PART 2

1) We are given the two-dimensional discrete impulse signal as:

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

2) Let's write the input signal x[m, n] as a superposition of shifted impulse signal as:

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

where we interpret x[k, l] as the coefficient of the impulse shifted by k, l units, which is $\delta[\mathrm{m}-\mathrm{k},\mathrm{n}-\mathrm{l}]$.

3) Let's call the response that the system gives to $\delta[\mathrm{m},\mathrm{n}]$ as h[m, n]. Using the space invariance property of the system, we recognize that the response of the system to $\delta[\mathrm{m}-\mathrm{k},\mathrm{n}-\mathrm{l}]$ should be h[m - k, n - l]. Now, we can use the linearity property to write the input – output relationship, where y[m, n] denotes the output:

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

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

PART 3

1) We are given that x[m, n] can only be nonzero within $0 \le m \le M_x - 1$. It is also established that h[m, n] can only be nonzero within $0 \le m \le M_h - 1$. In Eq. 5, we have h[k, l], and so this term can only be nonzero within $0 \le k \le M_h - 1$ (a). In this equation, we also have x[m – k, n – l], which can only be nonzero within $0 \le m - k \le M_x - 1$ (b).

Using (a), we get the interval for k: $[0, M_h - 1]$.

Using (b), we get: $k \le m$ and $m - M_x + 1 \le k$ => $m - M_x + 1 \le k \le m$, which can be represented as $[m - M_x + 1, m]$.

If we combine these two intervals, we can say that $m \ge 0$ and $M_h - 1 \ge m - M_x + 1$ so k can satisfy these inequalities. Hence, we have $0 \le m \le M_h + M_x - 2$. We are required to show that y[m, n] can only be nonzero within $0 \le m \le M_y - 1$. Using the inequality we found, we can say that $M_y - 1 = M_h + M_x - 2$ => $M_y = M_h + M_x - 1$.

2) We are given that x[m, n] can only be nonzero within $0 \leq n \leq N_x - 1$. It is also established that h[m, n] can only be nonzero within $0 \leq n \leq N_h - 1$. In Eq. 5, we have h[k, l], and so this term can only be nonzero within $0 \leq l \leq N_h - 1$ (a). In this equation, we also have x[m – k, n – l], which can only be nonzero within $0 \leq n - l \leq N_x - 1$ (b).

Using (a), we get the interval for l: $[0, N_h - 1]$.

Using (b), we get: $l \leq n$ and $n - N_x + 1 \leq l \Rightarrow n - N_x + 1 \leq l \leq n$, which can be represented as $[n - N_x + 1, n]$.

If we combine these two intervals, we can say that $n \geq 0$ and $N_h - 1 \geq n - N_x + 1$ so l can satisfy these inequalities. Hence, we have $0 \leq n \leq N_h + N_x - 2$. We are required to show that y[m, n] can only be nonzero within $0 \leq n \leq N_y - 1$. Using the inequality we found, we can say that $N_y - 1 = N_h + N_x - 2 \Rightarrow N_y = N_h + N_x - 1$.

MATLAB code:

```
x = [2 1 1; -3 0 2; 1 -1 2];
h = [2 1; -1 0];

disp( DSLSI2D(h, x));

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
```

When I supplied the given inputs to the function, the correct matrix was displayed in the console:

```
>> MA2P1
     4     4     3     1
    -8    -4     3     2
     5    -1     1     2
    -1     1    -2     0
```

PART 4

Code for adding noise:

```
x = ReadMyImage("Part4.bmp");
mean = 0;
std1 = 0.1;
std2 = 0.25;
std3 = 0.5;
s = size(x);
image1 = x + random('norm', mean, std1, s);
image2 = x + random('norm', mean, std2, s);
image3 = x + random('norm', mean, std3, s);

DisplayMyImage(image1);
DisplayMyImage(image2);
DisplayMyImage(image3);
```

