MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 7. IMAGE PROCESSING, 2D FFT, FILTERING, EDGE DETECTION

I. Introduction

In this experiment, one of the topics of signal processing, image processing is considered. Image processing uses mathematical operations to process images, or video. In image processing, an image is considered as a two-dimensional signal where x and y coordinates correspond to the location of information. While this treats an image as a continuous signal, it can also be used to represent discrete digital images where x and y are the positive integer indices for pixel values. Hence image is treated as a grid of discrete elements. In this respect, signal processing theory can be applied directly to this two dimensional signal. However the increase in computational complexity prohibits some techniques especially for systems with limited computational resources. In a similar manner, a video can be modeled as a three dimensional signal where the third dimension is time.

The acquisition of images is called as imaging and requires image capturing devices and interfaces. Computer vision is a closely related field where high-level processing methods are used including segmentation, recognition and reconstruction. This leads to the scene analysis mimicking what the human brain does to understand the image.

In this experiment, an introduction to image processing is done including image filtering, edge detection, two-dimensional Fourier transformation, and inverse filtering, etc.

II. Preliminary Work

1) Read the following information on image processing.

1. Image Representation and Filtering

An image can be either a grayscale or color image. A grayscale image is simple to consider. It can be considered as a surface graph with x and y coordinates (or indices) represent the location and z axis represents the intensity of light. Brighter areas correspond to higher z-axis values. Grayscale image depth is the range of pixel values and expressed in terms of the number of bits each pixel value is represented ,i.e., 2^N. The higher is the depth, the larger is the memory required to store the image. 1024x768 8-bit grayscale image would require Memory=1024x768x8 bits.

Color images are represented using either Red-Green_blue (RGB) or Hue-Saturation_Luminance (HSL) models. When 8-bit RGB channels are considered, a color image would require Memory=1024x768x8x3 bits.

Image filters either suppress or enhance data with respect to a certain criterion. Examples of filter operations are edge detection (or highpass filtering), and smoothing (or lowpass filtering). Image filters can be linear or nonlinear. Linear filters can be implemented through convolution hence their 2D impulse responses are also called as convolution kernels. An example of such a filter is given in Fig. 1 where a smoothing (or lowpass filter) is considered. While Fig. 1 (a) and (b) are the same filters, the filter in (a) may lead to image clipping. In other words, when the value of a pixel and its closest neighbours are added, resulting value may exceed the image depth leading to clipping. The filter in Fig. 1 (b) does not have this problem and its values add up to one indicating that the clipping will not occur.

0	1	0
1	1	1
0	1	0

(a)

0	1/5	0
1/5	1/5	1/5
0	1/5	0

(b)

Fig. 1. Smoothing filters. (a) and (b) are the same filter except a scale factor. (b) does not have image clipping.

Image clipping can be handled during the filter implementation by dividing with the scalar value and hence usually the image filters are represented with their integer values as in Fig. 1 (a) for simplicity. In Fig. 1, 3x3 filter is presented. Larger filters can be used to improve the filter characteristics with the additional computational complexity.

2D convolution of a NxM image, f(n,m), and LxL convolution kernel h(n,m) can be expressed as

$$s(n,m) = f(n,m) * h(n,m) = \sum_{k=0}^{L-1}\sum_{p=0}^{L-1} h(k,p)\, f(n-k, m-p), \quad 0 \le n \le L+N-1,\ 0 \le m \le L+M-1$$

(1)

Convolution in time is multiplication in frequency and the above equation is written in frequency as,

$$S(w_1, w_2) = F(w_1, w_2)H(w_1, w_2), \qquad -\pi \le w_1 \le \pi, \quad -\pi \le w_2 \le \pi \qquad (2)$$

where $S(w_1,w_2)$ is the 2D DTFT of s(n,m). Note that w1 and w2 are continuous variables and cannot be implemented in digital systems. Hence 2D DFT should be used for practical applications. In the following part, the definition of 2D DFT is given.

1.1. 2D DFT

2D discrete Fourier transform is important for image processing applications. In practice, FFT algorithm is used to have the DFT coefficients since it is a computationally efficient algorithm. The 2D DFT of h(n,m) is also a discrete sequence given as,

$$H(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}h(n_1,n_2)e^{-j\frac{2\pi n_1 k_1}{N_1}}e^{-j\frac{2\pi n_2 k_2}{N_2}},\quad 0\le k_1\le N_1-1,\ 0\le k_2\le N_2-1 \qquad (3)$$

Inverse 2D DFT is written as,

$$h(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}H(k_1,k_2)e^{j\frac{2\pi n_1 k_1}{N_1}}e^{j\frac{2\pi n_2 k_2}{N_2}},\quad 0\le n_1\le N_1-1,\ 0\le n_2\le N_2-1 \qquad (4)$$

Note that DFT in (3) can be grouped in two summations and hence 2D DFT can be obtained by two 1D DFT's applied first to the columns and then to the rows. The same does apply for the use of 1D FFT in obtaining 2D FFT.

