Practice V. Decimation, Interpolation and filtering of 2D signals

1 Objectives

- Understand the decimation and interpolation process in the spatial domain.
- Characterize decimated/interpolated 2D signals in the frequency domain.
- Study a practical use of 2D filtering for automatic image segmentation

2 Previous study

2.1 Fourier analysis of decimated images

In this section we will revise the process of decimating images. For this, we will use a periodic image composed of a sinusoidal function of horizontal frequency F_a and vertical frequency F_b

$$x[m,n] = cos(2\pi F_a m + 2\pi F_b n) \qquad \text{with } \frac{1}{4} < F_a < \frac{1}{2} \text{ and } 0 < F_b < \frac{1}{4} \qquad (Eq.\ 1)$$

and the decimated version y[m,n] by a factor 2 (y[m,n] = x[2m,2n]).

1. Express y[m,n] as a function of m and n and obtain the relationship between the frequencies appearing in the Fourier transform of image y[m,n] and the pair of frequencies F_a and F_b of x[m,n] (Hint: take into account the possible aliasing). Draw the Fourier Transform $X(F_1,F_2)$ and $Y(F_1,F_2)$.

In the laboratory we will create an image x[m,n] of NxN (N=128) pixels as the inverse discrete Fourier transform of

$$\mathrm{X[k,l]} = \frac{NxN}{2}\{\delta[\mathrm{k}-40,\mathrm{l}-8] + \delta[\mathrm{k}-88,\mathrm{l}-120]\} \qquad \text{for} \quad \mathrm{k,l} = 0,\ldots,\mathrm{N}-1$$

2. Obtain the discrete frequencies that correspond to the non-zero samples of the above DFT expression. Calculate the 64x64 discrete Fourier Transform of y[m,n]=x[2m,2n] and give the discrete frequencies that correspond to the non-zero samples.

2.2 Fourier analysis of interpolated images

The interpolation of images is the opposite process to the decimation. In this case, zero-valued samples are added to the signal (zero-filled) and an interpolation filter gives values to those inserted samples. Using the image defined in (*Eq. 1*), its interpolated version by a factor 2 y[m,n] and h[m,n] the interpolation filter:

3. Express the Fourier Transform $Y(F_1,F_2)$ as a function of $X(F_1,F_2)$ and $H(F_1,F_2)$. Draw the Fourier Transform $Y(F_1,F_2)$ if h[m,n] is an ideal interpolation filter.

In image processing, two common interpolation filters employed are the “nearest neighbor” and the “bilinear”. The “nearest neighbor” gives values to the inserted zeros by copying the value of the closest pixel in the image. In general the filter can be expressed as $x(m_0,n_0) = x[round(m_0), round(n_0)]$, where

m_0 and n_0 represent any position (not integer) between actual pixel values of the image. Furthermore the round() function is usually defined as half round down so round(0.5) = 0.

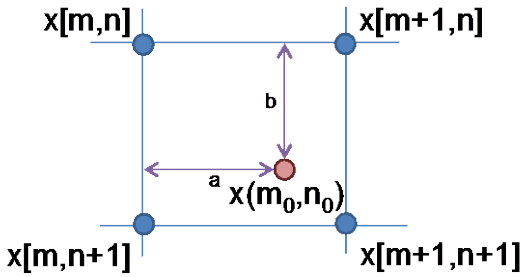


Figure 2.1: Interpolated positions

The "bilinear" filter is a combination of a linear interpolation first in one direction, and then again in the other direction, and it can be expressed as:

$$x(m_0, n_0) = (1-a)(1-b)\cdot x[m,n] + a(1-b)\cdot x[m+1,n] + \\ +(1-a)b\cdot x[m,n+1] + ab\cdot x[m+1,n+1].$$

where a =|m-m_0| and b=|n-n_0|.

In the particular case of interpolation by a factor of 2 (in both dimensions), the interpolated values y_i (represented by an empty circle in Figure 2.2) will be filled taking into account the values of the original pixels x_i (represented as blue filled circles).

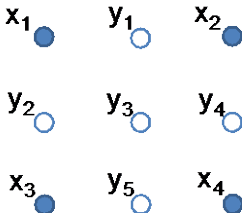


Figure 2.2: Interpolated positions by a factor of 2 (in both dimensions)

In this case, the nearest neighbor filter will assign the following values to y_i:

$$y_1 = x_1,\ y_2 = x_1,\ y_3 = x_1,\ y_4 = x_2,\ y_5 = x_3$$

While the bilinear filter will assign the following values corresponding to m_0=n_0=0.5 in the general formula:

$$y_1 = \frac{1}{2}x_1 + \frac{1}{2}x_2,\ y_2 = \frac{1}{2}x_1 + \frac{1}{2}x_3,\ y_3 = \frac{1}{4}x_1 + \frac{1}{4}x_2 + \frac{1}{4}x_3 + \frac{1}{4}x_4,\ y_4 = \frac{1}{2}x_2 + \frac{1}{2}x_4,\ y_5 = \frac{1}{2}x_3 + \frac{1}{2}x_4$$

To study the advantages and disadvantages of these two interpolation methods, let's use them to interpolate the test image of Fig. 2.3 by a factor of 2. The test image has 30x30 pixels and has two parts: the upper right part is white and the lower left part is black.