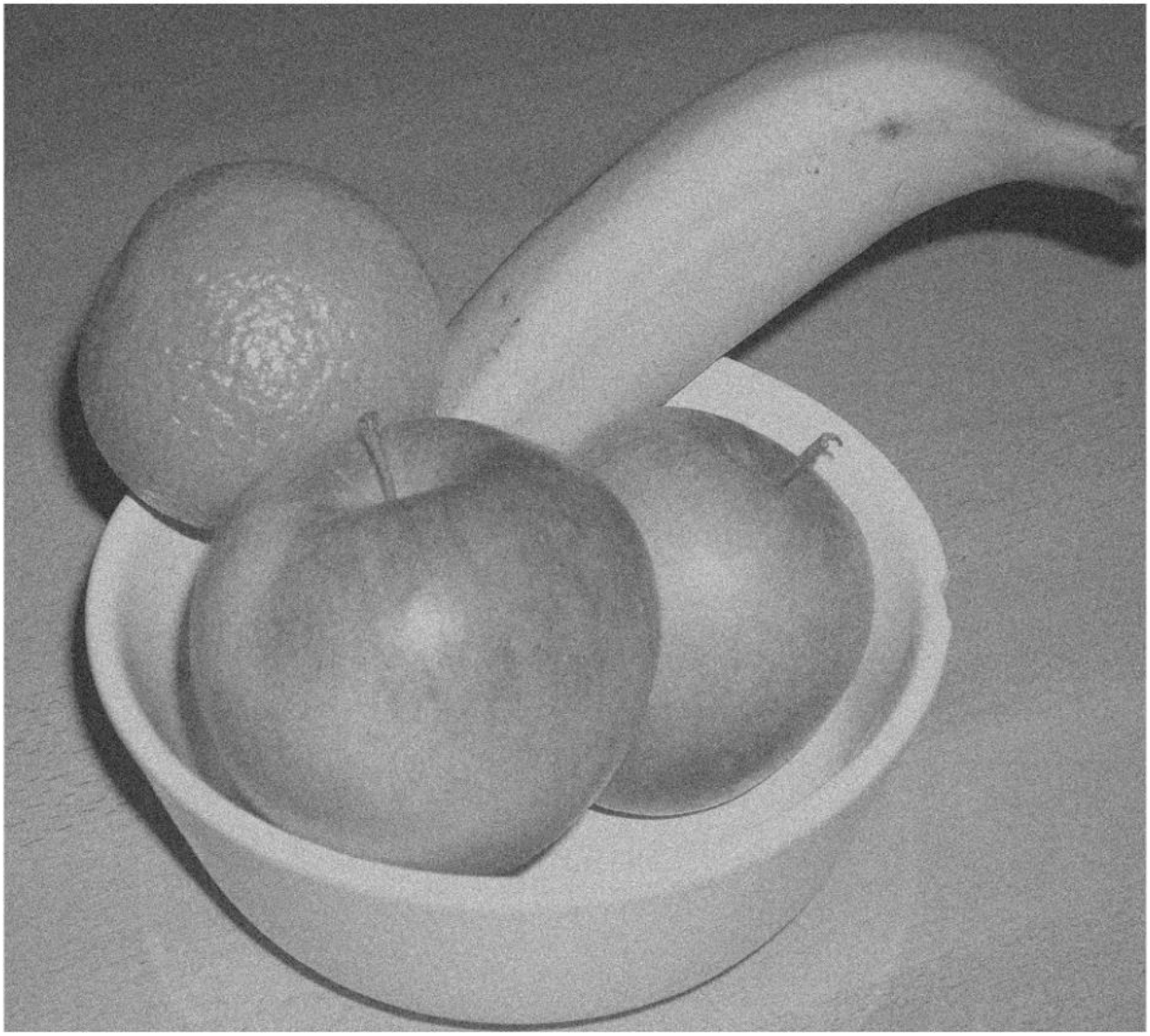

Figure 1: Photo with noise (mean = 0, std = 0.1)

