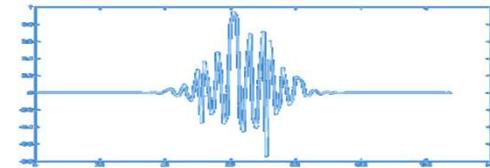




Facultad de Ciencias Exactas, Físicas y Naturales  
Departamento de Matemáticas  
**LAPSE**  
Laboratorio de Procesamiento de Señales  
Av. Vélez Sarsfield 1611 - Ciudad Universitaria – Córdoba  
República Argentina  
+54 351 434 4982



## Curso Procesamiento de Imágenes en Python

Dra. Valeria S. Rulloni - Dra. Laura M. Vargas

Congreso- Escuela en Estadística Espacial  
23,24 y 25 de septiembre

Jornada II: IMÁGENES- Mejora y  
Filtros

# Tipos de Transformaciones de Imagen

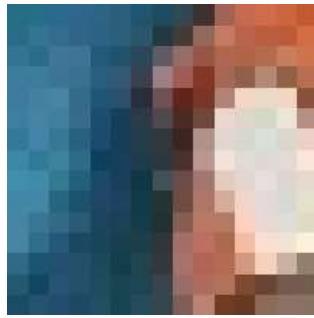
## Revisión

# Qué es una imagen?



Una imagen en color es un arreglo tridimensional

# Qué es una imagen?



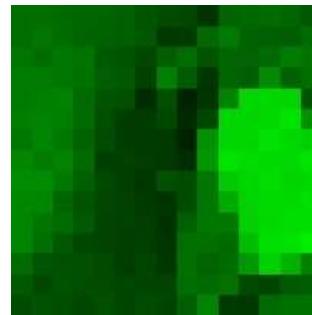
Mezcla

Cuáles son los valores  
communes de  
intensidad?

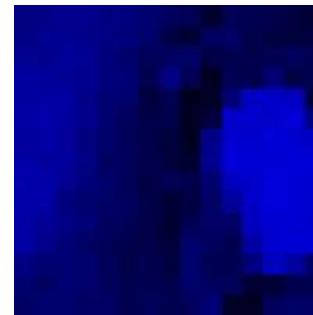
rojo



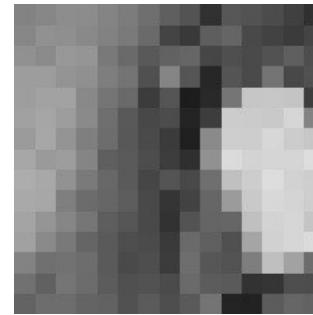
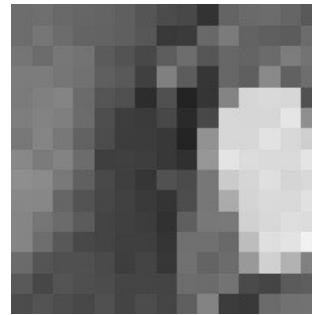
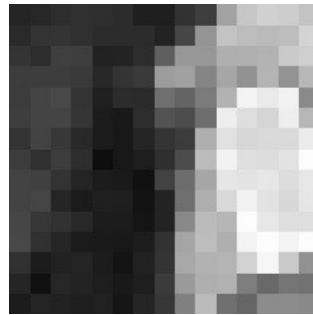
green



blue



Viendo los colores



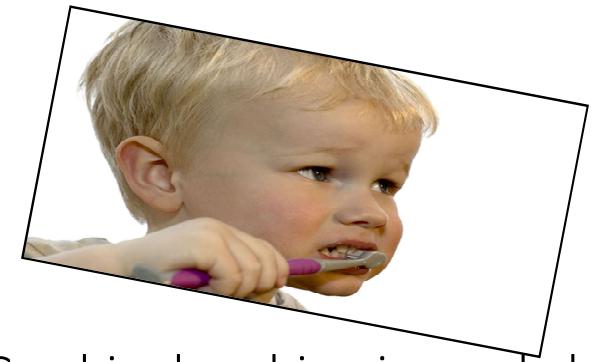
Viendo los valores de intensidad de cada canal  
en la gama de grises

Cada canal de  
color  
corresponde a  
una matriz 2D

# Qué tipos de transformaciones se pueden hacer?



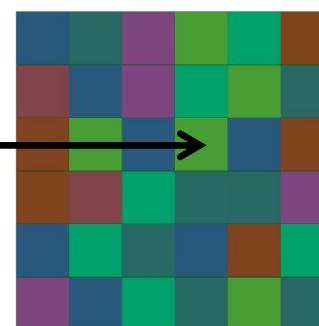
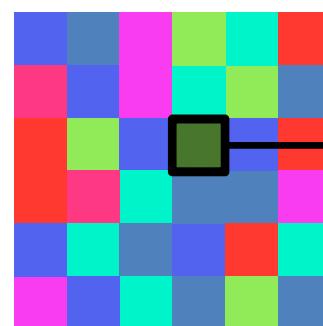
Cambiar los valores de las intensidades de los píxeles



Cambiar las ubicaciones de los píxeles

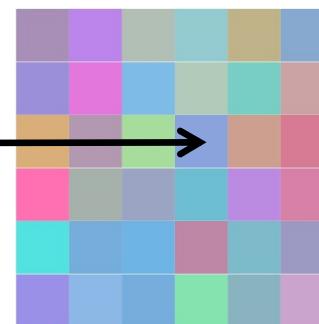
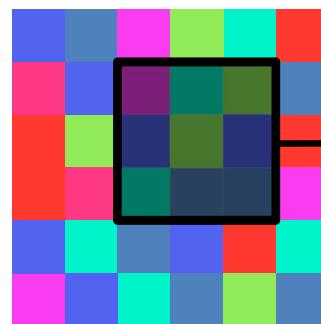
# Operaciones que cambian el valor de intensidad

Operación de Punto



Procesamiento de punto

Operación que tiene en cuenta los vecinos



Filtrado

# Procesamiento de Punto

# Ejemplos de Procesamiento de Punto

original



darken



lower contrast



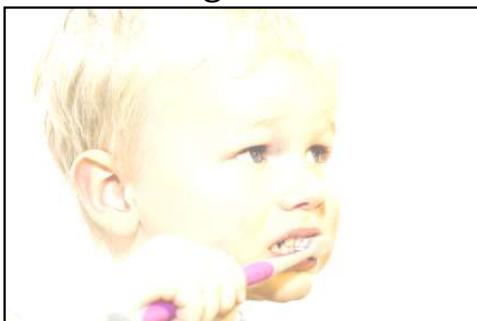
non-linear lower contrast



invert



lighten



raise contrast



non-linear raise contrast



Cómo se  
implementan?

# Ejemplos de Procesamiento de Punto

original



darken



lower contrast



non-linear lower contrast

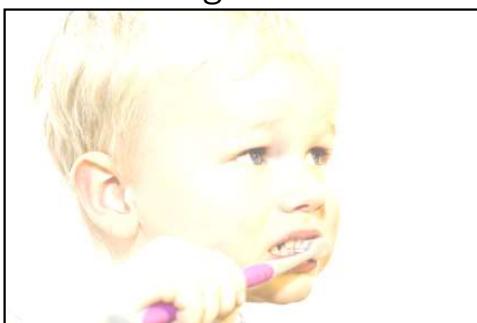


*x*

invert



lighten



raise contrast



non-linear raise contrast



Cómo se  
implementan?

# Ejemplos de Procesamiento de Punto

original



$$x$$

darken



$$x - 128$$

lower contrast



$$\frac{x}{2}$$

non-linear lower contrast



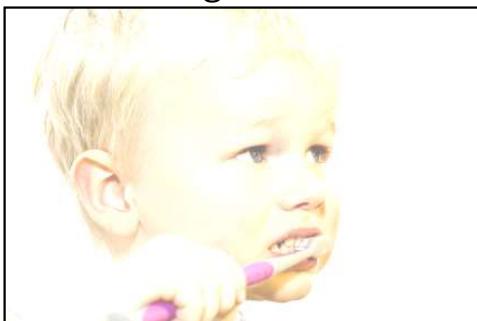
$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

invert



$$255 - x$$

lighten



$$x + 128$$

raise contrast



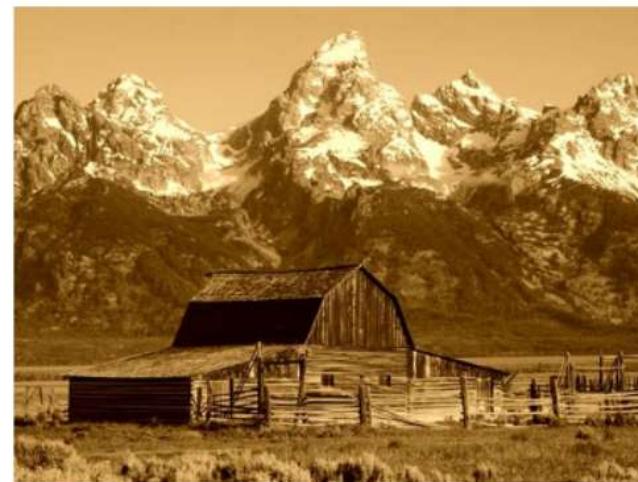
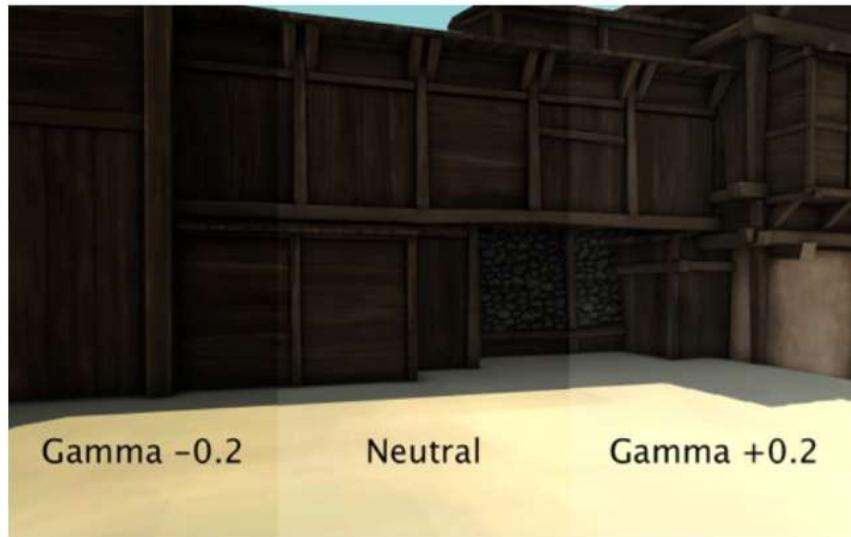
$$x \times 2$$

non-linear raise contrast



$$\left(\frac{x}{255}\right)^2 \times 255$$

# Ejemplos de Procesamiento de Punto



Filtrado

# Filtrado Lineal Invariante al corrimiento

- Reemplazar cada pixel por una combinación lineal de sus vecinos (que lo puede incluir a él mismo).
- La combinación está determinada por el kernel (máscara).
- El kernel es desplazado y pasa por todos los pixels de modo que todos sufren la misma combinación lineal, cada uno influenciado por sus vecinos.

# Ejemplo:

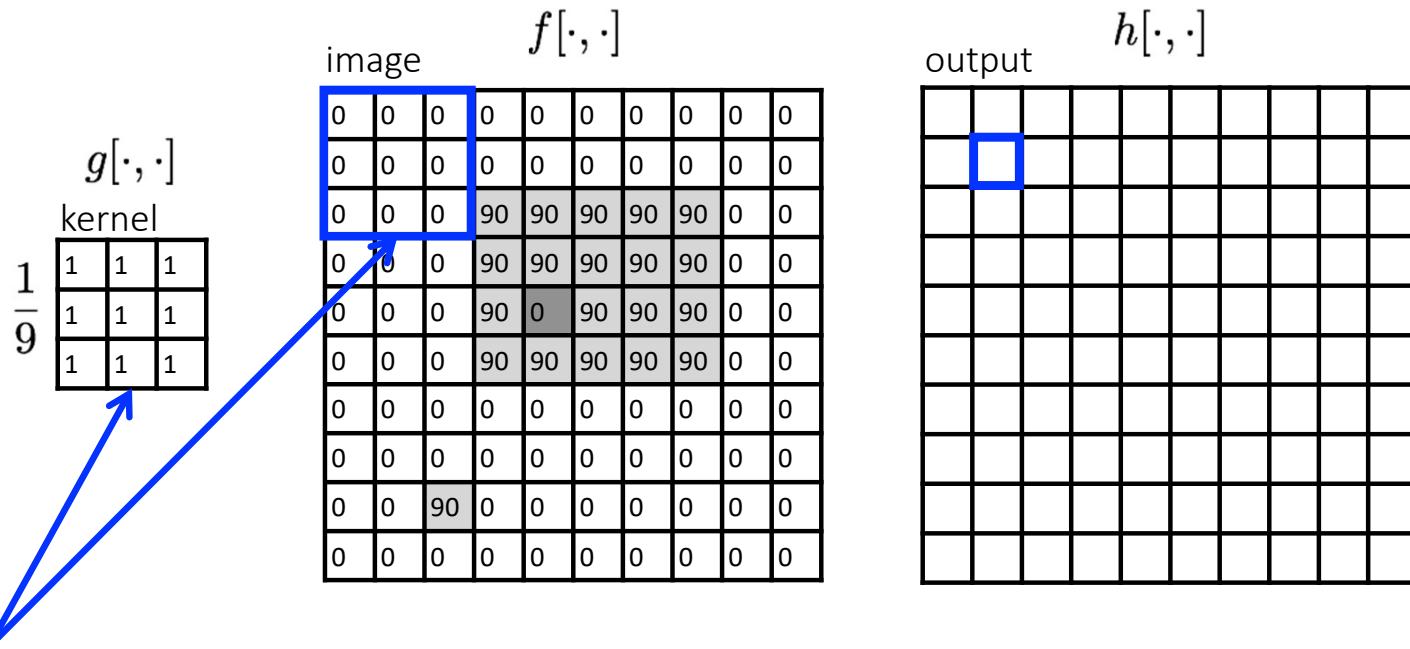
- Filtro 2D
- Conocido como filtro de valor medio (box filter).

