IMAGE PROCESSING COURSE (EE6553)

Assignment 2

OCTOBER 30, 2020

SAEED KAZEMI
Student Number 3713280

DUE: OCT 30, 2020

A. Abstract

In this assignment, I worked on a MATLAB program for implementing a lowpass filter called Butterworth and Spatial and frequency operations on an image for finding the correlation between two images. This project contained four functions and the main m-file that call all functions for solving examples. In section B, the techniques used for this assignment will be discussed. In the next section, the output results are checked based on images that are shown in part D, and finally, in the appendix, my codes are added.

B. Technical discussion

In this section, the techniques used in this program are discussed. For making the low pass filter, we need to compute the 2D DFT from the input image. To do this, we utilized the fft function in MATLAB. The output of this function is a matrix of the same size as the input image but with complex value. If the magnitude of the output is plotted, we can see the input image in the frequency domain. In the frequency domain, all the low frequencies are in the corners. So, we used the fftshift for putting low frequencies in the middle of the image, this could help us to understand better. Now for making a low pass filter, we need a function that has a high gain in the low frequencies and a low gain in the high frequencies. The below image shows an ideal low pass filter.

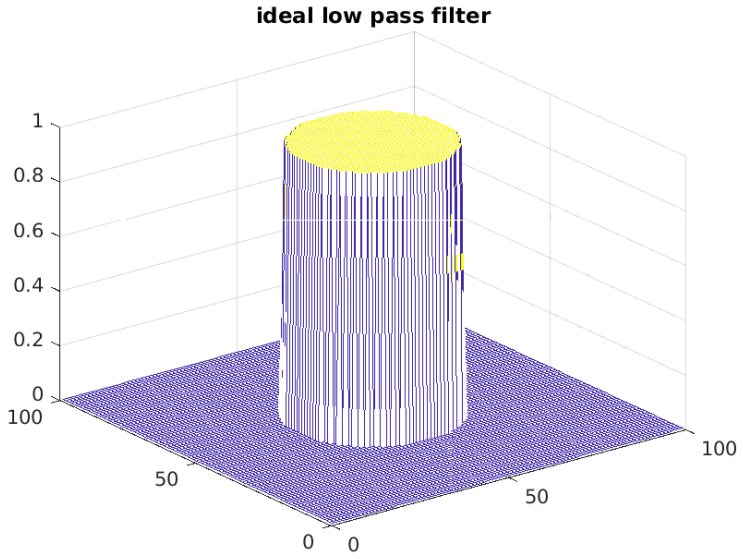

Figure 1) the ideal low pass filter.

This filter has some ringing effects on the output image. Another filter that we can use it is Butterworth. This filter has a smooth and fixed gain for the passband and zero gain for others. Also in its cut frequency, the gain is 0.5. In addition, we can tune this change with the order of Butterworth. If we increase the order, this filter change to the ideal filter. Figure 2 shows one dimension filter for a different order. As can be seen, the highest order has a flat bandpass, as well as a complex implication. To make this kind of filter, we need to calculate the Euclidian distance of each pixel to the center of the image. For calculating the center of the image, if the input image has an even dimension, the last column or row was eliminated. For example, a 100*100 image changed to 99*99. Then based on the cut frequency and the order of the filter, the below matrix was calculated.

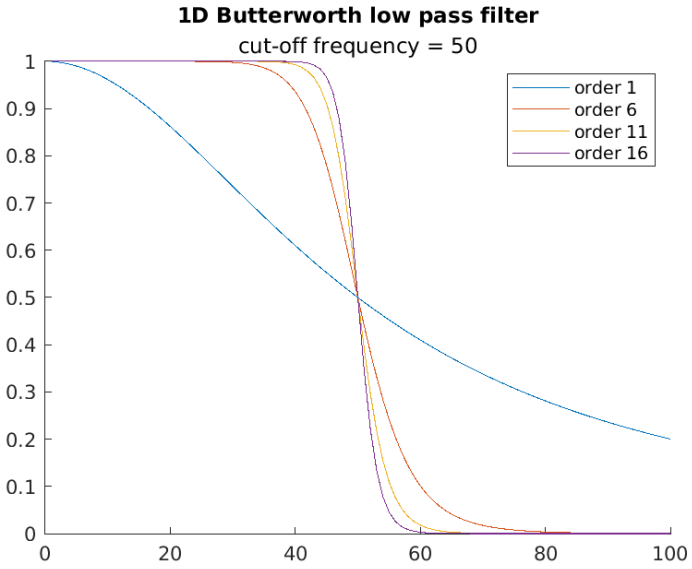

Figure 2) one dimension low pass filter with different orders

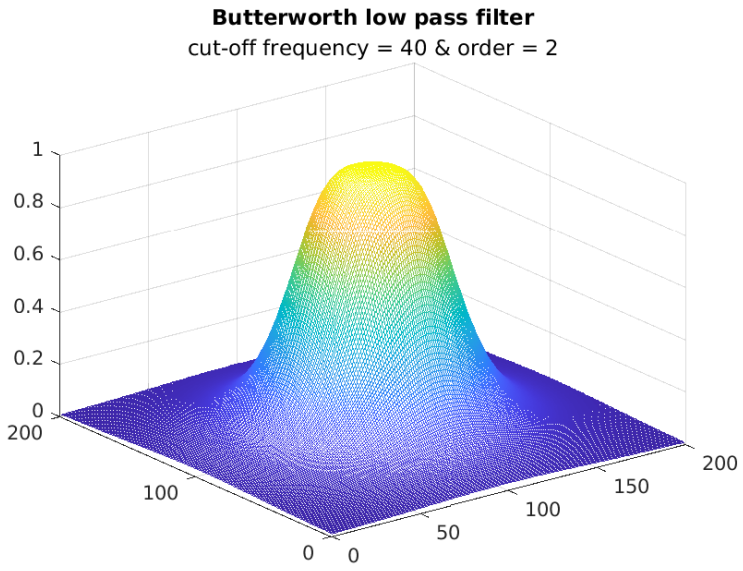

Figure 3) Two-dimension Butterworth filter

*filter(i,j)= 1/(1 + (D/CutFrequency)^(2*FilterOrder));*

This matrix is multiplied element-wise to the fft image to produce the output image in the frequency domain. So as the filter shows, the coefficients of the low frequencies are more than the coefficients of the high frequency.

Correlation is a technique for finding an object in an image or calculating the similarity of two images. In this technique, both images multiply element-wise and if they have the same value the output has a high value. For this understanding better, consider two vectors below.

