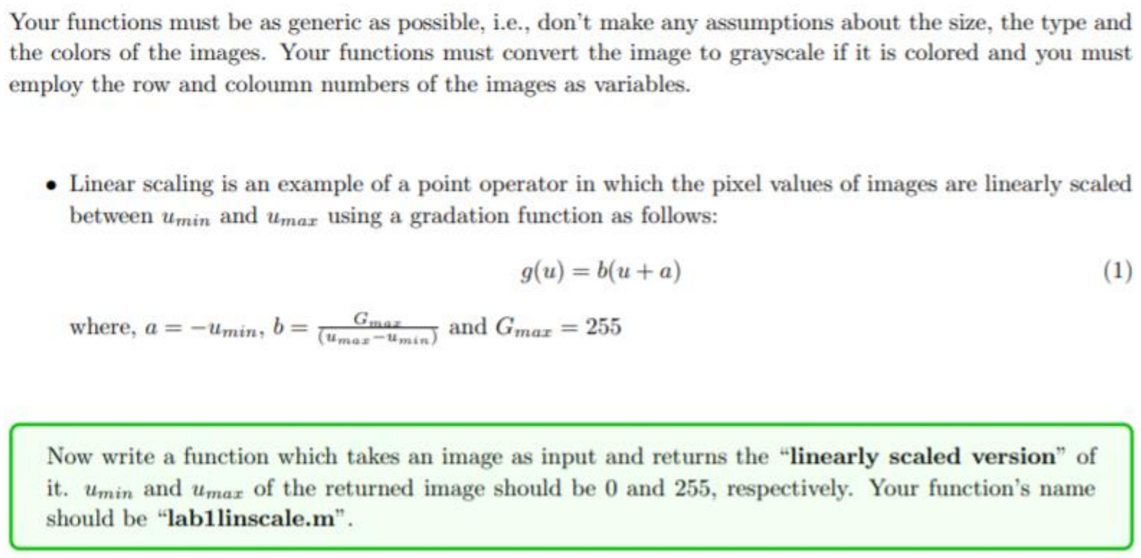
EE 417-POSTLAB REPORT 1

NAME: Nidanur Günay
ID:24231

LAB TOPIC: Scaling and Filtering in Grayscale Images

4 tasks were given during the lab hours. Each formula for the given task is explained in pdf.

- This was the first task that we supposed to implement the linear scaling as MatLab function. This filter is used for the generation of each pixel to get in range 0-255. By this filter image becomes more equalized distributed as seen in histogram diagrams in Fig 1.4.



Your functions must be as generic as possible, i.e., don't make any assumptions about the size, the type and the colors of the images. Your functions must convert the image to grayscale if it is colored and you must employ the row and coloumn numbers of the images as variables.

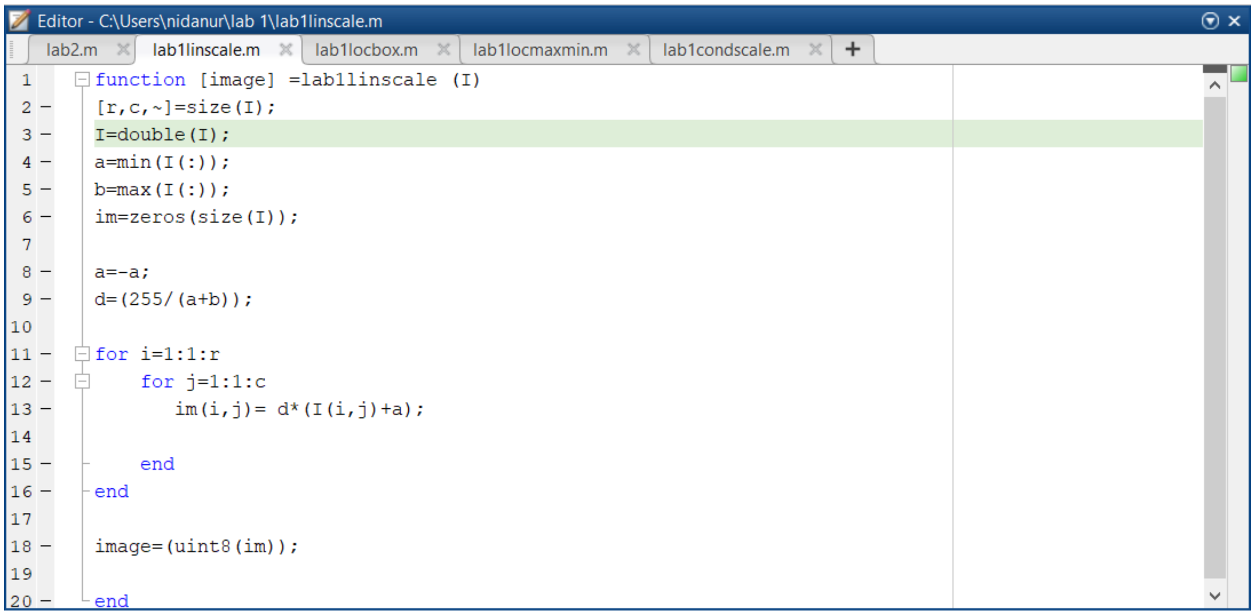
- Linear scaling is an example of a point operator in which the pixel values of images are linearly scaled between u_{min} and u_{max} using a gradation function as follows:

$$g(u) = b(u + a) \tag{1}$$

where, $a = -u_{min}$, $b = \frac{G_{max}}{(u_{max} - u_{min})}$ and $G_{max} = 255$

Now write a function which takes an image as input and returns the "**linearly scaled version**" of it. u_{min} and u_{max} of the returned image should be 0 and 255, respectively. Your function's name should be "**lab1linscale.m**".

Fig 1.1



```
function [image] =lab1linscale (I)
[r,c,~]=size(I);
I=double(I);
a=min(I(:));
b=max(I(:));
im=zeros(size(I));

a=-a;
d=(255/(a+b));

for i=1:1:r
    for j=1:1:c
       im(i,j)= d*(I(i,j)+a);

    end
end

image=(uint8(im));

end
```

Fig 1.2

This is the function that I wrote in the lab lecture on 2nd October. My function takes image as parameter and I store the number of rows as r parameter and number of columns on c parameter.I changed my values to double type to have accurate result. To find min and max value of my image matrix, I need to convert my matrix to an array. I used (I(:)) parameter in order to convert my matrix to an array. I created zeros matrix with the size of my image matrix to store the linearly scaled values. I created a and b values according to given formula. Then I scaled the each pixel by the inner 2 for loop. And my result image is on the Fig 1.3. Also my result image's histogram graph is on Fig 1.5.