1.2. Convolution Kernels

Convolution kernel h(n,m) is said to be separable if it can be written as $h(n,m)=h_1(n)h_2(m)$. In this case, 2D convolution can be implemented as two 1D convolutions. First, rows of f(n,m) are convolved with $h_1(n)$ and then columns are convolved by $h_2(m)$. Same idea does apply to Fourier domain processing. Zeros of separable filters are easily found due to the Fundamental Theorem of Algebra which guarantees that there are n zeros of a single variable polynomial whose order is n. Unfortunately such a theorem does not exist for 2D nonseparable filters since they are polynomials with two variables. Therefore analysis of nonseparable convolution kernels is not trivial as in the case of 1D or separable filters.

1.2.1 Gaussian Filter

Gaussian filter can be used for noise removal. It is effectively a lowpass filter. The coefficients of this filter are given in Fig. 2.

1	2	1
2	4	2
1	2	1

Fig. 2. Gaussian kernel.

1.2.2. Gradient Filter

An interesting type of filter is the gradient filter. This filter is especially useful when intensity variations along a certain axis are enhanced. An example of this type of filter is given in Fig. 3 where the enhancement is being done in 45^{o} axis.

0	-1	-1
1	0	-1
1	1	0

Fig. 3. Gradient filter.

1.2.3. Laplacian Filter

Laplacian filter is an omnidirectional gradient filter. Hence it is a highpass filter. It is usually used for edge detection. There are different variants of Laplace filter. Fig. 4. (a) shows the general form a Laplace filter while Fig. 4. (b) is a special case used for edge detection.

A	B	C
D	X	D
C	B	A

(a)

-1	-1	-1
-1	8	-1
-1	-1	-1

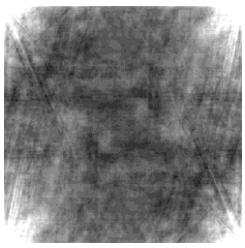
(b)

Fig. 4. Laplacian filter.

2. 2D Linear Phase Filters

A 2D image (or signal) can only be completely represented by using its DFT magnitude and phase together. In addition, it is also known that phase information is relatively more important than the magnitude information. In order to understand this, consider the image in Fig. 5. (a).

(a)



(b)

(c)

Fig. 5. (a) Original Image, (b) Image reconstructed by magnitude only, (c) Image reconstructed by phase only.

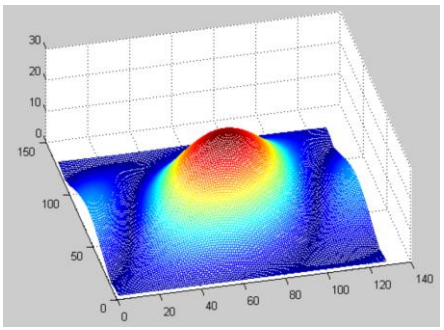
The original image 2D DFT is obtained and only the magnitude is extracted for reconstruction through 2D IDFT. The resulting image is shown in Fig. 5. (b). A similar operation is performed for the phase of 2D DFT by assuming the magnitude as one. Fig. 5. (c) shows the reconstructed image. As it is seen, phase information is more critical in image processing.
In order to preserve the phase information during 2D filtering, linear phase filter should be used. A 2D linear phase should satisfy the following condition, i.e.,

$$h(n_1,n_2) = h(N_1-1-n_1, N_2-1-n_2),\ \ 0 \le n_1 \le N_1-1, 0 \le n_2 \le N_2-1 \quad (5)$$

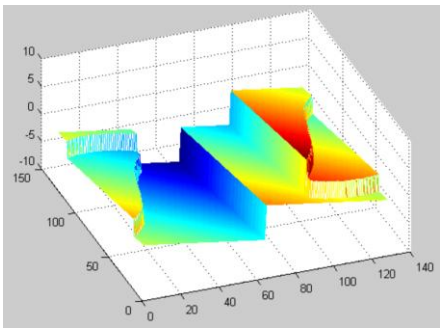
The following example is a 2D linear phase filter, with its magnitude and phase characteristics.

1	2	3
4	5	4
3	2	1

(a)



(b)



(c)

Fig. 6. (a) $h(n_1,n_2)$, (b) $|H(k_1,k_2)|$, (c) $\angle H(k_1,k_2)$.

3. Inverse Filtering

Inverse filtering or deconvolution is used to obtain the original image when its filtered version is given. Inverse filtering problem arises in different situations. A widely known case is about the Hubble Telescope. When Hubble telescope is stationed in Earth's orbit for transferring the star images, it is realized that the images are blurred due to malfunctioning lens. Hence the scientists should find a way to correct the filtering effect to obtain a sharper image. The solution was known for a long time namely the inverse filtering. In its simplest form, the filter that distorts the image is known. Unfortunately, it is unknown in many practical applications.
The inverse filtering problem is given in Fig. 7.

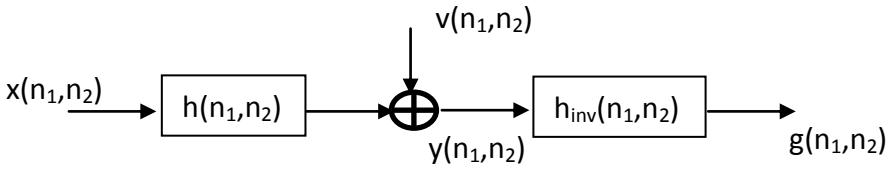


Fig. 7. Inverse filtering problem.