A = [2, 3, 6, -1, 4]

B = [3, 6, -1]

$$Corr_{AB} = \sum_m \sum_n A[n]B[n-m] = [18, 46, 8]$$

As the correlation vector shows the max value shows the similarity of the two signals. But this method has a disadvantage that if the energy of one of the signals is high, the output will be high without considering the similarity. So for solving this, we can divide the result of the element-wise multiply into the root energies of both signals. This technique is called the normalized correlation. The equation below shows the formula for two 1D signals.

$$Corr_{AB} = \frac{\sum_m \sum_n A[n]B[n-m]}{\sqrt[2]{\sum_m A^2[m] \times \sum_n B^2[n]}}$$

The above methods are applied in the spatial domain. We can apply a correlation in the frequency domain. The correlation formula looks like the convolution if instead of m, use -m.

$$\text{Corr}(\ u(m), v(m)\ ) = u^*(-m) * v(m) = u(-m) * v(m)\ [u(m) \text{ is a real signal, so } u^*(m)= u(m)]$$

So, the above formula change to multiply in the frequency domain. In the first step, we calculate the fft for both images. In this step, because the object size is less than the scene image, we use the fft function like fft(object, X_size_of_scene, Y_size_of_scene). Y_size_of_scene and X_size_of_scene are the sizes of the scene image. Therefore, both Fourier transformations have the same size in the frequency domain. Then, we use the below equation for calculating the correlation.

$$\text{Corr}(\ u(m), v(m)\ ) = u^*(-m) * v(m) = \mathcal{F}^{-1}\{\ U^*(f)\ V(f)\}$$

Also, we can use this formula to be independent of the energy level.

$$Norm\_Corr\big(\ u(m), v(m)\big) = \frac{u^*(-m)*v(m)}{abs(\ u(-m) * v(m)\ )} = \mathcal{F}^{-1}\{\frac{U^*(f)\ V(f)}{|U^*(f)\ V(f)|}\}$$

The max output of these equations shows the location of the object in the scene image. For this lab, we implement three functions for correlation.

C. Discussion of results

In this part, the result of these two functions is discussed. For the test, we wrote an m-file that called the functions. The first function is called 'Butterworth'. This function has three arguments, the input image, the cut-off frequency, and the order of the filter. Also, this function returns the output result as well as the filter in the frequency domain. This function plots the input image in the spatial domain and frequency domain, filter in the frequency domain, and the output image.

For the Butterworth low pass filter, we use two types of images, artificial and real images. The first image shows a real image that is combined with a sin noise. This kind of noise is a delta in the frequency domain (Figure 6). As can be seen, for the first image, we have two delta functions in the image. Figure 5 shows the picture in the spatial domain. The second image is an image that is composed of a delta function. This image has all of the frequencies in the frequency domain. Figure 4 shows the Fourier transform for the 2D delta function. As Figure 4 shows, there is a fixed value for all frequencies.

Figure 4) the Fourier transform of the delta function

So for the first image, the input image has a sin noise. This noise in the frequency domain is two peaks on his frequency (Figure 6). To eliminate this noise, we use the low pass filter. The output does not have any sin noise, and the resulting image looks blurred image because all high frequencies that show edges are removed.

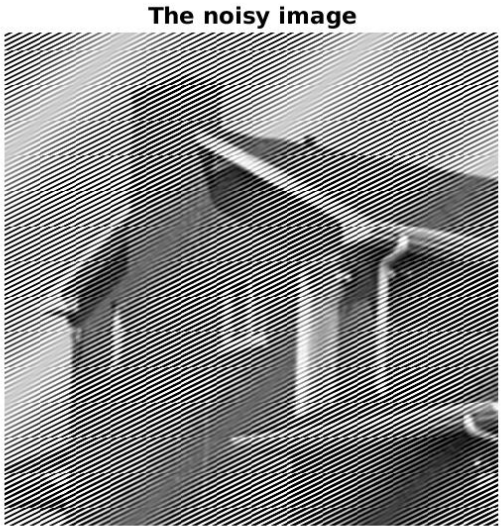

Figure 5) The real image with a sine noise

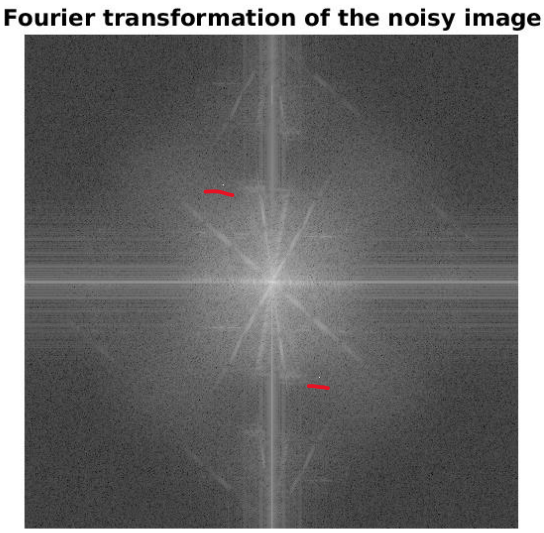

Figure 6) the sine function in the frequency domain. Red lines show the delta

Increasing the order of filter caused the output image to blurred. Although this increase has less effect on the output image for the orders more than four. To compute the frequency of sin noise. We need to find the distance of the peak from the center of the image. For example in Figure 6, the frequency of sin is 112.

In the second test, we use a single-pixel impulse as the input of our filter. This image has all frequencies. As a result, if we apply a low pass filter on it, the high frequencies are removed, and the output changes to a blurred image, with smooth edges. Figure 13 shows the results.

For question two, we implemented the correlation function in three methods, cross-correlation in the spatial domain, normalized cross-correlation in the spatial and frequency domain. These functions have two input arguments, the scene image, and the object image, and returns the correlation results and the spatial location of the maximum point in the correlation.

To test these functions, we used Figure 7 as a scene image to find the location of the K letter. Also, we used another image to show the similarity (Figure 32). In the second test, we rotated and scaled the object image to find out how much the algorithm is robust in terms of rotation and scale invariance. Figure 8 shows the object images for this part of the test.

Figure 7) The scene image

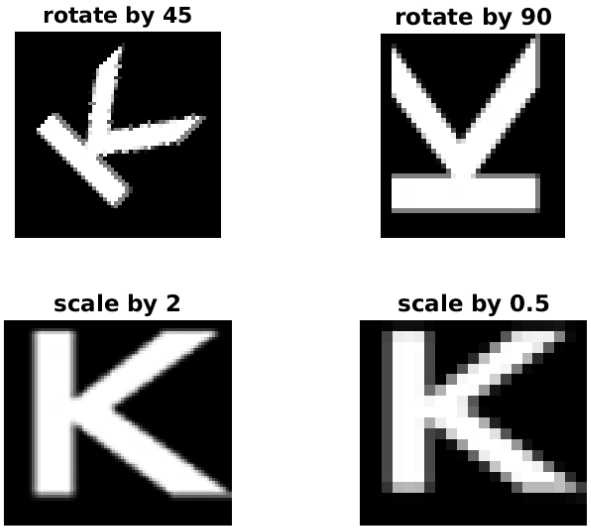

Figure 8) Test the correlator for rotation and scale invariance

As can be seen in Figure 14, 15, and 16, there are not different among these three methods. But in the second image, the cross-correlation method could not find the right location of the object. That would be because the brighter part of the image has more energy and the output will increase (Figure 17). But the normalization cross-correlation in frequency and spatial domain could find the right location of the object (Figure 18 and Figure 19).

