EEE 321 LAB4 OFFLAB ASSIGMENT REPORT

Name/Surname: Muhammet Melih Çelik ID:22003836

PART 1: Displaying Images

The functions for the assignment was implemented successfully, the image 'Part5.bmp' was seen in the MATLAB.

PART 2: Basics for 2D Signals and Systems

The convolution formula in 2D was derived in an analytic manner, by starting the definition of 2D discrete impulse function, which is given in the assignment:

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

For 2D convolution formula, the system assumed as an LTI system so that, time shift of the 2D impulse generates time shift on the impulse response as:

$$\delta[m-k,n] \rightarrow LTI\ System\ \rightarrow h[m-k,n]$$

or

$$\delta[m,n-k] \rightarrow LTI\ System\ \rightarrow h[m,n-k]$$

Also, as the system is an LTI system, the inputs above can be superposed which yields the superposed output:

$$\{\delta[m-k,n] + \delta[m,n-k]\} \rightarrow LTI\ System \rightarrow \{h[m-k,n] + h[m,n-k]\}$$

So, the 2D input x[m, n] can be sampled with the 2D impulse and given to the system, which would yield the sampled output y[m, n] as below:

$$x[k,n]\delta[m-k,n] \rightarrow LTI\ System \rightarrow x[k,n]h[m-k,n]$$

Hence, for one dimension by summing both sides of the system input-output from minus infinity to the plus infinity, we can achieve 1D convolution sum formula:

$$\sum_{k=-\infty}^{+\infty} x[k,n]\delta[m-k,n] \rightarrow LTI\ System \rightarrow \sum_{k=-\infty}^{+\infty} x[k,n]h[m-k,n]$$

Therefore, for just one dimension the convolution sum was realized as below:

$$y[m,n] = \sum_{k=-\infty}^{+\infty} x[k,n]h[m-k,n]\text{, for some particular } n \in \Re$$

Hence, in 2D convolution sum, by summing second times, complete formula below was derived:

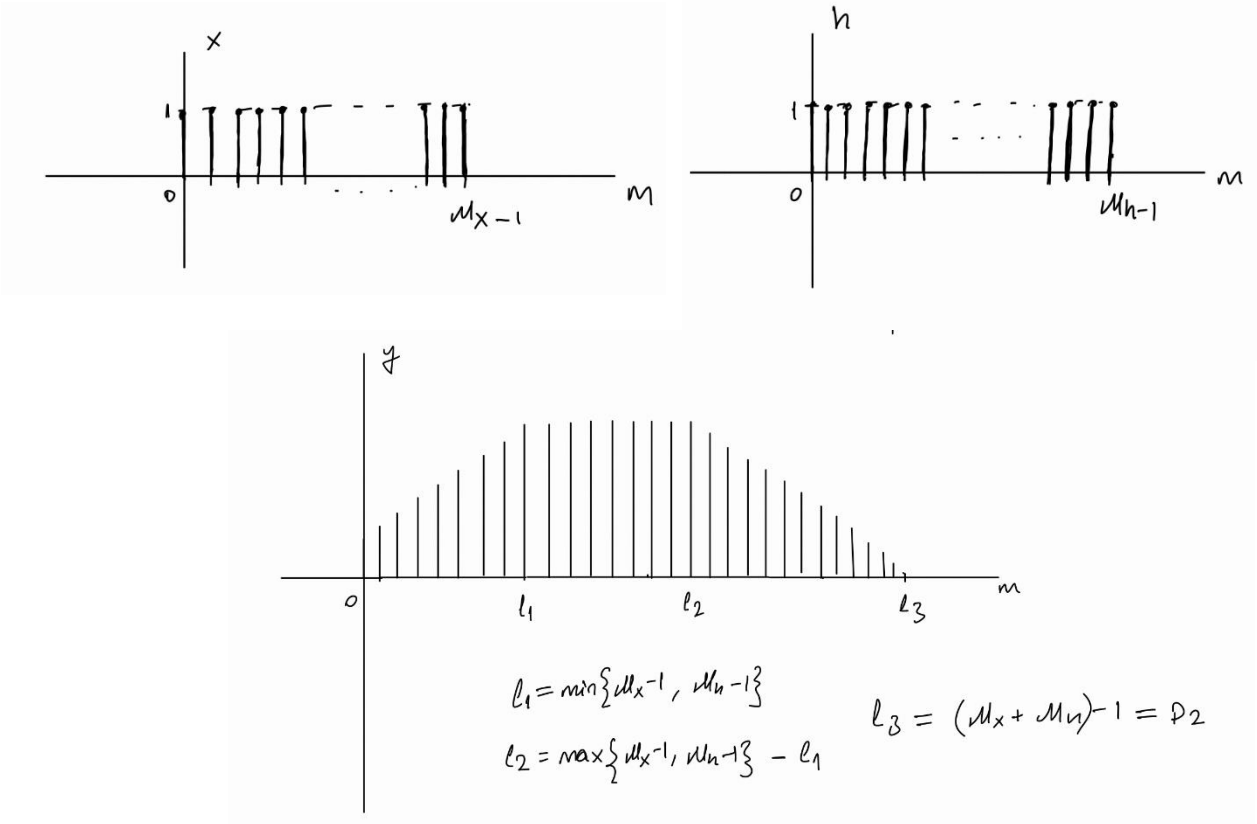
$$y[m,n] = \sum_{l=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} x[k,l]h[m-k,n-l]$$