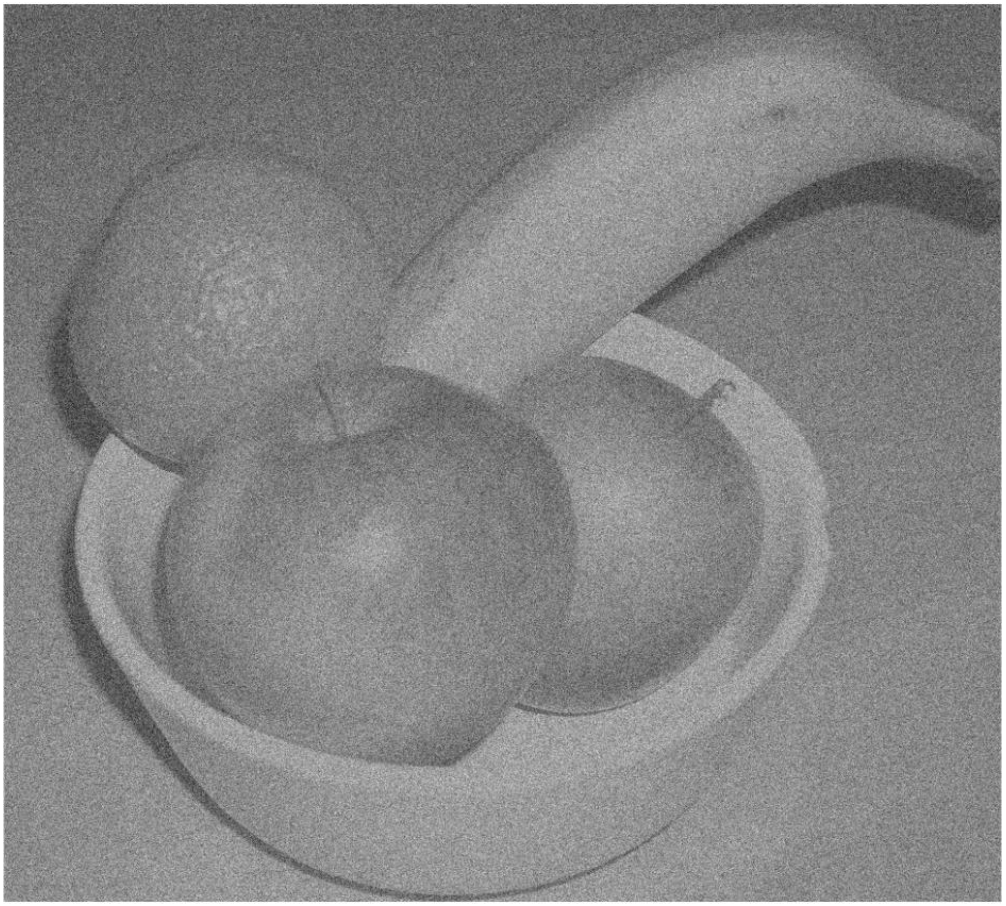

Figure 2: Photo with noise (mean = 0, std = 0.25)

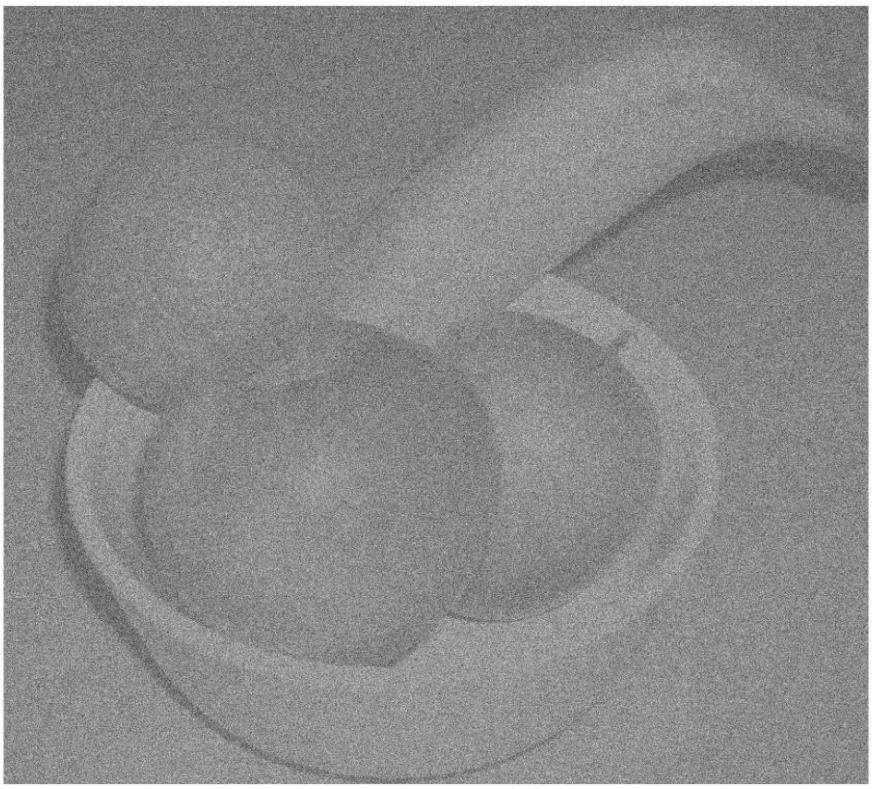

Figure 3: Photo with noise (mean = 0, std = 0.5)

