EEE-321: Signals and Systems

Section-1

LAB Assignment 4

Boran KILIÇ

22103444

10/11/2023

Part 2

Derivation of 2D convolution formula:

2D impulse signal is defined as:

$$\delta[m,n] = \begin{cases} 1 & \text{if } m=0, n=0 \\ 0 & \text{otherwise} \end{cases}$$

$x[m,n]$ can be written as superposition of shifted impulse signals as follows:

$$x[m,n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l]\,\delta[m-k, n-l]$$

where $x[k,l]$ is the scale of impulse shifted k units in the vertical direction and l units in the horizontal direction.

Let $h[m,n]$ be the response of the system to $\delta[m,n]$ i.e. impulse response. Using the space invariance of the system, if $\delta[m-k, n-l]$ is the input, the output becomes $h[m-k, n-l]$. After that, using the linearity of the system $y[m,n]$ can be found as follows:

$$y[m,n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l]\,h[m-k, n-l]$$

which is donated as:

$$y[m,n] = x[m,n] ** h[m,n]$$

Let's interchange the order of summation by changing the indices of summation

Let $i = m-k$ and $j = n-l$.

$$y[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=\infty}^{\infty} x[m-i, n-j]\,h[i,j]$$

$$y[m,n] = h[m,n] ** x[m,n]$$

Part 3

3.1 Size of y[m,n]

Given relations:

$x[m,n]$ can only be nonzero within $0 \le m \le M_x-1$ and $0 \le n \le N_x-1$

$h[m,n]$ can only be nonzero within $0 \le m \le M_h-1$ and $0 \le n \le N_h-1$

$y[m,n]$ can only be nonzero within $0 \le m \le M_y-1$ and $0 \le n \le N_y-1$

we have the relation:

$$y[m,n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l]\, h[m-k, n-l]$$

Changing the variables according to this relation we have:

$x[k,l]$ can only be nonzero within $0 \le k \le M_x-1$ and $0 \le l \le N_x-1$ (1)

$h[M-k, n-l]$ can only be nonzero within $0 \le m-k \le M_h-1$ and $0 \le n-l \le N_h-1$ (2)

$y[m,n]$ can only be nonzero within $0 \le m \le M_y-1$ and $0 \le n \le N_y-1$ (3)

from (1) and (2) we have:

$$\begin{array}{l} 0 \le k \le M_x-1 \\ +\ 0 \le m-k \le M_h-1 \\ \hline 0 \le m \le M_x+M_h-2 \end{array} \qquad \begin{array}{l} 0 \le l \le N_x-1 \\ +\ 0 \le n-l \le N_h-1 \\ \hline 0 \le n \le N_x+N_h-2 \end{array} \quad (5)$$

combining (3) and (5)

$$0 \le m \le M_x+M_h-2 \qquad 0 \le n \le N_x+N_h-2$$

$$0 \le m \le M_y-1 \qquad 0 \le n \le N_y-1$$

It can be concluded that, for these inequalities to be true:

$M_y-1 \ge M_x+M_h-2$ therefore $M_y-1 = M_x+M_h-2$ at least

Similarly

$N_y-1 \ge N_x+N_h-2$ therefore $N_y-1 = N_x+N_h-2$ at least.

Therefore the answer becomes

$M_y = M_x+M_h-1$ and $N_y = N_x+N_h-1$

3.2 DSLSI2D function in MATLAB

Based on the calculations DSLSI2D function which calculates the 2D convolution of a given signal x with a given impulse response h is below:

```
function [y]=DSLSI2D(h,x)
[Mh,Nh] = size(h);
[Mx,Nx] = size(x);
y = zeros(Mx+Mh-1,Nx+Nh-1);
for k=0:Mh-1
      for l=0:Nh-1
      y(k+1:k+Mx,l+1:l+Nx)=y(k+1:k+Mx,l+1:l+Nx)+h(k+1,l+1)*x;
      end
end
end
```

The function is checked with the following code:

```
clear
x = [1 0 2; -1 3 1; -2 4 0]
h = [1 -1; 0 2]
y = DSLSI2D(h,x)
```

The output of this code is below:

```
x =
     1     0     2
    -1     3     1
    -2     4     0

h =
     1    -1
     0     2

y =
     1    -1     2    -2
    -1     6    -2     3
    -2     4     2     2
     0    -4     8     0
```

The output was consistent with the expected output in the lab manual therefore the code is working as expected.

