Practice III. Image histogram and 2D DFT.

1 Objectives

- Characterize images through histograms.
- Analyze the frequency representation of images with 2D DFT.
- Understand how two different kinds of modifications on the image are reflected on the Fourier domain. In particular, the effect of granular noise.

2 Previous Study

2.1 Histogram equalization

In this section, we will review an important grey-level transformation known as histogram equalization. As its name suggests, this transformation aims at expanding as much as possible the image histogram in order to enhance its contrast. In order to understand how the histogram equalization works, an illustrative exercise is proposed:

Consider a NxN image quantified with 3 bits that exhibits the following normalized histogram:

$h_r(0)=1/8$	$h_r(1)=1/6$	$h_r(2)=1/3$	$h_r(3)=1/4$
$h_r(4)=1/8$	$h_r(5)=0$	$h_r(6)=0$	$h_r(7)=0$

In order to improve its contrast, this image is transformed by using the transformation s=t(r). The resulting image has the following histogram:

$h_s(0)=450$	$h_s(1)=0$	$h_s(2)=600$	$h_s(3)=1200$
$h_s(4)=0$	$h_s(5)=900$	$h_s(6)=0$	$h_s(7)=450$

1. What is the size N of the image?
2. Could you find the monotonic transformation s=t(r) that has been applied?
3. The previous transformation improves the contrast but does not equalize the histogram. Find the transformation s=t(r) that "equalizes" the histogram of the original NxN image. Fill in this table:

Original level (r)	Transformed level (s)
0	
1	
2	
3	
4	
5	
6	
7	

2.2 Discrete Fourier Transform (2D-DFT) of images

The discrete Fourier transform (2D-DFT) of MxN samples of an image x[m,n] is defined as:

$$DFT_{MxN}\{x[m,n]\} = X_{MxN}[k,l] = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[m,n]e^{-j\frac{2\pi}{M}km}e^{-j\frac{2\pi}{N}ln}, \quad 0 \le k \le M-1 \quad 0 \le l \le N-1$$

and the inverse 2D-DFT as:

$$x[m,n] = IDFT_{MxN}\left\{X[k,l]\right\} = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} X[k,l]e^{j\frac{2\pi}{M}km}e^{j\frac{2\pi}{N}ln}, \quad 0 \le m \le M-1 \quad 0 \le n \le N-1$$

4. Compute analytically the 2D-DFT of MxN samples of a discrete impulse $x[m,n] = \begin{cases} 1 & m=n=0 \\ 0 & other\ m,n \end{cases}$

5. Compute analytically the 2D-DFT of MxN samples of a 2D rectangular pulse of PxQ samples (with P<M and Q<N) $x[m,n] = \begin{cases} 1 & 0 \le m \le P-1 \quad 0 \le n \le Q-1 \\ 0 & other\ m,n \end{cases}$

Let us now analyze some properties of the 2D-DFT of images. In particular:

a) Analyze the **circular** shift of the 2D-DFT. The property states that the 2D-DFT of MxN samples, $Y[k,l]$, of an image $y[m,n] = x[m,n] \cdot e^{j2\pi\left(\frac{mr}{M}+\frac{ns}{N}\right)}$ is equal to the circular shift of the 2D-DFT of x[m,n]: $Y[k,l] = \tilde{X}[k-r,l-s]$. Now we are interested in expressing this property in terms of $X[k,l]$ directly.

6. Take M=N=8 and write the 2D-DFT of both $y_1[m,n] = x[m,n] \cdot e^{j2\pi\left(\frac{m}{M}4+\frac{n}{N}2\right)}$ and $y_2[m,n] = x[m,n] \cdot e^{j2\pi\left(\frac{m}{M}10+\frac{n}{N}2\right)}$ in terms of X[k,l] k = 0, …, M-1 and l = 0,…, N-1, the DFT of $x[m,n]$. (Hint: for Y_1[k, l] distinguish the following cases: a) k is lower or not than 4; and, b) l is lower or not than 2. Proceed for Y_2[k, l] equivalently).

b) Consider the swapping of the spatial axes:

7. Obtain the DFT_{NxM} of y[m,n] = x[n,m] in terms of the DFT_{MxN} of x[m,n].

In the laboratory, a picture will be processed to remove a background pattern similar to a grain texture like this:

r[m,n]= (a + b·cos(2πF_Am)) · (c + d·cos(2πF_Bn)) for 0≤m<M, 0≤n<N.