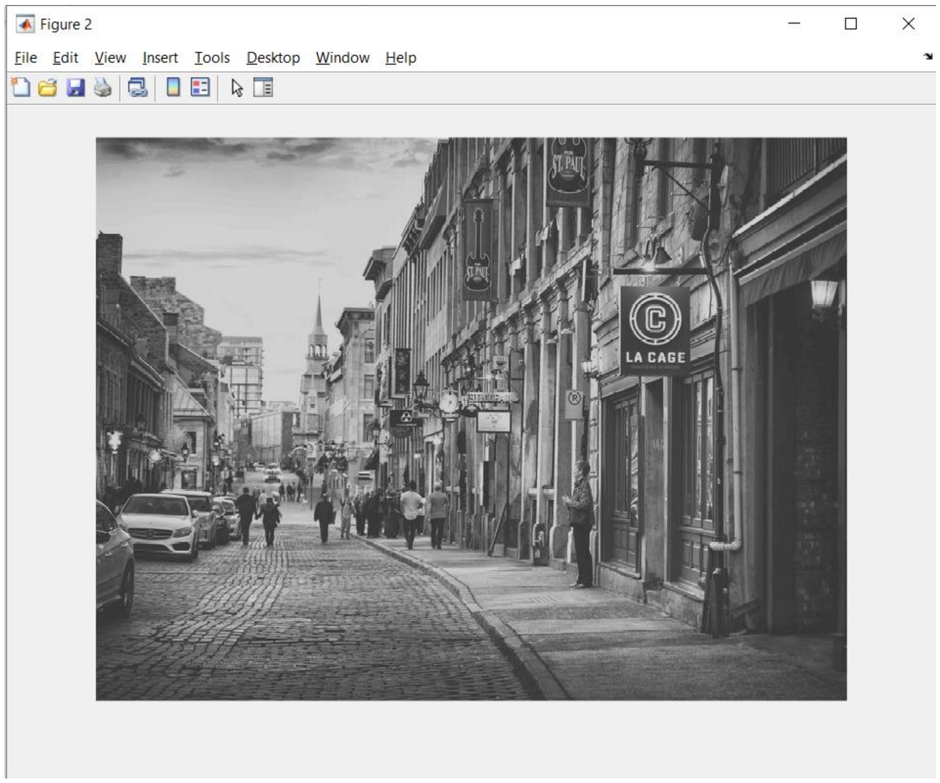


Figure 2
File Edit View Insert Tools Desktop Window Help
LA CAGE

Fig 1.3

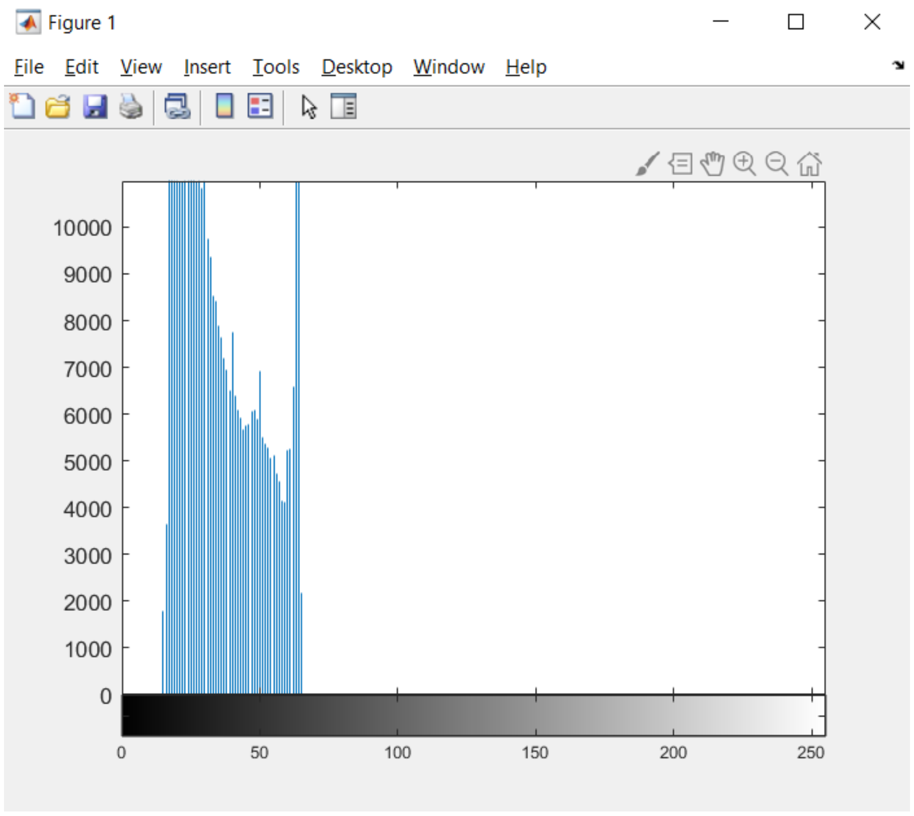


Figure 1
File Edit View Insert Tools Desktop Window Help
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

Fig 1.4

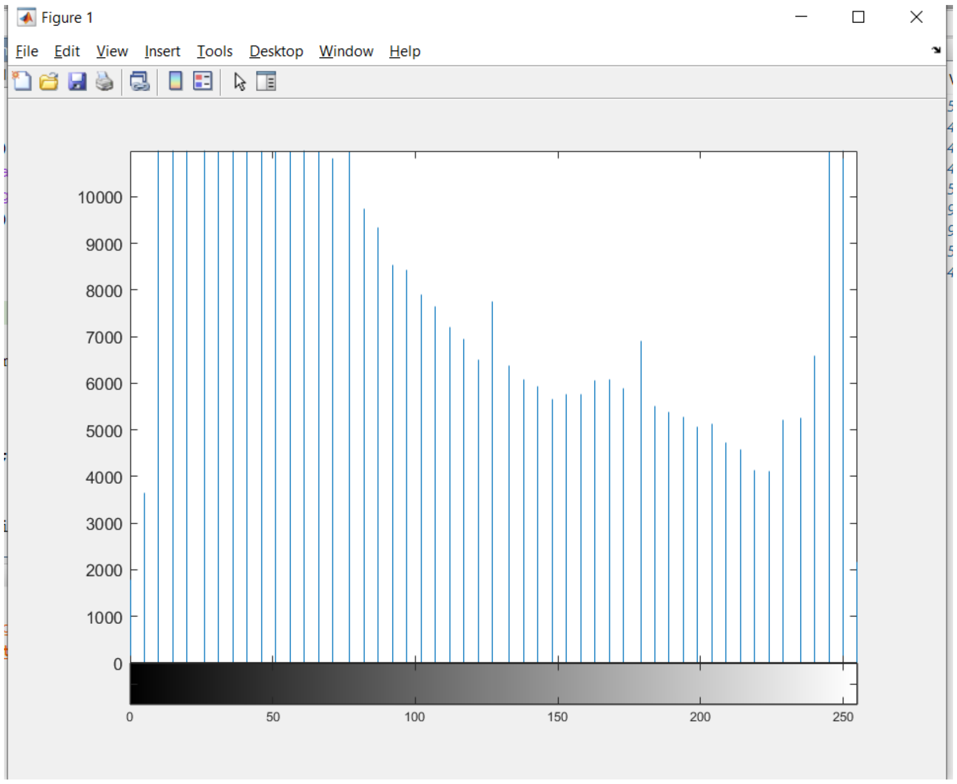
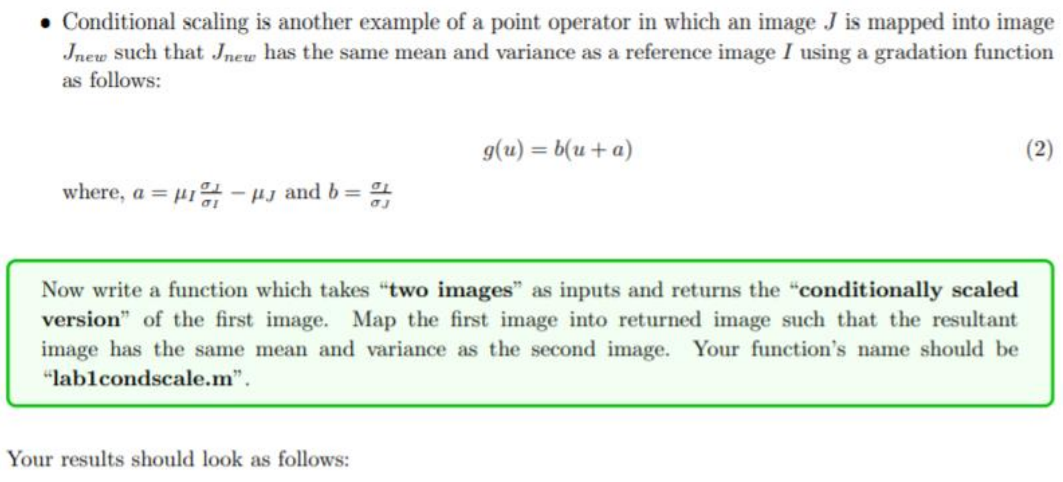


Fig 1.5

As seen in Fig 1.4, previous image histogram become dense in the range 0-75. After the filtering, image pixels became more equally distributed in the range 0-255 (Fig 1.5).

- The second task that we supposed to is conditional scaling. It is the scaling type that is refferenced by another images mean and standard deviation value.Explanation about what is expecting us is on Fig 2.1.