The input image $x(n_1,n_2)$ is filtered by $h(n_1,n_2)$ and corrupted by noise, $v(n_1,n_2)$. Given the output image, $y(n_1,n_2)$, the problem is to obtain an estimate of $x(n_1,n_2)$ by designing an inverse filter $h_{inv}(n_1,n_2)$. In the ideal case where $v(n_1,n_2)=0$ and 2D DFT $H(k_1,k_2)$ is invertible,

$$H_{inv}(k_1,k_2)=\frac{1}{H(k_1,k_2)} \qquad (6)$$

Note that H(k1,k2) is invertible if all of its poles and zeros are inside the unit circle (hence minimum-phase filter). Especially when there is noise and H(k1,k2) is small, there is noise amplification. It is possible to obtain optimum solution in MSE sense. The result is the Wiener inverse (or deconvolution) filter and it is given as,

$$H_{inv}^{Wiener}(k_1,k_2)=\frac{|H(k_1,k_2)|^2}{|H(k_1,k_2)|^2\,H(k_1,k_2)+\dfrac{1}{SNR(k_1,k_2)}} \qquad (7)$$

In (7), $SNR(k_1,k_2)$ is the signal-to-noise ratio for the k_1,k_2 frequency bins. Note that in the absence of this information, it should be estimated. Otherwise, a constant value for SNR results a suboptimum solution. Nevertheless $SNR(k_1,k_2)$=constant is a simple choice used in practice in order to have a rough idea about the performance of the inverse filtering.

2) Please include all the plots in your preliminary work!!!

a) Write the following MATLAB code to plot the image A and its 2FFT magnitude.

- **Comment on** the magnitude of 2FFT of the image A.
- Why do we scale AFFT in line 5?
- Why do we use ***fftshift*** to plot the magnitude of 2FFT?

```
1 –  A=ones(81,81);
2 –  figure
3 –  imshow(A);
4 –  AFFT=fft2(A);
5 –  AFFT=AFFT/max(max(abs(AFFT)));
6 –  figure
7 –  imshow(fftshift(abs(AFFT)));
```

b) Write the following MATLAB code to plot the image B and its 2FFT magnitude.

- **Comment on** the magnitude of 2FFT of the image B.

```
10 –  B=zeros(81,81);
11 –  B(41,41)=1;
12 –  figure
13 –  imshow(B);
14 –  BFFT=fft2(B);
15 –  BFFT=BFFT/max(max(abs(BFFT)));
16 –  figure
17 –  imshow(fftshift(abs(BFFT)));
```

c) Write the following MATLAB code to plot the image C and its 2FFT magnitude.

- **Comment on** the magnitude of 2FFT of the image C.

```
C=zeros(81,81);
C(35:45,:)=1;
figure
imshow(C);
CFFT=fft2(C);
CFFT=CFFT/max(max(abs(CFFT)));
figure
imshow(fftshift(abs(CFFT)));
```

d) Write the following MATLAB code to plot the image D and its 2FFT magnitude.

- **Comment on** the magnitude of 2FFT of the image D.

```
D=zeros(81,81);
for dd=0:8
    D(:,10*dd+1)=1;
end
figure
imshow(D);
DFFT=fft2(D);
DFFT=DFFT/max(max(abs(DFFT)));
figure
imshow(fftshift(abs(DFFT)));
```

e) Write the following MATLAB code to filter the image D with the filter F1.

- What is the type of the filter F1, i.e, lowpass or highpass?
- **Comment on** the effect of the filter F1 on the image D.

```
F1=[ 0 1/5 0; 1/5 1/5 1/5; 0 1/5 0];
DF=filter2(F1,D);
figure
imshow(DF)
```

f) Write the following MATLAB code to plot the image E and its filtered version EF.

- What is the type of the filter F2, i.e, lowpass or highpass?
- What's the special name of the filter F2?
- **Comment on** the effect of the filter F2 on the image E.

```
47 –    E=zeros(81,81);
48 –    E(15:25, 20:40)=1;
49 –    for ee=50:70
50 –      E(ee,60+50-ee:60-50+ee)=1;
51 –    end
52 –    figure
53 –    imshow(E);
54
55 –    F2=[ -1 -1 -1; -1  8  -1;  -1  -1  -1];
56 –    EF=filter2(F2,E);
57 –    figure
58 –    imshow(EF)
```

g) Write the following MATLAB code to plot the image G and its filtered version GF.

