

# Visión por Computadora I

Ing. Andrés F. Brumovsky  
(abrumov@fi.uba.ar)

Laboratorio de Sistemas Embebidos -FIUBA



# FILTROS LINEALES



Los ajustes adaptativos de histograma o códigos LBP son ejemplos de operadores de vecindad (se utiliza el valor de los píxeles vecinos para determinar el valor de salida del píxel en cuestión)

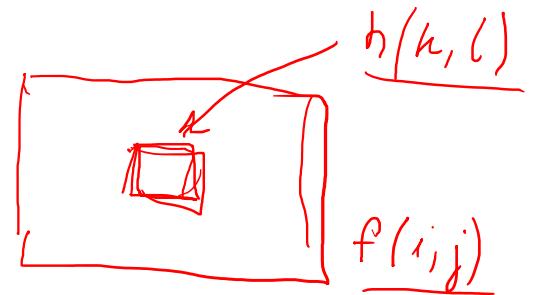
- Además de estas operaciones los operadores de vecindario se pueden utilizar también para filtrar imágenes de manera de:

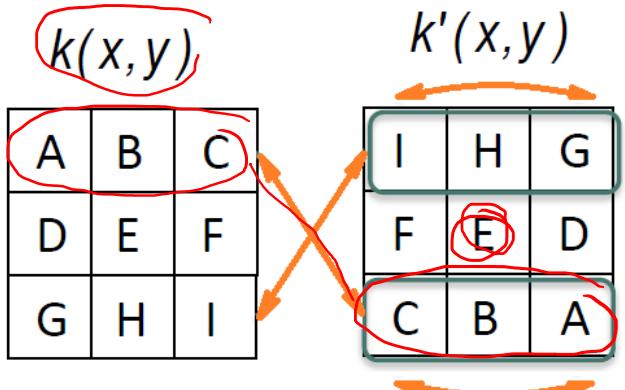
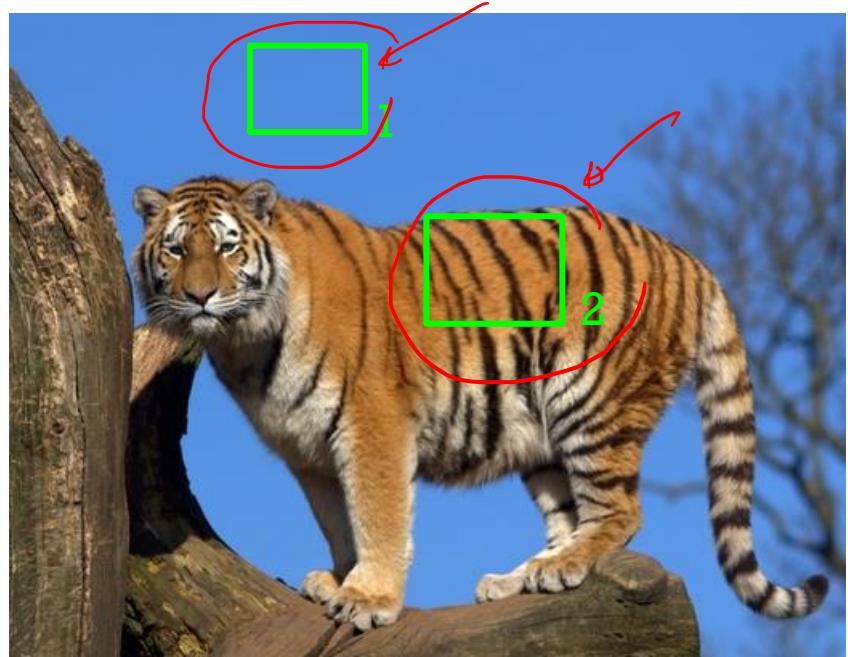
1. Agregar suavizados
2. Remarcar detalles
3. Acentuar bordes
4. Remover ruido
5. Etc.

- A continuación vamos a ver el caso de Operadores Lineales.

$$g(i,j) = \sum_{(k,l)} f(i+k, j+l) \cdot h(k, l) \rightarrow g = f \otimes h$$

$h(k, l)$ : Kernel, máscara, coef. filtro, núcleo  
 $\otimes$ : Operador correlación.





Correlación  
con  $k(x,y)$

Convolución  
con  $k'(x,y)$

# FILTROS LINEALES

- Una variante de la fórmula es invertir los offsets sobre  $f(i,j)$

$$g(i,j) = \sum_{(k,l)} f(i-k, j-l) \cdot h(k, l)$$

$$= \sum_{(k,l)} f(k, l) \cdot h(i-k, j-l)$$

Esto se escribe como

$$g = f * h$$

Donde  $*$  es el operador convolución

- ¿Por qué invertimos el núcleo?

Porque si lo convolucionamos con la señal impulso  $\delta(i,j) = 1$  resulta  $h * \delta = h$

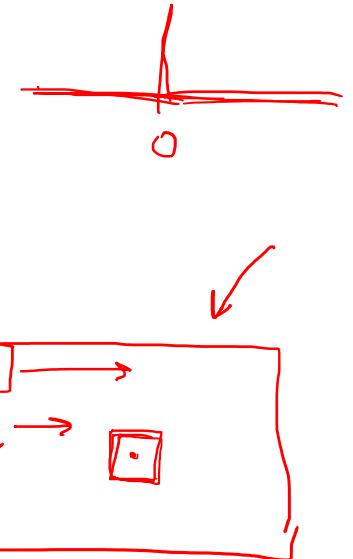
Ejemplo:

$$h(i,j) = [-1, 0, 1]; \delta(i,j) = 1$$

$$h'(i,j) = [1, 0, -1] \rightarrow g(i,j) = h' \otimes \delta = [-1, 0, 1] = h = h * \delta$$

cometación

¿Qué pasa si el núcleo es simétrico?



# FILTROS LINEALES

- Tanto convolución como correlación pueden implementarse también como la multiplicación por una matriz escasa (sparse)

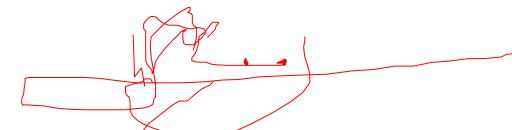
Ejemplo:

$$\begin{bmatrix} 72 & 88 & 62 & 52 & 37 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix} \leftrightarrow \begin{bmatrix} 1 \\ 4 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ . & 1 & 2 & 1 \\ . & . & 1 & 2 \\ . & . & . & 1 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix} \rightarrow g = H.f$$

- Tanto la correlación como la convolución son operadores lineales, invariantes al desplazamiento (LSI), que obedecen tanto el principio de superposición como al de invarianza al desplazamiento.

$$1. \quad h * (f_0 + f_1) = h * f_0 + h * f_1$$

$$2. \quad g(i, j) = f(i + k, j + l) \Leftrightarrow (h * g)(i, j) = (h * f)(i + k, j + l)$$

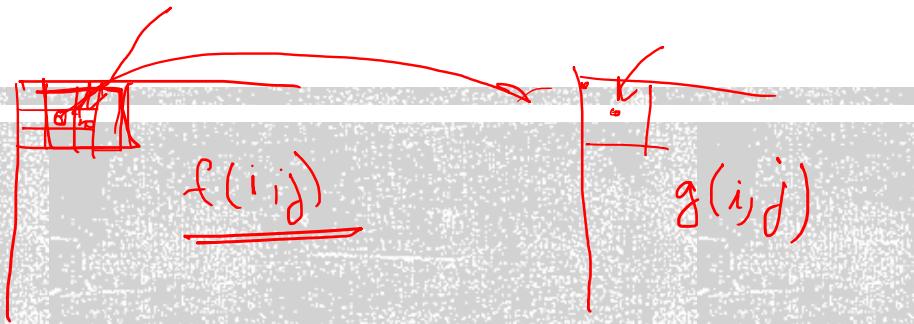


- Una diferencia importante entre convolución y correlación es que la primera es asociativa. Si F y G son núcleos e I una imagen:

$$F * (G * I) = (F * G) * I$$



# EJEMPLOS DE NÚCLEOS TÍPICOS



- Algunos ejemplos de núcleos (Kernels) típicos:

Enfoque:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Log:

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

Roberts:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}; \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Sobel:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Gauss:

$$\frac{1}{121} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

Prewitt:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}; \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\frac{\sum_{i=1}^3 \sum_{j=1}^3 f(i,j)g(i,j)}{\sum_{i=1}^3 \sum_{j=1}^3 g(i,j)}$$

$$\frac{\sum_{i=1}^3 \sum_{j=1}^3 f(i,j)g(i,j)}{\sum_{i=1}^3 \sum_{j=1}^3 g(i,j)} = 50$$

Desenfoque:

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

Sharpen:

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ -1 & -2 & 1 \end{bmatrix}$$

Gauss:

$$\frac{1}{121} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

Sobel:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}; \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel y Prewitt son a su vez ejemplos de filtros pasa alto, direccionalables.

