

Multidimensional Signal Processing ETD003

Laboratory Experiment 1: Convolutions and Fourier Transforms

1 Introduction

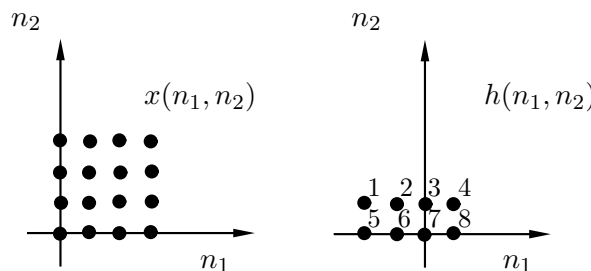
The purpose of this laboratory experiment is to give examples in basic elements of image processing such as convolutions and fourier transforms. We emphasize on the two-dimensional frequency concept, and its interpretation in image processing. The **purpose** of the experiment is also to give an **introduction** to the *Image Processing Toolbox* in *Matlab*, which will be used in the following experiments and in the project assignment.

Several of the tasks in the experiment are based on the pre-assignments below, which must be solved prior to the time of the experiment.

2 Pre-assignments

Pre-assignment 1: The Convolution

Calculate the two-dimensional convolution $y(n_1, n_2) = x(n_1, n_2) * h(n_1, n_2)$ where $x(n_1, n_2)$ is the input signal (e.g. an image) and $h(n_1, n_2)$ is the impulse response according to the figure below. The input signal consists of 16 ones, whereas the values of the impulse response are given in the figure. Calculate the **specific** part of the output signal that has the **same** support region as the input signal.



Pre-assignment 2: The Fourier Transform

a) The two-dimensional discrete Fourier Transform is given by equation (3.7a) and (3.7b) in Lim. Let \mathbf{x} and \mathbf{X} be a **signal** matrix (e.g. an image) and a **transformation** matrix respectively,

¹This laboratory experiment tutorial has been modified by Jan-Olof Gustavsson 1999, Benny Lövfström 2001 and Benny Sällberg 2003

according to:

$$\mathbf{x} = \begin{pmatrix} x(0,0) & x(0,1) & \cdots & x(0,N_2-1) \\ x(1,0) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x(N_1-1,0) & \cdots & \cdots & x(N_1-1,N_2-1) \end{pmatrix}$$

and

$$\mathbf{X} = \begin{pmatrix} X(0,0) & X(0,1) & \cdots & X(0,N_2-1) \\ X(1,0) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ X(N_1-1,0) & \cdots & \cdots & X(N_1-1,N_2-1) \end{pmatrix}$$

Express the two-dimensional DFT according to equation (3.7a) in Lim as a relation of matrices

$$\mathbf{X} = \mathbf{F}_1 \mathbf{x} \mathbf{F}_2$$

and derive the expressions of the matrices \mathbf{F}_1 and \mathbf{F}_2 . Calculate the inverse of this relationship and thereby derive the inverse DFT, as given by equation (3.7b) in Lim. Furthermore, try to find out how to create the matrices \mathbf{F}_1 and \mathbf{F}_2 in Matlab.

b) Derive the two-dimensional Fourier Transform $W(\omega_1, \omega_2)$ of the windowing function

$$w(n_1, n_2) = \begin{cases} 1 & 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{💬}$$

Is $w(n_1, n_2)$ a separable sequence?

c) Derive the two-dimensional DFT $X(k_1, k_2)$ of the cosine functional

$$x(n_1, n_2) = \begin{cases} \cos(\omega_1 n_1 + \omega_2 n_2) & 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1 \\ 0 & \text{otherwise} \end{cases}$$

for arbitrary values of (ω_1, ω_2) , and when $\omega_1 = 2\pi l_1/N_1$ and $\omega_2 = 2\pi l_2/N_2$, l_1 and l_2 are integers. Hint: rewrite the cosine-functional using Euler's Formulas, note that the results in b) could be used.

Is $x(n_1, n_2)$ a separable sequence?

3 Tasks

Task 1: The Convolution

Use the functions **conv2** and **filter2** in Matlab to verify pre-assignment 1. The function **conv2** gives the total convolution whereas **filter2** gives the part of the output signal that has the same support as the input signal. Note that **filter2** uses an 180 degree rotation of the impulse response as input parameter whereas **conv2** uses the impulse response itself.