- What's the special name of the filter F3?
- **Comment on** the effect of the filter F3 on the image G.

```
61 –    G=rand(81,81);
62 –    figure
63 –    imshow(G)
64
65 –    F3=[ 0 -1 -1; 1  0  -1;  1  1  0];
66 –    GF=filter2(F3,G);
67 –    figure
68 –    imshow(GF)
```

PART 1

MATLAB Programming Tasks

a) Write a ***"expriment7.m"*** file to load two images, ***'cameraman.tif'*** and ***'lena.png'***. Use ***'imshow'*** to display the images.

b) Take the 2D Fourier transform of cameraman in order to visualize the low and high frequency components. Use ***'fftshift'*** to center the DC component.

c) Show that 2D fft, ***'fft2'***, is equivalent to applying ***'fft'*** to rows and columns respectively.

d) Obtain the phase components of ***cameraman***. Assume that the magnitude components are one and use ***ifft2*** to reconstruct the image by using only the phase information as shown in Fig. 5. ***Comment on*** your findings. Try alternatives for the magnitude terms during the reconstruction. One alternative is to use random numbers, i.e. ***rand()***. Another alternative is to use the magnitude of ***lena*** image. Which one gives ***better reconstruction***?

e) Consider the following ***9x9 nonlinear phase lowpass kernel***.

```
h1=1/4248*[92    35    75    96    22    39    70    30    81;
           82    40    38    34   100    49    36     3     3;
           16    66    60    71    57    88    80     8    21;
           82    37    15    93    81    90    98    17    91;
           82     9     4    41    52    92    68    34    82;
           49    37    56    54    84     6    62    98    50;
           82     9    28    88     9    36    73    83    40;
            2    58     2    40    48    38    54    79    19;
           80    77     6    54    58    50    35    60    84;];
```

Design a ***9x9 separable lowpass linear phase filter***, h2=ha'*hb. Let

```
ha=[1 2 3 4 5 4 3 2 1];
hb=[1 1 2 2 3 2 2 1 1];
```

Plot the ***magnitude*** and ***phase*** characteristics of these filters as shown in Fig. 6. Filter the ***cameraman*** and ***lena*** images by these filters and ***comment on*** the phase distortions introduced by the nonlinear phase filter. Note that you may need appropriate scaling for better visualization. You need to implement the filters in two ways. One is convolution in time. The other is the frequency domain implementation by using fft2. Use ***tic-toc*** commands to measure the computation time for each type of filtering.

f) Consider the filter, ***h2***, in ***e)***. Filter the ***lena*** image with this filter and obtain the output ***$y(n_1,n_2)$*** as shown in Fig. 7 by adding random noise with a known variance. Display the resulting image with ***imshow*** command. Design the inverse filter as it is given in equation (7). You should use an appropriate FFT size to obtain linear convolution. Filter the output, $y(n_1,n_2)$ with the inverse filter. Display the result of the inverse filtering and ***comment on*** the deconvolution performance. Try different noise levels and comment on the reconstruction performance. ***Determine*** the least squares error in reconstruction. Try different images and comment on their reconstruction quality. Change the h2 filter (by changing ha and hb) and ***comment on*** the inverse filtering quality based on the filter characteristics. What is the effect of zeros of ***ha*** and ***hb*** on the reconstruction quality?

PART 2

Real-time Programming Tasks

Part A:

In this part, you can use ***Edge Detection with 2D Convolution.vi*** which you can find in ***"Find Examples"*** item in the Help Menu. Copy this VI and modify its contents accordingly to implement the operations described in the MATLAB programming tasks.

The project structure in Experiment 7 is given in Fig. 8. Note that you will implement your routines in myRIO for Part A whereas Part B is implemented in PC.

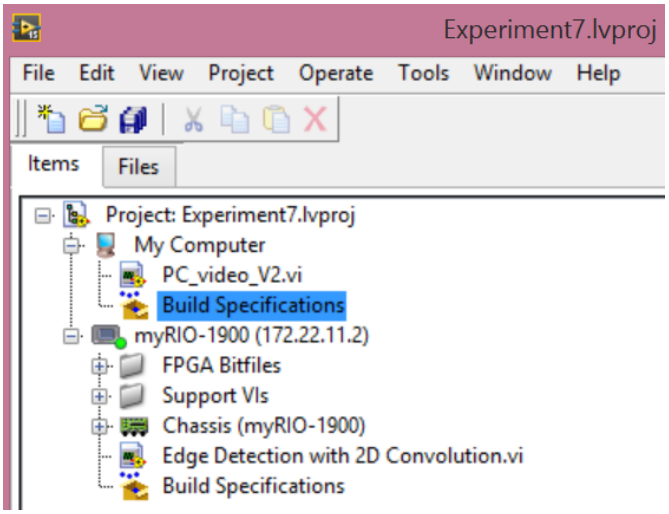


Fig. 8. Project structure for Experiment 7.

A part of the front panel of Experiment 7 is given in Fig. 9. Note that FFT VI in LabVIEW has an input which you can set as True to obtain ***"fftshift"*** effect in MATLAB. Also the front panel does not include certain parts that you need to implement and hence given only for guidance.

a) Modify the contents of Edge Detection with 2D Convolution.vi such that you have a front panel similar to Fig. 9. A part of the Block Diagram after modification looks like as shown in Fig. 10.

b) Obtain the 2D Fourier transform of ***cameraman*** in order to visualize the low and high frequency components. You can try other image sources such as ***"Baboon"***, ***"Colombia"*** and ***"Couple"*** in order to see the differences between frequency contents. **Comment on** each image and its frequency characteristics.

c) Obtain the phase components of cameraman. Assume that the magnitude components are one and use 2D ifft to reconstruct the image by using only the phase information as shown in Fig. 5. Comment on your findings. Try alternatives for the magnitude terms during the reconstruction. Can you obtain better reconstruction quality with alternative selections?

d) Increase the size of the Convolution Mask in Fig. 9 to 9x9. You can easily do that by enlarging the box with mouse and entering numerical values. Repeat ***part e)*** in ***MATLAB*** Tasks. In this case you only need to implement the filtering using ***"convolution.vi"***.

e) Repeat ***part f)*** in ***MATLAB*** Tasks. In this case, you can use the available image sources instead of ***Lena***. Try different images and comment on their reconstruction quality. Change the h2 filter and comment on the inverse filtering quality based on the filter characteristics.

