

Filtrado Espacial

Contenido

Filtrado Espacial

Suavizado

Filtros basados en derivadas de la
función Gaussiana

Mejoramiento de la nitidez

Filtros Espaciales – 1/2

- Modifican la contribución de ciertos rangos de frecuencia (bajas, medianas, altas)
- Se aplican directamente a la imagen (espacio) y no a una transformada de ella (frecuencia)
- El nivel de gris de un pixel se obtiene de los valores de sus vecinos
- El filtrado se realiza por convolución de la imagen con los filtros espaciales

Filtros Espaciales – 2/2

Categorías según los rangos de frecuencia :

- **Filtros Paso-Bajas (LPF) , Smoothing Filters**
 - Reducción de ruido
 - Suavizado
 - Pérdida de nitidez
- **Filtros Paso-Banda (BPF)**
 - Detección de patrones de ruido
 - Eliminan demasiado contenido de la imagen
- **Filtros Paso-Altas (HPF) , Sharpening Filters**
 - Detección de cambios de luminosidad
 - Detección de patrones (bordes y contornos)
 - Resaltado de detalles finos

Suavizado

Suavizado

- Filtros de bloque
 - (máscaras = kernels)
- Difuminado (blurring)
- Filtros binomiales (orden 0)

Filtros de Bloque – 1/4

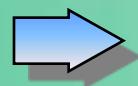
Máscara o kernel : Matriz que representa el filtro

- Al aplicar la convolución el filtrado de cada pixel coincide con la posición del valor central de la máscara (mask)
- El filtrado es función de los vecinos (bloque) alrededor del pixel central a filtrar
- El filtrado corresponde a la suma de productos entre los valores de la máscara y los valores de los pixels para cada posición de la máscara

Filtros de Bloque – 2/4

Características de una máscara o kernel :

- Sus valores se llaman coeficientes
- Filtros paso-bajas o filtros paso-banda : La suma de sus coeficientes debe ser uno (1)



Normalización

- Filtros paso-altas : La suma de sus coeficientes debe ser cero (0)

Filtros de Bloque – 3/4

Aplicación Iterativa

Convolucionar iterativamente N veces una imagen con un filtro de tamaño M corresponde a aplicar una sola convolución de un filtro de tamaño L :

$$L = 2 \times ((M-1) / 2) \times N + 1$$

$2 \times$ un filtro de tamaño 3 = 1 x un filtro de tamaño 5
 $3 \times$ un filtro de tamaño 3 = 1 x un filtro de tamaño 7

$$(f \otimes h) \otimes g = f \otimes (h \otimes g)$$

$$2 \times g_3 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \iff \quad g_5 = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Filtros de Bloque – 4/4

Separabilidad

El filtro Gaussiano y el filtro promediador son separables

$$f \otimes g = (f \otimes g_x) \otimes g_y = (f \otimes g_y) \otimes g_x$$

$$G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad Gy = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad Gx = \frac{1}{4} [1 \ 2 \ 1] \quad \text{ya que} \quad \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times [1 \ 2 \ 1] = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$B = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad Bx = \frac{1}{5} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad By = \frac{1}{5} [1 \ 1 \ 1 \ 1 \ 1] \quad \text{ya que} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \times [1 \ 1 \ 1 \ 1 \ 1] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Ventaja : **Complejidad** computational para un filtro $M \times N$:

Implementación no separable: $M \times N$ operaciones

Implementation separable: $M + N$ operaciones

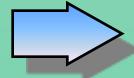
Filtro Promediador (blur) – 1/2

- El filtrado corresponde a la convolución con el siguiente kernel :

$$g = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$g = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

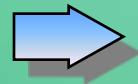
$$F(x,y) = f \otimes g \Leftrightarrow F(x,y) = \frac{1}{9} \sum_{u=-1}^{u=1} \sum_{v=-1}^{v=1} f(x+v, y+u)$$



Incrementando tamaño (blur more)

$$g = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Kernel más grande = más difuminado



¡ Complejidad computacional !

Filtro Promediador (blur) – 2/2



Imagen con ruido suavizada con un kernel de 7×7

Filtros Paso-bajas

Se usa la función Gaussiana

- Se aproxima en su forma discreta a través de los **filtros binomiales** de orden 0 ($N = 0$)
- Los **filtros binomiales** se especifican para distintos tamaños o longitudes L ($x = 0, 1, \dots, L$)

Caso continuo en 1D :
$$f(x) = ke^{-ax^2} \quad k = \frac{1}{2\sigma^2}$$

Filtro Binomial orden 0 – 1/3

Caso discreto en 1D :

$$f_L(x) = C_L^x = \binom{L}{x} = \frac{L!}{x!(L-x)!} \quad x = 0, 1, \dots, L$$

Triángulo de Pascal :

$$f_2(x) = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$f_3(x) = \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}$$

L				1		
1			1		1	
2		1		2		1
3		1	3		3	1
4	1	4		6	4	1

Filtro Binomial orden 0 – 2/3

Propiedades – 1/2

- Son separables en 2D : se aplica un filtro 1D en dirección x y después en dirección y
- La convolución de un filtro de tamaño L consigo mismo produce uno de tamaño $2L$:

$$f_2(x) \otimes f_2(x) = f_4(x)$$

$$[1 \ 2 \ 1] \otimes [1 \ 2 \ 1] = [1 \ 4 \ 6 \ 4 \ 1]$$

Filtro Binomial orden 0 – 3/3

Propiedades – 2/2

- En 2D se obtienen como : $[f_L(x)]^T \times [f_L(x)]$

Si $L = 2$, entonces :

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times [1 \ 2 \ 1] = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Para no alterar la luminancia en la imagen, debido a la suma, se **normalizan** los filtros :

$$f_2(x, y) = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Filtro Paso-bajas Gaussiano (soften)

Aproximación discreta de un filtro

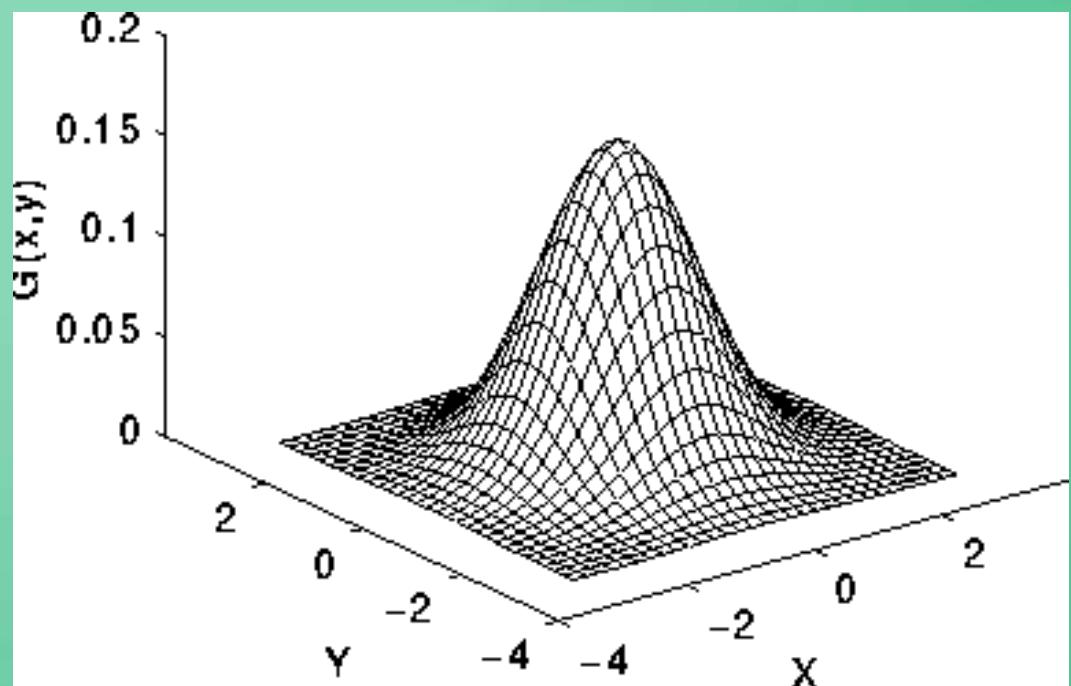
2D Gaussiano :

Suma ponderada, los pixels centrales
son más importantes que los pixels
de los bordes

$$g_3 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$g_5 = \frac{1}{246} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$G(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)}$$



