Politecnico di Milano
Master of Science program in Biomedical Engineering

Biomedical Image Processing Lab class

(5 credits)

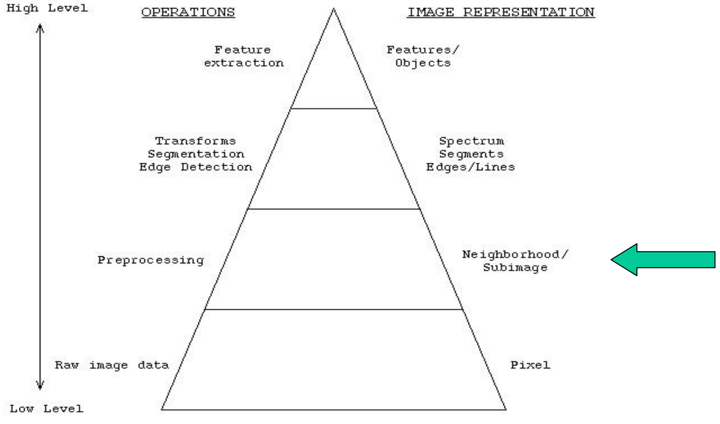
Class III

March 30th 2012

Enrico Caiani, PhD

Image Analysis Pyramid

- Hierarchical image pyramid: Consists of levels for processing of images



High Level

OPERATIONS

IMAGE REPRESENTATION

Feature extraction

Features/ Objects

Transforms
Segmentation
Edge Detection

Spectrum
Segments
Edges/Lines

Preprocessing

Neighborhood/ Subimage

Raw image data

Pixel

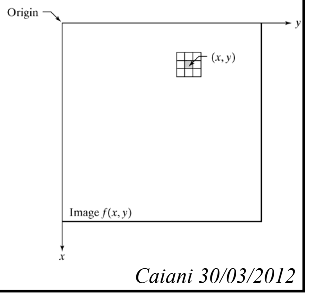
Low Level

NEIGHBORHOOD PROCESSING

It consists of several steps:

1) Selecting a center point (x,y)
2) Performing an operation that involves only the pixels in a predefined neighborhood about (x,y)
3) Letting the results of the operation be the “response” of the process at that point (x,y): g(x,y)=T[f(x,y)]
4) Repeating the process for every point in the image.

If the computations performed on the pixels of the neighborhood are linear, the operation is called **linear spatial filtering** (spatial convolution), otherwise it is called **nonlinear spatial filtering**.



Spatial Filtering

The reasons for spatial filtering are related to the need to improve image quality by removing noise, or modify its content for specific processing, based on neighborhood operations.

2D convolution is at the base of linear systems theory. It results in a filtered image g(x,y), starting from the original image f(x,y) and a 2D filter mask (kernel) w(x,y):

g(x,y)=f(x,y)*w(x,y)

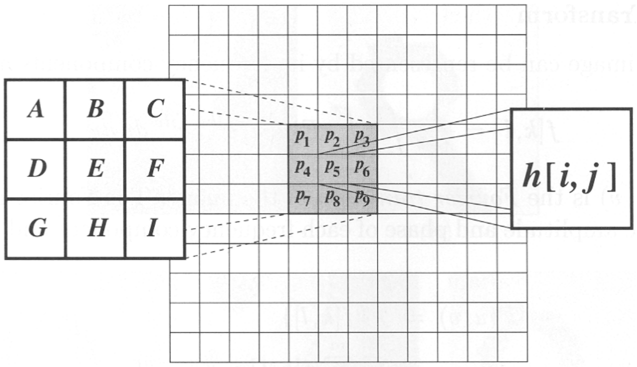
Continuous Case

$$g(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')w(x-x',y-y')dx'dy'$$

Discrete Case

$$g(i,j)=\sum_{k=1}^{n}\sum_{l=1}^{m} f(k,l)w(i-k,j-l)$$

Convolution



$$h(i,j) = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9$$

- Overlay the mask on the image
- Multiply the coincident terms
- Sum all the results
- Move to the next pixel, across the entire image

CONVOLUTION IN MATLAB

In Matlab, 2D convolution is performed by **conv2:**

For example, given the image f and the filter w=[4 -3 1;4 6 2], convolution in Matlab is performed using these steps:

1) The mask w is rotated by 180°

h = **rot90**(w,2); rotates w by 90*2

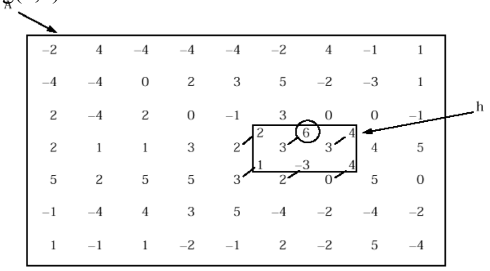
$$h = \begin{matrix} 2 & 6 & 4 \\ 1 & -3 & 4 \end{matrix}$$

2) The central pixel of h is determined as:
centro=***floor***((***size***(h)+1)/2)

size(h)= 2, 3
+1= 3, 4
/2= 1.5, 2.0
floor= 1, 2
Floor rounds to the integer closer to -∞

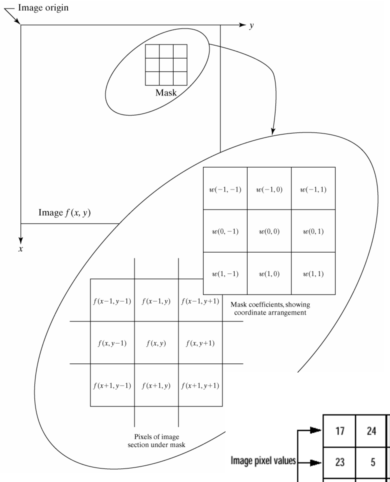
3) The value of g is computed by overimposing the h central pixel to each pixel of f, and performing sum of products between corresponding pixels..

g(4,6) = 2*2 + 3*6 + 3*4 + 3*1 + 2*–3 + 0*4 = 31



–2	4	–4	–4	–4	–2	4	–1	1
–4	–4	0	2	3	5	–2	–3	1
2	–4	2	0	–1	3	0	0	–1
2	1	1	3	2	3	3	4	5
5	2	5	5	3	2	0	5	0
–1	–4	4	3	5	–4	–2	–4	–2
1	–1	1	–2	–1	2	–2	5	–4

CORRELATION AND CONVOLUTION



In convolution, the mask w is passed on the image array f after a 180 degrees rotation.

```
A = [17  24   1   8  15
     23   5   7  14  16
      4   6  13  20  22
     10  12  19  21   3
     11  18  25   2   9]
```

```
h = [8   1   6
     3   5   7
     4   9   2]
```

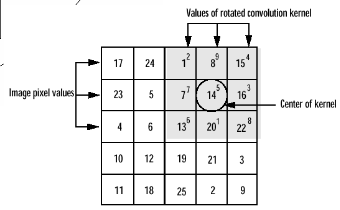


Figure 7-1: Computing the (2,4) Output of Convolution

$1 \cdot 2 + 8 \cdot 9 + 15 \cdot 4 + 7 \cdot 7 + 14 \cdot 5 + 16 \cdot 3 + 13 \cdot 6 + 20 \cdot 1 + 22 \cdot 8 = 575$

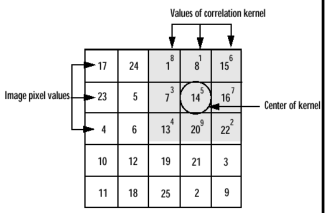


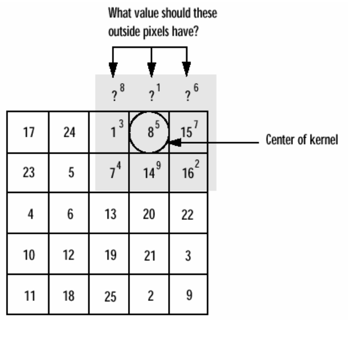
Figure 7-2: Computing the (2,4) Output of Correlation

$1 \cdot 8 + 8 \cdot 1 + 15 \cdot 6 + 7 \cdot 3 + 14 \cdot 5 + 16 \cdot 7 + 13 \cdot 4 + 20 \cdot 9 + 22 \cdot 2 = 585$

Caiani 30/03/2012

Padding

Overimposing the filter mask to the image, based on its central pixel, some elements of the filter fall out of the original image.