Part 4 Image Denoising

In this part the task was to extract the noise from a given image. The noise is high frequency components of the image therefore in order to eliminate the noise a low pass filter should be used. The impulse response of a typically used 2D FIR low pass filter is given below:

$$h[m,n] = \begin{cases} 0 & \text{if } m<0 \text{ or } n<0 \text{ or } m>M_h-1 \text{ or } n>N_h-1 \\ \text{sinc}\left\{B\left(m-\frac{M_h-1}{2}\right)\right\}\text{sinc}\left\{B\left(n-\frac{N_h-1}{2}\right)\right\} & \text{otherwise} \end{cases}$$

Where B is the free parameter which determines the bandwidth of the filter. In order to implement this filter on MATLAB the following code is written. Before writing the MATLAB code I have deleted the "`figure;`" command from the **DisplayMyImage** function in order to be able to use the **subplot** command, otherwise it would not work.

```
clear
figure;
x=ReadMyImage('Part4.bmp');
subplot(2, 2, 1);
DisplayMyImage(x);
title('unprocessed image')

D = 22103444;
D7 =rem(D,7);
Mh = 30 +D7;
Nh = Mh;

B = 0.7;
h=zeros(Mh-1,Nh-1);
for m = 1:Mh-1
for n = 1:Nh-1
h(m,n)= sinc(B*(m-(Mh-1)/2))*sinc(B*(n-(Nh-1)/2));
end
end
y = DSLSI2D(h,x);
subplot(2, 2, 2);
DisplayMyImage(y);
title('B=0.7')

B = 0.4;
h=zeros(Mh-1,Nh-1);
for m = 1:Mh-1
for n = 1:Nh-1
h(m,n)= sinc(B*(m-(Mh-1)/2))*sinc(B*(n-(Nh-1)/2));
end
end
y = DSLSI2D(h,x);
subplot(2, 2, 3);
DisplayMyImage(y);
title('B=0.4')

B = 0.1;
h=zeros(Mh-1,Nh-1);
for m = 1:Mh-1
for n = 1:Nh-1
h(m,n)= sinc(B*(m-(Mh-1)/2))*sinc(B*(n-(Nh-1)/2));
end
end
y = DSLSI2D(h,x);
```

```
subplot(2, 2, 4);
DisplayMyImage(y);
title('B=0.1')
```