In the second test, we used objects in Figure 8 as an object. For rotating 45, all three methods could not find the object (Figure 20, Figure 21, and Figure 22). However, for 90, these algorithms could find the object (Figure 23, Figure 24, and Figure 25). For scaling, we used two scales 2 and 0.5. for zooming out all three methods found the object. While

for zoom in, the result was not good (from Figure 26 to Figure 31). In general, we can conclude that these methods are not robust for scaling and rotating.

D. Image Results

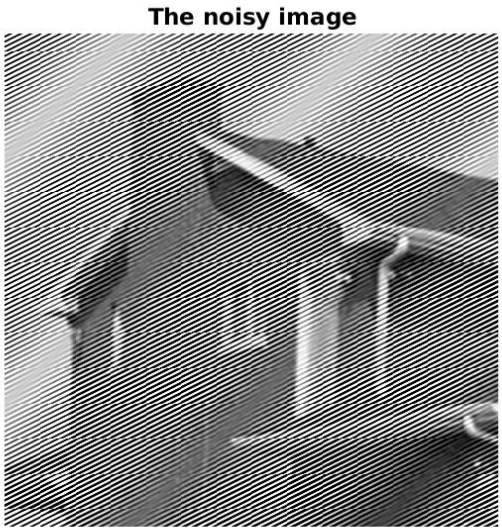

Figure 9) The noisy image

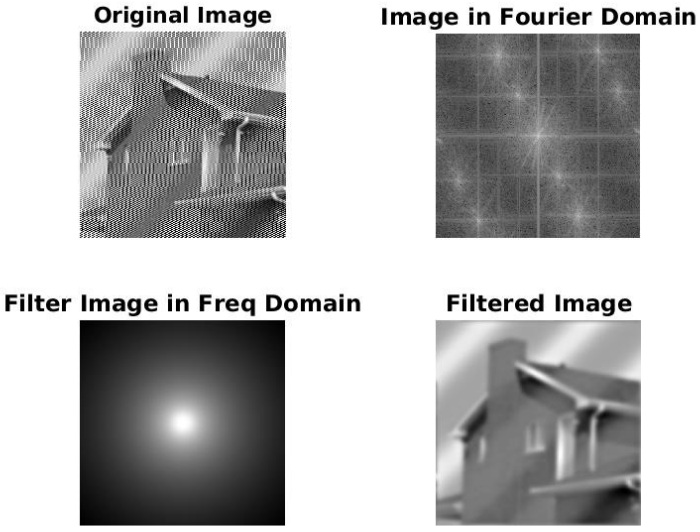

Figure 10) The result of the low pass filter, as it is obvious the output is blurred. The cut-off frequency and order are 20 and 2 respectively.

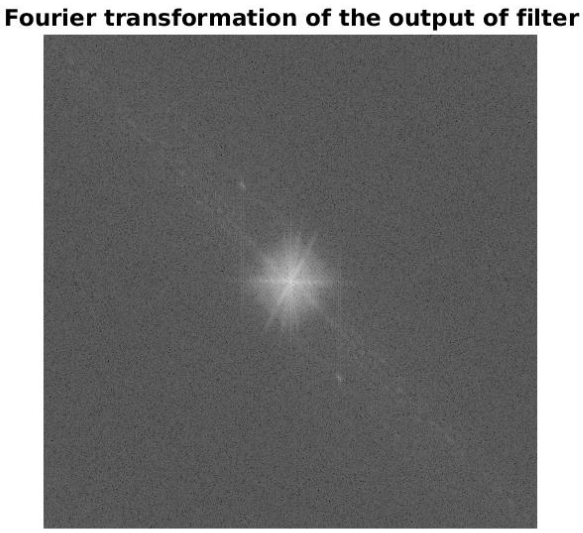

Figure 11) the Fourier transform of the output

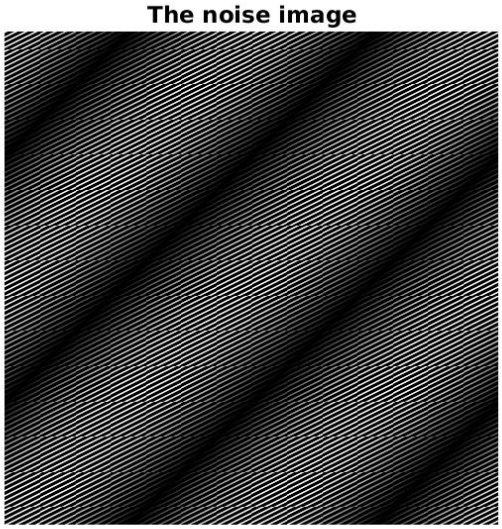

Figure 12) the sine noise

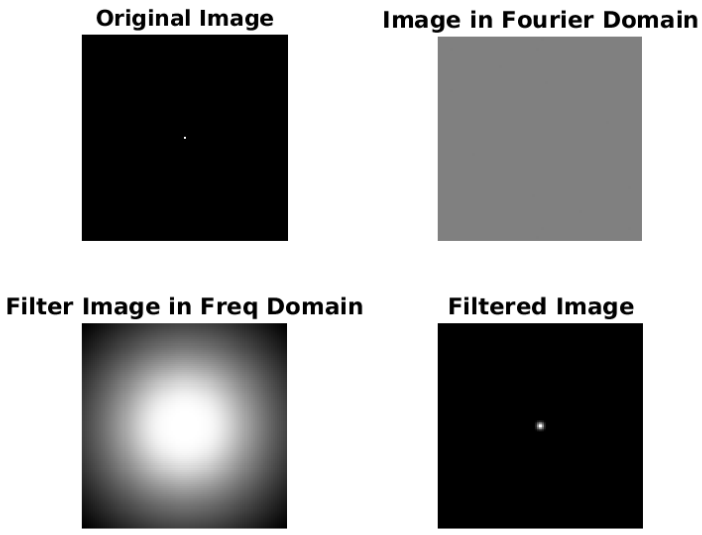

Figure 13) The result of the low pass filter on the delta function, as the second image shows delta has all of the frequencies. The output is blurred. The cut-off frequency and order are 20 and 2 respectively.

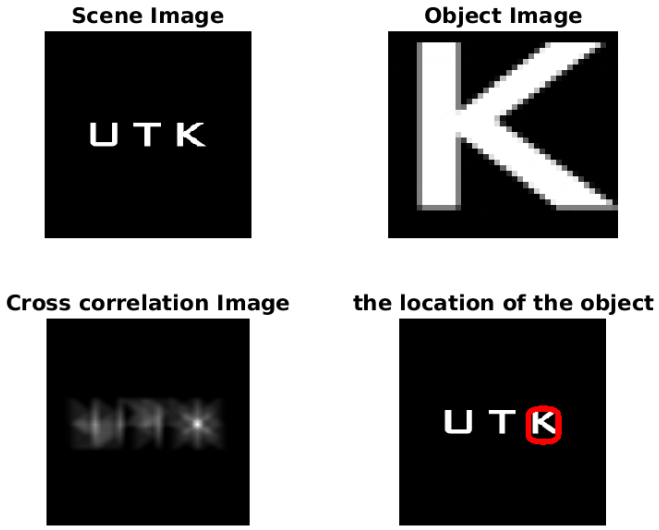

Figure 14) the output of the cross-correlation in the spatial domain without normalization.