PART 3: 2D FIR Filter

The values for D1, D2, D3, D4 can be found by defining signals x and h which is 1 for all m and n values that are in the boundaries of $M_{h,x-1}$ and $N_{h,x-1}$. The purpose here to find the boundaries of the output function y which is nonzero. The illustration below was implemented for just one dimension as it is same in the other dimension as well.

$$x[m,n] = u[m,n] - u[m-(M_x-1),n] \text{ for some } n \in \Re$$

$$h[m,n] = u[m,n] - u[m-(M_h-1),n] \text{ for some } n \in \Re$$



It is clear that the D1, D2, D3 and D4 values now can be determined simply. As the system is causal both D1 and D3 are zero. D2 and D4 values are sum of boundaries subtracted one. All of them given clearly in the next page.

D1	0
D2	$(M_x + M_h) - 1$
D3	0
D4	$(N_x + N_h) - 1$

Table 1: Boundary Coefficients

The MATLAB code for the 2D convolution sum function was given below:

```
%2D convolution function

function [y] = DSLSI2D(x,h)

    x_row = size(x,1); %M_x
    x_col = size(x,2); %N_x

    h_row = size(h,1); %M_h
    h_col = size(h,2); %N_h

    y_row = (x_row + h_row)-1;
    y_col = (x_col + h_col)-1;

    h_rot = rot90(h,2);

    y = zeros(y_row,y_col);

    temp_mat1 = zeros((x_row + 2*h_row),(x_col + 2*h_col));
    temp_mat1((h_row+1):(x_row+h_row),(h_col+1):(x_col + h_col)) = x;

    for l = 0:(x_row + 2*h_row)-2 %column iteration
        for k  = 0:(x_col + 2*h_col)-2 %row iteration

            temp_mat2 = zeros((x_row + 2*h_row),(x_col + 2*h_col));
            temp_mat2(1+k:h_row+k,1+l:h_col+l) = h_rot;
            mul_mat = (temp_mat2.*temp_mat1);
            total  = sum(mul_mat(:));

            if (k + 1 < x_row + 2*h_row - 1) && ...
               (l + 1 < x_row + 2*h_row - 1)
                y(k+1,l+1) = total;
            end

        end
    end

    y = y(2:end,2:end);

end
```