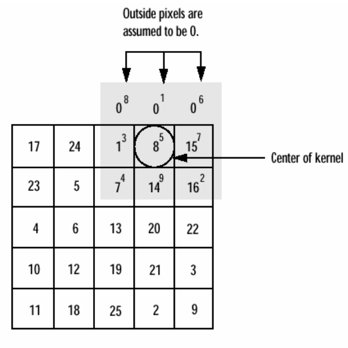


Figure 7-4: Zero-Padding of Outside Pixels

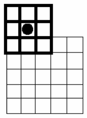
The command g=**conv2**(f,w,'shape'), as well as the command g=**imfilter**(f,w,'shape') or g=**xcorr2**(f,w,'shape'), in order to be able to compute the values correspondent to the image borders, generates additional zeros around the image f (**zero-padding**).

Caiani 30/03/2012

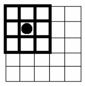
There are different options:

1. **'full'** (default): the output image g is the result of the convolution applied with zero-padding to allow every possible mask position.

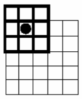
N.B.: Only with this option active, the commutative property of the convolution is maintained.



2. **'valid'**: the output image g is made by those filter mask positions that don’t require zero padding for convolution.

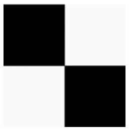


3. **'same'**: the output image g has the same dimensions of the input image f. Zero-padding is applied only by overimposing the filter mask to the position of the pixels of f.



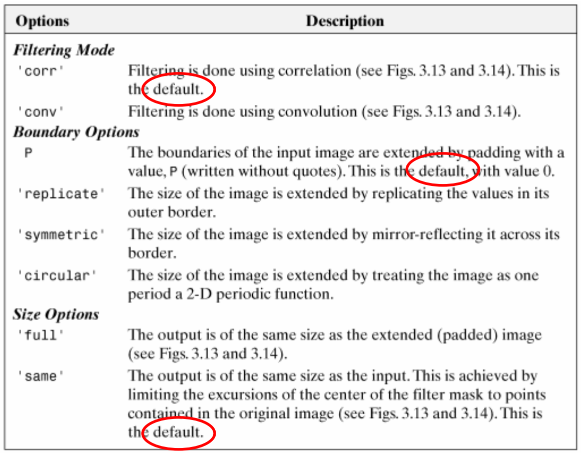
The optimal solution would be to replicate the border pixels and to apply the option ‘valid’.

Create the image A (512x512) and double, as 4 equal quadrants of zeros and ones (see picture). Generate the kernel k=ones(31), and convolute A with k using the different options for zero padding, observing the differences.



Caiani 30/03/2012

The general comman to execute spatial linear filtering is:
G= **imfilter** (f,w,filtering_mode, boundary_options, size_options)



Options	Description
Filtering Mode	
`'corr'`	Filtering is done using correlation (see Figs. 3.13 and 3.14). This is the default.
`'conv'`	Filtering is done using convolution (see Figs. 3.13 and 3.14).
Boundary Options	
`P`	The boundaries of the input image are extended by padding with a value, `P` (written without quotes). This is the default, with value 0.
`'replicate'`	The size of the image is extended by replicating the values in its outer border.
`'symmetric'`	The size of the image is extended by mirror-reflecting it across its border.
`'circular'`	The size of the image is extended by treating the image as one period a 2-D periodic function.
Size Options	
`'full'`	The output is of the same size as the extended (padded) image (see Figs. 3.13 and 3.14).
`'same'`	The output is of the same size as the input. This is achieved by limiting the excursions of the center of the filter mask to points contained in the original image (see Figs. 3.13 and 3.14). This is the default.

TABLE 3.2 Options for function `imfilter`.

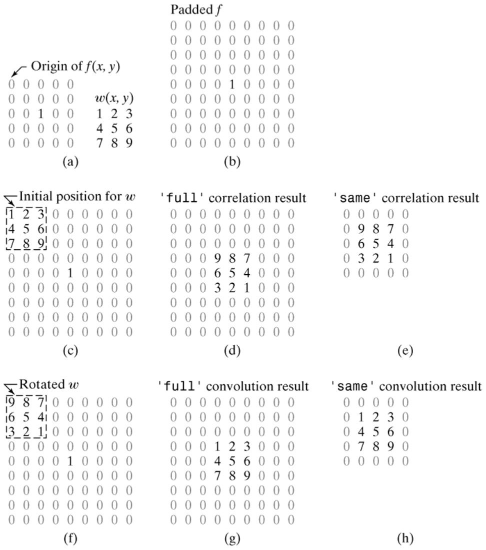
Using A and w generated in the previous example, try the different options, in particular using ‘corr’ or ‘conv’, and comment the results.

