Filtros

Temario

- Conectividad
- Filtros en el dominio espacial
- Filtro de caja
- Filtros gaussiano
- Imágenes híbridas

Conectividad

- **Conectividad**. La forma en que los pixeles están conectados entre sí.
- Tres tipos de conectividad.
- Conectividad de 4 vecinos, N4(p).
- Conectividad de 8 vecinos, N8(p).
- Conectividad mixta, M-connectivity.

Conectividad de 4 vecinos N4(p)

	$p(x, y-1)$	
$p(x-1, y)$	$p(x,y)$	$p(x+1, y)$
	$p(x, y+1)$	

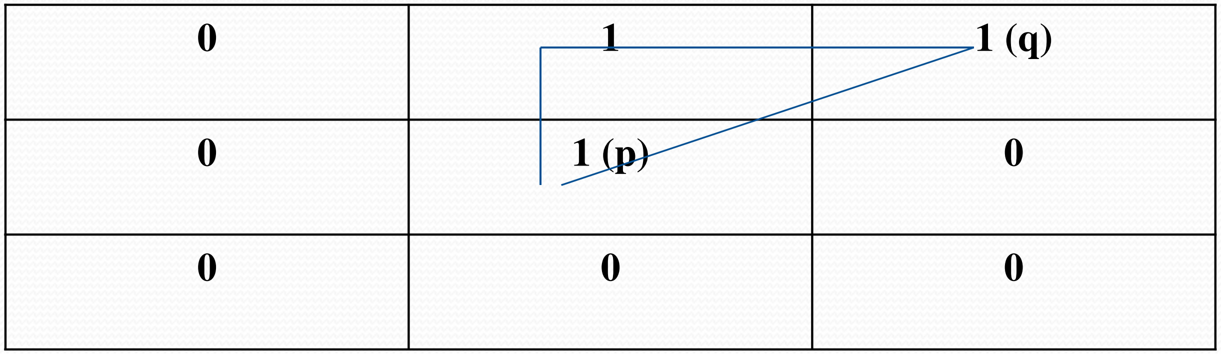
Conectividad de 8 vecinos N8(p)

$p(x - 1, y - 1)$	**$p(x, y - 1)$**	**$p(x + 1, y - 1)$**
$p(x - 1, y)$	**$p(x, y)$**	**$p(x + 1, y)$**
$p(x - 1, y + 1)$	**$p(x, y + 1)$**	**$p(x + 1, y + 1)$**

$$N8(p) = N4(p) + Nd(p)$$

Conectividad mixta (M-connectivity)

- La conectividad N8 tiene una ambigüedad:
- Es posible de que haya más de un camino entre 2 pixeles.
- Ejemplo, hay 2 caminos entre p y q:



0	**1**	**1 (q)**
0	**1 (p)**	**0**
0	**0**	**0**

Conectividad mixta (M-connectivity)

- Definición:
- Un pixel p está conectado con otro pixel q si se cumple alguna de estas 2 condiciones:

1. El pixel q está en N4(p).
2. El pixel q está en Nd(p) y N4(p) ∩ N4(q) = Ø.

Ejemplo 1

- Los pixeles p y q no tienen M-connectivity.

0	**1**	**1 (q)**
0	**1 (p)**	**0**
0	**0**	**0**

1. El pixel q no está en $N4(p)$.
2. El pixel q está en $Nd(p)$, pero $N4(p) \cap N4(q) \neq \emptyset$.

Ejemplo 2

- Los pixeles p y q si tienen M-connectivity.

0	**1**	**1**
0	**1 (p)**	**0**
0	**0**	**1 (q)**

1. El pixel q no está en N4(p).
2. El pixel q está en Nd(p), y N4(p) ∩ N4(q) = ∅.

Distancia entre pixeles

- Dados dos pixeles: p(x1, y1) y q(x2, y2), su distancia se calcula mediante alguna de las siguientes métricas.

1. Distancia de Manhattan (Distancia L1):

$$D1 = |x1 - x2| + |y1 - y2|$$

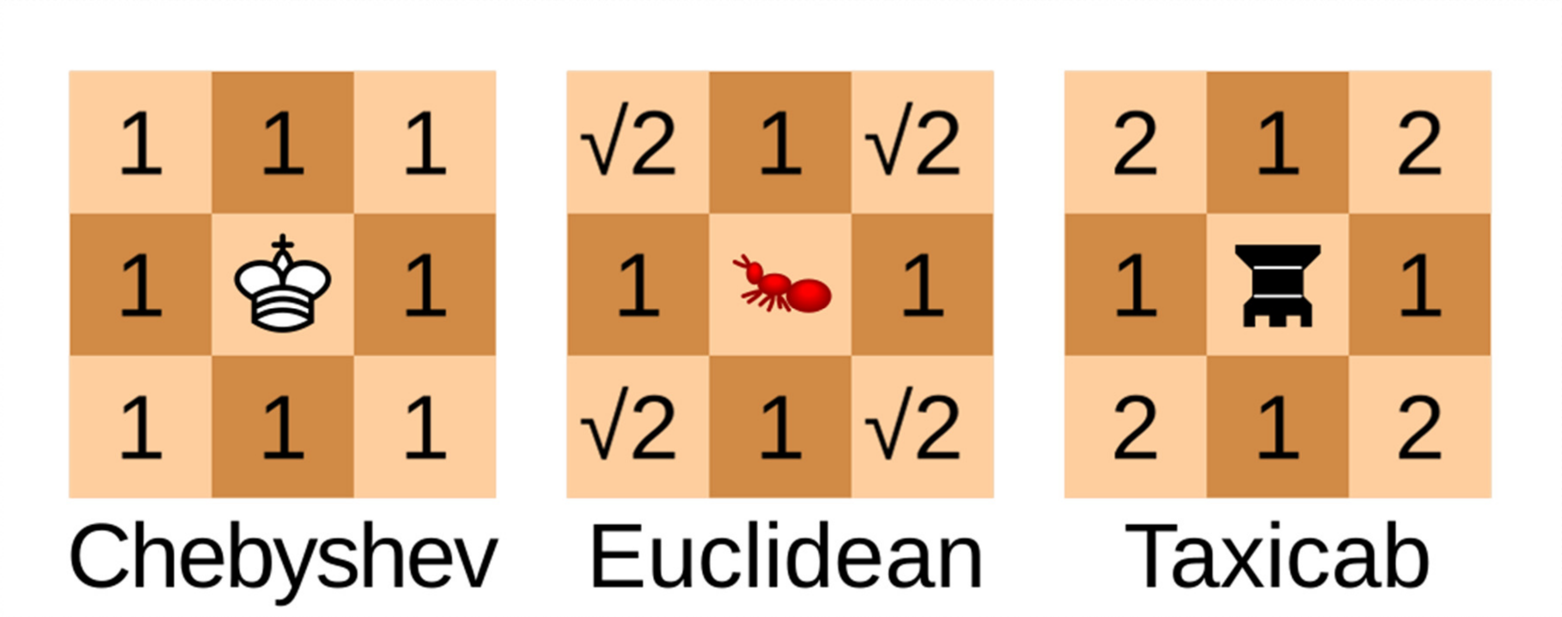
2. Distancia Euclidiana (Distancia L2):

$$D2 = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

3. Distancia de Chebyshev o de ajedrez (Distancia L∞):

$$L\infty = \max(|x1 - x2|, |y1 - y2|)$$

Distancia entre pixeles



1	1	1
1	♔	1
1	1	1

Chebyshev

√2	1	√2
1	🐜	1
√2	1	√2

Euclidean

2	1	2
1	♜	1
2	1	2

Taxicab

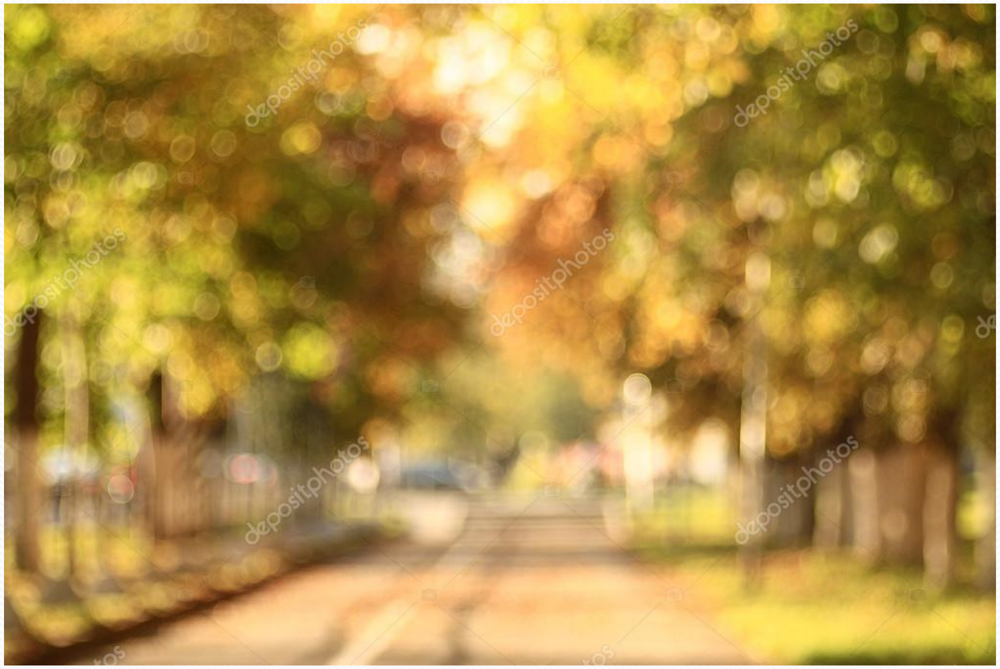
Fuente: By Cmglee - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=140170366

Imagen nítida

- En una imagen nítida (sharp) la intensidad local aumenta o disminuye bruscamente (es decir, la diferencia entre pixeles vecinos es grande).