Figures 2.3: Original test image (left), interpolated versions by a factor of 2 using the "nearest neighbor" filter (middle) and the "bilinear" filter (right)

4. Looking at figure 2.3, discuss the different effect on the diagonal contour resulting of using the nearest neighbor or bilinear filters as interpolation filters.

2.3 Image binarization

Consider a 32x32 gray-scale image x[m,n] like the one if Fig. 2.4, in which there are four squares on a background whose upper half is brighter than its lower half. In order to automatically determine the number of squares on the image, x[m,n] is binarized using the following transformation:

$$z = T(x) = \begin{cases} 1\ (white) & y > threshold \\ 0\ (black) & y \leq threshold \end{cases}$$

The objective is selecting a satisfactory value for the threshold that results in a binary image z[m,n] with white squares (gray level 1) and black background (gray level 0).

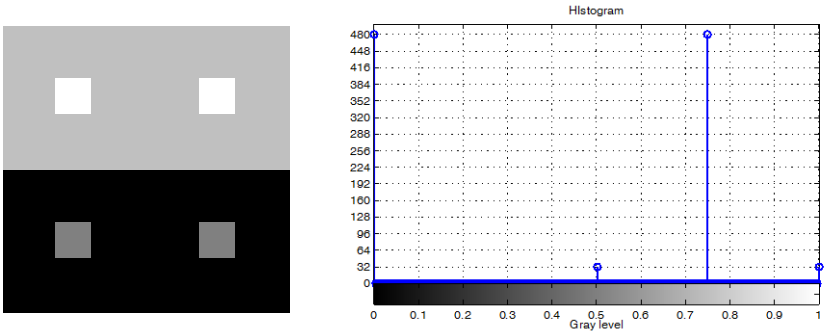


Figure 2.4 and 2.5: synthetic 32x32 gray-scale image (left) and its histogram (right)

5. Using the information provided in Fig. 2.5, find the gray level of the different parts of image x[m,n] in Fig. 2.4 (squares and background).
6. Looking at the histogram in fig. 2.5, explain why it is not possible to binarize this image (Fig. 2.1) in such a way that the squares take gray value 1 and the background value 0.

The cause of this to problem is the different illumination of the upper and lower half of the image. The solution that you will test in the laboratory consists in subtracting the illumination image i[m,n] from the original image x[m,n] before binarization. In order to understand the rationale for this, consider that i[m,n] is equal to a constant C within the interval 0≤n<16 (upper half) and zero otherwise.

7. Find a value for constant C that makes possible to binarize x[m,n]-i[m,n] correctly.

In practice, i[m,n] will be obtained from x[m,n] by means of a low-pass filter $h_i[m,n]$. Consider that $h_i[m,n]$ is the following arithmetic mean filter:

$$h_i[m,n] = \begin{cases} \frac{1}{17^2} & -8 \leq m,n < 8 \\ 0 & otherwise \end{cases}$$

8. Describe *qualitatively* the filtered image y[m,n] = x[m,n]*h_i[m,n] for x[m,n] the image in Fig. 2.4. Assume "mirroring" for the 2D convolution.
9. Discuss how the size of the squares impact on the computation of the illumination image i[m,n].

The complete implementation of the image processing before binarization is provided in the figure below. According to this scheme, the irregular illumination i[m,n] is extracted from the original image using a low-pass filter with impulse response h_i[m,n]. Additionally, the average illumination

$$\mu = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} i[m,n]$$

is computed as well and subtracted from i[m,n] in order to guarantee that the average illumination of the original image is preserved in the output image y[m,n].