Caiani 30/03/2012

Applying correlation with a 180° rotated mask gives the same result than applying convolution using the original non-rotated mask:

G=**imfilter**(f,w,’conv’,’replicate’)

G=**rot90**(w,2),’replicate’) is preceded by: G=**imfilter**(f,**rot90**(w,2),’replicate’)


Origin of $f(x, y)$
(a)
Padded f
(b)
Initial position for w
(c)
'full' correlation result
(d)
'same' correlation result
(e)
Rotated w
(f)
'full' convolution result
(g)
'same' convolution result
(h)

(a) Origin of $f(x, y)$; $w(x, y)$

```
0 0 0 0 0
0 0 0 0 0    w(x, y)
0 0 1 0 0    1 2 3
0 0 0 0 0    4 5 6
0 0 0 0 0    7 8 9
```

(b) Padded f

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

(c) Initial position for w

```
1 2 3 0 0 0 0 0 0
4 5 6 0 0 0 0 0 0
7 8 9 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

(d) 'full' correlation result

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 9 8 7 0 0 0
0 0 0 6 5 4 0 0 0
0 0 0 3 2 1 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

(e) 'same' correlation result

```
0 0 0 0 0
0 9 8 7 0
0 6 5 4 0
0 3 2 1 0
0 0 0 0 0
```

(f) Rotated w

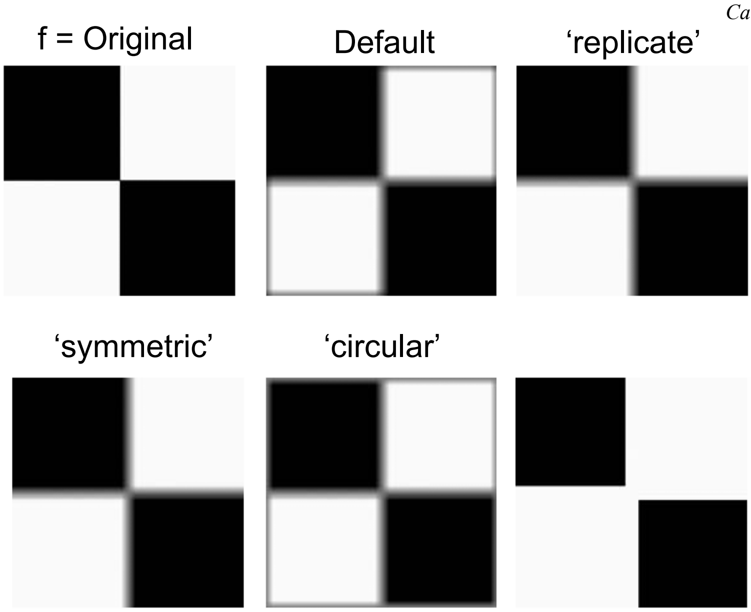
```
9 8 7 0 0 0 0 0 0
6 5 4 0 0 0 0 0 0
3 2 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

(g) 'full' convolution result

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 1 2 3 0 0 0
0 0 0 4 5 6 0 0 0
0 0 0 7 8 9 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

(h) 'same' convolution result

```
0 0 0 0 0
0 1 2 3 0
0 4 5 6 0
0 7 8 9 0
0 0 0 0 0
```

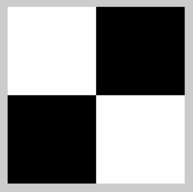


f8=**im2uint8**(f)
g8r=**imfilter**(f8,w,’replicate’)

Again on zero-padding…

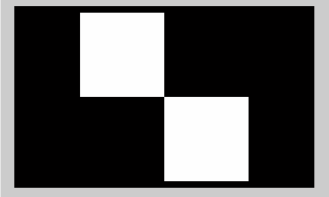
Caiani 30/03/2012

Other than using zero-padding with **conv2, xcorr2** or **imfilter**, it is possible to perform it using the command **padarray**:
B=**padarray**(A,PADSIZE) adds zeroes along the k-dimension of A:



B=**padarray**(A,[10 100]);

A (256,256)   B (276,456)



B=**padarray**(A,PADSIZE,PADVAL) adds PADVAL along the k-dimension of A:
B=**padarray**(A,PADSIZE, PADVAL, DIRECTION)
with DIRECTION='pre' 'post' 'both'
B=**padarray**(A,PADSIZE, METHOD, DIRECTION)
with METHOD='circular' 'replicate' 'symmetric'

Load spect.jpg. Convert it into intensity image. Apply padarray, testing the different options, resulting into a new image with sizes equal to power of 2.

