EEE 321-Signals and Systems

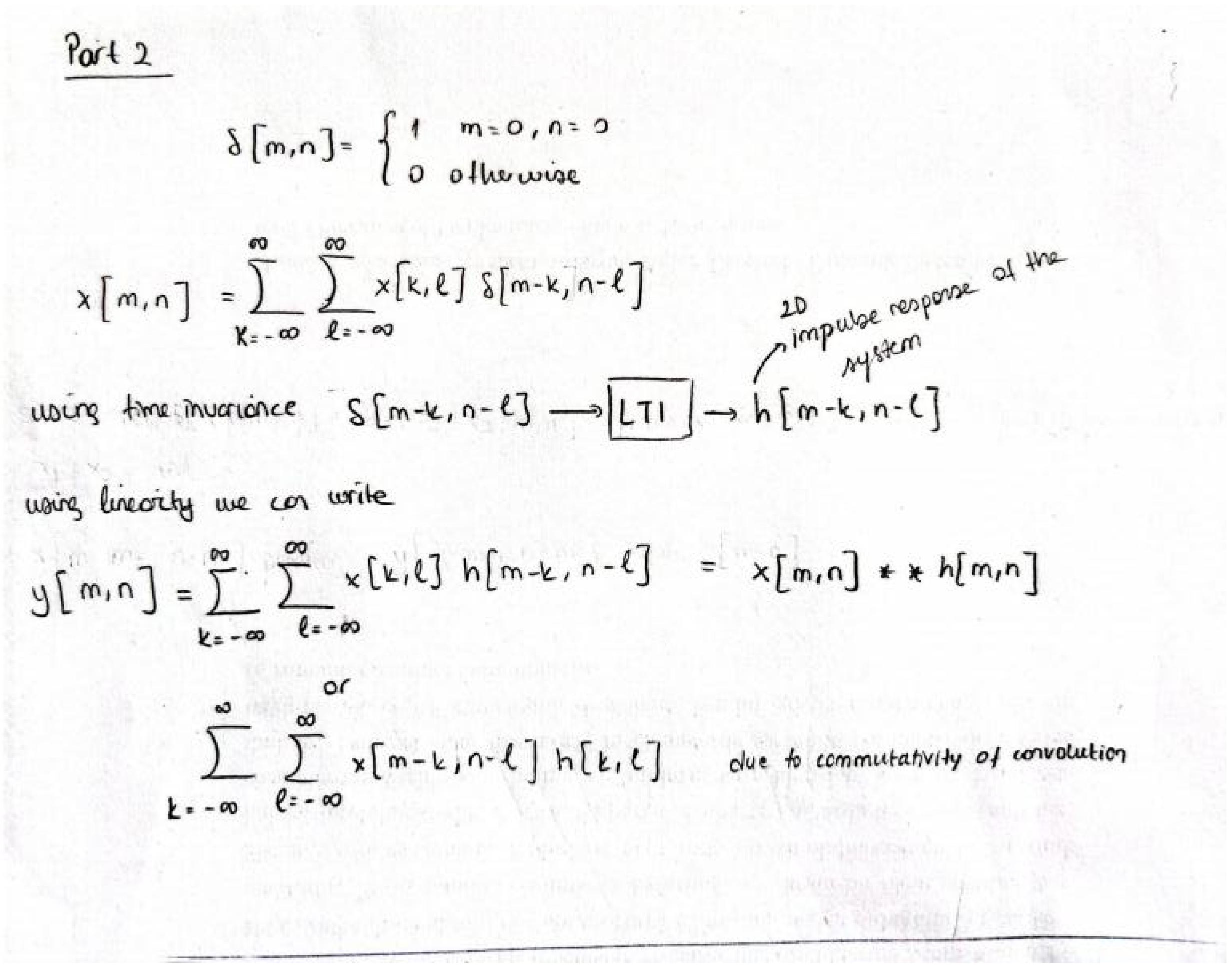
-Lab 04-

Defne Yaz Kılıç
Section 001
22102167
11.10.23

Image Processing

Part 2

For a 2D DT LTI system the convolution sum is as follows.



