EEE391 - Matlab Assignment 2 Report

Name Surname: Ahmet Faruk Ulutaş

ID: 21803717

EEE 391-1

Part 2:

We may use the same method we used for the 1D DT LTI system to determine the input-output relation of a 2D DS LSI system. The 2D discrete impulse signal is first defined as:

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

The input signal x[m,n] may therefore be expressed as a superposition of shifted impulse signals:

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

The response of the 2D DS LSI system to [m,n] may thus be defined as h[m,n] (impulse response). We understand that the system's answer to [mk,nl] should be h[mk,nl] using the spatial invariance property of the system. The input-output relation of the 2D DS LSI system may therefore be expressed as follows utilizing the system's linearity property:

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

The 2D convolution of x[m,n] and h[m,n] may then be used to describe the operation in Eq. (5), which we designate as:

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

Eq. (2), which represents the input-output relation for a 1D DT LTI system, is presented here in a 2D form.

Part 3:

The contents of the DSLSI2D.m file are listed below.

```
function [y] = DSLSI2D(h,x)

Mh = size(h, 1);

Nh = size(h, 2);

Mx = size(x, 1);

Nx = size(x, 2);

for k=0:Mh-1

  for l=0:Nh-1

      y(k+1:k+Mx,l+1:l+Nx)=0;

  end

end

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 console output is shown below.

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

     2     1     1
    -3     0     2
     1    -1     2

>> h = [2 1;-1 0]
h =

     2     1
    -1     0

>> DSLSI2D(h, x)
ans =

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

Part 4:

The altered DisplayMyImage.m for part 4 is shown below.

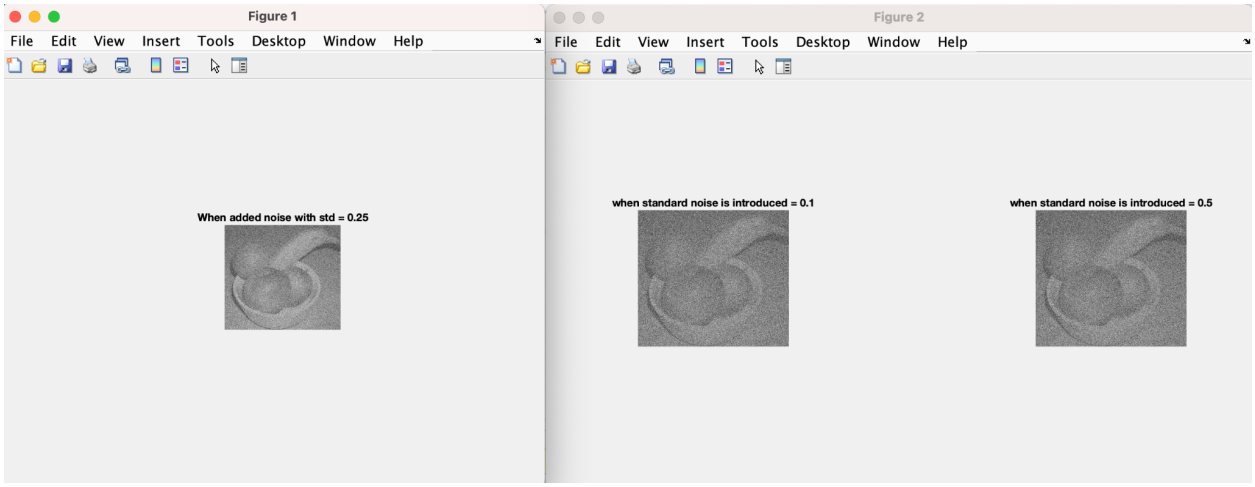
```
function []=DisplayMyImage(Image, im_title)

Image=Image-min(min(Image));

%figure;

imshow(uint8(255*Image/max(max(abs(Image)))));

title(im_title);
```

The Matlab code for introducing noise to the picture and showing the results is provided below.