kernel

$$g[\cdot, \cdot] = \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

- Ejemplo: 3x3
- Efecto de suavizado (efecto blurring)

# Cómo actúa?

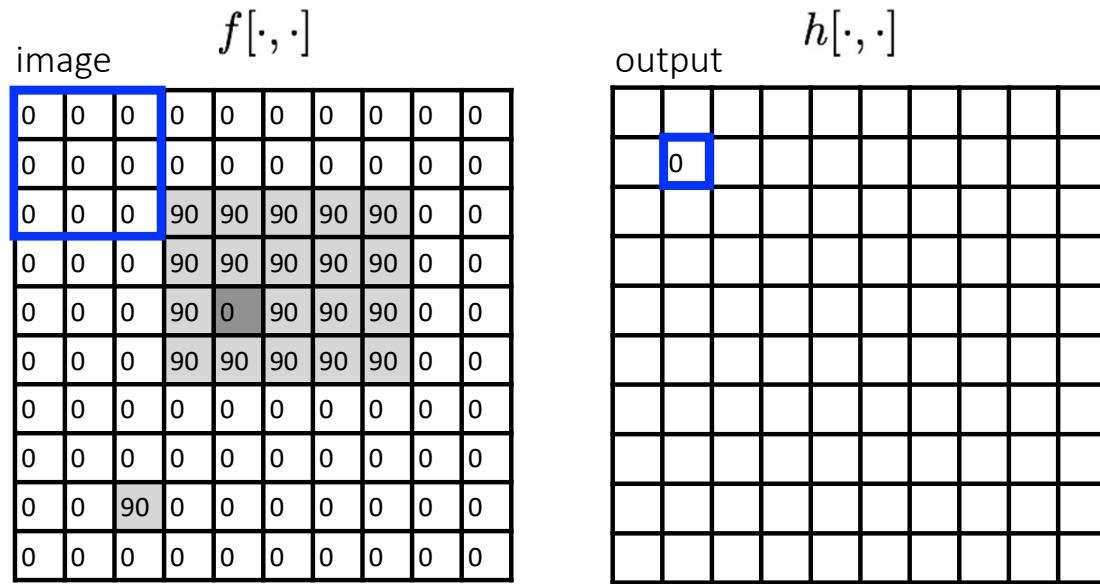


$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

salida      kernel      imagen (señal)

# Cómo actúa?

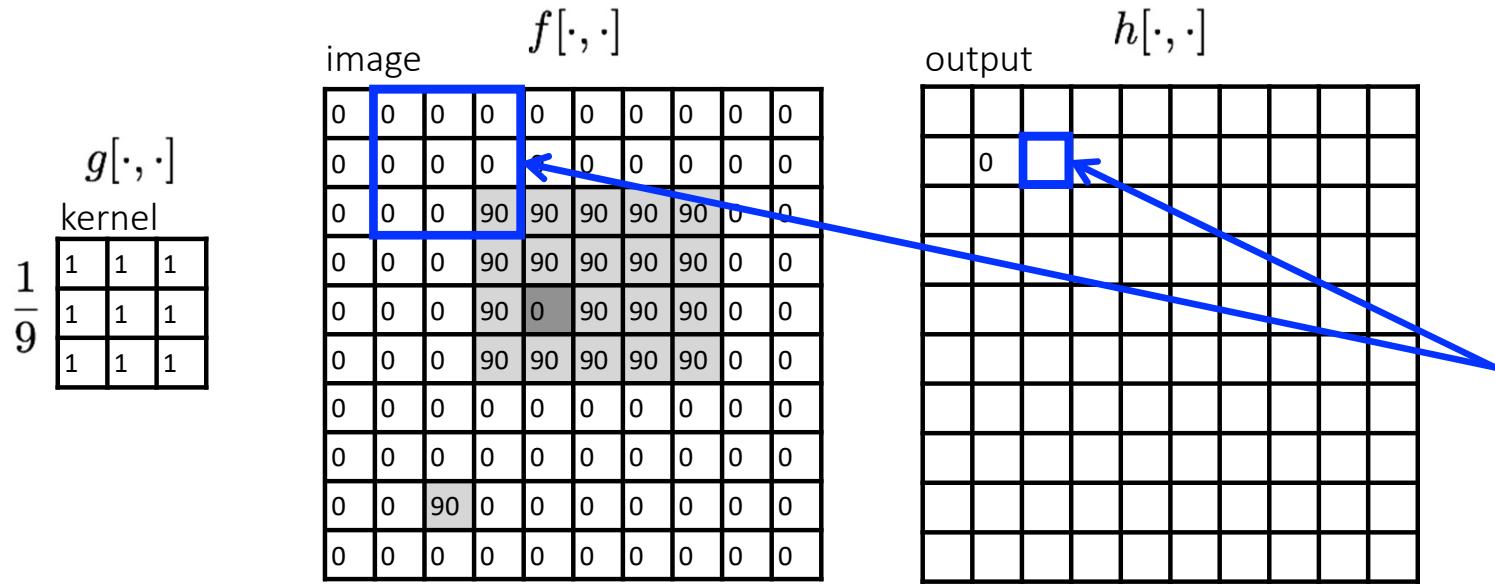
$$\frac{1}{9} \begin{matrix} g[\cdot, \cdot] \\ \text{kernel} \\ \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \end{matrix}$$



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Cómo actúa?



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

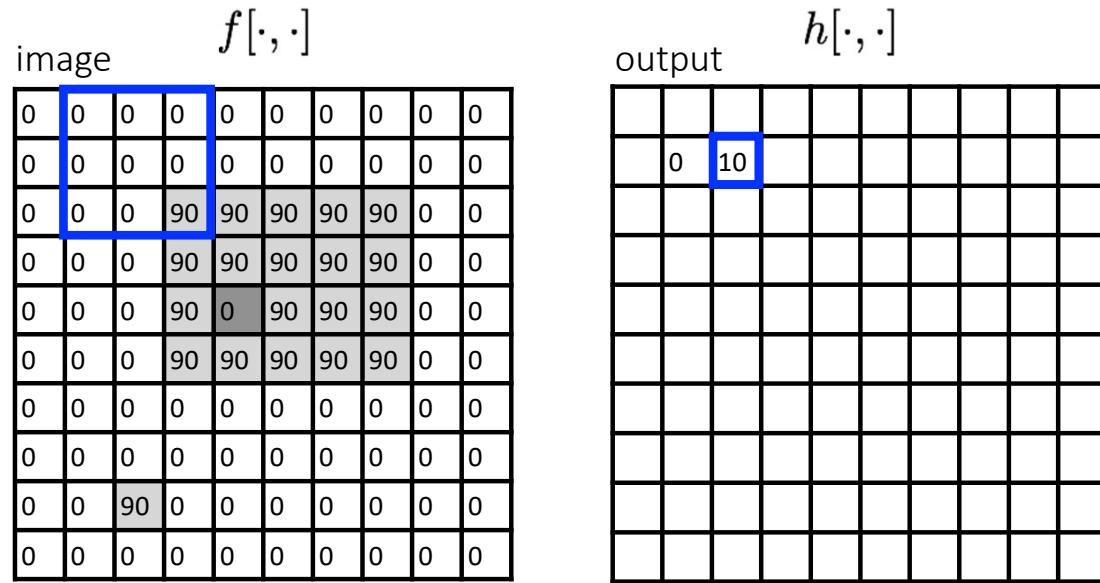
shift-invariant:  
cuando se  
desplaza el  
pixel, también  
se desplaza el  
kernel

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

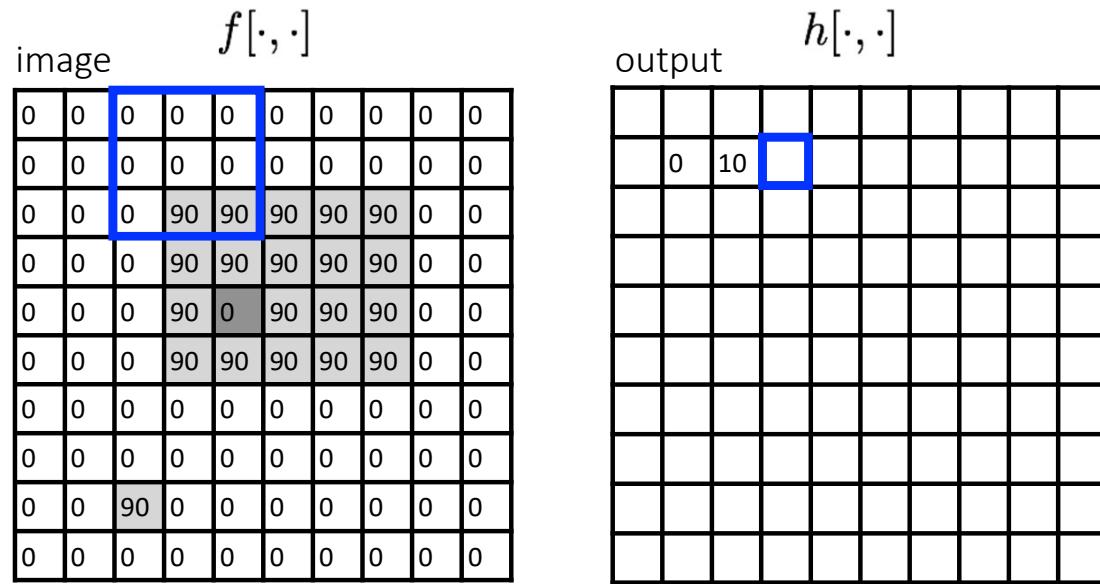
output      filter      image (signal)

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

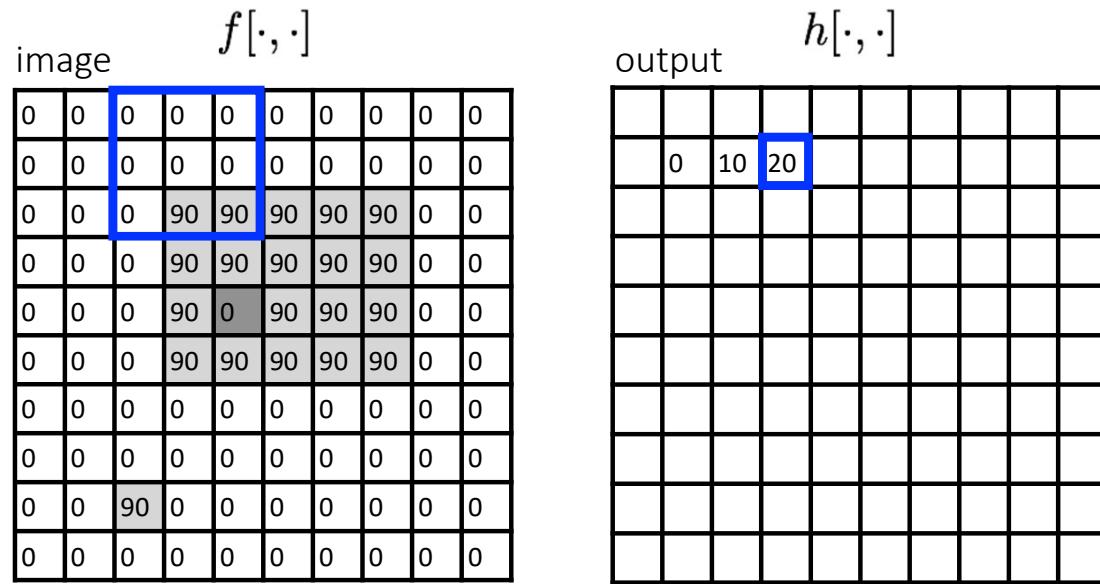
output       $k, l$       filter      image (signal)

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Cómo actúa?

The diagram illustrates a convolution operation mapping an input image to an output. The input image  $f[\cdot, \cdot]$  is a 10x10 grid where the central 3x3 region is highlighted in blue. This region contains values 90, 90, 90, 90, 90, 90, 90, 90, 90. The kernel  $g[\cdot, \cdot]$ , labeled as "kernel" with a value of  $\frac{1}{9}$ , is a 3x3 grid of ones. The output  $h[\cdot, \cdot]$  is a 10x10 grid where the element at position (3, 3) is highlighted in blue and has a value of 20. All other elements in the output grid are zero.

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

## Cómo actúa?

The diagram illustrates a convolution operation mapping an input image  $f[\cdot, \cdot]$  to an output  $h[\cdot, \cdot]$ .