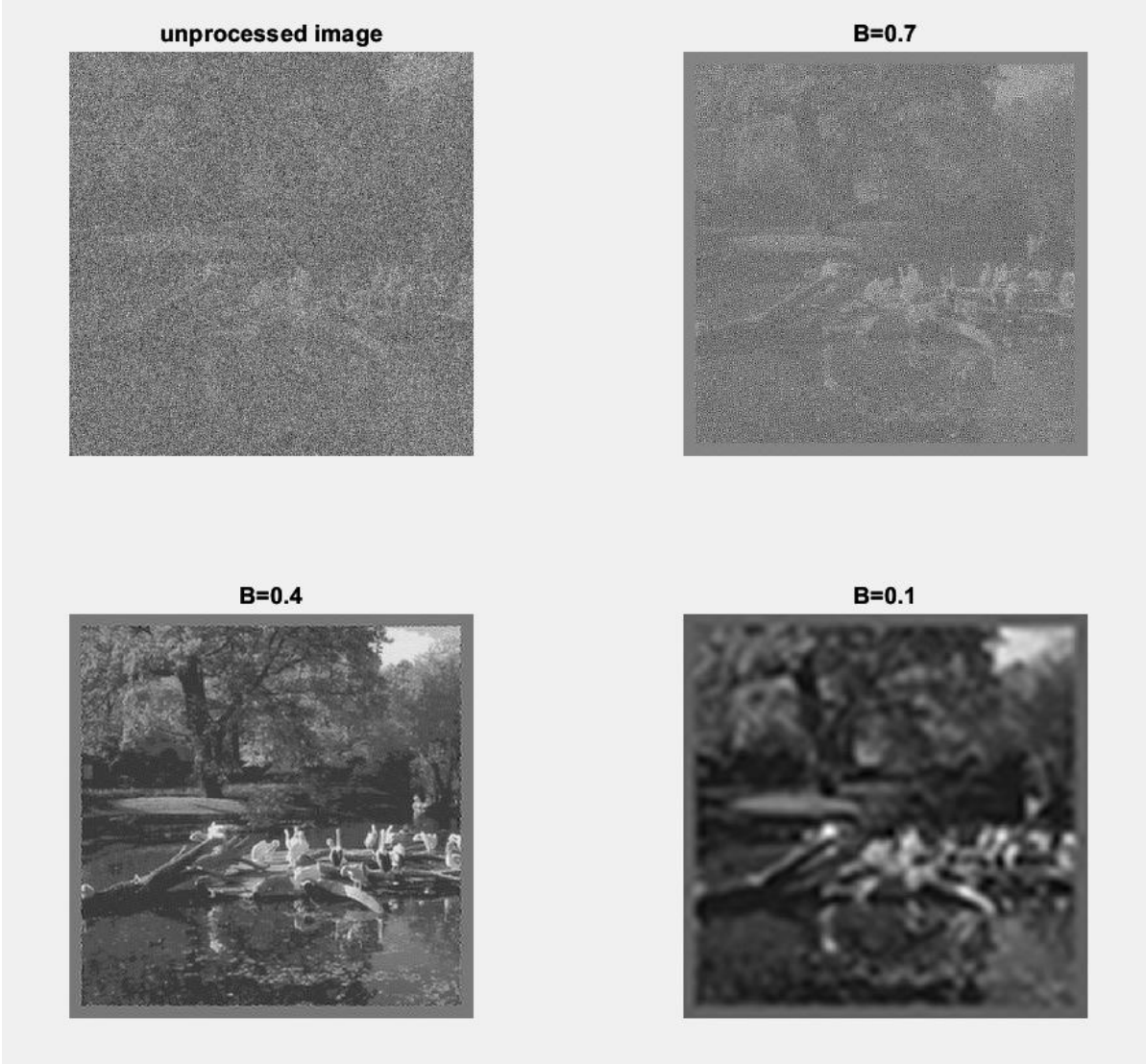

Figure 1: Processed images "Part4.bmp" with different bandwidths

Since B determines the bandwidth of the low pass filter as B decreases the bandwidth of the filter gets narrower i.e. cut off frequency gets smaller. As it can be seen from the Figure 1, when B was 0.7 there are still some quite visible noise. When B was 0.4 it can be seen that the image is rather clear. However, if we increase the bandwidth more the edges will be deleted as well since they are also high frequency components of the original image. Therefore the resolution of the image will drop. So the optimal point to fully recover the original image without demaging it from the noise was B = 0.4 for this case. Image

denoising is a useful tool in the sense of image processing, since it enables recovery of a possibly important image.

Part 5 Edge Detection

In this part, edge detection process is applied to the following image. The edges of an image is the high frequency components of the image. Therefore in order to be able to detect the edges one should take convolution of the image with the impulse response of a high pass filter.

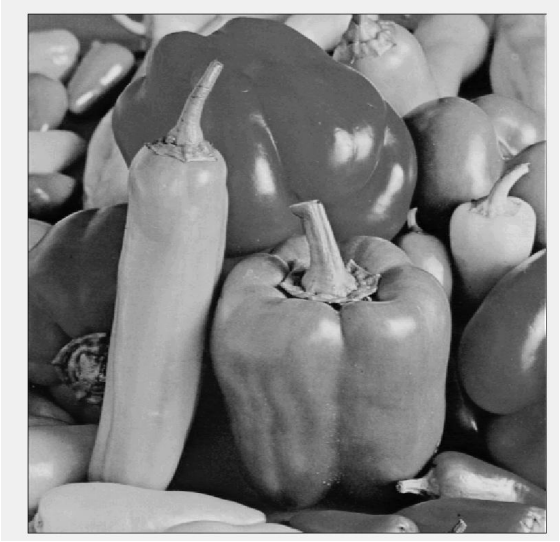

Figure 2: Original image named "Part5.bmp"

We have used three different high pass filters to process the image, one was to detect vertical edges of the image, another was to detect the horizontal edges an the last was a combination of the two which gave all the vertical and horizontal edges.

$$h_1[m,n] = \begin{cases} 0.5 & \text{if } m = 0, n = 0 \\ -0.5 & \text{if } m = 0, n = 1 \\ 0 & \text{otherwise} \end{cases}$$

h1[m,n] checks for the high frequencies in the horizontal direction therefore it detects the vertical edges.

$$h_2[m,n] = \begin{cases} 0.5 & \text{if } m = 0, n = 0 \\ -0.5 & \text{if } m = 1, n = 0\,. \\ 0 & \text{otherwise} \end{cases}$$

h2[m,n] checks for the high frequencies in the vertical direction therefore it detects the horizontal edges.

h1[m,n] and h2[m,n] are called edge detection kernels.

$$h_3[m,n] = 0.5h_1[m,n] + 0.5h_2[m,n]$$

h3[m,n] is the combination of h1[m,n] and h2[m,n] so, it checks both vertical and horizontal high frequencies therefore it detects both vertical and horizontal edges.