- Conditional scaling is another example of a point operator in which an image J is mapped into image J_{new} such that J_{new} has the same mean and variance as a reference image I using a gradation function as follows:

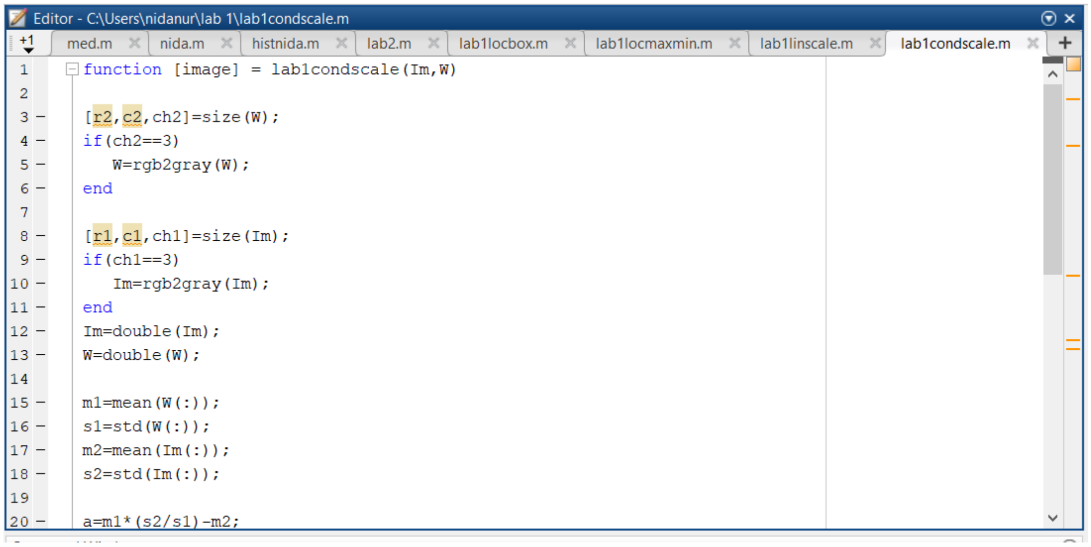
$$g(u) = b(u + a) \qquad (2)$$

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

Now write a function which takes "**two images**" as inputs and returns the "**conditionally scaled version**" of the first image. Map the first image into returned image such that the resultant image has the same mean and variance as the second image. Your function's name should be "**lab1condscale.m**".

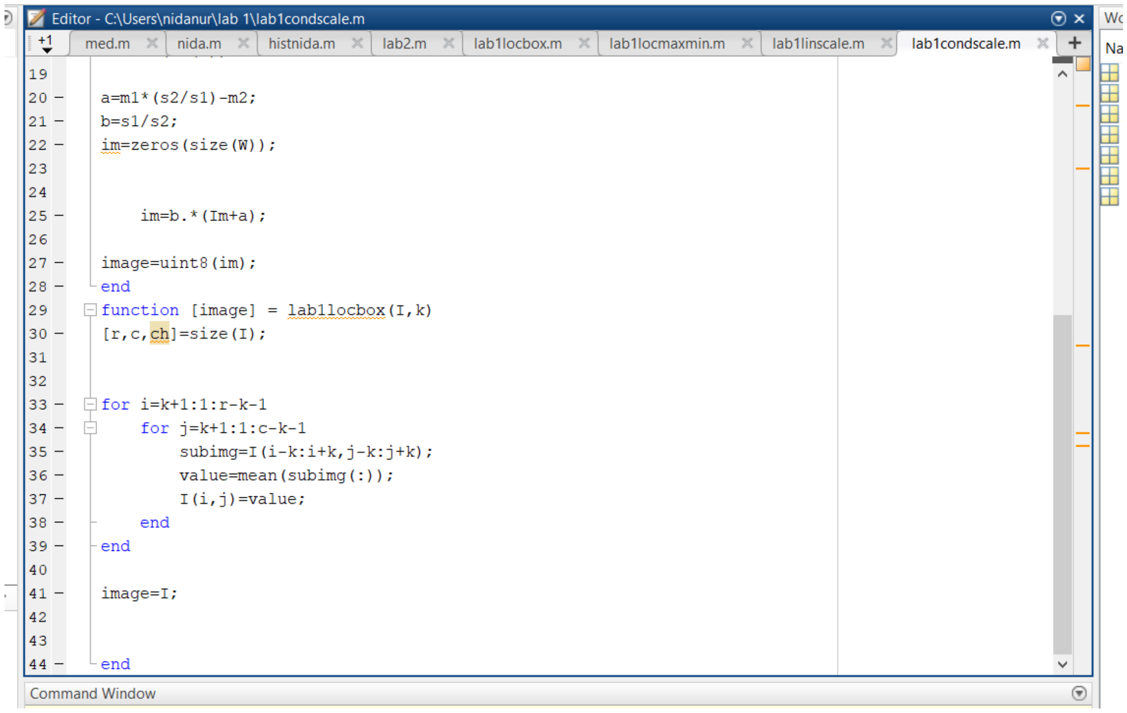
Your results should look as follows:

Fig 2.1



```
function [image] = lab1condscale(Im,W)

[r2,c2,ch2]=size(W);
if(ch2==3)
   W=rgb2gray(W);
end

[r1,c1,ch1]=size(Im);
if(ch1==3)
   Im=rgb2gray(Im);
end
Im=double(Im);
W=double(W);

m1=mean(W(:));
s1=std(W(:));
m2=mean(Im(:));
s2=std(Im(:));

a=m1*(s2/s1)-m2;
```

Fig 2.2



```

a=m1*(s2/s1)-m2;
b=s1/s2;
im=zeros(size(W));


    im=b.*(Im+a);

image=uint8(im);
end
function [image] = lab1locbox(I,k)
[r,c,ch]=size(I);


for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=mean(subimg(:));
        I(i,j)=value;
    end
end

image=I;


end
```

Fig 2.3

As the Fig 2.2 and 2.3 I created a function which takes 2 images as parameter.I checked them if they are rgb or not by if condition. If they are rgb image, I converted them into grayscale image.I converted the image matrix's values to double value type in order to calculate exact standard deviation and mean. After the converting step I calculated the mean and standard deviation for each image. Then I calculated the a and b values according to given formula. And I implement the formula for the each element in the main image by using 2 inner for loop. My result image is shown on the Fig 2.4.

Fig 2.5

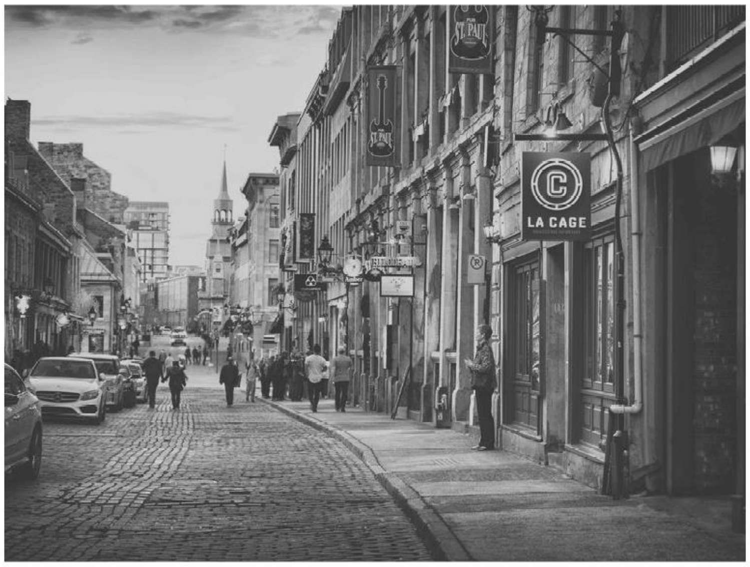


Fig 2.4

Main purpose of this filtering is taking reference by another image. And we are trying to change the pixel values in the image as the reference image's variance and mean values. Reference image is shown in the Fig 2.5.