Comparación : Gaussiano vs Promedio



Imagen original

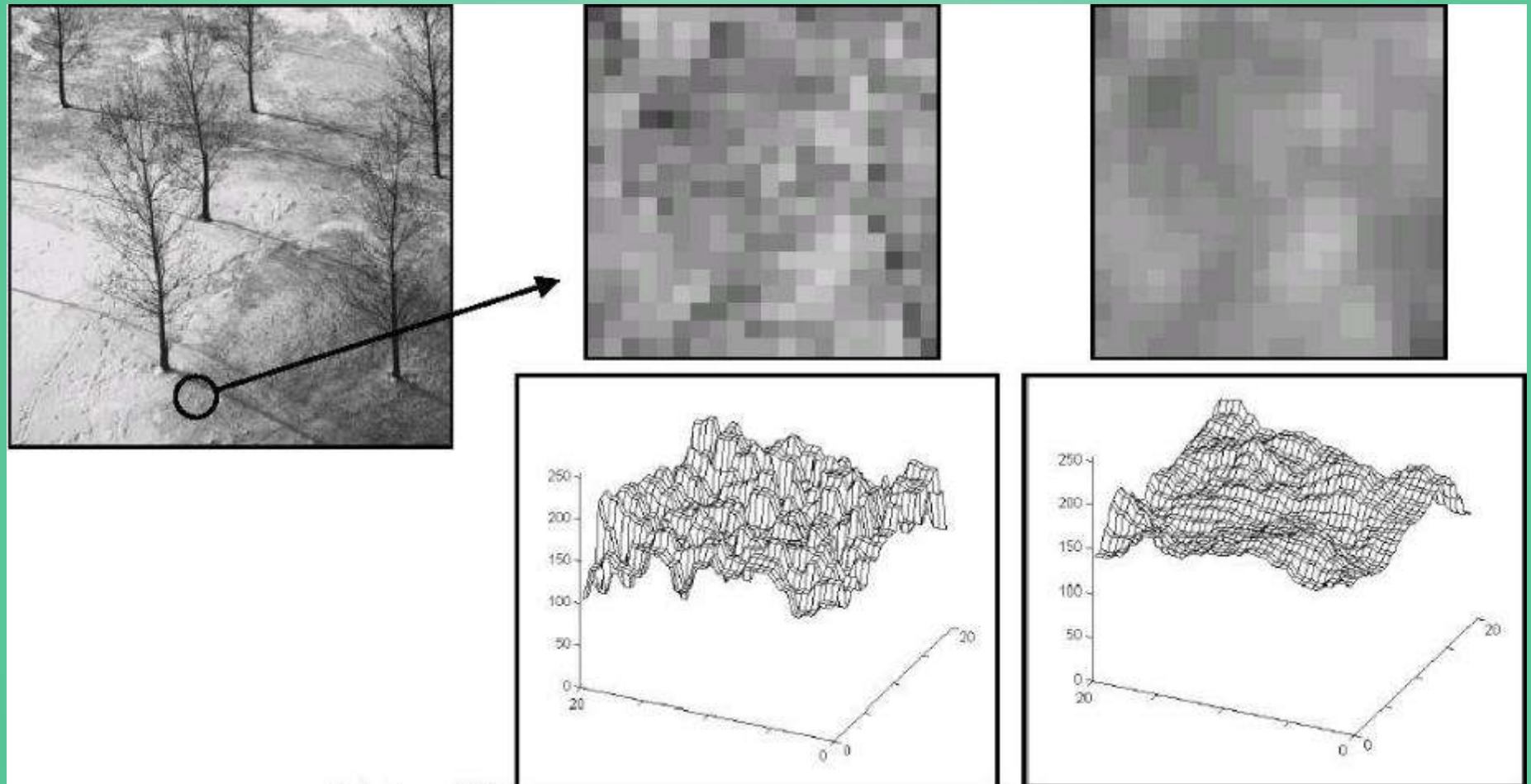


Filtro promediador

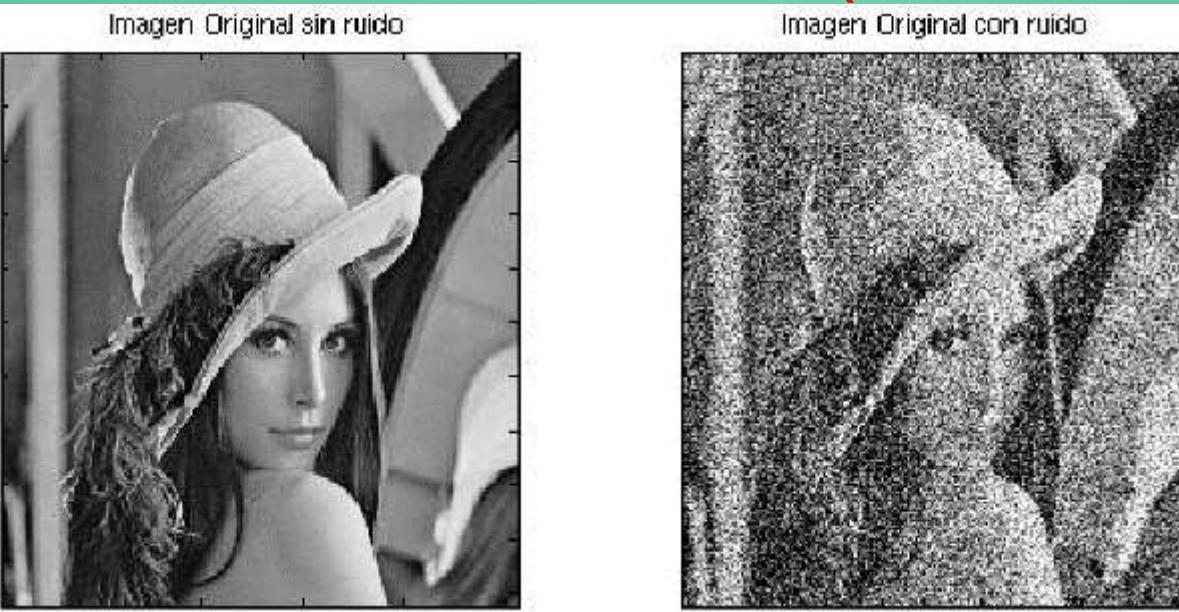


Filtro Gaussiano

Filtro Paso-bajas Gaussiano : Detalles



Filtro Binomial (soften)



Im. ruido. Filtro binomial 7x7



Imagen con ruido suavizada con un kernel de 7×7

Comparación : Binomial vs Promedio



Promediador

Binomial

Filtros basados en derivadas de la función Gaussiana

Filtros de Derivadas de Gaussianas

Filtros Gaussianos

- derivadas
- discretización

Filtros binomiales (orden N)

Detección de bordes

- Gradientes
- Filtro de Roberts
- Filtro de Prewitt
- Filtro de Sobel
- Filtro Laplaciano
- Filtro de Canny
- Filtro de Deriche

Derivadas de Filtros Gaussianos – 1/4

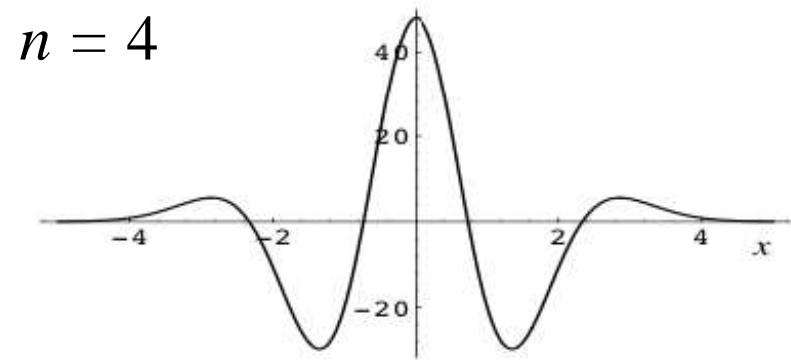
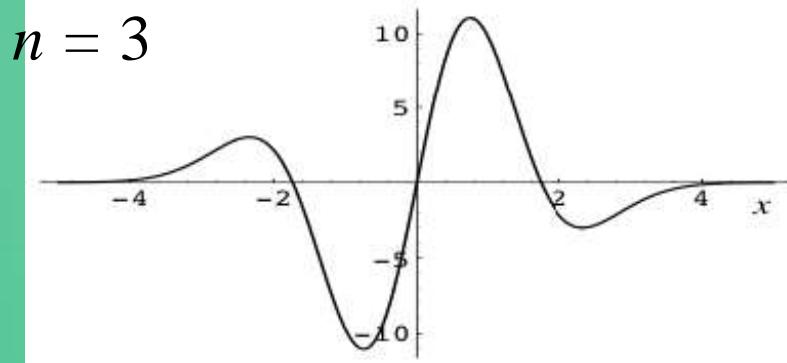
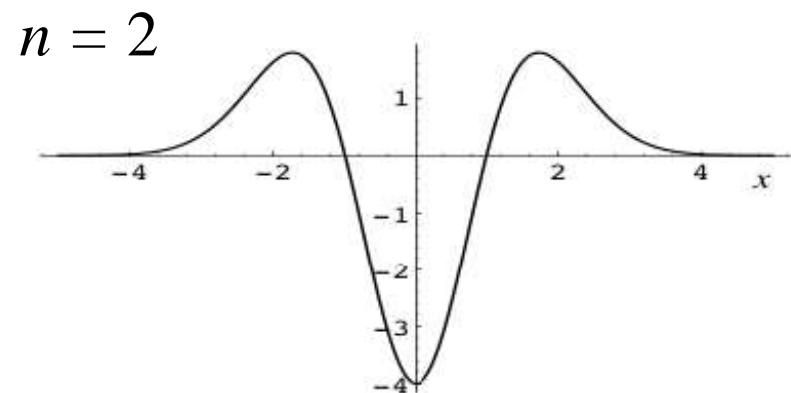
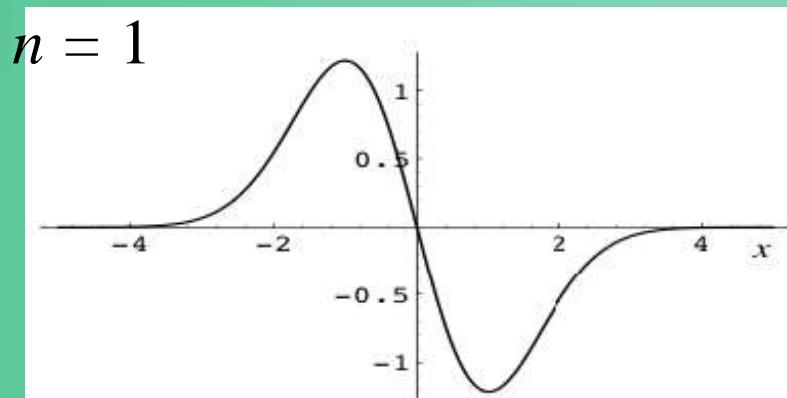
- Corresponden a filtros paso-altas
- Se especializan en la detección de cambios bruscos :
 - Bordes
 - Contornos
 - Líneas