# LOG (LAPLACIANO DEL GAUSSIANO)

- Podemos definir matemáticamente el Laplaciano como

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

$$\frac{\partial^2}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \rightarrow \text{kernel } x: \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

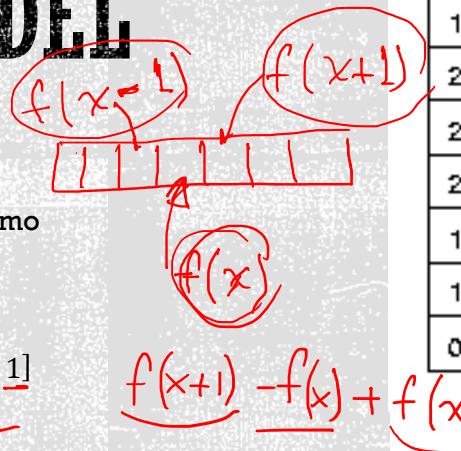
$$\frac{\partial^2}{\partial y^2} = f(y+1) + f(y-1) - 2f(y) \rightarrow \text{kernel } y: \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Detecta los cruces por cero (cambio de signo)
- No usa dos núcleos con orientaciones distintas (como Prewitt, Sobel) y pierde información de orientación
- Es isotrópico → magnitud de borde uniforme en toda dirección
- Es muy sensible al ruido

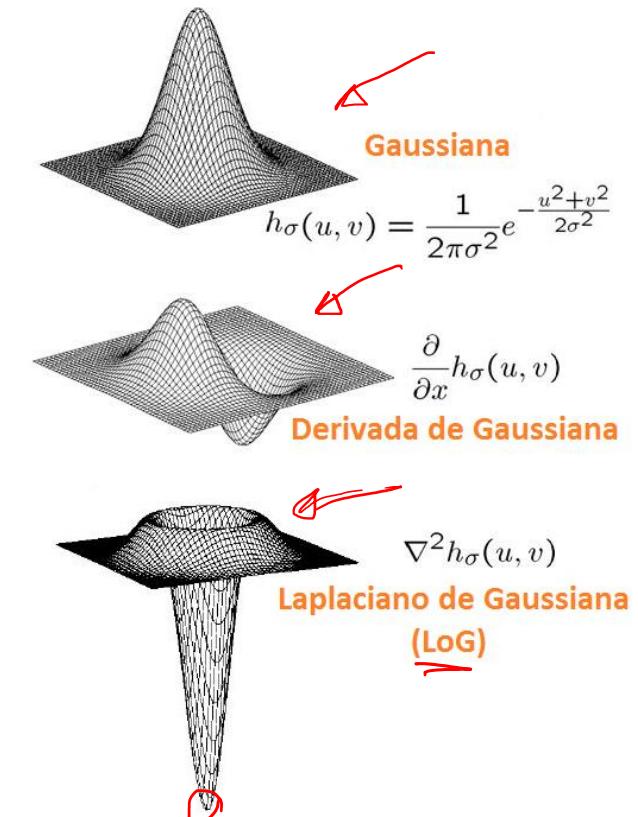
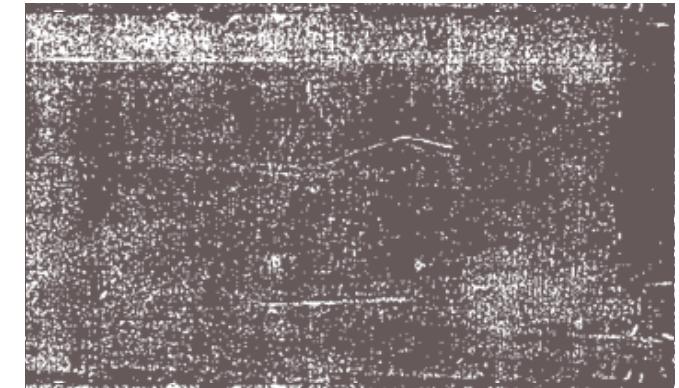
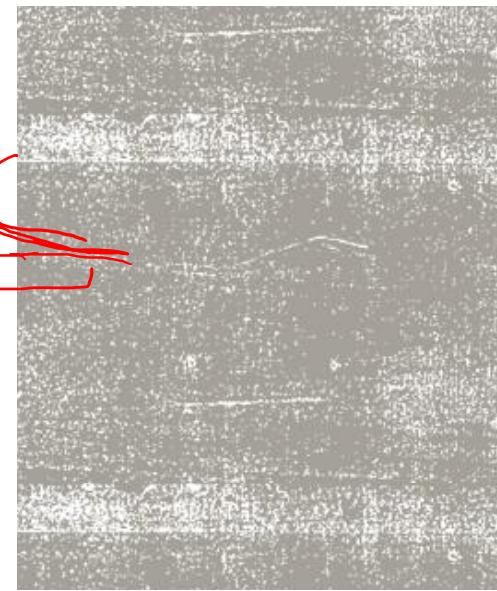
Para reducir el ruido se le puede aplicar previamente un filtro gaussiano a la imagen, o...hacerlo en un solo paso!

$$\nabla^2(f * g) = f * \nabla^2(g) \rightarrow \text{Convolución con el Laplaciano de la Gaussiana LoG}$$

Matemáticamente,  $\text{LoG}(x, y) = \frac{1}{\pi\sigma^4} \left[ 1 - \frac{x^2+y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} \dots$  y podemos muestrear...



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0



# FILTROS SEPARABLES

- La convolución con un núcleo implica procesar por cada píxel  $K^2$  operaciones de multiplicación+suma donde  $K$  es el tamaño del núcleo ( $N \times M$ )
- Esto puede acelerarse realizando primero una convolución unidimensional horizontal seguida de una convolución unidimensional vertical. Requeriendo así  $2K$  operaciones.