```
x = ReadMyImage("Part4.bmp");

mean = 0;  % the Gaussian noise's average

size_x = size(x);  % size of the image

std = 0.1;   % Gaussian noise' standard deviation

noise = random('norm', mean, std, size_x);  % produce Gaussian

noise

std = 0.25;   % Gaussian noise' standard deviation

noise = random('norm', mean, std, size_x);  % produce Gaussian

noise

noisy_x_0_25 = x + noise;

subplot(1,3,2);

im_title = "When added noise with std = 0.25";

DisplayMyImage(noisy_x_0_25, im_title);

std = 0.5;   % Gaussian noise' standard deviation

noise = random('norm', mean, std, size_x);  % produce Gaussian

noise

noisy_x_0_1 = x + noise;
```

```
figure;

subplot(1,3,1);

im_title = "when standard noise is introduced = 0.1";

DisplayMyImage(noisy_x_0_1, im_title);

noisy_x_0_5 = x + noise;

subplot(1,3,3);

im_title = "when standard noise is introduced = 0.5";

DisplayMyImage(noisy_x_0_5, im_title);
```

The noise-filled photos are shown below.



As can be observed from the figure, the image becomes increasingly hazy and unrecognizably indistinct as we raise the standard of the Gaussian noise.

Low pass filters may not be the greatest option for removing noise from a picture for a number of reasons, despite the fact that they attenuate the high frequency components of an image while maintaining the low frequency components:

- By removing small details and sharp edges, low pass filters can blur the image.

- High frequency noise may still be present even after using low pass filters.

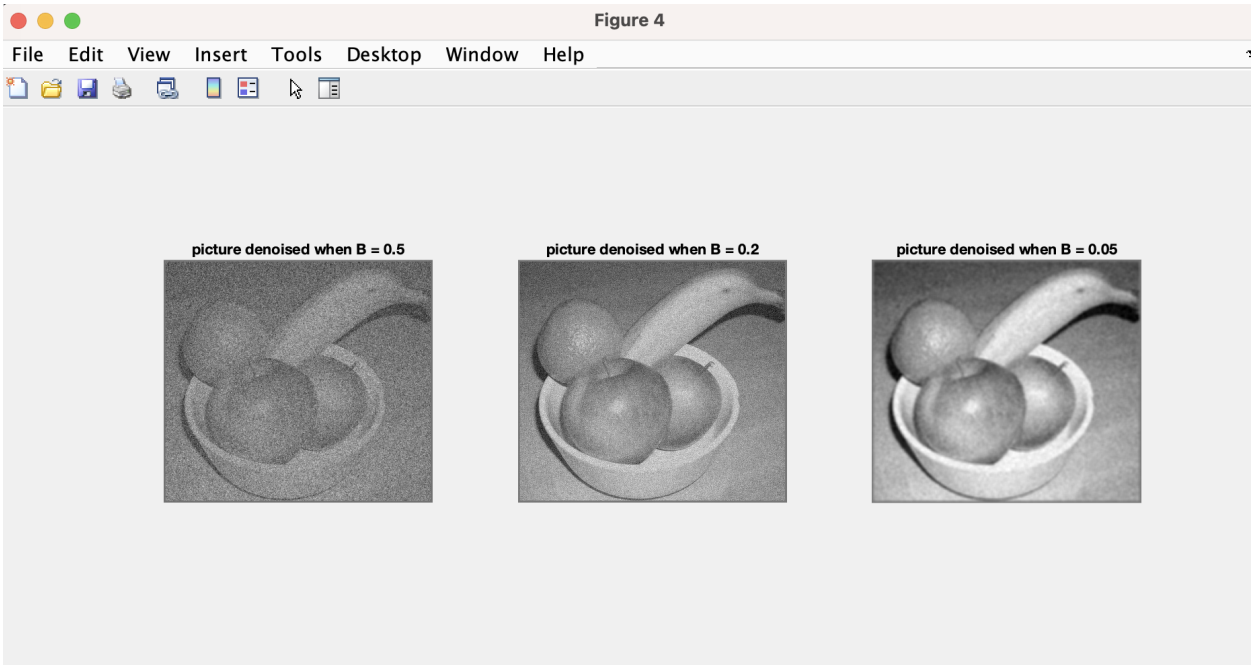
- Low pass filters may also eliminate crucial visual elements like borders and texture, which might lower the image's overall quality.

The code for denoising the image and presenting the results may be found below.