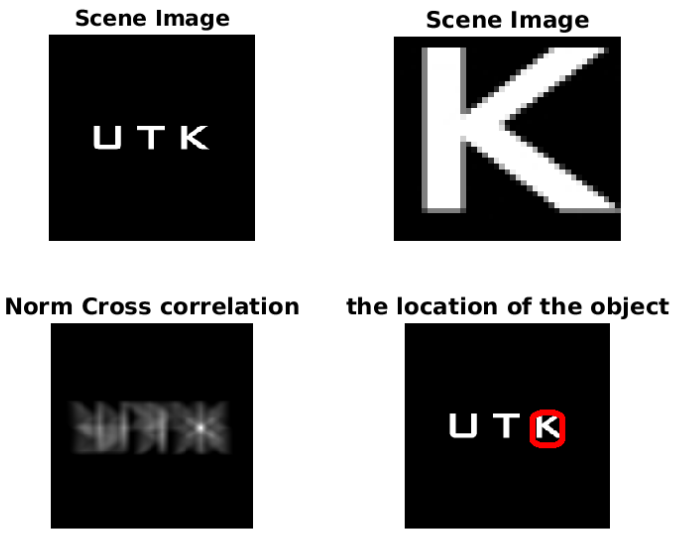

Figure 15) the output of the cross-correlation in the spatial domain with normalization.

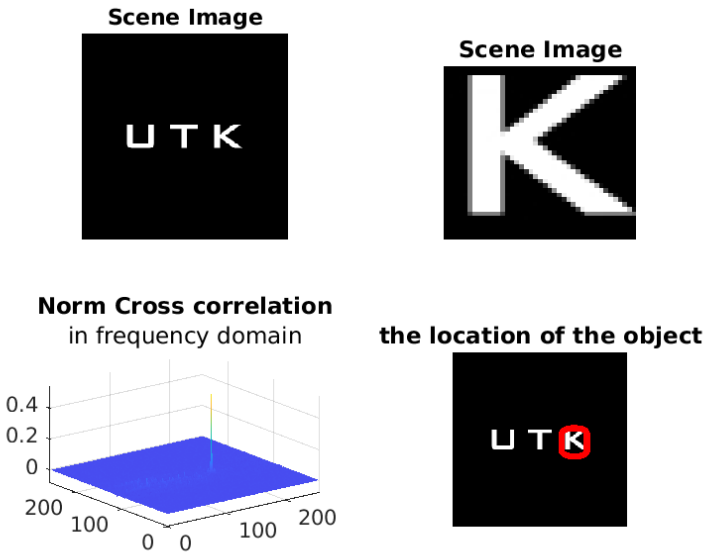

Figure 16) the output of the cross-correlation in the frequency domain with normalization.

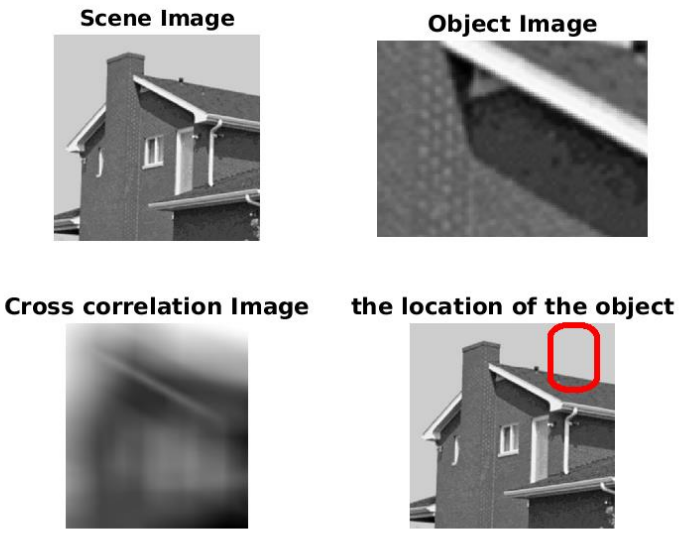

Figure 17) the output of the cross-correlation in the spatial domain without normalization.

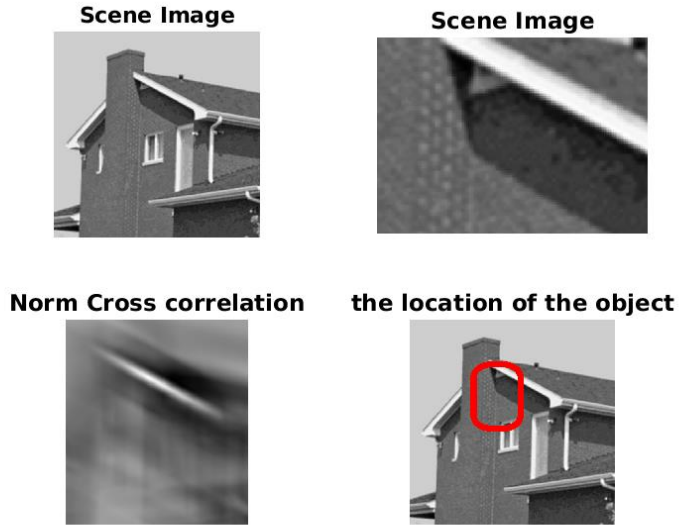

Figure 18) the output of the cross-correlation in the spatial domain with normalization.

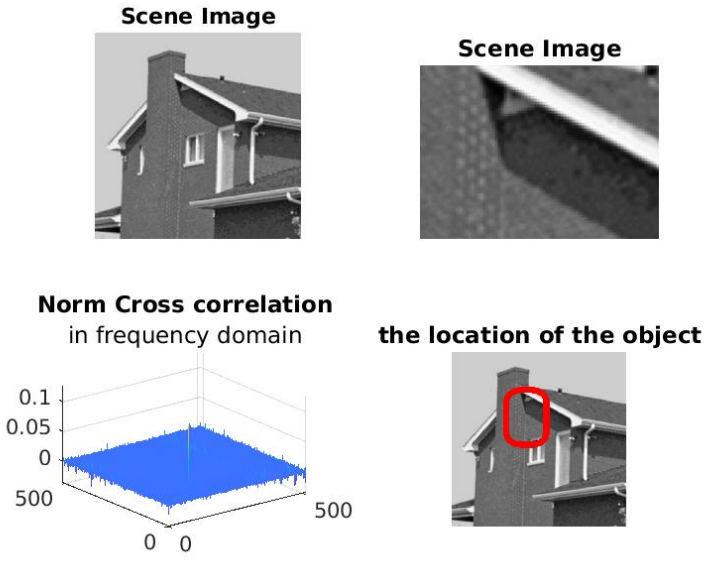

Figure 19) the output of the cross-correlation in the frequency domain with normalization.