**Input Image ( $f[\cdot, \cdot]$ ):**

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

**Kernel ( $g[\cdot, \cdot]$ ):**

1	1	1
1	1	1
1	1	1

**Output ( $h[\cdot, \cdot]$ ):**

			0	10	20	30			

The diagram shows the convolution process. The kernel is applied to the image with a stride of 2. The result is an output grid where the value at position (2, 3) is 30, highlighted by a blue box. The other values are 0 because the kernel only covers the first row of the third column in the image.

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

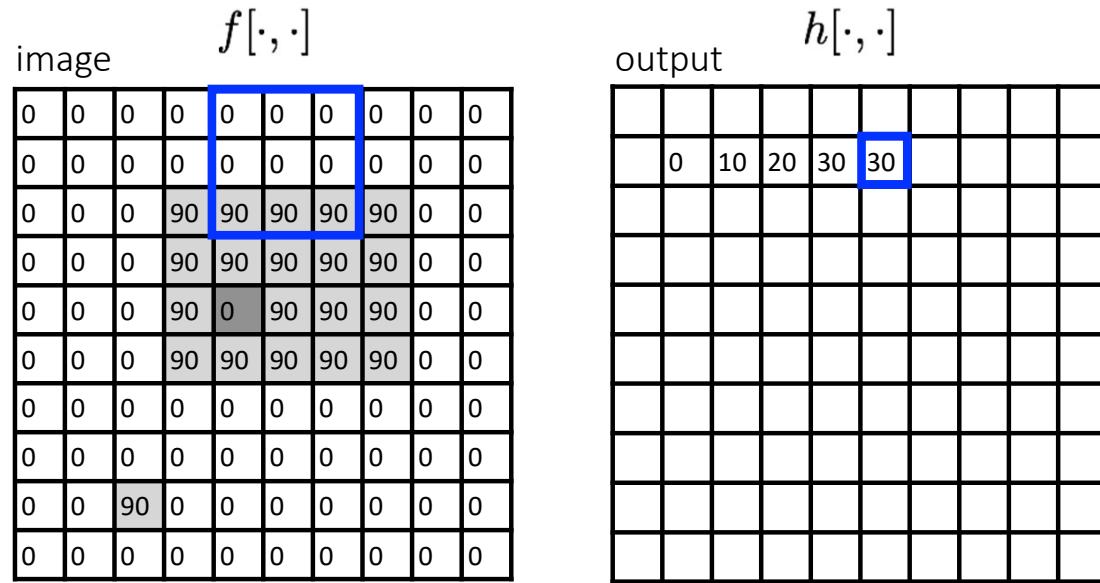
output       $k, l$       filter      image (signal)

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

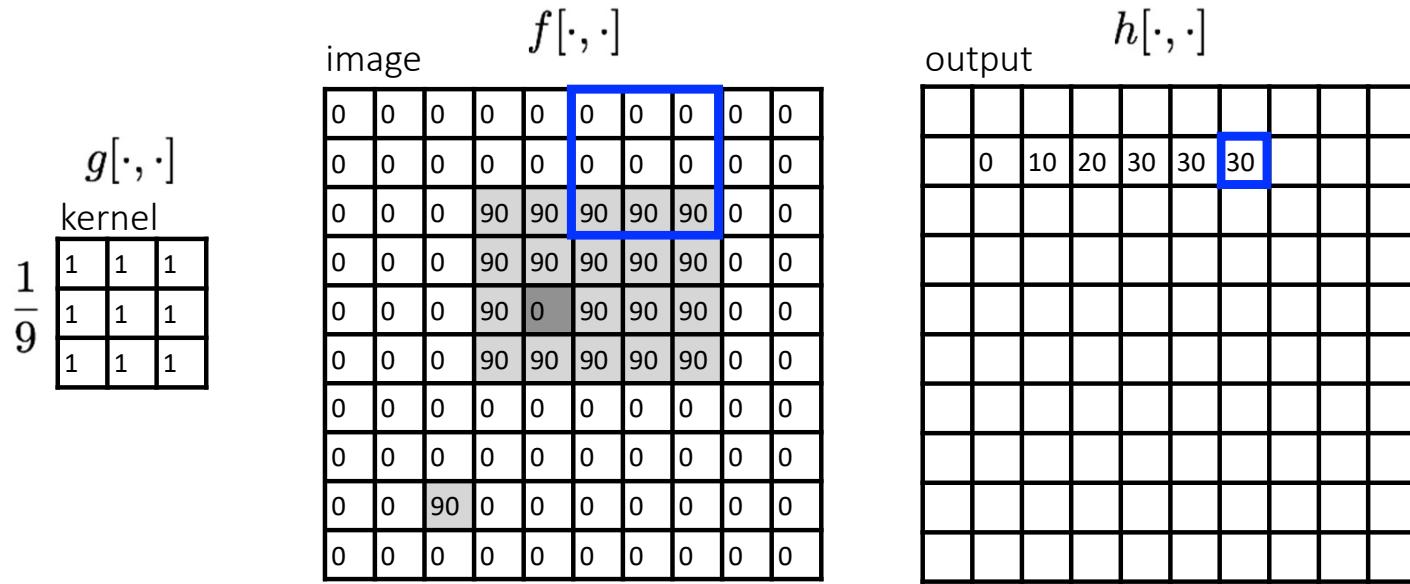
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

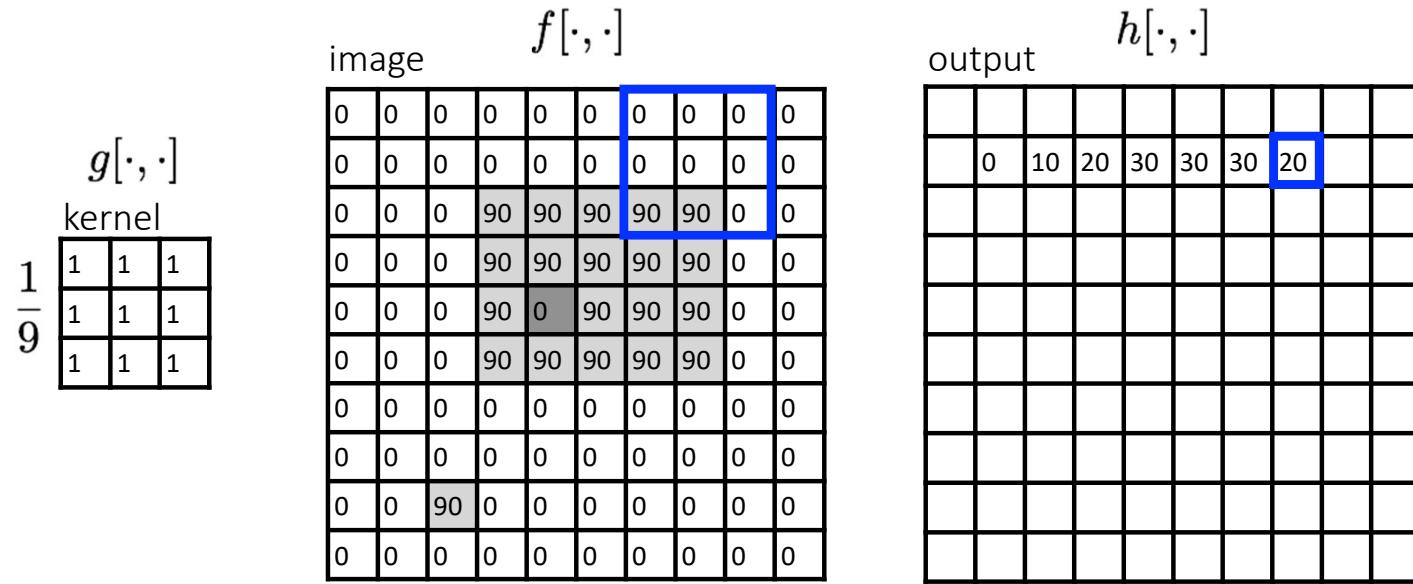
# Cómo actúa?



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output       $k, l$       filter      image (signal)

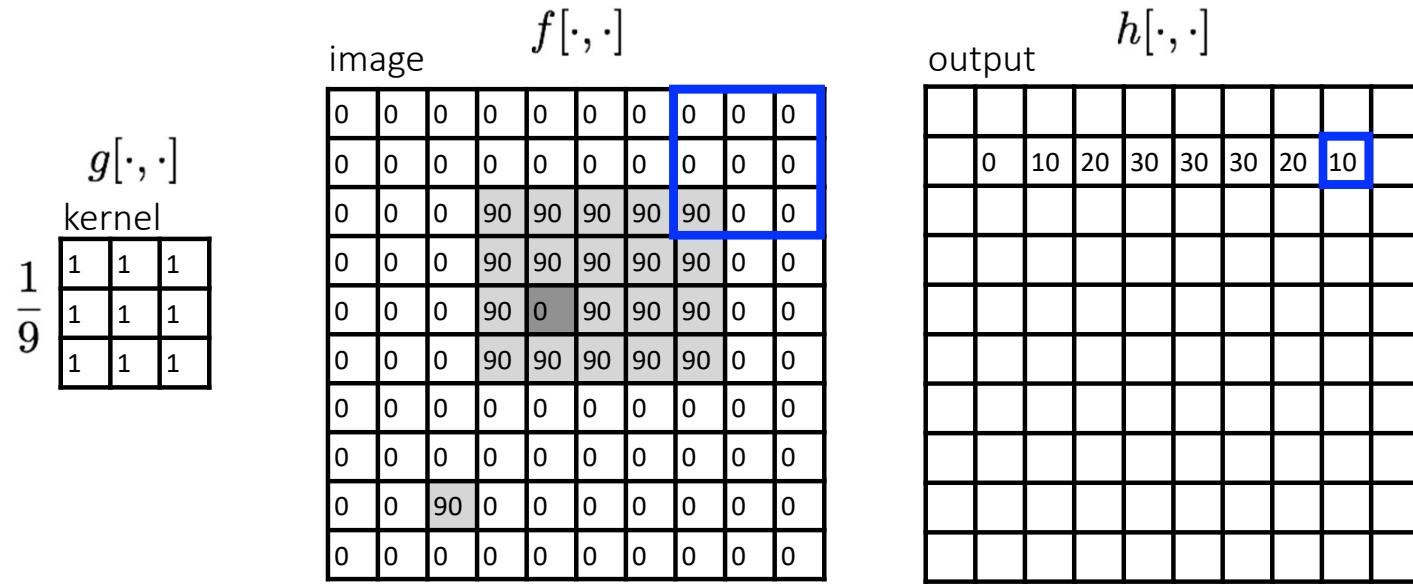
# Cómo actúa?



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Cómo actúa?



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

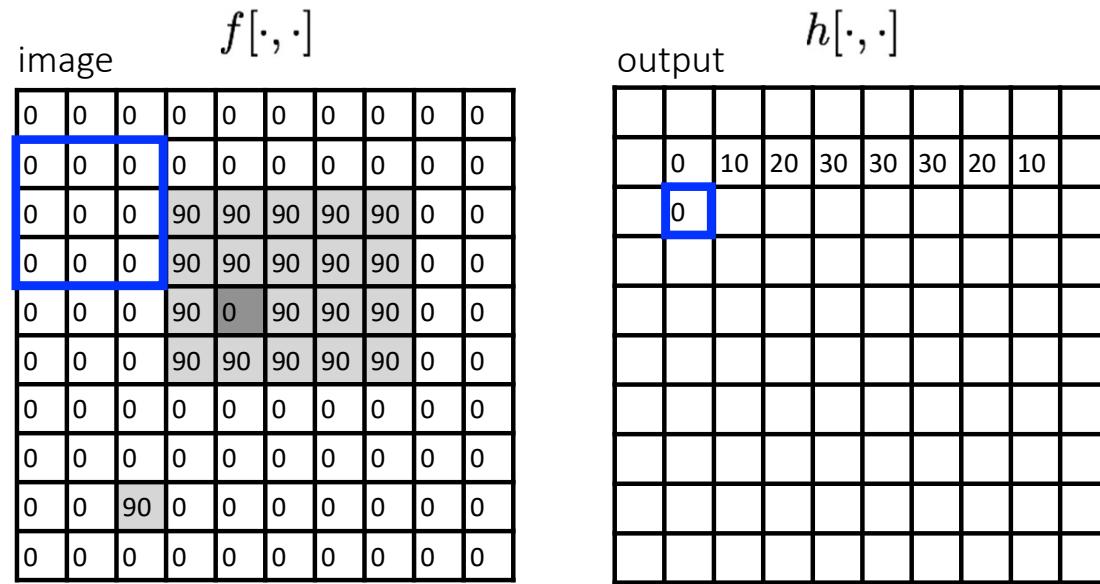
output      filter      image (signal)