```
D17 = rem(21803717, 17);
Mh=20+D17; % how many rows are in h
Nh=20+D17; % how many columns are in h
figure;
subplot(1, 3, 1);
B = 0.5;  % the system's bandwidth
h = zeros(Mh, Nh);  % initialize h with a zero-based matrix
% for loops that are nested over m and n
for m = 1:Mh
  for n = 1:Nh
      % how to calculate the value of h[m, n]
      h(m, n) = sinc(B*(m-(Mh-1)/2)) * sinc(B*(n-(Nh-1)/2));
  end
end
reduced_x_0_5 = DSLSI2D(h, noisy_x_0_5);
im_title = "picture denoised when B = 0.5";
DisplayMyImage(reduced_x_0_5, im_title);
subplot(1, 3, 2);
B = 0.2;  % the system's bandwidth
h = zeros(Mh, Nh);  % initialize h with a zero-based matrix
% for loops that are nested over m and n
for m = 1:Mh
  for n = 1:Nh
      % how to calculate the value of h[m, n]
      h(m, n) = sinc(B*(m-(Mh-1)/2)) * sinc(B*(n-(Nh-1)/2));
  end
```

```
end

reduced_x_0_2 = DSLSI2D(h, noisy_x_0_5);

im_title = "picture denoised when B = 0.2";

DisplayMyImage(reduced_x_0_2, im_title);

subplot(1, 3, 3);

B = 0.05;  % the system's bandwidth

h = zeros(Mh, Nh);  % initialize h with a zero-based matrix

% for loops that are nested over m and n

for m = 1:Mh

  for n = 1:Nh

      % how to calculate the value of h[m, n]

      h(m, n) = sinc(B*(m-(Mh-1)/2)) * sinc(B*(n-(Nh-1)/2));

  end

end

reduced_x_0_05 = DSLSI2D(h, noisy_x_0_5);

im_title = "picture denoised when B = 0.05";

