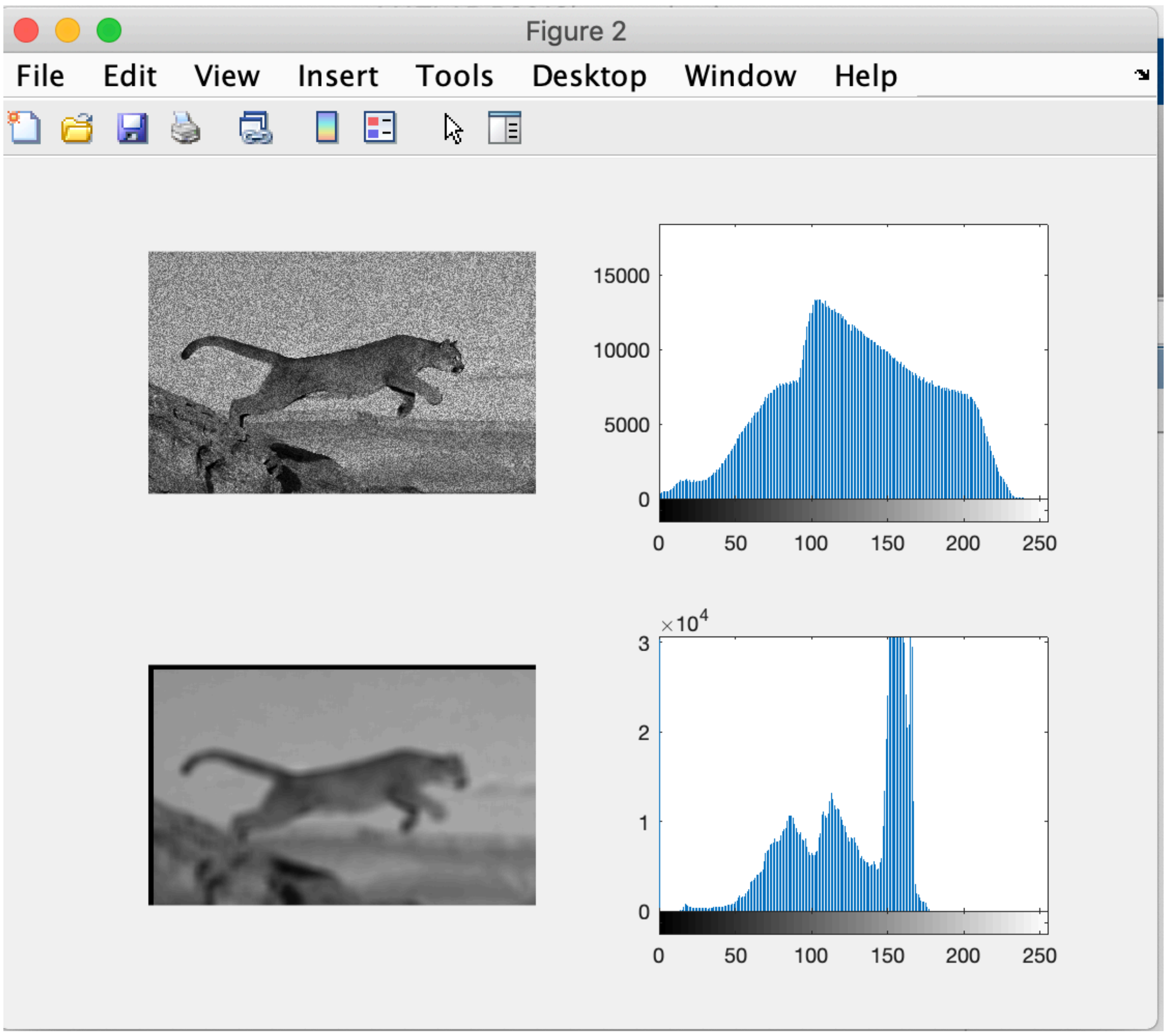

Figure 4. Box filtered image with K = 20

Comparing Figure 3 and Figure 4 one can see that both box filters increased the blur of the image but when the window size gets bigger image becomes more blurry. This is because box filters takes the mean value of window. Using bigger window size leds less pixels with different values. In other words, adjacent pixels will have similar intensity values.