8. Fill this table in with the pattern r[m,n] considering that a=b=c=d=1, F_A=F_B=0.25 and M=N=8.

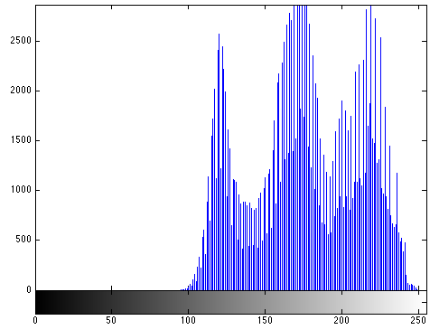
9. Write the Fourier transform of r[m,n] for a=c=1, b=d=1/2 and F_A= F_B= 0.3 in terms of the Fourier transform of the rectangular window of MxN samples.

3 Laboratory

3.1 Histograms

The objective of this section is to characterize the images by means of their histogram, and study the effects of the histogram equalization on the contrast of the images. Let us load an image and observe its histogram:

```
x = imread('plaza.bmp');
figure(1),imshow(x);
figure(2),imhist(x), axis 'tight';
```



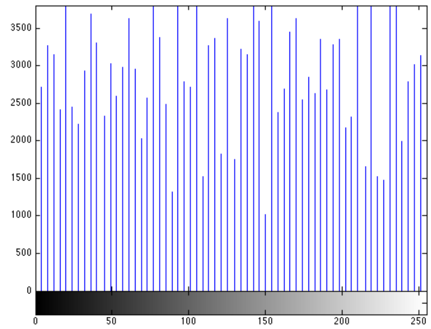
Figures 3.1 and 3.2: Original image (left) and its histogram (right)

The image **x** consists of 500 x 375 pixels, represented by a unsigned integer of 8 bits, which gives 256 possible gray levels. As you can see the loaded image has poor contrast and that is reflected on its histogram.

1. Interpret the histogram of the image **x**. Are all gray levels used equally? What image gray levels are missing in the original image **x**?
2. Looking at the histogram, what is the probability of a pixel of the image **x** having a gray value of 32? And 200?

In order to improve the contrast of the image we will equalize its histogram.

```
[y,t] = histeq(x);
figure(3),imshow(y);
figure(4),imhist(y), axis 'tight';
```



Figures 3.3 and 3.4: Equalized image (left) and its histogram (right)

As it can be seen from the previous figures, the histogram equalization produces an image with increased contrast as the number of gray levels used is greater.

3. From the equalized histogram, what is now the probability of a pixel of the equalized image **y** having a gray value of 32? And 200?
4. Are all gray levels used equally in the equalized image? Why doesn't the histogram equalization accomplish a completely flat histogram?

The resulting gray scale transformation that maps gray levels in the original image **x** to gray levels in the equalized image **y** can be represented as follows:

```
figure(5),plot(0:255,round(t*255)); axis([0 255 0 255]);
```

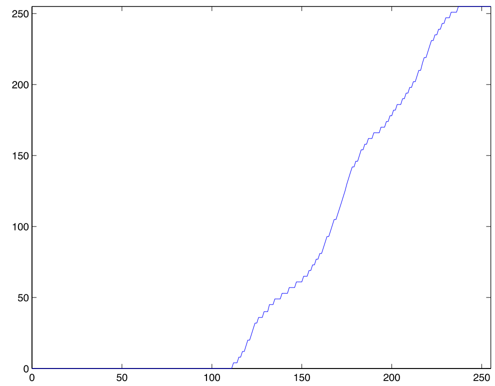


Figure 3.5: Gray level transformation to equalize image **x**

5. From the gray level transformation, what has happened to the gray levels between 100 and 255 in the original image **x**?
6. Looking at the histogram of the original image **x** in figure 3.2 and the gray-level transformation in figure 3.5, calculate again the probability of pixels with gray value 32 and 200 of the equalized image **y** (you have already got this result in your answer to question3).