DisplayMyImage(reduced_x_0_05, im_title);
```

The photos that have been denoised are shown below.

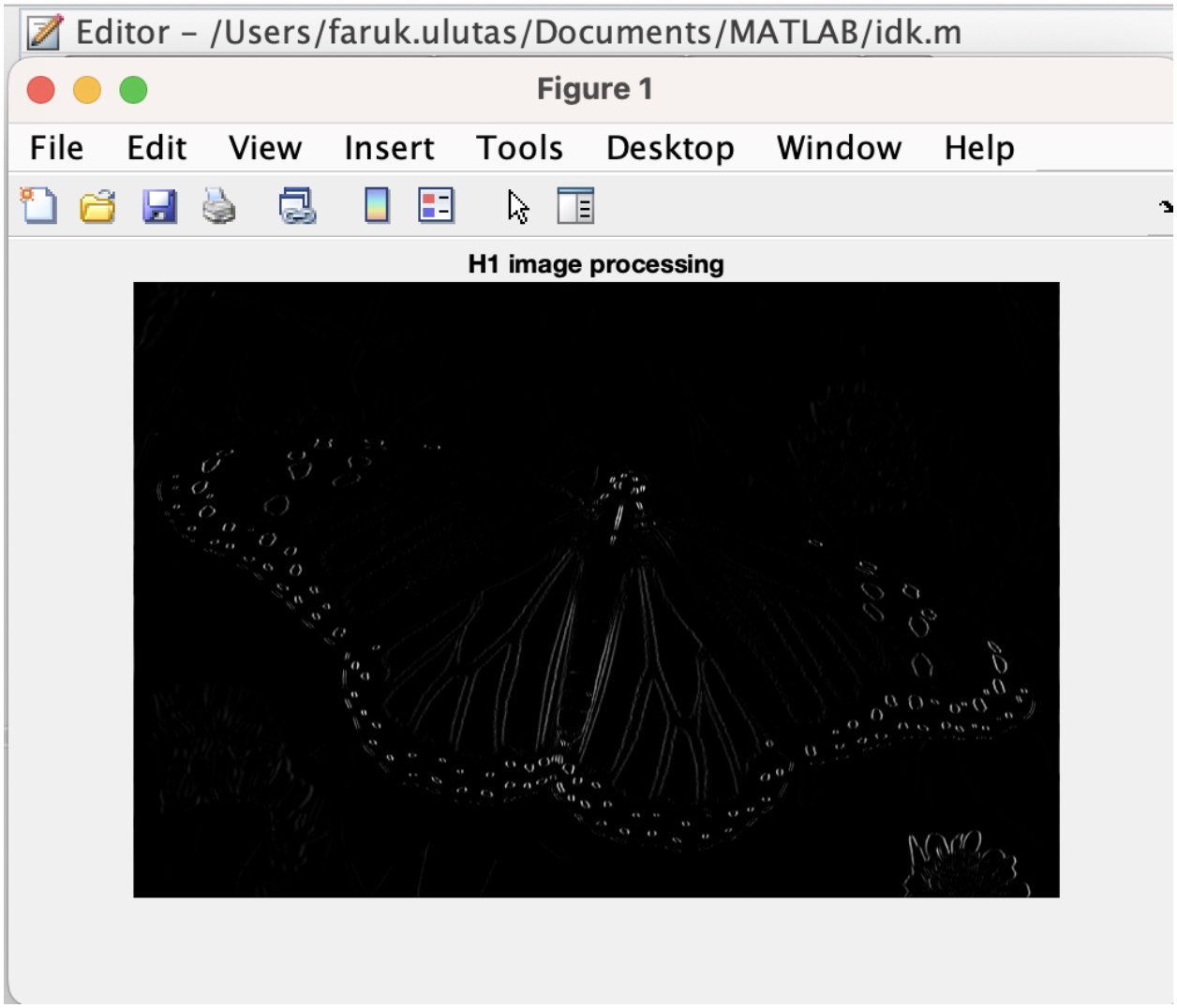


As can be seen from the picture, the image gets clearer and easier to recognize as we reduce the free parameter B, which sets the filter's bandwidth. As a