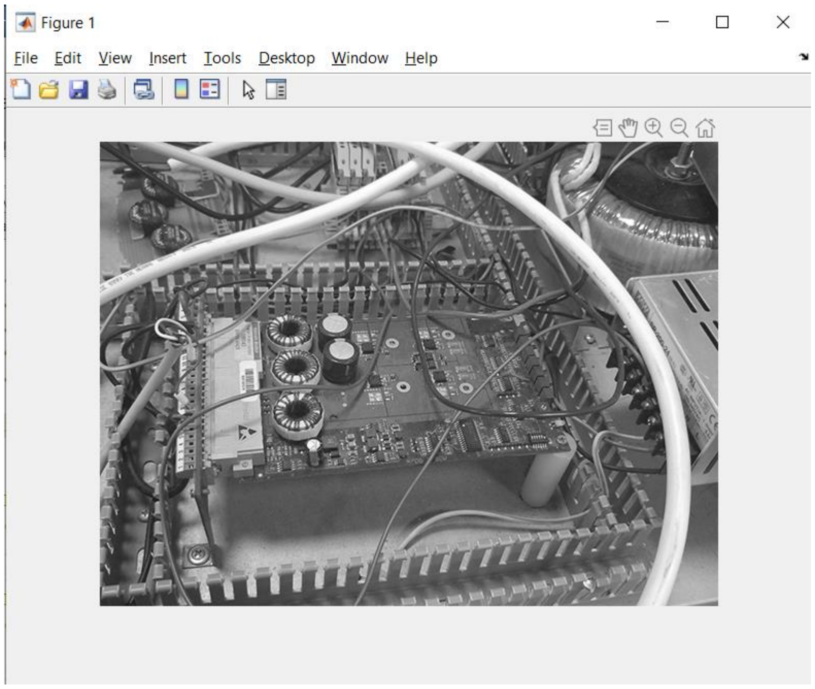
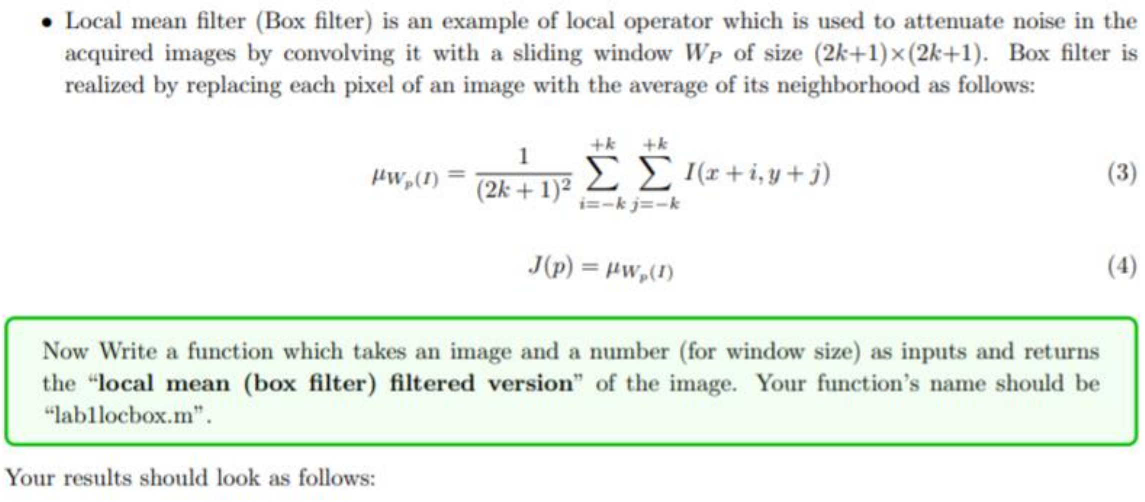


Fig 2.5

- The third task that is expecting us is implementing the Matlab code for the box filtering or local mean filter.The given task explained in the Fig 3.1



- Local mean filter (Box filter) is an example of local operator which is used to attenuate noise in the acquired images by convolving it with a sliding window W_P of size $(2k+1)\times(2k+1)$. Box filter is realized by replacing each pixel of an image with the average of its neighborhood as follows:

$$\mu_{W_p(I)} = \frac{1}{(2k+1)^2}\sum_{i=-k}^{+k}\sum_{j=-k}^{+k} I(x+i, y+j) \tag{3}$$

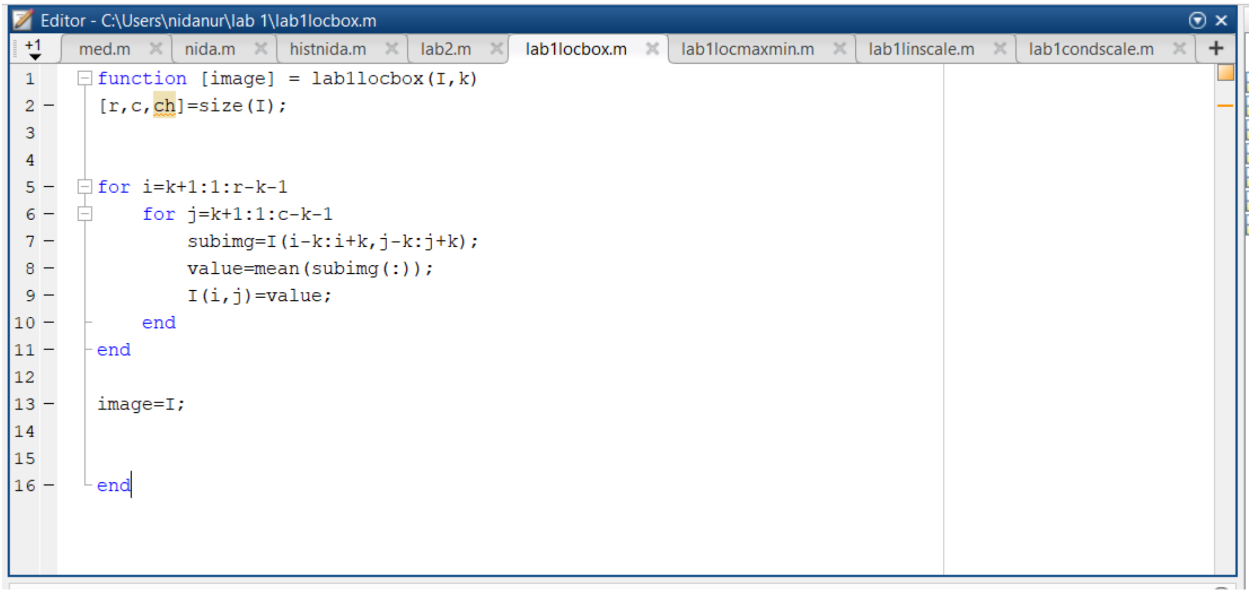
$$J(p) = \mu_{W_p(I)} \tag{4}$$

Now Write a function which takes an image and a number (for window size) as inputs and returns the "**local mean (box filter) filtered version**" of the image. Your function's name should be "lab1locbox.m".

Your results should look as follows:

Fig 3.1

As shown in the Fig 3.2 function gets 2 parameter as integer to determine the window size and image.To start from the first index, inner 2 loop starts the index I and j from k+1 and they iterates until the rownumber-k-1 and columnnumber-k-1.



Editor - C:\Users\nidanur\lab 1\lab1locbox.m

```
function [image] = lab1locbox(I,k)
[r,c,ch]=size(I);


for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=mean(subimg(:));
        I(i,j)=value;
    end
end

image=I;


end
```

Fig 3.2

We cropped our image as the given k value and applied the formula for the each subimage's matrix. In order to calculate the mean of submatrix we convert it to an array form.My f iltered image is shown on the Fig 3.3.It is more smoothened image and we get rid of the noise of the main image.