- Un núcleo que permite hacer esto se dice ser **separable**

- Un núcleo  $K$  "separable", es decir corresponda a una convolución sucesiva de un núcleo horizontal  $h$  y un núcleo vertical  $v$  debe cumplir:

$$K = vh^t$$

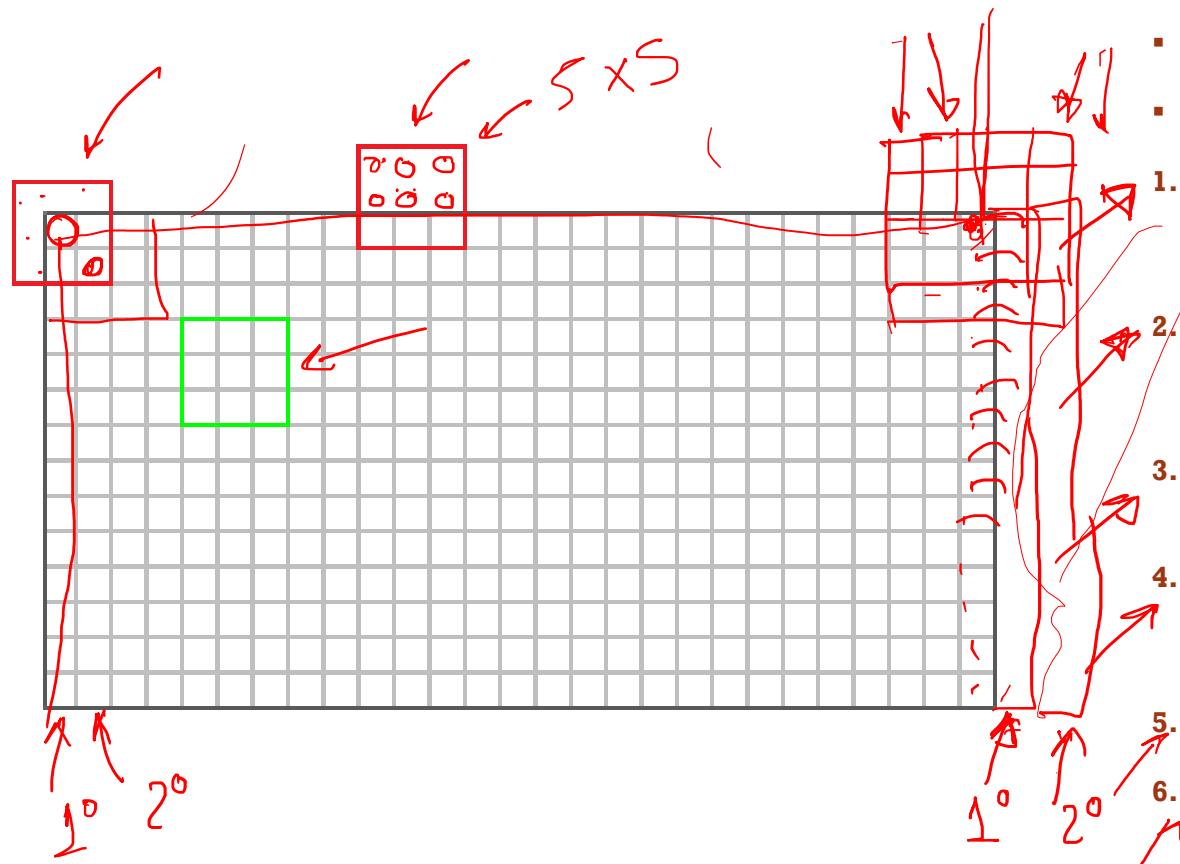
Ejemplos:

- Filtros de suavizado:  $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \frac{1}{3} [1 \ 1 \ 1] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- Filtro Gaussiano:  $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \frac{1}{4} [1 \ 2 \ 1] = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
- Sobel (vale para Prewitt):  $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [1 \ 0 \ -1] = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
- Los filtros Laplaciaños también pueden ser implementados como filtros separables.

¿Cómo sabemos si un filtro es separable?

1. Por inspección
2. Tratar al núcleo  $K$  como matriz y tomar su descomposición en valores singulares (SVD):  $K = \sum_i \sigma_i \mu_i v_i^T$ 
  - Si solo el primer valor singular  $\sigma_0$  es distinto de cero, entonces el kernel es separable y  $\sqrt{\sigma_0} \mu_0$  y  $\sqrt{\sigma_0} v_0^T$  son los kernels verticales y horizontales respectivamente.

# EFFECTO DE BORDES (PADDING)



▪ ¿Qué pasa en los bordes de la imagen?

▪ Opciones:

**Cero:** establecer todos los píxeles fuera de la imagen de origen en 0 (una buena opción para imágenes de recorte del canal alfa)

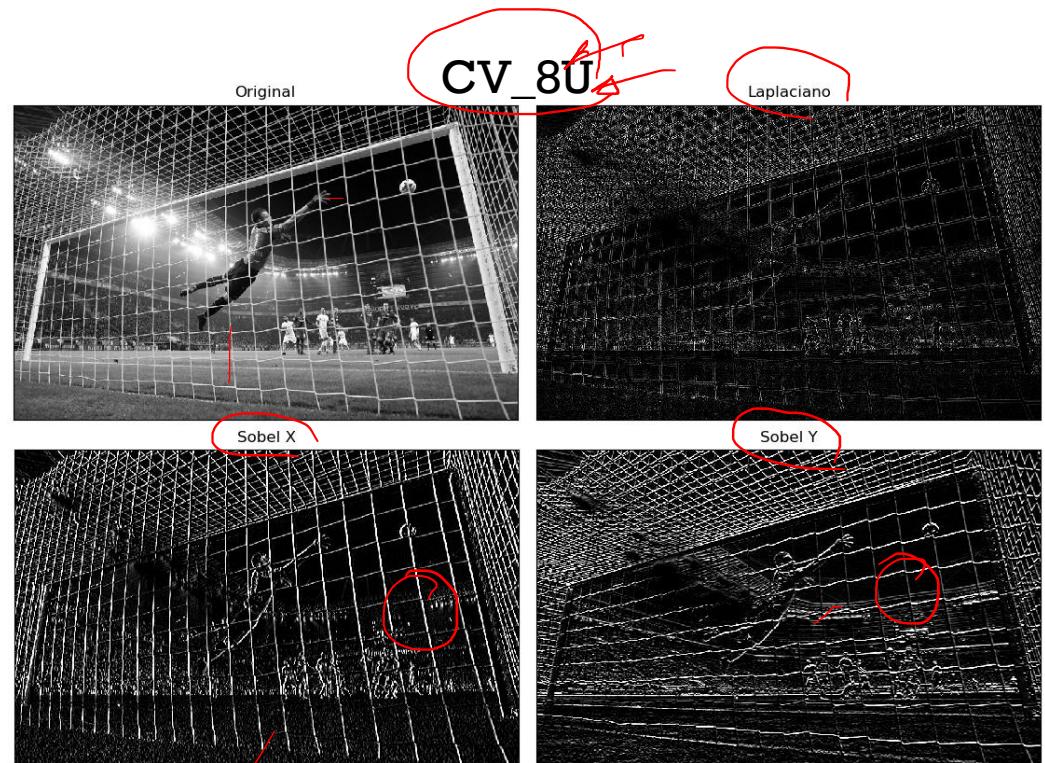
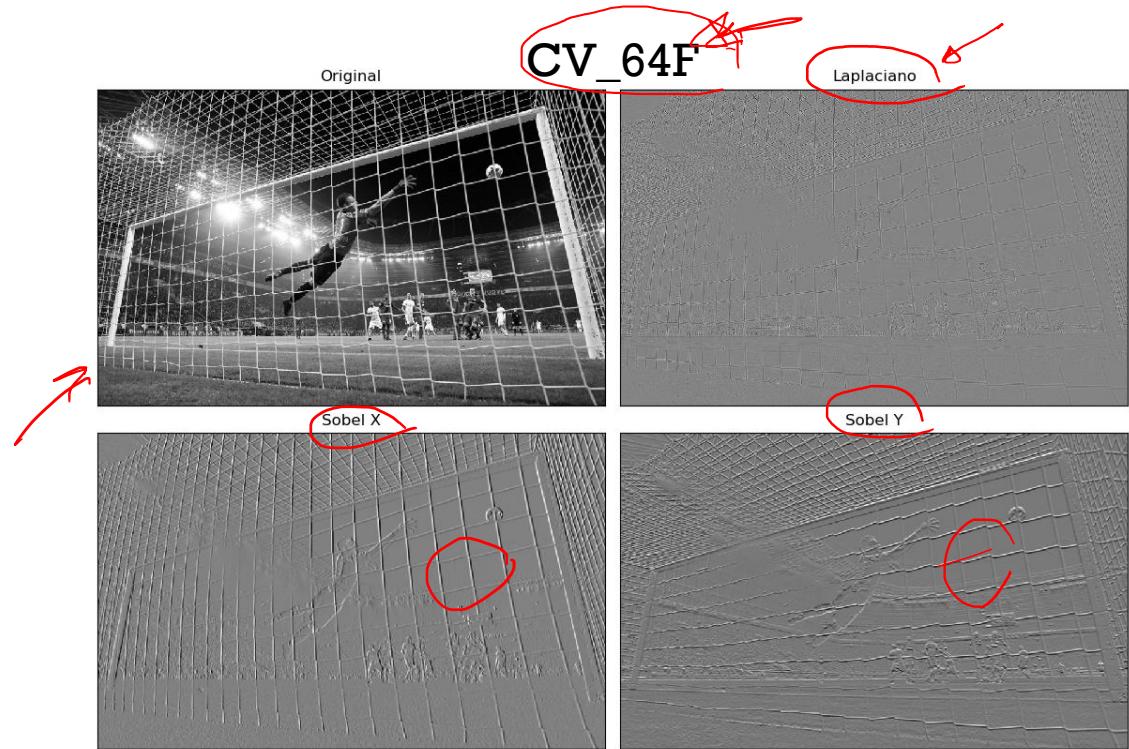
**Constante (color del borde):** establece todos los píxeles fuera de la imagen de origen en un valor de borde especificado