result, B = 0.05 appears to be the best B value for this activity.

Part 5:

The Matlab code to process the picture with the filter whose impulse response is h1 is provided below, along with a display of the image code.

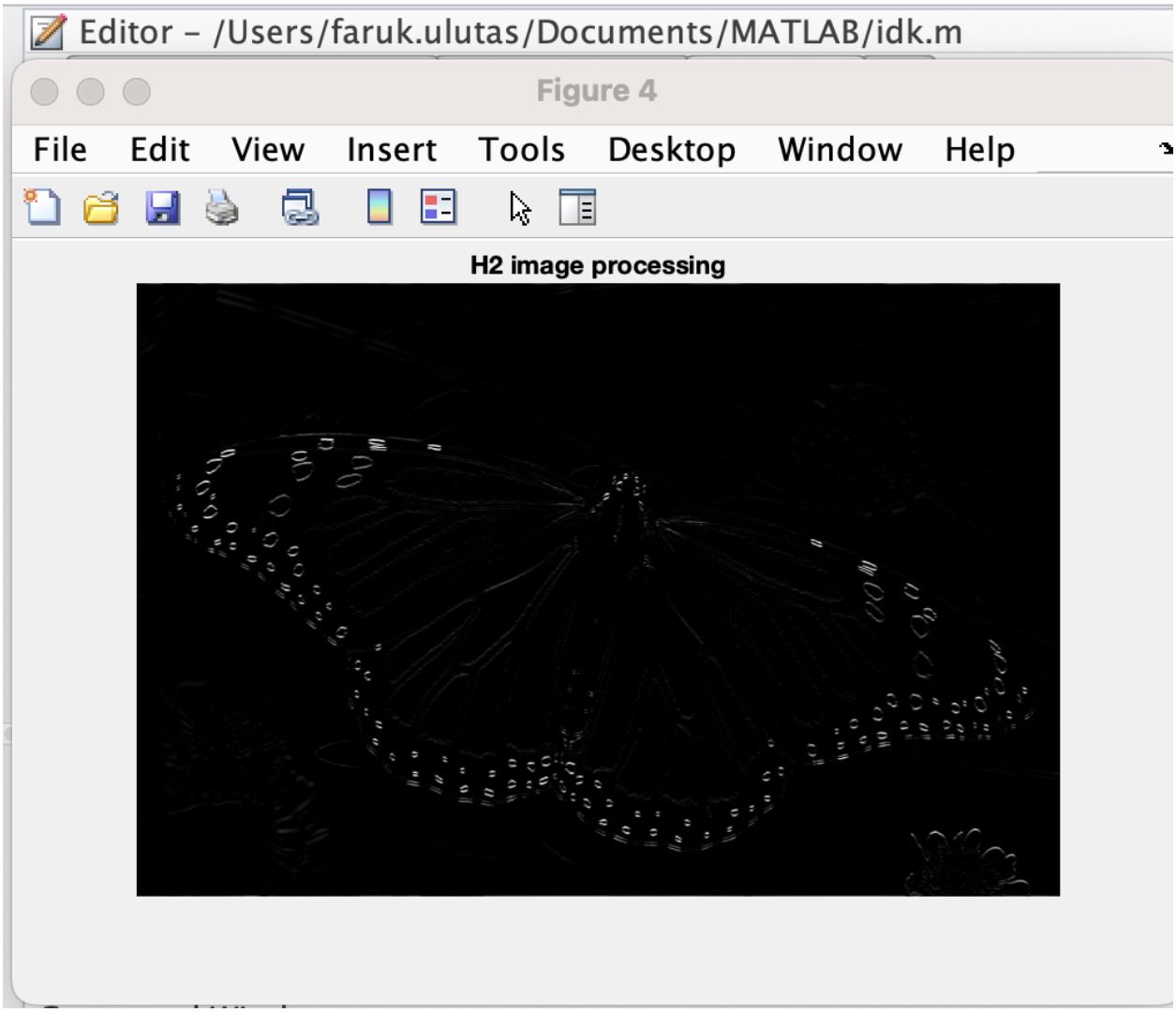
```
x = ReadMyImage("Part5.bmp");