From Figures 1, 2 and 3, we can observe that as std increases, the image gets more and more distorted. We can deduce that the std value for adding noise is directly proportional to how noisy the image gets, as std goes up the noisiness also does.

```
% code for preparing h
D = 21901548;
D17 = rem(D, 17);
Mh = 20 + D17;
Nh = 20 + D17;
B1 = 0.5;
B2 = 0.2;
B3 = 0.05;
h1 = zeros(Mh, Nh);
for m = 1 : Mh
    for n = 1 : Nh
        h1(m, n) = sinc(B1 * (m - (Mh - 1) / 2)) .* sinc(B1 * (n - (Nh - 1)
/ 2)); % line that prepares the matrix h
    end
end

h2 = zeros(Mh, Nh);
for m = 1 : Mh
    for n = 1 : Nh
        h2(m, n) = sinc(B2 * (m - (Mh - 1) / 2)) .* sinc(B2 * (n - (Nh - 1)
/ 2)); % line that prepares the matrix h
    end
end

h3 = zeros(Mh, Nh);
for m = 1 : Mh
    for n = 1 : Nh
        h3(m, n) = sinc(B3 * (m - (Mh - 1) / 2)) .* sinc(B3 * (n - (Nh - 1)
/ 2)); % line that prepares the matrix h
    end
end

% code for displaying processed images
out1 = DSLSI2D(h1, image3);
subplot(1,3,1);
imshow(out1, []);
title("Image denoised with B = " + string(B1));

out2 = DSLSI2D(h2, image3);
subplot(1,3,2);
imshow(out2, []);
title("Image denoised with B = " + string(B2));

out3 = DSLSI2D(h3, image3);
subplot(1,3,3);
imshow(out3, []);
title("Image denoised with B = " + string(B3));
```

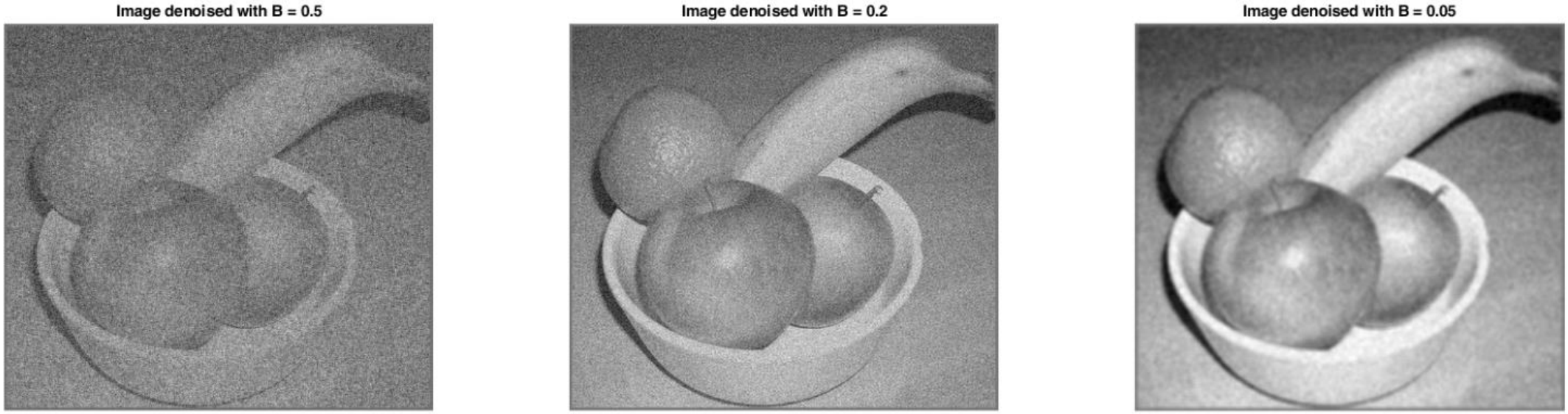

Figure 4: Image with noise std = 0.5 denoised using three different B values

The image that had Gaussian noise with std = 0.5 was processed with B values 0.5, 0.2 and 0.05. The image that resulted from denoising with B value 0.2 seemed to yield the best result, as the image that resulted from denoising with B value 0.5 still appears very distorted and noisy, perhaps even more so than the original noisy image. On the other hand, the image that was denoised with B = 0.05 seems less distorted than the one with B = 0.2, however in that case the sharpness of the image has disappeared and it looks very blurry. It can be deduced from these results that as B gets smaller, the noise is better removed, however the image loses its sharpness, and becomes blurry. If B is higher, the image remains very noisy but the edges are better preserved. In this case, the optimum value of B that retains sharp edges and eliminates noise the best is 0.2.

