EEE- 391: Basics of Signals and Systems
MATLAB Assignment 2

Assigned: 18.11.2022

Deadline: 23.12.2022 23:59

Along with this pdf file, you will find a .zip file containing two Matlab functions andseveral pictures. Unzip the archive and place all files contained in it under the currentdirectory of Matlab.

1 Part 1

In this assignment, you are going to do a bit of image processing. You will perform some simple operations on gray scale (white and black) digital images and see the results.

In this course up to now, we have mainly dealt with one-dimensional signals such as $x(t)$ or $x[n]$. These signals are said to be one-dimensional because they are functions of one independent variable (t or n).

Images, on the other hand are signals of two independent variables. Therefore they are two-dimensional signals.

A gray scale digital image can be represented with a 2D discrete function $x[m,n]$ such that $x[m,n]$ denotes the light intensity of the (m,n)th pixel of the image (m and n are integers). Since practical images are of finite size (such as 512×512, 1024×1024 etc.), we usually have $x[m,n]=0$ for $m\notin[0,M_x-1]$ or $n\notin[0,N_x-1]$ where the image size is $M_x\times N_x$.

Recall that we store 1D signals in 1D arrays in Matlab. Since images are 2D signals, they are stored in matrices in Matlab. An $M_x\times N_x$ image is stored in an $M_x\times N_x$ matrix in Matlab.

In this first part, you will just practice reading images in Matlab and displaying them. **You will not write any code for this part but use the m-files provided for this assignment.**

Place the m-files named **ReadMyImage.m** and **DisplayMyImage.m** and the image named **Part5.bmp** under the current directory of Matlab.

Issue the command

A=ReadMyImage('Part5.bmp');

You will see that a 532×800 matrix of type double will be created and stored in the workspace. This matrix contains the image. The original image is a color image, but it is converted to a gray scale image by the **ReadMyImage** function.

Next, issue the command

DisplayMyImage(A);

A new figure window will open and the image will be displayed.

During the rest of this assignment, you will use these commands to read and display various images.

For this part, just make sure that you run the functions above and see the image without any problem. You do not need to provide anything for the report.

2 Part 2

Recall that 1D discrete time (DT) systems map a 1D input signal $x[n]$ to another 1D signal $y[n]$ (that we name the output signal). (We say discrete **time** since the independent variable usually denotes time.)

Similarly, a 2D discrete space (DS) system maps a 2D input signal $x[m,n]$ to a 2D output signal $y[m,n]$. (Now we say discrete **space** since the independent variables usually denote the space coordinates.) Recall that a 1D DT system is called linear **time** invariant (LTI) if it satisfies the following two properties:

- $\alpha_1 x_1[n] + \alpha_2 x_2[n]$ produces $\alpha_1 y_1[n] + \alpha_2 y_2[n]$ for all $x_1[n]$, $x_2[n]$, α_1, α_2.
- $x[n-n_0]$ produces $y[n-n_0]$ for all $x[n]$ and n_0.

Similarly, a 2D DS system is called linear **space** invariant (LSI) if it satisfies:

- $\alpha_1 x_1[m,n] + \alpha_2 x_2[m,n]$ produces $\alpha_1 y_1[m,n] + \alpha_2 y_2[m,n]$ for all $x_1[m,n]$, $x_2[m,n]$, α_1, α_2.
- $x[m-m_0, n-n_0]$ produces $y[m-m_0, n-n_0]$ for all $x[m,n]$ and (m_0, n_0).

Now let us develop the input output representation of 2D DS LSI systems by forming an analogy with 1D DT LTI systems.

Recall that a 1D discrete impulse signal is defined as

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

While developing the input-output relation for a 1D DT LTI system, we first wrote the input signal $x[n]$ as a superposition of shifted impulse signals as:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{1}$$

where we interpreted $x[k]$ as the coefficient of the impulse shifted by k units (that is, $x[k]$ is the coefficient of $\delta[n-k]$). Then, we named the response that the system gives to $\delta[n]$ as $h[n]$ (impulse response). Using the time invariance property of the system, we recognized that the response of the system to $\delta[n-k]$ should be $h[n-k]$. Then, using the linearity property of the system, we wrote the input-output relation of the 1D DT LTI system as:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{2}$$

We named the above operation as the convolution of $x[n]$ and $h[n]$ and used the short hand notation

$$y[n] = x[n] * h[n] \tag{3}$$

to denote it.