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

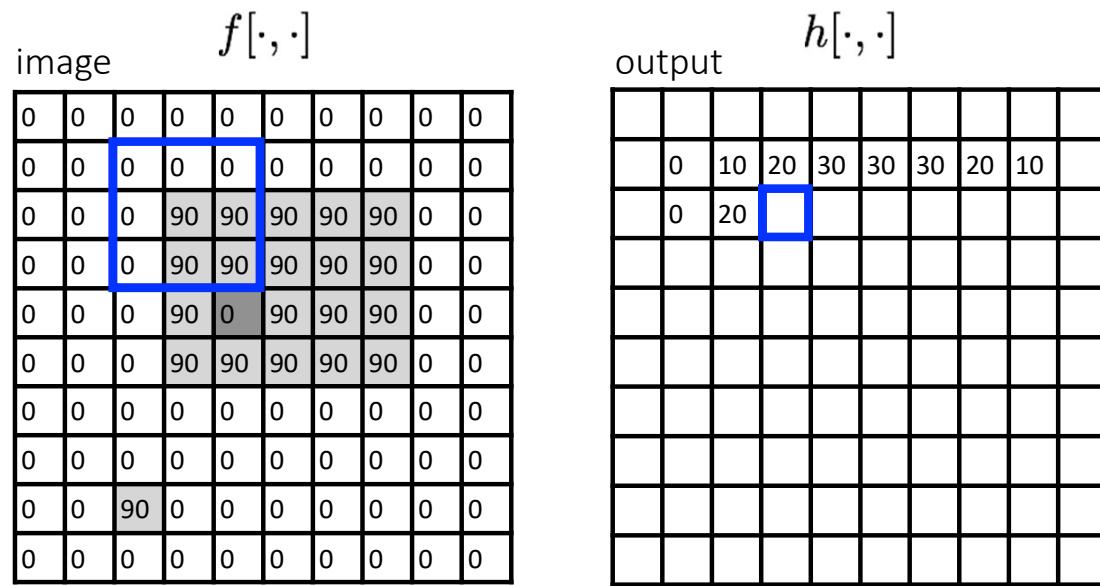
output       $k, l$       filter      image (signal)

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output       $k, l$       filter      image (signal)

## Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

1	1	1
1	1	1
1	1	1

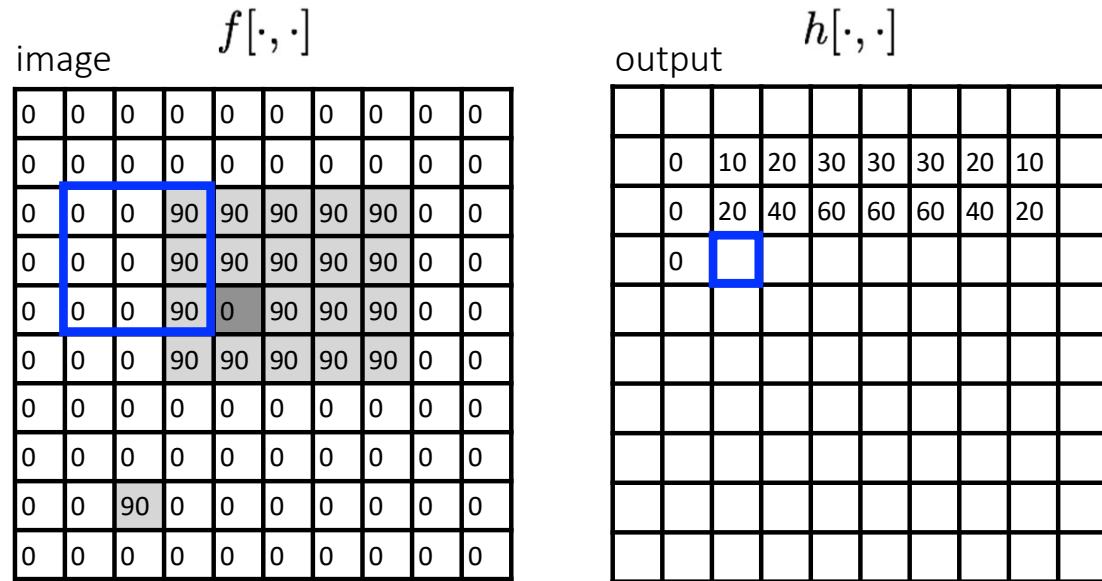
$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output       $k, l$       filter      image (signal)

# Cómo actúa?

The diagram illustrates a convolution operation mapping an input image  $f[\cdot, \cdot]$  to an output  $h[\cdot, \cdot]$  using a kernel  $g[\cdot, \cdot]$ .

**Input Image ( $f[\cdot, \cdot]$ ):**

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

**Kernel ( $g[\cdot, \cdot]$ ):**

1	1	1
1	1	1
1	1	1

**Output ( $h[\cdot, \cdot]$ ):**

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30							

A blue box highlights the central 3x3 receptive field of the output unit at position (3,3), which corresponds to the value 30 in the output grid. This receptive field covers the central 3x3 area of the input image, where all values are 90 except for the center cell which is 0.

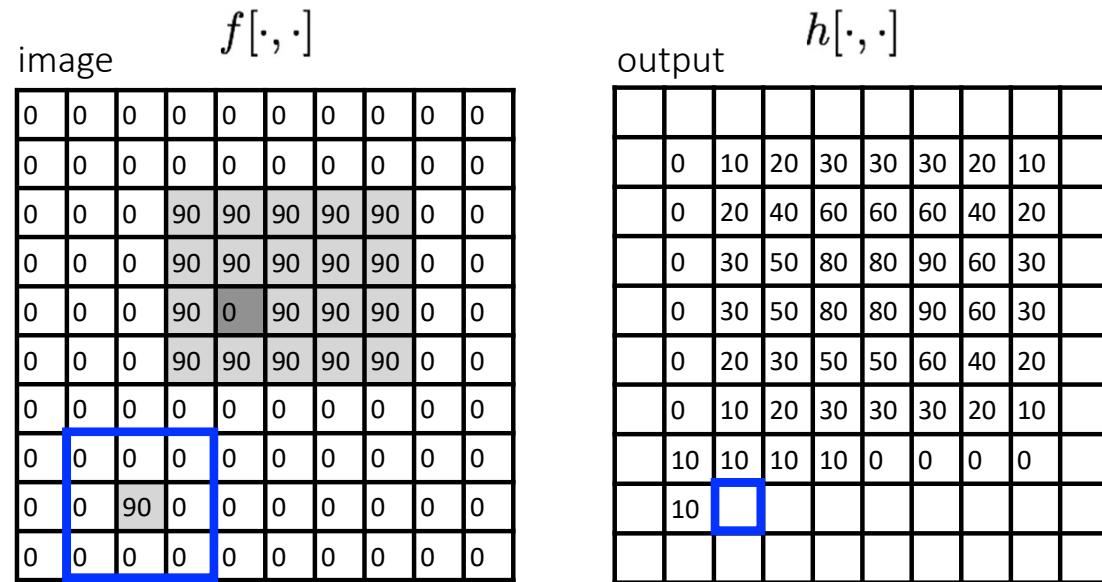
$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

# Cómo actúa?

$$g[\cdot, \cdot]$$

kernel

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



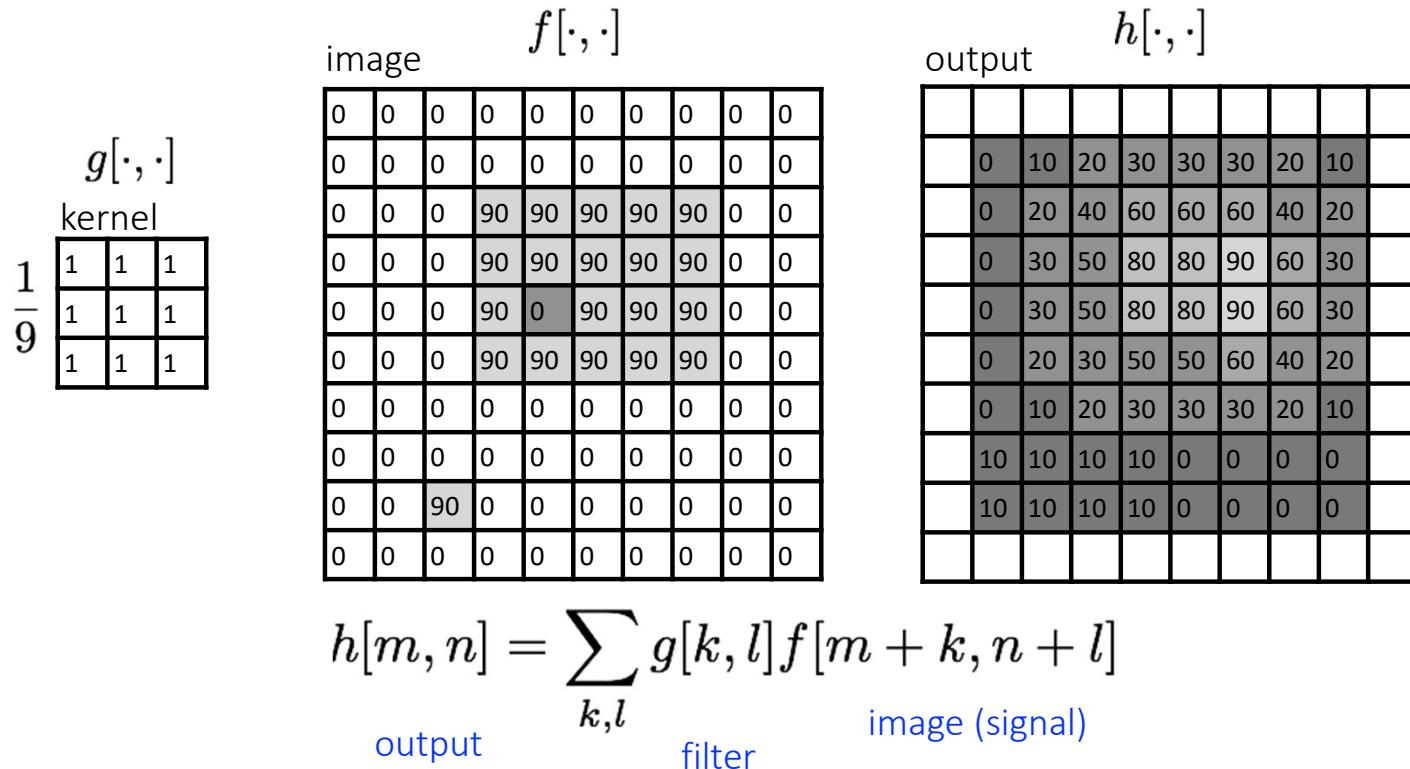
$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output       $k, l$       filter      image (signal)

# Cómo actúa?

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

... y el resultado final es:



**Observación:** no cambiaron los valores de las filas y columnas extremas.

**Solución:** agregar filas y columnas. El agregado dependerá del tamaño del kernel.

... y el resultado final es:

$$g[\cdot, \cdot]$$

kernel

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

image	f[·, ·]	output	h[·, ·]
$f[\cdot, \cdot]$	$h[\cdot, \cdot]$	$h[\cdot, \cdot]$	$h[\cdot, \cdot]$
$0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$	$0 \ 10 \ 20 \ 30 \ 30 \ 30 \ 30 \ 20 \ 10$	$0 \ 10 \ 20 \ 30 \ 30 \ 30 \ 30 \ 20 \ 10$	$0 \ 10 \ 20 \ 30 \ 30 \ 30 \ 30 \ 20 \ 10$
$0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$	$0 \ 20 \ 40 \ 60 \ 60 \ 60 \ 60 \ 40 \ 20$	$0 \ 20 \ 40 \ 60 \ 60 \ 60 \ 60 \ 40 \ 20$	$0 \ 20 \ 40 \ 60 \ 60 \ 60 \ 60 \ 40 \ 20$
$0 \ 0 \ 0 \ 90 \ 90 \ 90 \ 90 \ 90 \ 0 \ 0 \ 0$	$0 \ 30 \ 50 \ 80 \ 80 \ 90 \ 60 \ 30$	$0 \ 30 \ 50 \ 80 \ 80 \ 90 \ 60 \ 30$	$0 \ 30 \ 50 \ 80 \ 80 \ 90 \ 60 \ 30$
$0 \ 0 \ 0 \ 90 \ 90 \ 90 \ 90 \ 90 \ 0 \ 0 \ 0$	$0 \ 30 \ 50 \ 80 \ 80 \ 90 \ 60 \ 30$	$0 \ 20 \ 30 \ 50 \ 50 \ 60 \ 40 \ 20$	$0 \ 20 \ 30 \ 50 \ 50 \ 60 \ 40 \ 20$
$0 \ 0 \ 0 \ 90 \ 0 \ 90 \ 90 \ 90 \ 90 \ 0 \ 0$	$0 \ 10 \ 20 \ 30 \ 30 \ 30 \ 30 \ 20 \ 10$	$0 \ 10 \ 20 \ 30 \ 30 \ 30 \ 30 \ 20 \ 10$	$0 \ 10 \ 20 \ 30 \ 30 \ 30 \ 30 \ 20 \ 10$
$0 \ 0 \ 0 \ 90 \ 90 \ 90 \ 90 \ 90 \ 90 \ 0 \ 0$	$10 \ 10 \ 10 \ 10 \ 0 \ 0 \ 0 \ 0 \ 0$	$10 \ 10 \ 10 \ 10 \ 0 \ 0 \ 0 \ 0 \ 0$	$10 \ 10 \ 10 \ 10 \ 0 \ 0 \ 0 \ 0 \ 0$
$0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$	$\vdots$	$\vdots$	$\vdots$