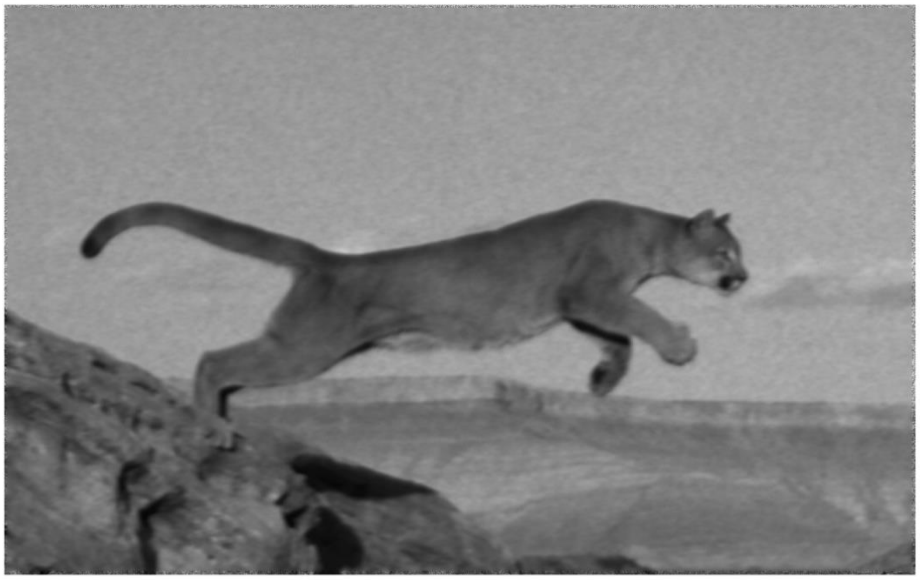
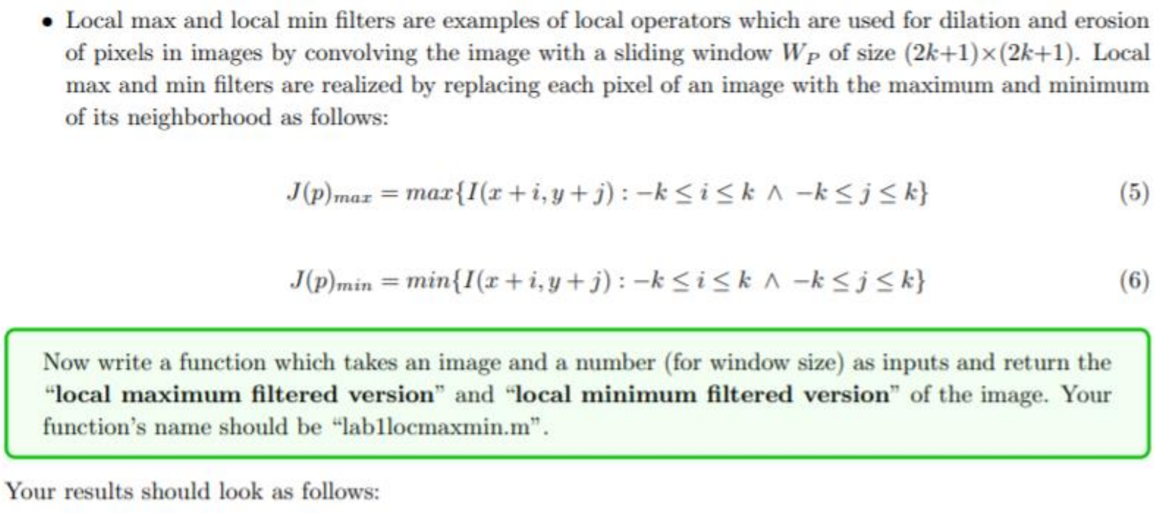


Fig 3.3

- The 4th task that we supposed to do is explained in the Fig 4.1. As the 3th task we supposed to take a window size and take an image as the parameter in our function.



- Local max and local min filters are examples of local operators which are used for dilation and erosion of pixels in images by convolving the image with a sliding window W_P of size $(2k+1)\times(2k+1)$. Local max and min filters are realized by replacing each pixel of an image with the maximum and minimum of its neighborhood as follows:

$$J(p)_{max} = max\{I(x+i, y+j) : -k \leq i \leq k \ \wedge \ -k \leq j \leq k\} \tag{5}$$

$$J(p)_{min} = min\{I(x+i, y+j) : -k \leq i \leq k \ \wedge \ -k \leq j \leq k\} \tag{6}$$

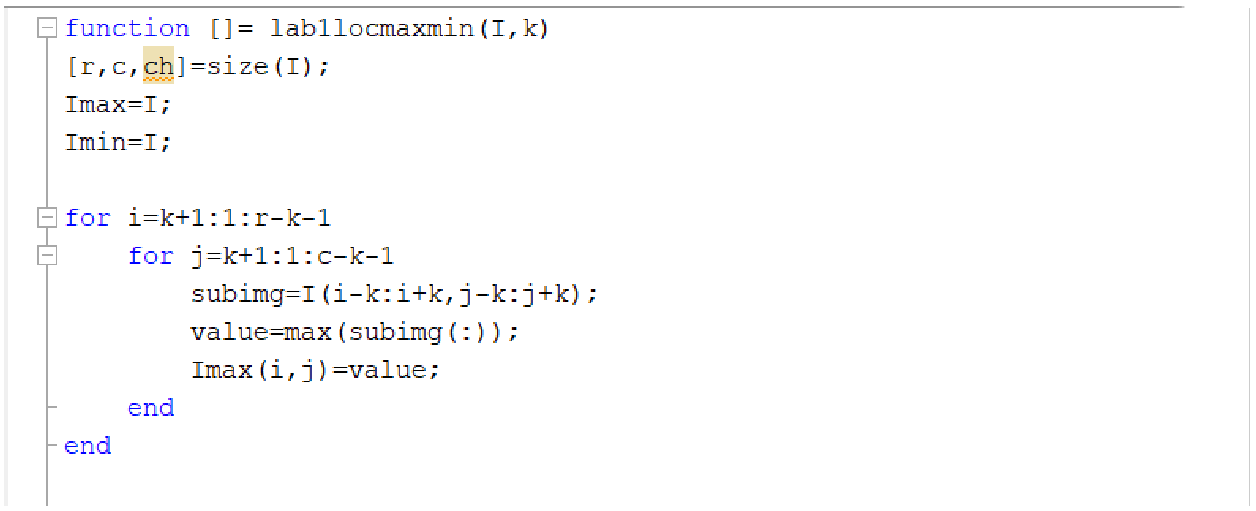
Now write a function which takes an image and a number (for window size) as inputs and return the "**local maximum filtered version**" and "**local minimum filtered version**" of the image. Your function's name should be "lab1locmaxmin.m".

Your results should look as follows:

Fig 4.1

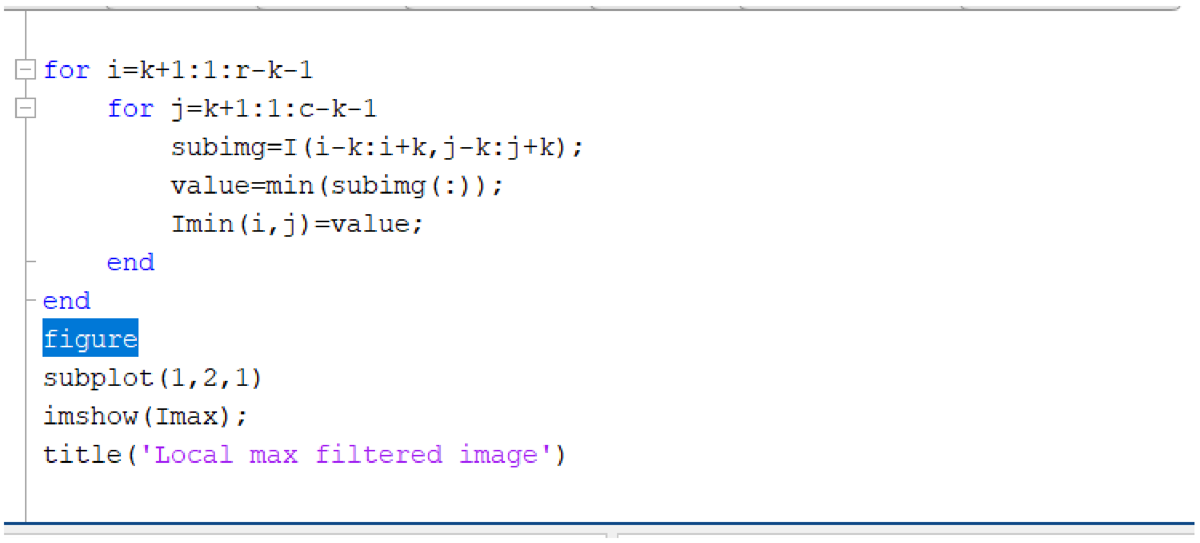
The function that I wrote during the lab hours is guve in the below. I took Image and k value as parameter in my fanction.I store the image size.I coppied my image into Imax and Imin variable. In my first loop I took the max values of the matrices that have been determined its sizes by the k value. And I changed my image by the local max filter. I applied the same steps to my Imin

image at the second loop.As a result I applied local max and local min filter to my Imax and Imin images.
My function converts void as a return value.



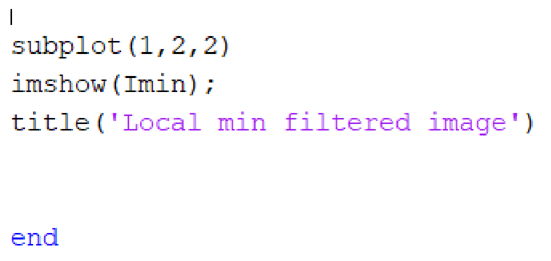
```
function []= lab1locmaxmin(I,k)
[r,c,ch]=size(I);
Imax=I;
Imin=I;

for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=max(subimg(:));
        Imax(i,j)=value;
    end
end
```

Fig 4.2



```
for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=min(subimg(:));
        Imin(i,j)=value;
    end
end
figure
subplot(1,2,1)
imshow(Imax);
title('Local max filtered image')
```

Fig 4.3



```
subplot(1,2,2)
imshow(Imin);
title('Local min filtered image')


end
```

Fig 4.4

However my resulted image is shown in Fig 4.5. Result image is not the images that I expected. The main reason is I forget the initialize the k value when I was calling the function as seen in Fig 4.6.