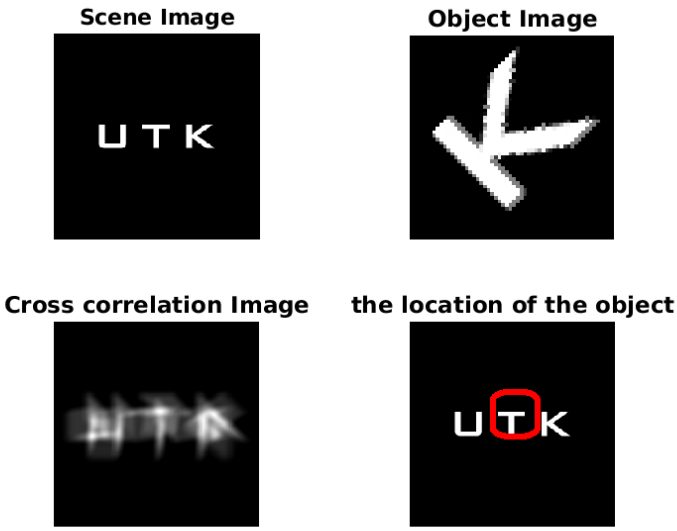

Figure 20) the output of the cross-correlation in the spatial domain without normalization and rotated object by 45 degrees.

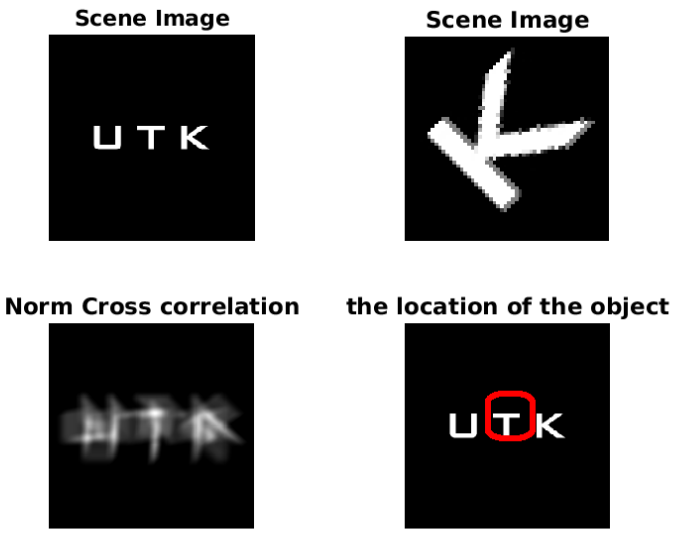

Figure 21) the output of the cross-correlation in the spatial domain with normalization and rotated object by 45 degrees.

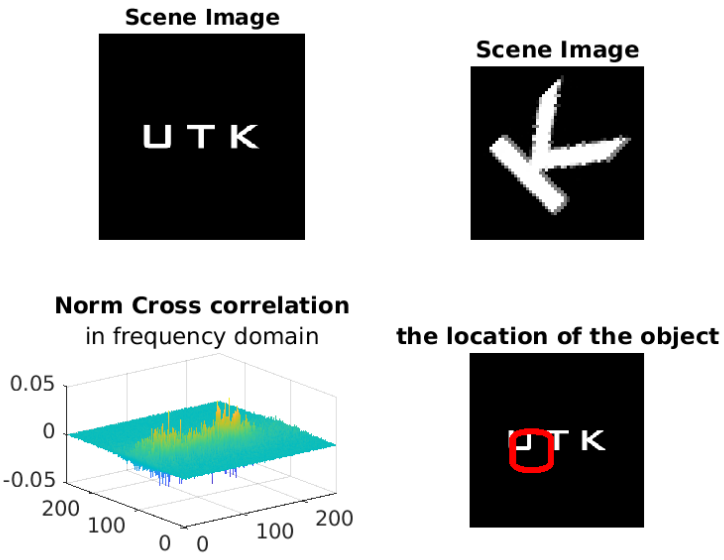

Figure 22) the output of the cross-correlation in the frequency domain with normalization and rotated object by 45 degrees.

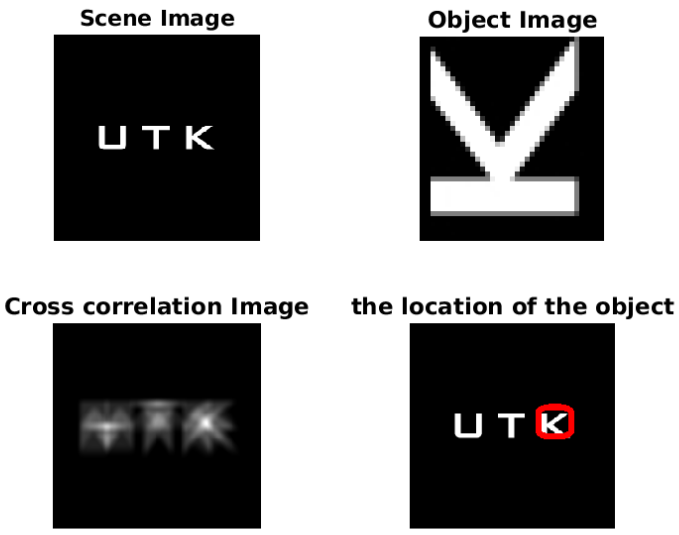

Figure 23) the output of the cross-correlation in the spatial domain without normalization and rotated object by 90 degrees.

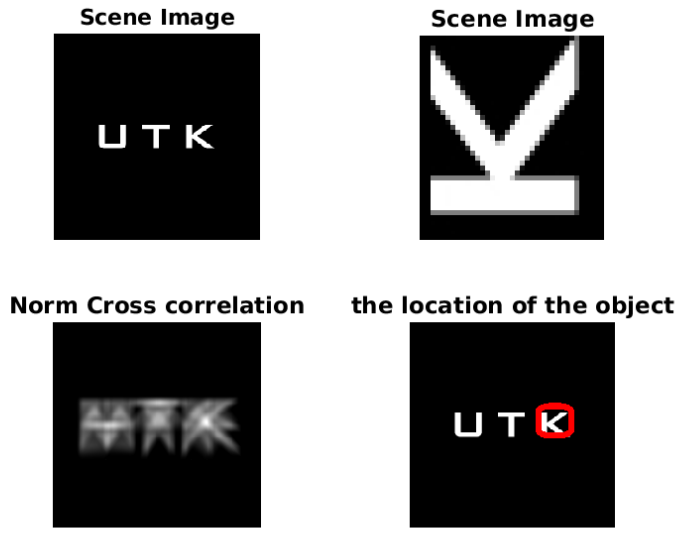

Figure 24) the output of the cross-correlation in the spatial domain with normalization and rotated object by 90 degrees.