Part 5

Figure 5: Displayed image for Part 5

```
x = ReadMyImage("Part5.bmp");
DisplayMyImage(x);

% processing image with h1 filter
h1 = [1 0 -1; 2 0 -2; 1 0 -1];
y1 = DSLSI2D(h1, x);
DisplayMyImage(y1.^2)
```

Figure 6: Image processed with h1 filter

When the image is processed with the h1 filter, we can see that the vertical lines on the wings of the butterfly are emphasized more. The circles on the wings and the leaves of the flower on the bottom right are also visible. All of the visible shapes have a vertical alignment, as the silhoutte of the leaves of the flower can also be perceived to be separate vertical lines. This suggests that this filter is good at detecting vertical edges.

```
% processing image with h2 filter
h2 = [1 2 1; 0 0 0; -1 -2 -1];
y2 = DSLSI2D(h2, x);
DisplayMyImage(y2.^2);
```

Figure 7: Image processed with h2 filter

When the image is processed with the h2 filter, it can be observed that now the horizontal edges have been emphasized more and vertical edges haven't appeared at all. The vertical lines on the wings of the butterfly, which were visible in the previous image have mostly disappeared in this one. The lines that are more horizontally aligned to the top of the wings are more visible. In Figure 7, the horizontally aligned piece of plant above the butterfly is visible, whereas it wasn't visible at all in the previous image. Also, the leaf to the bottom left of the butterfly, whose edges are horizontally aligned is also visible but in the image that was processed with the h1 filter, it wasn't visible. Additionally, the tips of the leaves of the flower on the bottom right are also visible, as the top parts of the leaves are also horizontal. In the previous picture, the sides of those leaves were visible as they are vertical. Finally, the circles on the wings of the butterfly are emphasized as in the previous picture, because the lines that create the top and bottom halves of the circles could be detected to be horizontal edges by the filter. These results suggest that this filter is good at detecting horizontal edges, while the previous filter was good at detecting vertical edges.

```
% processing image with s3
DisplayMyImage(sqrt(y1.^2 + y2.^2));
```

Figure 8: Image processed with last filter

In the third image (Figure 8), it is observed that almost all of the edges present in the original picture are strongly emphasized. The image processed with the h1 filter could only detect vertical edges, while the h2 filter could only detect horizontal edges. We can say that the last filter creates a good combination of the previous two filters to accurately detect and emphasize the edges in the original picture.

Part 6

Figure 9: Displayed image for Part6x

Figure 10: Displayed image for Part6h

```
x = ReadMyImage("Part6x.bmp");
DisplayMyImage(x);
y = ReadMyImage("Part6h.bmp");
DisplayMyImage(y);
```

```
% passing the image through the system
out = DSLSI2D(y, x);
DisplayMyImage(abs(out));
```

