

Visión por Computadora I

Ing. Maxim Dorogov
(mdorogov@fi.uba.ar)

Esp. Ing. Leandro Torrent
(ingtorrentleandro@gmail.com)

Laboratorio de Sistemas Embebidos -FIUBA



MOTIVACIÓN

Hasta ahora usamos operadores de pixel que permiten procesar y modificar la imagen de entrada:



$$g(x,y) = f(x,y) + 20$$



$$g(x,y) = f(-x,y)$$

- Restauración / Corrección de defectos
- Extracción de características
- Enfoque/Desenfoque digital
- Entre otros...

- Requieren operaciones mas complejas



FILTROS LINEALES



(a)



(b)



(c)



(d)

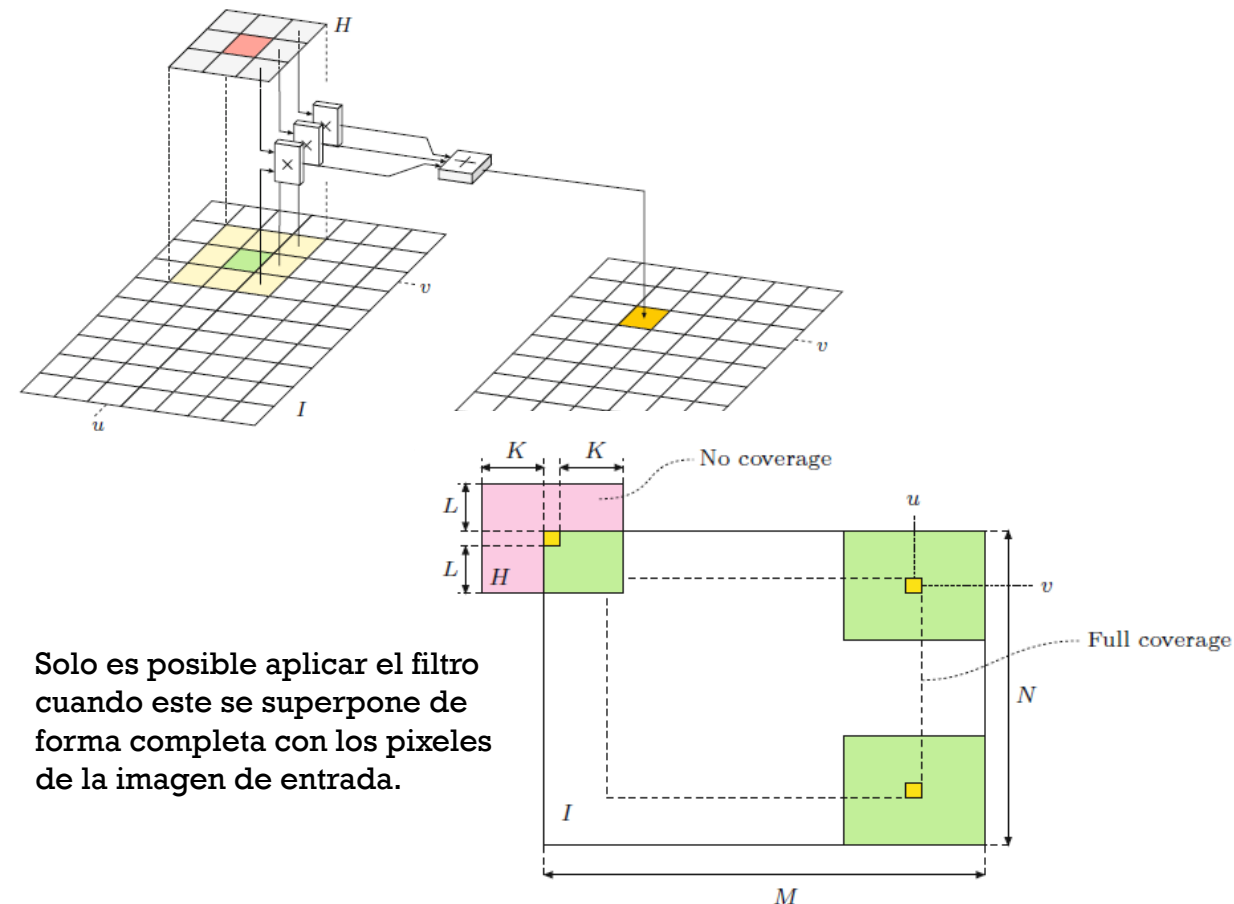
- A diferencia de los operadores de píxeles los filtros utilizan varios píxeles de la imagen original para computar 1 píxel en la imagen de salida (operadores de vecindad)
- Los ajustes adaptativos de histograma o códigos LBP son ejemplos de operadores de vecindad (se utiliza el valor de los píxeles vecinos para determinar el valor de salida del píxel en cuestión)
- Se pueden aplicar filtros a imágenes de manera de:
 1. Agregar suavizados
 2. Remarcar detalles
 3. Acentuar bordes
 4. Remover ruido
 5. Extraer características para su posterior procesamiento.
- A continuación vamos a ver el caso de operadores lineales.

$$I'(u, v) \leftarrow \sum_{(i, j) \in R_H} I(u + i, v + j) \cdot H(i, j), \quad \rightarrow I' = f \otimes h$$

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



FILTROS LINEALES



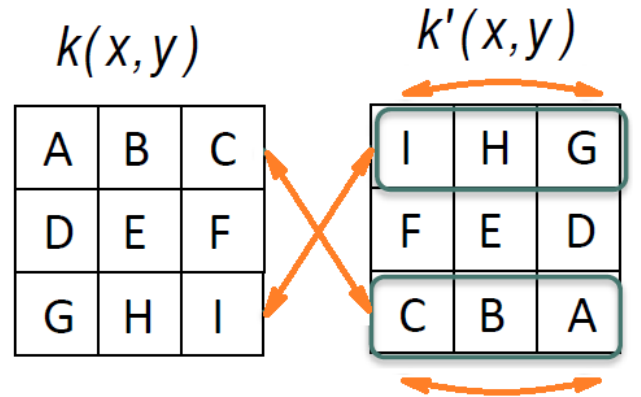
Conceptos generales:

- Consideramos al punto central del filtro como origen (0, 0). (Los filtros generalmente son impares)
- Desplazamos el filtro a lo largo y ancho de la imagen y multiplicamos los coeficientes punto a punto, sumamos y obtenemos el valor del pixel en la posición (u,v) de la imagen de salida.
- La imagen de salida resulta mas chica que la imagen original.
- Definimos:
 - **Stride:** Numero de pixeles que desplazo el filtro en X como en Y.
 - **Padding:** Agregado de pixeles en los bordes de la imagen original para preservar las dimensiones en la imagen de salida.
 - **Ganancia:** Es la suma de todos los coeficientes. Vamos a buscar siempre filtros de ganancia unitaria para evitar la saturación de luminancia.

$$H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

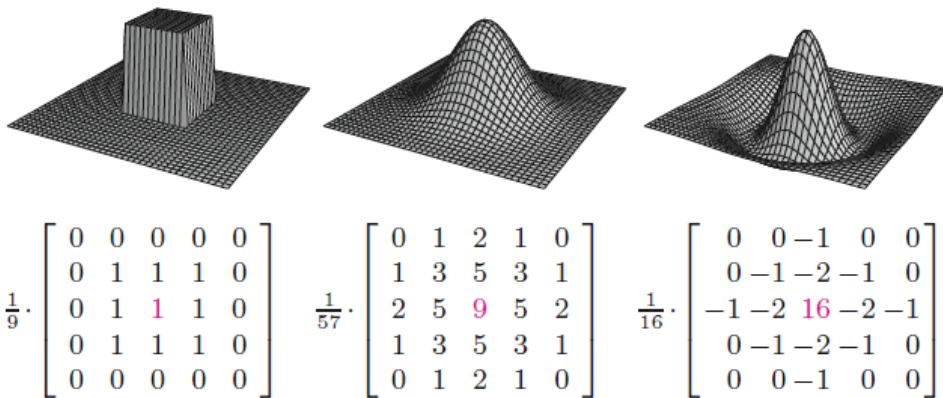


FILTROS LINEALES



Correlación
con $k(x,y)$

Convolución
con $k'(x,y)$



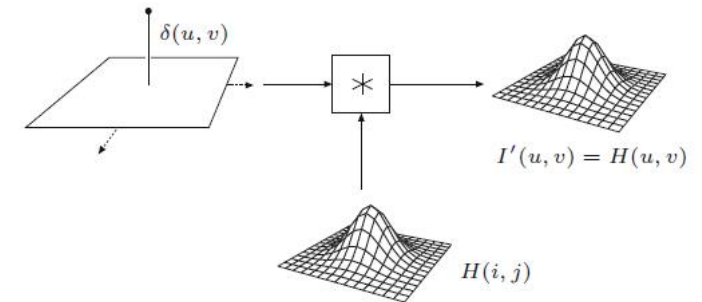
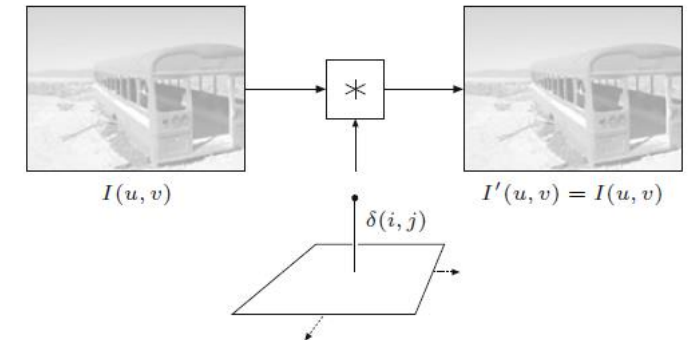
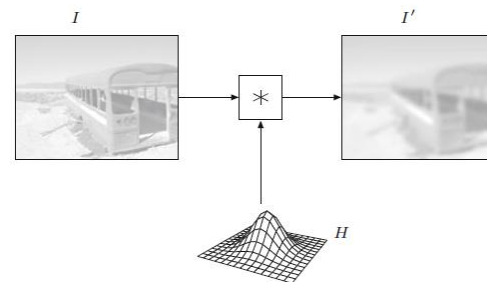
- En la practica se invierten los offsets sobre $I(i,j)$

$$\begin{aligned} I'(u,v) &= \sum_{(i,j) \in R_H} I(u-i, v-j) \cdot H(i,j) \\ &= \sum_{(i,j) \in R_H} I(u+i, v+j) \cdot H(-i, -j) \\ &= \sum_{(i,j) \in R_H} I(u+i, v+j) \cdot H^*(i,j). \end{aligned}$$

Esto se escribe como

$$I' = I * H$$

Donde $*$ es el operador convolución



¿Qué pasa si el núcleo es simétrico?



FILTROS LINEALES

Propiedades de la convolución:

1. Conmutativa:

$$h * f = f * h$$

2. Linealidad:

$$s \cdot (f * h) = f * (s \cdot h) = (s \cdot f) * h$$

3. Asociativa

$$f * (h_1 * h_2) = (f * h_1) * h_2$$

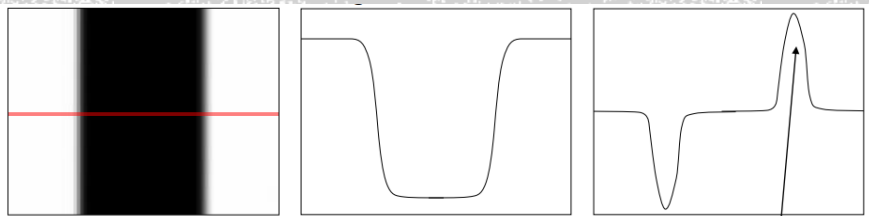
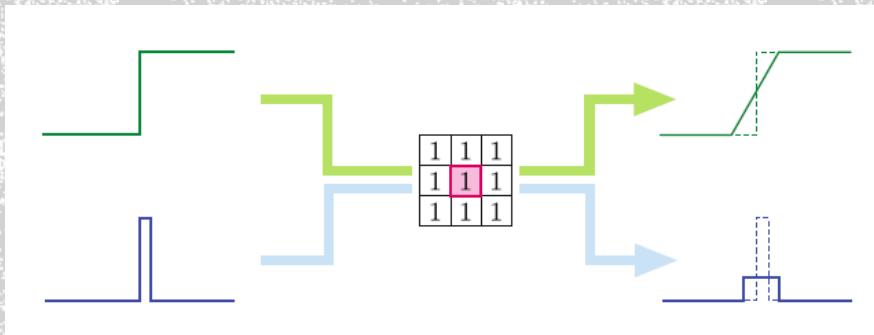
4. Invariante en el tiempo (o a desplazamientos para el caso de imágenes):

$$f[u - d, v - d] * h = (f * h)[u - d, v - d]$$

- Una diferencia importante entre correlación y convolución es que la primera no es asociativa para operaciones con imágenes.