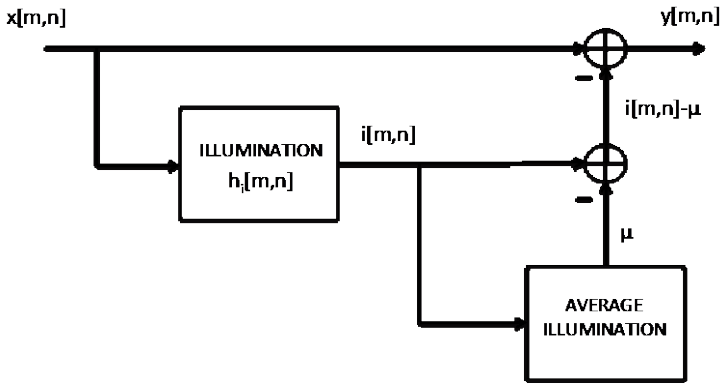


Figure 2.6: the whole processing system applied before image binarization

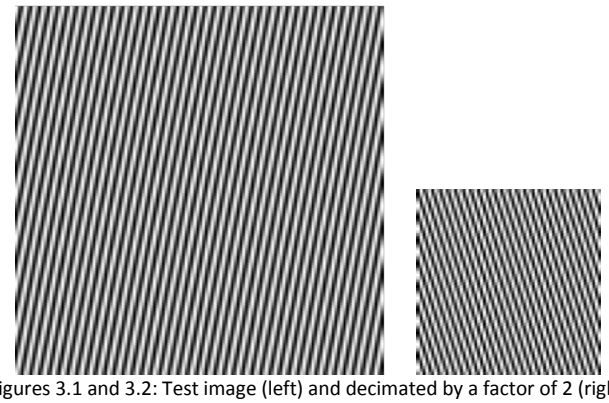
10. If we consider the entire system that gives y[m,n] from x[m,n] as a blackbox, what kind of filtering (low-pass, high-pass, band-pass or band-stop) does it render? Explain why.

3 Laboratory

3.1 Decimation of images

The objective of this section is to study the effects of decimating images. We will study the decimation process with a test image of 128x128 pixels defined in (Eq.1) of the previous study. We will create the test image by means of an inverse Fourier transform and we will decimate it by a factor of 2 (without anti-aliasing filter):

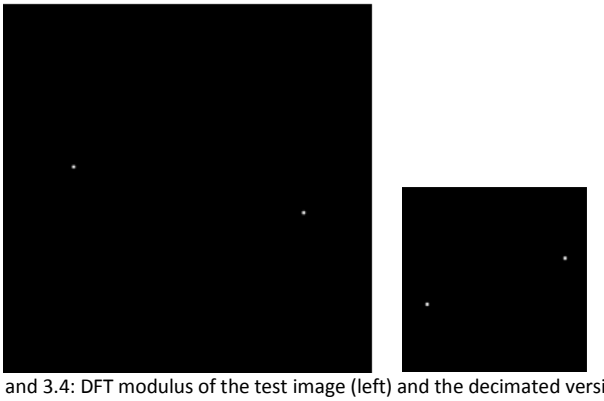
```
N=128;                                  % Size of the image NxN
F1=40; F2=8;                            % DFT position of the wavefront frequencies
X = zeros(N,N);                         % First create an DFT of zeros
X(F2+1,F1+1)=N*N/2;                     % Set 2 samples values
X(N-F2+1,N-F1+1) = N*N/2;
x=ifft2(X);                             % Obtain the inverse DFT
x_dec = imresize(x,1/2,'nearest');      % Decimate by 2 (without filter)
figure(1), imshow(x,[]);                % Represent images
figure(2); imshow(x_dec,[]);
```



Figures 3.1 and 3.2: Test image (left) and decimated by a factor of 2 (right)

We will also represent their respective centered DFTs (notice that the size of the DFT is the same that the size of the image):

```
X = fftshift(log(1+abs(fft2(x))));              % DFT modulus of x
figure(3),imshow(X,[]);
X_dec=fftshift(log(1+abs(fft2(x_dec))));        % DFT modulus of x_dec
figure(4),imshow(X_dec,[]);
```



Figures 3.3 and 3.4: DFT modulus of the test image (left) and the decimated version (right)

1. Obtain the discrete frequency of the sinusoids in both images and relate them to the frequencies obtained in your answer to question 2 of the previous study. Could you describe the difference you observe between figures 3.1 and 3.2?

Let us load a real image and observe the effects of the decimation process (by a factor of 2). First of all, to avoid aliasing, we will use a low pass filter to limit the frequency band of the input image:

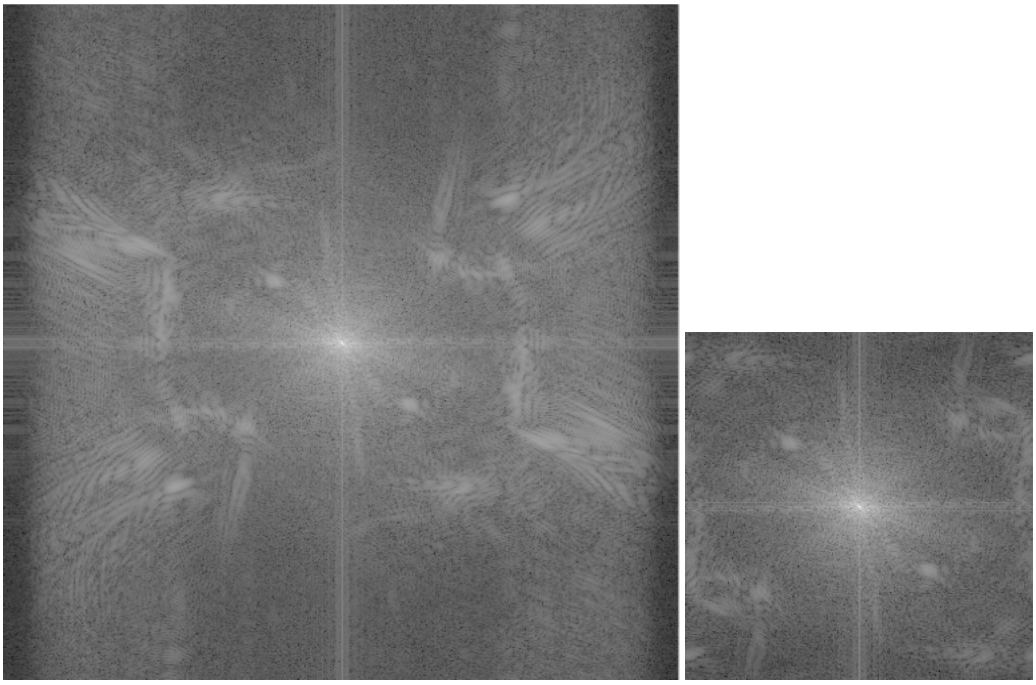
```
x = double(imread('barbara.png'));              % Read image
figure(5),imshow(x,[]); title('Original');      % Display in window
```

```
g = fspecial('gaussian',5,0.8);                  % Prepare a low-pass filter
x_low = imfilter(x,g,'replicate');               % Filter the image with the low-pass filter
y_low = imresize(x_low,0.5,'nearest');           % Create a low pass version of the image
figure(6),imshow(y_low,[]);title('Decimated');   % Decimate the low pass version
```

Figures 3.5 and 3.6: Original image (left) and decimated version by a factor of 2 (right)

Observe their respective Fourier transforms:

```
X = fftshift(log(1+abs(fft2(x))));                % DFT of image
figure(7),imshow(X,[]); title('Original DFT');
Y_low = fftshift(log(1+abs(fft2(y_low))));        % DFT of low pass version
figure(8),imshow(Y_low,[]);
title('Low pass filtered decimated DFT');
```

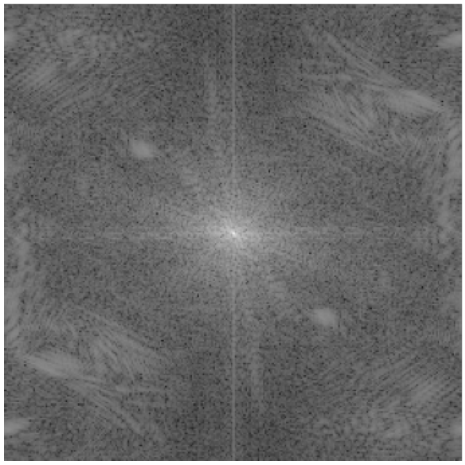


Figures 3.7 and 3.8: DFT of Original image (left) and DFT of decimated version by a factor of 2 (right)

2. What frequency components does the decimated image preserve? Which frequency components are lost?

In the case that no filter is used the decimation process can introduce aliasing artifacts. Let's try to decimate the same image by the same factor without using any low-pass filter:

```
y = imresize(x,0.5,'nearest');                % Decimate by 2 without filter
figure(9),imshow(y,[]);
title('Original decimated');
Y = fftshift(log(1+abs(fft2(y))));            % DFT of decimated image (no filter)
figure(10),imshow(Y,[]);
title('Original decimated DFT');
```

Figures 3.9 and 3.10: Decimated version without anti-aliasing filter (left) and its DFT (right)

3. Can you see the aliasing in the decimated image (left)? Where is it located in the image? Observe the pants of the girl in the image. Based on the conclusions drawn in question 1 with the test image, can you interpret why the direction of the stripes on the pants have changed?
4. From the DFT representation (compare Figures 3.8 and 3.10), what are the main differences between using or not the anti-aliasing filter? Which frequencies get more affected?