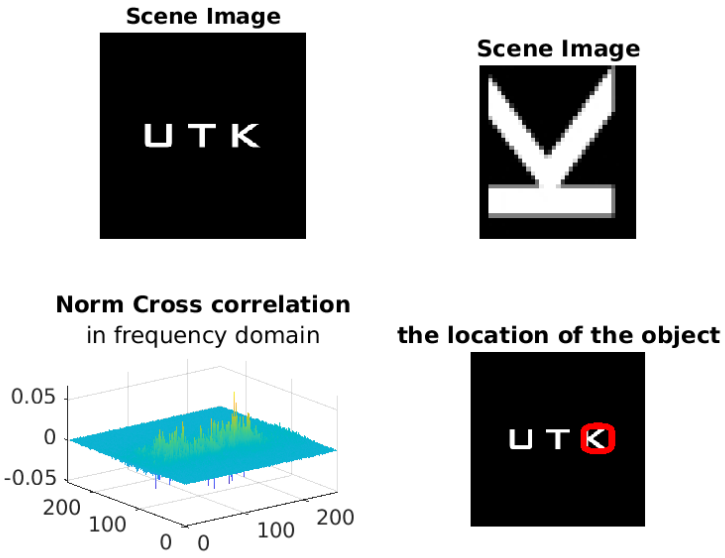

Figure 25) the output of the cross-correlation in the frequency domain with normalization and rotated object by 90 degrees.

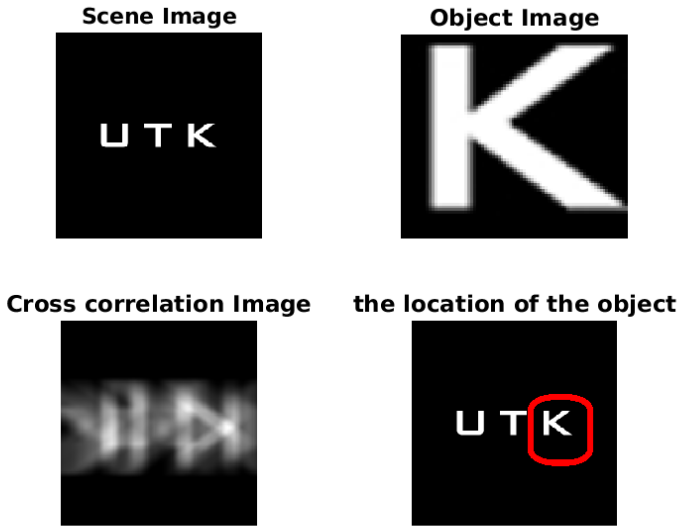

Figure 26) the output of the cross-correlation in the spatial domain without normalization and scaled object by 2.

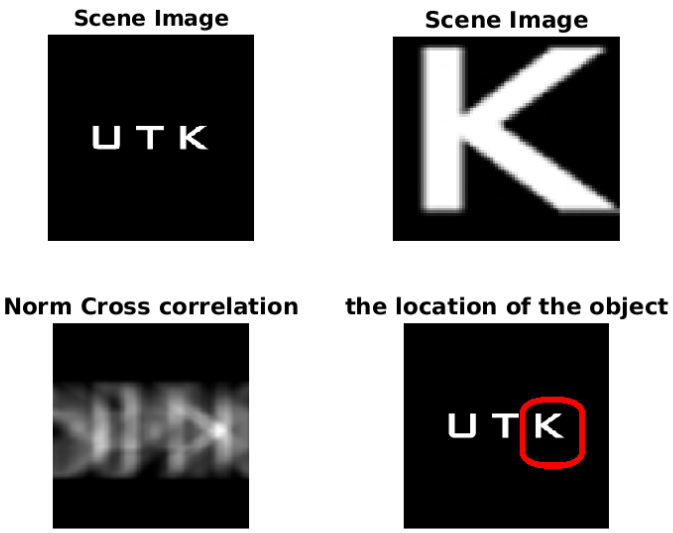

Figure 27) the output of the cross-correlation in the spatial domain with normalization and scaled object by 2.

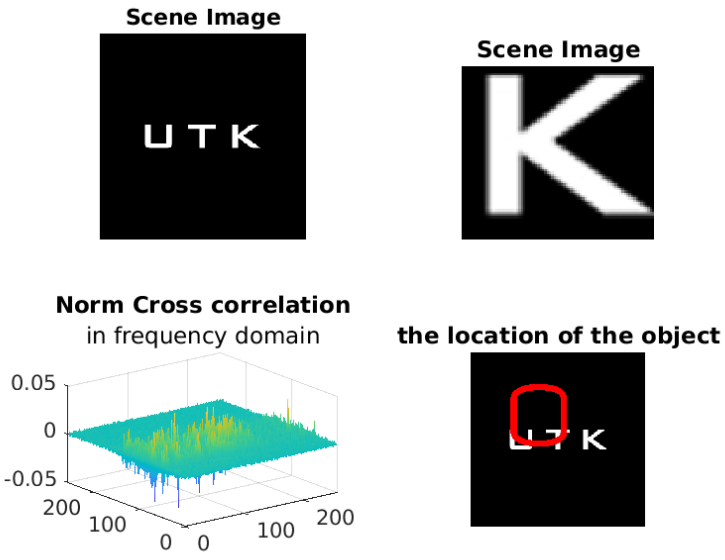

Figure 28) the output of the cross-correlation in the frequency domain with normalization and scaled object by 2.

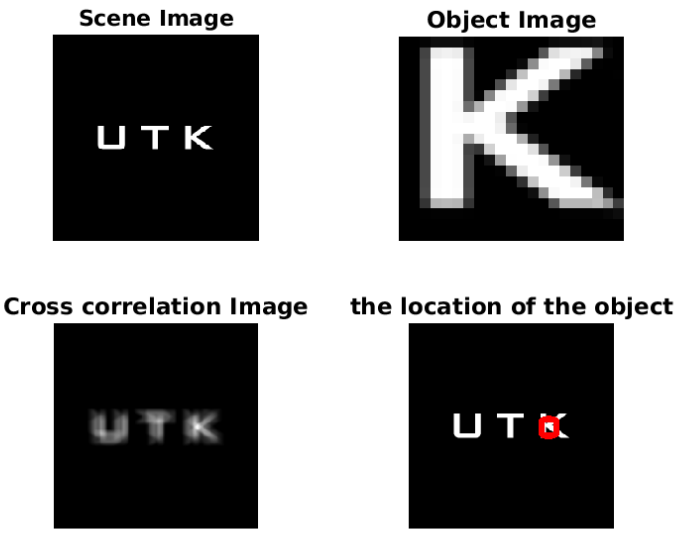

Figure 29) the output of the cross-correlation in the spatial domain without normalization and scaled object by 0.5.