NON-LINEAR SPATIAL FILTERING

Also non linear spatial filtering is based on neighborhood operations, and on the sliding of a filter mask through an image.
However, it is based on non linear operations involving pixels in the niehborhood (i.e., the maximum pixel value).

In Matlab, two functions provide general nonlinear filtering: **nlfilter** and **colfilt**.
The former performs in 2D, the latter organizes the data in the form of columns and it is more computationally efficient.

B = **nlfilter**(A,[m n],FUN) applies the function FUN to each m-by-n sliding block of A. FUN is a function that accepts an m-by-n matrix as input and returns a scalar:

B = **nlfilter**(A,[3 3],@myfun);   where **function** scalar = myfun(x)
scalar = **median**(x(:));

*To the previous image, apply a non linear filter to blocks of [11 11], that compute the log of the **sum of** videointensity.*

Caiani 30/03/2012

NON-LINEAR SPATIAL FILTERING

B = **colfilt**(f,[m n],’sliding’,fun)

Given f (MxN) and a mask of mxn, it generates a matrix of maximum size mnxMN, in which each column corresponds to the pixels encompassed by the neighborhood centered at that location in the image (i.e., first column: mask centered at top left pixel).

fun is a function that must operate on each of the column and return a row vector

When using colfilt, padding has to be performed explicitly before filtering.

Example:

```
I = imread('tire.tif');
imview(I)
I2 = uint8(colfilt(I,[5 5],'sliding',@mean));
imview(I2)
```

Caiani 30/03/2012

NON-LINEAR SPATIAL FILTERING

It is possible to generate other nonlinear spatial filters by **ordfilt2**, that generates rank filters. The filter response is based on the ranking operation on the elements of the image contained in the filter mask:

G=**ordfilt2**(F,ORDER,DOMAIN);

with ORDER equal to the rank of the element to substitute in the central pixel, and DOMAIN defining the mask from size.

G=**ordfilt2**(F,1,**ones**(3,3))
filters F sobstituting to the central pixel the first value of the ordered 9-eleemnts sequence (i.e., the minumum value).

Besides such command, there is also **medfilt2**, that applies directly the operation with the 50th percentile in the sequence (median value) :

G=**medfilt**2(F,[M N], PADOPT) [3x3, ‘zeros’]
where MxN indicate the dimensions of the mask, and PADOPT can be ‘zeros’, ‘symmetric’ or ‘indexed’.
Such filter is particularly suited to remove “salt and pepper” noise without modifying the contours.

Load an image, and try to apply this kind of filtering with different parameters values.