Imagen borrosa

- En una imagen borrosa (blurred, fuzzy) la función de intensidad local es suave.


depositphotos

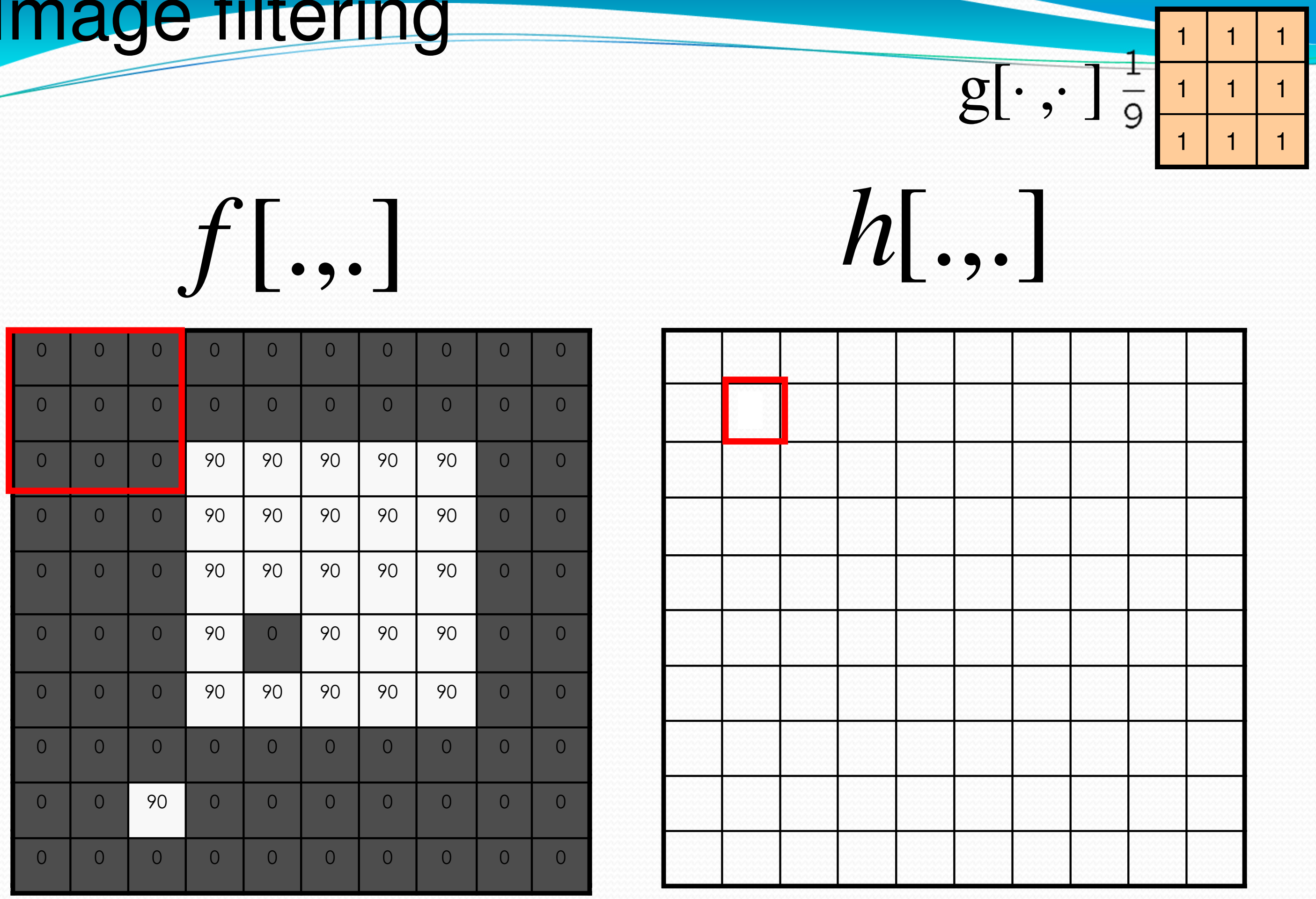
Filtros

- **Filtrado de imágenes**. Para cada pixel, se calcula una función tomando en cuenta sus valor y el valor de sus vecinos.
- Funciones de los filtros:
- Mejorar imágenes.
  - Eliminar ruido, cambiar el tamaño, etc.
- Extraer información de imágenes.
  - Textura, bordes, puntos distintivos, etc.
- Detectar patrones
  - Template matching

Ejemplo: filtro de caja

- Filtro de caja de 3 x 3:
- $H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Image filtering



$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

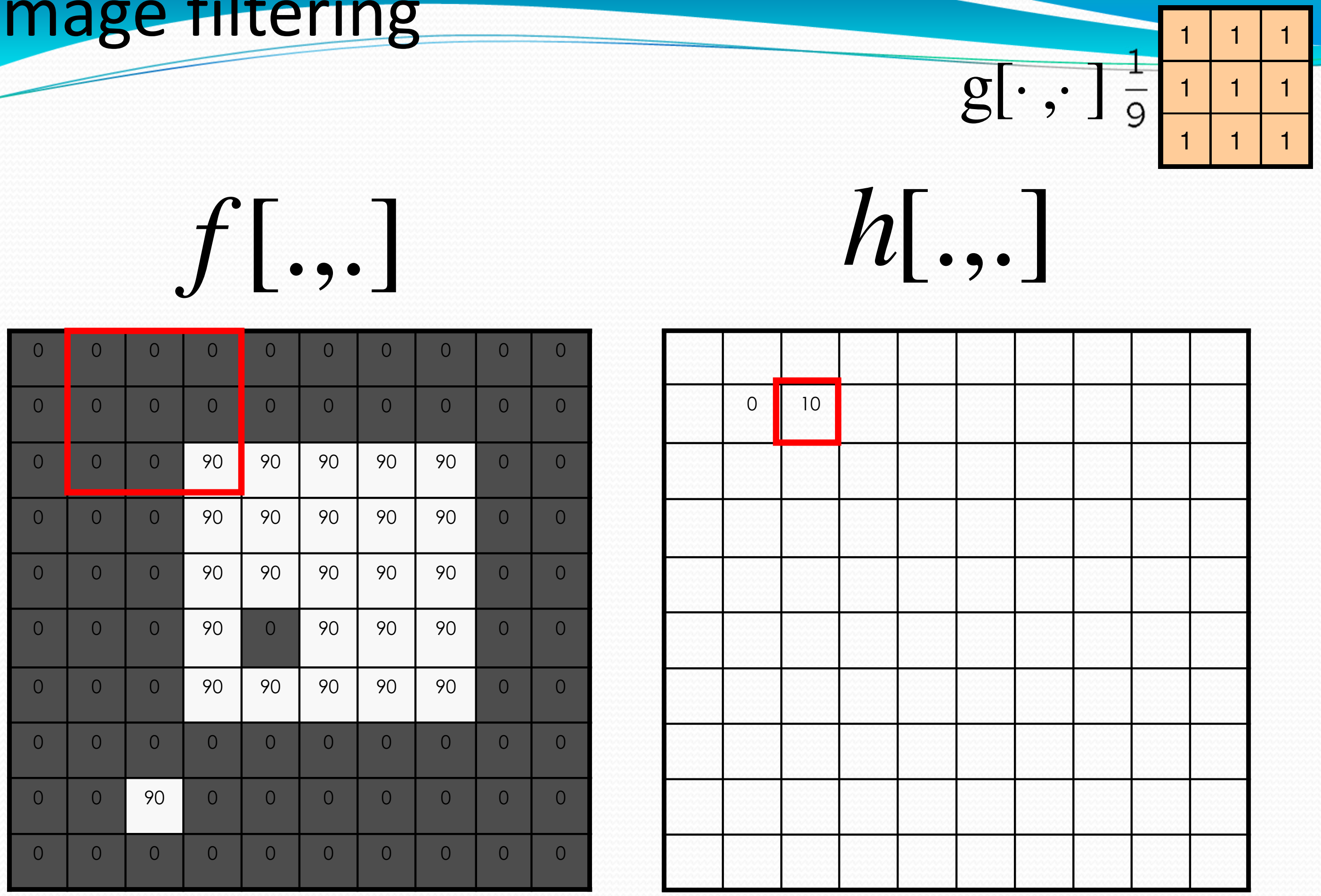
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