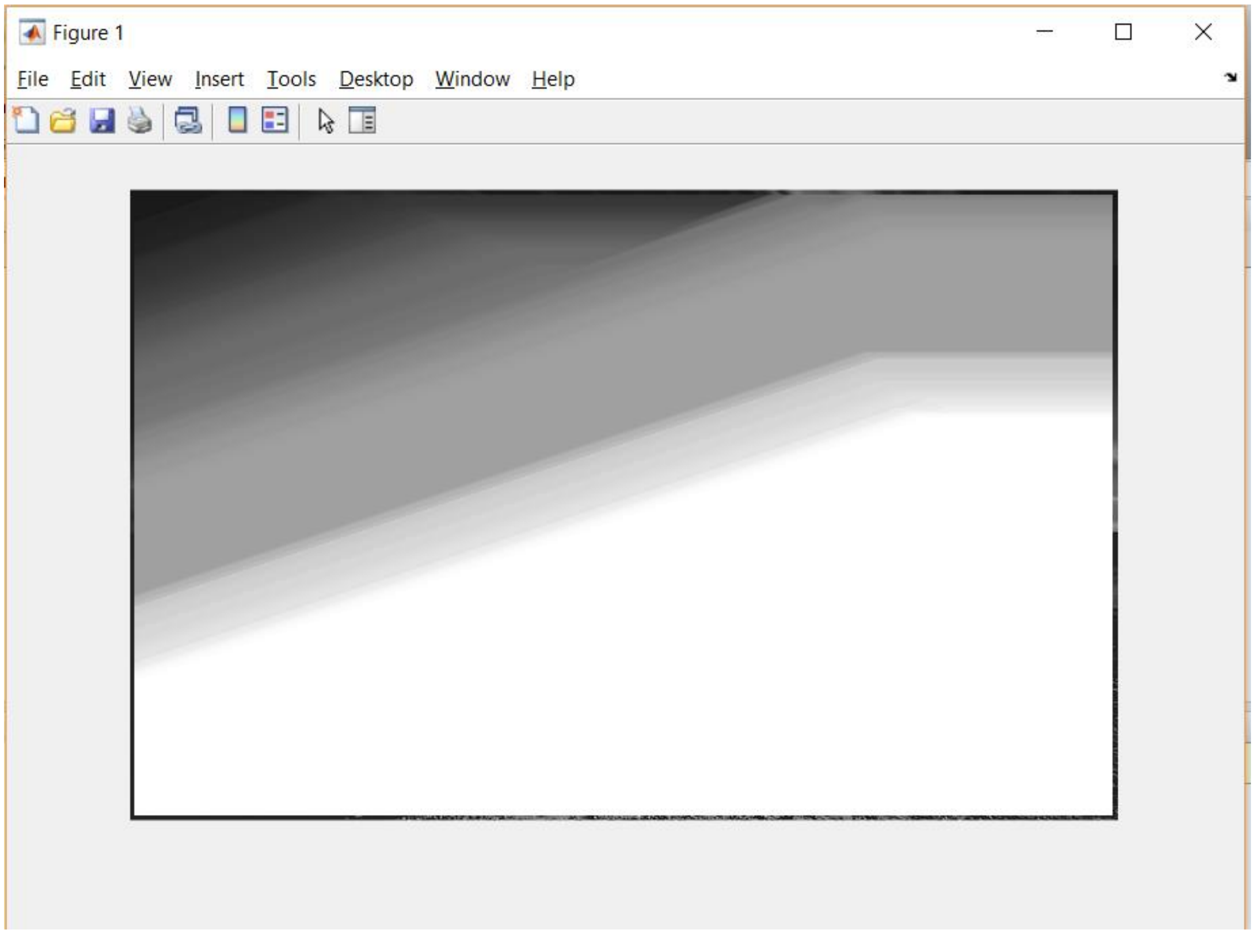
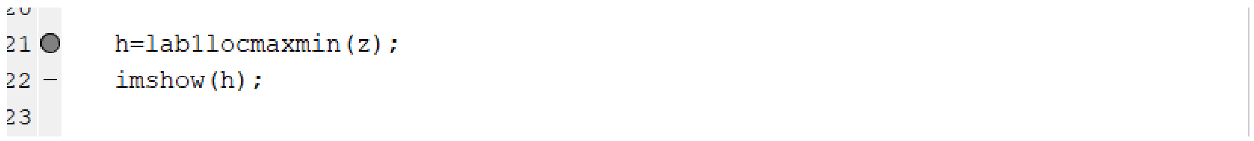


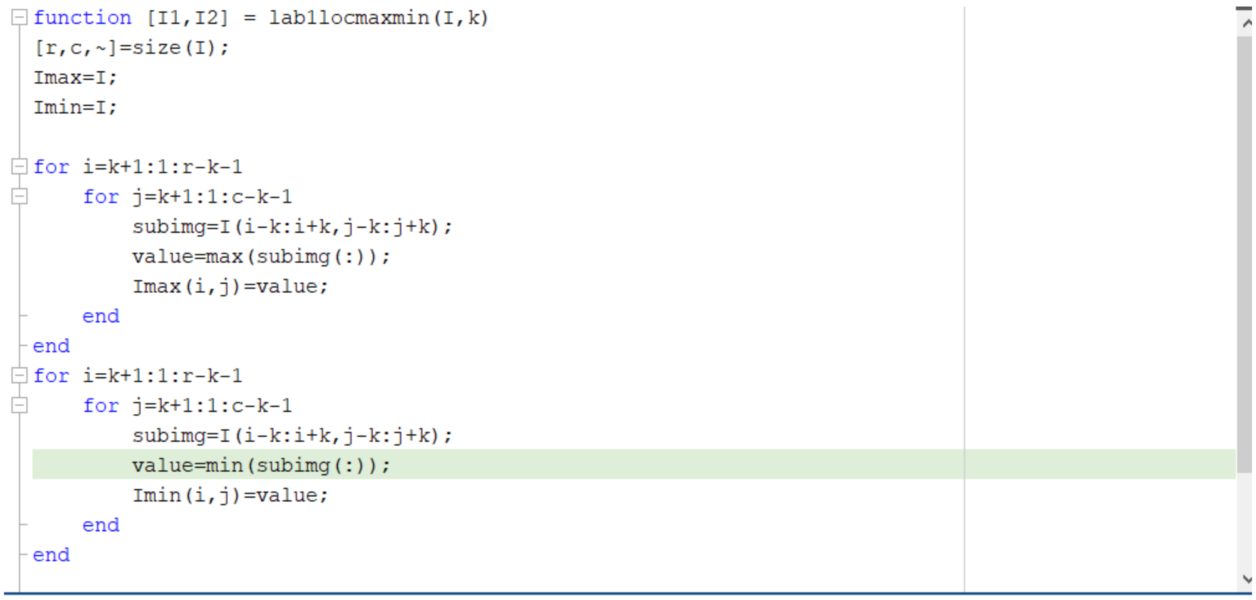
Fig 4.5



```
h=lab1locmaxmin(z);
imshow(h);
```