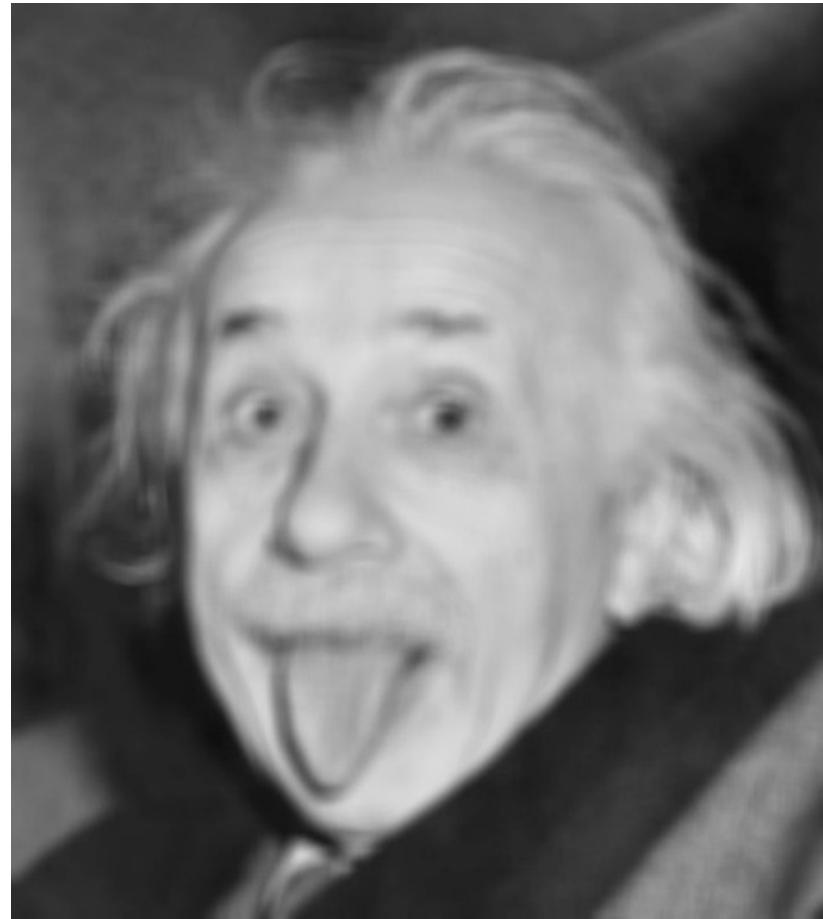
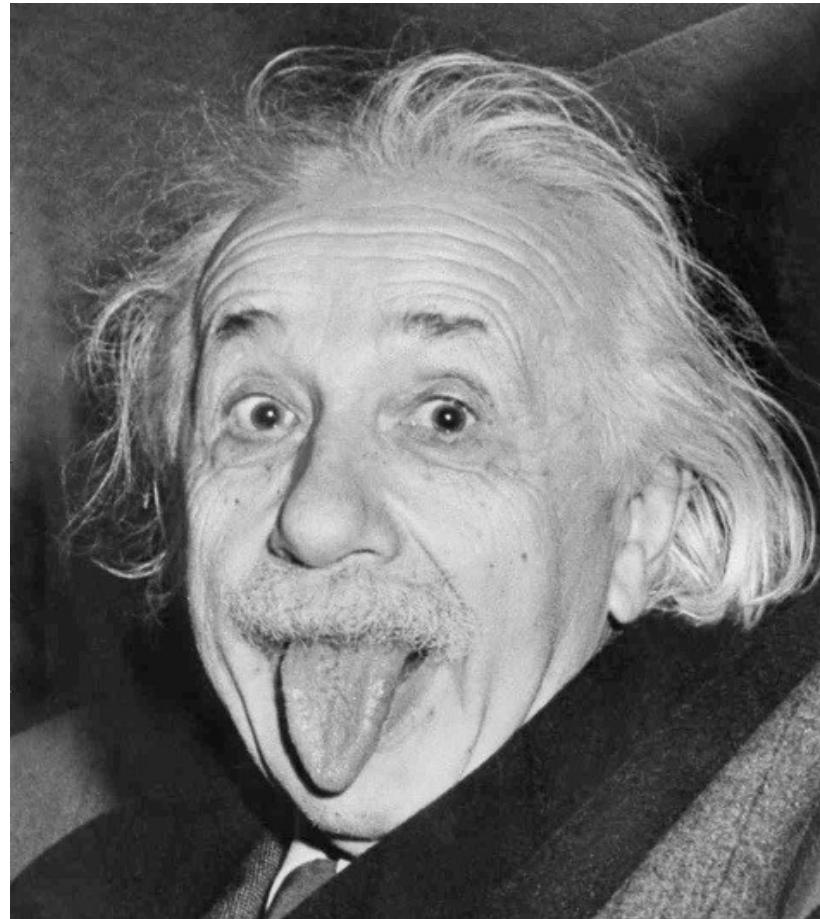
$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

output                      filter                      image (signal)

¿Cuántas filas y columnas agregaría para una matriz 3x3? ¿Y para una 5x5?

Posibilidades: filas y columnas nuevas rellenas con ceros (zero padding) o repitiendo los extremos, o reflejando, etc.

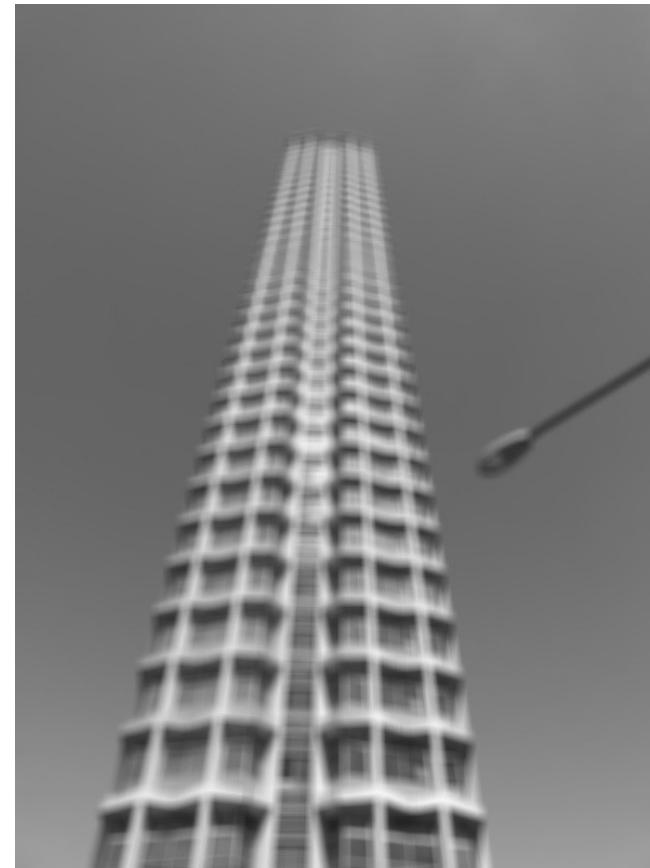
# Ejemplos



# Ejemplos



# Ejemplos

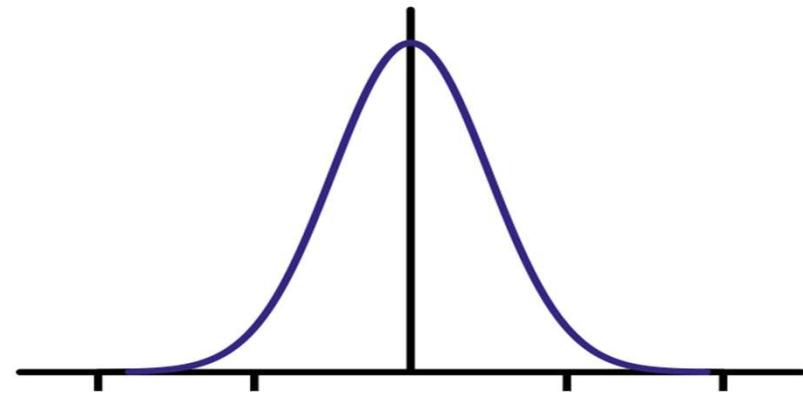


# Filtro Gaussiano

- Los valores del Kernel se toman de la distribución gaussiana.
- Se trata de dar más peso a los pixels vecinos más cercanos.
- Función de Gauss:

$$f(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- El peso desciende de acuerdo a la distancia al pixel central.
- Teóricamente infinita, en la práctica se trunca.
- (2 o 3 sigma)



kernel  $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

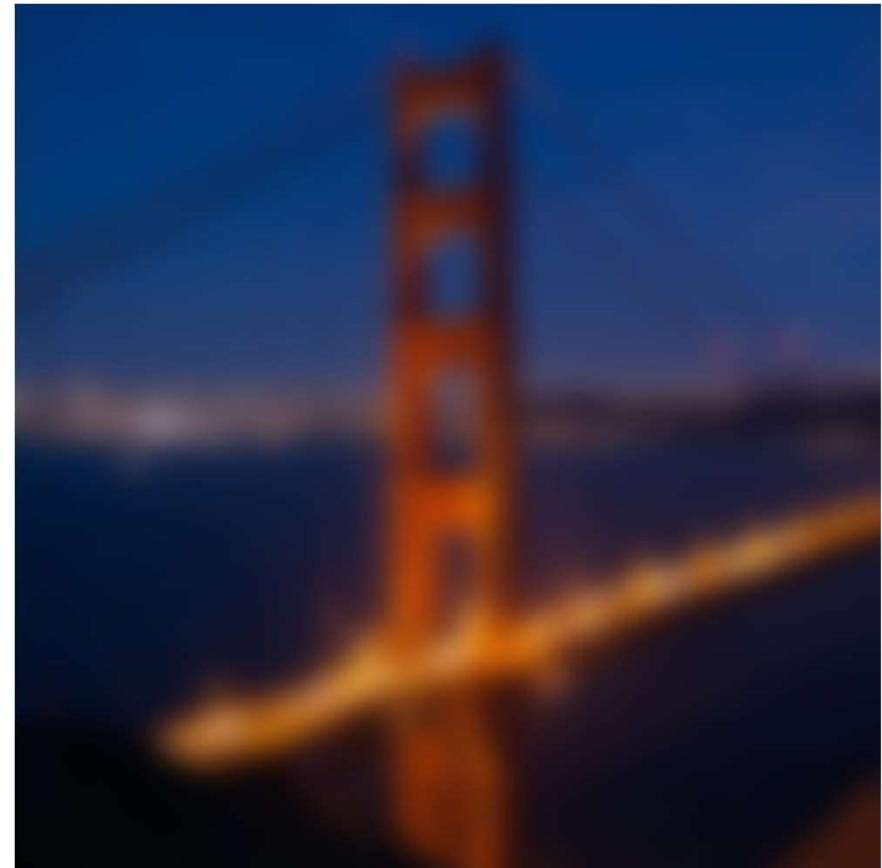
# Filtro Gaussiano

- Máscara gaussiana para 5x5

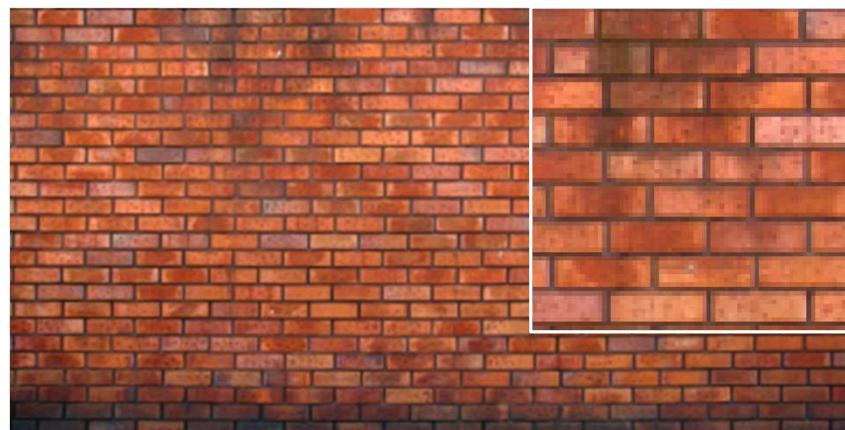
1/256 x

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

# Ejemplo

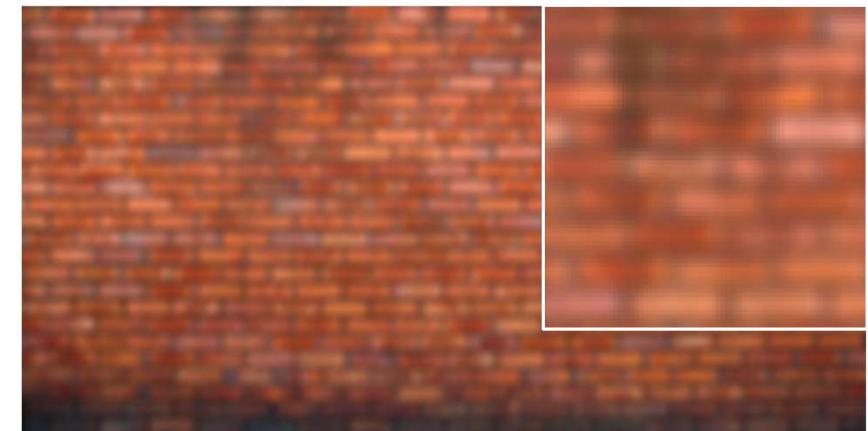


# Comparación entre los efectos del filtro de media y el gaussiano

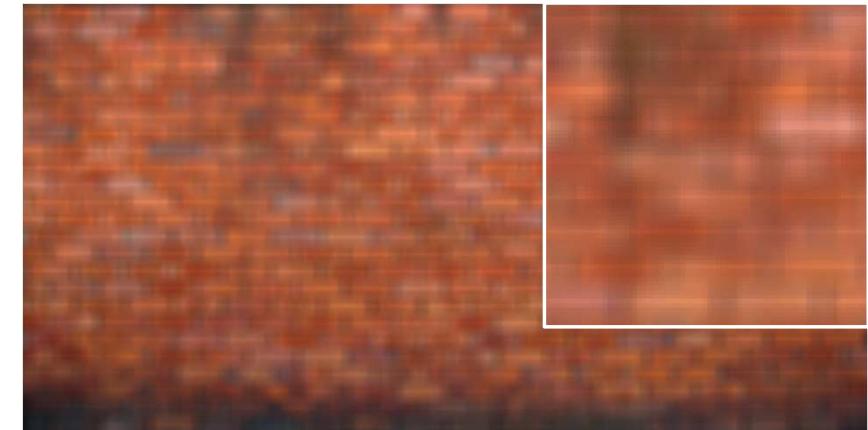


original

Cuál le parece mejor?