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

Credit: S. Seitz

Image filtering



$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

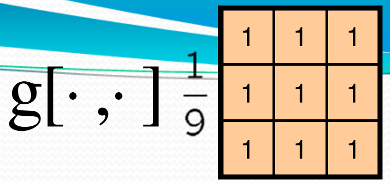
$h[.,.]$

	0	10							

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

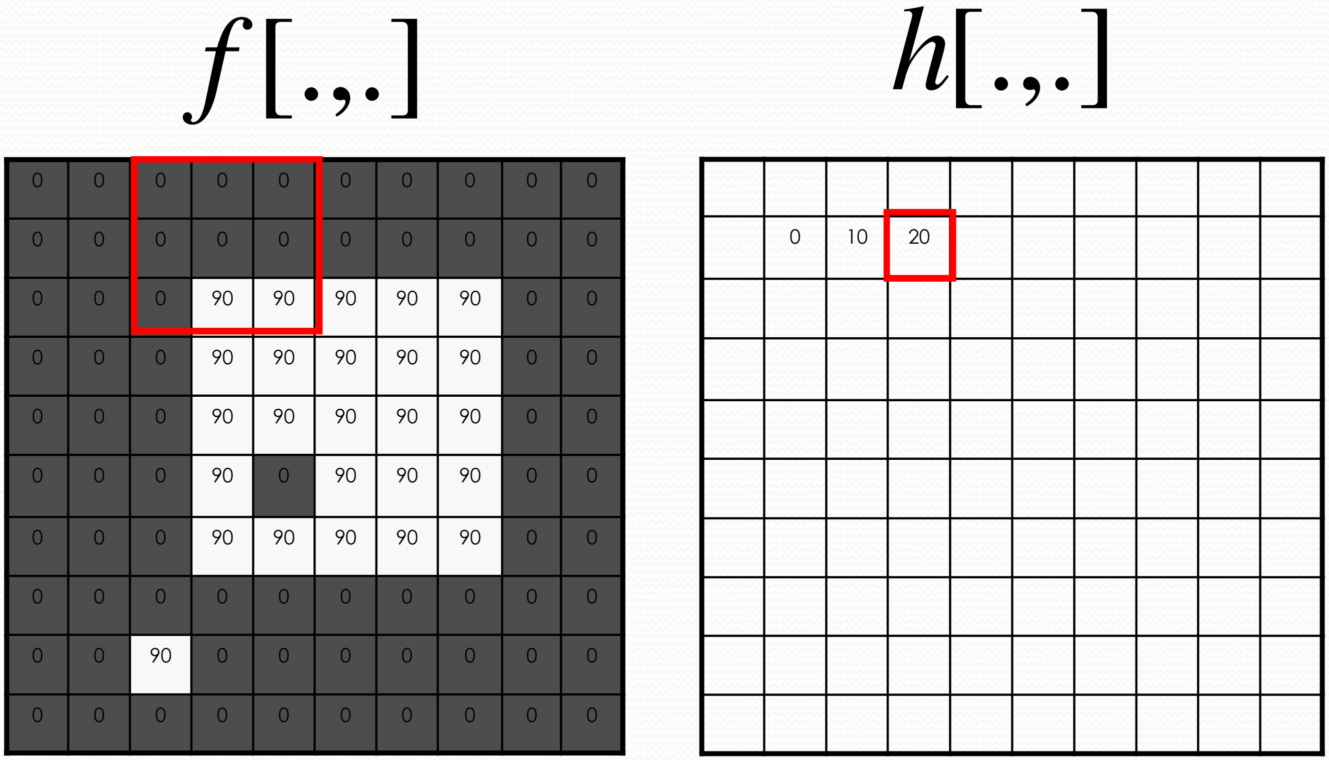
Credit: S. Seitz

Image filtering



$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1



$f[.,.]$

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

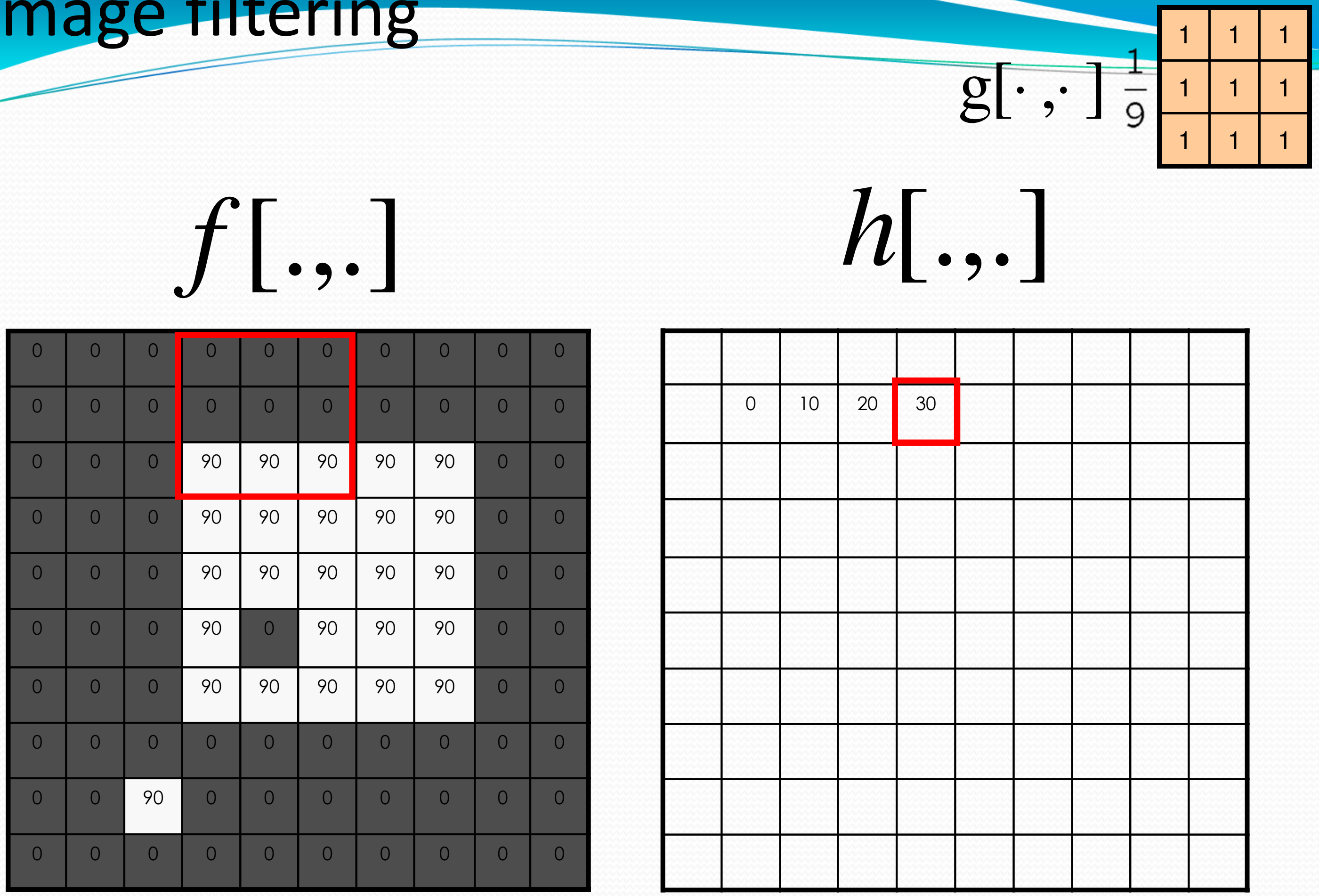
$h[.,.]$

	0	10	20						

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

Credit: S. Seitz

Image filtering



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

$f[.,.]$

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

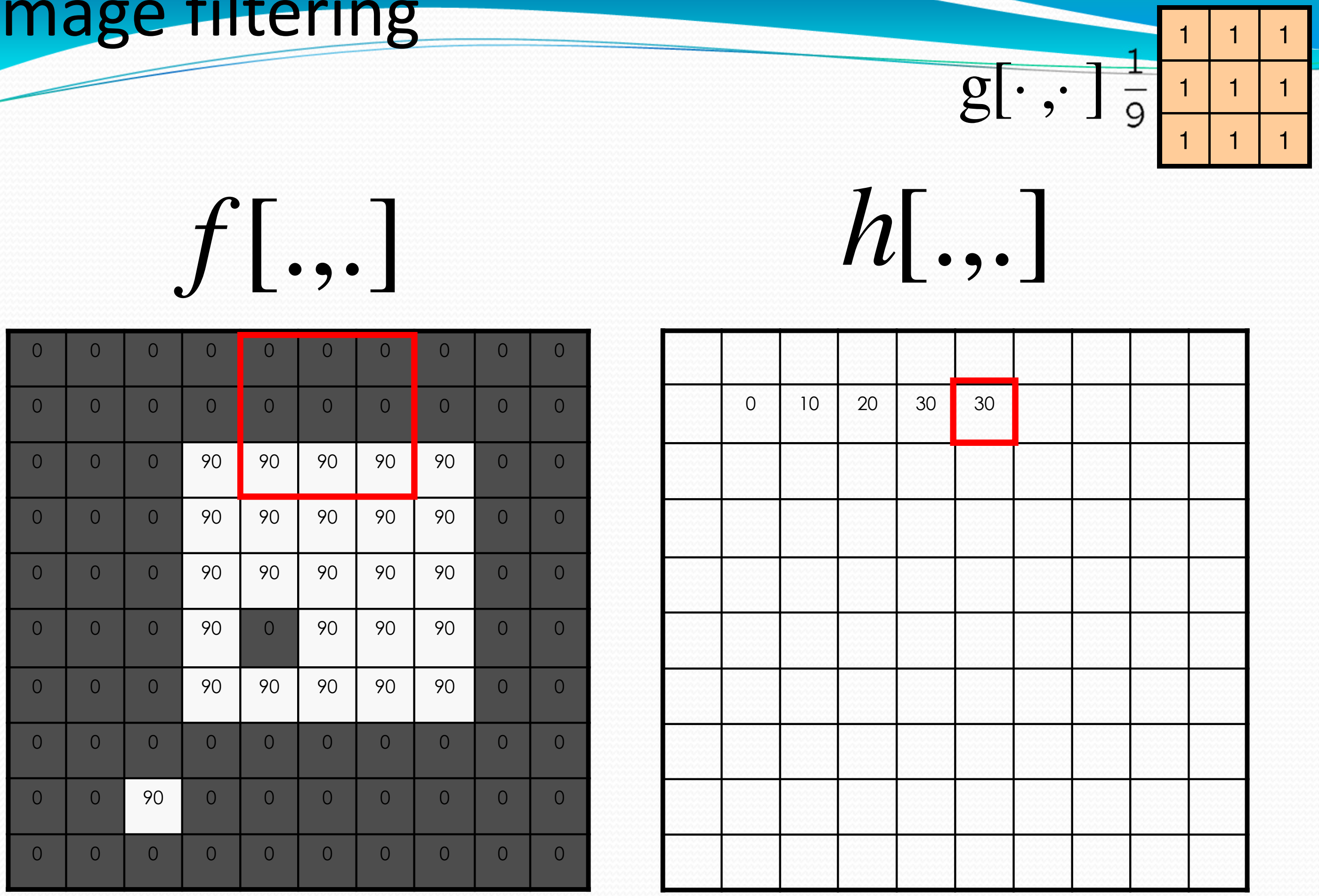
$h[.,.]$

	0	10	20	30					

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

Credit: S. Seitz

Image filtering



$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

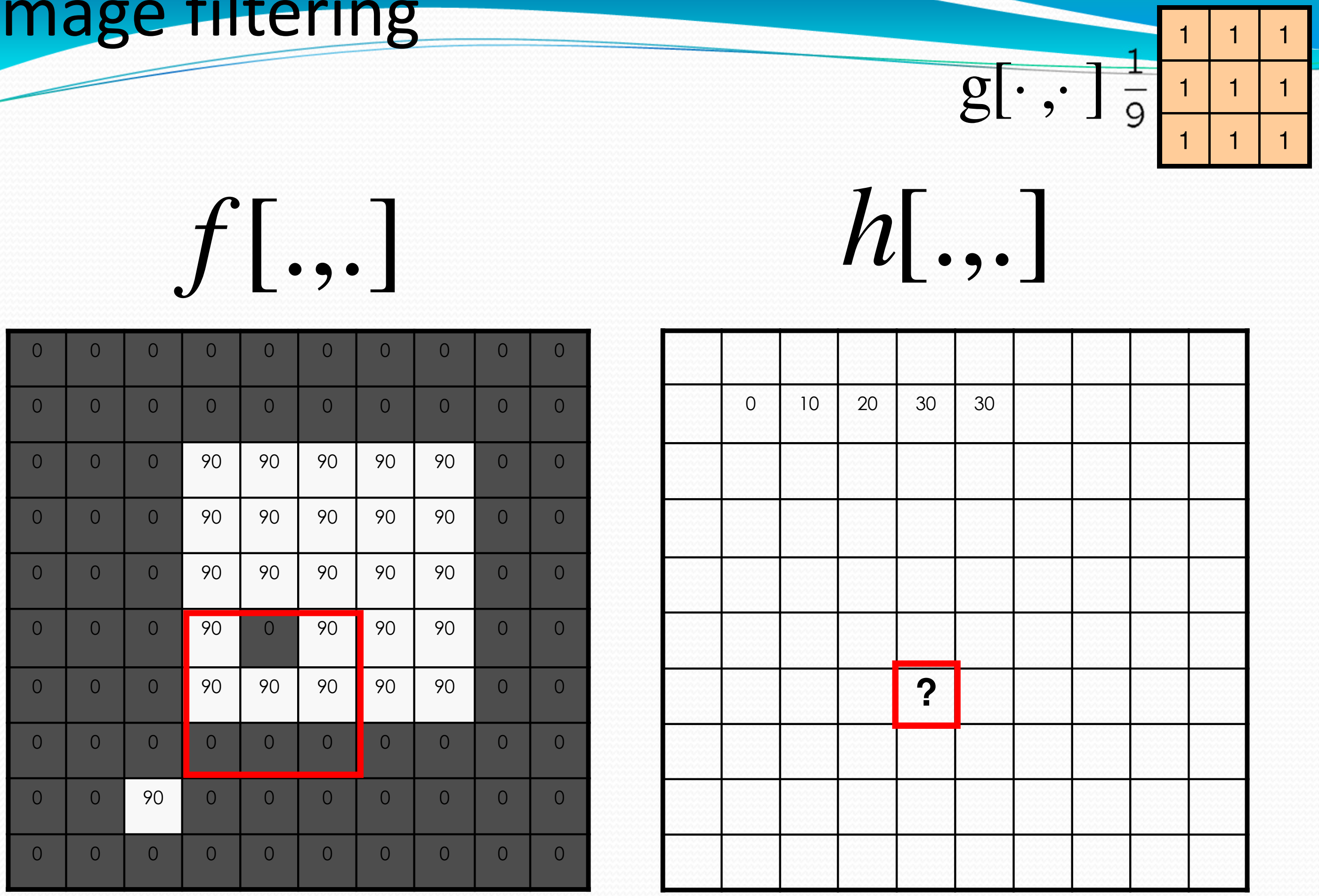
$h[.,.]$

	0	10	20	30	30				

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