MATLAB code for computing y1 y2 and y3 is below:

```
clear
x=ReadMyImage('Part5.bmp');
figure;
DisplayMyImage(x);


h1=zeros(2,2);
h1(1,1)=0.5;
h1(1,2)=-0.5;
y1 = DSLSI2D(h1,x);


s1 = y1.^2;
figure;
subplot(1, 2, 1);
DisplayMyImage(x);
subplot(1, 2, 2);
DisplayMyImage(s1);


h2=zeros(2,2);
h2(1,1) = h1(1,1);
h2(2,1) = h1(1,2);
y2 = DSLSI2D(h2,x);
s2 = y2.^2;
figure;
subplot(1, 2, 1);
DisplayMyImage(x);
subplot(1, 2, 2);
DisplayMyImage(s2);

h3=zeros(2,2);
h3 = 0.5.*h1+0.5.*h2;
y3 = DSLSI2D(h3,x);
s3 = y3.^2;
figure;
subplot(1, 2, 1);
DisplayMyImage(x);
subplot(1, 2, 2);
DisplayMyImage(s3);
```

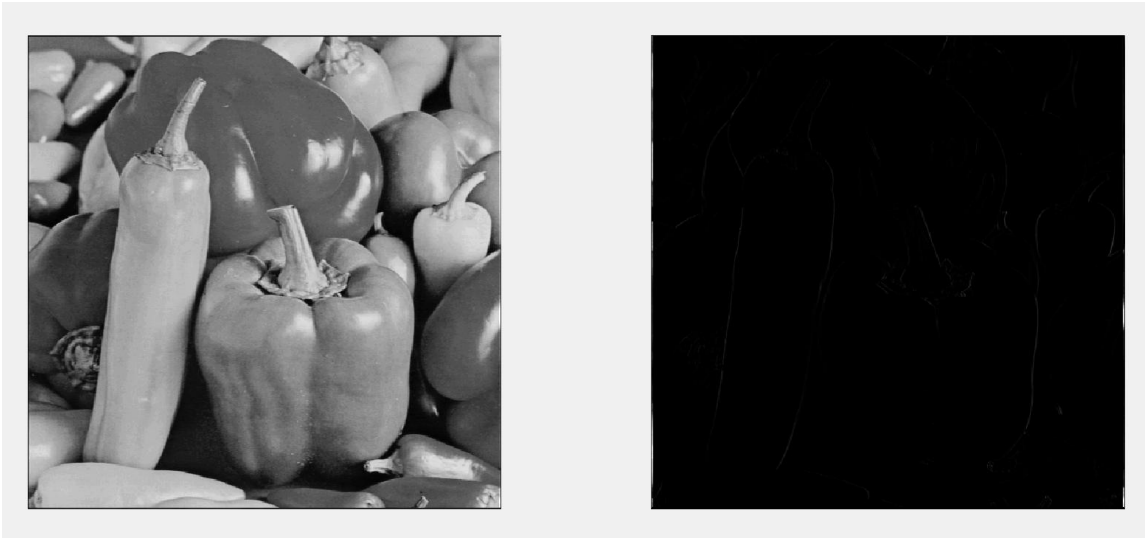

Figure 3: Vertical edges

In Figure 3 it can be seen that when the kernel h1which contains high frequency signal in the horizontal direction is convoluted with the low frequency image on the right, high changes on the horizontal direction are highlighted and the low frequency parts are darkened. This is because convolution basically a dot product and dot product is used to detect similarities because its magnitude is the highest when a similarity occurs. High numbers correspond to white pixels and low numbers correspond to black pixels. Therefore in the resulting image, it can be seen that vertical edges are highlighted white.

Figure 4: Horizontal edges

Explanation of Figure 4 is very similar to the Figure 3 where the only difference is we are checking vertical changes with the kernel h2 which contains high frequency signal on the vertical direction. Therefore horizontal edges are detected and highlighted with white.

Figure 5: Vertical and horizontal edges

In Figure 5, kernels h1 and h2 are combined it means it detects and highlights vertical and horizontal edges simultaneously

The reason why we display the squares of the y signals instead of just displaying them is, we want to have a uniform display where edges are white and the other is black. If we did not take the square of the signals edges will be black or white based on the correlation and other places would be gray.