EJEMPLOS DE NÚCLEOS TÍPICOS



Derivada positiva y negativa para detectar bordes (transiciones)

- Algunos ejemplos de núcleos (Kernels) típicos:

- $Enfoque: \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
 $Desenfoque: \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
 $Repujado: \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$

- $LoG: \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
 $Sharpen: \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ -1 & -2 & 1 \end{bmatrix}$
 $Gauss: \frac{1}{121} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$

- $Sobel: \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
 $Prewitt y: \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$
 $Prewitt x: \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

- La suma de los filtros debe dar '1' para que la respuesta del filtro no tenga efecto en regiones homogéneas
- Sobel y Prewitt se utilizan para la detección de bordes.
- Sobel y Prewitt son a su vez ejemplos de filtros pasa alto, direccionables.

LOG (LAPLACIANO DEL GAUSSIANO)

- Podemos definir matemáticamente el Laplaciano como

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

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

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

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

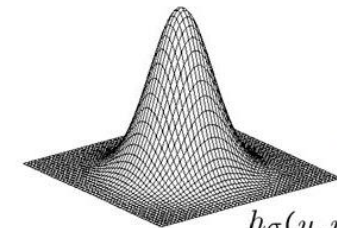
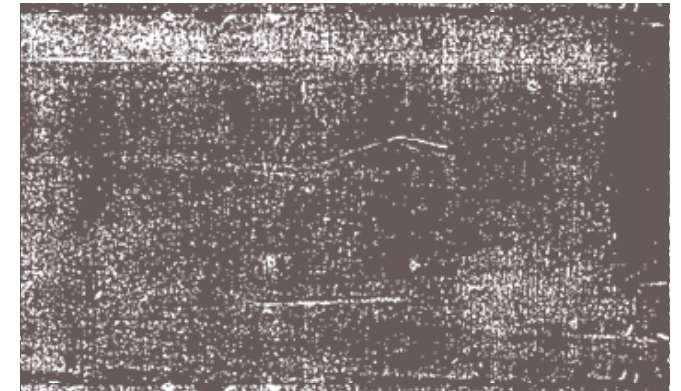
- Detecta los cruces por cero (cambio de signo)
- No usa dos núcleos con orientaciones distintas (como Prewitt, Sobel) y pierde información de orientación
- Es isotrópico → magnitud de borde uniforme en toda dirección
- Es muy sensible al ruido

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

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

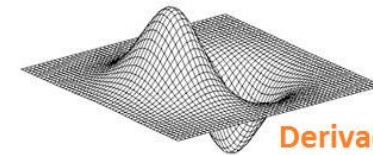
Matemáticamente, $LoG(x, y) = \frac{1}{\pi\sigma^4} \left[1 - \frac{x^2+y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$...y podemos muestrear...

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

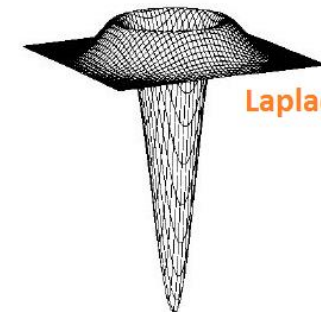


Gaussiana

$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



$\frac{\partial}{\partial x} h_\sigma(u, v)$
Derivada de Gaussiana



$\nabla^2 h_\sigma(u, v)$
Laplaciano de Gaussiana (LoG)



SENSIBILIDAD AL RUIDO

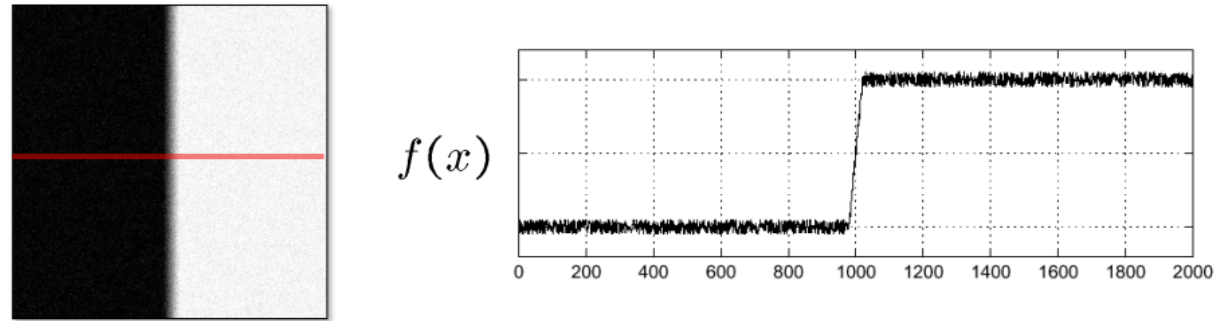
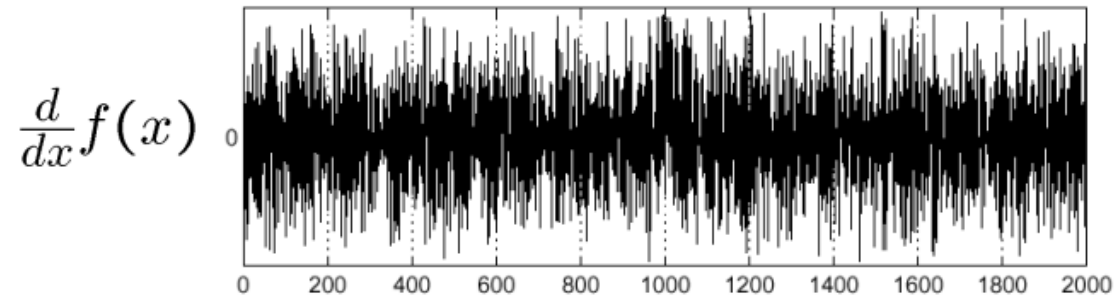