The result of the example in the assignment matches the output in the example as below:

```
%%
%PART3:2D FIR Filter
x = [1 0 2;
    -1 3 1;
    -2 4 0];

h = [1 -1;
     0  2];

y = DSLSI2D(x,h);
disp(y);
```

Command Window

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

fx >>
```

Figure 1: Example Output

PART 4: Image Denoising

For the impulse response the function below was written:

```
function [h] = impulseResponse(M_h,N_h,B)


    h = zeros(M_h,N_h);

    for k = 0:M_h-1
        for l = 0:N_h-1
            h(k+1,l+1) = sinc(B*(k-0.5*(M_h-1))).*sinc(B*(l-0.5*(M_h-1)));
        end
    end

end
```

The plots of the picture with different B coefficient values were given below:

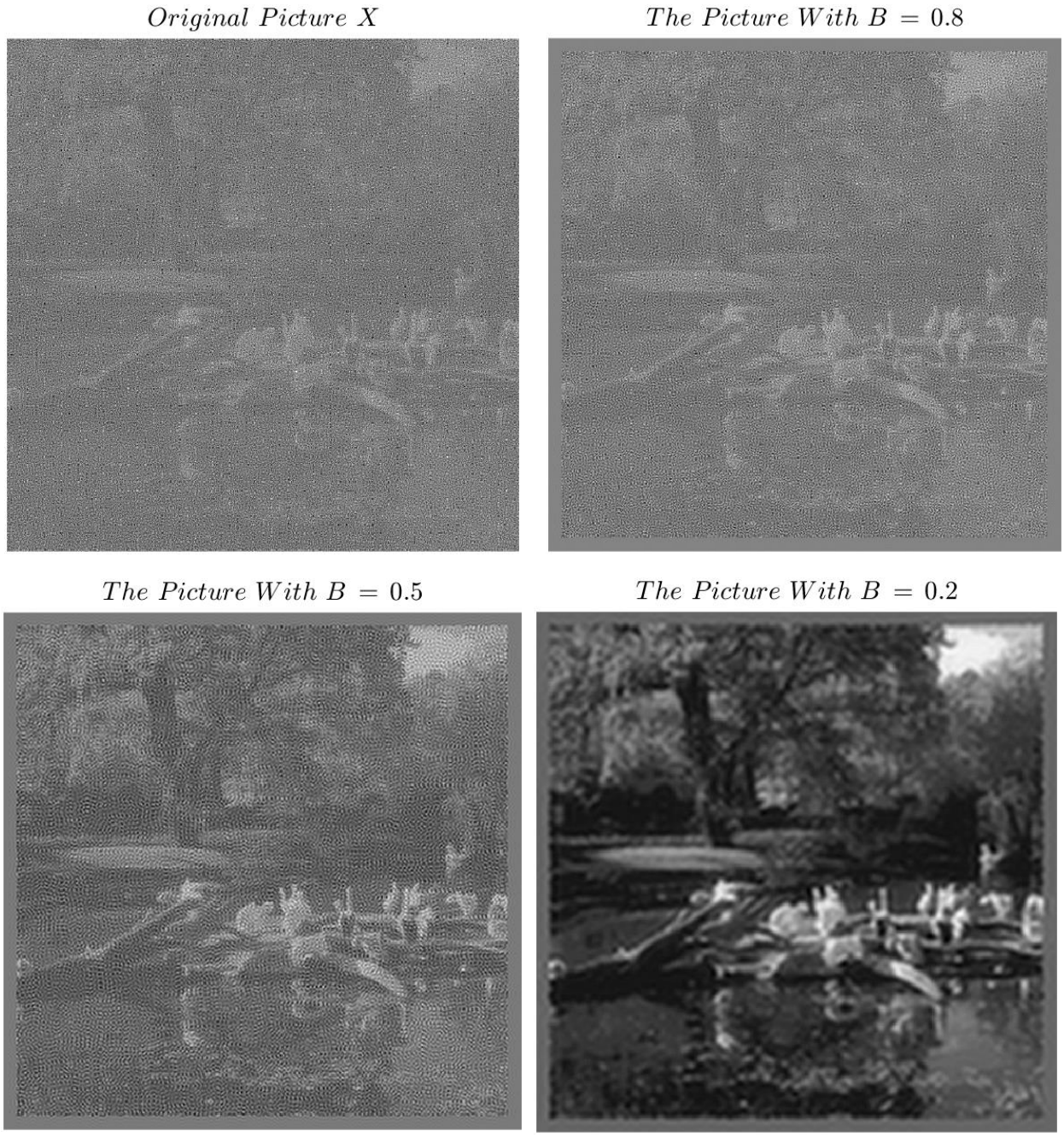


Figure 2: Filtered Images

It can be said that the value of 0.5 for the B coefficient is the most appropriate one as while this value remove the noise considerably, it does not degrade the image; whereas, the value of 0.2 does.

PART 5: Edge Detection

With the new impulse response, the edge detection was conducted on the image ‘Part5.bmp’ was given below:

Original Image

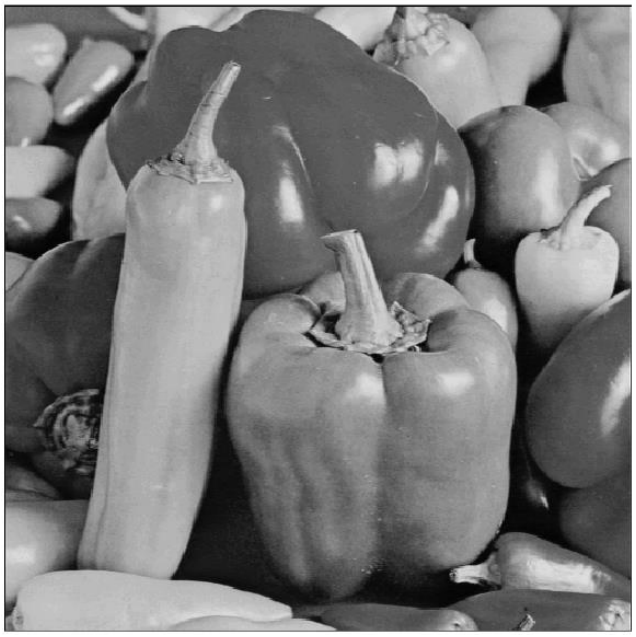


Figure 3

Vertical Edge Detection

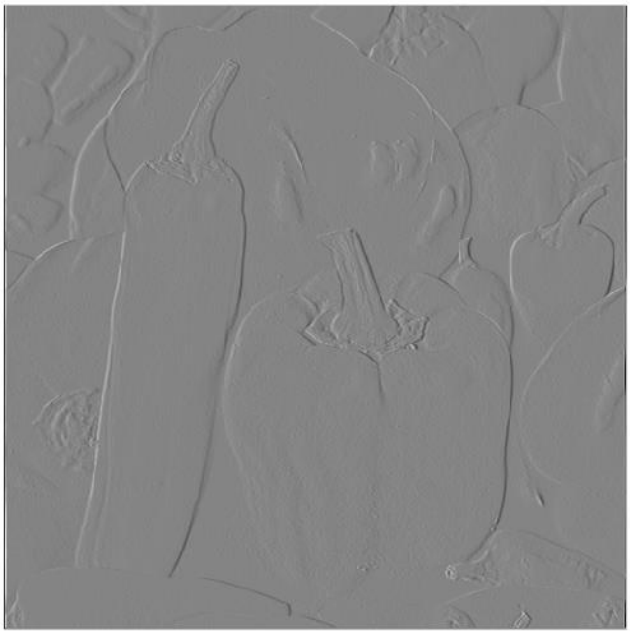


Figure 4

Image S_1

Figure 5

Even though, it seems faint; the edges of the image ‘Part5.bmp’ were highlighted. This can be seen in the Figure 5 on just above.