Part 2

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

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

2D impulse response of the system

using time invariance $\delta[m-k,n-\ell] \longrightarrow \boxed{\text{LTI}} \longrightarrow h[m-k,n-\ell]$

using linearity we can write

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

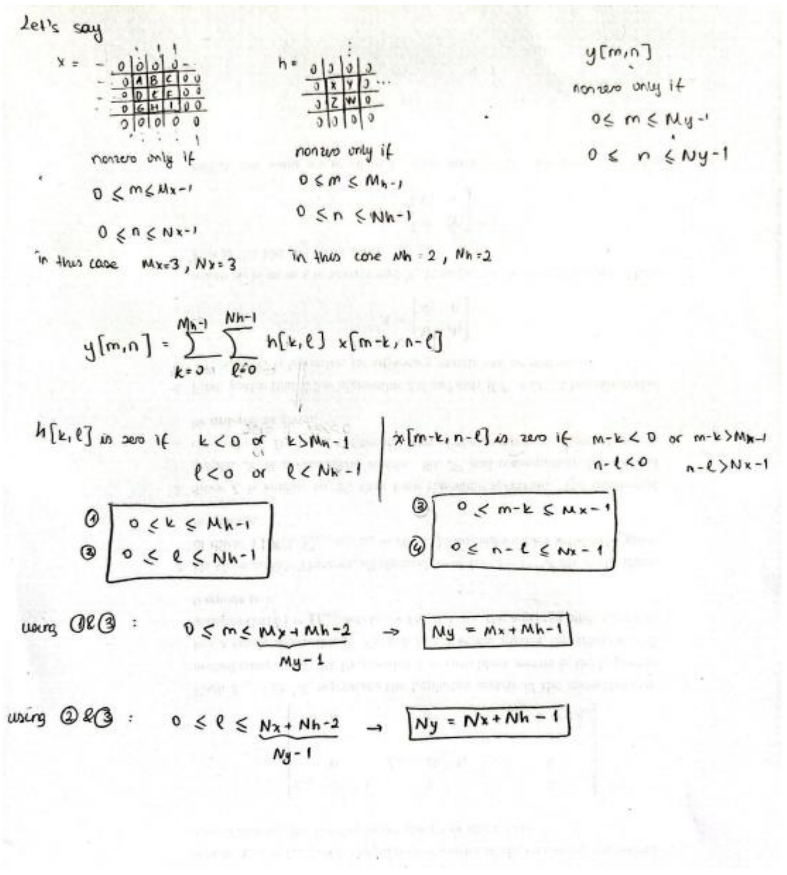
or

$$\sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} x[m-k,n-\ell]\, h[k,\ell]$$ due to commutativity of convolution

Fig.1. *Part 2 explanations*

Part 3

The nonzero input boundaries for output matrix can be calculated as follows.



Let's say

$x = \begin{array}{cccccc} 0 & 0 & 0 & 0 & & \\ 0 & A & B & C & 0 & 0 \\ 0 & D & E & F & 0 & 0 \\ 0 & G & H & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$ $\quad h = \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & X & Y & 0 \\ 0 & Z & W & 0 \\ 0 & 0 & 0 & 0 \end{array}$ $\quad y[m,n]$

nonzero only if $\quad$ nonzero only if $\quad$ nonzero only if

$0 \leq m \leq M_x-1 \quad 0 \leq m \leq M_h-1 \quad 0 \leq m \leq M_y-1$

$0 \leq n \leq N_x-1 \quad 0 \leq n \leq N_h-1 \quad 0 \leq n \leq N_y-1$

in this case $M_x=3$, $N_x=3$ $\quad$ in this case $M_h=2$, $N_h=2$

$$y[m,n] = \sum_{k=0}^{M_h-1} \sum_{\ell=0}^{N_h-1} h[k,\ell]\, x[m-k, n-\ell]$$

$h[k,\ell]$ is zero if $k<0$ or $k>M_h-1$, $\ell<0$ or $\ell<N_h-1$ $\quad|\quad$ $x[m-k,n-\ell]$ is zero if $m-k<0$ or $m-k>M_x-1$, $n-\ell<0$ $\quad n-\ell>N_x-1$