**Replicado (clamp):** repetir los píxeles del borde indefinidamente;

**Envolver (cíclica) (repetición o mosaico):** bucle "alrededor" de la imagen en una configuración "toroidal";

**Espejo:** reflejar píxeles en el borde de la imagen;

**Extender:** extiende la señal restando la versión reflejada de la señal del valor de píxel de borde.



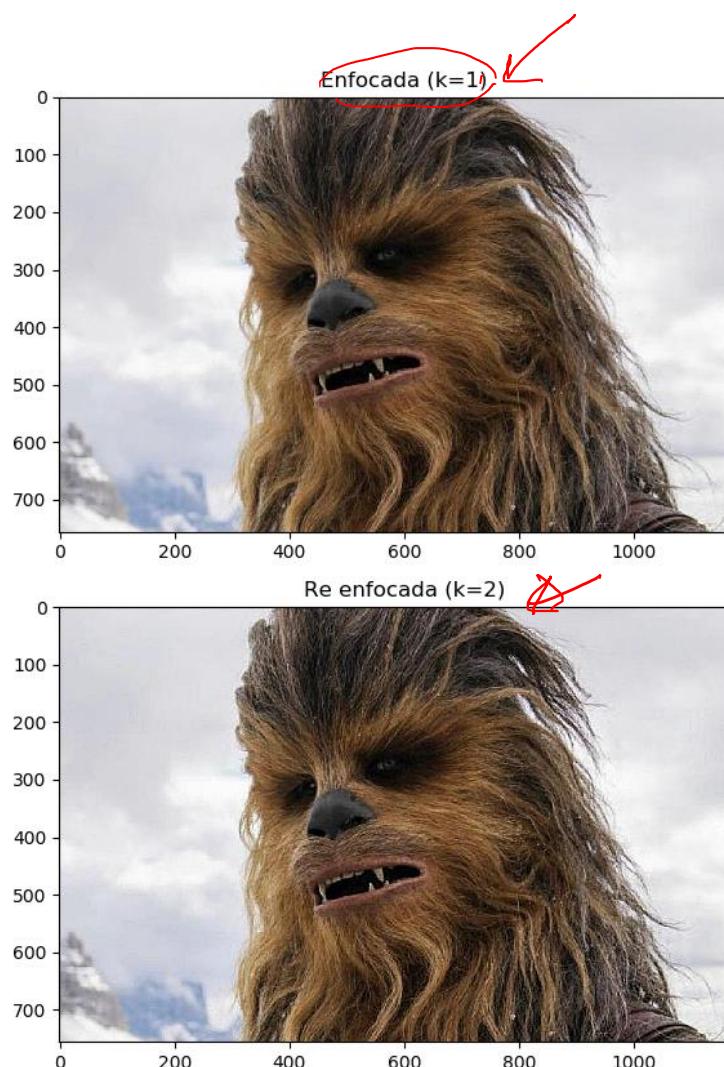
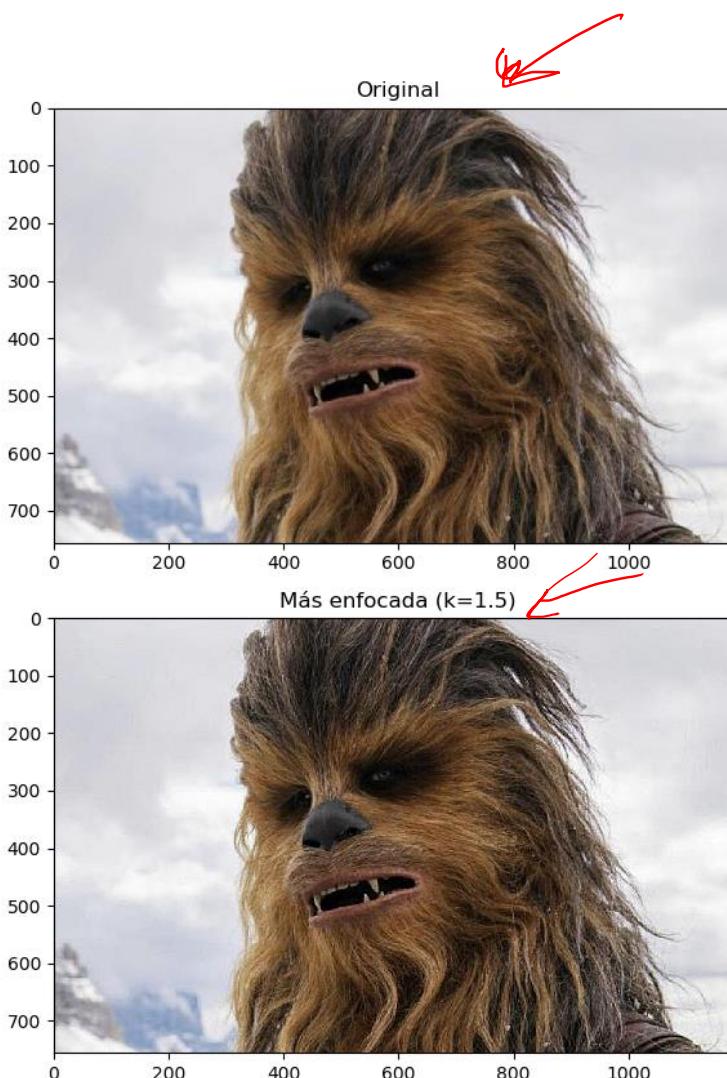
# EJEMPLO

- En los ejemplos de la derecha el tipo de datos de salida es cv.CV\_8U. Pero hay un problema:
  - La transición de negro a blanco (pendiente positiva) tiene un valor positivo.
  - Transición de blanco a negro (pendiente negativa) tiene valor negativo.

▪ Cuando se convierte datos a `UINT8`, todas las pendientes negativas se hacen cero. En palabras simples, echas de menos esa ventaja.

Si se desea detectar ambos bordes, la mejor opción es mantener el tipo de datos de salida en algunas formas superiores, como `cv.CV_16S`, `cv.CV_64F`, etc., tomar su valor absoluto y luego volver a convertirlo en `cv.CV_8U`.