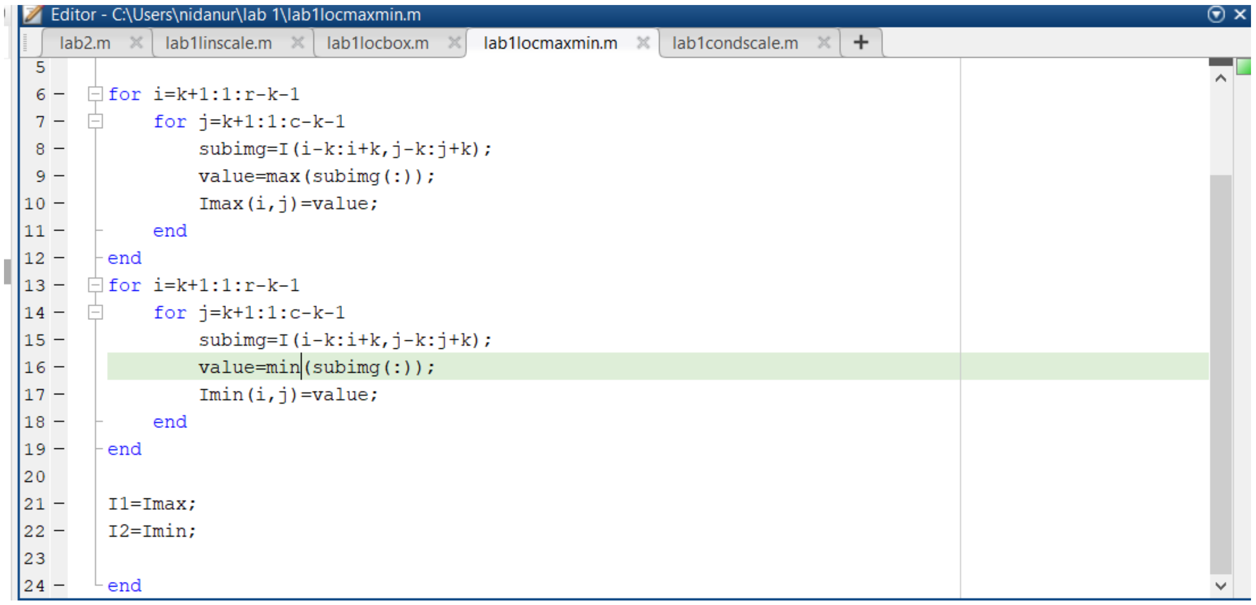
Fig 4.6.

I found the solution to returning to images as retur values. As a return value , I picked Imax and imin images As shown in the Fig 4.7 and 4.8.



```matlab
function [I1,I2] = lab1locmaxmin(I,k)
[r,c,~]=size(I);
Imax=I;
Imin=I;

for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=max(subimg(:));
        Imax(i,j)=value;
    end
end
for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=min(subimg(:));
        Imin(i,j)=value;
    end
end
```

Fig 4.7

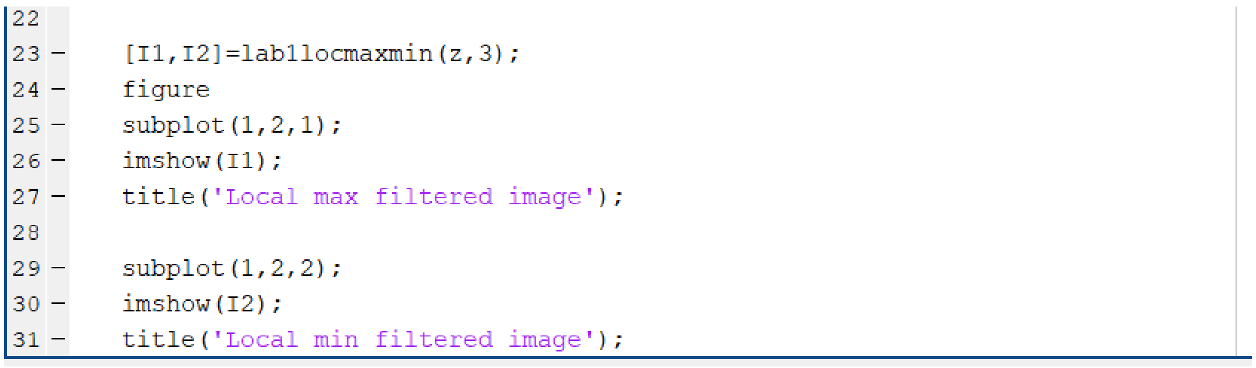


Editor - C:\Users\nidanur\lab 1\lab1locmaxmin.m

```matlab

for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=max(subimg(:));
        Imax(i,j)=value;
    end
end
for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=min(subimg(:));
        Imin(i,j)=value;
    end
end

I1=Imax;
I2=Imin;

end
```

Fig 4.8

My main in the Fig 4.9. I took Image array as I1 and I2. And I created a figure as shown in Figure 4.10.And by the change that ive done after, I get result what expected.



```
[I1,I2]=lab1locmaxmin(z,3);
figure
subplot(1,2,1);
imshow(I1);
title('Local max filtered image');

subplot(1,2,2);
imshow(I2);
title('Local min filtered image');
```

Fig 4.9

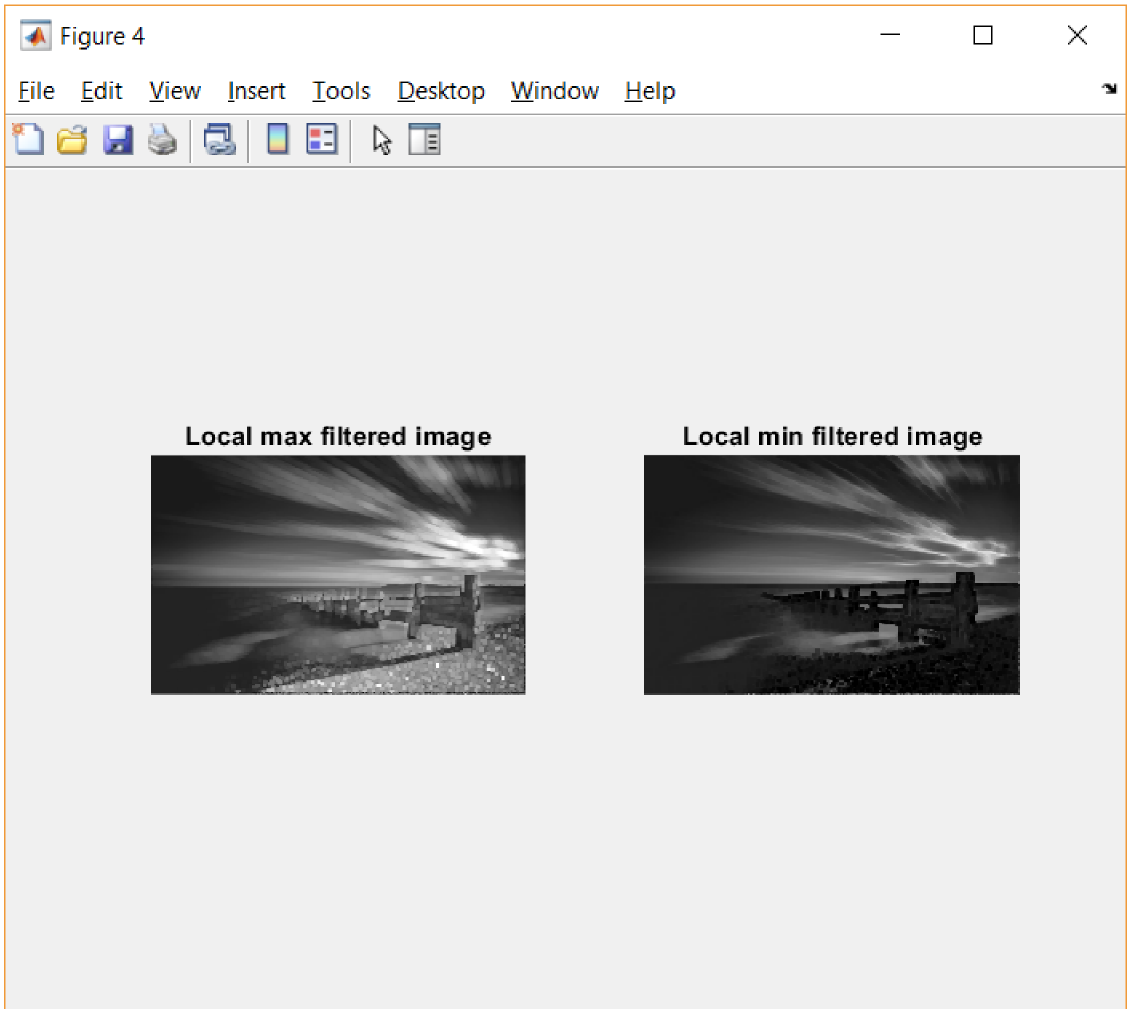
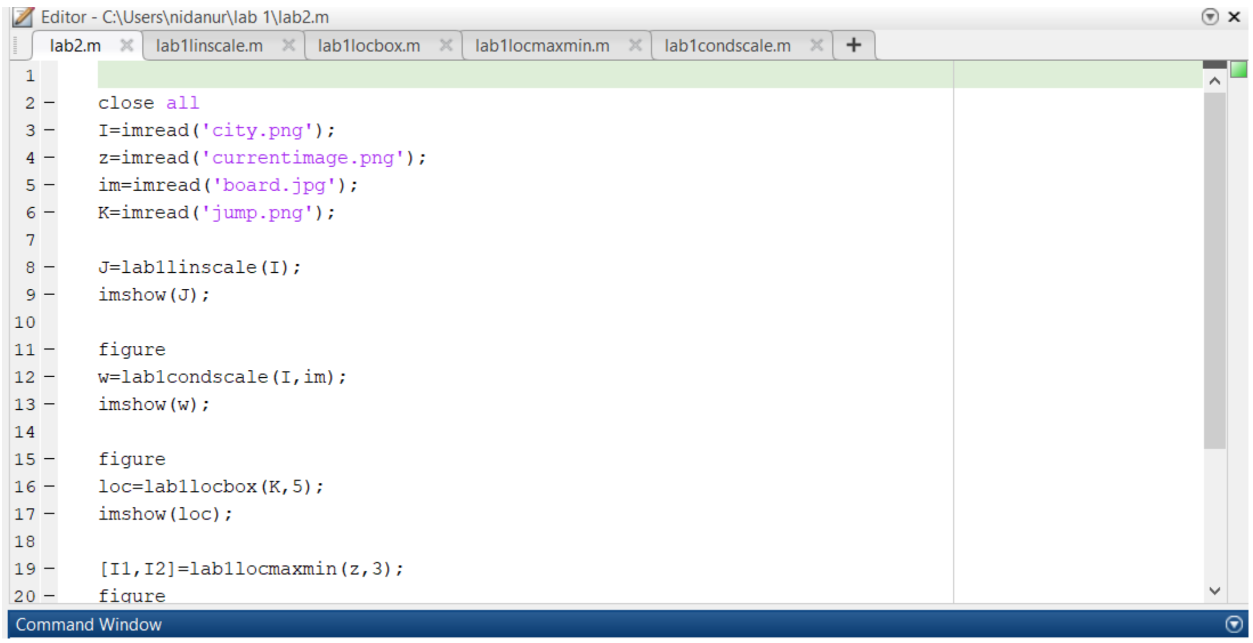