Primeras derivadas en 1D : $f^{(0)}(x) = f(x) = ke^{-ax^2}$

$$f^{(1)}(x) = -k2a(x)e^{-ax^2} , \quad f^{(2)}(x) = k2a(2ax^2 - 1)e^{-ax^2}$$

Derivadas de Filtros Gaussianos – 2/4

Primeras cuatro derivadas de Gaussianas ($k = \sigma = 1$)



$$(n = 0, 1, \dots, N)$$

Derivadas de Filtros Gaussianos – 3/4

Discretización de las derivadas de Gaussianas

Se define en términos de diferencias (finitas)

Forma discreta de la primera derivada Gaussiana :

$$f^{(1)}(x) = f(x) - f(x-1)$$

Diseñando un filtro tal que $g(x) = h(x) \otimes f(x)$ y $g(x) = f(x) - f(x-1)$

Sabiendo que $f(x) = f(x) \otimes \delta(x)$ y $f(x-1) = f(x-1) \otimes \delta(x-1)$

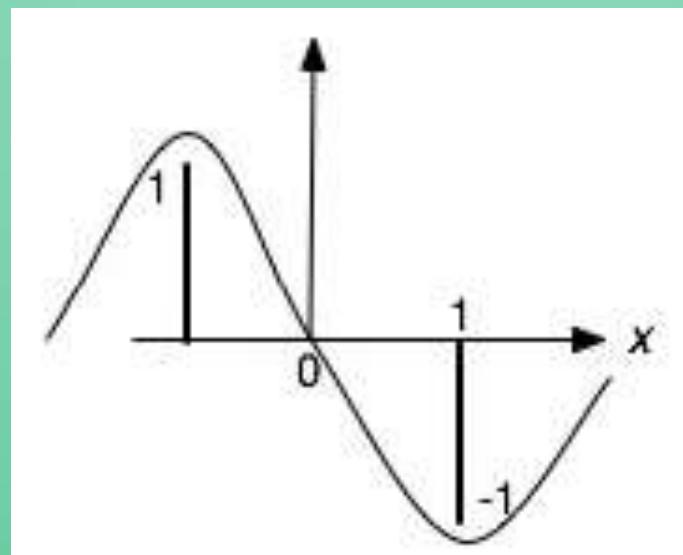
entonces : $h(x) = \delta(x) - \delta(x-1)$ ó $h(x) = [1 \quad -1]$

Derivadas de Filtros Gaussianos – 4/4

Agregando un cero a $h(x)$: $h(x) = [1 \quad 0 \quad -1]$

se puede reescribir como : $h(x) = \delta(x+1) - \delta(x-1)$

$h(x)$ posee un comportamiento discreto de la primera derivada Gaussiana



Filtro Binomial orden N

Caso discreto en 1D :

$$f_L^{(n)}(x) = \nabla^n C_{L-n}^x = \nabla^n \binom{L-n}{x} = \nabla^n \left(\frac{(L-n)!}{x!(L-n-x)!} \right)$$

$$\nabla f(x) = f(x) - f(x-1) = f^{(1)}(x) \quad x = 0, 1, \dots, L \\ n = 0, 1, \dots, N$$

Triángulo de Pascal :

(primera derivada)

$$f_2^{(1)}(x) = [1 \ 0 \ -1]$$

$$f_4^{(1)}(x) = [1 \ 2 \ 0 \ -2 \ -1]$$

	L		1			
	1		1	-1		
	2		1	0	-1	
	3		1	1	-1	-1
	4	1	2	0	-2	-1

Detección de Bordes

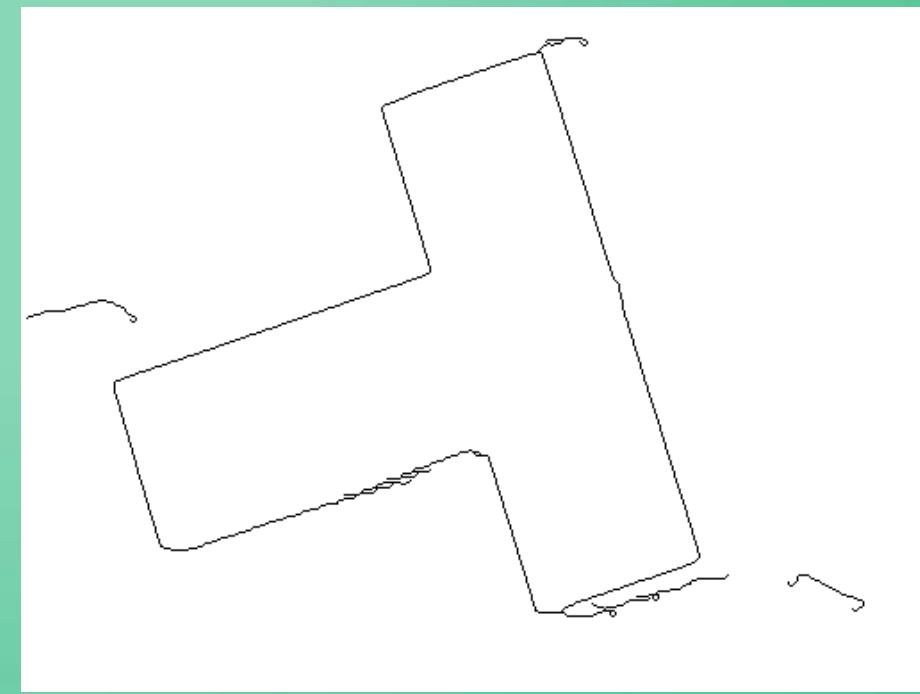
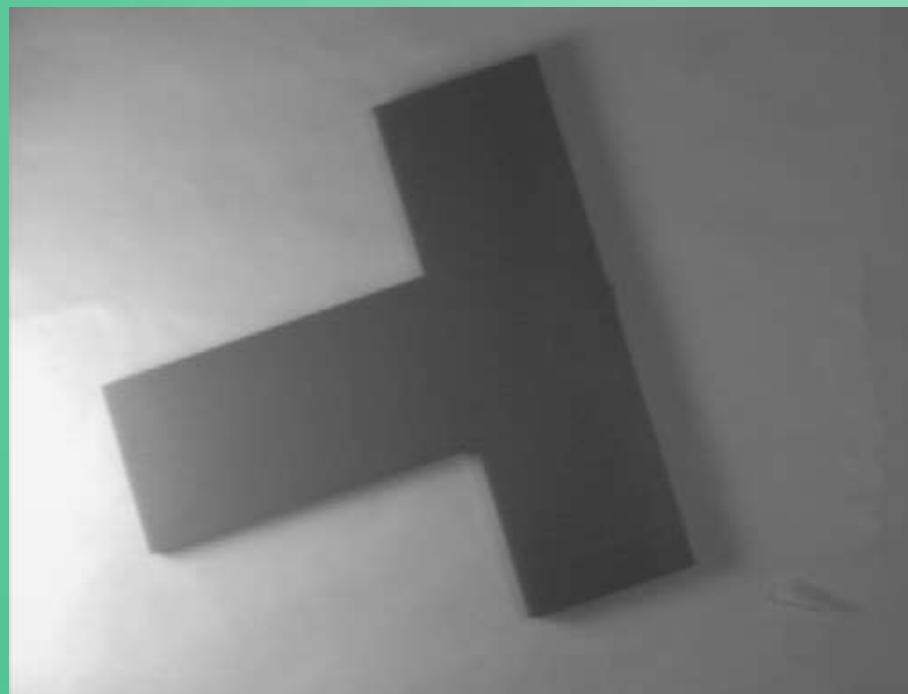
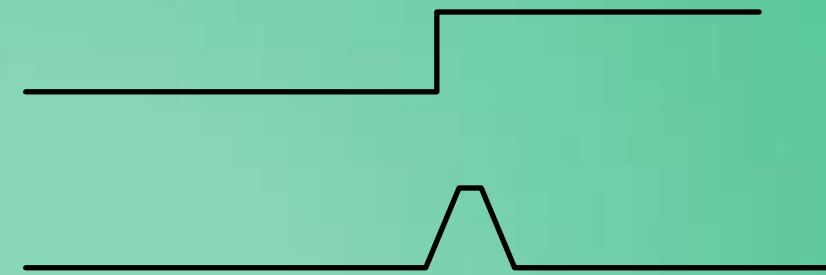
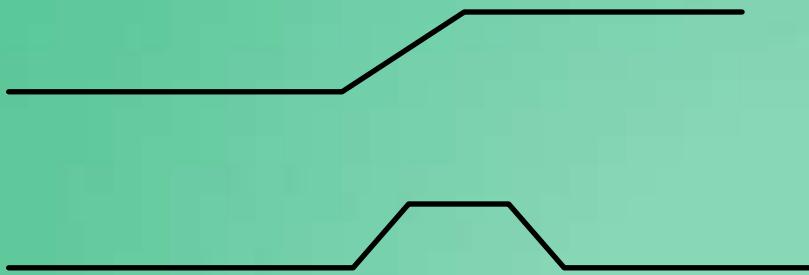
- Gradientes
- Filtro de Roberts
- Filtro de Prewitt
- Filtro de Sobel
- Filtro Laplaciano
- Filtro de Canny
- Filtro de Deriche