## UNSHARP MASKING

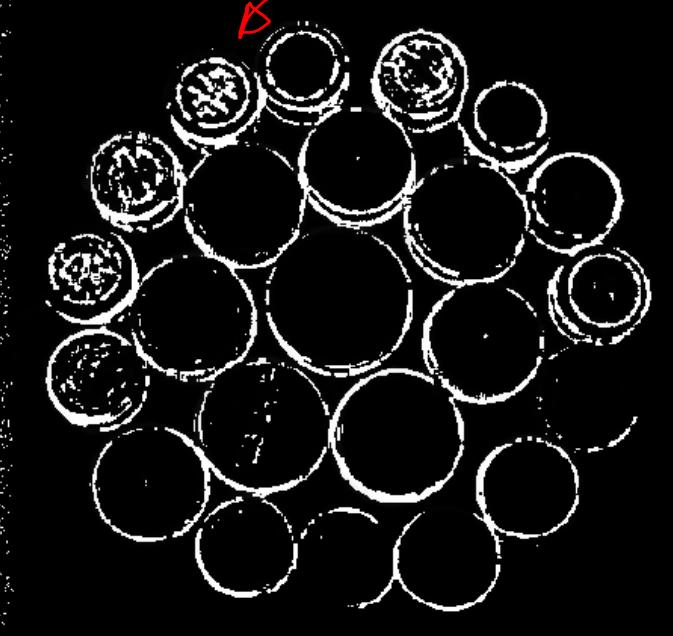
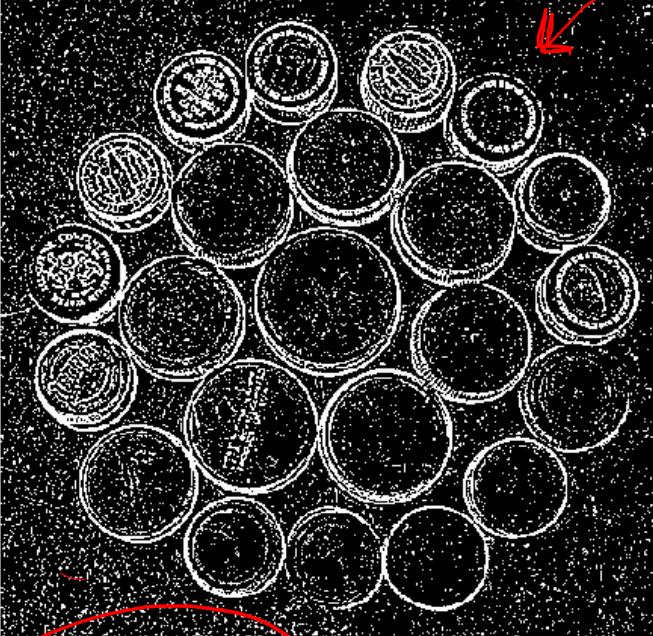
- Es un algoritmo que tiene enfocar o mejorar los bordes con un filtro de suavizado
  1. Desenfoque de la imagen con un filtro suavizador (ej. Gaussiano)  
 $f(x, y) \rightarrow f_B(x, y)$  Blur
  2. Resta de la imagen original con la suavizada (sobreviven componentes de alta frecuencia)  
 $m(x, y) = f(x, y) - f_B(x, y)$
  3. Agregar esa máscara a la imagen original (realce de las componentes de alta frecuencia)  
 $g(x, y) = f(x, y) + k \cdot m(x, y)$

De manera genérica:

$$g(x, y) = (k + 1) \cdot f(x, y) - k \cdot f_B(x, y)$$

$k = 1$ : Máscara de desenfoque (unsharp masking)  
 $k > 1$ : Filtrado de alto impulso (highboost filtering)



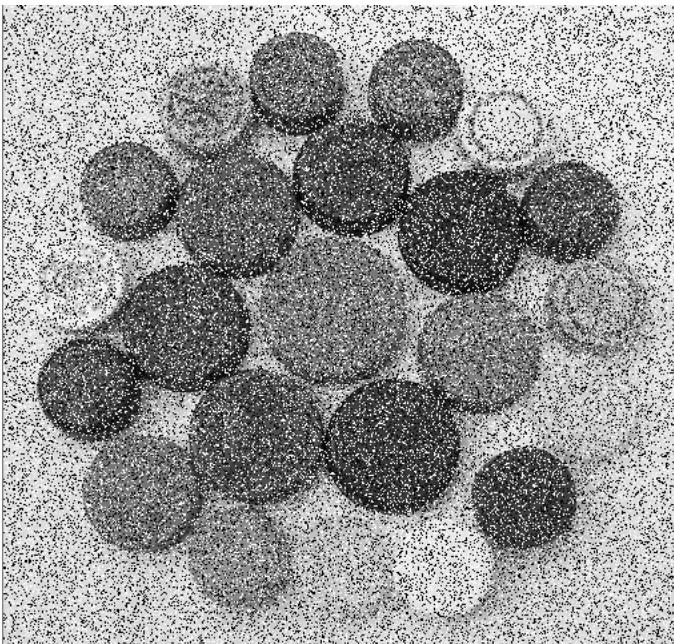


$f(x,y)$

$g_1(x,y) - g_2(x,y)$

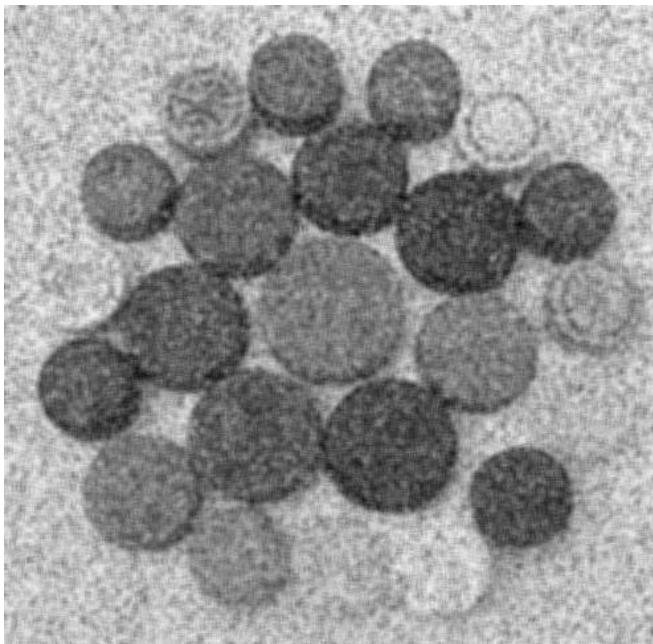
- De los filtros de suavizado Gaussianos sabemos que el desvío estándar incide directamente en el grado de suavizado.
- A mayor sigma mayor nivel de desenfoque
- Por lo tanto si utilizamos dos sigma distintos y aplicamos el suavizado a una misma imagen para después restar las salidas esto actuará como un filtro pasabanda