A problem of the histogram equalization appears when dealing with color images. As the equalization procedure is defined per channel, equalizing RGB images independently for each channel might create false colors in the resulting image. To study this effect and possible solutions we will load a new image into Matlab. The original image and a low contrast version that we want to equalize.

```
flor_orig = double(imread('flor_orig.png'))/255;
figure(6),imshow(flor_orig);

flor = double(imread('flor_lc.png'))/255;
figure(7),imshow(flor);
```

Figures 3.6 and 3.7: Original image (left) and its low contrast version to equalize (right)

First of all we will apply the histogram equalization to each red/green/blue channel of the low contrast image and we will compare the result with the original image.

```
r = flor(:,:,1);       % RED channel of image
g = flor(:,:,2);       % GREEN channel of image
b = flor(:,:,3);       % BLUE channel of image

flor_eq1 = zeros(size(flor));
flor_eq1(:,:,1) = histeq(r);
flor_eq1(:,:,2) = histeq(g);
flor_eq1(:,:,3) = histeq(b);

figure(8),imshow(flor_eq1);
```

Figures 3.8: Equalized image

7. Has the contrast of the equalized image increased with respect to the low contrast version? Are the colors of the original image preserved? Why are colors changed in the equalized image?

In order to minimize the problem of false colors a possible solution is to work in another color space, such as YCbCr, where the luminance and the chrominance components of the color image are separated into different channels. Then, only the luminance component, in the case of the YCbCr the Y channel, is equalized.

```
flor_ycbcr = rgb2ycbcr(flor);

y  = flor_ycbcr(:,:,1);      % Y channel of image
cb = flor_ycbcr(:,:,2);      % Cb channel of image
cr = flor_ycbcr(:,:,3);      % Cr channel of image

% Equalization of luminance
z = zeros(size(flor));
z(:,:,1) = histeq(y);
z(:,:,2) = cb;
z(:,:,3) = cr;

flor_eq2 = ycbcr2rgb(z);
figure(9),imshow(flor_eq2);
```

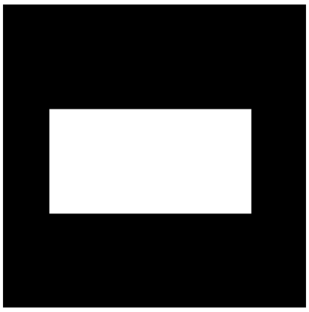
Figures 3.9: Equalized image (using only luminance)

8. In the case of using only the luminance component to equalize the image, has the contrast of the equalized image increased with respect to the low contrast version? Do you think colors are better preserved in this case than equalizing all RGB channels?

3.2 Image characterization with DFT

Let us analyse the DFT of a simple image. Load the following image:

```
image_rec = imread('rectangle.bmp');
figure(10),imshow(image_rec,[]);
[H,W] = size(image_rec)
```



Figures 3.10: Original binary image 'rectangle'

Then compute its DFT (with the FFT command) and present the DFT after a shift so that the frequency (Fx,Fy)=(0,0) appears in the image center. Notice that the size of the computed DFT equals the size of the image.

```
TR_image_rec = fft2(double(image_rec));
TR_image_rec = fftshift(TR_image_rec);
figure(11); colormap(jet);
imagesc(log(abs(TR_image_rec)))
axis image; title('{\bf Magnitude of the DFT}');
```

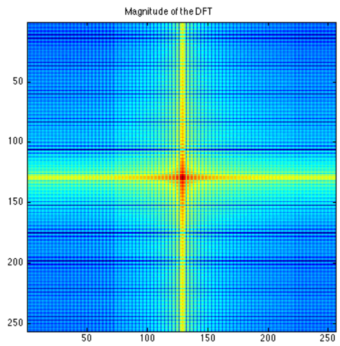


Figure 3.11: Magnitude of the DFT of the rectangle image

Let us illustrate the Fourier transform of an image when multiplied by a cosine function. We will use the rectangle image above and we will multiply it by a vertical cosine function of discrete frequency 1/8:

```
for m=1:H
    for n=1:W
       image_rec_cos(m,n) = double(image_rec(m,n)) * cos(2*pi*m/8);
    end
end
figure(12); imshow(image_rec_cos,[]);
```