With the second impulse response the results were obtained as below:

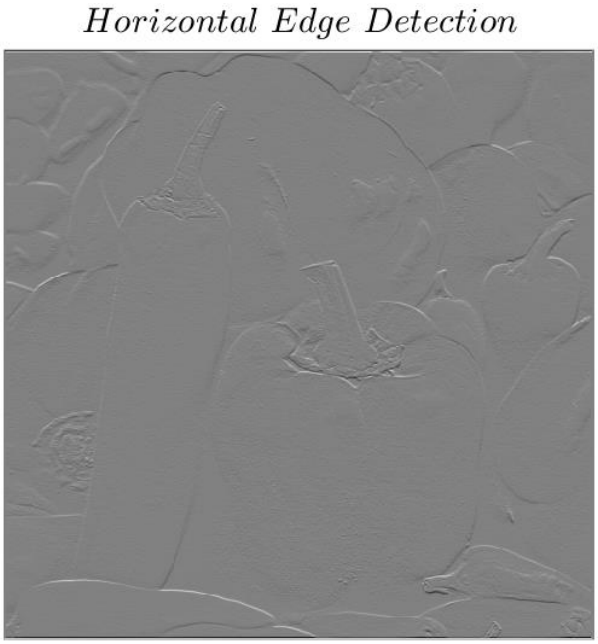


Figure 6

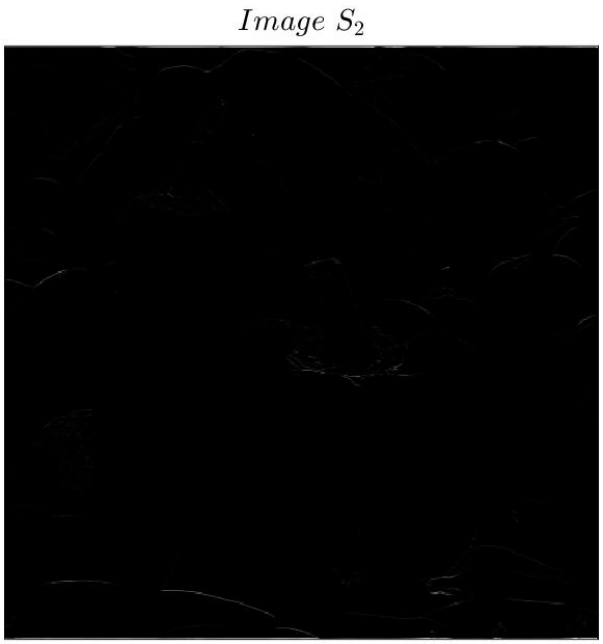


Figure 7

While in the previous part, most of the edges were vertical; here, it can be seen that the edges are horizontal.

Finally, with the last impulse response, following results were observed:

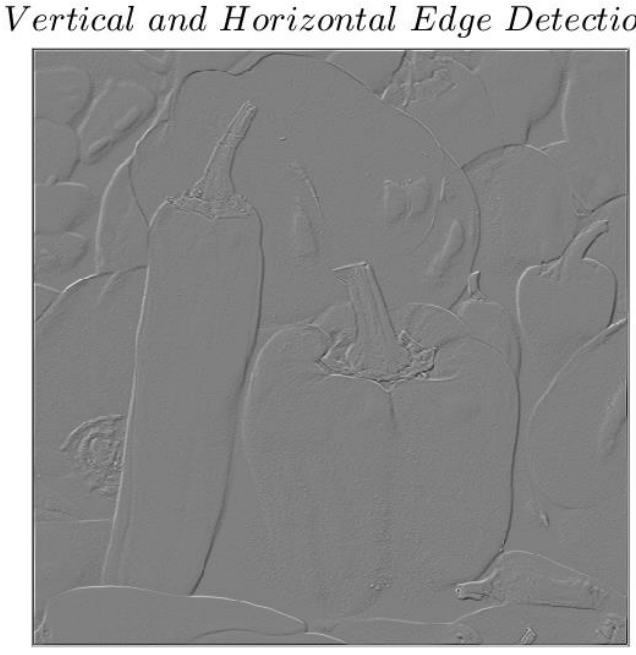


Figure 8

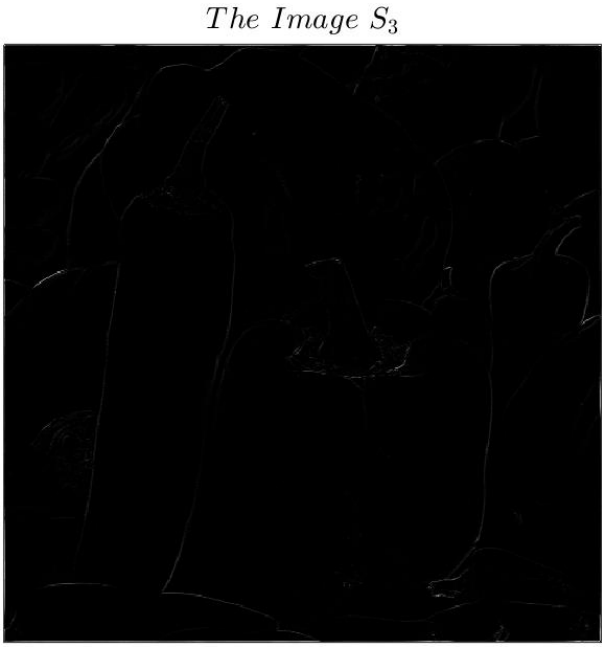


Figure 9

The results observed indicates all edges includes both vertical and horizontal features are present in the plots.

EEE 321 LAB4 ONLAB ASSIGMENT REPORT

Name/Surname: Muhammet Melih Çelik ID:22003836