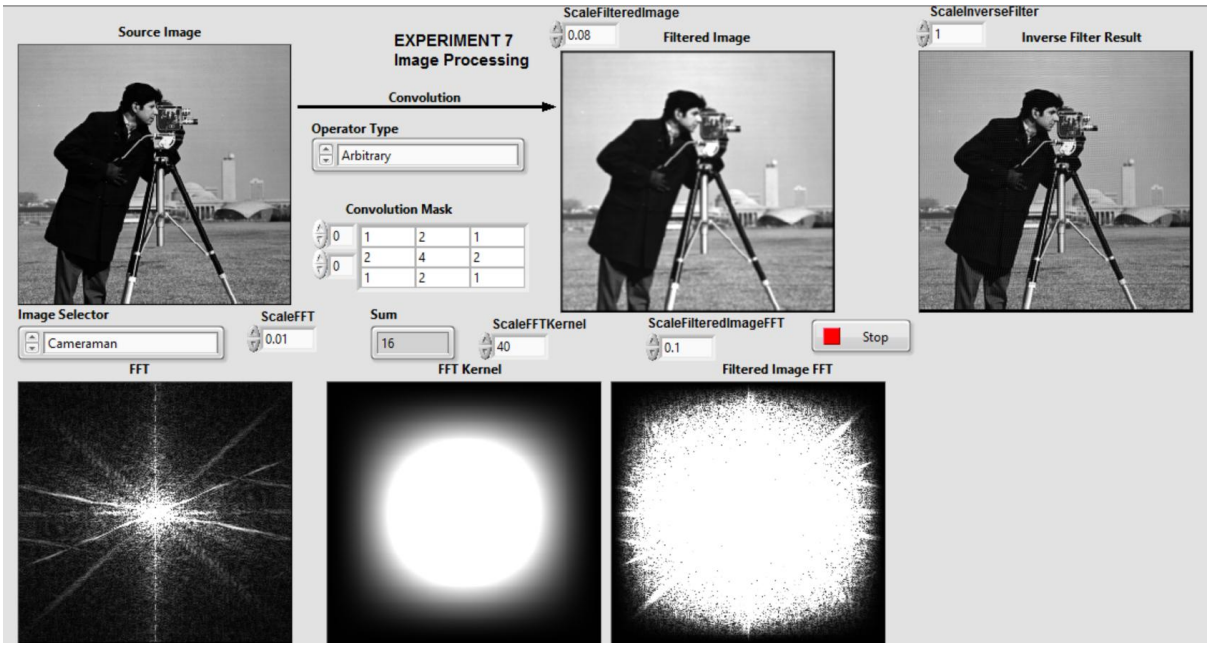


Fig. 9. Front panel of Experiment 7.

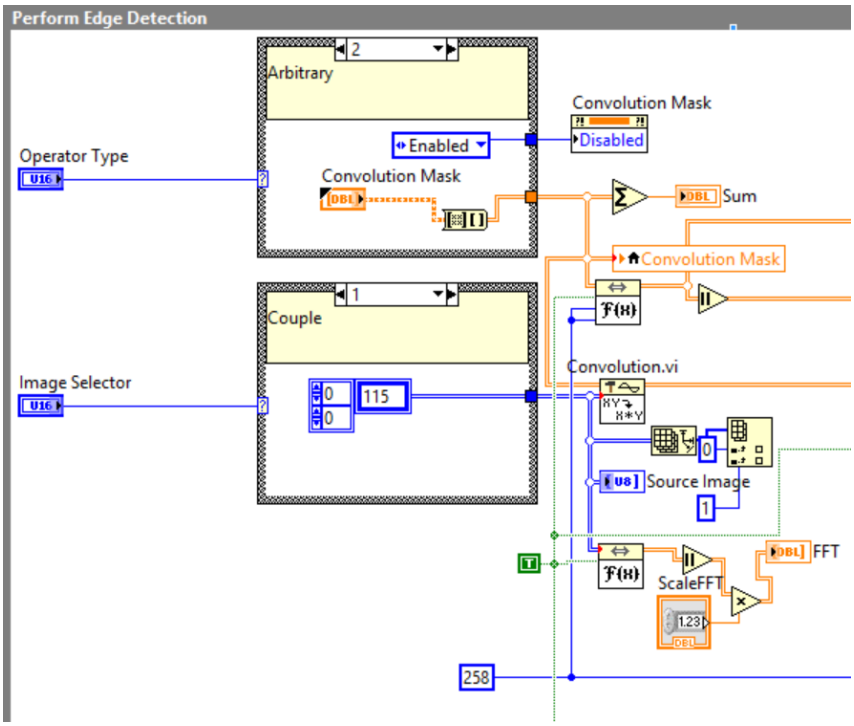


Fig. 10. Part of the Block Diagram in Experiment 7 Part A.