The function **filter2** always interpret the impulse response as in pre-assignment 1, i.e. when the impulse response has an even number of rows and/or columns, the origin will be in the center of the impulse response, down to the right. When the impulse response has an un-even number of rows and columns, the origin is in the middle of the impulse response.

Monochrome images could be displayed at the screen by the function

```
>> imshow(I,64)
```

where I is the intensity matrix of the image with values between 0 and 1. A 0 corresponds to least intensity (black) and a 1 corresponds to highest intensity (white). The second argument determines the number of levels of gray that will be presented.

To transform an arbitrary matrix to an intensity matrix the function `mat2gray` could be used. The following command will thus give a gray-level image from an arbitrary matrix A.

```
>> imshow(mat2gray(A),64)
```

Produce a gray-level image of the convolution above, and compare the levels of gray with the values in the corresponding intensity matrix. Try other simple intensity matrices as well, such as


```
>> imshow(ones(16,1)*(0:1/15:1),64)
```

which will display an image having 16 levels of gray.

Most of the images that are stored in the Image Processing Toolbox are 'indexed color-images', refer to the Matlab manual. Such images can be loaded and displayed with the following commands

```
>> load mandrill;  
>> imshow(X,map)
```

Here X is a matrix with a color-index for each pixel position and map is a matrix which defines the transform from a color-index to an intensity in the corresponding pixel's color-components (r=red, g=green, b=blue). An indexed color-image could be transformed into a monochrome image and displayed on screen by the commands

```
>> I=ind2gray(X,map);  
>> imshow(I,64) 
```

If you get coloring problems after displaying monochrome images you could try the command

```
>> colormap('default')
```

to reset the settings. Some other available images are **trees**, **forest**, **earth**, **cape** and **clown**.

Task 2: The Fourier Transform

a) Use the function `fft2` to verify pre-assignment 2 a). The command

```
>> X=fft2(x);
```

should give the same result as the expression $\mathbf{X} = \mathbf{F}_1 \mathbf{x} \mathbf{F}_2$, as in pre-assignment 2 a). Use an arbitrary signal matrix for verification. The `fft2` calculation is as most effective when N_1 and N_2 are powers of two, i.e. 2^L where L is an integer.

b) The Fourier Transform of a 2-D window $w(n_1, n_2)$ according to pre-assignment 2 b) having $N_1 = 8$ and $N_2 = 8$ are calculated and plotted in three dimensions by the following commands

```
>> N1=8; N2=8;
>> x=ones(N1,N2);
>> X=fft2(x,64,64);
>> ax=-pi:2*pi/63:pi;
>> colormap('default')
>> mesh(ax,ax,20*log10(abs(fftshift(X'))+0.01)) % the constant 0.01 will render a floor at -40
dB
>> xlabel('omega1')
>> ylabel('omega2')
```

The second and third argument to **fft2** will pad **zeros** to the **signal** matrix such that the DFT will get the frequency **resolution** 64×64 . This means that we calculate the Fourier Transform of the windowing function $w(n_1, n_2)$ in 64×64 **frequency coordinates**. The function **fftshift** will **move** the **origin** of the **frequency** to the **center** of the **matrix**. Try other values of N_1 and N_2 . Do the results conform to pre-assignment 2 b) ?

Instead of using a 3-D plot with **mesh** as above, the Fourier Transform could be displayed using a 2-D color-plot with axis of frequencies as matrix coordinates, using the following commands

```
>> colormap(jet(64))
>> imagesc(20*log10(abs(X)+0.01)) % will put the origin of frequency in the top left corner
>> colorbar
>> pause
>> colormap('default')
```

or

```
>> colormap(jet(64))
>> imagesc(20*log10(abs(fftshift(X))+0.01)) % will put the origin of frequency in the center
>> colorbar
>> pause
>> colormap('default')
```

At the course's web-page (<http://www.its.bth.se/courses/etd003>) are five FIR filters **h** stored (in ZIP-format), in the files **hlp1**, **hlp2**, **hhp1**, **hhp2** and **hk**. There are **two** low pass filters, two **high** pass filters and one **undefined** filter. Download these files (the experiment instructors have them on disc as well) to your local directory. You could add a **directory** to the Matlab **path** by using the **command**

path(path,'desired directory path').