The original picture and the picture that is the response of the system were given below. The original picture was defined as an input to the system and the reverse picture was defined as the system's response. Hence, the output was obtained by the 2D convolution.

Figure 10: The Input: Image x

Figure 11: The System's Response h

Output Of The System

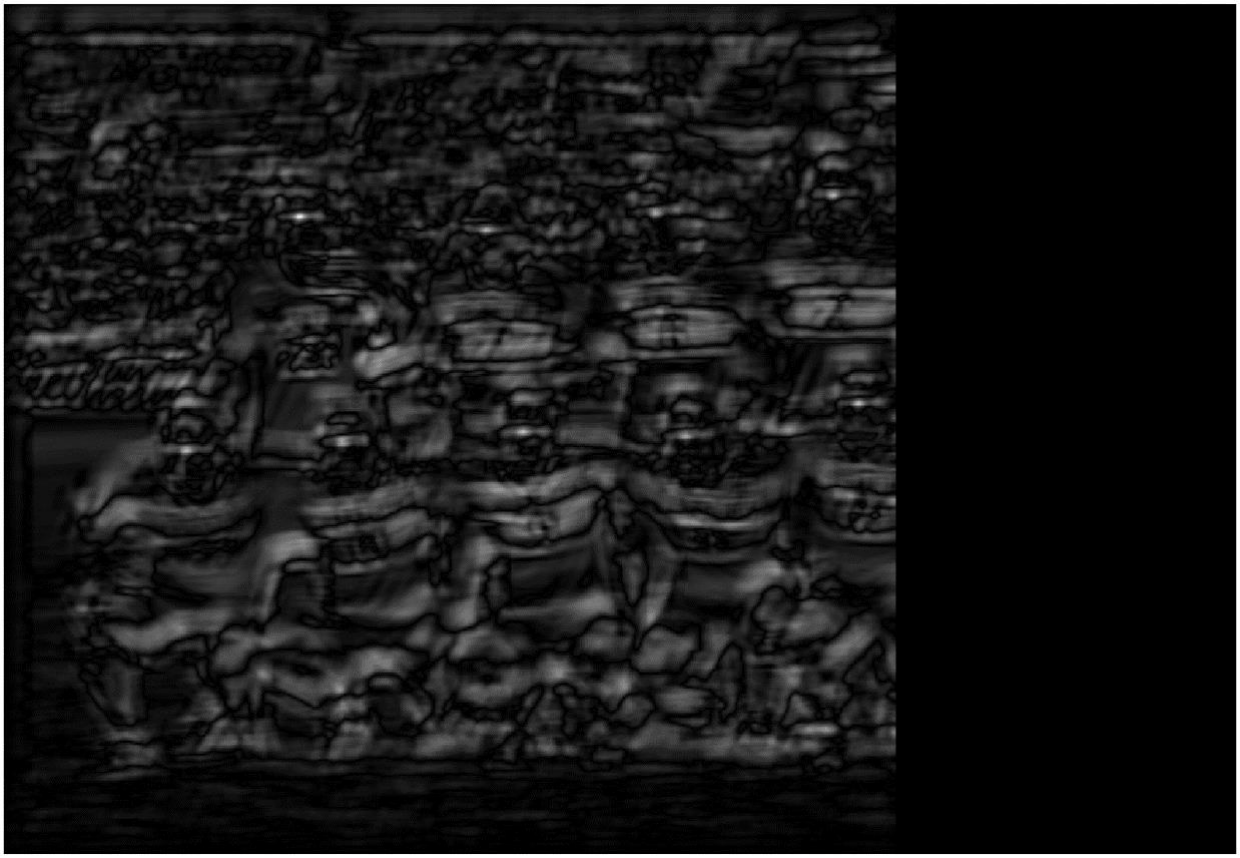


Figure 12

The bright points occur where there is a player's face present. Also, there is brightness where there is not face, but this brightness is not condensed in a single point as it is on the places where there is a face. In the below, also the fourth and sixth power of the output signal can be seen.

$Fourth\ Power\ Of\ The\ Output$

Figure 13

$Sixth\ Power\ Of\ The\ Output$

Figure 14