Imagen con ruido y un perfil de la región marcada



Mismo perfil en la imagen luego de aplicar un filtro de bordes

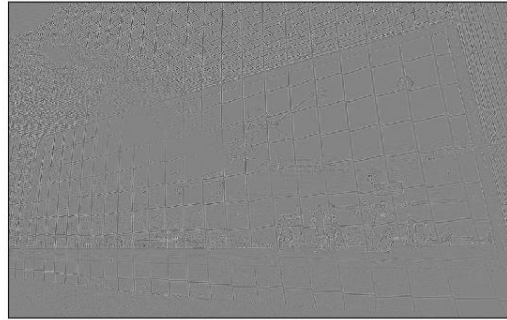


FP64

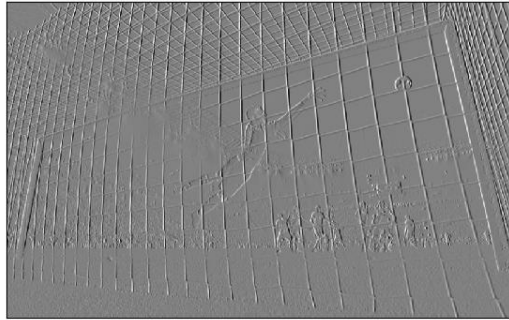
Original



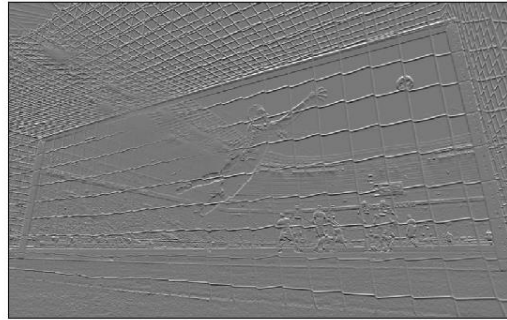
Laplaciano



Sobel X



Sobel Y

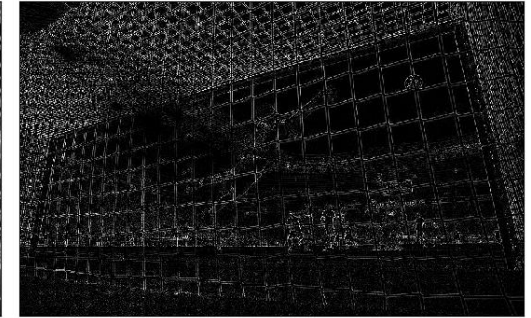


UINT8

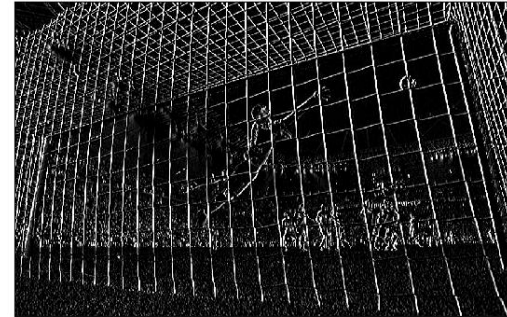
Original



Laplaciano



Sobel X



Sobel Y



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`.



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. Requiriendo así $2K$ operaciones.
- Un núcleo que permita 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:

$$K = v h^t$$

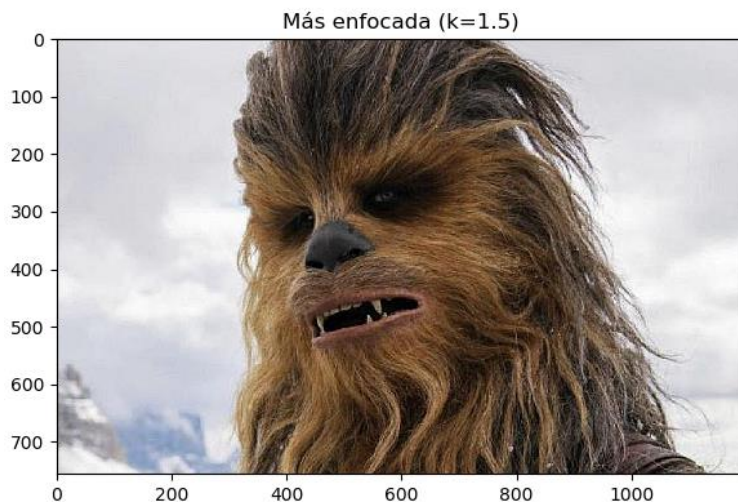
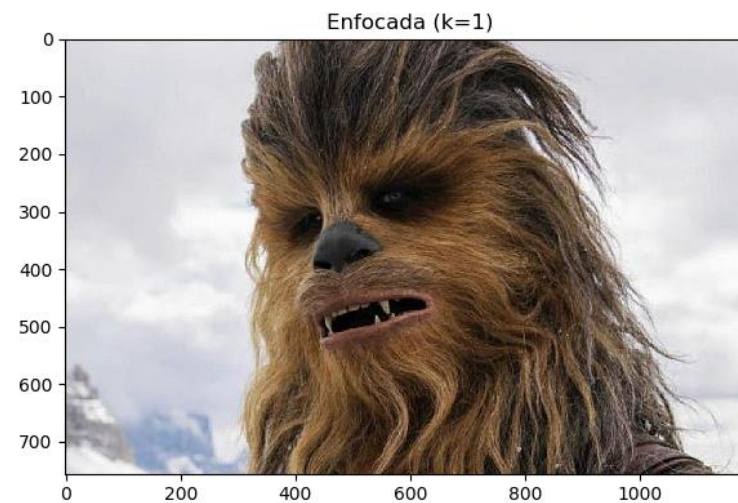
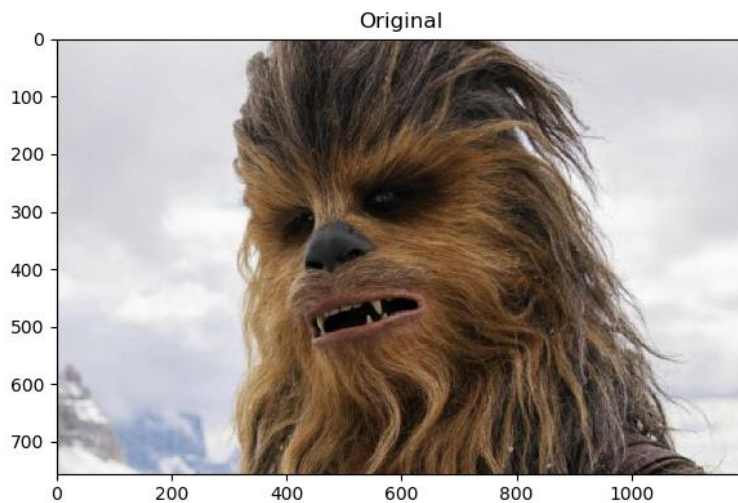
Ejemplos:

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

¿Cómo sabemos si un filtro es separable?