① $\boxed{0 \leq k \leq M_h-1}$ ③ $\boxed{0 \leq m-k \leq M_x-1}$

③ $\boxed{0 \leq \ell \leq N_h-1}$ ④ $\boxed{0 \leq n-\ell \leq N_x-1}$

using ①&③ : $0 \leq m \leq \underbrace{M_x+M_h-2}_{M_y-1} \rightarrow \boxed{M_y = M_x+M_h-1}$

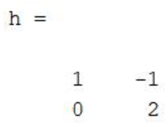
using ②&③ : $0 \leq \ell \leq \underbrace{N_x+N_h-2}_{N_y-1} \rightarrow \boxed{N_y = N_x+N_h-1}$

Fig.2. *Part 3 explanations*

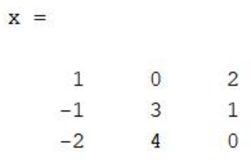
The function **DSLSI2D** given below calculates the 2D convolution of two matrices.

```
function [y]=DSLSI2D(h,x)
hs= size(h);
Mh=hs(1);
Nh=hs(2);
xs=size(x);
Mx=xs(1);
Nx=xs(2);
size_y= (xs+hs-1);
y=zeros(size_y(1),size_y(2));
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
```

For given h and x values:

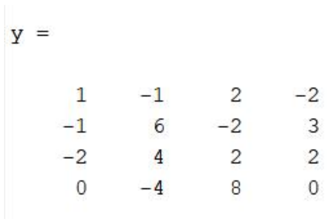


```
h =

     1    -1
     0     2
```



```
x =

     1     0     2
    -1     3     1
    -2     4     0
```

The function gives the result as:



```
y =

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

Part 3

Noise in an image corresponds to random color and brightness variations, it has a high frequency content therefore in order to eliminate the noise high frequencies should be eliminated as well. This can be achieved by a LPF which enables low frequencies to pass and high frequencies to cancel out.

The LPF impulse response can be represented as the multiplication of two sinc functions. The width of these two functions determines the bandwidth of the filter. Larger bandwidth means the canceling effect is less sensitive, the pass-band range increases hence some noise still continues to be present in the image. However for smaller bandwidths, the selectivity of the filter is so sensitive that even the slightest high frequency content of the image becomes eliminated. This also causes some elimination in the image’s own frequencies as well. The higher frequencies of the image are present on the edges hence elimination of these frequencies causes less defined edges, in other words blurry image.

In this lab the bandwidth is indicated by B. The following figure shows the results for different B values.

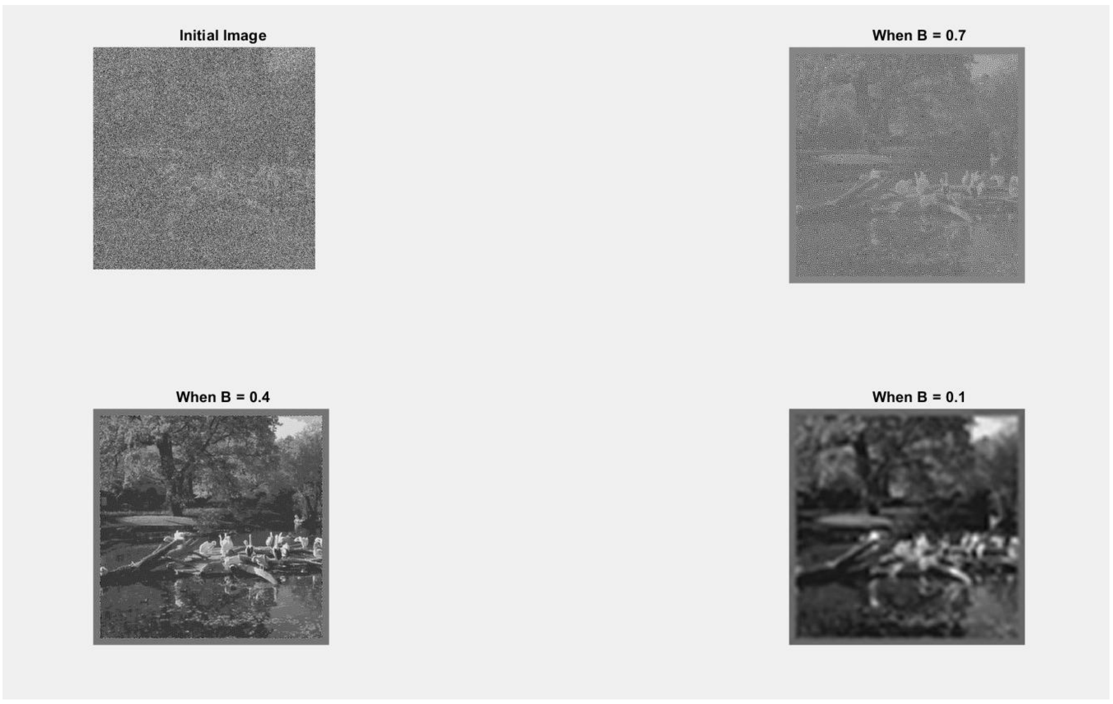


Fig.3. *Results of Image Denoising by LPF*

As can be seen from Fig.3. the highest B value is 0.7 which barely eliminates the noise in the initial image. The lowest B value is 0.1 which also eliminates the higher frequencies of the image itself causing a blurry results. The most suitable B value is hence 0.4.