Part B:

In this part, you will implement a video processing task in real-time using the computer resources rather than myRIO. Pattern recognition is a high level processing and can be implemented in LabVIEW using built-in functions. In this part, you need to capture the video from the USB camera and display it in your front panel. Next, the image sequence is searched for a specific pattern that you defined before. Once this pattern is found, it is identified with a red rectangle encircling the object and its position is displayed on a x-y coordinate system which is plotted by using mathscript.

- First, connect your webcam to the computer.

- Construct the block diagram in Fig. 11. Note that ***"Session In"*** is the control of **IMAQdx Open Camera.vi** and ***"im1"*** and ***"im2"*** are constants.

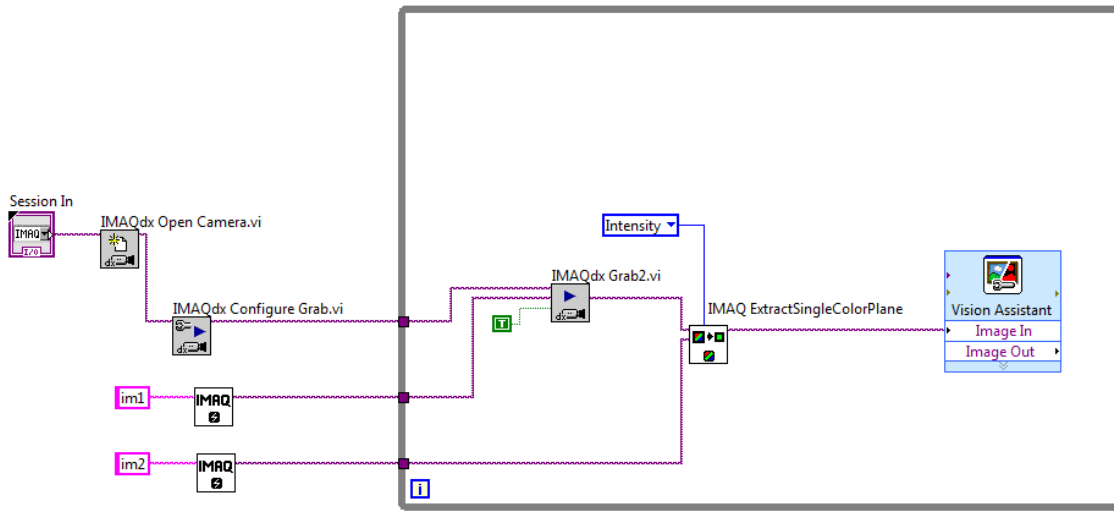


Fig. 11. First Part of the Block Diagram for the Experiment 7, Part B.

- As it is seen in Fig. 11, you first identify the input using ***Session Input***. Then ***IMAQdx Open Camera*** and ***Configure Grab.vi*** are used to configure the system to obtain a frame from the video. Inside the while loop, a frame is extracted from the video sequence through ***IMAQdx Grab2*** function. Then this image is fed to ***IMAQ ExtractSingleColorPlane*** function to obtain only one color or in our case intensity image. The output is the input to the Vision Assistant where you configure it for pattern recognition task.

- On a A4 paper plot a dark circle and obtain an image of it through the camera by running the VI. When you run the VI, an error window will appear as shown in Fig. 12. Click **Continue** and after a while, stop the VI.

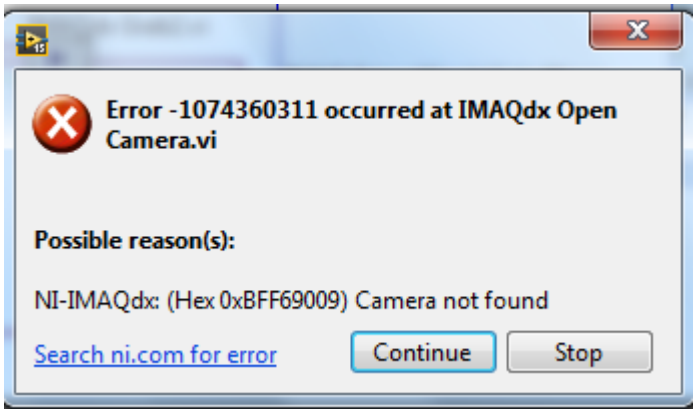


Fig. 12. Error Window.

- Now, you are ready to configure pattern matching function. You acquired an image that constitutes your template. Double click the Vision Assistant you can access to several high level functions one of them being the “PatternMatching” as shown in Fig. 13.