Credit: S. Seitz

Image filtering



$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

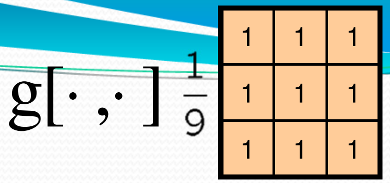
$h[.,.]$

	0	10	20	30	30				
				?					

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

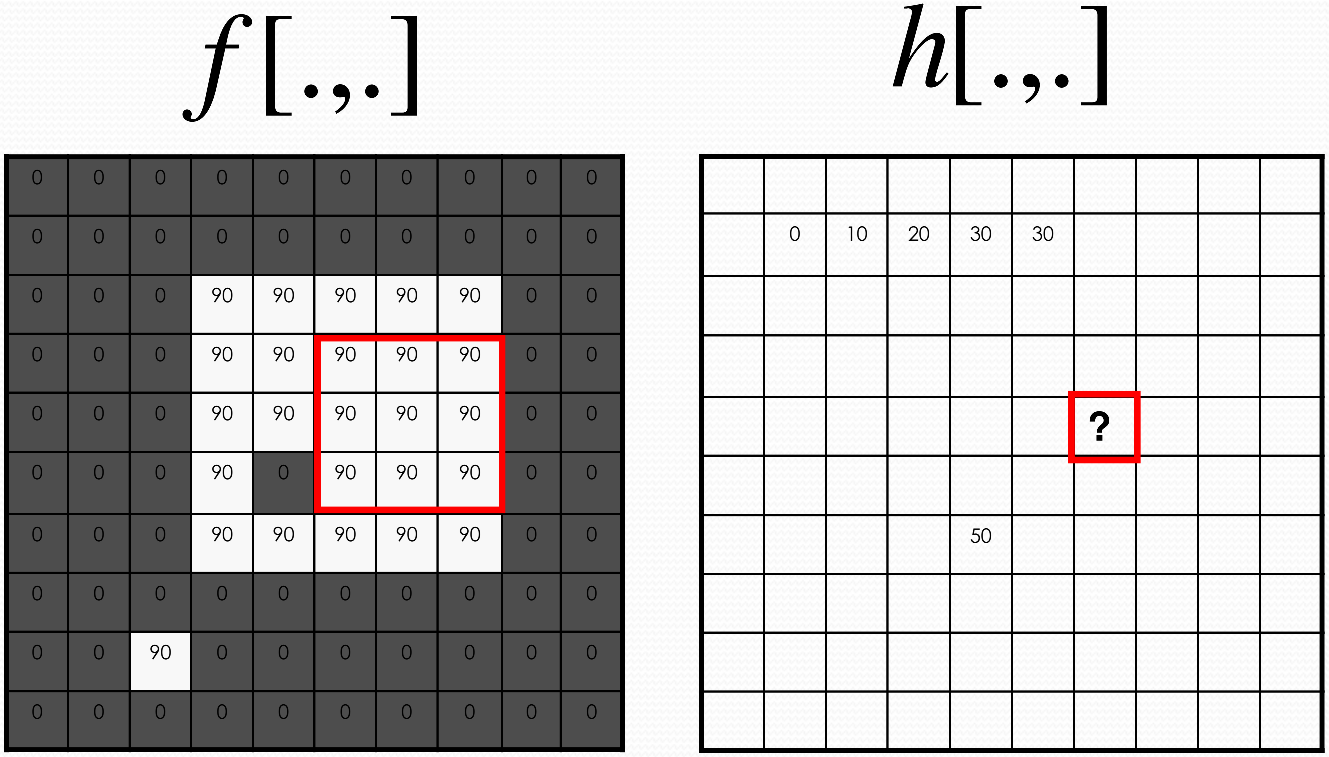
Credit: S. Seitz

Image filtering



$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1



$f[.,.]$

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

$h[.,.]$

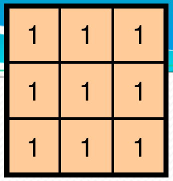
	0	10	20	30	30				
						?			
				50					

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

Credit: S. Seitz

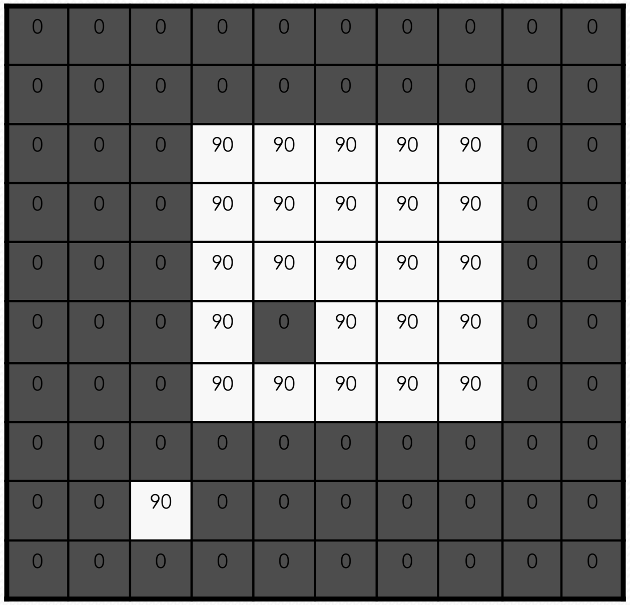
Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$



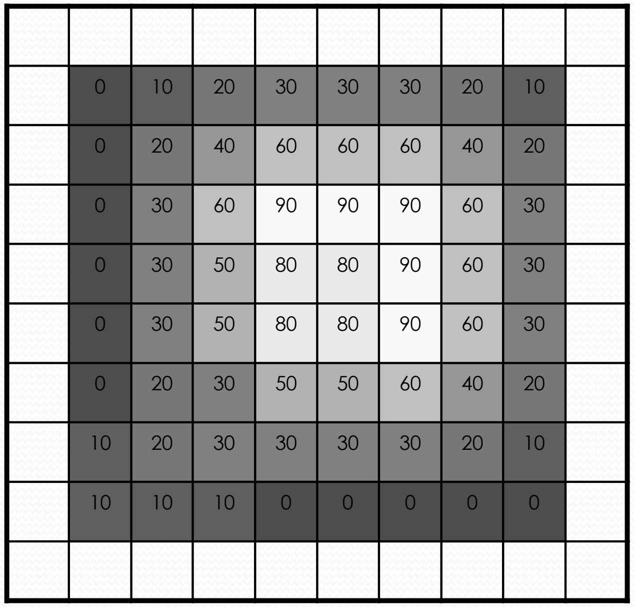
1	1	1
1	1	1
1	1	1

$f[.,.]$



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

$h[.,.]$



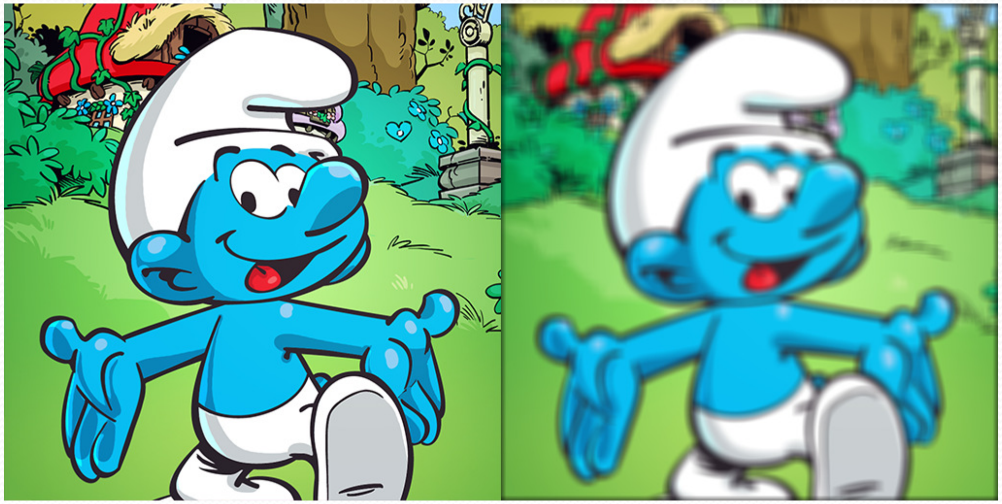
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Credit: S. Seitz

Conclusión

- El filtro de caja reemplaza el valor de cada pixel por el promedio de sus vecinos.
- Suaviza la imagen.