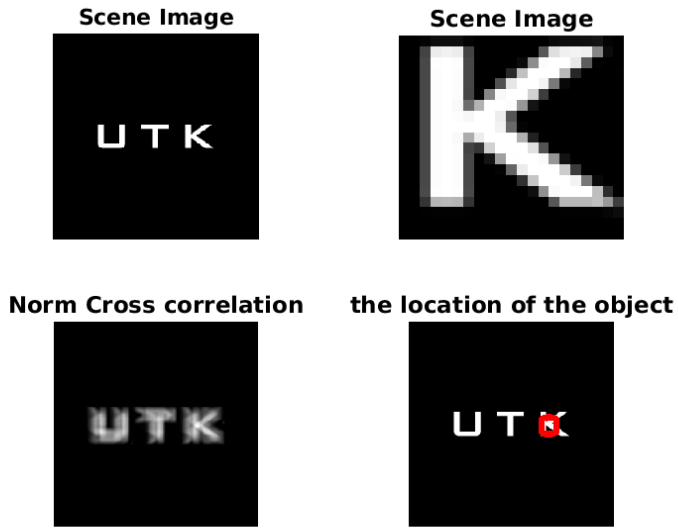

Figure 30) the output of the cross-correlation in the spatial domain with normalization and scaled object by 0.5.

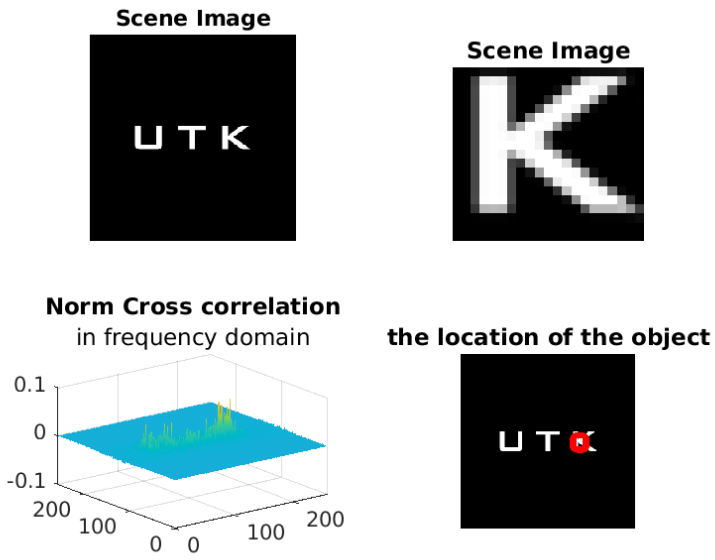

Figure 31) the output of the cross-correlation in the frequency domain with normalization and scaled object by 0.5.

Figure 32) the scene image for the correlation function

E. Appendix (codes)

I. M-file

```
% Saeed Kazemi
% Student Number 3713280


clearvars
close all
clc


%% ideal low pass filter
CutFrequency = 20;
output = zeros(100,100);
for i = 1:100
for j =1:100
D = ((i-50)^2 + (j-50)^2)^.5;
if D < CutFrequency
output(i,j) = 1;
end
end
end
mesh(output)
title('ideal low pass filter')
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/1-ideal low pass','png')
close(gcf)




%% Butterworth low pass filter
n = 2;
CutFrequency = 40;
filter = zeros(200,200);
[x, y] = size(filter);
for i = 1:x
for j =1:y
D = ((i-(x+1)/2)^2 + (j-(y+1)/2)^2)^.5;
filter(i,j)= 1/(1 + (D/CutFrequency)^(2*n));
end
end
mesh(filter)
title('Butterworth low pass filter')
subtitle('cut-off frequency = 40 & order = 2')
%legend('cut-off frequency = 40/n order = 2')
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/2-Butterworth low pass','png')
close(gcf)
```

```
%% 1D Butterworth low pass filter
CutFrequency = 50;
filter = zeros(1,100);
for i = 1 : 5 :20
for j = 1:100
filter(j)= 1/(1 + (j/CutFrequency)^(2*i));
end
hold on
plot(1:100,filter)
end
title('1D Butterworth low pass filter')
subtitle('cut-off frequency = 50')
legend("order 1", "order 6", "order 11", "order 16")


saveas(gcf, '/MATLAB Drive/EE6553/Lab2/3-1D Butterworth low pass','png')
close(gcf)
```

```
%% testing the low pass filter
I = double(rgb2gray( imread('highhouse.png')));
[x1, y1] = size(I);


I1 = zeros(x1, y1);
I1(156, 206) =35794094;
I1(356, 306) =35794094;



fftInput = fft2(I);
fftInput = I1 + fftshift(fftInput);


figure(1)
imshow(log(abs(fftInput)),[])


title('Fourier transformation of the noisy image')
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/4A-Fourier transformation of the
noisy image','png')
close(gcf)
```

```
filtered_image = ifft2(ifftshift(fftInput));
filtered_image = real(filtered_image(1:x1,1:y1));
OutputImage = uint8(filtered_image);
figure(2)
imshow(OutputImage,[])
title('The noisy image')
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/4B-The noisy image','png')
close(gcf)



figure(3)
OutputImage1 = Butterworth(OutputImage, 20, 2);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/4C-The output of Butterworth
filter','png')
close(gcf)


fftInput = fft2(OutputImage1);
fftInput = fftshift(fftInput);
figure(4)
imshow(log(abs(fftInput)),[])
title('Fourier transformation of the output of filter')
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/4D-Fourier transformation of the
output of filter','png')
close(gcf)




filtered_image = ifft2(ifftshift(I1));
filtered_image = real(filtered_image(1:x1,1:y1));
OutputImage = uint8(filtered_image);
figure(5)
imshow(OutputImage,[])
title('The noise image')
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/4E-The noise image','png')
close(gcf)



%% testing the low pass filter with delta function
I = zeros(101,101);
I(51,51) = 256;
```

```
OutputImage1 = Butterworth(I, 20, 2);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/5-applying the low pass filter on
the delta function','png')
close(gcf)
```

```
%% Cross correlation test 1
I = imread("image.jpg");
O = I(112:148,160:200);
```

```
% cross correlation
figure;
[OutputImage, ypeak, xpeak] = Cross_Corr(I, O);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/6A-applying cross
correlation','png')
close(gcf)
```

```
% Norm cross correlation
figure;
[OutputImage, ypeak, xpeak] = Norm_Cross_Corr(I, O);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/6B-applying Norm cross
correlation','png')
close(gcf)
```

```
% Norm cross correlation frequency domain
figure;
[OutputImage, ypeak, xpeak] = Norm_Cross_Corr_FrequencyDomain(I, O);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/6C-applying Norm cross correlation
in frequency domain','png')
close(gcf)
```

```
%% Cross correlation test 2
I1 = rgb2gray( imread("highhouse.png") );
O1 = I1(112:228,160:320);




% cross correlation
figure;
[OutputImage, ypeak, xpeak] = Cross_Corr(I1, O1);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/6A-applying cross
correlation','png')
close(gcf)




% Norm cross correlation
figure;
[OutputImage, ypeak, xpeak] = Norm_Cross_Corr(I1, O1);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/6B-applying Norm cross
correlation','png')
close(gcf)




% Norm cross correlation frequency domain
figure;
[OutputImage, ypeak, xpeak] = Norm_Cross_Corr_FrequencyDomain(I1, O1);
saveas(gcf, '/MATLAB Drive/EE6553/Lab2/6C-applying Norm cross correlation
in frequency domain','png')
close(gcf)


%% Cross correlation test 3
I = imread("image.jpg");
  O = I(112:148,160:200);


  O45 = imrotate(O, 45);
  O90 = imrotate(O, 90);
```

```
S02 = imresize(O, 2);
S01 = imresize(O, 0.5);


subplot(2,2,1)
imshow(O45,[])
title('rotate by 45')


subplot(2,2,2)
imshow(O90,[])
title('rotate by 90')


subplot(2,2,3)
imshow(S02,[])
title('scale by 2')


subplot(2,2,4)
imshow(S01,[])
title('scale by 0.5')
w{1} = O45;
w{2} = O90;
w{3} = S02;
w{4} = S01;


saveas(gcf, '/MATLAB Drive/EE6553/Lab2/7A-Test the correlation for the
rotation and scale','png')
close(gcf)


for i = 1 :4

    % cross correlation
    figure;
    [OutputImage, ypeak, xpeak] = Cross_Corr(I, w{i});
    saveas(gcf, append('/MATLAB Drive/EE6553/Lab2/7B-applying cross
correlation',num2str(i)),'png')
    close(gcf)

    % Norm cross correlation
    figure;
    [OutputImage, ypeak, xpeak] = Norm_Cross_Corr(I, w{i});
    saveas(gcf, append('/MATLAB Drive/EE6553/Lab2/7C-applying Norm
cross correlation',num2str(i)),'png')
    close(gcf)

    % Norm cross correlation frequency domain
    figure;
```

```
    [OutputImage, ypeak, xpeak] = Norm_Cross_Corr_FrequencyDomain(I,
w{i});
    saveas(gcf, append('/MATLAB Drive/EE6553/Lab2/7D-applying Norm
cross correlation in frequency domain',num2str(i)),'png')
    close(gcf)

end
```