- Por inspección
- 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 σ_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.





UNSHARP MASKING

- Es un algoritmo que tiene enfocar o mejorar los bordes con un filtro de suavizado

- Desenfoque de la imagen con un filtro suavizador (ej. Gaussiano)

$$f(x, y) \rightarrow f_B(x, y)$$

- Resta de la imagen original con la suavizada (sobreviven componentes de alta frecuencia)

$$m(x, y) = f(x, y) - f_B(x, y)$$

- Agregar esa máscara a la imagen original (realce de las componentes de alta frecuencia)

$$g(x, y) = f(x, y) + k \cdot m(x, y)$$

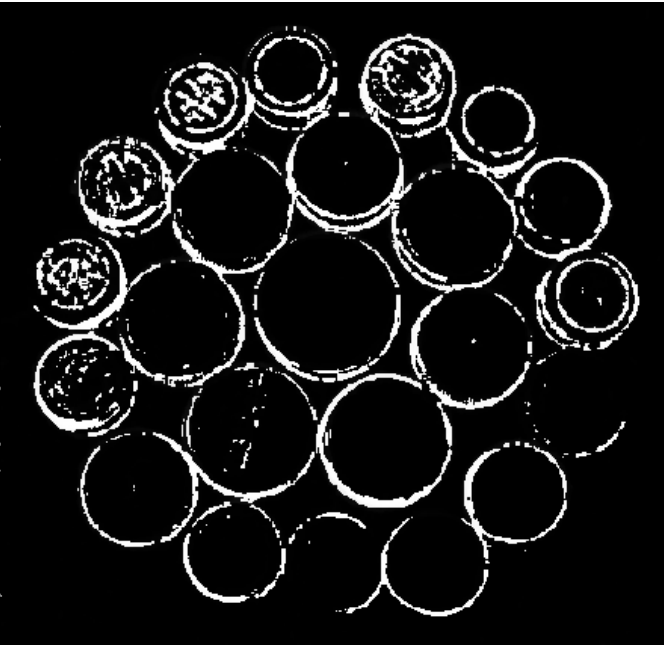
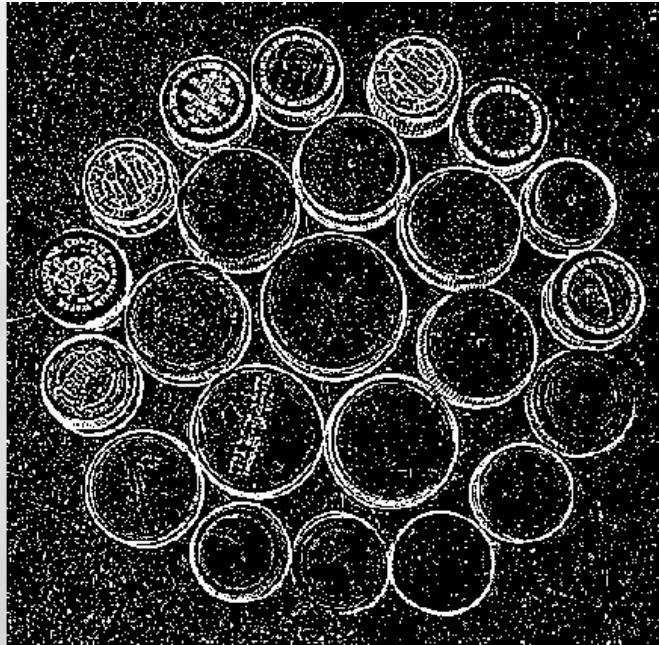
De manera genérica:

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

$k = 1$: Máscara de desenfoque (unsharp masking)

$k > 1$: Filtrado de alto impulso (highboost filtering)





DOG (DIFFERENCE OF GAUSSIAN)

- 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

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



FILTROS NO LINEALES

Realizan operaciones no lineales entre pixeles vecinos para obtener el pixel de salida. Dentro de esta familia los utilizados mas frecuentemente son:

- Filtros de mínima y máxima

$$I'(u, v) = \min_{(i,j) \in R} \{I(u + i, v + j)\}$$

$$I'(u, v) = \max_{(i,j) \in R} \{I(u + i, v + j)\}$$

- Filtro de Mediana

$$I'(u, v) = \text{median}_{(i,j) \in R} \{I(u + i, v + j)\}$$

Para el caso en que el num. de elementos sea par se toma el promedio de los dos centrales generando así un nivel de intensidad nuevo que no estaba presente en la imagen.

- Filtro de Mediana ponderada

Se multiplica la región por una ventana de pesos, generalmente ponderando los pixeles centrales.