Calculate and plot the **amplitude** of the Fourier Transform (=the frequency function) for any of these filters in the same manner as above.

c) Let $N_1 = 64$ and $N_2 = 64$. Calculate and **plot** the **two-dimensional** DFT of the cosine signal in pre-assignment 2 c) for the **frequencies** $(\omega_1, \omega_2) = (2\pi 8/64, 2\pi 16/64)$ and $(\omega_1, \omega_2) = (2\pi 8.5/64, 2\pi 16.5/64)$. Tip: the cosine signal could be **created** in the following manner


```
>> n1=(0:N1-1)*ones(1,N2);
>> n2=ones(N1,1)*(0:N2-1);
>> x=cos(w1*n1+w2*n2);
```

Try different frequencies as well. Show the image of x in gray by using **imshow**. Could you explain the directions of the stripes? Plot the DFT of x by using the commands above. Could you explain the difference in the results when $(\omega_1, \omega_2) = (2\pi l_1/64, 2\pi l_2/64)$ and l_1 and l_2 are integers compared to when l_1 and l_2 are non-integers? Compare the results with pre-assignment 2 c).

Repeat the procedure above having the signal dimensions $N_1 = N_2 = 8$, $(\omega_1, \omega_2) = (2\pi/8, 2\pi/8)$ but using the same DFT dimension as before, i.e. using the frequency resolution 64×64 . Produce a **mesh** and a 2-D color-plot from the results. What do you see? Repeat one more time, now having the signal dimensions $N_1 = N_2 = 16$, $(\omega_1, \omega_2) = (2\pi/16, 2\pi/16)$ and the same DFT dimension as before. Could you explain the phenomenon spectral leakage from these results?

d) Load the mandrill image once more, and **resize** it to 128×256 pixels using the commands

```
>> I=ind2gray(X,map);
>> I=I(1:128,120:120+256-1);
```



Create a 2-D DFT of the resized mandrill image. Now, remove any phase information, though still keeping the amplitude information in the image, by adding a random phase to the DFT. This could be performed by using **Ib=fft2(I).*exp(j*2*pi*rand(128,256))**. Apply the inverse transformation by using **real(ifft2(Ib))** such that the small imaginary part, due to numerical inaccuracies, disappears. Display the results as a monochrome image.

Repeat the same experiment but keep the phase information and remove the absolute information. This could be performed by element-wise division with the amplitude function, **If=fft2(I)./abs(fft2(I))**. Apply the inverse transformation as above and display the results as a monochrome image.

Which of the two quantities seems to be of most importance in an image, the amplitude or the phase?

Task 3: Filtering

a) Filter the resized mandrill image above (see Task 2 d)) using the filters **hlp1**, **hlp2**, **hhp1**, **hhp2** and **hk** (see Task 2 b)). Comment the result with relation to the frequency functions of the filters. Tip: make one program that reads an image, filters it with the filter **h**, displays the results, before and after the filtering, as monochrome images.

b) Expand your program in a) such that you could study the effects of a cosine disturbance $\cos(\omega_1 n_1 + \omega_2 n_2)$. Also, write a program that calculates and plots the frequency response function of a filter **h**, loads an image, adds the disturbing noisy element $\cos(\omega_1 n_1 + \omega_2 n_2)$ to the intensity matrix of the image, filters the noisy image with the filter **h**, and displays the results; the original image, the noisy image and the filtered image as monochrome images.

Tell, for each and one of the filters **hlp1**, **hlp2**, **hhp1**, **hhp2** and **hk**, which two-dimensional frequencies that can be suppressed in the noisy image.

c) An image-noise **n**, colored by the filter **hbrus** (e.g. one of the filters given above) could be created by using the command

```
>> n=filter2(hbrus,rand(N1,N2));
```

Modify the program in b) such that you instead of the cosine disturbance, creates an image-noise colored by a 2-D FIR filter, and filters the disturbed image by another suitable FIR filter. The result could be displayed as in b). For which types of noise does the image restoration give best performance, and why?