Now, define the two dimensional discrete impulse signal as

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

and going through the same steps, **derive** the input-output relation of a 2D DS LSI system, or derive the formula for 2D convolution of $x[m,n]$ and $h[m,n]$ that we denote as: $x[m,n] * * h[m,n]$. In other words, derive the 2D version of Eq. 2. Show—explain your steps as is done above for the 1D case. Include your work to your report. You should obtain the following result:

$$\begin{aligned} y[m,n] &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l]h[m-k,n-l] \\ &= x[m,n] * * h[m,n] \\ &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h[k,l]x[m-k,n-l] \end{aligned} \tag{4}$$

3 Part 3

Recall that a 1D DT LTI system is called FIR if the impulse response $h[n]$ contains a finite number of nonzero values such that $h[n] = 0$ for $n \notin [0, M_h - 1]$ where $M_h \in \mathcal{Z}^+$. Similarly, a 2D DS LSI system is called FIR if the impulse response $h[m,n]$ contains a finite number of nonzero values such that $h[m,n] = 0$ for $m \notin [0, M_h - 1]$ and $n \notin [0, N_h - 1]$ where $M_h, N_h \in \mathcal{Z}^+$.

In this part, you will write a Matlab function that computes the output when a finite sized input image $x[m,n]$ (of size $M_x \times N_x$) is input to a 2D FIR DS LSI system whose impulse response $h[m,n]$ is of size $M_h \times N_h$. From another perspective, your code will compute the 2D convolution of two finite-size 2D signals $x[m,n]$ and $h[m,n]$. We assume that

- $x[m,n]$ can only be nonzero within $0 \le m \le M_x - 1$ and $0 \le n \le N_x - 1$.
- $h[m,n]$ can only be nonzero within $0 \le m \le M_h - 1$ and $0 \le n \le N_h - 1$.

Under these conditions Eq. 4 reduces to:

$$y[m,n] = \sum_{k=0}^{M_h-1} \sum_{l=0}^{N_h-1} h[k,l]x[m-k,n-l] \tag{5}$$

Based on the above equation, show that $y[m,n]$ can only be nonzero within $0 \le m \le M_y - 1$ and $0 \le n \le N_y - 1$. Determine M_y and N_y in terms of M_x, N_x, M_h and N_h. Include your work to your report.

Next, write a Matlab function of the following form

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

- **h** of size $M_h \times N_h$ denotes the impulse response of the system, such that **h**(1,1)=$h[0,0]$, **h**(1,2)=$h[0,1]$, ..., **h**(k,l)=$h[k-1,l-1]$.
- **x** of size $M_x \times N_x$ denotes the input signal $x[m,n]$, such that **x**(1,1)=$x[0,0]$, **x**(1,2)=$x[0,1]$, ..., **x**(k,l)=$x[k-1,l-1]$.
- **y** of size $M_y \times N_y$ denotes the output signal, such that **y**(1,1)=$y[0,0]$, **y**(1,2)=$y[0,1]$, ..., **x**(k,l)=$y[k-1,l-1]$.

Basically, you will implement Eq. 5. You should first create a zero matrix of size $M_y \times N_y$ for **y**. Then, you can use a nested for loop as in the following code:

```
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
```

Note: Do not use any built in command of Matlab. Directly implement Eq. 5.

Note: Before writing your code, carry out the mini exercise below. You do not need to provide anything for the report, but make sure that you fully understand the interpretation of the following commands.

- Type **A=zeros(3,3);**. Then type **A(1:2,1)=[1;2];**. Examine **A**.
- Now type **A(2:3,2:3)=[5 8;6 9];**. Examine **A**.
- Now type **A(3:3,1:1)=[3];**. Examine **A**.
- Now type **A(1:1,2:3)=[4 7];**. Examine **A**.
- Let **B=A(1:2,2:3);**. Examine **B**.
- Let **B=A(3:3,1:2);**. Examine **B**.

Check your function: If you take

$$\mathrm{x} = \begin{bmatrix} 2 & 1 & 1 \\ -3 & 0 & 2 \\ 1 & -1 & 2 \end{bmatrix}$$

and

$$\mathrm{h} = \begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix}$$

you should get

$$y = \begin{bmatrix} 4 & 4 & 3 & 1 \\ -8 & -4 & 3 & 2 \\ 5 & -1 & 1 & 2 \\ -1 & 1 & -2 & 0 \end{bmatrix}$$

Include your code to your report.

4 Part 4 - Adding Noise and Image Denoising

Now it is time to see some practical image processing examples.

Read the picture named **Part4.bmp** in a matrix named **x** and display it. In this part, first of all we add noise to the picture and we will try to rescue this image from the noise without disturbing the image itself as much as possible.

For the first step, write a simple MATLAB code to add Gaussian noise with different mean and variances as given in the table below, and then comment about the differences you see in the picture.

In order to add noise you can use the following command:

random('norm', mean , std ,size)

if you need further information about this command, look for it in MATLAB help.