```
D = 22102167;
D7 = rem(D,7); % D7 = 3
B = [0.7 0.4 0.1];
h1= zeros(30+D7);
h2= zeros(30+D7,30+D7);
h3= zeros(30+D7,30+D7);
for i = 1:length(B)
b = B(i);
for m = 1:Mh
for n = 1:Nh
if b == 0.7
h1(m,n) = sinc(b*(m-1-(29+D7)/2))*sinc(b*(n-1-(29+D7)/2));

elseif b == 0.4
h2(m,n) = sinc(b*(m-1-(29+D7)/2))*sinc(b*(n-1-(29+D7)/2));

elseif b == 0.1
h3(m,n) = sinc(b*(m-1-(29+D7)/2))*sinc(b*(n-1-(29+D7)/2));
end
end
end
end
x = ReadMyImage("Part4.bmp");
subplot(2,2,1);
hold on;
```

```
title("Initial Image");
DisplayMyImage(x);
y1 = DSLSI2D(h1,x);
subplot(2,2,2);
hold on;
DisplayMyImage(y1);
title("When B = 0.7");
y2 = DSLSI2D(h2,x);
subplot(2,2,3);
hold on;
DisplayMyImage(y2);
title("When B = 0.4");
y3 = DSLSI2D(h3,x);
subplot(2,2,4);
hold on;
DisplayMyImage(y3);
title("When B = 0.1");
```

Part 5 Edge Detection