3.2 Interpolation of images

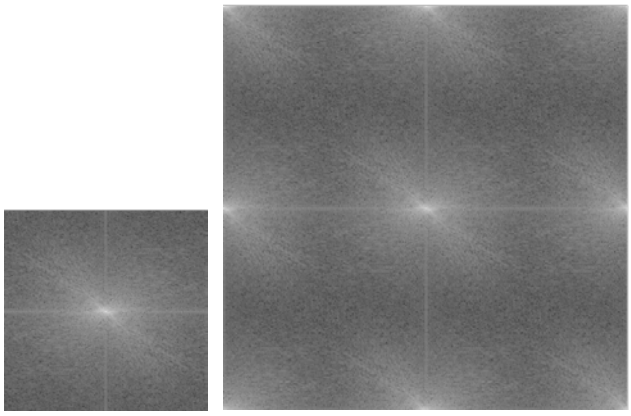
The interpolation of images is the opposite process to the decimation. In this case, zero-valued samples are added to the signal (zero-filled) and an interpolation filter gives values to those inserted samples. Let us first analyze in the spatial and frequency domain the process of inserting zero-valued samples to an image.

```
x = double(imread('lena.bmp'));               % Read Lena image
figure(11), imshow(x,[]);
x2 = zeros(2*size(x));
x2(1:2:end,1:2:end) = x;                      % Zero-stuffing version (N=2)
figure(12),imshow(x2,[]),truesize;
```


Figure 3.11 and 3.12: Lena image and zero-filled version by a factor of 2

As in the case of the decimation process, be careful when re-sizing Matlab figures as they will suffer undesired decimation/interpolation processes.
Next, we can analyze these images in the frequency domain:

```
X = fftshift(log(1+abs(fft2(x))));          % DFT modulus of the Lena image
figure(13),imshow(X,[]);
X2 = fftshift(log(1+abs(fft2(x2))));        % DFT modulus of the zero-stuffing version
figure(14),imshow(X2,[]);
```



Figures 3.13 and 3.14: DFT of the original image (left) and the zero-filled version (right)

5. How does the DFT of the zero-filled version relate to the DFT of the original image?

As it has been seen in the previous study, two common interpolation filters employed in image processing are the “nearest neighbor” and the “bilinear” filters. We will use the Lena image to further study the advantages and disadvantages of them.

```
x2 = inter2(x);
figure(15),imshow(x2,[]),truesize;
title('Bilinear filter');
x3 = imresize(x,2,'nearest'); x3=x3(1:end-1,1:end-1);
figure(16),imshow(x3,[]),truesize;
title('Nearest neighbor filter');
```

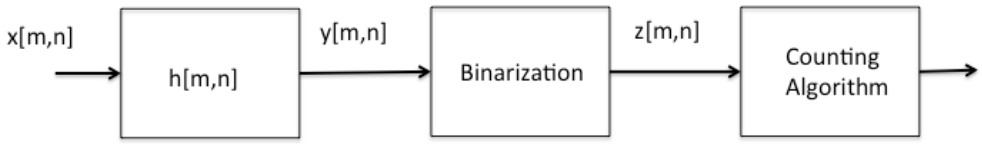
Figures 3.15 and 3.16: Interpolated version using “nearest neighbor” filter (left) and interpolated version by a factor of 2 using “bilinear” filter (right)

6. Observing the figures and your answer to the question 4 of the previous study, what are the advantages of using the “bilinear” interpolation filter instead of the “nearest neighbor”?

3.3 Image filtering to remove shadows (non-uniform illumination)

In this section, the aim is to develop an automatic procedure to count the number of rice grains in a gray-scale image like the one given in Fig. 3.17. The proposed method consists in binarizing the original image

and applying later a convenient counting algorithm to the binarized image. As it is evidenced when we look at Fig. 3.17, the original image is not uniformly illuminated. In particular, we observe that the bottom of the image is darker than the top of the image. As it will be apparent later, a non-uniform illumination makes difficult to binarize the original image correctly and, therefore, to calculate the number of rice grains in the image accurately. In order to improve the image illumination before binarization, we propose to filter the original image using a 2D linear filter h[m,n]. The complete block diagram is given in next figure:



Let us start by representing the original image and its histogram:

```
clear all, close all;
FileName = 'rice'; % Original image
x = double(imread([FileName,'.png']))/255;
figure(17),imshow(x,[]),title('Original image');
figure(18),imhist(x), axis 'tight', title('Histogram of the original image');
```