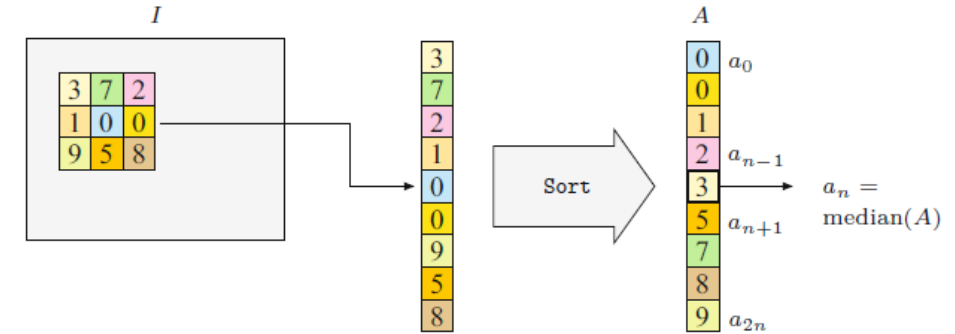


Fig.1: Ejemplo de un filtro de mediana clásico

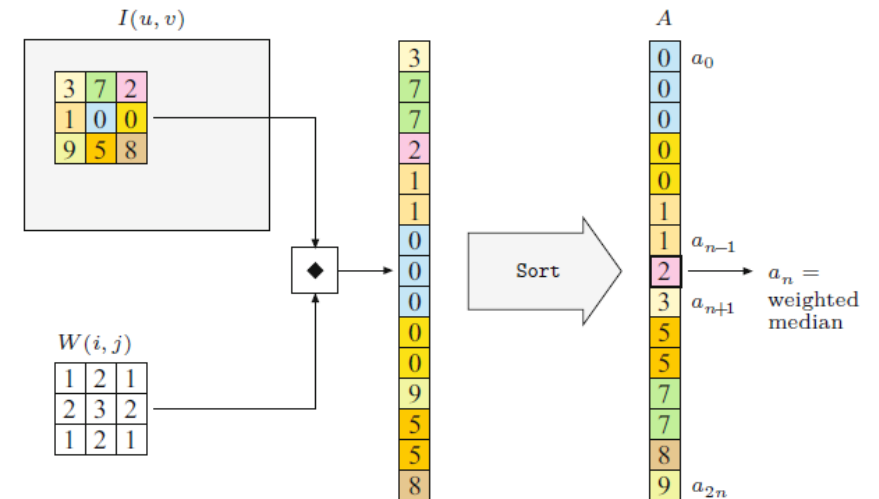


Fig. 2: Ejemplo de un filtro de mediana ponderada



FILTROS NO LINEALES



Fig.1: Imagen con ruido Salt & Pepper



Fig.2: Filtrado con desenfoque uniforme

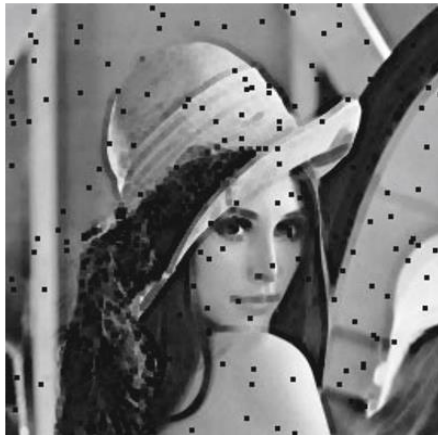


Fig.3: Filtrado con filtro de mínima (izquierda) y máxima (derecha)

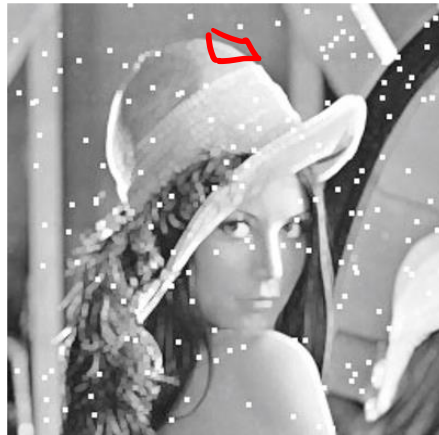
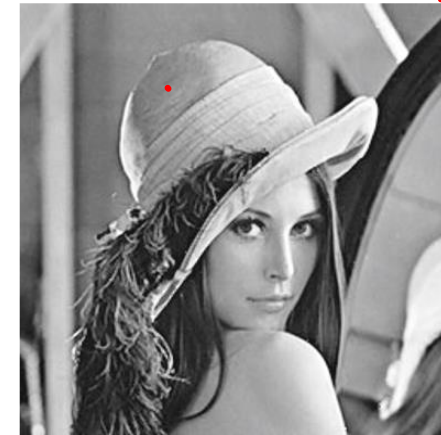


Fig.4: Filtrado con filtro de mediana (izquierda) y mediana ponderada



$$W = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



FOURIER

- La transformada de Fourier se utiliza:

1. Para analizar el comportamiento en frecuencia de sistemas y señales.
2. En imágenes (2D DFT), para encontrar las componentes de frecuencia. El algoritmo que implementa esto de forma eficiente es la Fast Fourier Transform (FFT).

$$H(k) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} h(x) e^{-j \frac{2\pi k x}{N}}; \text{ vale } \forall k, \text{ pero solo tiene sentido en } k \in \left[-\frac{N}{2}, \frac{N}{2}\right]$$

- $DFT \rightarrow N^2 \text{ operaciones}; FFT \rightarrow N \log_2 N$

Para 2 dimensiones nos queda:

$$G(m, n) = \frac{1}{\sqrt{MN}} \cdot \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} g(u, v) \cdot e^{-i2\pi \frac{mu}{M}} \cdot e^{-i2\pi \frac{nv}{N}}$$

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

1. En bordes
2. 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:

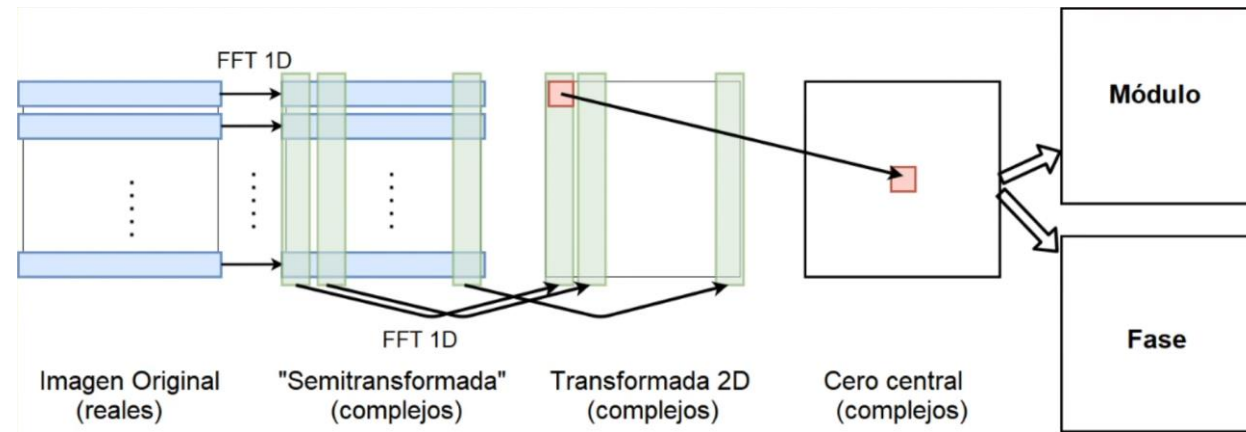
$$f(x)h(x) \rightarrow F(w) * H(w)$$

- Escalamiento:

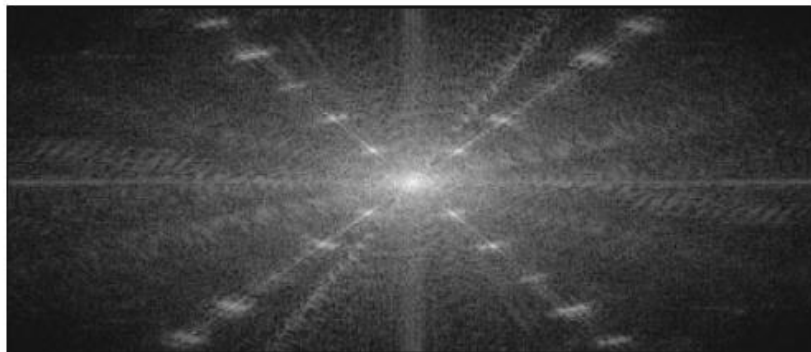
$$f(ax) \rightarrow \frac{1}{a} F\left(\frac{w}{a}\right)$$

- Diferenciación:

$$f'(x) \rightarrow jwF(w)$$

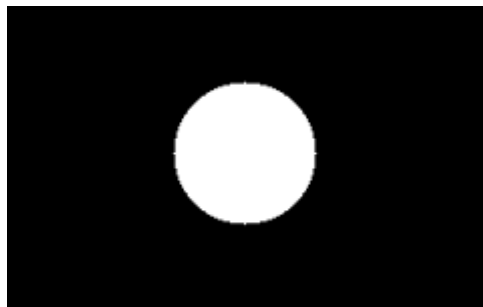


(a)

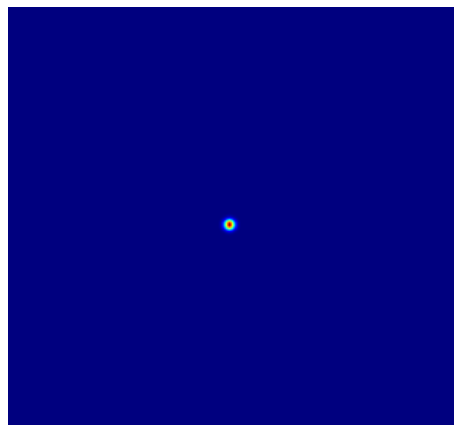


(b)

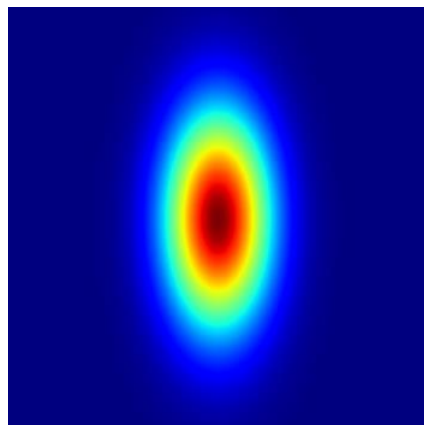




Filtros pasa bajos (PB), izquierda y pasa altos (PA), derecha.
Ambos definidos en el dominio espectral.



Filtro Gaussiano
(Grafico del kernel)



Transformada de Fourier del Filtro
Gaussiano

FOURIER: FILTRADO

Los filtros en el dominio espectral se pueden construir a partir de un filtro en el dominio espacial y aplicarle la transformada de Fourier.

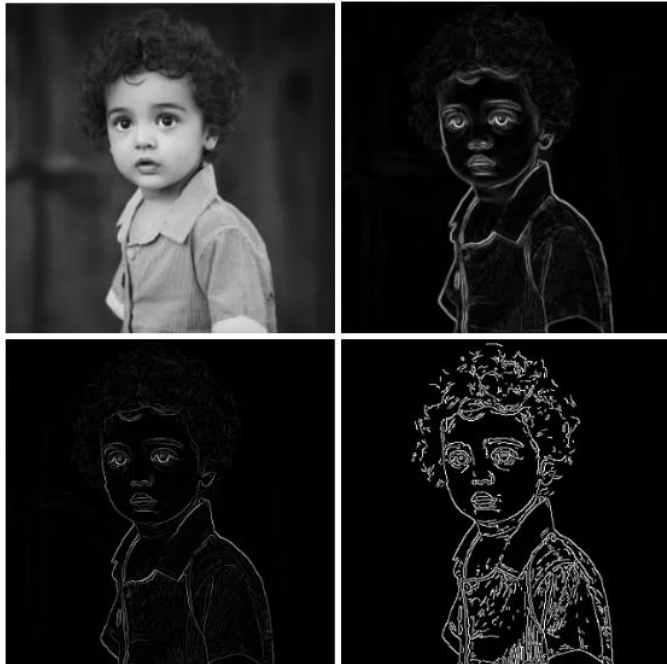
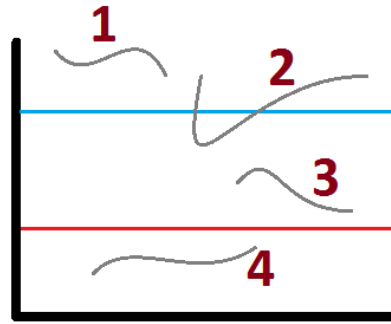
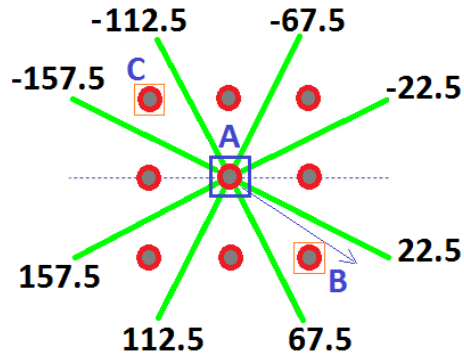
Otra opción, un poco mas compleja, es construirlo directamente en el dominio espectral. Pero es importante diseñar el filtro de forma tal que sea realizable en el dominio espacial.