h1 = [1 0 -1; 2 0 -2; 1 0 -1];

y1 = imfilter(x, h1);

s1 = y1.^2;

figure;

DisplayMyImage(s1,"H1 image processing");
```



The butterfly's borders, as well as its body and head, are highlighted in this picture, as can be seen.

The Matlab code to process the picture with the filter whose impulse response is h1 is provided below, along with a display of the image code.

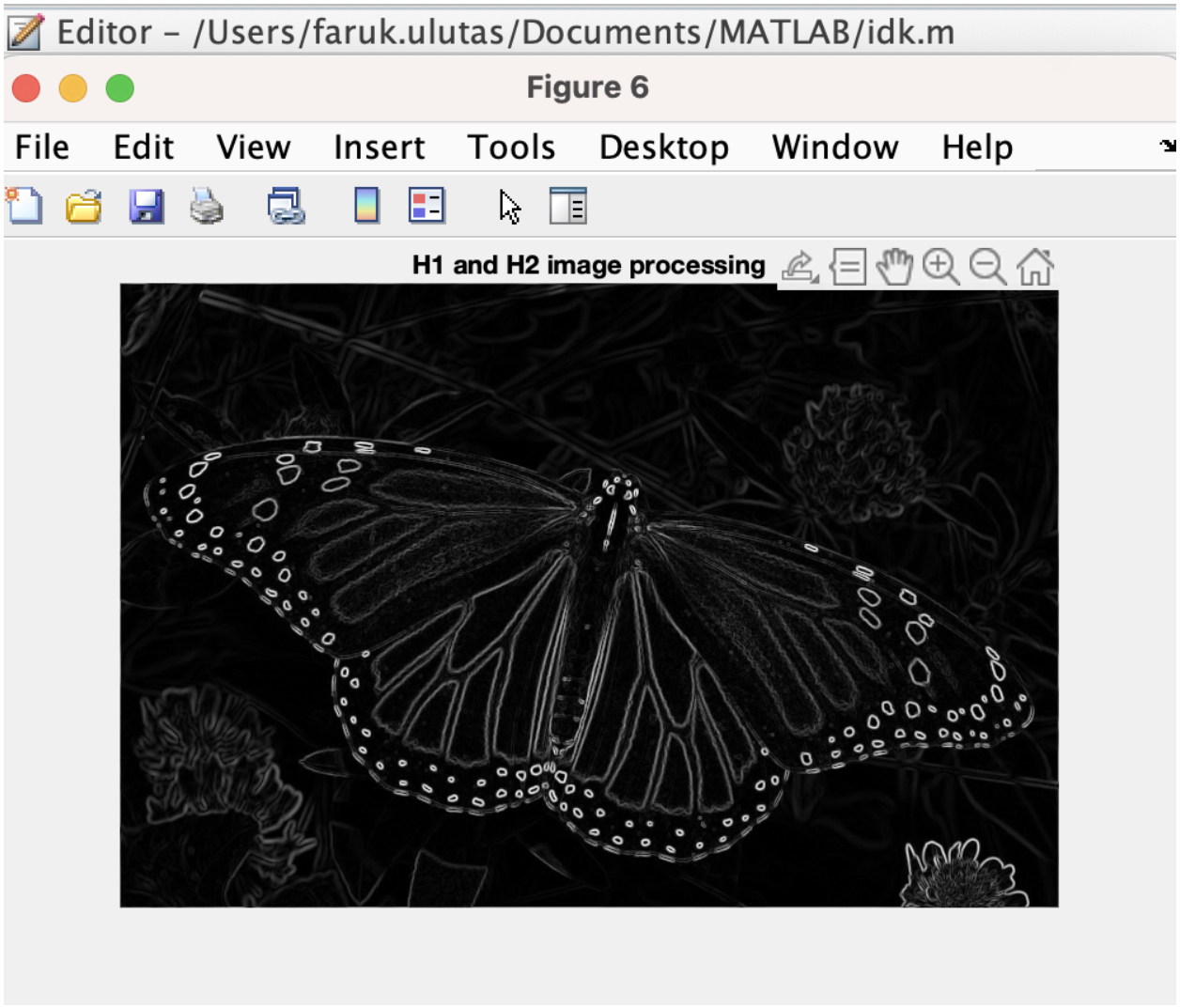
```
h2 = [1 2 1; 0 0 0; -1 -2 -1];