The edges of an image corresponds to higher frequencies since the change in these spots are more drastic and sudden. By applying a HPF to an image we can select the higher frequencies and eliminate lower ones. This selection emphasizes edges due to the higher frequency allowance.
For vertical edge detection, the impulse response of the HPF is expressed as:

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

For horizontal edge detection, the impulse response of the HPF is expressed as:

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

For both horizontal and vertical edge detection we can use superposition and define the impulse response as:

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

The convolution results can be seen in the Figure below.

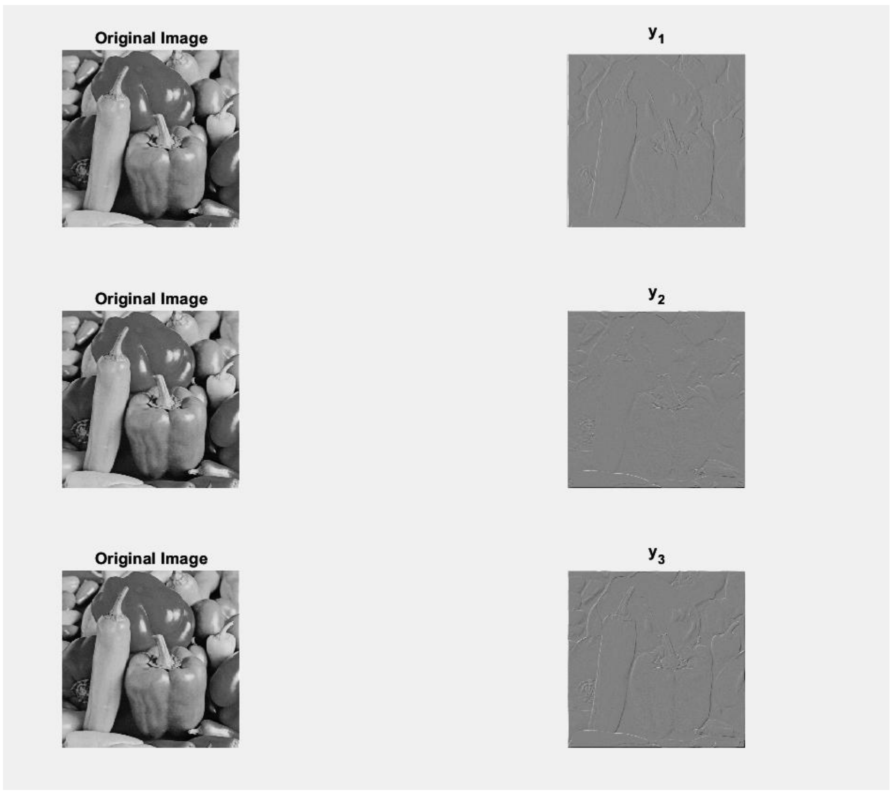


Fig.4. *Results of Edge Detection*