(Presentado en Inglés)

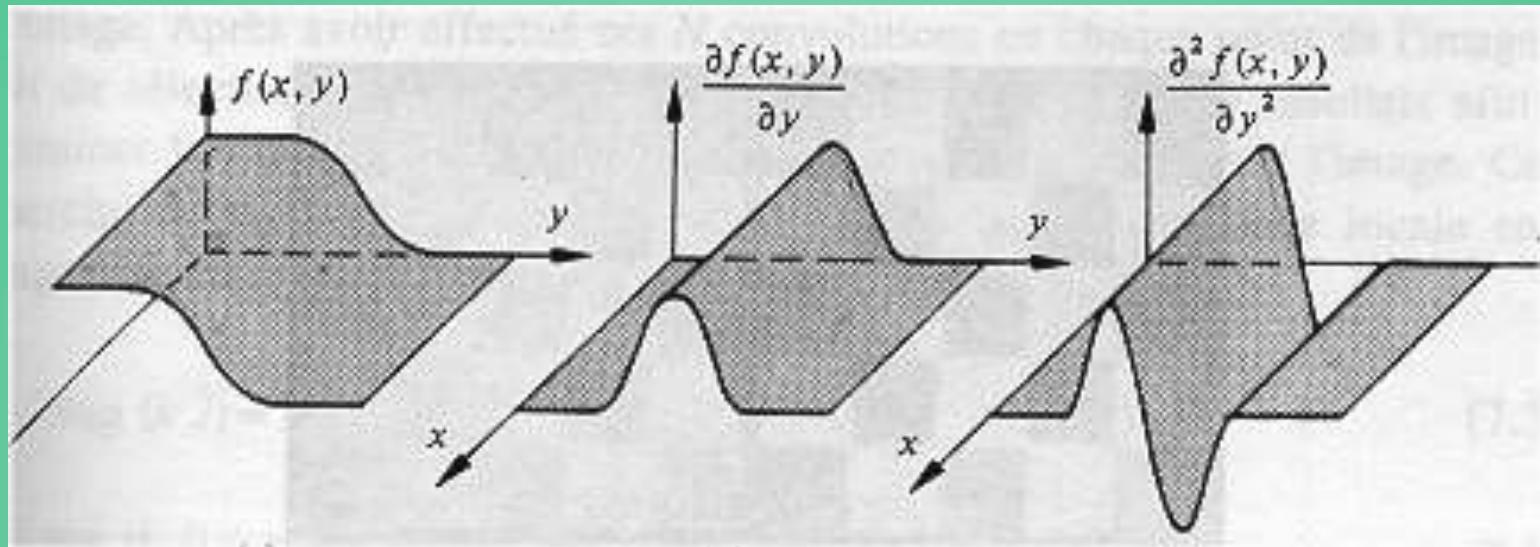
[Tomado del curso “ Vision Industrielle ” del Dr. Carlos Rivero]

Edges and Contours

Edges: changes in the image intensity



Edge detection: gradients and derivatives



Two possible approaches to edge detection:

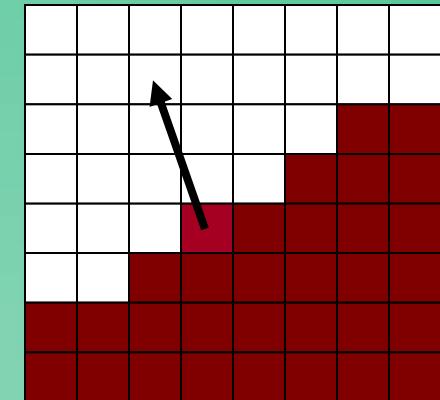
- Detection of the **maxima** of the **gradient**
- Detection of the **zero crossings** of **second derivative**

Most methods are gradient based.

The edge pixels need to be connected together to get to a sequence of edge points! (“**edge linking**”)

The gradient

The gradient is a vector with a norm and a direction:



$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\theta = \tan^{-1} \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

The partial derivatives can be computed by convolution with appropriate linear filter masks:

$$\frac{\partial f}{\partial d}(x, y) = [f \otimes h_d](x, y)$$

Gradient: alternatives

Different possible norms:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2}$$

$$G_d = \frac{\partial f}{\partial d}$$

$$|\nabla f| = |G_x| + |G_y|$$

$$|\nabla f| = \max(|G_x|, |G_y|)$$

Taking the discrete nature of an image into account:

$$|\nabla f| = \max_{d=0^\circ, 45^\circ, 90^\circ, 135^\circ} (G_d)$$

$$\phi(\nabla f) = \arg \max_{d=0^\circ, 45^\circ, 90^\circ, 135^\circ} (G_d)$$

Numerical approximation of the gradient

Theorem of Taylor LaGrange applied to an image $z=f(x,y)$

$$f(x_0 + h, y_0 + k) = f(x_0, y_0) + h \frac{\partial f}{\partial x} + k \frac{\partial f}{\partial y} + O(\sqrt{h^2 + k^2})$$

$$\frac{\partial f(x_0, y_0)}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h}$$

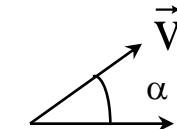
$(x_0, y_0) \longrightarrow (x_0 + h, y_0)$

(x_0, y_0)

$$\frac{\partial f(x_0, y_0)}{\partial y} = \lim_{k \rightarrow 0} \frac{f(x_0, y_0 + k) - f(x_0, y_0)}{k}$$