```
%2D convolution function
function [y] = DSLSI2D(x,h)

    x_row = size(x,1); %M_x
    x_col = size(x,2); %N_x

    h_row = size(h,1); %M_h
    h_col = size(h,2); %N_h

    y_row = (x_row + h_row)-1;
    y_col = (x_col + h_col)-1;

    h_rot = rot90(h,2);

    y = zeros(y_row,y_col);

    %the matrices on which the loop iterates.
    temp_mat1 = zeros((x_row + 2*h_row),(x_col + 2*h_col));
    temp_mat1((h_row+1):(x_row+h_row),(h_col+1):(x_col + h_col)) = x;

    for l = 0:(x_row + 2*h_row)-h_col %column iteration
        for k  = 0:(x_col + 2*h_col)-h_row %row iteration

            temp_mat2 = zeros((x_row + 2*h_row),(x_col + 2*h_col));
            temp_mat2(1+k:h_row+k,1+l:h_col+l) = h_rot;
            mul_mat = (temp_mat2.*temp_mat1);
            total  = sum(sum(mul_mat));

            if (k + 1 < x_row + 2*h_row - 1) && ...
               (l + 1 < x_row + 2*h_row - 1)
                y(k+1,l+1) = total;
            end

        end
    end

    y = y(2:end,2:end);

end
```

Published with MATLAB® R2023b

```
function [h] = impulseResponse(M_h,N_h,B)

    h = zeros(M_h,N_h);

    for k = 0:M_h-1
        for l = 0:N_h-1
            h(k+1,l+1) = sinc(B*(k-0.5*(M_h-1))).*sinc(B*(l-0.5*(M_h-1)));
        end
    end

end
```

Published with MATLAB® R2023b

```
%----------------------EE3321 LAB4 OFFLAB CODE----------------------------
%PART1:Displaying Images
A = ReadMyImage('Part5.bmp');
DisplayMyImage(A);