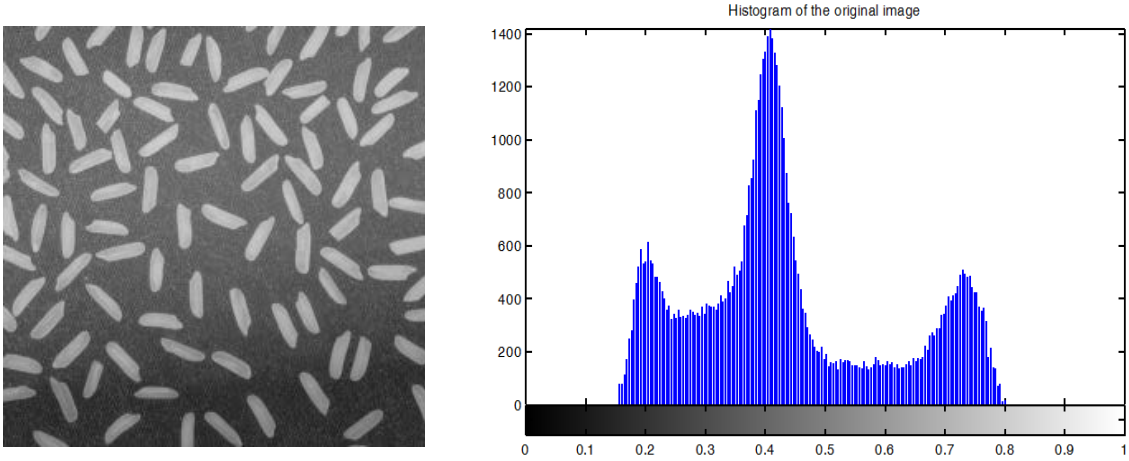


Figure 3.17 and 3.18: Non-uniformly illuminated original image (left) and its histogram (right).

Let us study next what happens if the original image is directly binarized without using any previous filter. In that case, we have that y[m,n]=x[m,n] and the binarized image z[m,n] is obtained by means of the following simple gray-scale transformation:

$$z = T(y) = \begin{cases} 1 & y > threshold \\ 0 & y \leq threshold \end{cases}$$

In particular, if the binarization threshold is set to 0.6, we obtain the image in Fig. 3.19 after executing the following Matlab script:

```
% Binarization of the original image
threshold = 0.6;
for m=1:size(x,1);
    for n=1:size(x,2);
        if x(m,n)> threshold;
            z(m,n)= 1;
        else
            z(m,n)=0;
        end
    end
end
figure(7),imshow(z,[])
```

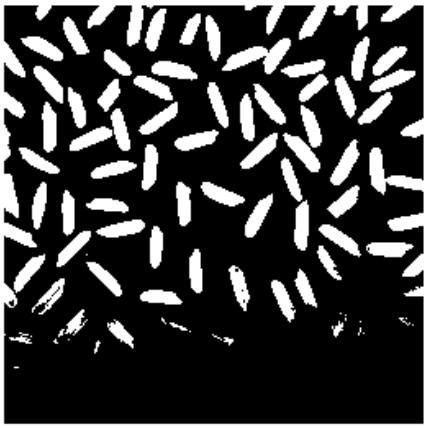


Figure 3.19: Binarized original image

7. What parts of the original image correspond to the histogram peak centered at gray level 0.4? To answer this question, you are recommended to binarize the original image using different thresholds and compare the resulting images.
8. Is there any threshold value that allows a correct binarization of the original image? If the answer is negative, explain why the original image cannot be binarized correctly. You are suggested to inspect how the gray level of rice grains and background changes as the data cursor is moved across the image.

As it has been in the previous study, in order to prepare the original image for binarization, a filter is applied to obtain a more uniform scene illumination. The implementation of this filter (see Figure 2.6 of the previous study) includes a low-pass filter with impulse response $h_i[m,n]$ and the computation of the average illumination of the low-pass filtered image. Let's select a candidate filter $h_i[m,n]$ of size 65x65 samples and represent the magnitude of its frequency response in Figs. 20 and 21.

```
% Illumination filter
L=65; N=256;
hi = fspecial('gaussian',L,100);
Hi = fftshift(fft2(hi,N,N));
f  = -0.5:1/N:0.5-1/N;

figure(20),surf(f,f,log(1+abs(Hi)),'EdgeColor','none');
axis tight, set(gca,'YDir','reverse');view(0,90);colorbar; colormap(jet);
title('Log - Magnitude of the illumination filter frequency response');
xlabel('F1'),ylabel('F2');
figure(21),surf(f,f,abs(Hi),'EdgeColor','none');
axis tight, set(gca,'YDir','reverse'); colormap(jet);
title('Magnitude of the illumination filter frequency response');
xlabel('F1'),ylabel('F2');
```