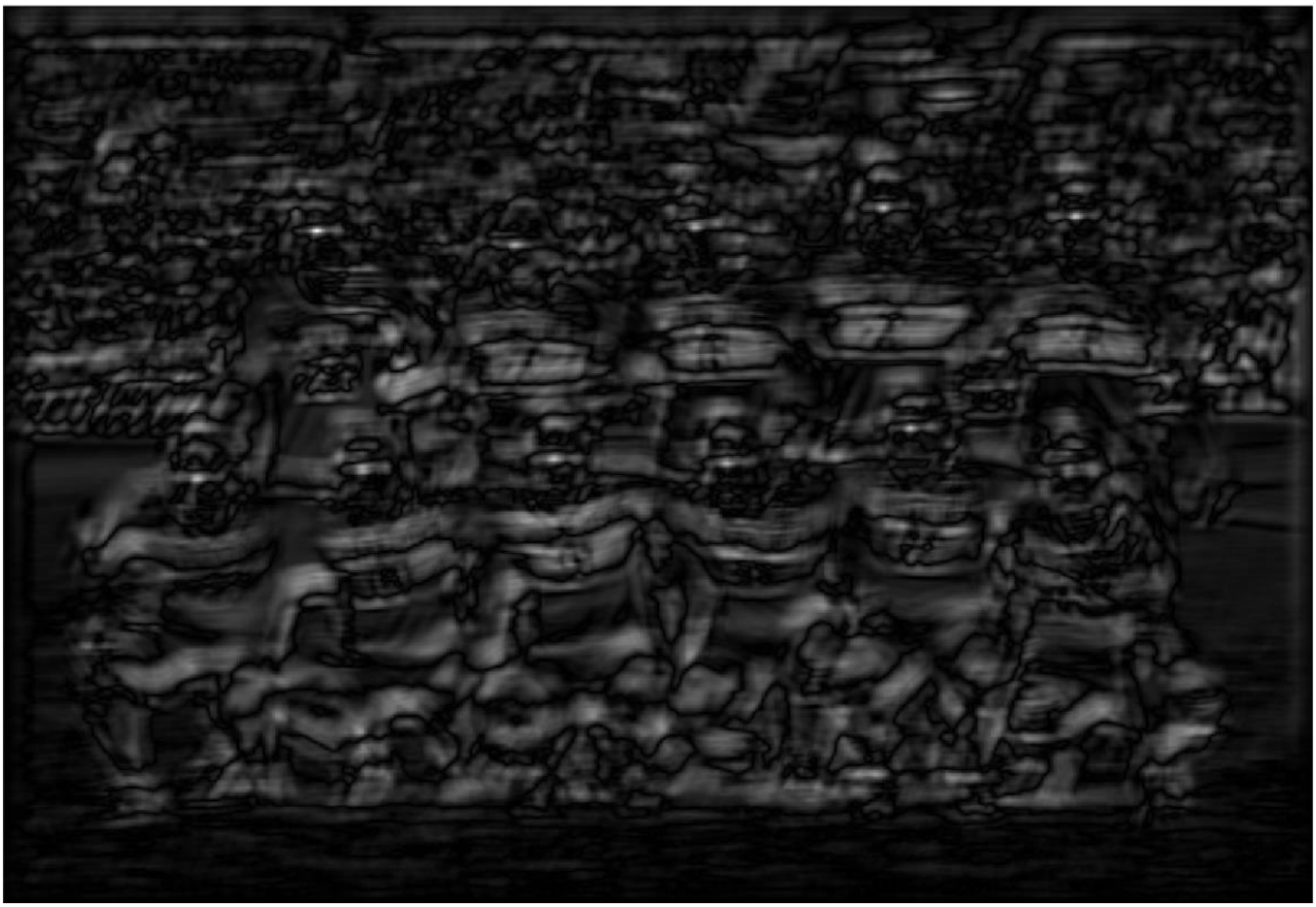

Figure 11: Output for |y[m,n]|

In Figure 11, it can be seen that there are distinct bright spots on all of the players' faces, however the spot on the face of the player who is on the bottom right isn't very visible. Also, there are a few bright places aside from the faces, such as on the background to the top right, where there is a bright sign in the original, colored image. The white parts of the jerseys of the players also seem to be bright and standing out.

Figure 12: Output for $|y[m,n]|^3$

Figure 13: Output for $|y[m,n]|^5$

We can see in Figure 12 that the bright spots on the faces of the players are much more emphasized when the power of 3 is taken, and there don't seem to be many other bright places in the picture. The bright spot on the face of Volkan (upper left) is the brightest, while the face of the player on the bottom right still seems to be unrecognized. The jerseys of the players are still faintly bright, but not as much as in the first picture (Figure 11).

In Figure 13, the faces of the players again have bright spots on them and Volkan's face is again the brightest, perhaps due to using his face as the impulse response of the system. There don't seem to be any other bright spots, such as the jerseys or in the background. However, it is harder to identify the bright spots and where the faces are in this case because the spots are very faint. It can be concluded that taking the third power to display the image yields the best result, where the faces can be detected without any confusion.

Overall, the matching filter method proved to be mostly successful, and it was pretty accurate at determining where Volkan's face is, as well as the other players' faces, with the exception of one player. In a more realistic case, probably the face of Volkan from another picture would be used to detect his face in this image, but mostly this method worked well.

Index of comments

1.1 Good work.

Total grade: 100