7x7 Median

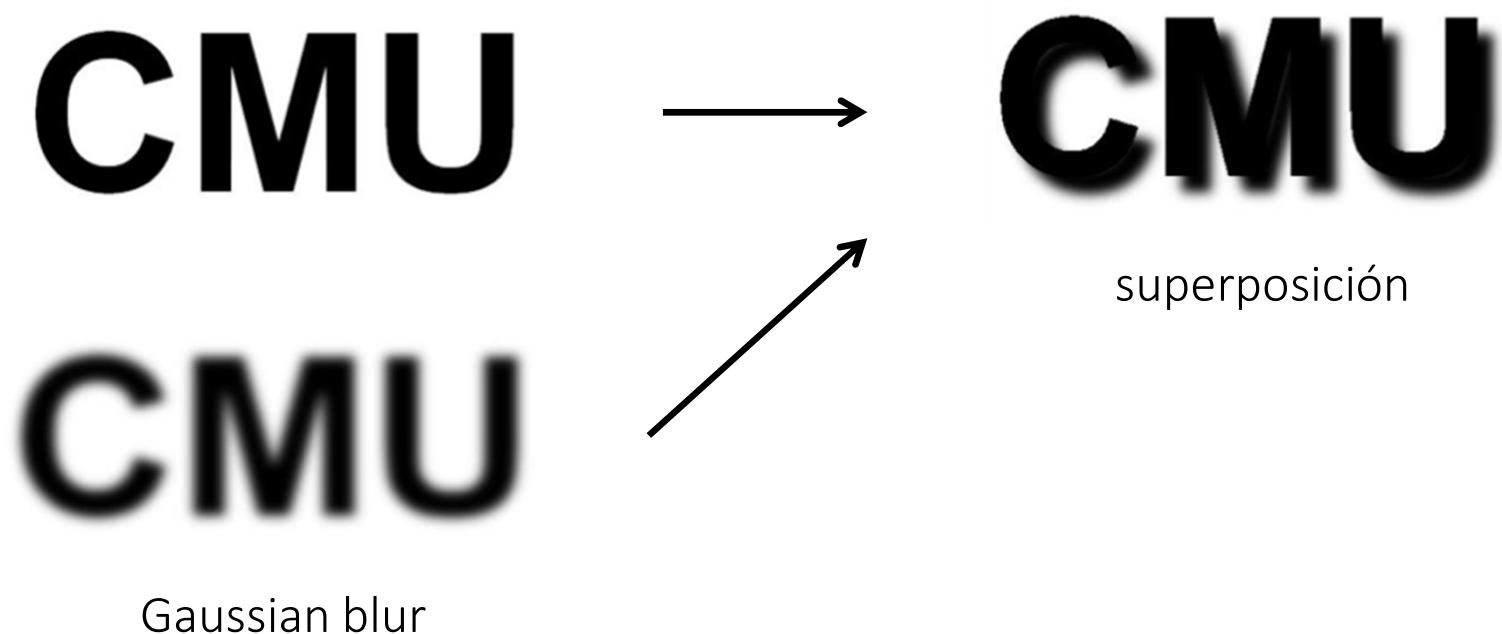


- - -

Cómo lograría este efecto?

**CMU** → **CMU**

Cómo lograría este efecto?



# Otros filtros

Entrada



Filtro

0	0	0
0	1	0
0	0	0

Salida

?

# Otros filtros

Entrada



filtro

0	0	0
0	1	0
0	0	0

salida



Sin  
cambios

# Otros filtros

entrada



filtro

0	0	0
0	1	0
0	0	0

salida



Sin  
cambios

entrada



filtro

0	0	0
0	0	1
0	0	0

salida?

?

# Otros filtros

entrada



filtro

0	0	0
0	1	0
0	0	0

salida



unchanged

entrada



filtro

0	0	0
0	0	1
0	0	0

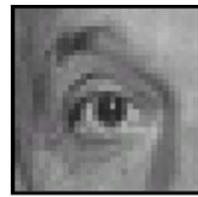
salida



Desplazado a la  
izquierda una  
unidad

# Otros filtros

entrada



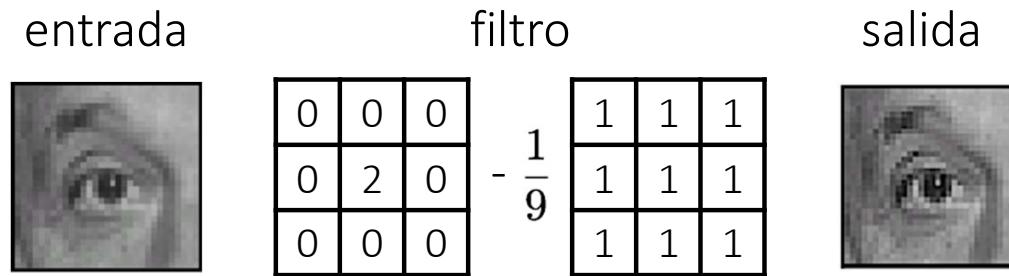
filtro

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

salida

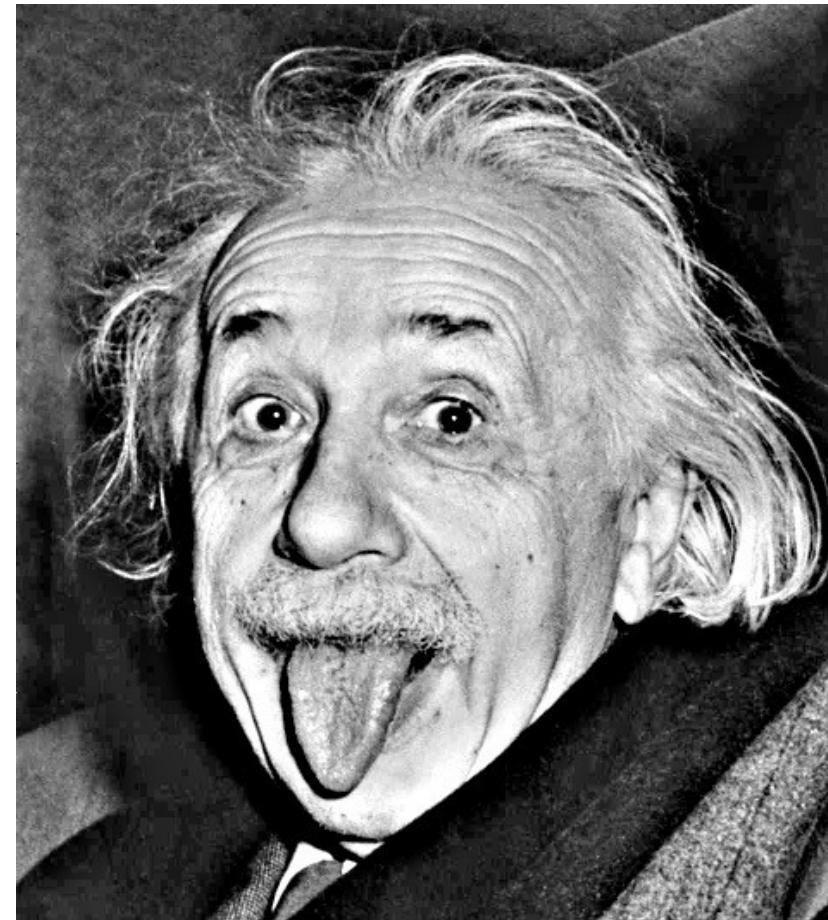
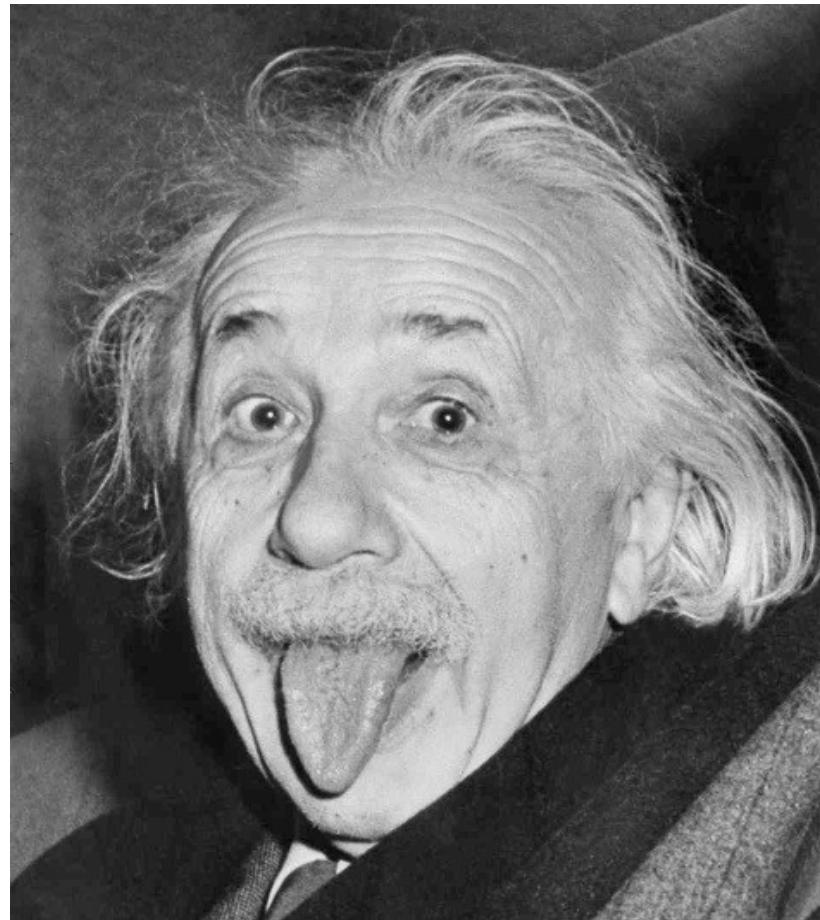
?

# Otros filtros



- No hace nada en áreas planas
- Aumenta la intensidad de los picos (afilamiento)

# Ejemplos



# Ejemplos



# Ejemplos



# Filtros de Orden

**De mediana**

**De moda**

**De máximo**

**De mínimo**

## **Discusión General:**

Ordenan los valores de una vecindad y toman de ellos: el valor central, el máximo o el mínimo, respectivamente.

Son de resultado impredecible, en especial el máximo y el mínimo. Estos últimos se pueden mejorar tomando no el mínimo sino el elemento que le sigue, lo mismo con el máximo.

Todos estos filtros, incluido el de media, pierden información.

***Filtros de media:*** sirven para atemperar el ruido gaussiano.

***Filtros de orden:*** sirven para atemperar el ruido sal y pimienta.

Bordes

# Detectar Bordes

Los bordes son discontinuidades.

Matemáticamente se utilizan derivadas, ya que toman valores altos en las discontinuidades.

En imágenes discretas (en señales discretas) se toman diferencias (diferencias finitas).