$$DoG = Gauss_{low\sigma} - Gauss_{high\sigma}$$

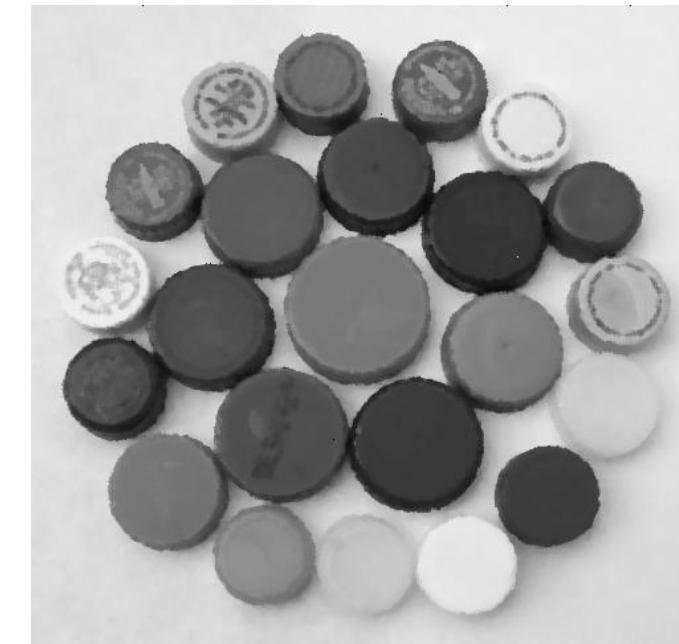



A

Original con ruido



Filtro media 5x5



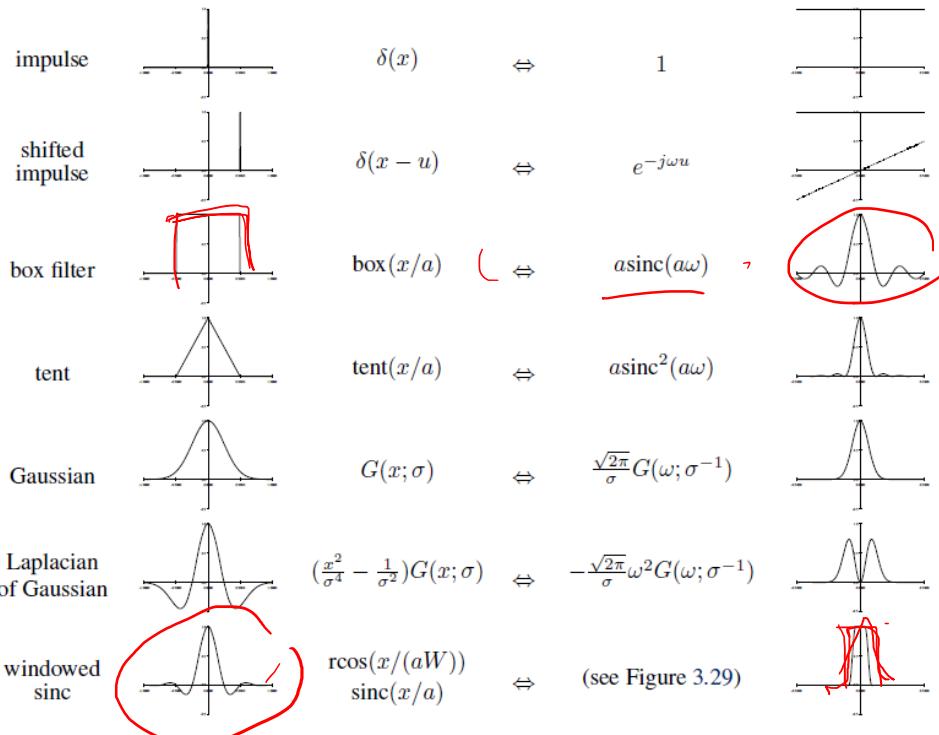
Filtro mediana 5x5

# FILTRO DE RUIDO IMPULSIVO

- Ruido impulsivo: Salt & Pepper
- Media: Valor Promedio
- Mediana: Valor medio del conjunto ordenado
- Moda: Valor más repetido
- Ej:  $\begin{bmatrix} 1 & 9 & 5 \\ 6 & 9 & 8 \\ 4 & 10 & 2 \end{bmatrix} \rightarrow \text{Media: } 6, 1 / \text{Mediana: } 6 / \text{Moda: } 9$
- Media:  $(1+9+5+6+9+8+4+10+3)/9$
- Mediana:  $[1; 3; 4; 5; 6; 8; 9; 9; 10] / 255 \text{ Moda: } 9$



## Señal



## Transformada

# FOURIER

- La transformada de Fourier se utiliza
  - Para analizar el comportamiento en frecuencia de varios filtros.
  - En imágenes (2D DFT), para encontrar las componentes de frecuencia. En general se utiliza la FFT.

$$H(k) = \frac{1}{N} \sum_{x=0}^{N-1} h(x) e^{-j\frac{2\pi kx}{N}}$$

vale  $\forall k$ , pero solo tiene sentido en  $k \in [-\frac{N}{2}, \frac{N}{2}]$

DFT →  $N^2$  operaciones; FFT →  $N \log_2 N$

- ¿Dónde es que varía fuertemente la amplitud en imágenes?

- En bordes
- En donde hay ruido

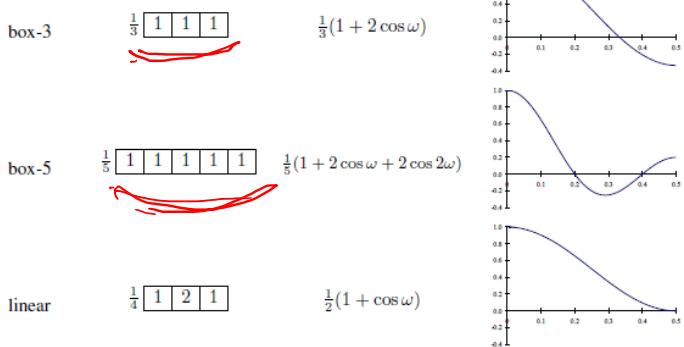
- Por lo tanto, bordes y ruido son componentes de alta frecuencia.

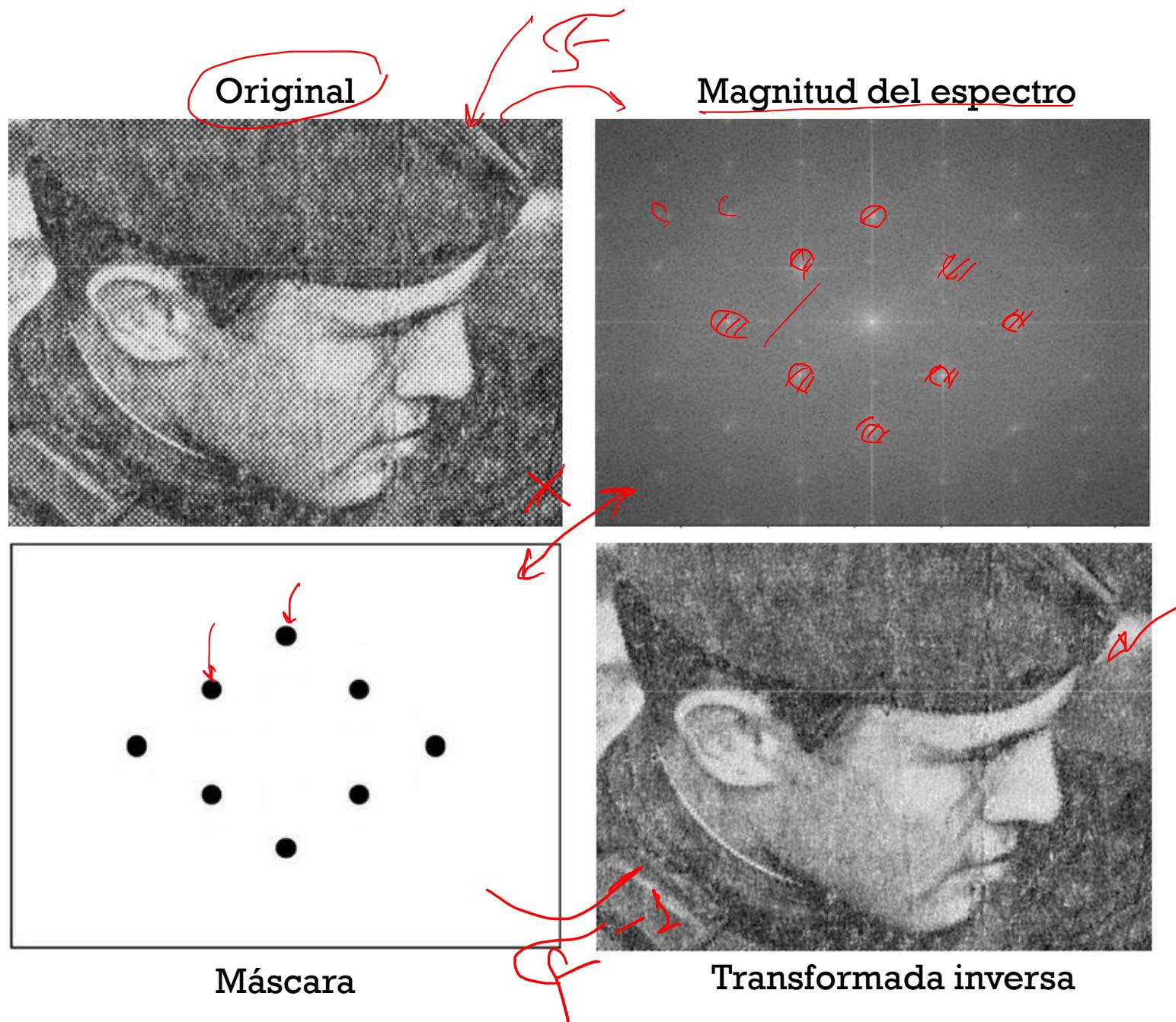
- Algunas propiedades:

- Superposición:  $f_1(x) + f_2(x) \rightarrow F_1(w) + F_2(w)$
- Inversa:  $f(-x) \rightarrow F^*(w)$
- Convolución:**  $f(x) * h(x) \rightarrow F(w)H(w)$
- Correlación:  $f(x) \otimes h(x) \rightarrow F(w)H^*(w)$
- Multiplicación:  $f(x)h(x) \rightarrow F(w) * H(w)$
- Escalamiento:  $f(ax) \rightarrow \frac{1}{a} F\left(\frac{w}{a}\right)$
- Diferenciación:  $f'(x) \rightarrow jwF(w)$

## Núcleo

## Transformada





## FOURIER

- Estas fórmulas vistas para el caso unidimensional pueden trasladarse al caso bidimensional (imágenes), donde las sinusoidales son en dos dimensiones

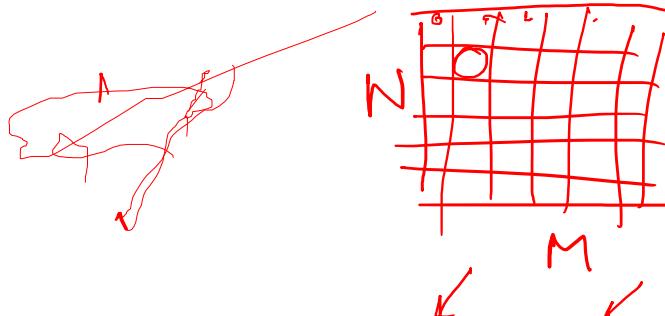
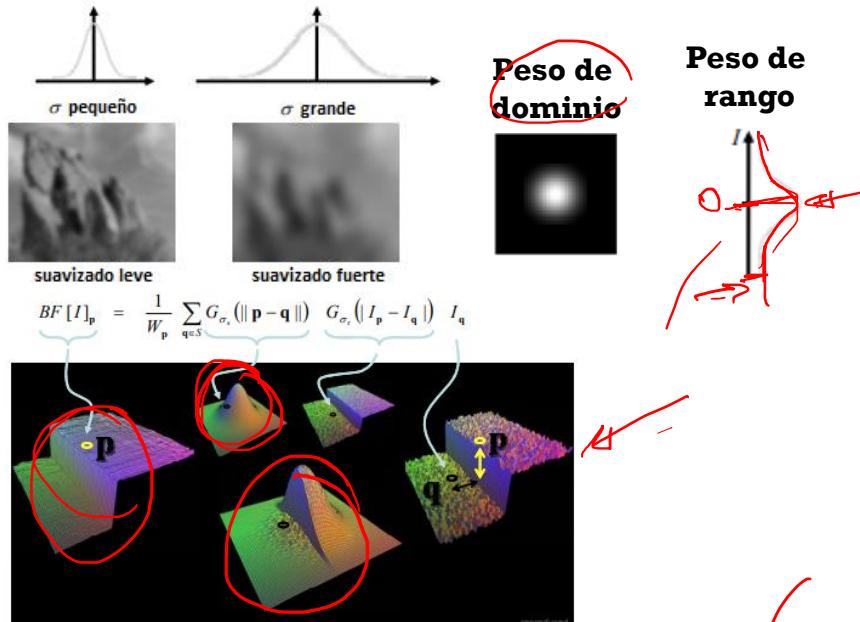
$$s(x, y) = \sin(w_x x + w_y y)$$

- Luego, la transformada de Fourier será

$$H(w_x, w_y) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-j2\pi \frac{k_x x + k_y y}{MN}}$$

- Donde M,N son ancho y alto de la imagen
- Todas las propiedades para 1-D valen de igual manera para 2-D cambiando  $x \rightarrow (x, y)$ ,  $w \rightarrow (w_x, w_y)$ ,  $a \rightarrow (a_x, a_y)$  y usando producto interno en lugar de multiplicación.

# FILTROS BILATERALES



✓ 255

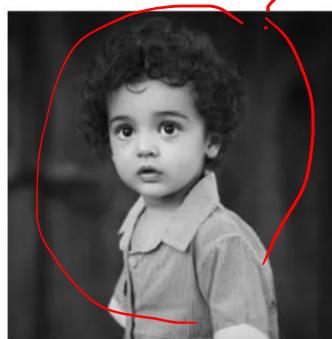
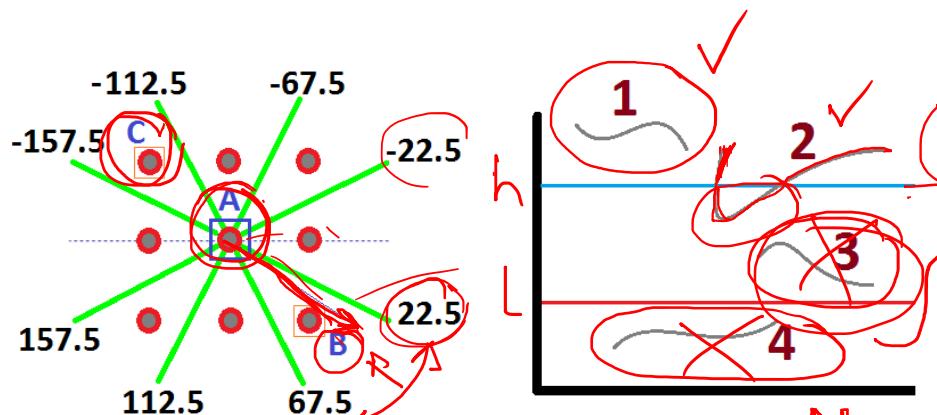
- Filtros de suavizado como Promedio, Mediana, Gaussiana.
  1. Útiles para eliminar distintos tipos de ruido
  2. También desenfocan los bordes!
- Las imágeles (como cualquier función) tienen:
  - Dominio: Todas las posiciones posibles
  - Rango: Todos los valores posibles
- Los filtros de Mediana, Gaussiana, etc son filtros de dominio.
 
$$\text{Ej: } \begin{bmatrix} 0 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 0 \end{bmatrix} * \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 10 \\ 0 & 4 & 0 \\ 10 & 0 & 0 \end{bmatrix} \rightarrow$$

*Cambio del pixel central*

- Veamos cómo se ve un suavizado Gaussiano
 
$$G_B[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q; \quad \sigma: \text{Parámetro espacial}$$
- Con una modificación lo transformamos en un filtro bilateral
 
$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) \cdot G_{\sigma_r}(|I_p - I_q|) I_q$$

$$W_p = \sum_{q \in S} G_{\sigma_s}(\|p - q\|) \cdot G_{\sigma_r}(|I_p - I_q|)$$

# DETECTOR DE BORDES DE CANNY



Algoritmo desarrollado por John F. Canny en 1986

## 1. Reducción de ruido

- Canny usa un filtro Gaussiano, en general de  $5 \times 5$

$$H_{i,j} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}} \quad \forall 1 \leq i, j \leq (2k+1)$$

## 2. Encontrar el gradiente de intensidad de la imagen

- Se puede utilizar un operador de Sobel (filtro direccional) y obtener  $G_x$ ,  $G_y$ . Luego,

$$G = \sqrt{G_x^2 + G_y^2} : \text{Magnitud}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) : \text{Dirección}$$

## 3. Supresión de no-máximos

- Técnica de adelgazamiento de bordes. Se verifica para cada píxel si hay un máximo local vecino en la dirección del gradiente o no. Si es un máximo local se retiene, sino se elimina.