Fig 4.10

My main fuction secreenshots are in below.Fig 5.1 and Fig 5.2.

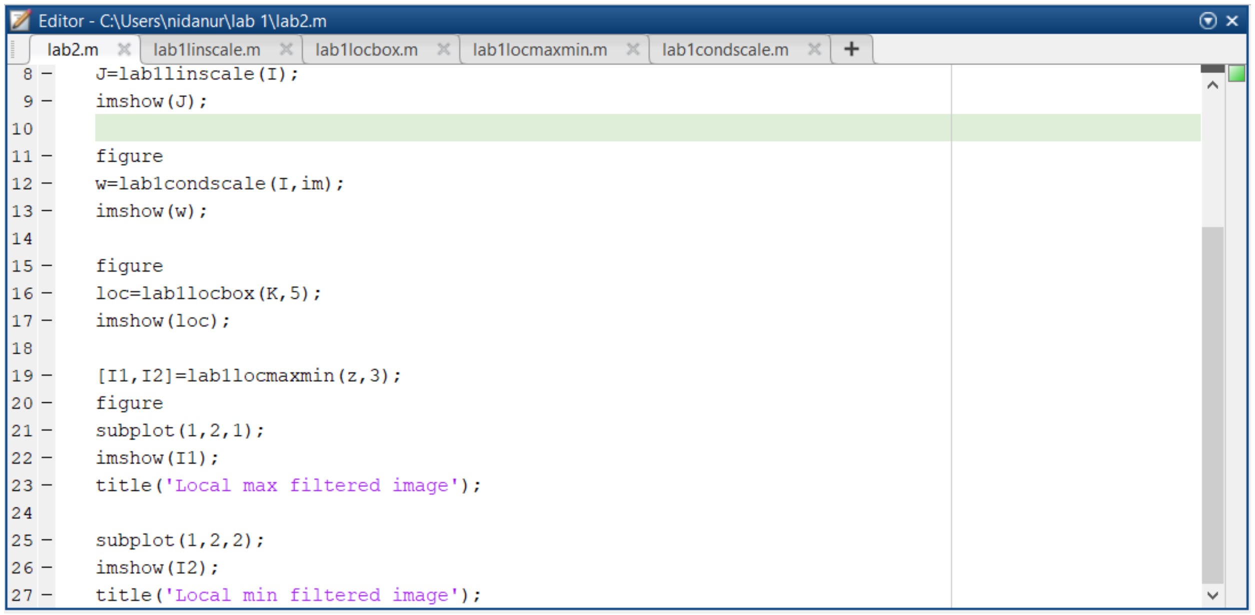

Editor - C:\Users\nidanur\lab 1\lab2.m

lab2.m | lab1linscale.m | lab1locbox.m | lab1locmaxmin.m | lab1condscale.m

```

close all
I=imread('city.png');
z=imread('currentimage.png');
im=imread('board.jpg');
K=imread('jump.png');

J=lab1linscale(I);
imshow(J);

figure
w=lab1condscale(I,im);
imshow(w);

figure
loc=lab1locbox(K,5);
imshow(loc);

[I1,I2]=lab1locmaxmin(z,3);
figure
```

Command Window

Fig 5.1


Editor - C:\Users\nidanur\lab 1\lab2.m

lab2.m | lab1linscale.m | lab1locbox.m | lab1locmaxmin.m | lab1condscale.m

```
J=lab1linscale(I);
imshow(J);

figure
w=lab1condscale(I,im);
imshow(w);

figure
loc=lab1locbox(K,5);
imshow(loc);

[I1,I2]=lab1locmaxmin(z,3);
figure
subplot(1,2,1);
imshow(I1);
title('Local max filtered image');

subplot(1,2,2);
imshow(I2);
title('Local min filtered image');
```

Fig 5.2

APPENDIX: CODES

lab1inscale.m

```
function [image] =lab1linscale (I)
[r,c,~]=size(I);
I=double(I);
a=min(I(:));
b=max(I(:));
im=zeros(size(I));

a=-a;
d=(255/(a+b));

for i=1:1:r
    for j=1:1:c
       im(i,j)= d*(I(i,j)+a);

    end
end

image=(uint8(im));

end
```

lab1locbox.m

```
function [image] = lab1locbox(I,k)
[r,c,~]=size(I);


for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=mean(subimg(:));
        I(i,j)=value;
    end
end

image=I;


end
```

lab1locmaxmin.m

```
function [I1,I2] = lab1locmaxmin(I,k)
[r,c,~]=size(I);
Imax=I;
Imin=I;

for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=max(subimg(:));
        Imax(i,j)=value;
    end
end
for i=k+1:1:r-k-1
    for j=k+1:1:c-k-1
        subimg=I(i-k:i+k,j-k:j+k);
        value=min(subimg(:));
        Imin(i,j)=value;
    end
end
```

lab1conscale.m

```
function [image] = lab1condscale(Im,W)

[r2,c2,ch2]=size(W);
if(ch2==3)
   W=rgb2gray(W);
end

[r1,c1,ch1]=size(Im);
if(ch1==3)
   Im=rgb2gray(Im);
end
Im=double(Im);
W=double(W);

m1=mean(W(:));
s1=std(W(:));
m2=mean(Im(:));
s2=std(Im(:));

a=m1*(s2/s1)-m2;
b=s1/s2;
im=zeros(size(W));


    im=b.*(Im+a);
```

```
image=uint8(im);
end
```

lab2.m

```
close all
I=imread('city.png');
z=imread('currentimage.png');
im=imread('board.jpg');
K=imread('jump.png');

J=lab1linscale(I);
imshow(J);
imhist(J);
figure
w=lab1condscale(I,im);
imshow(w);

figure
loc=lab1locbox(K,5);
imshow(loc);

[I1,I2]=lab1locmaxmin(z,3);
figure
subplot(1,2,1);
imshow(I1);
title('Local max filtered image');

subplot(1,2,2);
imshow(I2);
title('Local min filtered image');
```