Física Experimental VI – 4300314

2º Semestre de 2017

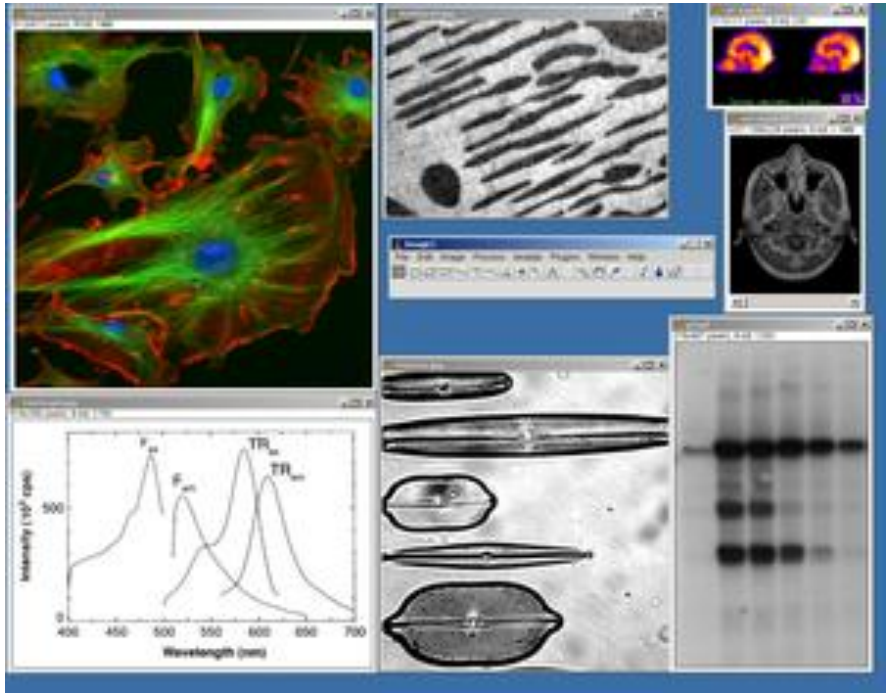
Instituto de Física
Universidade de São Paulo

Professor: Antonio Domingues dos Santos

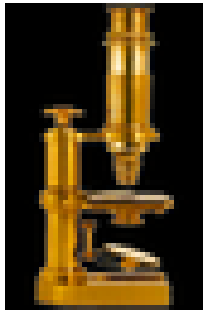
E-mail: adsantos@if.usp.br
Fone: 3091.6886

04 de outubro

Processamento de imagens com ImageJ



Screenshot of ImageJ



Developer(s)	Wayne Rasband (NIH)
Stable release	1.51k / 22 March 2017[1]
Repository	github.com/imagej/imagej1
Operating system	Any (Java-based)
Type	Image processing
License	Public Domain
Website	imagej.nih.gov/ij/

Convolução

,

$$c = a \otimes b = a * b$$

Em 2D , espaço contínuo: $$c(x,y) = a(x,y) \otimes b(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} a(\chi,\zeta) b(x-\chi, y-\zeta)\, d\chi\, d\zeta$$

Em 2D, espaço discreto: $$c[m,n] = a[m,n] \otimes b[m,n] = \sum_{j=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} a[j,k] b[m-j, n-k]$$

Propriedades da Convolução

* Convolução é *comutativa*. $$c = a \otimes b = b \otimes a$$

* Convolução é *associativa*. $$c = a \otimes (b \otimes c) = (a \otimes b) \otimes c = a \otimes b \otimes c$$

* Convolução é *distributiva*. $$c = a \otimes (b + d) = (a \otimes b) + (a \otimes d)$$

onde *a*, *b*, *c*, e *d* são imagens, contínuas ou discretas.

Linear Filters

Uniform filter - The output image is based on a local averaging of the input filter where all of the values within the filter support have the same weight.
Examples for (J=K=5) and (R=2.5) :

Rectangular case

$$h_{rect}[j,k] = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Circular case

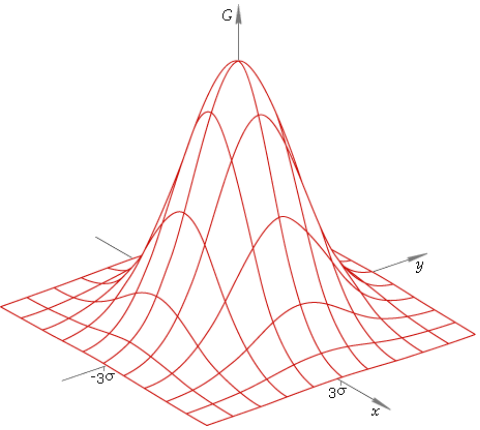
$$h_{circ}[j,k] = \frac{1}{21}\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Pyramidal filter

$$h_{rect}[j,k] = \frac{1}{81}\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

Cone filter

$$h_{circ}[j,k] = \frac{1}{25}\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 1 & 2 & 5 & 2 & 1 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$



Gaussian filter - The use of the Gaussian kernel for smoothing has become extremely popular.