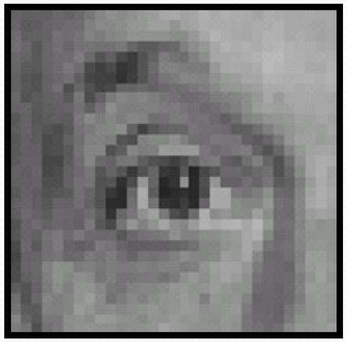


Otro ejemplo

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

Fuente: Derek Hoiem (UIUC)

Practice with linear filters



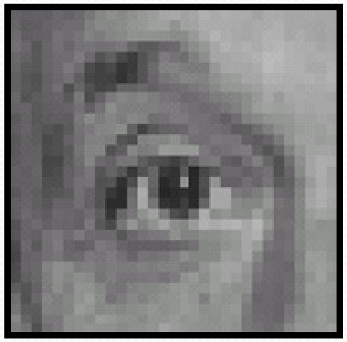
Original

0	0	0
0	1	0
0	0	0

?

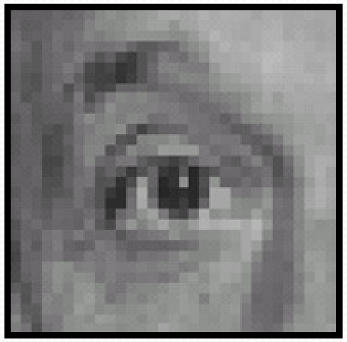
Source: D. Lowe

Practice with linear filters



Original

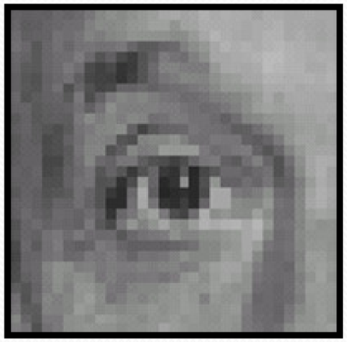
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



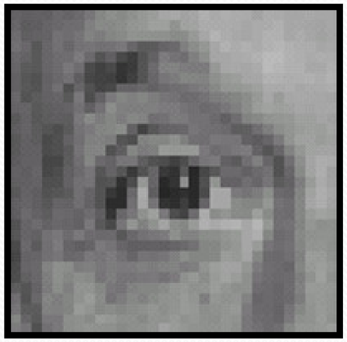
Original

0	0	0
0	0	1
0	0	0

?

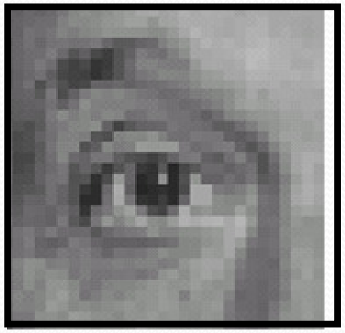
Source: D. Lowe

Practice with linear filters



Original

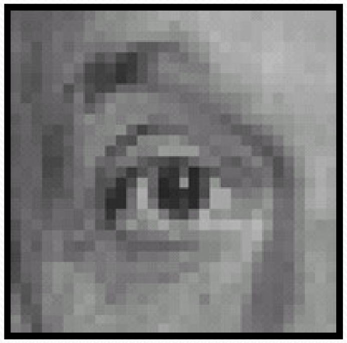
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters



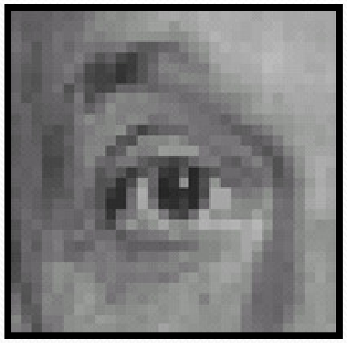
Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad ?$$

(Note that filter sums to 1)

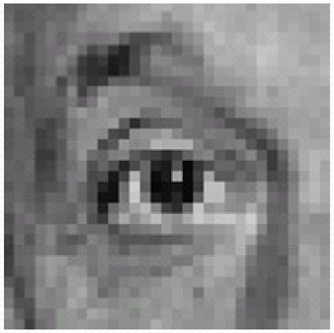
Source: D. Lowe

Practice with linear filters



Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

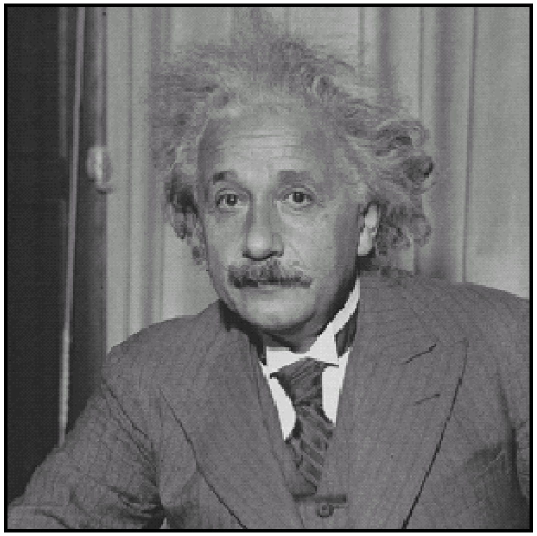


Sharpening filter

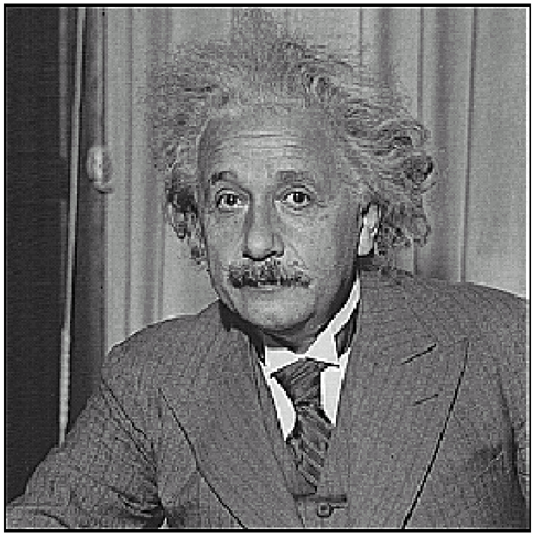
- Accentuates differences with local average

Source: D. Lowe

Sharpening



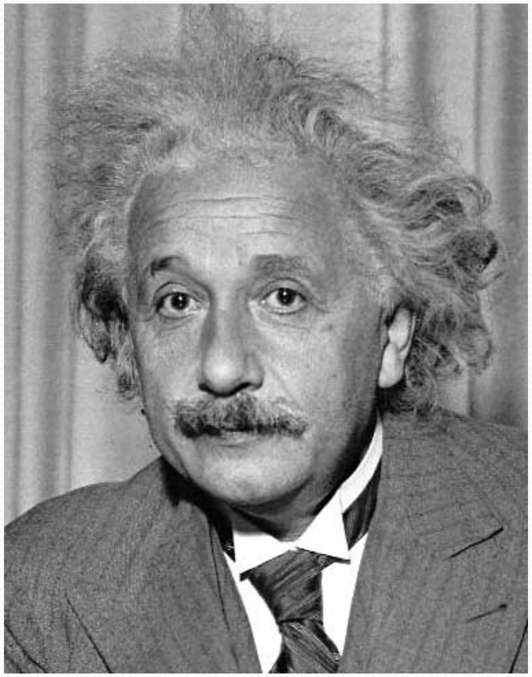
before



after

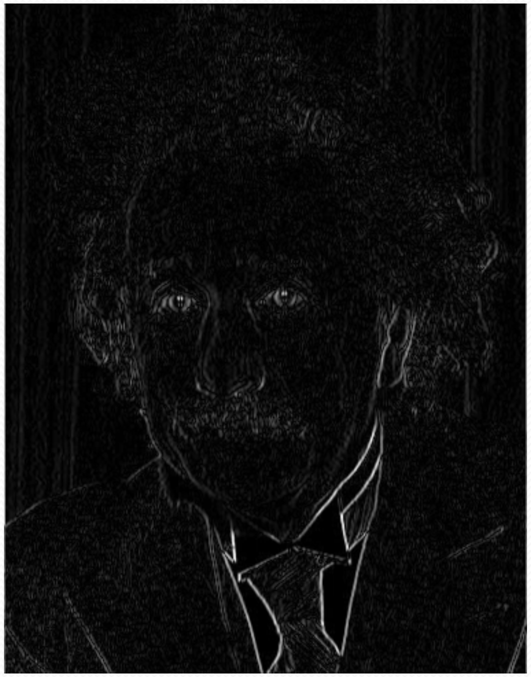
Source: D. Lowe

Other filters



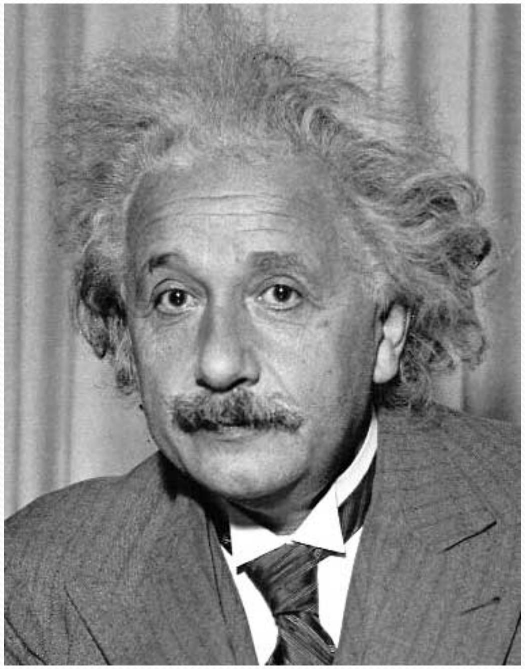
1	0	-1
2	0	-2
1	0	-1

Sobel



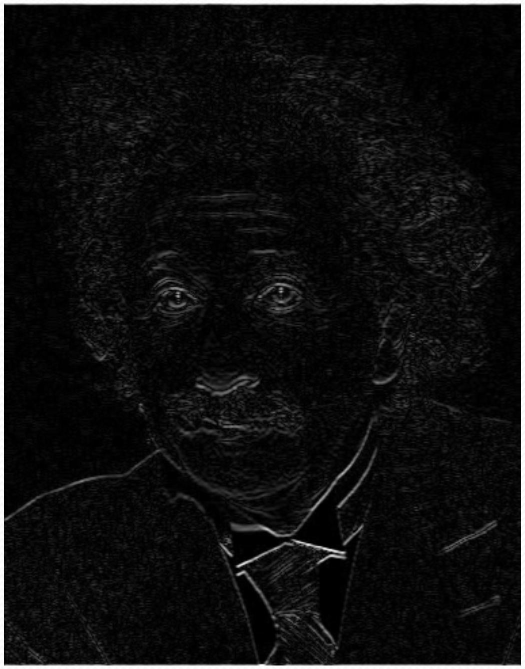
Vertical Edge
(absolute value)