Caiani 30/03/2012

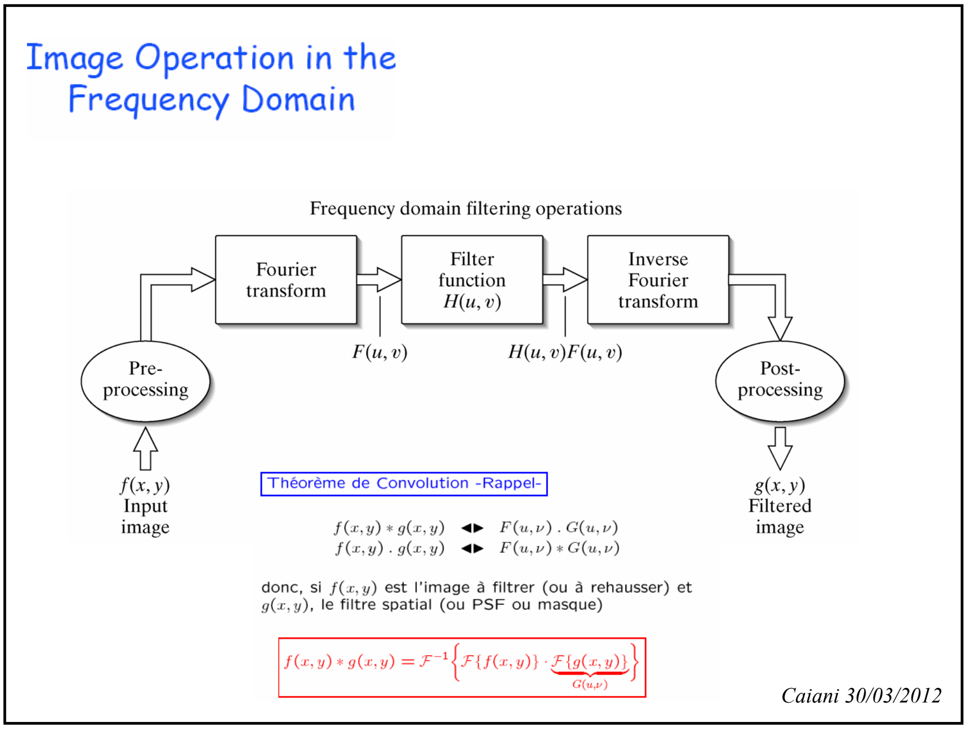


Image Operation in the Frequency Domain

Frequency domain filtering operations

Pre- processing → Fourier transform → Filter function $H(u, v)$ → Inverse Fourier transform → Post- processing

$F(u, v)$

$H(u, v)F(u, v)$

$f(x, y)$ Input image

$g(x, y)$ Filtered image

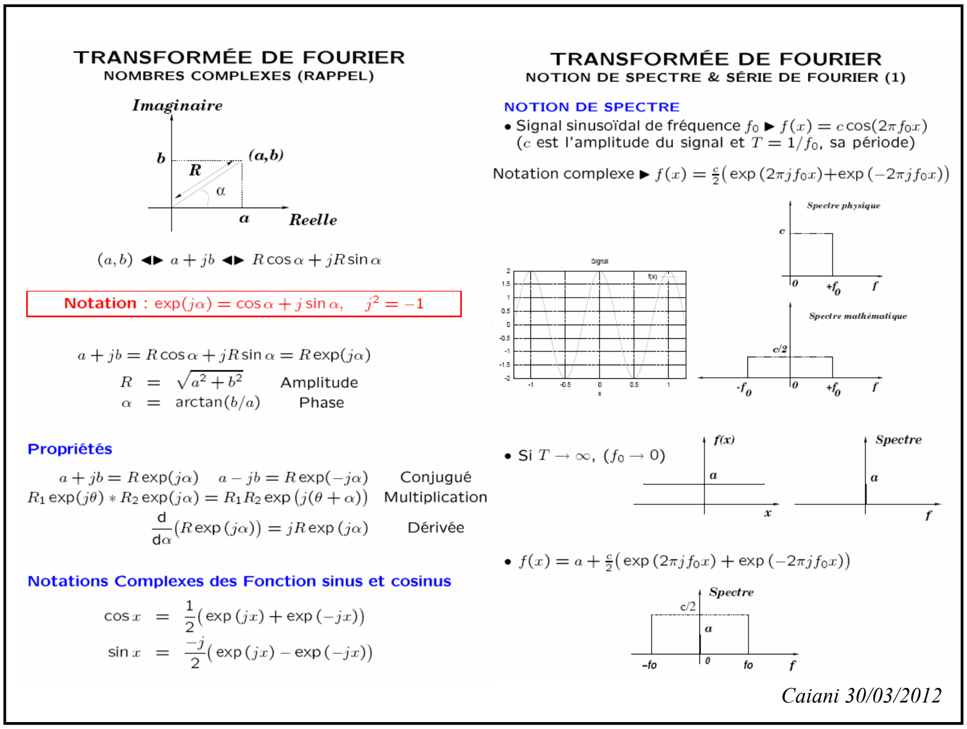
Théorème de Convolution -Rappel-

$$f(x,y) * g(x,y) \quad ◀▶ \quad F(u,\nu)\,.\,G(u,\nu)$$
$$f(x,y)\,.\,g(x,y) \quad ◀▶ \quad F(u,\nu) * G(u,\nu)$$

donc, si $f(x,y)$ est l'image à filtrer (ou à rehausser) et $g(x,y)$, le filtre spatial (ou PSF ou masque)

$$\boxed{f(x,y) * g(x,y) = \mathcal{F}^{-1}\Big\{\mathcal{F}\{f(x,y)\}\cdot\underbrace{\mathcal{F}\{g(x,y)\}}_{G(u,\nu)}\Big\}}$$