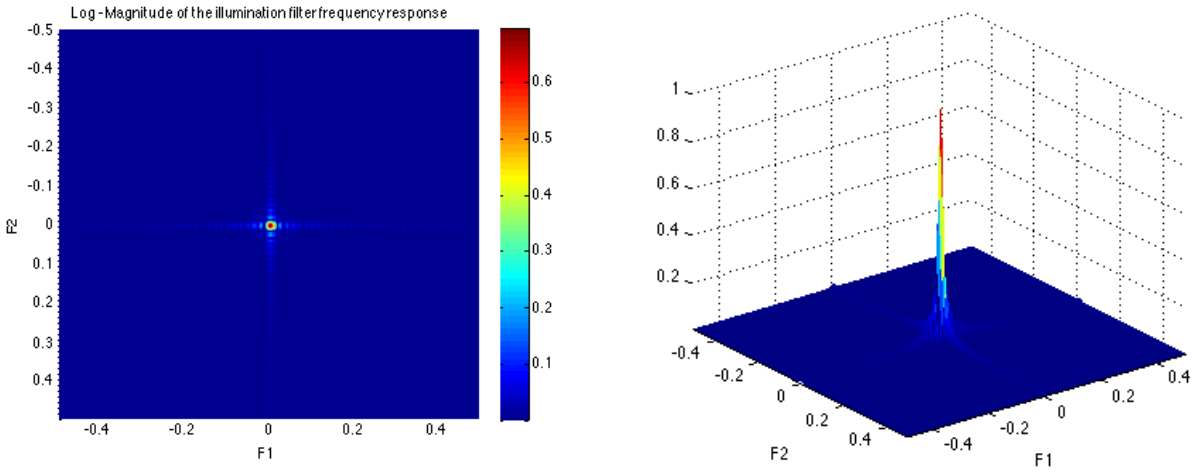


Figure 3.20 and 3.21: Frequency response magnitude of the illumination filter.

Then, filter $h_i[m,n]$ is applied to the original image to obtain the illumination pattern i[m,n] (Fig. 22) and, eventually, the processed image y[m,n] (Fig. 23).

```
i = imfilter(x,hi,'replicate');
figure(22),imshow(i); title('Image illumination');
mu= mean(mean(i));
y=x-i+mu;
figure(23),imshow(y,[]); title('Processed image');
```

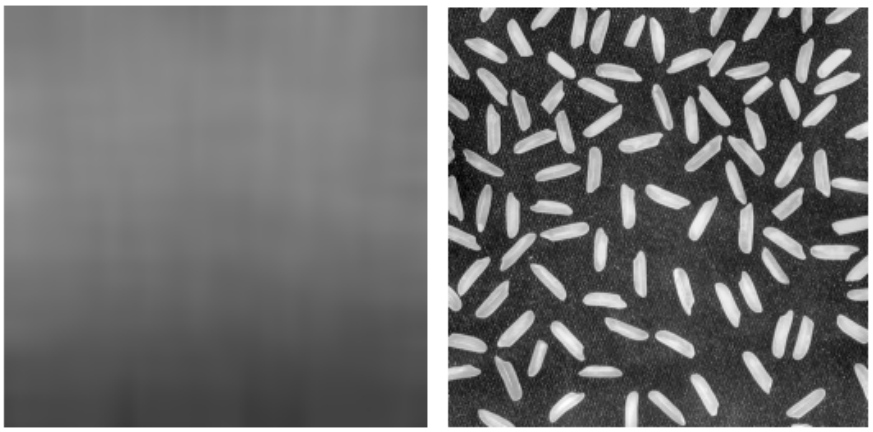
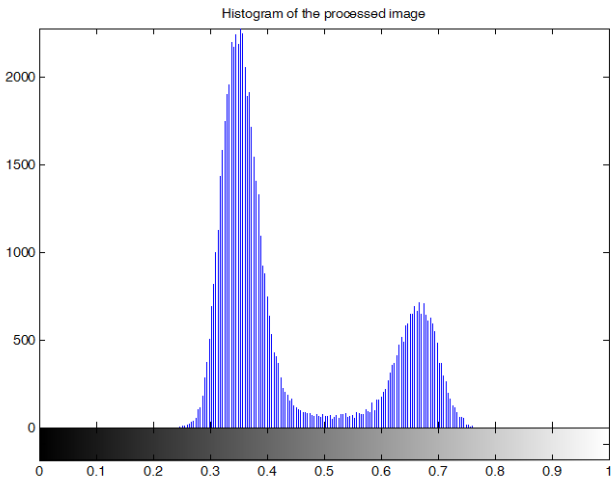


Figure 3.22 and 3.23: Irregular illumination i[m,n] (left) and filtered image y[m,n] (right).