$(x_0, y_0 + k)$

$$\frac{\partial f(x_0, y_0)}{\partial \vec{v}} = \lim_{t \rightarrow 0} \frac{f(x_0 + t \cos \alpha, y_0 + t \sin \alpha) - f(x_0, y_0)}{t}$$



$$\frac{\partial f(x_0, y_0)}{\partial x} \cong f(x_0 + 1, y_0) - f(x_0, y_0)$$

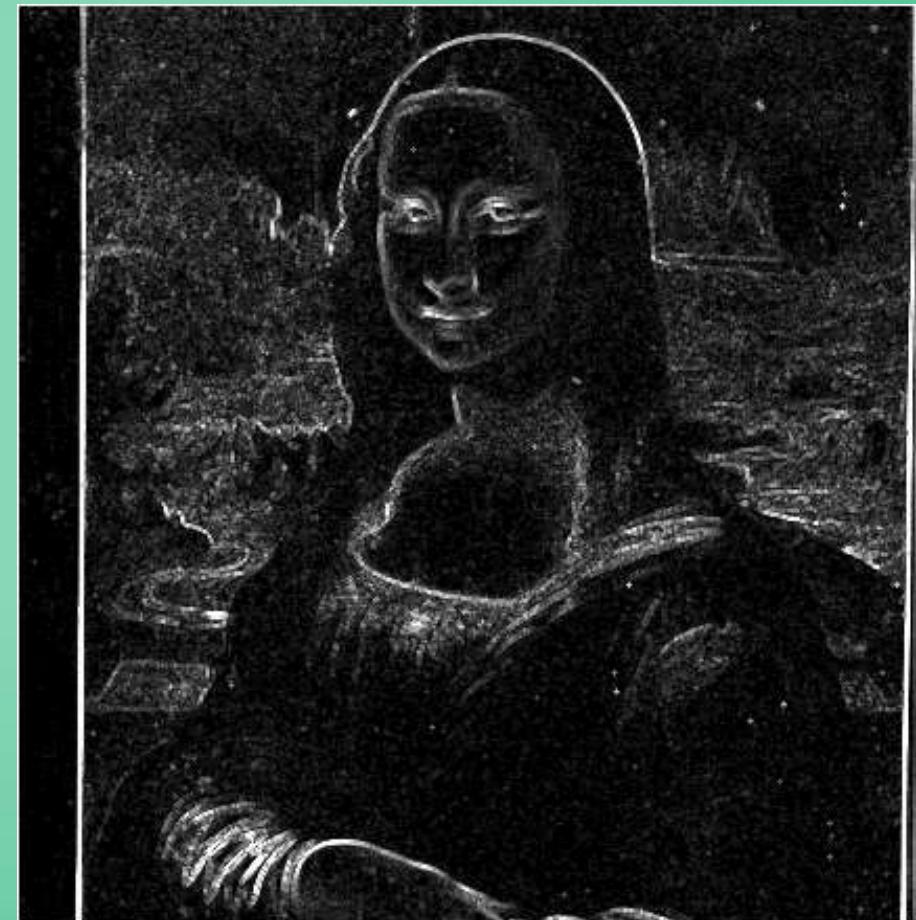
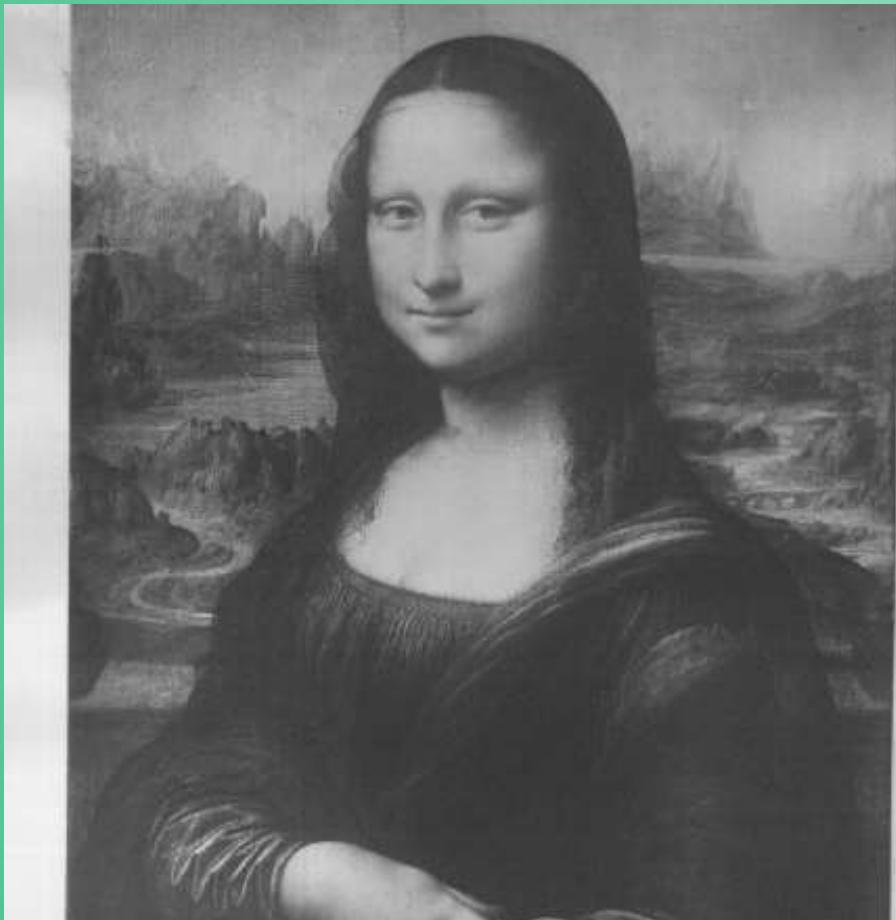
$$\frac{\partial f(x_0, y_0)}{\partial y} \cong f(x_0, y_0 + 1) - f(x_0, y_0)$$

The Roberts filter

$$\frac{\partial f}{\partial x} \cong f \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial y} \cong f \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

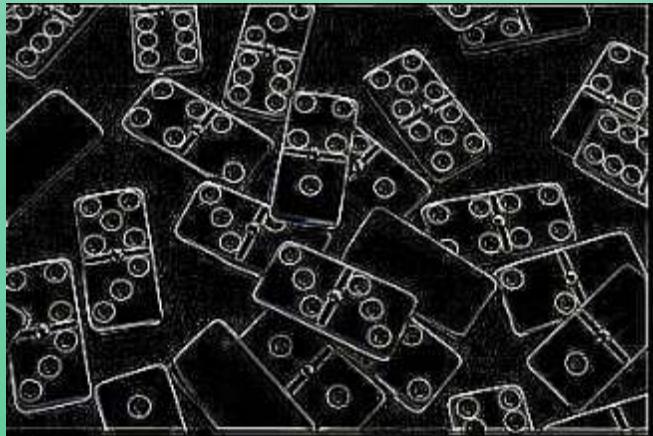
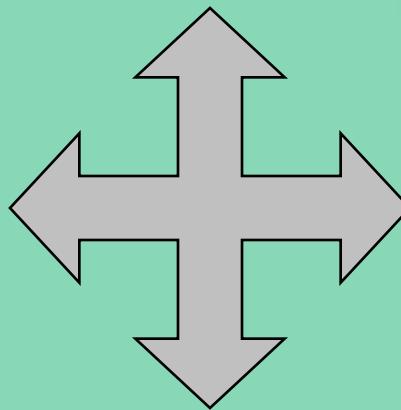
$$\frac{\partial f}{\partial \vec{v}} \cong f \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ pour } \alpha = \frac{\pi}{4}$$



$$\frac{\partial f}{\partial x} \cong f \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



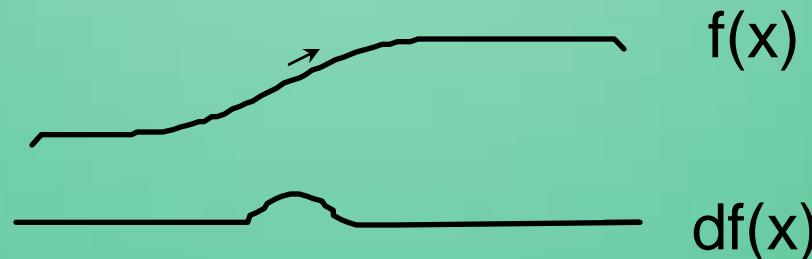
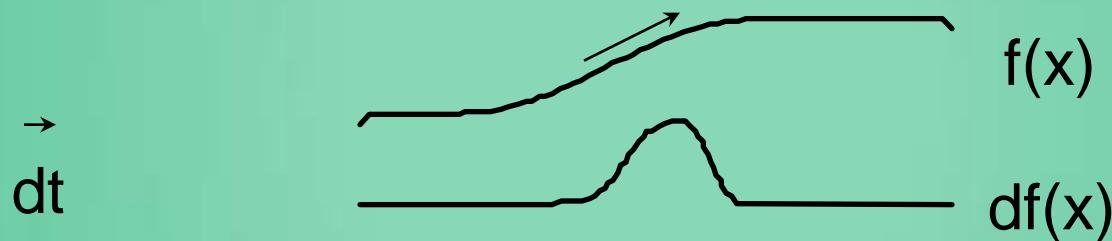
$$\frac{\partial f}{\partial y} \cong f \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$



$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Problem of edge width

Differentiation with 1 pixel allows to detect fast transitions (sharp edges) but not slower transition (blurred edges)



Increasing the differentiation step

$$\frac{\partial f}{\partial x} \approx f \otimes [1 \ 0 \ -1] \quad \frac{\partial f}{\partial y} \approx f \otimes \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad \frac{\partial f}{\partial v} \approx f \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \text{ pour } \alpha = \frac{\pi}{4}$$

The gradient can be smoothed by averaging the pixel in the neighborhood.