Table 1: Noise parameter

σ	μ
0.1	0
0.25	0
0.5	0

Since we know that the noise on the picture has high frequency content (just as many other types of noise that we encounter in signal processing). We also know that typi-cal daily life images taken by a typical camera have low-frequency content. Therefore, why should not we apply a low pass filter to the noisy image and try to eliminate the noise? A typically used 2D FIR low pass filter has the following impulse response:

$$h[m,n] = \begin{cases} 0 & \text{if } m<0 \text{ or } n<0 \text{ or } m>M_h-1 \text{ or } n>N_h-1 \\ \text{sinc}\left\{B\left(m-\frac{M_h-1}{2}\right)\right\}\text{sinc}\left\{B\left(n-\frac{N_h-1}{2}\right)\right\} & \text{otherwise} \end{cases}$$

where B is a free parameter that determines the bandwidth of the filter. We should select B between 0 and 1.

Let D denote your ID number, and let D_{17} denote your ID number in modulo 17. That is

$$D \equiv D_{17} \mod 17$$

with $0 \leq D_{17} \leq 16$. You can compute D_{17} using the **rem** command of Matlab. To learn how **rem** works, type **help rem** in Matlab command window.

Now, let $M_h = N_h = 20 + D_{17}$ and $B = 0.5$. Prepare the **h** matrix that represents $h[m, n]$. If you wish, you can use nested for loops over m and n, this time it is allowed. Recall that Matlab already has a built in function named **sinc** to compute the values of sinc(.) function. Include your code for preparing **h** to your report.

Using the code that you developed in Part 3, process the noisy image with noise standard deviation equal to 0.5 using this filter. Display the output image. Repeat the exercise with $B = 0.2$ and $B = 0.05$ as well. Include all the output images to your report. Use **subplot** command, and show the images on the same Matlab figure.

Comment on your results. Which value of B seems to be the most appropriate one? Include your comments to your report.

As you see, it is possible to greatly clarify the information in a corrupted signal using only the simple concepts of convolution, LSI systems, etc. In this part, we tried to remove the noise from a corrupted image. Such problems are called **image denoising** problems. Many algorithms have been developed by many researchers that try to clear the images from the corrupting noise while giving minimum damage to the original image.

5 Part 5 - Edge Detection

Another widely studied interesting problem of image processing is the detection of edges in an image, which is called the **edge detection** problem. In a typical image, edges are the set of points in the close vicinity of which the pixel values change abruptly. Since changes are sudden and great, edges are inherently associated with high frequencies. Therefore, we can make use of high pass filters to detect edges.

Read the picture named **Part5.bmp** in a matrix named **x** and display it.

Consider a 2D FIR DS LSI system whose impulse response h_1 is equal to:

$$h_1 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Prepare $h_1[m,n]$ and process your image with this filter. Let $y_1[m,n]$ denote the resulting image. Display the image which is defined as $s_1[m, n] = y_1^2[m, n]$. Include this image to your report. Which parts of the original image are emphasized? Include your answer to your report.

Now let

$$h_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Again, prepare $h_2[m, n]$ and process your image with this filter. Let $y_2[m, n]$ denote the resulting image. Display the image which is defined as $s_2[m, n] = y_2^2[m, n]$. Include this image to your report. Which parts of the original image are emphasized now? Comment on the difference with the image you obtained with $h_1[m, n]$. Include your answer and comments to your report.

Finally, Display the image which is defined as $s_3[m,n] = \sqrt{y_1^2[m,n] + y_2^2[m,n]}$. Include this image to your report. Which parts of the original image are emphasized now? Comment on the difference with the images you obtained with $h_1[m,n]$ and $h_2[m,n]$. Include your answer and comments to your report.

6 Part 6 - A sample pattern recognition application

A main problem of image (or signal) processing is the detection of certain objects within a picture.

Read and display the image named **Part6x.bmp**. You will see our national soccer team. Our purpose in this part is to detect the faces in the image. Such problems are named **pattern recognition** problems. Again, many algorithms have been developed to solve such problems. Here, we will use one of the basic methods which is called **matching filter** method. Read the image **Part6h.bmp** into Matlab and think of it as the impulse response of a 2D DS LSI FIR system. Display the image to see how the impulse response looks like. You will see an inverted face. That is, to detect the faces within the image, we are using a 2D DS LSI system whose impulse response is an inverted face! Note that by inverted, we mean that the face is rotated by 180 degrees. (When the rotation is 180 degrees, its direction - clockwise or counterclockwise - is unimportant.)

Now, pass the input image through this system and find the output image. Display the absolute value of the output image, that is, display $|y[m, n]|$. Include this image to your report. Search for the points that look bright. Where do they occur? Do you always see a face at the location that you think there is a bright point, or do you sometimes find bright points where there is no face? Include your answers to your report.

To make the bright points more visible, compute the image $|y\,[m, n]|^3$ and display it. Include this image to your report. Also compute the image $|y\,[m, n]|^5$ and display it. Include this image to your report. Taking which power is sufficient for you to detect the faces without any confusion? Include your answer to your report.

Comment on the success of the method. Include your comments to the report.