# Diferencias Finitas

Derivada:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

Acá usamos:

$$f'(x) = \frac{f(x + 1) - f(x - 1)}{2}$$

(fijamos  $h=2$ ,  $x$  es el punto central)

-1	0	1
----	---	---

1	0	-1
---	---	----

# Filtro de Sobel

$$\begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline \end{array}$$

Deriva

Filtro de Sobel

Blurring

Le da más peso a los elementos más cercanos al píxel central

# Filtro de Sobel

Filtro de Sobel Horizontal:

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

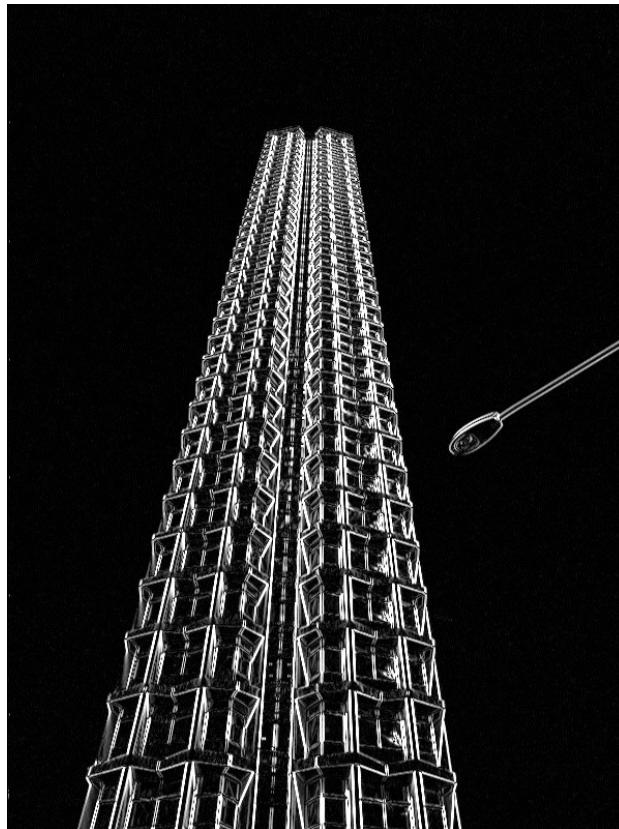
Filtro de Sobel Vertical:

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

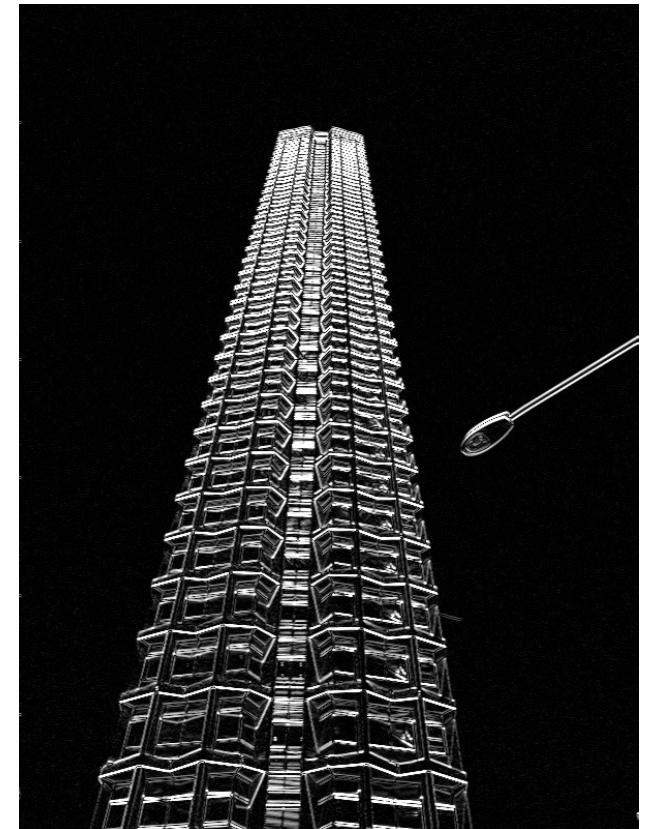
# Ejemplo filtro de Sobel



original

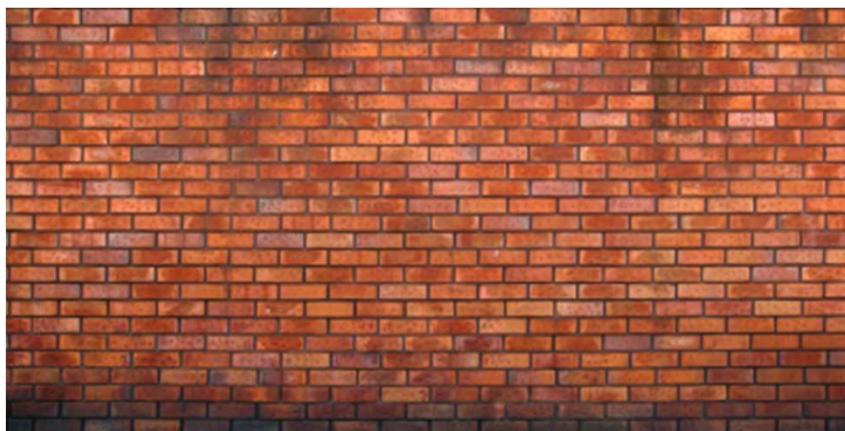


Filtro de Sobel Horizontal



Filtro de Sobel Vertical

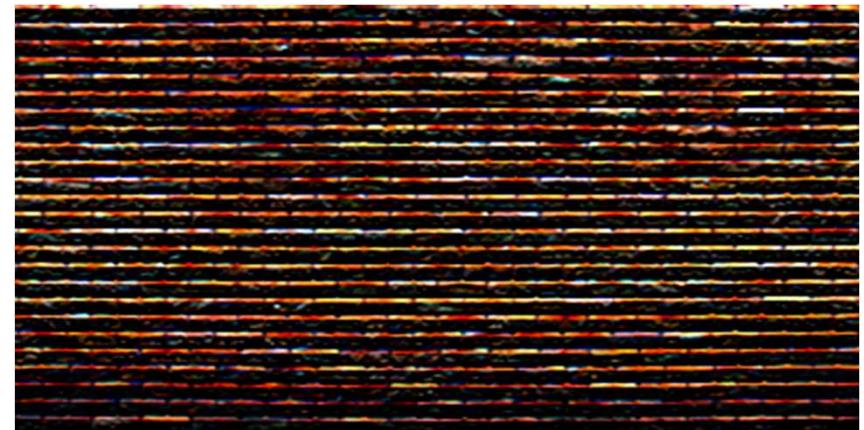
# Ejemplo Filtro Sobel



original



Sobel Horizontal



Sobel Vertical

# Filtros de Borde

Sobel

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

## Tamaño de la máscara:

A mayor tamaño, menos sensibilidad al ruido,

menor precisión en la ubicación

A mayor tamaño, más complejidad computacional. Se usan de 3x3

# Cálculo del gradiente (mayor variación)

- ## 1. Elegimos usar el kernel de Sobel

$$S_x = \begin{array}{|c|c|c|} \hline & 1 & 0 & -1 \\ \hline & 2 & 0 & -2 \\ \hline & 1 & 0 & -1 \\ \hline \end{array}$$

$$S_y = \begin{array}{|c|c|c|} \hline & 1 & 2 & 1 \\ \hline 0 & 0 & 0 & 0 \\ \hline -1 & -2 & -1 & \\ \hline \end{array}$$

2. Aplicamos la máscara a la imagen, como se vio.

$$\frac{\partial f}{\partial x} = S_x \otimes f$$

$$\frac{\partial f}{\partial y} = S_y \otimes f$$

3. Calculamos el módulo y la dirección del gradiente

$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

gradient

# Cálculo del gradiente (mayor variación)

$$|G| = |G_x| + |G_y|$$

Se puede aproximar el módulo con la suma de valores absolutos porque:

No interesa el valor en sí, sino en forma relativa.

Tiene menor tiempo de cómputo que calcular el módulo.

Se tiene en cuenta un umbral.

$$g(x, y) = \begin{cases} 1 & \text{si } G(f(x, y)) > T \\ 0 & \text{si } G(f(x, y)) \leq T \end{cases}$$

## Ejemplo de Cálculo

$$\begin{bmatrix} 0 & 8 & 8 & 8 & 8 & 8 \\ 1 & \textcolor{red}{8} & 9 & 7 & 6 & 8 \\ 2 & 3 & 4 & 8 & 8 & 8 \\ 2 & 2 & 2 & 8 & 8 & 8 \\ 2 & 2 & 2 & 2 & 8 & 8 \end{bmatrix}$$

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

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

Queremos calcular el gradiente en el punto marcado.

Si aplicamos los operadores de Sobel obtenemos:

$$G_c = 8 + (18 - 2) + (4 - 2) = 26$$

$$G_f = (0 + 2) + (-16 + 6) + (-8 + 4) = -12$$

Si el umbral es  $T = 30$

$$|G| = |G_f| + |G_c| = 38 > 30 \text{ hay borde}$$

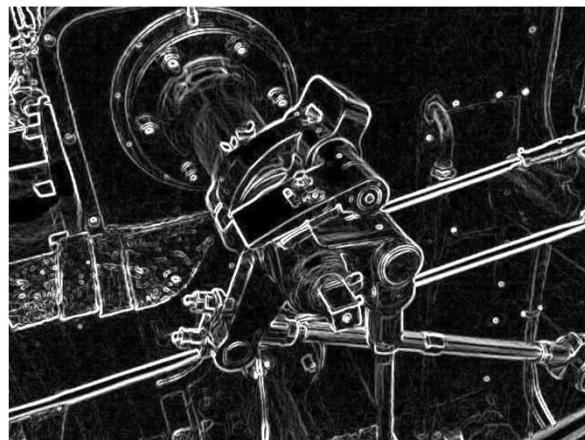
Si el umbral  $T$  fuera mayor que 38 no sería considerado borde.

# Ejemplo

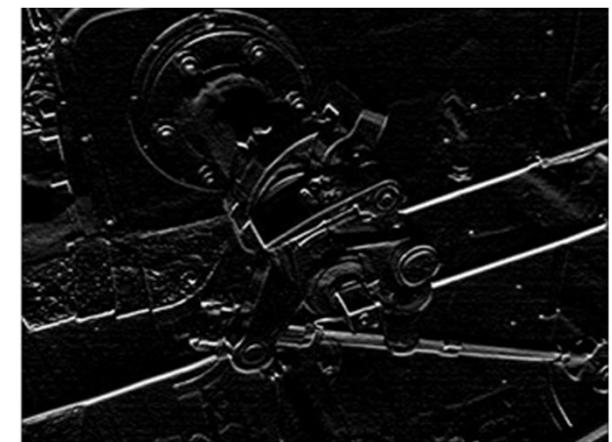
original



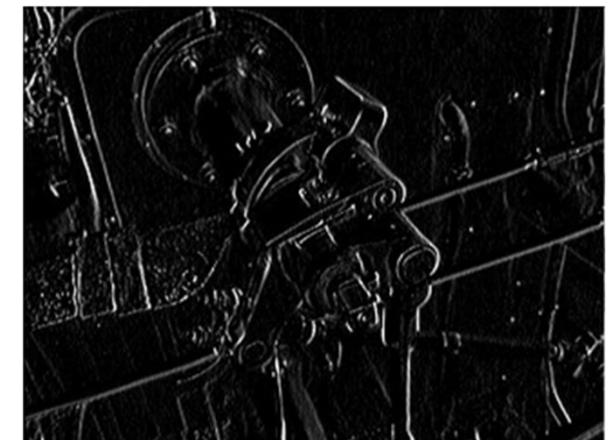
Amplitud del  
gradiente



vertical



horizontal



La operación derivación primera es muy sensible al ruido. Conviene aplicar primero una operación de blurring.

# Filtros Basados en la Derivada Segunda

Las **máscaras basadas en el gradiente** son adecuadas para **transiciones de niveles de gris bruscas** (función tipo escalón o paso).

Las máscaras basadas en el gradiente son **poco sensibles a cambios graduales en los niveles de gris (funciones tipo rampa)**.

Para estas funciones (cambios graduales en los niveles de gris) conviene usar operadores basados en las derivadas de segundo orden, como el **operador laplaciano**.

En el caso de imágenes digitales (discretas) el operador se obtiene aplicando consecutivamente los operadores gradiente en dirección x e y.

# Filtro de Laplace

- Matemáticamente, se basa en la derivada segunda.
- Lo llevamos a diferencias finitas.

Diferencia finita  
De primer orden

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

→ 1D derivative filter

1	0	-1
---	---	----

Diferencia finita  
De segundo orden

$$f''(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

→ Laplace filter  
?

1	-2	1
---	----	---

Se puede combinar también el laplaciano con el gaussiano: FILTRO LoG

# Operador Laplaciano

Trabaja considerando la derivada segunda.

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

Laplaciano de una función  
bidimensional

La aproximación para una función discreta es:

$$\nabla^2 f(m, n) = D_x(D_x(f(m, n))) + D_y(D_y(f(m, n)))$$

$$\nabla^2 f(m, n) = D_x(f(m+1, n) - f(m, n)) + D_y(f(m, n+1) - f(m, n))$$

$$\begin{aligned}\nabla^2 f(m, n) = & [(f(m+1, n) - f(m, n)) - (f(m, n) - f(m-1, n))] + \\ & [(f(m, n+1) - f(m, n)) - (f(m, n) - f(m, n-1))]\end{aligned}$$

$$\nabla^2 f(m, n) = f(m+1, n) + f(m, n+1) + f(m-1, n) + f(m, n-1) - 4f(m, n)$$

Resulta la siguiente matriz de convolución:

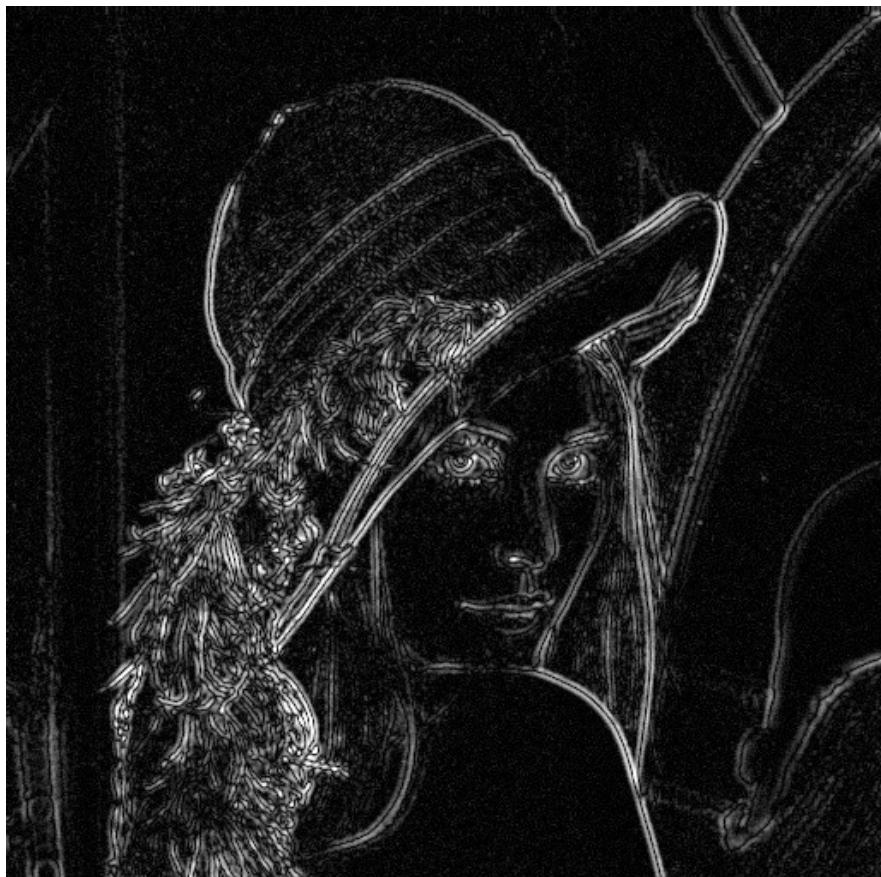
$$\nabla^2 f = f * M$$

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

O sea que se puede interpretar como un filtro de primer orden.