Prewitt gradient filter

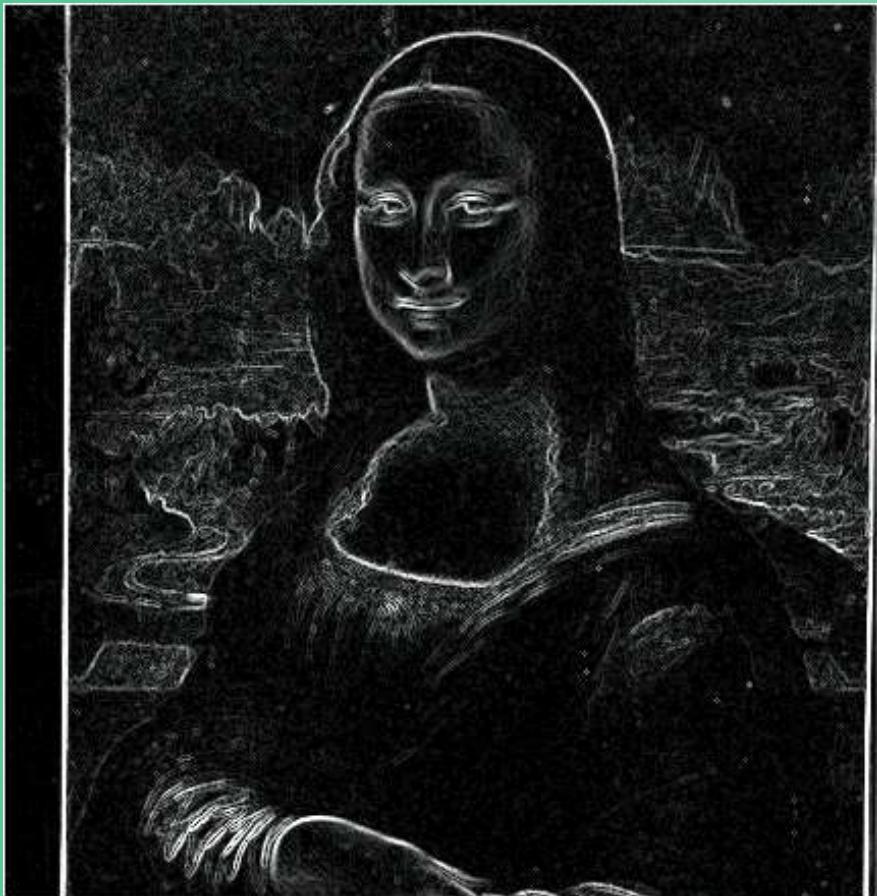
$$\frac{\partial f}{\partial x} \approx f \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial y} \approx f \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

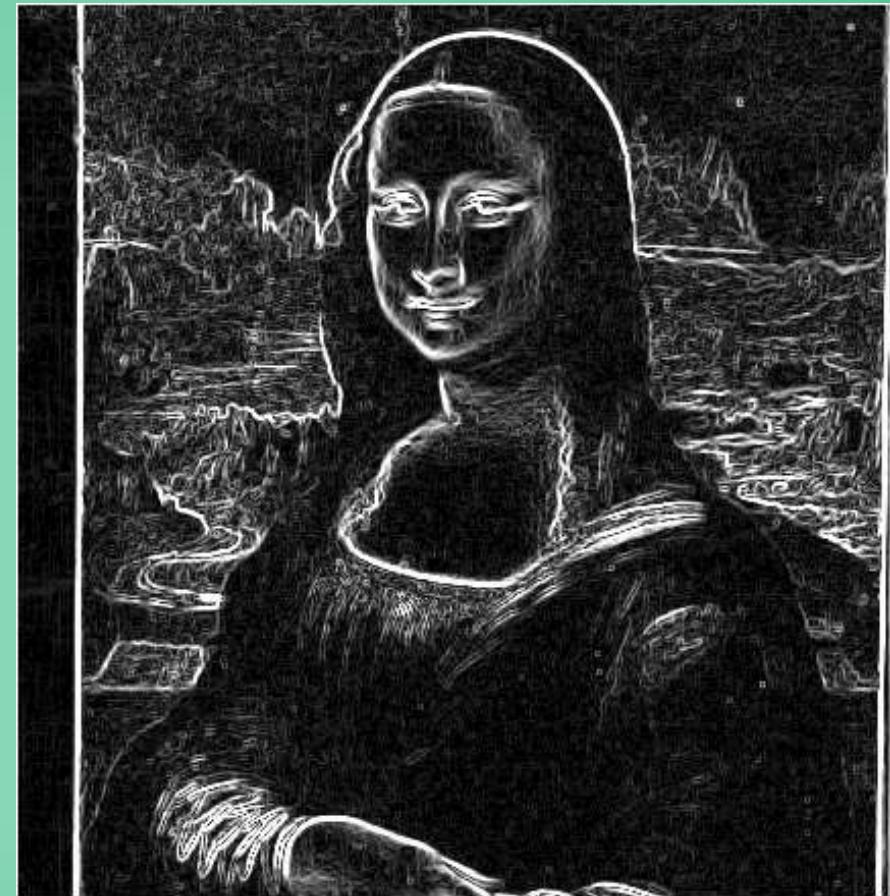
$$\frac{\partial f}{\partial v1} \approx f \otimes \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial v2} \approx f \otimes \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

Prewitt filter: example



Roberts filter



Prewitt filter

Prewitt filter: the principal edges are better detected.

Differentiation: noise suppression

$$\frac{\partial f}{\partial x} \cong f \otimes \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \frac{\partial f}{\partial y} \cong f \otimes \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial v1} \cong f \otimes \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \quad \frac{\partial f}{\partial v2} \cong f \otimes \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

Sobel filter



Gradient masks: summary

Roberts

$$\frac{\partial f}{\partial x} \cong f \otimes [1 \quad -1]$$

$$\frac{\partial f}{\partial y} \cong f \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Prewitt

$$\frac{\partial f}{\partial x} \cong f \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial y} \cong f \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel

$$\frac{\partial f}{\partial x} \cong f \otimes \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial y} \cong f \otimes \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Mask size

- A bigger mask means less **sensitivity** to noise
- A bigger mask means higher computational **complexity**
- A bigger mask means less **localization precision**

The Canny/Deriche gradient operator

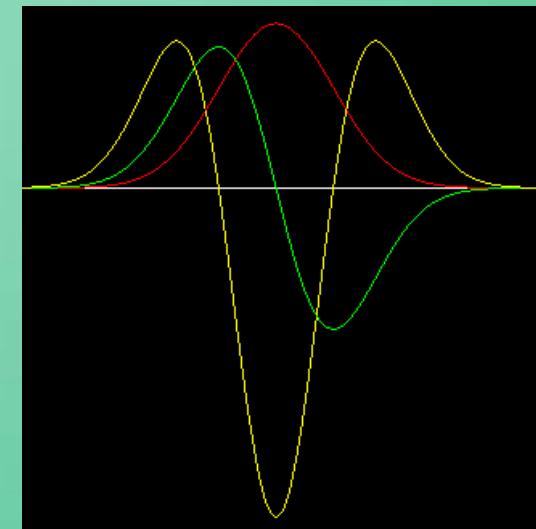


In 1983 Canny proposed 3 [criteria for edge detection](#):

- Detection quality (maximum signal to noise ratio)
- Localization precision
- Uniqueness (one response per edge)

Maximization of these criteria leads to the solution of a differential equation. The solution can be approximated by the [derivative of a Gaussian](#):

$$h(x) = -\frac{x}{\tau^2} \exp \left\{ -\frac{x^2}{2\tau^2} \right\}$$



The Deriche solution

Canny's solution has been developed for an finite impulse response filter (FIR). Deriche developed an infinite impulse response filter (IIR) from the same equations and different initial conditions:

$$h(x) = k \ x \ e^{-\frac{\alpha}{2}|x|}$$

Scale parameter

The filter is implemented as a recursive filter, i.e. the filter result of one pixel depends on the results of the preceding pixel.