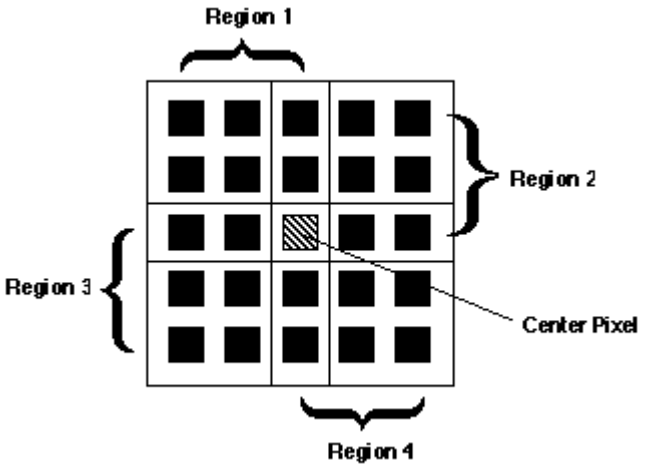
$$h(x,y) = g_{2D}(x,y) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\left(x^2/2\sigma^2\right)}\right) \bullet \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\left(y^2/2\sigma^2\right)}\right)$$

$$= g_{1D}(x) \bullet g_{1D}(y)$$

Non-Linear Filters

Median filter - A median filter is based upon moving a window over an image (as in a convolution) and computing the output pixel as the median value of the brightnesses within the input window. If the window is *J* x *K* in size we can order the *J*K* pixels in brightness value from smallest to largest. If *J*K* is odd then the median will be the (*J*K*+1)/2 entry in the list of ordered brightnesses. Note that the value selected will be exactly equal to one of the existing brightnesses so that no roundoff error will be involved if we want to work exclusively with integer brightness values.

Kuwahara filter - Edges play an important role in our perception of images as well as in the analysis of images. As such it is important to be able to smooth images without disturbing the sharpness and, if possible, the position of edges. A filter that accomplishes this goal is termed an *edge-preserving filter* and one particular example is the Kuwahara filter . Although this filter can be implemented for a variety of different window shapes, the algorithm will be described for a square window of size $J = K = 4L + 1$ where L is an integer. The window is partitioned into four regions as shown below.



In each of the four regions (i=1,2,3,4), the mean brightness, m_i in eq. , and the *variance*$_i$, s_i^2 in eq. , are measured. The output value of the center pixel in the window is the mean value of that region that has the smallest variance.

Examples of the effect of various smoothing algorithms

Original

Uniforme 5x5

Gaussiano (R=2,5)

Mediana 5x5

Kawahara 5x5

First Derivatives

Gradient filters - It is also possible to generate a vector derivative description as the *gradient,* $\nabla a[m,n]$, of an image:

$$\nabla a = \frac{\partial a}{\partial x}\vec{i}_x + \frac{\partial a}{\partial y}\vec{i}_y = (h_x \otimes a)\vec{i}_x + (h_y \otimes a)\vec{i}_y$$

where

This leads to two descriptions:

Gradient magnitude -

$$|\nabla a| = \sqrt{(h_x \otimes a)^2 + (h_y \otimes a)^2}$$

and

Gradient direction -

$$\psi(\nabla a) = \arctan\left\{ (h_y \otimes a) \Big/ (h_x \otimes a) \right\}$$

The gradient magnitude is sometimes

Approx. Gradient magnitude -

$$|\nabla a| \cong |h_x \otimes a| + |h_y \otimes a|$$

Prewitt gradient filters :

$$[\mathbf{h}_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$[\mathbf{h}_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Sobel gradient filters :

$$[\mathbf{h}_x] = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$[\mathbf{h}_y] = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Examples of the effect of various *derivative algorithms* on a noisy figure.
The effect of various *magnitude gradient algorithms* are shown.