Se puede combinar también el laplaciano con el gaussiano: FILTRO LoG

# Ejemplos

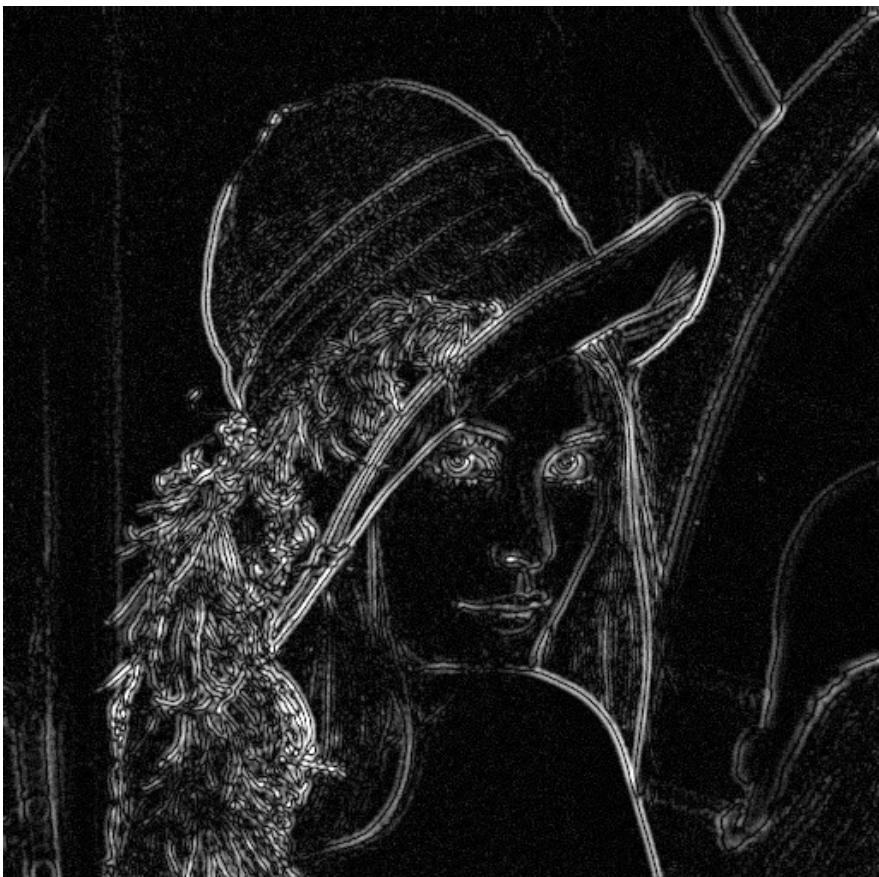


Laplaciano de Gaussiano



Filtro de Laplace

# Laplacian de Gaussian vs Derivada de Gaussiano

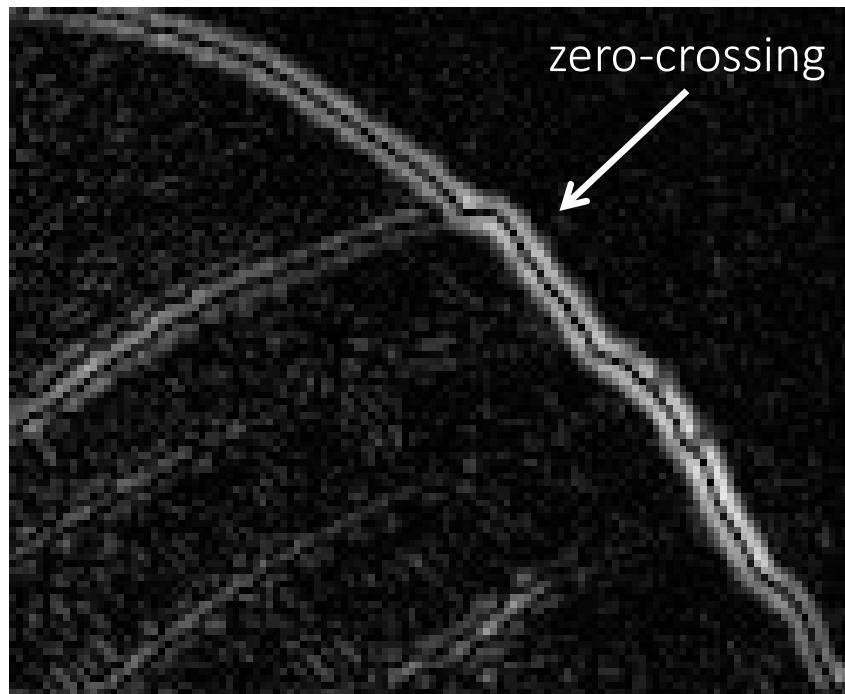


Laplacian de Gaussiano

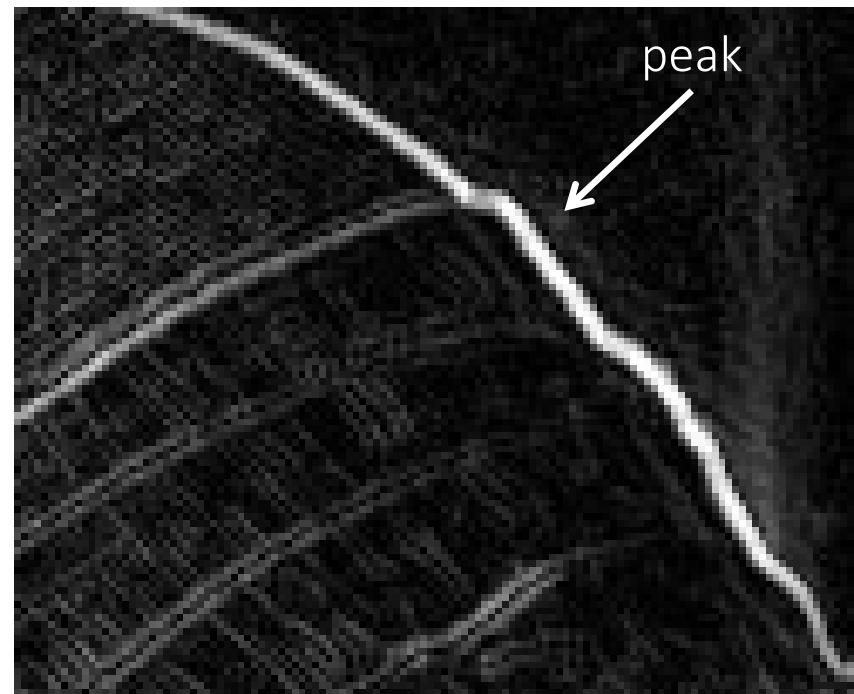


Derivada de gaussiano

# Laplacian of Gaussian vs Derivative of Gaussian



Laplacian of Gaussian filtering



Derivative of Gaussian filtering

# Filtro de Canny

Es un muy buen filtro, pero más complejo

## Etapas

- Se suaviza la imagen mediante un **filtro gaussiano**, con un valor dado de sigma.
  - Estima **el gradiente** (magnitud y dirección) para cada píxel de la imagen (filtro Sobel). Los gradientes son perpendiculares a los bordes.
  - En esta matriz, debe suprimir los puntos que no son máximos. Para esto compara cada punto con sus vecinos y umbraliza.
- Supresión no maximal** Asigna un umbral para eliminar los bordes espurios. Es lo más delicado porque si se asigna mal aparece ruido.
- Trabaja con **dos umbrales**: uno inferior y otro superior. Por debajo del inferior rechaza y por encima del superior acepta (bordes fuertes). El resto corresponde a los llamados bordes débiles (entre T1 y T2). Puede tomar  $T_2 > 2T_1$ . El resultado es una imagen binaria.
  - Enlace de bordes**: a los bordes débiles los acomoda: si están **conectados** a los píxeles cuyo valor es superior al umbral superior se aceptan. Se consideran así, si están dentro de un entorno  $3 \times 3$  del borde fuerte. Así se eliminan los falsos bordes.

# Filtro de Canny



# Referencias

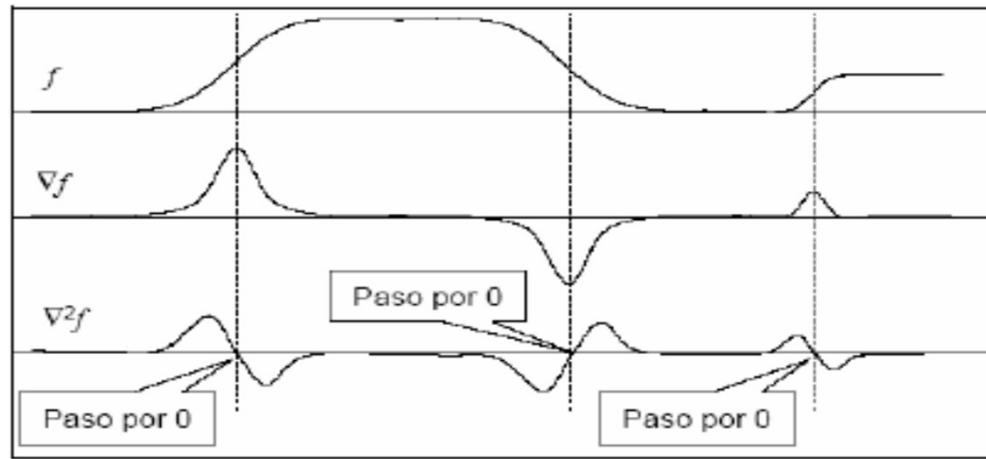
## Bibliografía Básica

- Szeliski Richard, Computer Vision: algorithms and applications. (disponible en Internet), Section 3.2

La mayoría de las filminas se tomaron de:

- Kris Kitani (15-463, Fall 2016).

## Diferencia entre el accionar del Gradiente y del Laplaciano



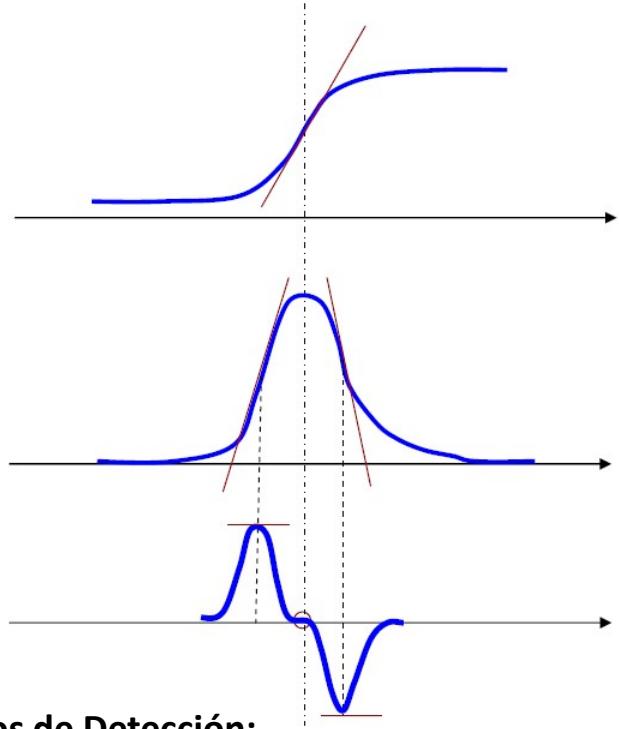
Como el laplaciano trabaja con las derivadas segundas **es más sensible al ruido** que los anteriores.

Una vez aplicada la máscara se tiene en cuenta un umbral.

# Diferencia entre el accionar del Gradiente y del Laplaciano

Función rampa creciente => Derivada primera siempre positiva.

Pero como la función presenta un punto de inflexión, la derivada primera tiene un máximo, y en ese punto la derivada segunda un cero.



## Criterios de Detección:

- 1.Cruce por cero en el caso de la derivada segunda.
- 2.Pico muy alto (mayor que un umbral) en el caso de la derivada primera)

La **derivada segunda** de una función rampa suavizada que puede representar el borde de un objeto, es una función que presentará un máximo y un mínimo, en lugar de ser una función con un máximo único (caso de la derivada primera).

Lo que se busca para determinar los bordes son los **puntos de cruce por cero** en la derivada segunda (imagen filtrada), es decir los cambios de signo, de positivo a negativo y viceversa.