%PART3:2D FIR Filter
x = [1 0 2;
    -1 3 1;
    -2 4 0];

h = [1 -1;
     0  2];

y = DSLSI2D(x,h);
disp(y);

%PART4:Image Denoising
x = ReadMyImage('Part4.bmp');
ID = 22003836;
D_8 = mod(ID,8);

B1 = 0.8;
B2 = 0.5;
B3 = 0.2;

M_h = 25 + D_8;
N_h = 25 + D_8;

h1 = impulseResponse(M_h,N_h,B1);
y1 = DSLSI2D(x,h1);

h2 = impulseResponse(M_h,N_h,B2);
y2 = DSLSI2D(x,h2);

h3 = impulseResponse(M_h,N_h,B3);
y3 = DSLSI2D(x,h3);

DisplayMyImage(x);
title ('$Original\,\,Picture\,\,X$','Interpreter','latex','FontSize',14);

DisplayMyImage(y1);
title ('$The\,\,Picture\,\,With\,\,B\,=\,0.8$',...
       'Interpreter','latex','FontSize',14);

DisplayMyImage(y2);
title ('$The\,\,Picture\,\,With\,\,B\,=\,0.5$',...
       'Interpreter','latex','FontSize',14);

DisplayMyImage(y3);
title ('$The\,\,Picture\,\,With\,\,B\,=\,0.2$',...
       'Interpreter','latex','FontSize',14);

%PART 5:Edge Detection
x = ReadMyImage('Part5.bmp');
```

```
DisplayMyImage(x);
title('$Original\,\,Image$','Interpreter','latex','FontSize',14);

h1 = [0.5 -0.5;
     0     0];
y1 = DSLSI2D(x,h1);
DisplayMyImage(y1);
title('$Edge\,\,Detected\,\,Image$','Interpreter','latex','Fontsize',14);

s1 = y1.*y1;
DisplayMyImage(s1);
title('$Image\,\,S_1$','Interpreter','latex','Fontsize',14);

h2  = [0.5 0;
      -0.5 0];
y2 = DSLSI2D(x,h2);
DisplayMyImage(y2);
title('$Edge\,\,Detected\,\,Image$','Interpreter','latex','Fontsize',14);

s2 = y2.*y2;
DisplayMyImage(s2);
title('$Image\,\,S_2$','Interpreter','latex','Fontsize',14);

h3 = 0.25*h1 + 0.25*h2;
y3 = DSLSI2D(x,h3);
DisplayMyImage(y3);
title('$Edge\,\,Detected\,\,Image$','Interpreter','latex','Fontsize',14);

s3 = y3.*y3;
DisplayMyImage(y3);
title('$The\,\,Image\,\,S_3$','Interpreter','latex','Fontsize',14);
```

Published with MATLAB® R2023b

```
%-----------------EEE321 LAB4 ONLAB CODE----------------------------------
x = ReadMyImage('Part6x.bmp'); %soccer team image.
DisplayMyImage(x);

h = ReadMyImage('Part6h.bmp'); %impulse response.
DisplayMyImage(h);

y = DSLSI2D(x,h);
DisplayMyImage(abs(y));
title('$Output\,\,Of\,\,The\,\,System$',...
       'Interpreter','latex','FontSize',14);

y1 = abs(y).^4;
DisplayMyImage(y1);
title('$Fourth\,\,Power\,\,Of\,\,The\,\,Output$',...
       'Interpreter','latex','FontSize',14);

y2 = abs(y).^6;
DisplayMyImage(y2);
title('$Sixth\,\,Power\,\,Of\,\,The\,\,Output$',...
       'Interpreter','latex','FontSize',14);
```

Published with MATLAB® R2023b