Caiani 30/03/2012



TRANSFORMÉE DE FOURIER

NOMBRES COMPLEXES (RAPPEL)

$$(a, b) \;◀▶\; a + jb \;◀▶\; R\cos\alpha + jR\sin\alpha$$

$$\boxed{\text{Notation : } \exp(j\alpha) = \cos\alpha + j\sin\alpha, \quad j^2 = -1}$$

$$a + jb = R\cos\alpha + jR\sin\alpha = R\exp(j\alpha)$$
$$R = \sqrt{a^2 + b^2} \qquad \text{Amplitude}$$
$$\alpha = \arctan(b/a) \qquad \text{Phase}$$

Propriétés

$$a + jb = R\exp(j\alpha) \quad a - jb = R\exp(-j\alpha) \qquad \text{Conjugué}$$
$$R_1\exp(j\theta) * R_2\exp(j\alpha) = R_1R_2\exp\big(j(\theta+\alpha)\big) \qquad \text{Multiplication}$$
$$\frac{d}{d\alpha}(R\exp(j\alpha)) = jR\exp(j\alpha) \qquad \text{Dérivée}$$

Notations Complexes des Fonction sinus et cosinus

$$\cos x = \frac{1}{2}\big(\exp(jx) + \exp(-jx)\big)$$
$$\sin x = \frac{-j}{2}\big(\exp(jx) - \exp(-jx)\big)$$

TRANSFORMÉE DE FOURIER

NOTION DE SPECTRE & SÉRIE DE FOURIER (1)

NOTION DE SPECTRE

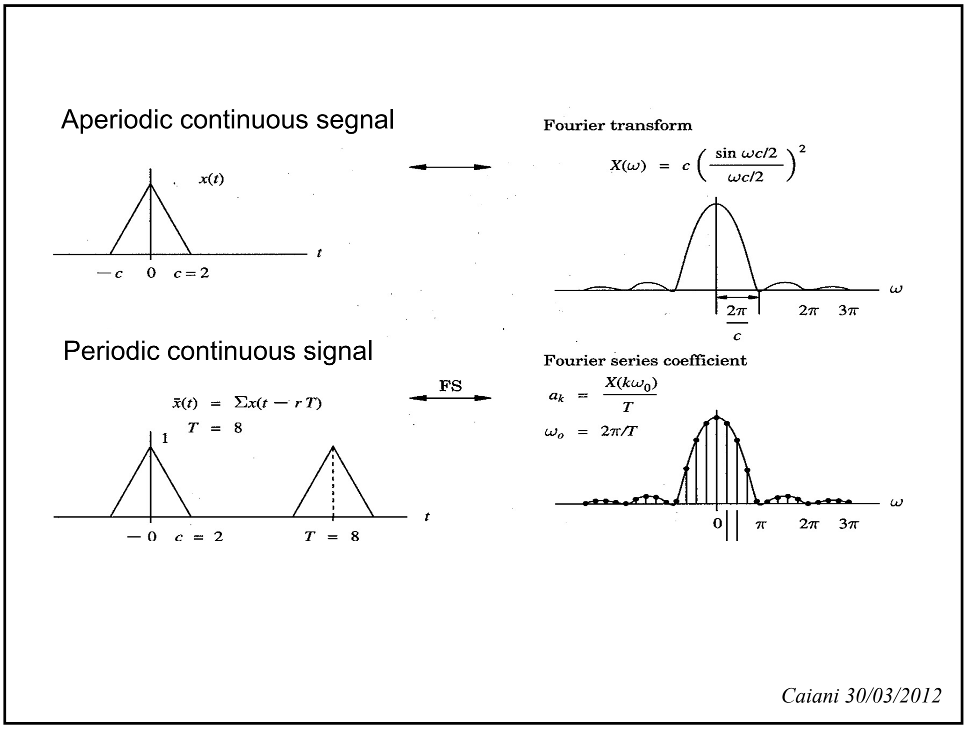
- Signal sinusoïdal de fréquence f_0 ▶ $f(x) = c\cos(2\pi f_0 x)$ (c est l'amplitude du signal et $T = 1/f_0$, sa période)