y2 = imfilter(x, h2);

s2 = y2.^2;

figure;

DisplayMyImage(s2,"H2 image processing");
```



The butterfly's head and its margins are highlighted in this photograph, as can be seen from the image.

The image itself and the Matlab code for producing the image s3[m, n] are shown below.

```
s3 = sqrt(s1 + s2);

figure;

DisplayMyImage(s3,"H1 and H2 image processing");
```



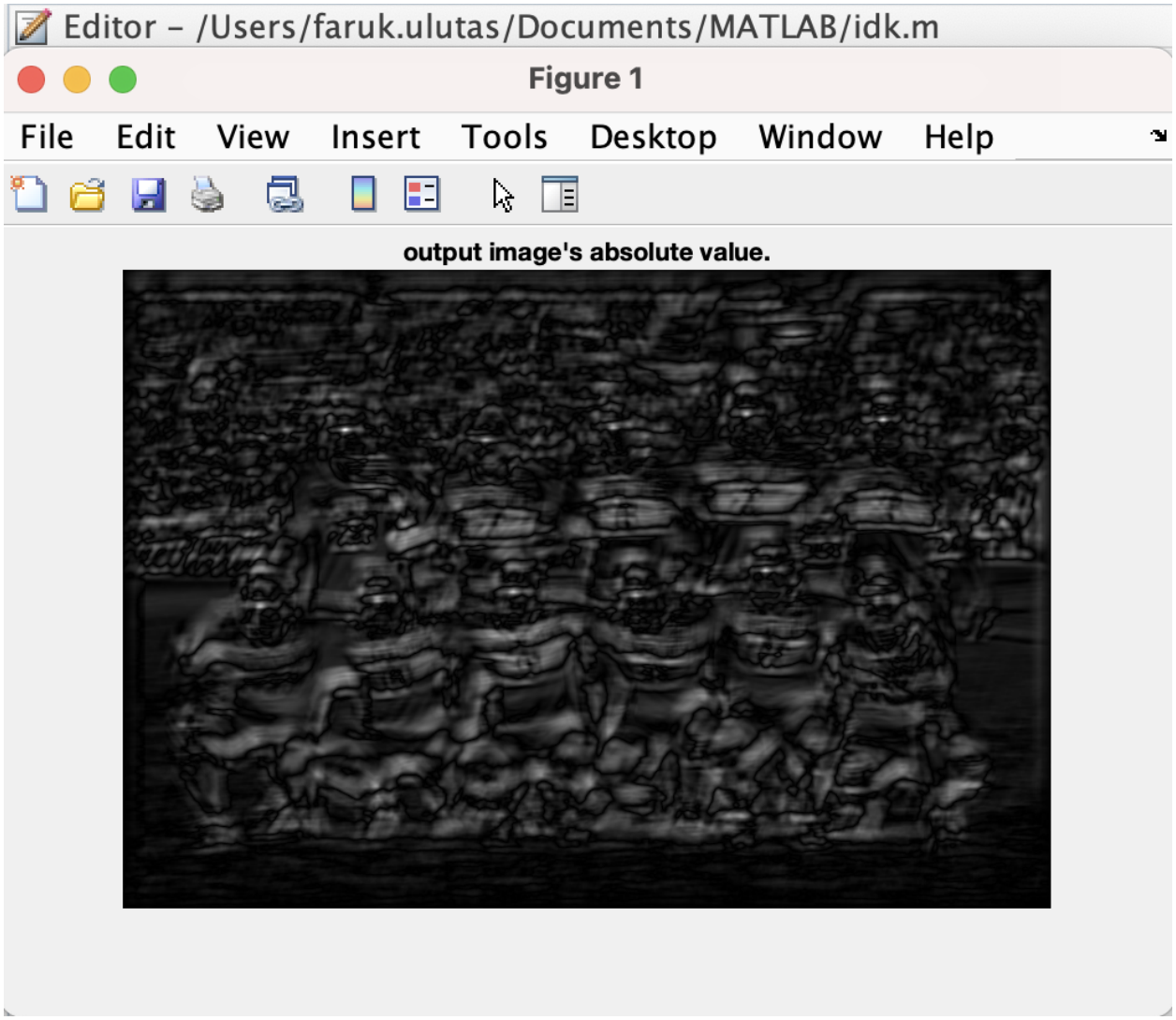
Most of the corners in the picture produced by h1 were highlighted. The picture produced by h2 highlighted edges the most. However, all high frequency items are highlighted in this picture.

Part 6:

Corners were highlighted the most in the image produced by h1. Most edges were highlighted in the picture produced by h2. But in this picture, all of the high-frequency material is highlighted.

```
x = ReadMyImage("Part6x.bmp");

DisplayMyImage(x,"");

h = ReadMyImage("Part6h.bmp");
```

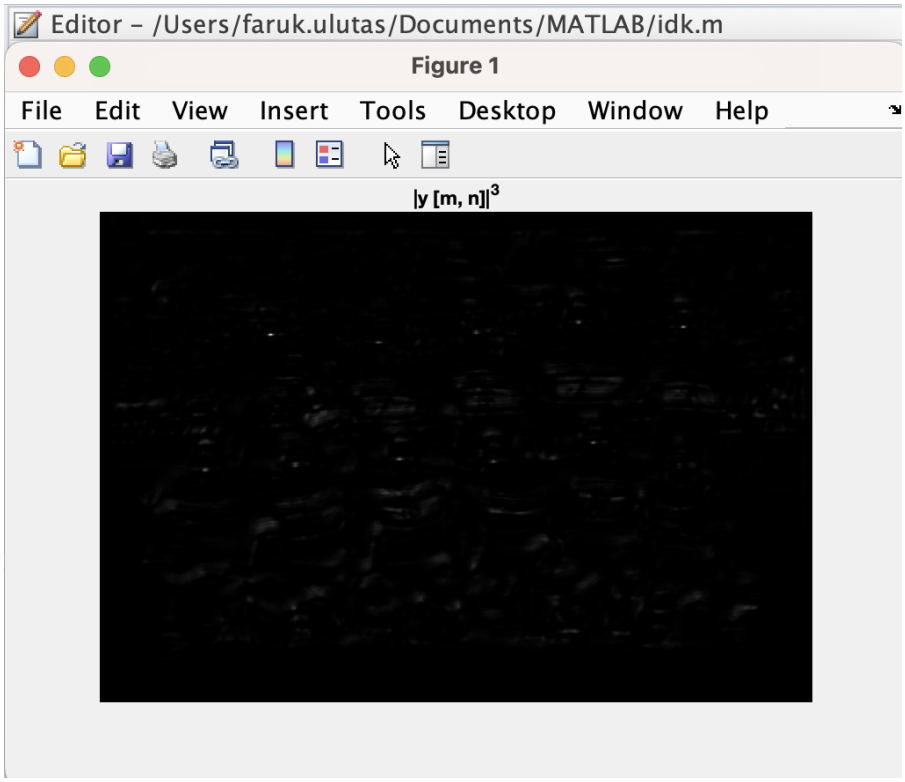
```
DisplayMyImage(h,"");