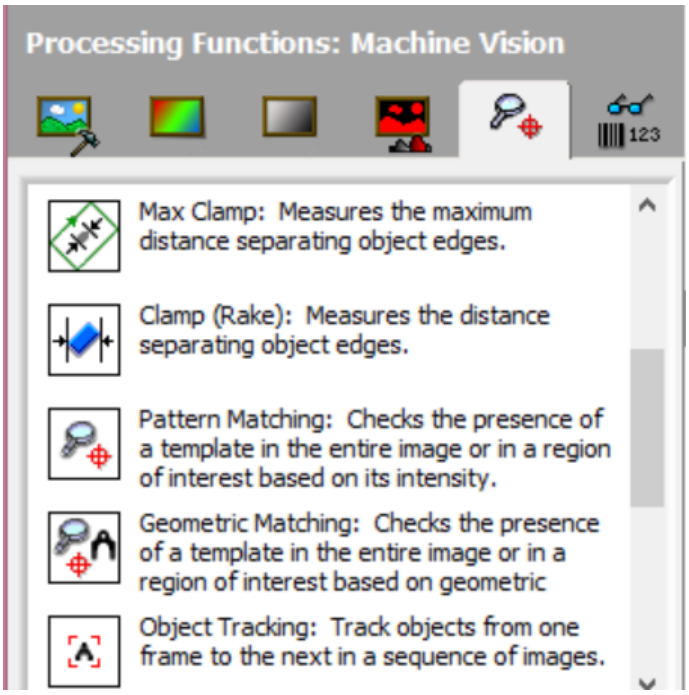


Fig. 13. Pattern Matching function in Vision Assistant.

- Click on Pattern Matching shown in Fig. 13.

- Configure the parameters as shown in Fig. 14 by clicking New Template and adjusting the circle in the middle of picture. The circle should be sufficiently large as in Fig. 14.

- Click Finish and save the image file inside your project directory.

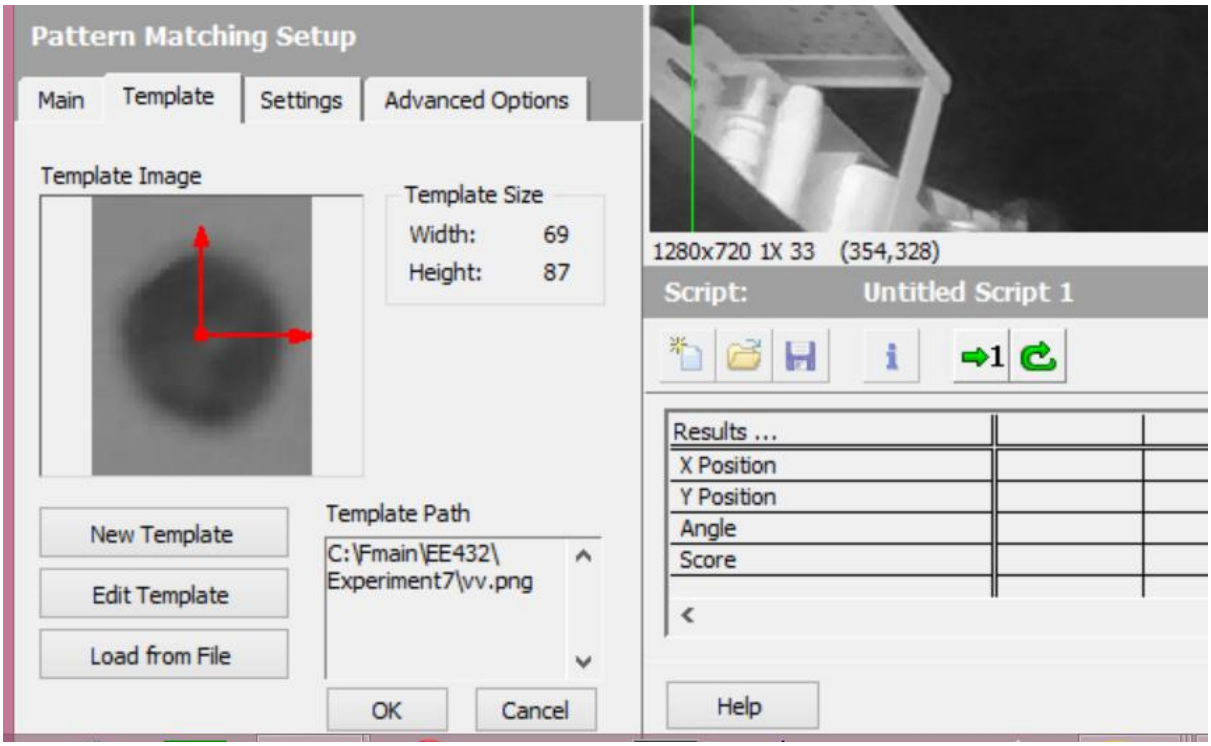


Fig. 14. Configuring the Pattern Matching template.

- Click OK under Template Path in Fig. 14. Click ***Select Controls*** shown in Fig. 15.