Part 6 A sample pattern recognition application

In this part a pattern recognition application is applied using a matching filter. The image to be investigated is below in Figure 6:

Figure 6: "Part6x.bmp", Turkish National Football Team

Form Figure 6 we want to find the pattern that matches with the below pattern in Figure 7

Figure 7: "Part6h.bmp", Volkan Demirel's face 180° rotated

The image in Figure 7 will be convoluted with the original image in Figure 6 and y[m,n] will be calculated and we will display the absolute value of y[m,n] in order to get more a clear view of the matching points.

The reason why this image was rotated is that it will be rotated 180° again when we apply convolution. As it is stated in the previous part when we apply convolution to an image it highlights the matching parts.

MATLAB code for pattern recognition application:

```
clear
x=ReadMyImage('Part6x.bmp');
figure;
DisplayMyImage(x);

figure;
h=ReadMyImage('Part6h.bmp');
DisplayMyImage(h);

y = DSLSI2D(h,x);
s1= abs(y);
figure;
DisplayMyImage(s1);

s3 = s1.^3;
figure;
DisplayMyImage(s3);

s5 = s1.^5;
figure;
DisplayMyImage(s5);
```

The resulting images are below:

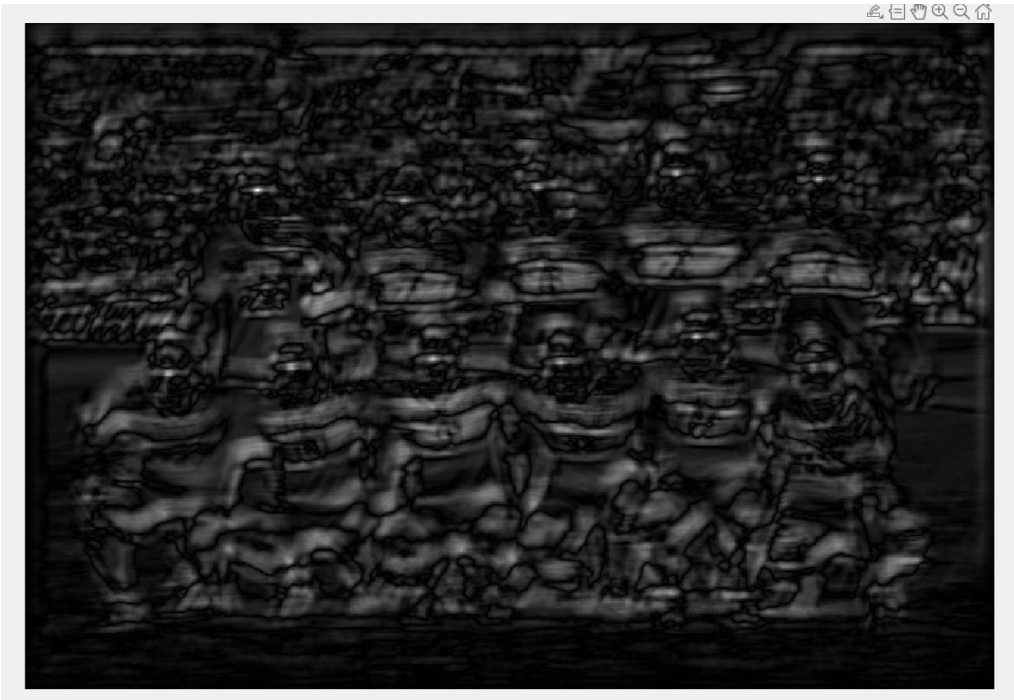

Figure 8: $|y[m,n]|$

If we look at Figure 8 it can be seen that the resulting image does not give clear information about where the pattern is matched with the original image, besides, it also highlights unrelated parts of the image as well.

Figure 9: $|y[m,n]|^3$

When we took the 3rd power magnitudes of the brighter points are magnified. Therefore, the location of the desired pattern can be found by locating the brightest point. Which is in the upper left in Figure 9.

Figure 10: $|y[m,n]|^5$

When the 5th power is taken the location of the matching pattern can be seen more clearly which is on the upper left on the Figure 10. Taking the 5th power yields a more clear result on the location of the matching pattern compared to Figures 8 and 9. However, taking the 3rd cube was pretty sufficient as it can be seen from Figure 9 that the location of the matching pattern is the brightest point.