II. Butterworth low pass filter

```
function OutputImage = Butterworth(InputImage,CutFrequency,
FilterOrder)


    I = (InputImage);

    [x, y] = size(I);
    if rem( x , 2) == 0 && rem( y , 2) == 0
        I = I(1:x-1,1:y-1);
    elseif rem( x , 2) == 0 && rem( y , 2) == 1
        I = I(1:x-1,1:y);
    elseif rem( x , 2) == 1 && rem( y , 2) == 0
        I = I(1:x,1:y-1);
    end

    [x, y] = size(I);

    fftInput = fft2(I);
    fftInput = fftshift(fftInput);
    subplot(2,2,1)
    imshow(I, []);
    title('Original Image')
    subplot(2,2,2)
    imshow(log(abs(fftInput)),[])
    title('Image in Fourier Domain')


    filter = ones(x,y);


    for i = 1:x
        for j =1:y
            D = ((i-(x+1)/2)^2 + (j-(y+1)/2)^2)^.5;


            filter(i,j)= 1/(1 + (D/CutFrequency)^(2*FilterOrder));


        end
    end
```

```
    subplot(2,2,3)
    imshow(log(abs(filter)),[])
    title('Filter Image in Freq Domain')

    filtered_image = filter.*fftInput;
    filtered_image = ifft2(ifftshift(filtered_image));
    filtered_image = real(filtered_image(1:x,1:y));
    OutputImage = uint8(filtered_image);
    subplot(2,2,4)
    imshow(OutputImage,[])
    title('Filtered Image')

end
```

III. Cross-Correlation Function

```
function [OutputImage, ypeak, xpeak] = Cross_Corr(InputImage, Object)


    I = double(InputImage);
    O = double(Object);



    subplot(2,2,1)
    imshow(I,[])
    title('Scene Image')

    subplot(2,2,2)
    imshow(O,[])
    title('Object Image')



    [xi, yi] = size(I);
    [xo, yo] = size(O);

    X = fix(xo/2)+1;
    Y = fix(yo/2)+1;

     OutputImage = zeros(xi-xo , yi-yo);



    for i=1:xi-xo
        for j=1:yi-yo
```

```
            II = I(i:i+xo-1,j:j+yo-1);
            OutputImage(i,j) = sum((II.*O) ,'all');
        end
    end

    subplot(2,2,3)
    imshow(OutputImage,[])
    title('Cross correlation Image')



    [ypeak, xpeak] = find(OutputImage == max(OutputImage(:)));

    subplot(2,2,4)
    imshow(I,[])
    title('the location of the object')


    rectangle('Position',[xpeak, ypeak, xo, yo],...
      'Curvature',[0.8,0.4],...
      'EdgeColor', 'r',...
      'LineWidth', 3,...
      'LineStyle','-')


end
```

IV. Norm Cross-Correlation Function

```
function [OutputImage, ypeak, xpeak] = Norm_Cross_Corr(InputImage,
Object)


    I = double(InputImage);
    O = double(Object);


    subplot(2,2,1)
    imshow(I,[])
    title('Scene Image')

    subplot(2,2,2)
    imshow(O,[])
    title('Scene Image')



    [xi, yi] = size(I);
    [xo, yo] = size(O);
```

```
X = fix(xo/2)+1;
Y = fix(yo/2)+1;
I1 = zeros(xi + xo, yi + yo);

OutputImage = zeros(xi - xo, yi - yo);


for i=1:xi-xo
    for j=1:yi-yo
        II = I(i:i+xo-1,j:j+yo-1);
        OutputImage(i,j) = sum((II.*O) ,'all')/...
          sqrt( sum(O.^2, 'all') * sum(II.^2, 'all') );
    end
end

subplot(2,2,3)
imshow(OutputImage,[])
title('Norm Cross correlation')


[ypeak, xpeak] = find(OutputImage == max(OutputImage(:)));


subplot(2,2,4)
imshow(I,[])
title('the location of the object')


rectangle('Position',[xpeak, ypeak, xo, yo],...
  'Curvature',[0.8,0.4],...
  'EdgeColor', 'r',...
  'LineWidth', 3,...
  'LineStyle','-')


end
```

V. Cross-Correlation in frequency domain Function

```
function [outputfft, ypeak, xpeak] =
Norm_Cross_Corr_FrequencyDomain(InputImage, Object)


    I = double(InputImage);
```

```
        O = double(Object);

        subplot(2,2,1)
        imshow(I,[])
        title('Scene Image')

        subplot(2,2,2)
        imshow(O,[])
        title('Scene Image')


        [xi, yi] = size(I);
        [xo, yo] = size(O);



        Sfft = fft2(I);
        Offt = fft2(O, xi, yi);
        outputfft = real(ifft2((Sfft.*conj(Offt))./abs(Sfft.*conj(Offt))));

        [ypeak, xpeak] = find(outputfft == max(outputfft(:)));


        subplot(2,2,3)
        surf(outputfft),
        shading interp;
        title('Norm Cross correlation')
        subtitle('in frequency domain')


        subplot(2,2,4)
        imshow(I,[])
        title('the location of the object')



        rectangle('Position',[xpeak, ypeak, xo, yo],...
          'Curvature',[0.8,0.4],...
          'EdgeColor', 'r',...
          'LineWidth', 3,...
          'LineStyle','-')

    end
```