Fig.4 shows that the in the image y1, the vertical edges are detected. For y2, horizontal edges are detected and for y3 both edges are detected. Taking the square of the y values increases the edge difference in the images. Hence non-edge parts are very black and edges are white. The result of squaring process can be seen in Fig.5.

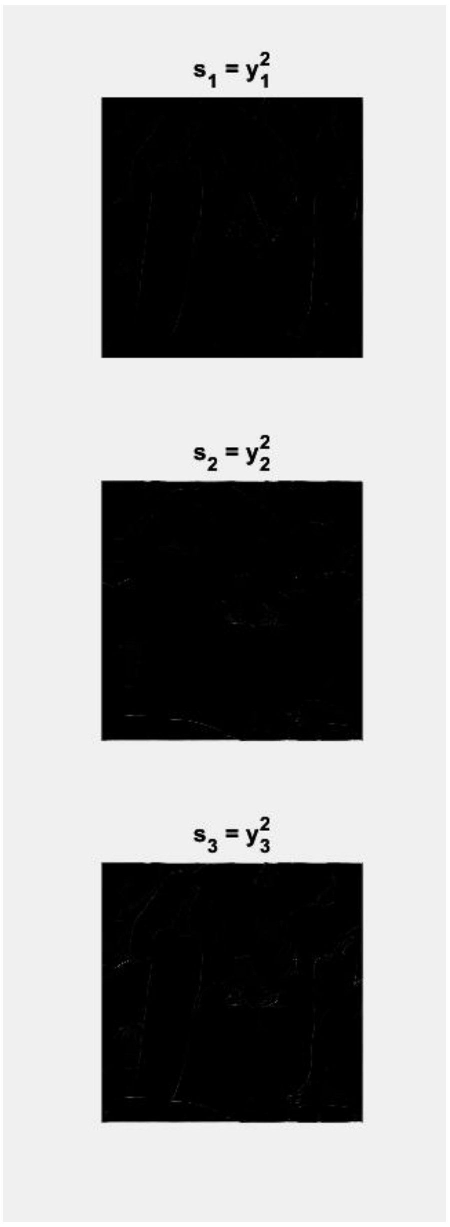


Fig.5. *Results of edge detection for higher powers*

Part 6 Pattern Recognition

A face recognition is tried to be achieved for this part. In Fig.5. the used image and the kernel can be seen.

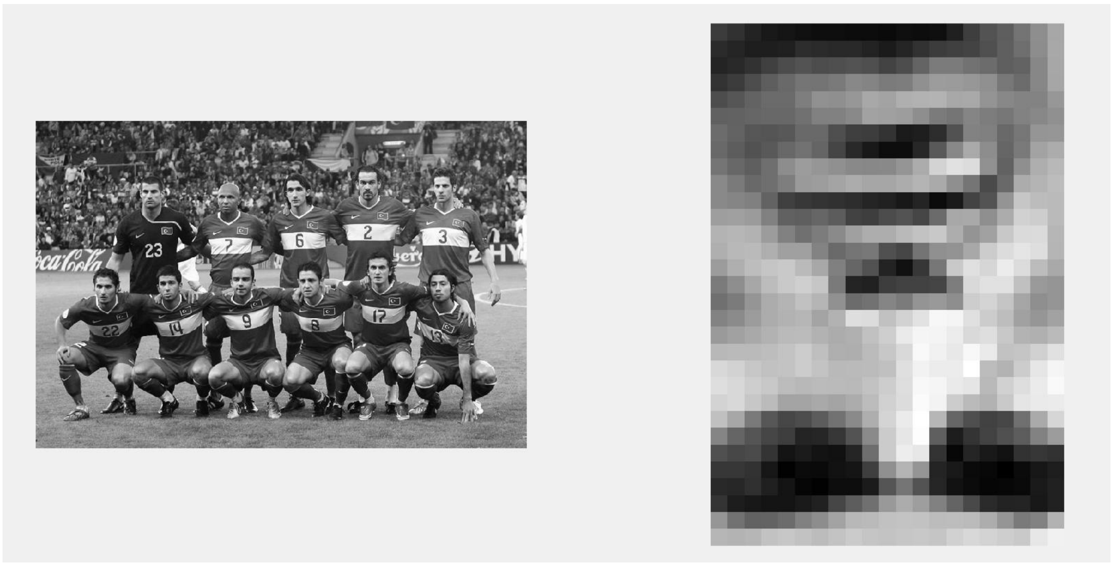


Fig.5. *Used image and the impulse response*

Doing the convolution the results can be seen below:

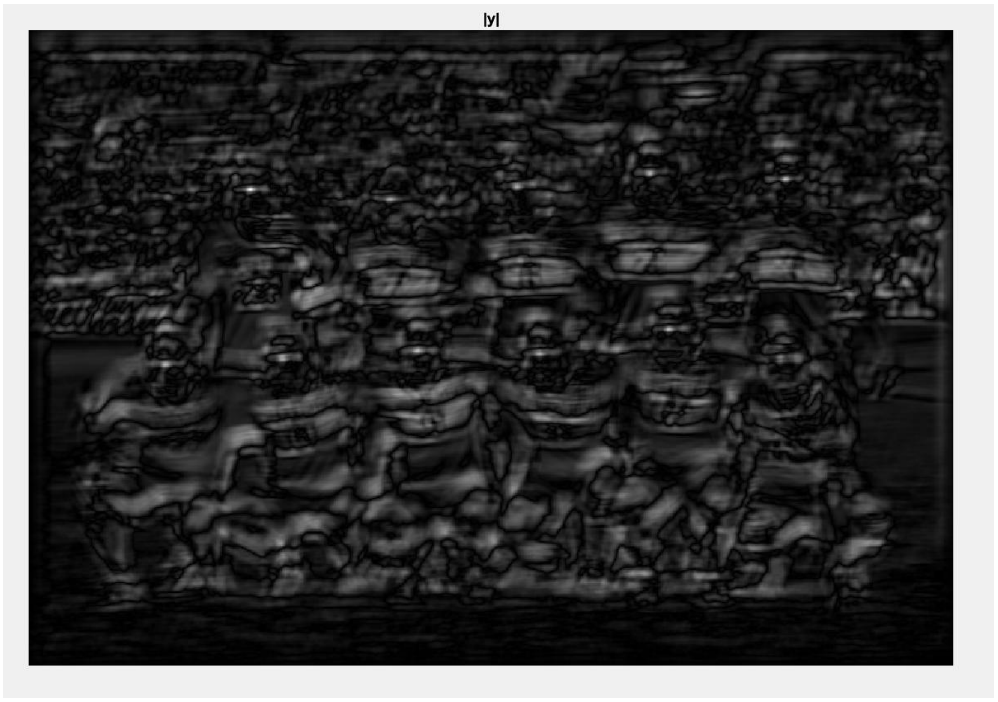


Fig.6. *The output image for |y|*

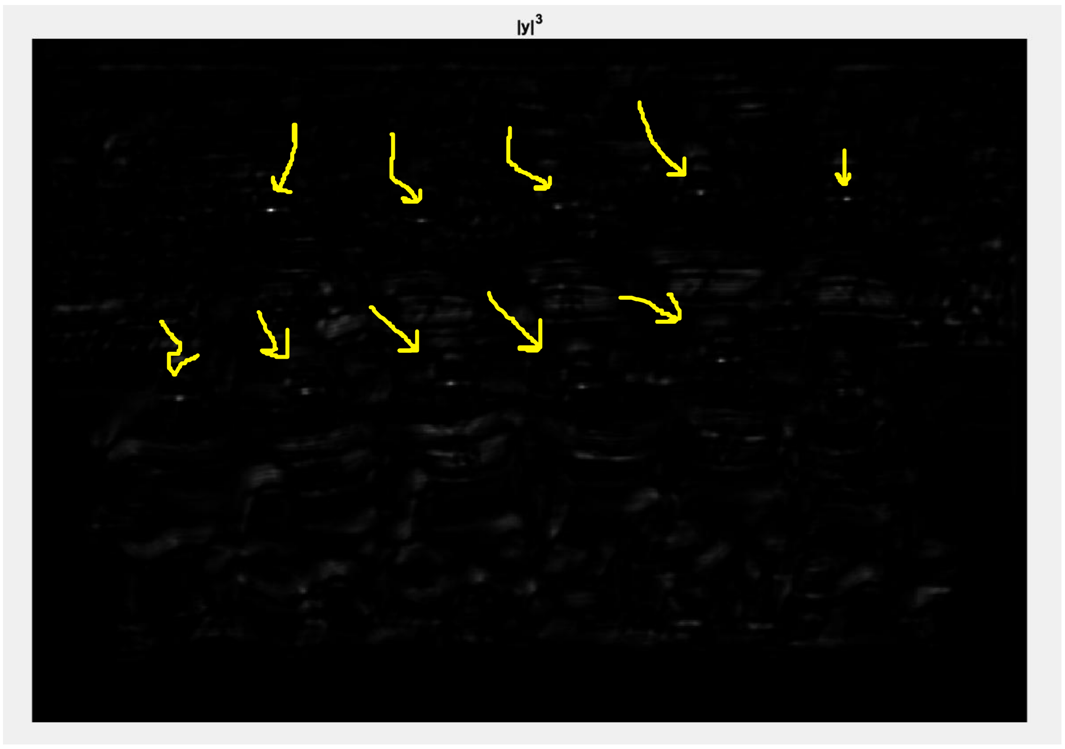


Fig.7. *The output image for* $|y|^3$

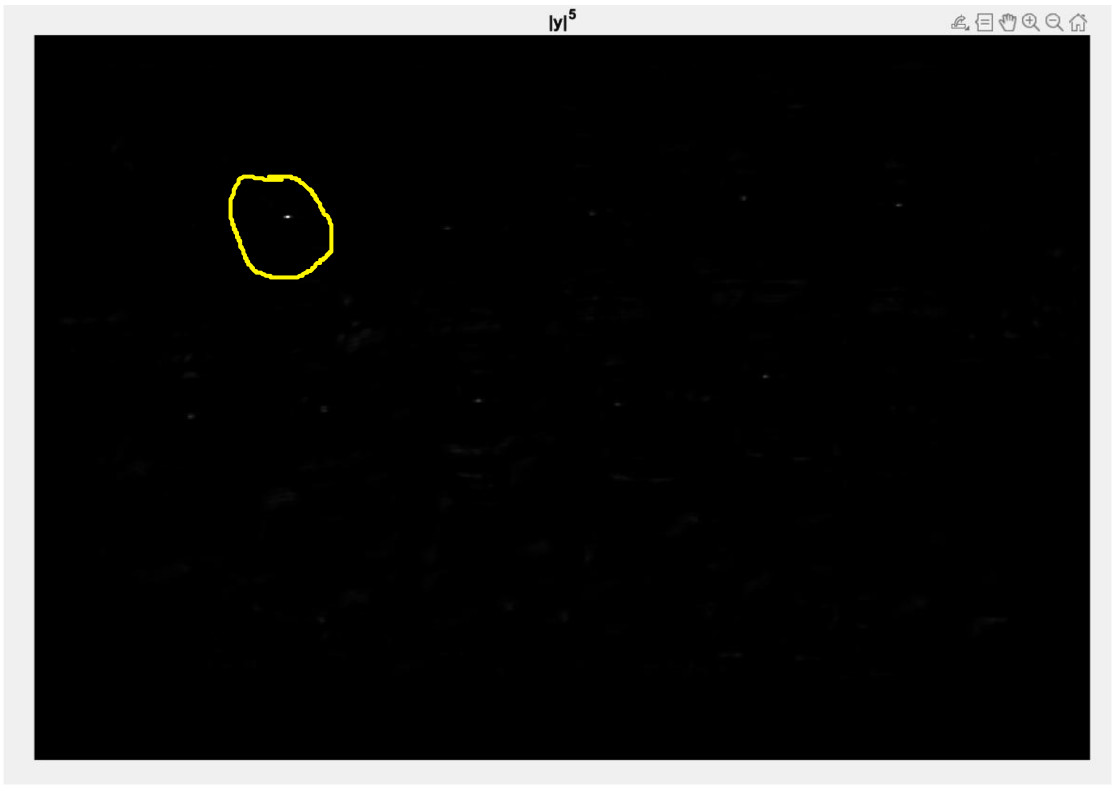


Fig.8. *The output image for* $|y|^5$

In Fig. 6. although there are bright points located at the faces, there are other white points as well which does not show exactly the searched face. As we increase the power of $|y|$, the bright points become brighter and darker points become more darker. This increases the shade difference hence the bright pixels located at faces become more detectable. Fig.7. is obtained by taking the 3rd power of $|y|$ and as can be seen the faces are more obvious nevertheless this is still not enough since there are white points for all faces although we are searching for a specific one. Therefore if the $|y|$ power is increased more and more, the face detection becomes more sensitive. For instance in Fig.8. the brightest point is clearly located at the earlier selected face for the impulse response. Hence for selecting the faces in general $|y|^3$ can be used but for detecting a specific face or object higher powers should be used in this case $|y|^5$.

Appendix

Part 5