QUESTION 4 – LOCAL MAXIMUM AND LOCAL MINIMUM

In this question, Sliding window method will be used again like previous question. But in this case, the center pixel of the window will be changed with the local minimum (minimum intensity valued pixel of the window) and the local maximum (maximum intensity valued pixel of the window).

```
function img2 = lab1locmaxmin(img,K)

[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end

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);
       maxValue=max(subImg(:));
       minValue=min(subImg(:));
       imgMin(i,j)= minValue;
       imgMax(i,j)= maxValue;
    end
end

img = uint8(img);
imgMin = uint8(imgMin);
imgMax = uint8(imgMax);

figure;
subplot(1,3,1);
imshow(img);
subplot(1,3,2);
imshow(imgMax);
subplot(1,3,3);
imshow(imgMin)
end
```

Used a double loop to iterate over the pixels and at each loop, code fills the imgMin and imgMax with local min and local max values of the window. After that converts three images to uint8 and plots them. (Used 2x2 grid in the lab for plotting, modified here for report purpose).

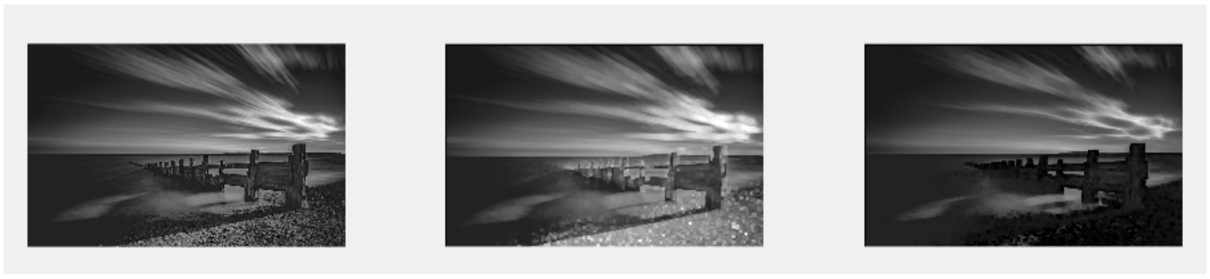

Figure 5. Shows the filtered images with K = 3

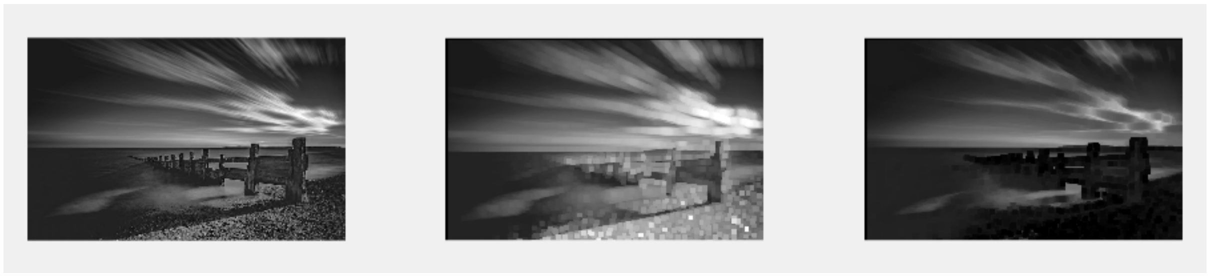

Figure 6. Shows the filtered images with K = 5

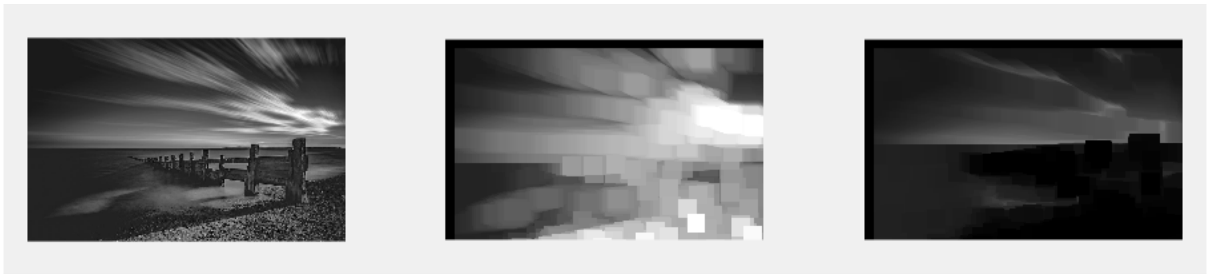

Figure 7. Shows the filtered images with K = 20

As one can guess from comparing Figures above, higher window size creates darker or brighter images. Higher window size will decrease the quality of image and result noise. Using max filter reduces the overall intensity values which are close to 0. Vice versa, having maximum filter reduces the number of pixels which are close to 255.

P.S. In order to run a function imread should be used e.g.

lab1locmaxmin(imread('currentImage.png'),20)