Other filters



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

Sobel



Horizontal Edge
(absolute value)

Aplicación de un filtro

- La aplicación de un filtro a una imagen se realiza mediante:
- **Correlación**. Es el proceso de mover la máscara de un filtro sobre la imagen y calcular la suma de productos en cada lugar, exactamente como se acaba de describir.
- **Convolución**. Es una correlación con el filtro girado $180°$.

Correlación vs convolución

- Correlación 2D:

  result = cv2.filter2D(image, -1, filter)

$$R[m,n] = \sum_k \sum_l H[k,l] I[m+k, n+l]$$

- Convolución 2D:

  result = scipy.signal.convolve2d(image, filter, [opts])

$$R[m,n] = \sum_k \sum_l H[k,l] I[m-k, n-l]$$

- La convolución es la correlación con el filtro en espejo.

  cv2.filter2D(image, -1, cv2.flip(filter, -1)) es igual que
  scipy.signal.convolve2d(image, filter, mode='same', boundary='symm')

Propiedades de los filtros lineales

- Linealidad:

  $H(f_1 + f_2) = H(f_1) + H(f_2)$

- Invarianza de corrimiento (shift invariance): mismo comportamiento independientemente de la posición del pixel.

  $H(\text{shift}(f) = \text{shift}(H(f))$

- Cualquier operador lineal con invarianza de corrimiento se puede representar como una convolución.
- Conmutatividad: $a \star b = b \star a$

  Conceptualmente no hay diferencia entre filtro e imagen.

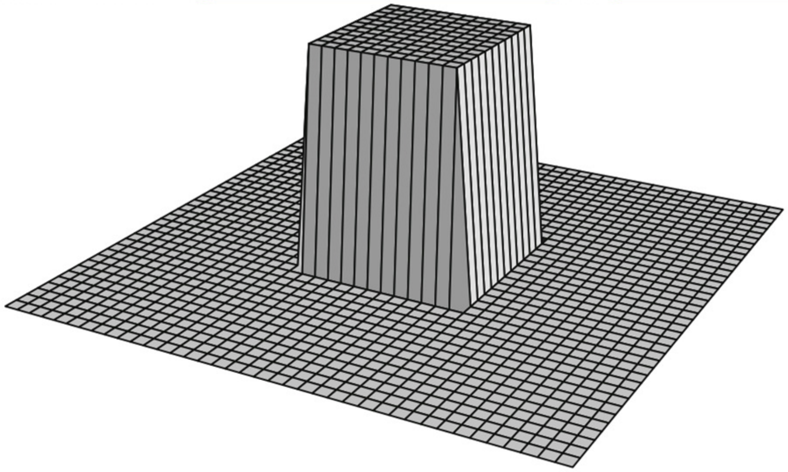
Propiedades de los filtros lineales

- Asociatividad: $a \star (b \star c) = (a \star b) \star c$

  Aplicar varios filtros en serie: $(((a \star b_1) \star b_2) \star b_3)$
- Es equivalente a aplicar un filtro: $a \star (b_1 \star b_2 \star b_3)$
- Distributividad sobre la suma: $a \star (b + c) = (a \star b) + (a \star c)$
- Los escalares factorizan: $ka \star b = a \star kb = k(a \star b)$
- Identidad: pulso unitario $e = [0, 0, 1, 0, 0]$

  $a \star e = a$

Filtro de caja

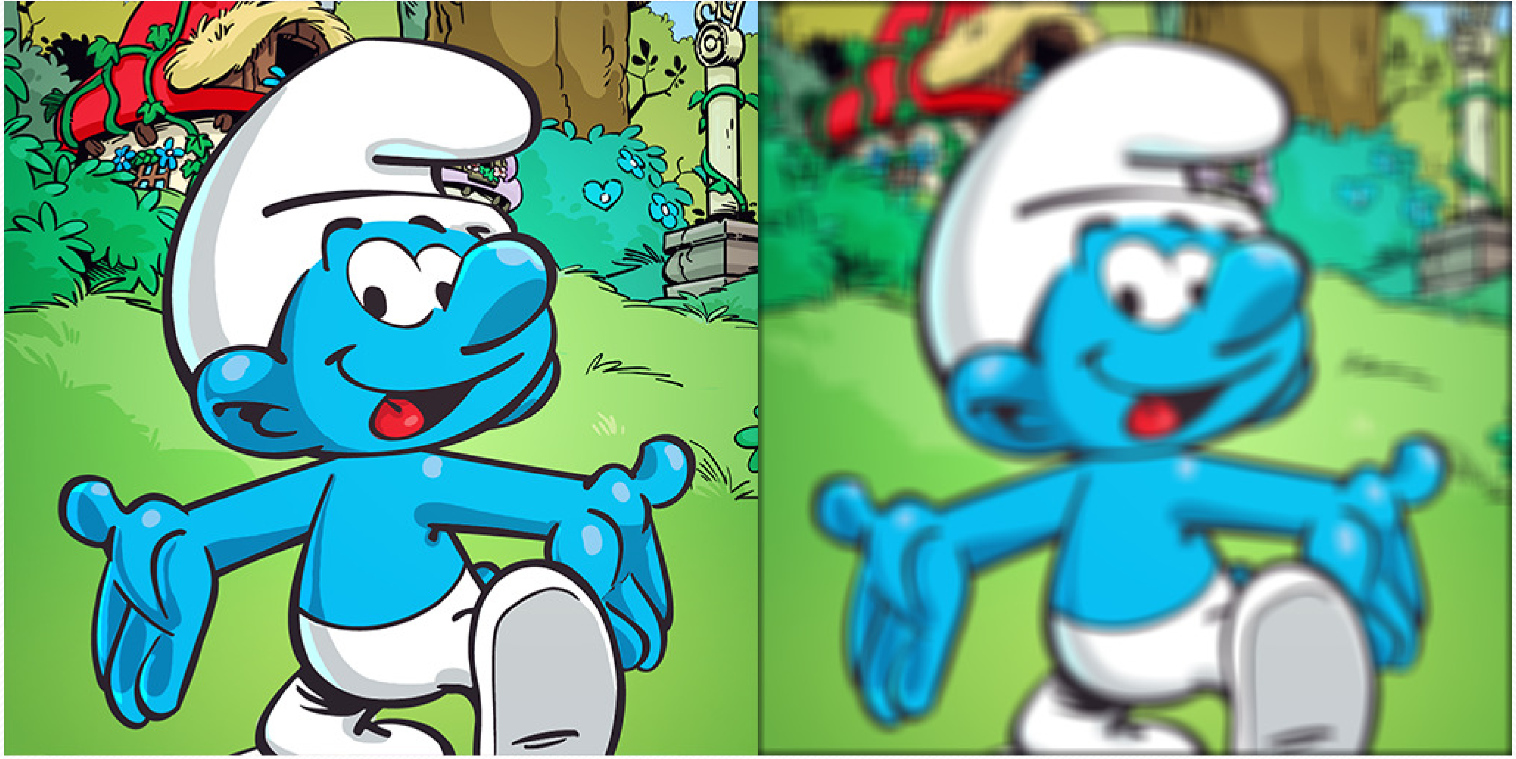


$$\frac{1}{9} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

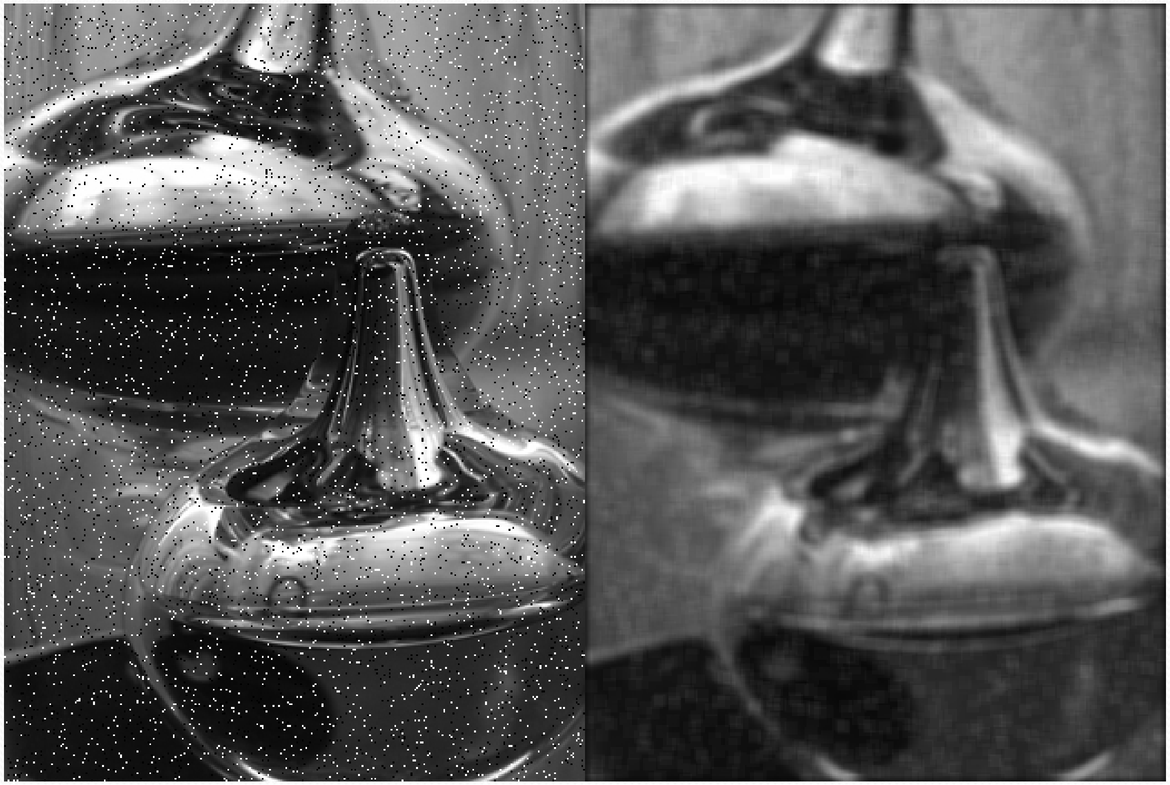
Filtro de caja

- Desventajas:
- No es isotrópico (es decir, no es circularmente simétrico). Suaviza más a lo largo de las diagonales que a lo largo de las filas y columnas.
- Los pesos tienen un corte abrupto en lugar de decaer gradualmente a cero, lo que, por lo general, produce artefactos en la imagen suavizada.