Notation complexe ▶ $f(x) = \frac{c}{2}\big(\exp(2\pi j f_0 x) + \exp(-2\pi j f_0 x)\big)$

- Si $T \to \infty$, $(f_0 \to 0)$

- $f(x) = a + \frac{c}{2}\big(\exp(2\pi j f_0 x) + \exp(-2\pi j f_0 x)\big)$

Caiani 30/03/2012



Aperiodic continuous segnal

Fourier transform

$$X(\omega) \;=\; c\left(\frac{\sin \omega c/2}{\omega c/2}\right)^2$$

Periodic continuous signal

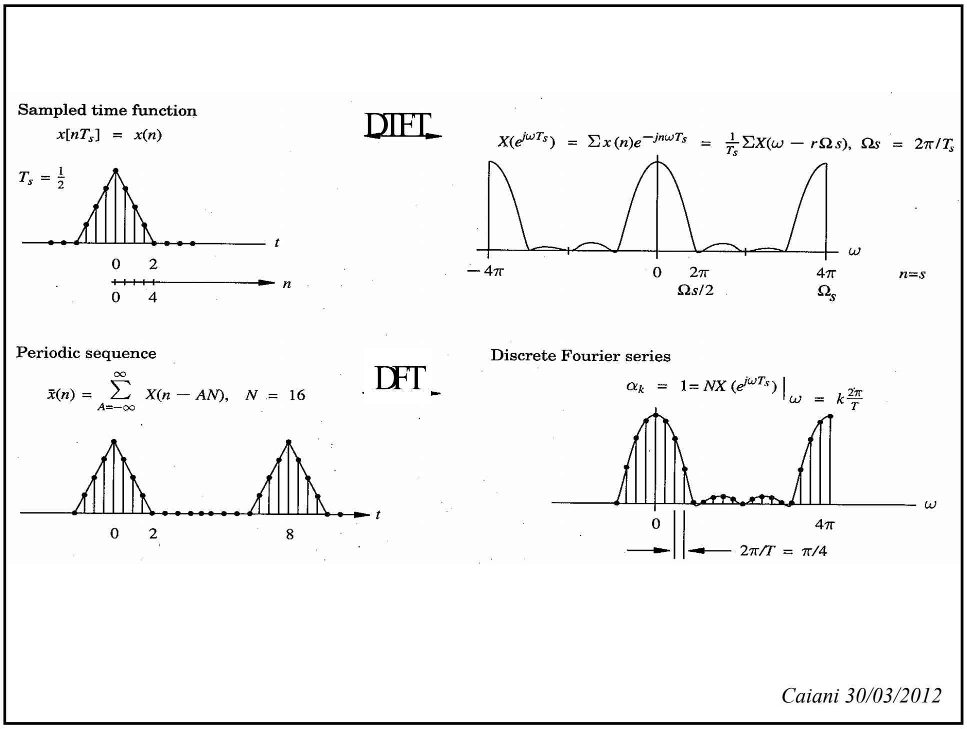
$$\tilde{x}(t) \;=\; \Sigma x(t - rT)$$
$$T \;=\; 8$$

FS

Fourier series coefficient

$$a_k \;=\; \frac{X(k\omega_0)}{T}$$
$$\omega_o \;=\; 2\pi/T$$

Caiani 30/03/2012



Sampled time function

$$x[nT_s] \;=\; x(n)$$

DTFT

$$X(e^{j\omega T_s}) \;=\; \Sigma x(n) e^{-jn\omega T_s} \;=\; \frac{1}{T_s}\Sigma X(\omega - r\Omega s),\;\; \Omega s \;=\; 2\pi/T_s$$

$$T_s = \tfrac{1}{2}$$

Periodic sequence

$$\tilde{x}(n) = \sum_{A=-\infty}^{\infty} X(n - AN), \quad N \;=\; 16$$

DFT

Discrete Fourier series

$$\alpha_k \;=\; 1 = NX\left(e^{j\omega T_s}\right)\Big|_{\omega \;=\; k\frac{2\pi}{T}}$$

Caiani 30/03/2012