Figure 3.12: Rectangle image multiplied by a cosine function

Then we compute its DFT and present its modulus:

```
TR_image_rec_cos = fft2(double(image_rec_cos));
TR_image_rec_cos = fftshift(TR_image_rec_cos);
figure(13); colormap(jet);
imagesc(log(abs(TR_image_rec_cos)))
axis image; title('{\bf Magnitude of the DFT}')
```

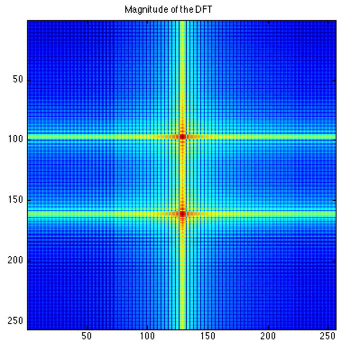


Figure 3.13: Magnitude of the DFT of the image multiplied by a cosine function

9. Interpret the magnitude of the DFT of the image. Why are two important frequency components appearing? Where (i.e., coordinates k and l) are they appearing in the frequency plane, why?

Let us now study the Fourier transform of a real image:

```
image_car = imread('car.bmp');
figure(14),imshow(image_car,[]);
[M,N] = size(image_car)
```

Figure 3.14: Original 'Car' image

Then, compute its DFT (with the FFT command) and present the DFT after a shift so that the frequency (Fx,Fy)=(0,0) appears in the image center.

```
TR_image_car = fft2(double(image_car));
TR_image_car = fftshift(TR_image_car);
figure(15); colormap(jet);
imagesc(log(abs(TR_image_car)))
axis image; title('{\bf Magnitude of the DFT}')
figure(16); colormap(jet);
imagesc(angle(TR_image_car))
axis image; title('{\bf Phase of the DFT}')
```

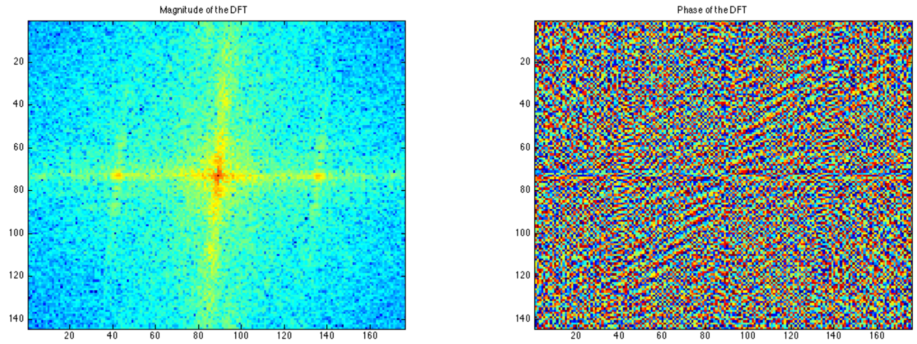


Figure 3.15 and 3.16: Magnitude and phase of the DFT of the car image

10. Interpret the magnitude of the DFT of the image. Important frequency components can be observed in the magnitude representation around coordinates (135, 73) and (43, 73). Compute the discrete frequencies corresponding to these components. What do they represent in the image? Can you prove your answer by measuring the periodicity of some feature in the original image?