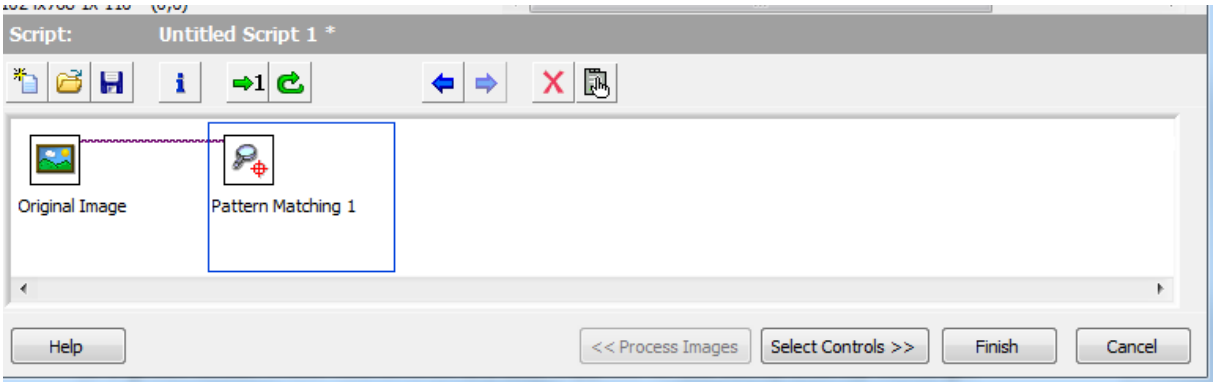


Fig. 15. Configuring the Pattern Matching template.

- Select ***Matches*** under ***Indicators→Pattern Matching 1*** and click **Finish**.
- Complete the block diagram as shown in Fig. 16.

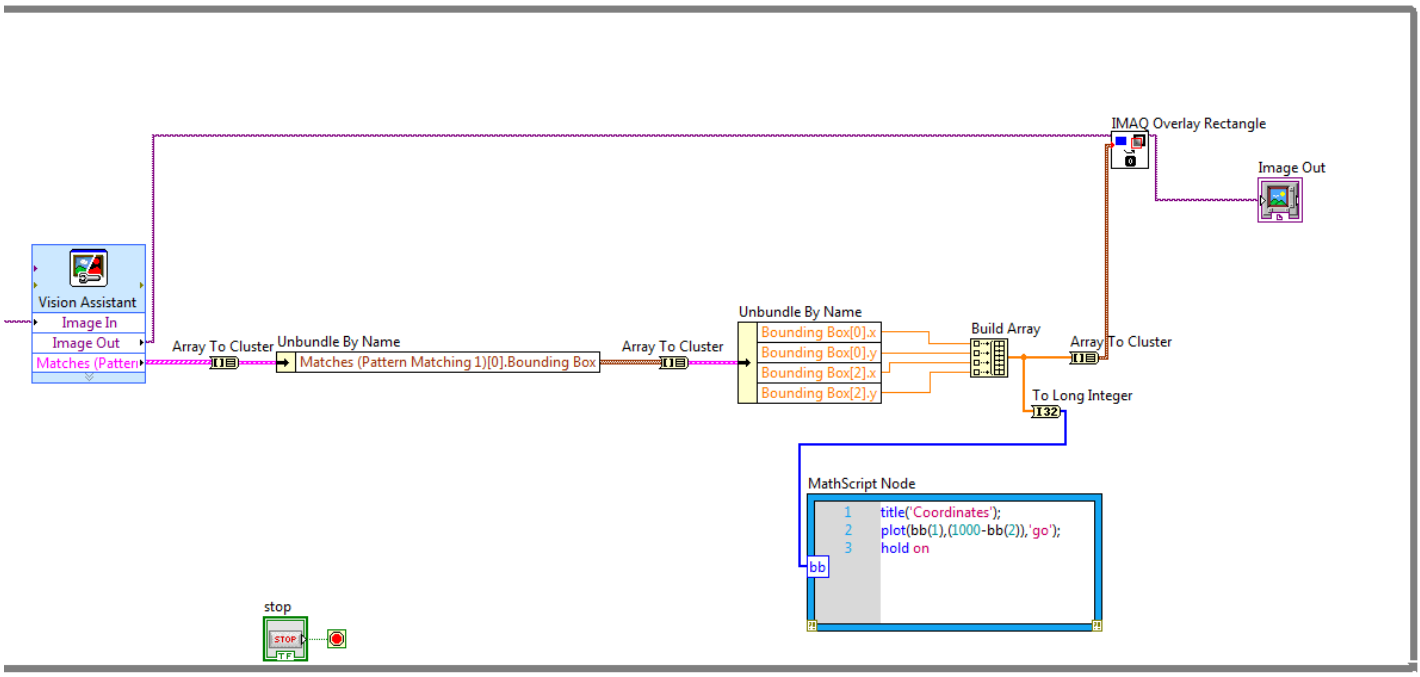


Fig. 16. Second Part of the Block Diagram for the Experiment 7, Part B.

- The output of the Vision Assistant is fed to **Array to Cluster** module and **Unbundle by name** to identify the unit for recognition. Then the x-y coordinates of the upper left and lower right corners of the bounding box for the pattern are identified. These coordinates are given as input for the ***MathScript*** where the position of the matched object is plotted as a green circle. These coordinates are also supplied to the ***IMAQ Overlay Rectangle*** function to plot a rectangle encircling the object. Note that the LabVIEW function can operate satisfactorily only for simple and small objects due to computational complexity and simplicity of the recognizing algorithm.

- Test your program and determine if it follows the circle by plotting the red rectangle around it.