output_image = DSLSI2D(x, h);

abs_value = abs(output_image);

DisplayMyImage(abs_value, "output image's absolute value.");
```



Bright spots can be observed throughout the photograph, including in the faces and other areas, as can be seen in the image above. Therefore, bright dots can both represent noise or other environmental elements as well as the presence of a face in the input image.

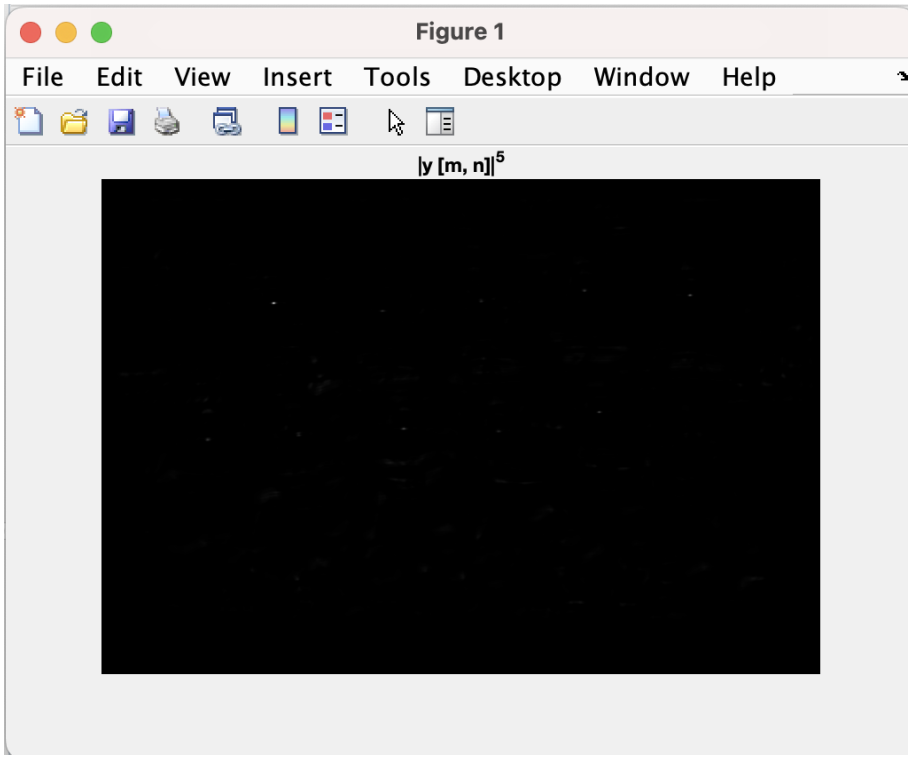
The code line in Matlab below computes and shows |y[m, n]^3.

```
DisplayMyImage(abs_value.^3, "|y [m, n]|^3");
```



The code line in Matlab below computes and shows |y[m, n]|^5.

```
DisplayMyImage(abs_value.^5, "|y [m, n]|^5");
```



The preceding image demonstrates that increasing by a factor of three does not adequately brighten only the faces in an image. Other than faces, there are still some shining areas. Additionally, when the 5th power is calculated, a face in the image is pointed out by around 10 brilliant spots. Therefore, you can identify the faces without any difficulty by using the 5th power. Although certain pattern recognition techniques exist that can even identify audience members' faces in the image's backdrop, the success of this approach is still not particularly effective. However, since this technique proved effective in identifying the football team's faces, it may be enough for images with minimal complexity.