Simple Derivative em X

Sobel

Gaussian

Simple Gradiente

Sobel

Gaussian

Second Derivatives

It is, of course, possible to compute higher-order derivatives of functions of two variables. In image processing, the second derivatives or Laplacian play an important role. The Laplacian is defined as:

$$\nabla^2 a = \frac{\partial^2 a}{\partial x^2} + \frac{\partial^2 a}{\partial y^2} = (h_{2x} \otimes a) + (h_{2y} \otimes a)$$

Where h_{2x} and h_{2y} are second derivative filters.

Basic second derivative filter - This filter is specified by:

$$[\mathbf{h}_{2x}] = [\mathbf{h}_{2y}]^t = [1 \quad -2 \quad 1]$$

two-dimensional filter (Laplacian) is:

$$[\mathbf{h}] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The effects of the second derivative filters are

Laplacian

+ Gaussian

Unsharp masking

A well-known technique from photography to improve the visual quality of an image is to enhance the edges of the image. The technique is called ***unsharp masking***. Edge enhancement means first isolating the edges in an image, amplifying them, and then adding them back into the image.

$$f_{sharp}(x,y) = f(x,y) - k * \nabla^2 f(x,y)$$

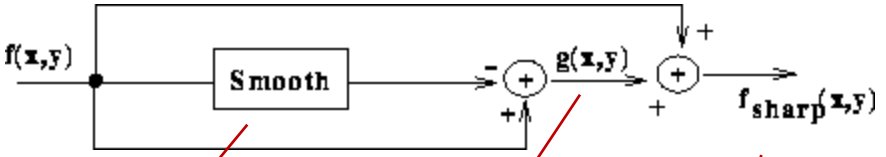
The term k is the amplifying term and $k > 0$. The effect of this technique is shown in Figure 48. The Laplacian used to produce Figure 48 is given by eq. (120) and the amplification term $k = 1$.

Original **Laplacian-enhanced**

Unsharp masking

The origin:

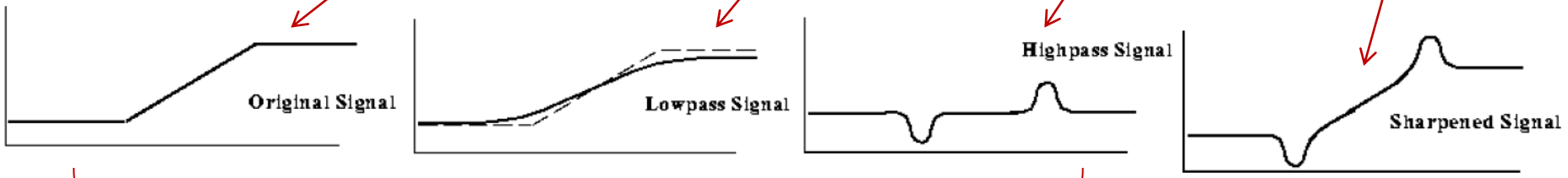
The unsharp mask filter can be defined by the equations:



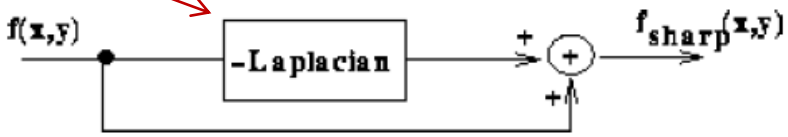
$$f_{sharp}(x,y) = f(x,y) + k * g(x,y)$$

$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

where *k* is a scaling constant. Reasonable values for *k* vary between 0.2 and 0.7, with the larger values providing increasing amounts of sharpening.



Alternative method => Laplacian



$$f_{sharp}(x,y) = f(x,y) - k * \nabla^2 f(x,y)$$

Original **Laplacian-enhanced**

Convolution in the frequency domain

there is an alternative method to implement the filtering of images through convolution. Based on eq. it appears possible to achieve the same result as in eq. by the following sequence of operations:

i) Compute $A(\Omega,\Psi) = F\{a[m,n]\}$

ii) Multiply $A(\Omega,\Psi)$ by the *precomputed* $(\Omega,\Psi) = F\{h[m,n]\}$

iii) Compute the result $c[m,n] = F^{-1}\{A(\Omega,\Psi)*(\Omega,\Psi)\}$

➔ **Transformada de Fourier (FFT)**
(próxima aula!)