- Los vecinos estarán a  $0^\circ, 45^\circ, 90^\circ, 135^\circ \rightarrow$  los valores del gradiente tienen que redondearse a esos rangos.

- Una vez definida la dirección:

*Si  $A > C \wedge A > B \Rightarrow$  Se retiene A, caso contrario se suprime*

## 4. Umbral con histéresis

- Del paso anterior se obtendrán distintos bordes, algunos más brillantes que otros. Para saber cuáles son bordes reales y cuales no, Canny usa histéresis. Se definen dos umbrales: Alto y Bajo

- Cualquier borde con intensidad mayor que 'Alto' es borde seguro, se retiene

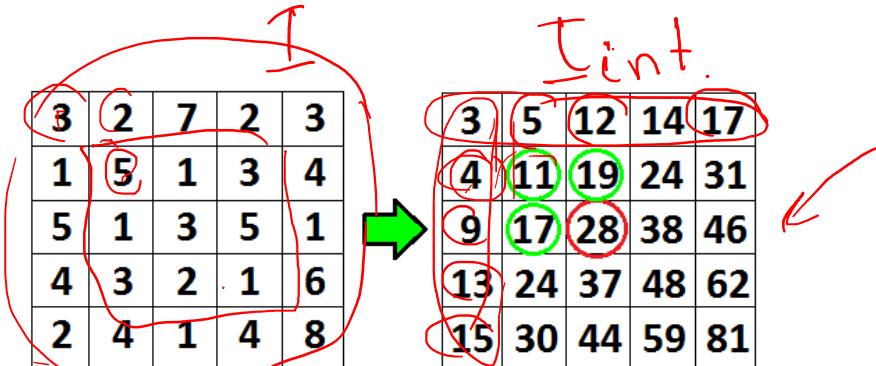
- Cualquier borde con intensidad menor que 'Bajo' no es borde, se descarta

- Los bordes entre los umbrales 'Alto' y 'Bajo' se clasifican como bordes solo si están conectados a un borde seguro, de otro modo se descarta.

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \\ -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

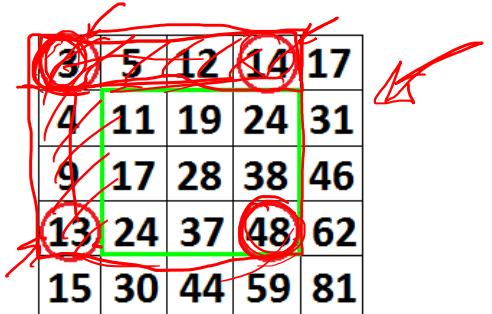
# IMAGEN INTEGRAL

Ej:



$$s(3,3) = s(2,3) + s(3,2) - s(2,2) + f(3,3)$$

$$s(3,3) = \underline{17} + \underline{19} - \underline{11} + \underline{3} = \underline{\underline{28}}$$



$$\text{Suma} = \underline{48} + \underline{3} - \underline{14} - \underline{13} = \underline{\underline{24}}$$

$$\sigma = \sqrt{\frac{1}{n} \left( S_2 - \frac{S_1^2}{n} \right)}$$

S1: Suma región rectangular

S2: Suma del cuadrado de esa región  
n: N° de píxeles en la región

- Las imágenes integrales nos permiten calcular de manera eficiente magnitudes como la medias, desviación estándar, etc.
- Fue introducido en 1984 por Frank Crow
- Se utilizan en varios ámbitos:
  1. Filtros
  2. Correspondencia de patrones (pattern matching)
  3. Detección de objetos y rostros. Boxlets
  4. Suma de suma de diferencias cuadradas (SSD). Algoritmo estéreo y Motion tracking
- Se obtiene sumando todos los píxeles anteriores, desde la esquina superior izquierda

$$s(i,j) = \sum_{k=0}^i \sum_{l=0}^j f(k,l)$$

Se puede computar fácilmente utilizando un algoritmo recursivo:

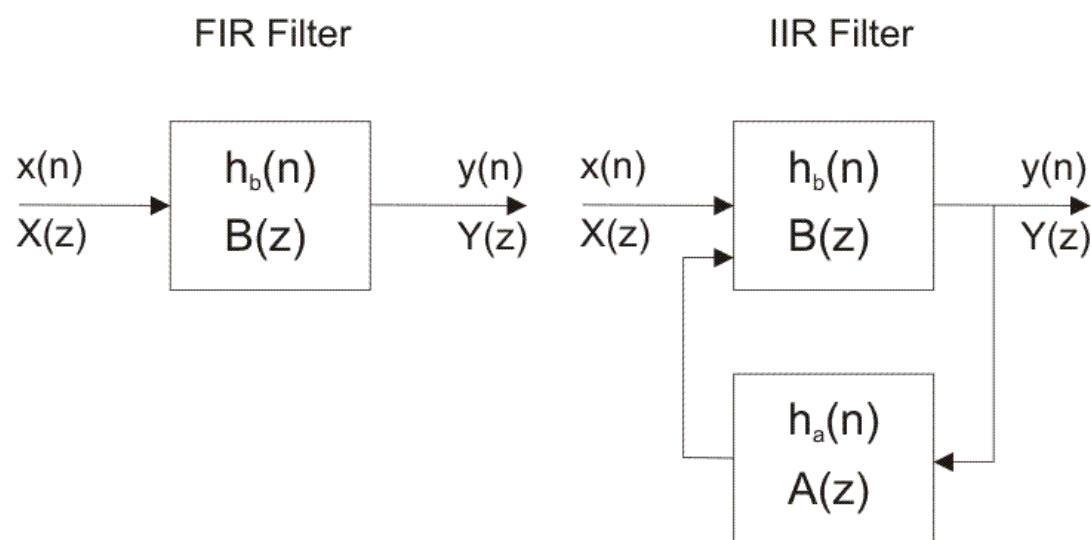
$$s(i,j) = \underline{s(i-1,j)} + \underline{s(i,j-1)} - \underline{s(i-1,j-1)} + \underline{\underline{f(i,j)}}$$

- Luego, la suma de píxeles en cualquier región rectangular se puede obtener con una simple operación y de **T constante**.

$$\text{Suma} = \underline{\text{Inf derecha}} + \underline{\text{Sup izquierda}} - \underline{\text{Sup derecha}} - \underline{\text{Inf izquierda}}$$

- Esto se puede pensar como suma y resta de áreas
- `cv2.integral(src[, sdepth]) / cv2.integral2(src[, sdepth[, sqdepth]])`

# FILTROS RECURSIVOS



- La fórmula incremental de suma para la matriz integral, donde los valores dependen de las salidas previas del filtro es un ejemplo de filtro recursivo.
- Estos filtros tienen una respuesta infinita al impulso (IIR)
- Se diferencian de los filtros que veníamos viendo (no recursivos) que dependían únicamente del valor actual de la imagen, y su respuesta al impulso era finita (FIR)
- Los filtros IIR suelen utilizarse para calcular cantidades que involucran grandes áreas, como funciones de distancia y componentes conexos. También en etapas de suavizado con grandes kernels.



# TP2 - CLASE 3

- Para la imagen suministrada “metal grid”. Implementar un algoritmo que:
  1. Calcule los gradientes en la imagen (dirección y módulo)
  2. Muestre imágenes de ángulo y módulo
  3. Marque con color las direcciones de gradientes más altos