Finally, the filtered image y[m,n] is binarized using a threshold that needs to be determined.

```
% Binarization of the processed image
figure(24), imhist(y), axis 'tight', title('Histogram of the processed image');
threshold = 0.35;            % *** Threshold to be determined ***
for m=1:size(y,1);
    for n=1:size(y,2);
        if y(m,n)> threshold;
            z2(m,n)= 1;
        else
            z2(m,n)=0;
        end
    end
end
figure(25),imshow(z2,[]),title('Binarized image (pre-filtered');
```



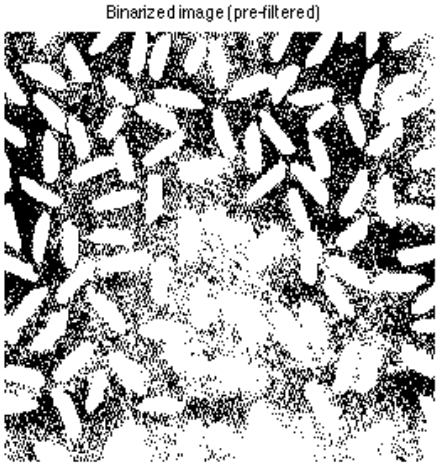


Figure 3.24 and 3.25: Histogram (left) and binarization (right) of the filtered image corresponding to threshold=0.35.

9. Select the threshold value that binarizes correctly the original image. Use the histogram in Fig. 3.24 as a guideline. A good binarization should provide an image in which pixels of rice grains are white, and the rest of pixels black.

10. Compare the histogram of y[m,n] in Fig. 3.24 with that of the original image x[m,n] in Fig. 3.18. What did it happen to the pixels that contributed to the disappeared central peak of Fig. 3.18?
11. Process again the original image x[m,n] using a shorter filter h_i[m,n]. Consider that L=15 and select the best possible binarization threshold. Compare the result with the one obtained in question 9.

To conclude, a counting algorithm is applied to the binarized images obtained in Fig. 3.19 (with no filtering) and in the Fig. 3.25 (with pre-filtering), using in both cases the optimal binarization threshold. The objective is to detect the 101 rice grains appearing in the original image. If the illumination of the original image is not compensated (Fig. 3.19), the algorithm counts 83 rice grains. On the other hand, if the original image is pre-filtered to compensate for the non-uniform illumination, the algorithms counts 100 rice grains. Results are obtained by means of the following Matlab code:

```
% Counting algorithm applied to the binarized image WITHOUT pre-processing
[regions_bin,regions,num]=counting_algorithm(x,z);
sprintf('Number of rice grains without pre-processing =%3d',num)
figure(26),imshow(regions_bin); title('Segmentation without pre-processing');
figure(27),imshow(regions); title('Segmentation without pre-processing');

% Counting algorithm applied to the binarized image WITH pre-processing
[regions_bin2,regions2,num2]=counting_algorithm(x,z2);
sprintf('Number of rice grains with pre-processing =%3d',num2)
figure(28), imshow(regions_bin2); title('Segmentation with pre-processing');
figure(29), imshow(regions2); title('Segmentation with pre-processing');
```

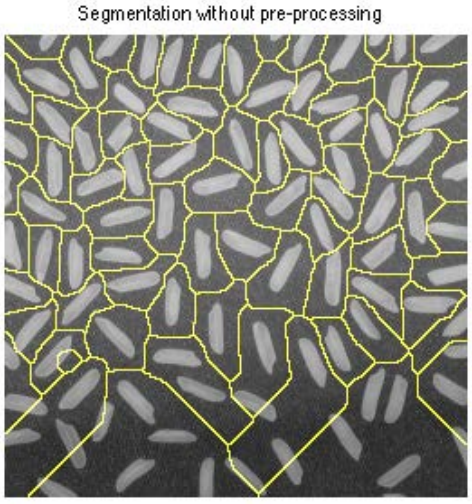


Figure 3.26 and 3.27: Segmentation of the original image without pre-processsing.
The set of regions where rice grains are localized are shown on the binarized and the original images, respectively.

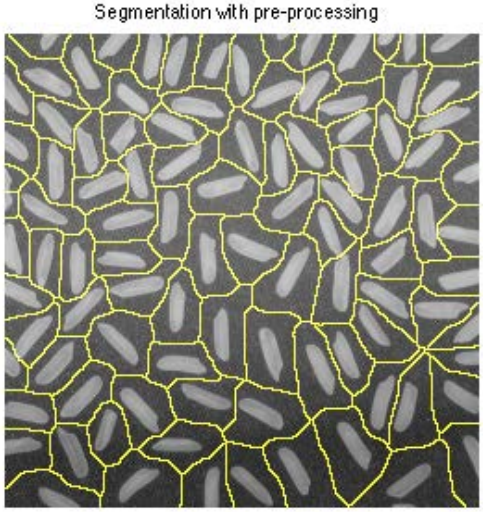


Figure 3.28 and 3.29: Segmentation of the original image with pre-processsing.
The set of regions where rice grains are localized are shown on the binarized and the original images, respectively.