```
h1 = [0.5 -0.5]; % v
y1 = DSLSI2D(h1,x);
s1 = y1.^2;
subplot(2,3,1);
DisplayMyImage(y1);
title("y_1");
subplot(2,3,2);
DisplayMyImage(s1);
title("s_1 = y_1^2");

h2 = [0.5;-0.5]; % h
y2 = DSLSI2D(h2,x);
s2 = y2.^2;
subplot(2,3,3);
DisplayMyImage(y2);
title("y_2");
subplot(2,3,4);
DisplayMyImage(s2);
title("s_2 = y_2^2");

h3 = 0.5*(h1+h2);
y3 = DSLSI2D(h3,x);
s3 = y3.^2;
subplot(2,3,5);
DisplayMyImage(y3);
title("y_3");
subplot(2,3,6);
DisplayMyImage(s3);
title("s_3 = y_3^2");
```

Part 6

```
%%

x = ReadMyImage("Part6x.bmp");
tiledlayout(1,2)
nexttile
DisplayMyImage(x);

h = ReadMyImage("Part6h.bmp");
nexttile
DisplayMyImage(h);

%%
y = DSLSI2D(h,x);
figure(1)

DisplayMyImage(abs(y));
title("|y|");
%%
figure(2)

DisplayMyImage(abs(y).^3);
```

```
title("|y|^3");
%%
figure(3)

DisplayMyImage(abs(y).^5);
title("|y|^5");



function [y]=DSLSI2D(h,x)
hs= size(h);
Mh=hs(1);
Nh=hs(2);
xs=size(x);
Mx=xs(1);
Nx=xs(2);
size_y= (xs+hs-1);
y=zeros(size_y(1),size_y(2));
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
```