A higher value of α means a higher sensitivity to detail



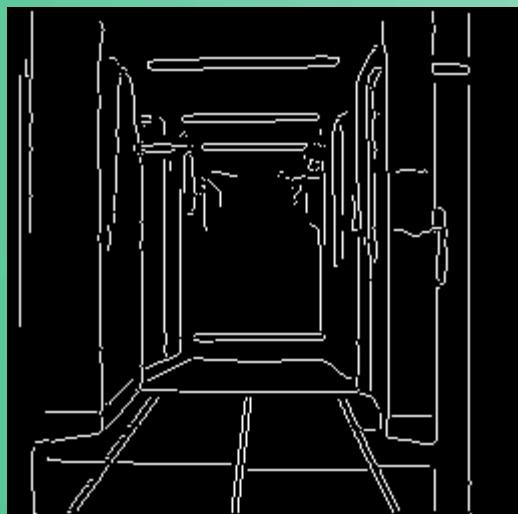
Original image



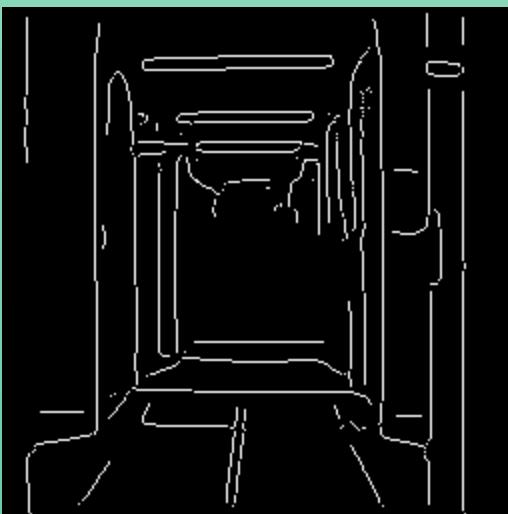
$\alpha=5$



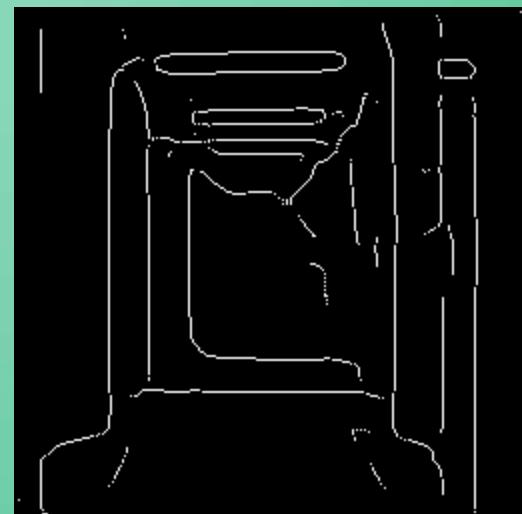
$\alpha=2$



$\alpha=1$

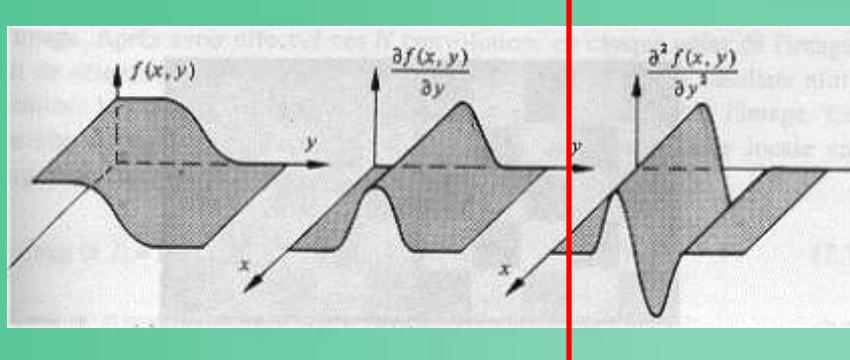


$\alpha=0.5$



$\alpha=0.25$

Zero-Crossings (the Laplacian filter)



$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

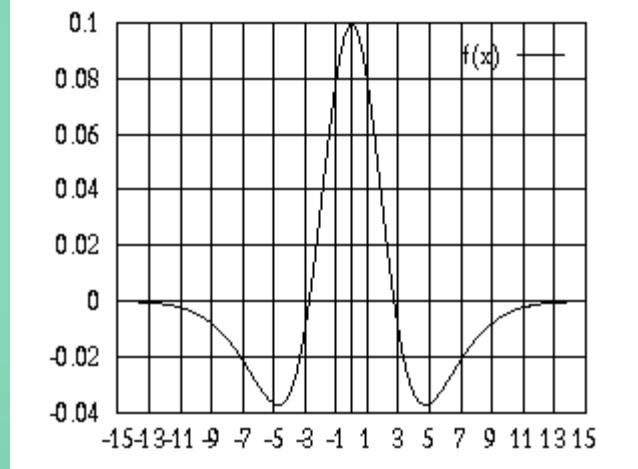
The laplacian operator

Usually, the image smoothed (e.g. with a Gaussian filter) before calculating the derivative. Using the properties of convolution, this can be done in one step:

$$\nabla^2(f \otimes h) = (\nabla^2 f) \otimes h = f \otimes (\nabla^2 h)$$

Instead of the maxima of the gradient we search the zero crossings of the second derivative.

The “mexican hat” filter:



$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

The Laplacian filter: properties

Advantages:

- Closer to mechanisms of visual perception (ON/OFF cells)
- One parameter only (size of the filter)
- No threshold
- Produces closed contours

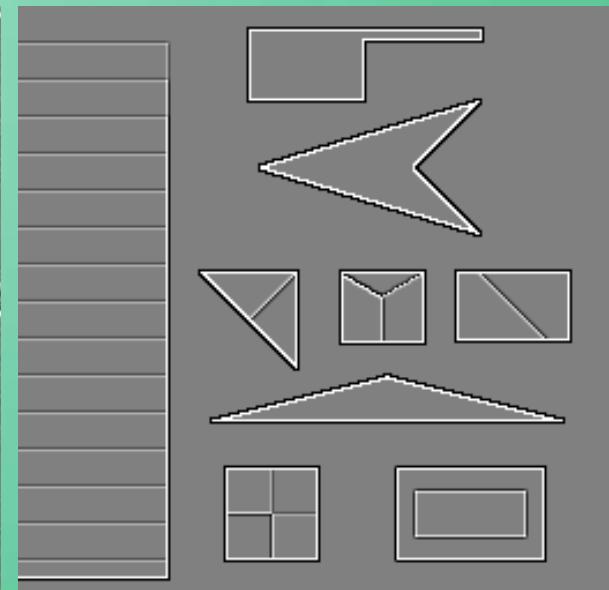
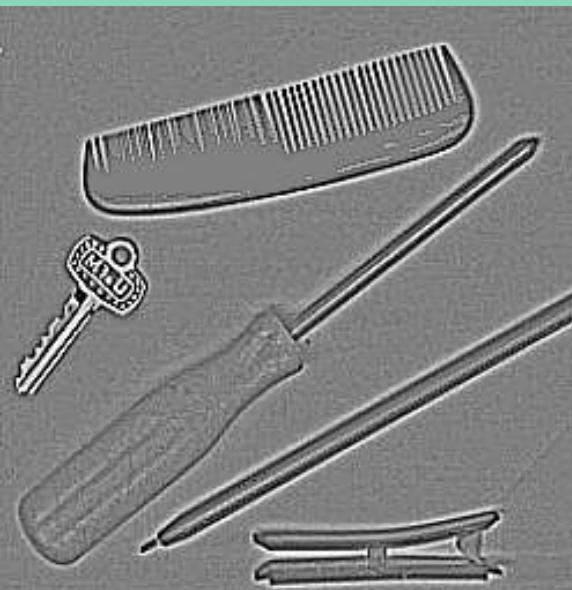
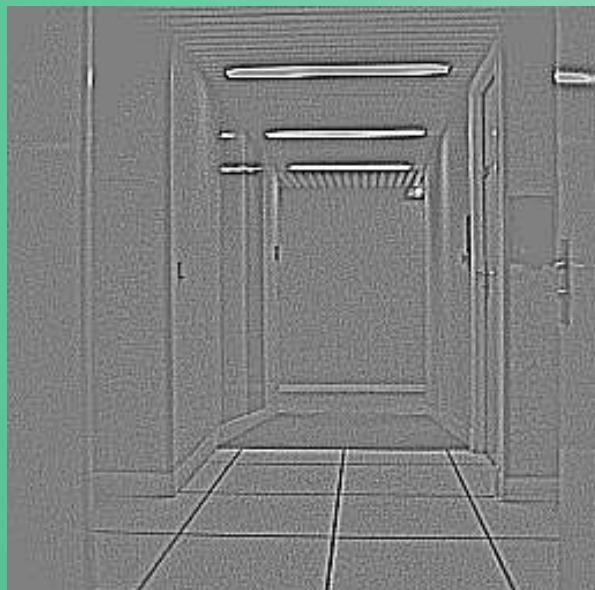
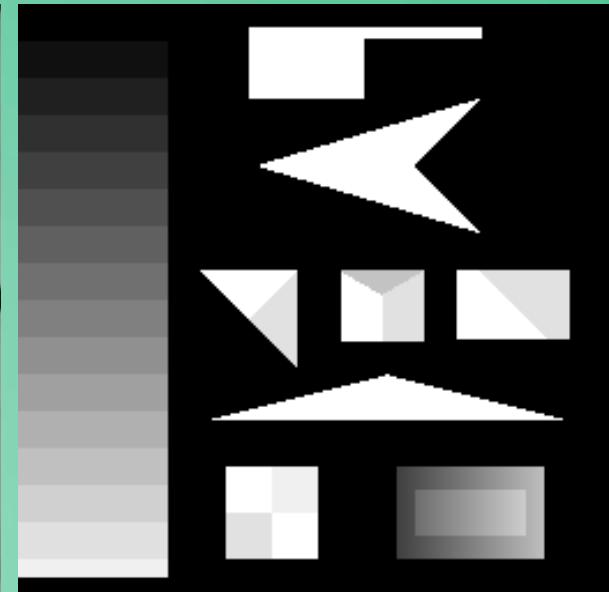
Disadvantages:

- Is more sensitive to noise (usage of second derivative)
- No information on the orientation of the contour

Combination of gradient and contour

- Search of zero-crossings of the Laplacian in the neighborhood of local maxima of the gradient

Laplacian filter: examples



Zero crossings: examples



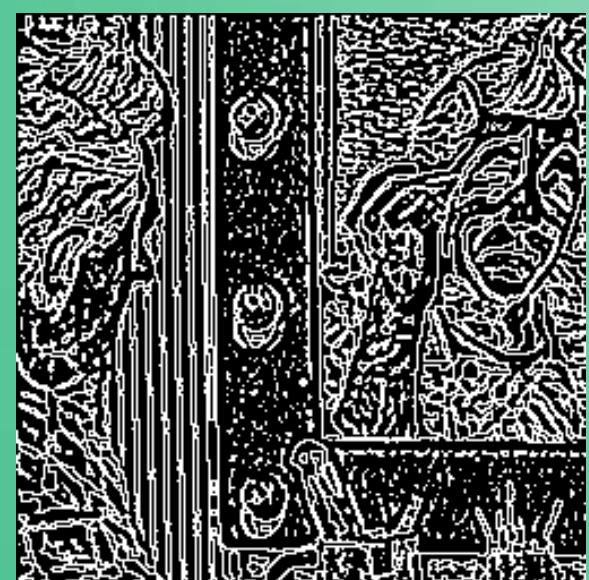
$\sigma=1$



$\sigma=2$



$\sigma=3$



Mejoramiento de la nitidez

Mejoramiento de la Nitidez – 1/2

- Corresponde a la mejora de la calidad visual de una imagen
- Se basa en los filtros “ unsharp masking ” o filtros de enmascaramiento de imagen borrosa

Principio :

Añadir detalles (frecuencias altas) a una imagen borrosa (frecuencias bajas)

Imagen mejorada = $(A-1)$ imagen original + imagen filtrada paso-altas

$$A > 1$$

Mejoramiento de la Nitidez – 2/2

imagen mejorada = $(A-1)$ imagen original + imagen filtrada paso-altas

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Filtro Laplaciano (filtro paso-altas)

$A=1$: Filtro Laplaciano estándar

$A>1$: una parte de la imagen original se añade a la paso-alta



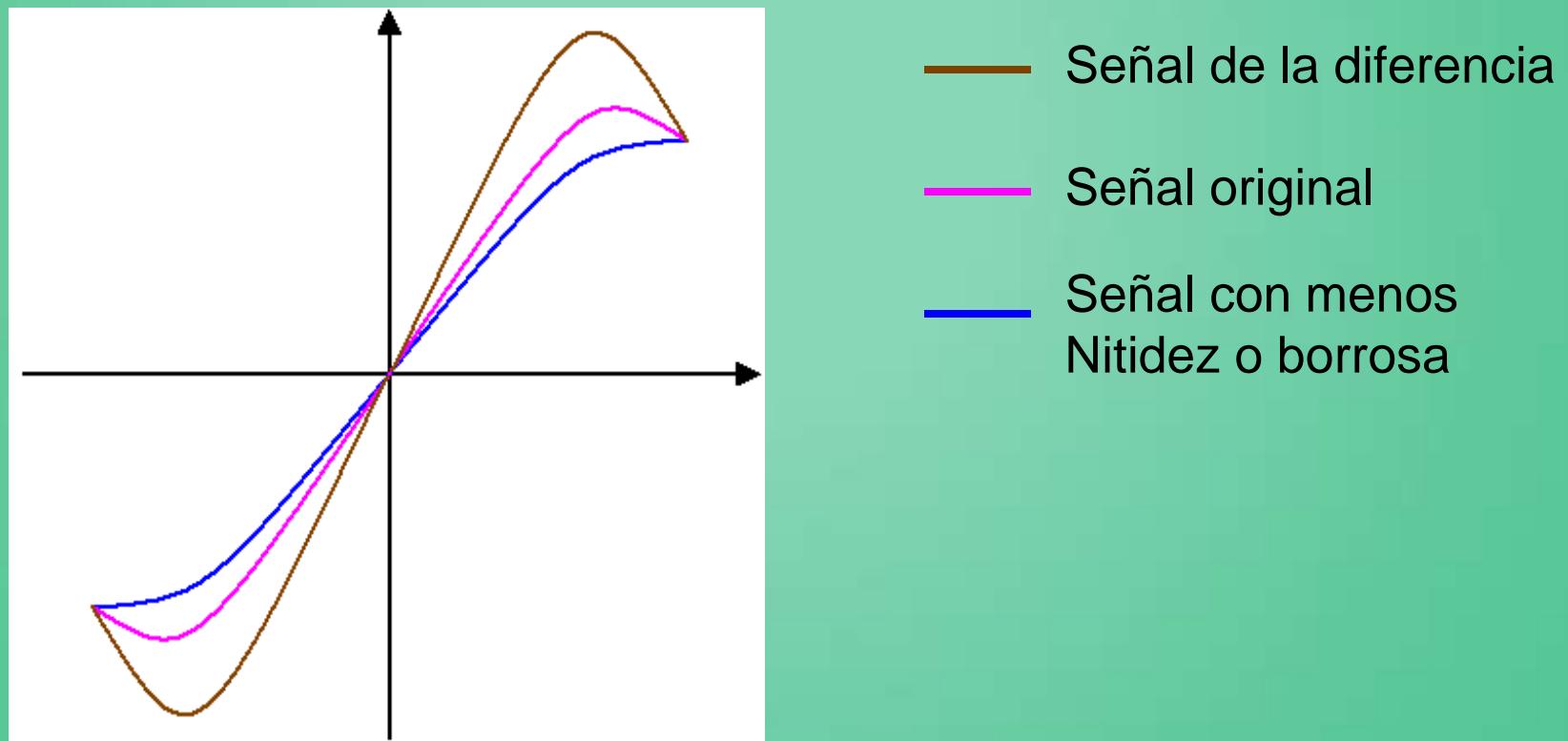
$A=2$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

¡ añade
ruido !

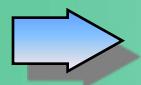
Filtro Unsharp Masking – 1/3

- Se tiene una imagen borrosa (pendiente pequeña) = f
- Se le resta con una pendiente aún más pequeña = f_{LPF}
- Lo anterior se multiplica por un factor = k (entre 1 y 3)
- La señal de la diferencia anterior se suma a la original



Filtro Unsharp Masking – 2/3

- Se tiene una imagen borrosa (pendiente pequeña) = f
- Se le resta con una pendiente aún más pequeña = f_{LPF}
- Lo anterior se multiplica por un factor = k (entre 1 y 3)
- La señal de la diferencia anterior se suma a la original

 señal con mayor resolución (pendiente grande)

$$k[f(x, y) - f_{LPF}(x, y)] + f(x, y) = f(x, y) \otimes h_{UM}(x, y)$$

Nótese que :

$$f_{HPF}(x, y) = f(x, y) - f_{LPF}(x, y)$$

con $k = 1$:

$$f(x, y) + f_{HPF}(x, y) = f(x, y) \otimes h_{UM}(x, y)$$

= imagen mejorada

Filtro Unsharp Masking – 3/3

De lo anterior se obtiene la definición del filtro Unsharp Masking $h_{UM}(x, y)$:

$$h_{UM}(x, y) = (1 + k)\delta(x, y) - kh_{LPF}(x, y)$$

- La forma del filtro Unsharp Masking $h_{UM}(x, y)$ depende de la forma del filtro paso-bajas $h_{LPF}(x, y)$

Ejemplo :

si
$$h_{LPF}(x, y) = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

entonces

$$h_{UM}(x, y) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1+k & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{k}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{k}{9} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8+9/k & -1 \\ -1 & -1 & -1 \end{bmatrix}$$