Filtro de caja



Filtro de caja con ruido sal y pimienta



Filtro de caja 3 x 3 con OpenCV

- Opción 1:

```
kernel = np.ones((3, 3), np.float32) / 9
image_blur = cv2.filter2D(image, ddepth=-1, kernel=kernel,
                          borderType=cv2.BORDER_CONSTANT)
```

- Opción 2:

```
image_blur = cv2.blur(image, ksize=(3, 3),
                      borderType=cv2.BORDER_CONSTANT)
```

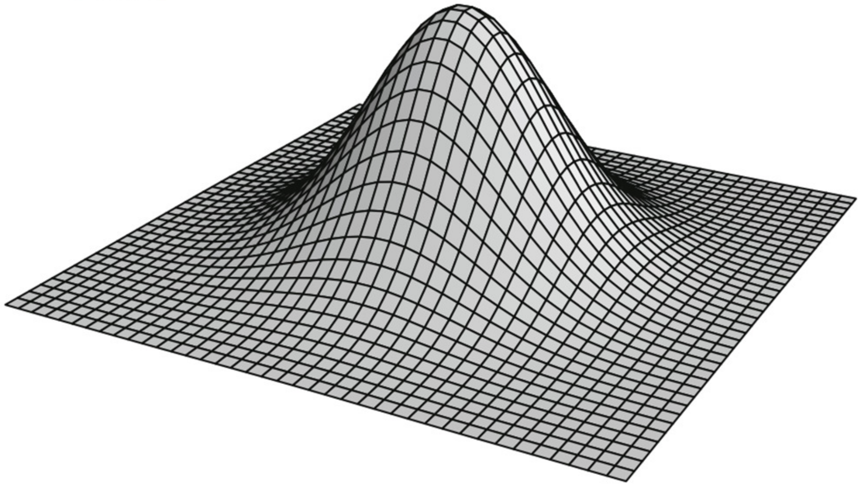
Filtro gaussiano

- Este filtro corresponde a la función gaussiana en 2D
- $H^{G,\sigma}(x,y) = e^{-\left(\frac{(x-x0)^2}{2\sigma_x^2}+\frac{(y-y0)^2}{2\sigma_y^2}\right)}$
- Donde:
- σ_x^2 y σ_y^2 es la varianza en x y en y, respectivamente, y determina la anchura de la función en x y en y.
- $(x0, y0)$ son las coordenadas del centro del filtro, generalmente $(0, 0)$.
- La distancia, r, con respecto al centro del filtro, es $r = \sqrt{x^2 + y^2}$.

Filtro gaussiano

- Si el centro del filtro está en $(0, 0)$ y si $\sigma_x = \sigma_y = \sigma$, la función Gaussiana normalizada es:
- $G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$
- Los pesos de los pixeles vecinos es por cercanía.

Filtro gaussiano



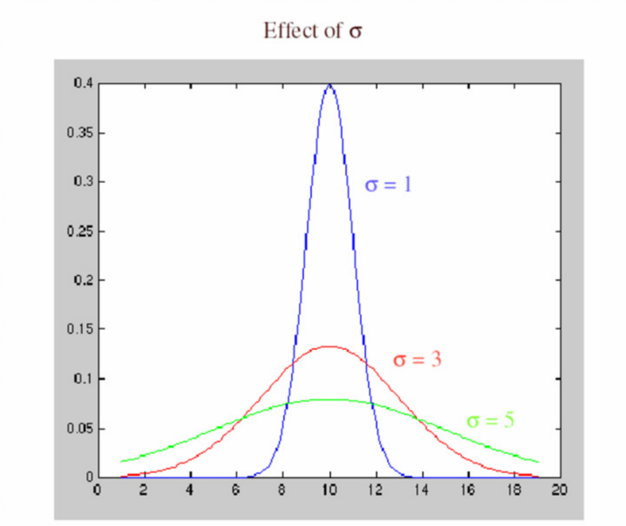
$$\frac{1}{57} \cdot \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 5 & 3 & 1 \\ 2 & 5 & 9 & 5 & 2 \\ 1 & 3 & 5 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

Filtro gaussiano

- El filtro gaussiano es isotrópico si la matriz del filtro es lo suficientemente grande (al menos 5 x 5).
- Es superior al filtro de caja.
- El filtro gaussiano 2D se puede separar en dos filtros 1D lo que facilita su implementación eficiente.
- La convolución consigo misma es otra gaussiana.
- Convolucionar dos veces con un kernel de ancho σ es equivalente a convolucionar una vez con un kernel de ancho $\sigma\sqrt{2}$.

Cuestiones prácticas

- Tamaño del filtro.
- Los valores en los bordes deben ser casi cero.
- Regla de dedo para un filtro gaussiano: poner la mitad del ancho $\geq 3\sigma$.



Cuestiones prácticas

- ¿Qué hacer cerca de los bordes?
- El kernel del filtro se sale de la imagen.
- Métodos:
- Color constante (color negro).
- Wrap around (envolver alrededor).
- Replicar.
- Reflejar.

Cuestiones prácticas

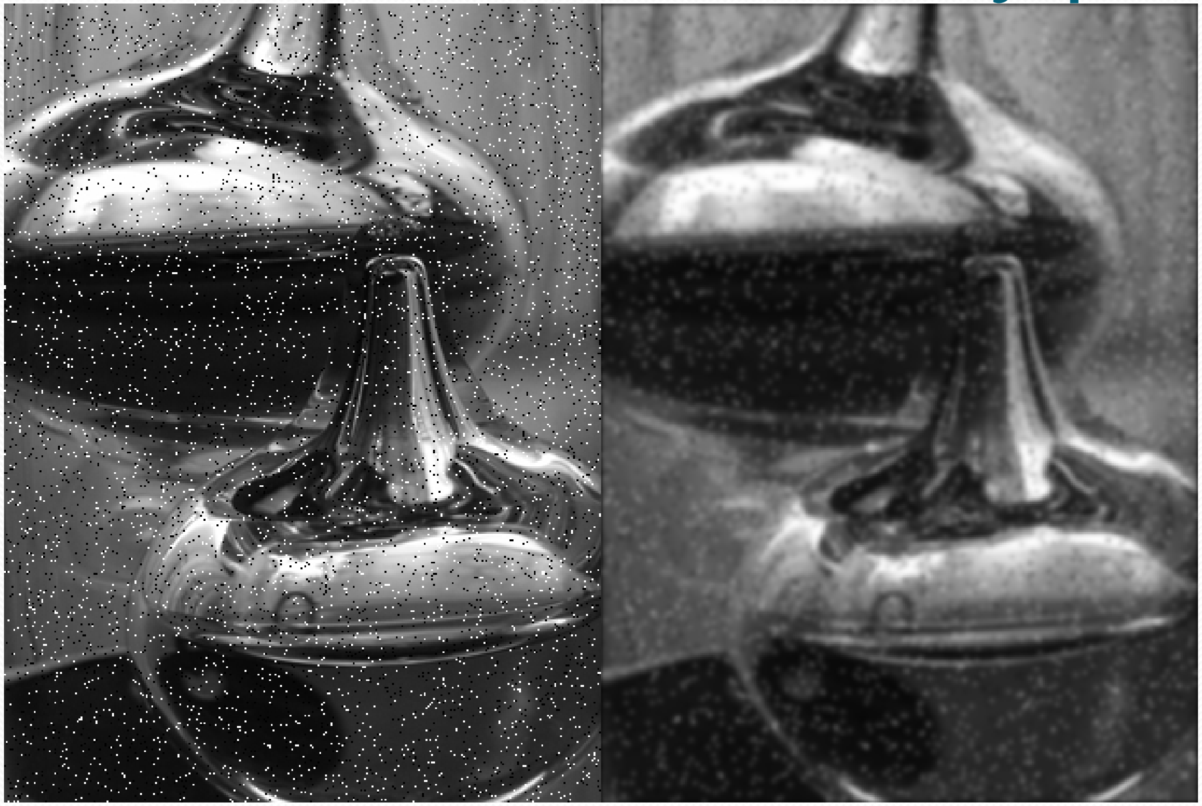
- Opciones en cv2.filter2D():

Constante	boundaryType=cv2.BORDER_CONSTANT
Replicar	boundaryType=cv2.BORDER_REPLICATE
Wrap	boundaryType=cv2.BORDER_WRAP
Reflejar	boundaryType=cv2.BORDER_REFLECT

Cuestiones prácticas

- Tamaño de la salida.
- En OpenCV, cv2.filter2D(), cv2.blur() y cv2.GaussianBlur() preservan el tamaño de la salida dependiendo del tipo de borde.