Algoritmo de filtrado

1. Transformamos la imagen al dominio de frecuencia espacial.
2. Detectamos las componentes que queremos eliminar.
3. Hacemos el producto elemento a elemento entre la TF de la imagen y el filtro.
4. Hacemos la transformada inversa para volver al dominio original



DETECTOR DE BORDES DE CANNY



- Algoritmo desarrollado por John F. Canny en 1986

- Consta de varias etapas:

1. Reducción de ruido

- Canny usa un filtro Gaussiano, en general de 5x5

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

2. Encontrar el gradiente de intensidad de la imagen

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

$$G = \sqrt{G_x^2 + G_y^2} : \text{Magnitud} \quad \theta = \text{atan}\left(\frac{G_y}{G_x}\right) : \text{Dirección}$$

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:

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

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.



TP2 - CLASE 3

■ Objetivo:

Implementar un detector de máximo enfoque sobre un video aplicando técnicas de análisis espectral similar al que utilizan las cámaras digitales modernas. El video a procesar será: "focus_video.mov"

2. Cambiar la metrica de enfoque eligiendo uno de los algoritmos explicados en el apéndice de:

1. Se debe implementar un algoritmo que dada una imagen, o región, calcule la métrica propuesta en el paper "Image Sharpness Measure for Blurred Images in Frequency Domain" y realizar tres experimentos:

1. Medición sobre todo el frame.
2. Medición sobre una ROI ubicada en el centro del frame. Area de la ROI = 5 o 10% del area total del frame.
3. Medición sobre una matriz de enfoque compuesta por un arreglo de $N \times M$ elementos rectangulares equiespaciados. N y M son valores arbitrarios, probar con varios valores 3x3, 7x5, etc ... (al menos 3)

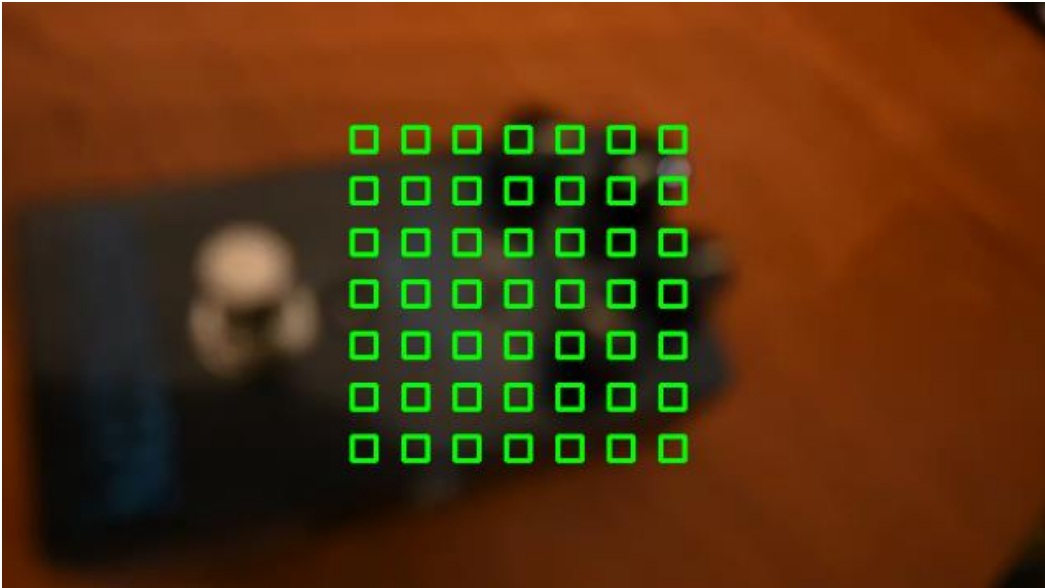
Para cada experimento se debe presentar :

- Una curva o varias curvas que muestren la evolución de la métrica frame a frame donde se vea claramente cuando el algoritmo detecto el punto de máximo enfoque.
- Video con la ROI o matriz, graficada en rojo y superpuesta al video original para los frames que no están en foco y verde para los frames donde se detecta la condición de máximo enfoque.

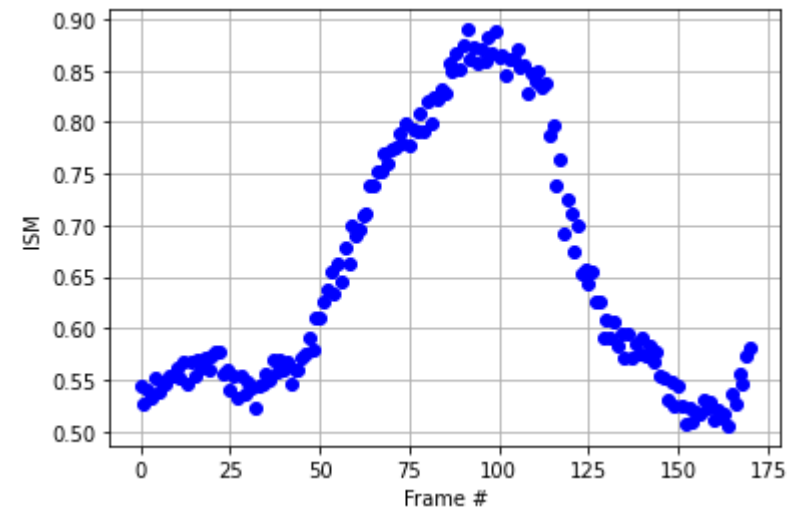
El algoritmo de detección a implementar debe detectar y devolver los puntos de máximo enfoque de manera automática.



TP2 - CLASE 3



Matriz de enfoque superpuesta a uno de los frames del video



Matriz de enfoque superpuesta a uno de los frames del video

Puntos extra: Aplicar unsharp masking para expandir la zona de enfoque y devolver.