3.3 Effect of texture in the transform domain.

In this section we will apply a processing that corrects the distortion introduced by a background pattern similar to a grain texture. First, we read the distorted image and show it and its DFT:

```
FileName= 'LenaText.png';
[LenaTx] = imread(FileName);
[M,N]=size(LenaTx)
figure(17),imshow(LenaTx),axis image, title('Lena with background pattern');

TLenaTx = fft2(LenaTx);
figure(18),imagesc(log(abs(fftshift(TLenaTx)))), axis image, title('DFT of Lena with
background pattern');
```

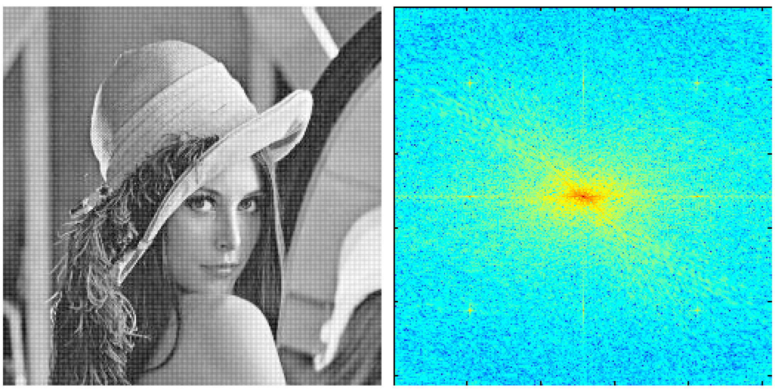


Figure 3.17 and 3.18: Lena image with grain texture and its DFT

11. Could you find in the DFT shown in Fig. 3.18 the frequency components introduced by the granular pattern? (Hint: compare this DFT with the DFT of the pattern r[m,n] of the previous study)

We will apply a mask to the DFT information to try to eliminate the contribution of the grain texture to the Fourier transform. The mask will eliminate (set to zero) the frequency components of the image related to the texture:

```
TLenaMask = TLenaTx;
FCenter=77;
MediaBanda=5;
for i=FCenter-MediaBanda:FCenter+MediaBanda
    TLenaMask(i,1)=0;
    TLenaMask(258-i,1)=0;
   for j=2:MediaBanda+1
        TLenaMask(i,j)=0;
        TLenaMask(i,258-j)=0;
        TLenaMask(258-i,j)=0;
        TLenaMask(258-i,258-j)=0;
    end
    for j=FCenter-MediaBanda:FCenter+MediaBanda
        TLenaMask(i,j)=0;
        TLenaMask(i,258-j)=0;
        TLenaMask(258-i,j)=0;
        TLenaMask(258-i,258-j)=0;
    end
end
for j=FCenter-MediaBanda:FCenter+MediaBanda
    TLenaMask(1,j)=0;
    TLenaMask(1,258-j)=0;
    for i=2:MediaBanda+1
        TLenaMask(i,j)=0;
        TLenaMask(i,258-j)=0;
        TLenaMask(258-i,j)=0;
        TLenaMask(258-i,258-j)=0;
    end
end

% showing the result
% the DFT with the mask
figure(19);imagesc(fftshift(log(abs(TLenaMask)))); axis image;
title('{\bf Fourier Transform multiplied by the mask }');
% the enhanced image
LenaMask = mat2gray(real(ifft2((TLenaMask))));
figure(20),imshow(LenaMask),axis image,title('Lena with Fourier Transf. mask');
```

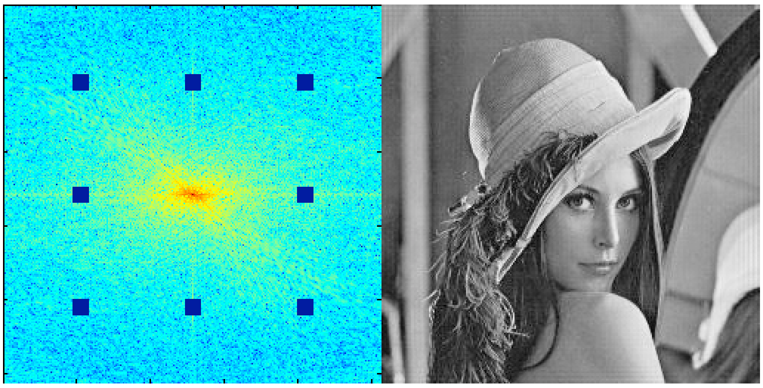


Figure 3.19 and 3.20: Mask applied to the DFT of Lena and the resulting image

12. Has the granular pattern been eliminated? Why? Has the average level of gray been changed in the image? Why?