Filtro gaussiano con ruido sal y pimienta



Filtro gaussiano 3 x 3 en OpenCV

- Opción 1:

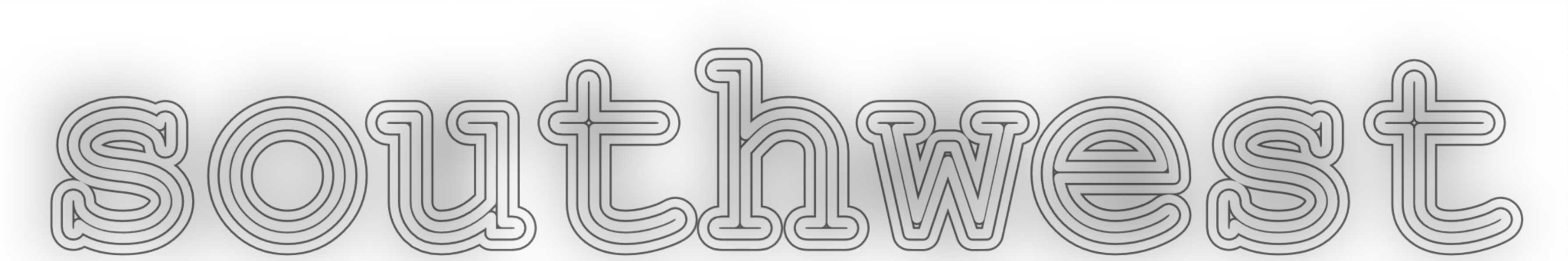
```
kernel = (cv.getGaussianKernel(ksize=3, sigma=0.4) @
              cv.getGaussianKernel(ksize=3, sigma=0.4).T)
image_blur = cv.filter2D(image, ddepth=-1, kernel=kernel,
                         borderType=cv.BORDER_REPLICATE)
```

- Opción 2:

```
image_blur = cv.GaussianBlur(image, ksize=(3, 3), sigmaX=0,
                             borderType=cv.BORDER_REPLICATE)
```

Aplicación: imágenes híbridas

- **Imagen híbrida**. Imagen que se percibe de dos maneras diferentes, dependiendo de la distancia.

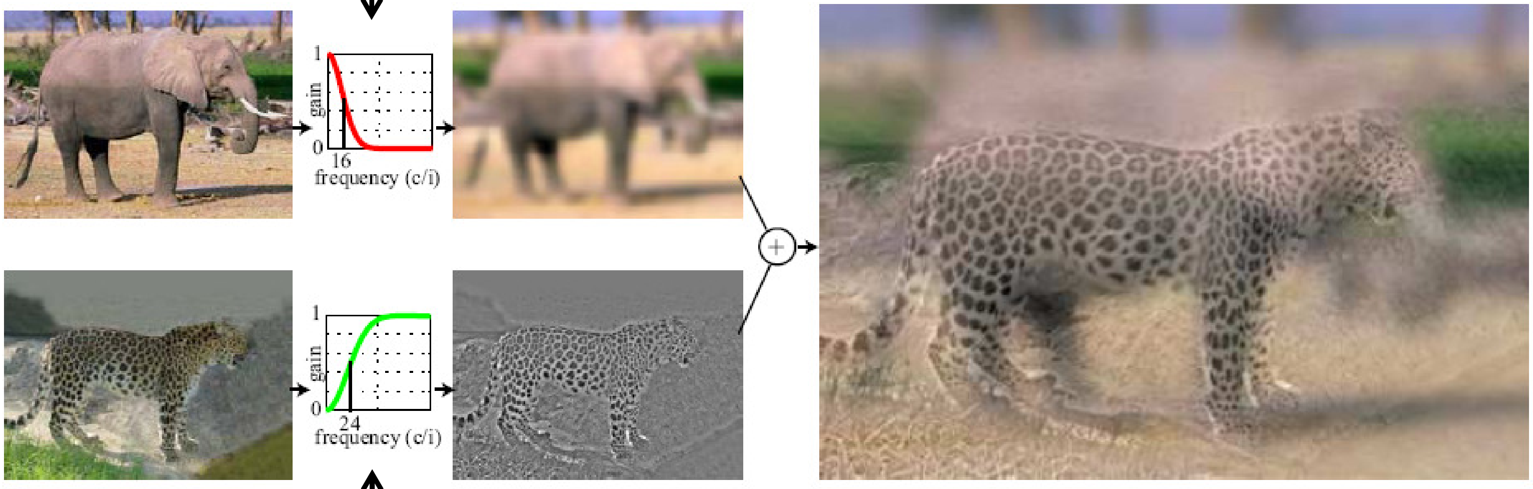


Fuente: By Cmglee - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60520675

Imágenes híbridas

- La idea es que las frecuencias altas tienden a dominar la percepción cuando la imagen está cerca.
- A la distancia, solo se puede ver la parte de baja frecuencia de la señal.
- Al combinar la parte de alta frecuencia de una imagen con la parte de baja frecuencia de otra, se obtiene una imagen híbrida que conduce a diferentes interpretaciones a diferentes distancias.

Imágenes híbridas

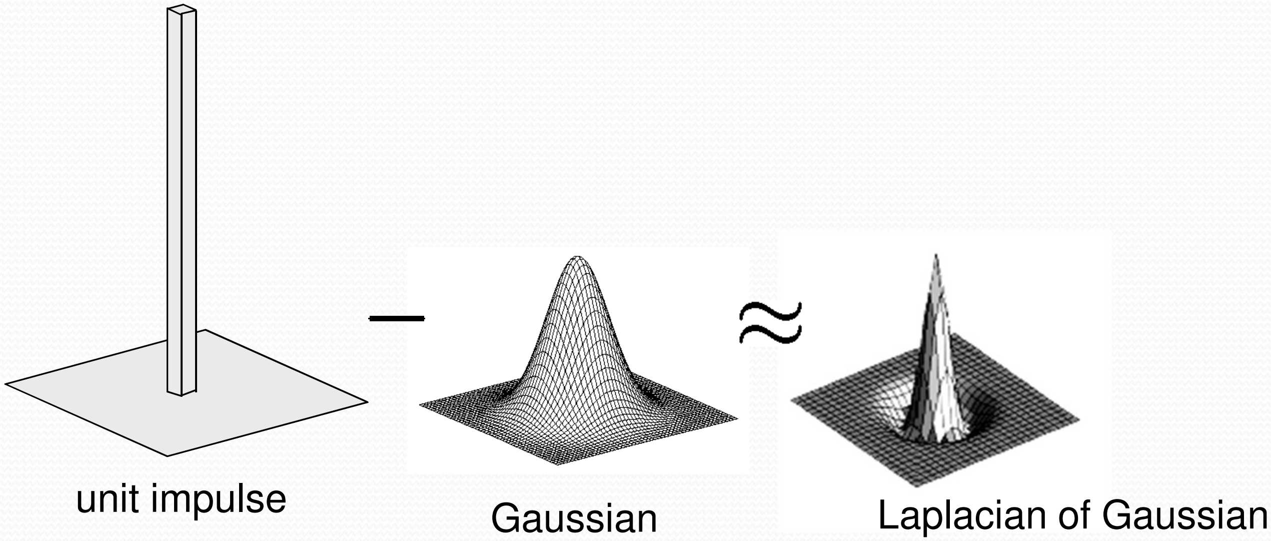

1
gain
0
16
frequency (c/i)
1
gain
0
24
frequency (c/i)
+

Gaussian Filter!

Laplacian Filter!

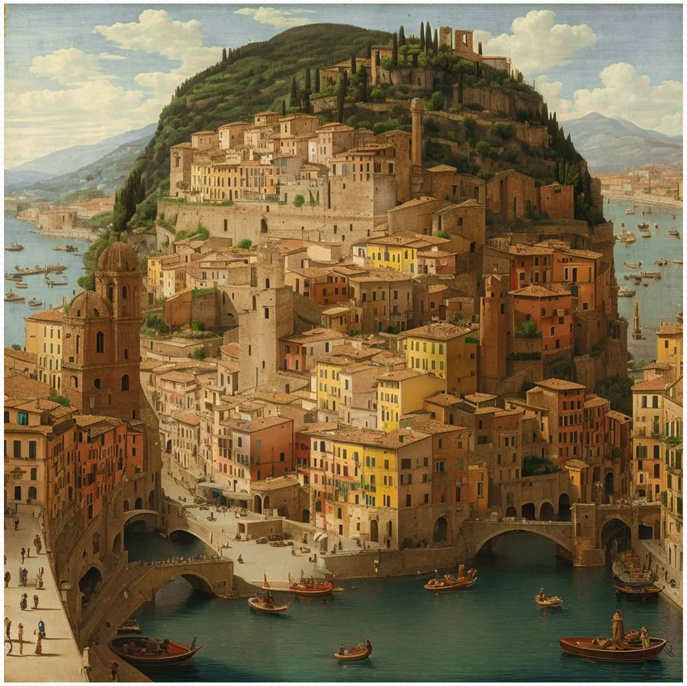
Fuente: Derek Hoiem (UIUC)

Imágenes híbridas


unit impulse
−
Gaussian
≈
Laplacian of Gaussian

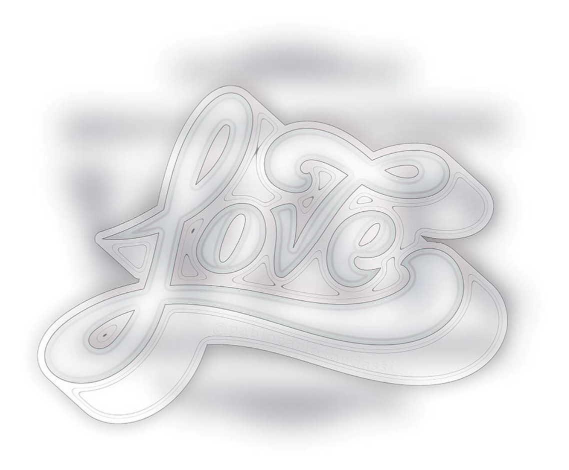
Fuente: Derek Hoiem (UIUC)

Imágenes híbridas



Fuente: By AI generator: Illusion Diffusion, base image: Leonardo da Vinci - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=149738918

Imágenes híbridas



Fuente: **By Pablo Carlos Budassi - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=118454167**

Resumen

- El filtrado lineal es un producto escalar en cada posición.
- Puede suavizar, sharpen, trasladar, etc.
- Tener en cuenta los detalles del tamaño del filtro y del tratamiento de los bordes.