Filtros

Temario

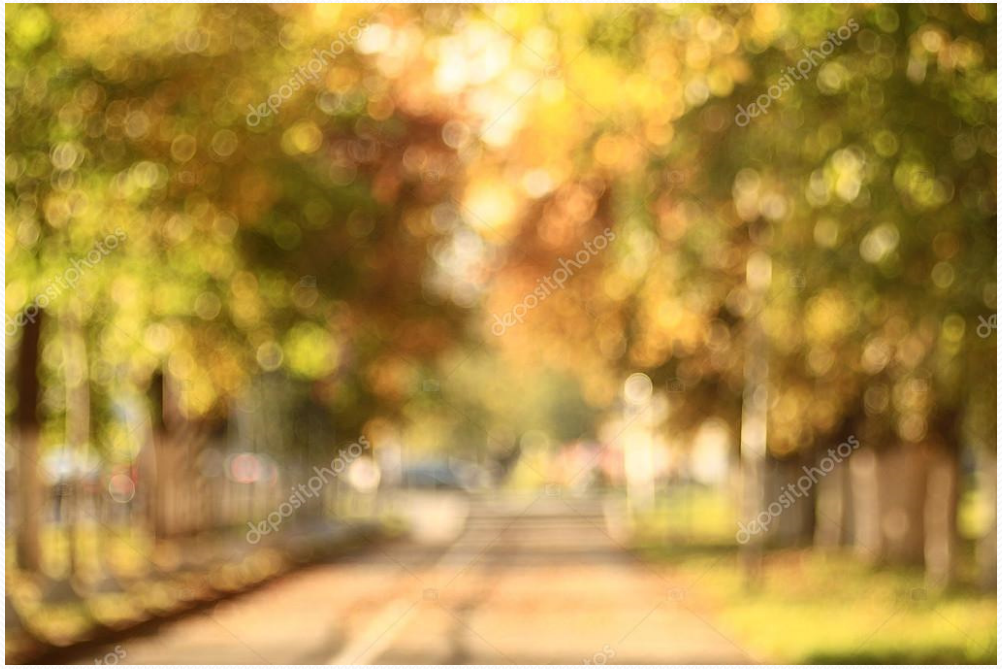
- Filtros en el dominio espacial
- Filtro de caja
- Filtros gaussiano
- Image sharpening
- Aplicaciones

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.



Filtros

- **Filtrado de imágenes**. Calcula una función de los vecinos en cada posición.
- Funciones de los filtros:
- Mejorar imágenes.
  - Eliminar ruido, cambiar el tamaño, aumentar el contraste, 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

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

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							

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

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

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

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					

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

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				

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

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				
				?					

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

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

$h[.,.]$

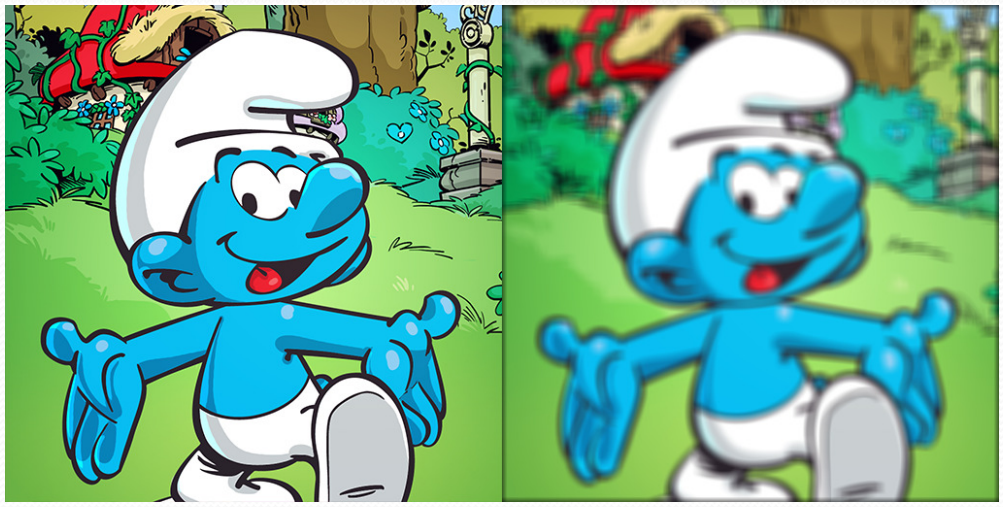
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

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

0	1	1	0
1	2	2	0
0	0	0	1
0	1	1	2

*

1	0	0
0	1	0
0	0	1

=

0	1	1	0
1	2	2	0
0	0	0	1
0	1	1	2

Fuente: Derek Hoiem (UIUC)

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

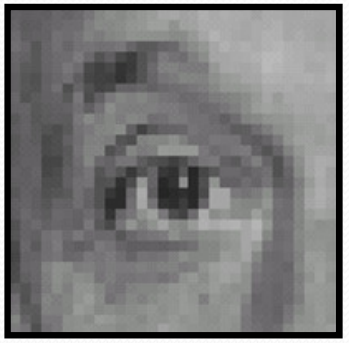
?

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

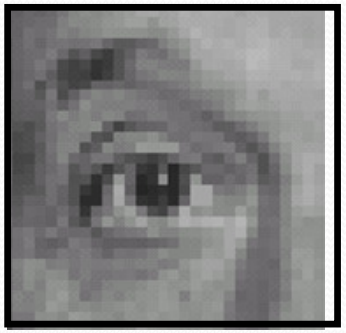
?

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

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

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

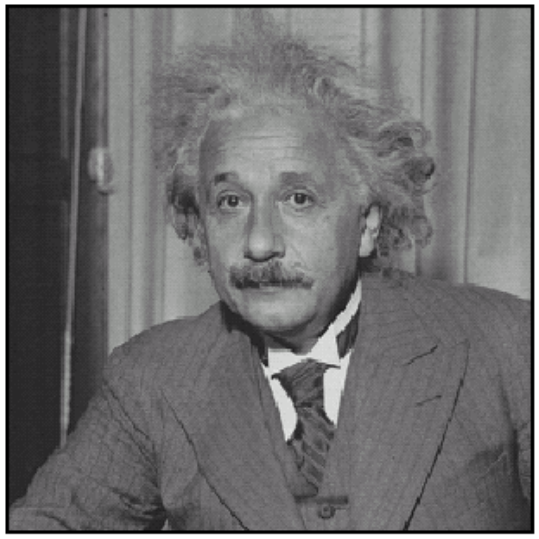
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

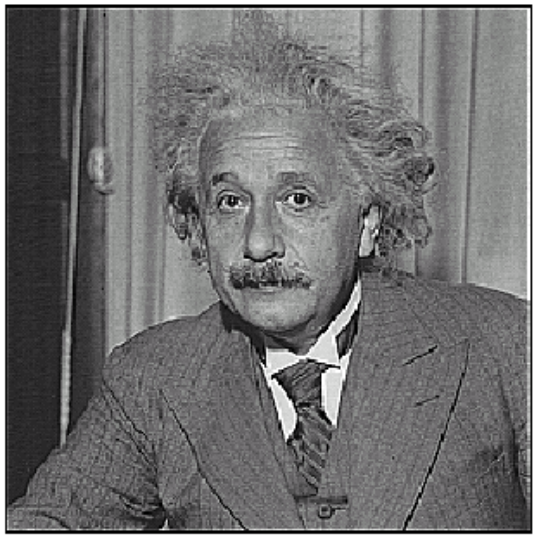
- Accentuates differences with local average

Source: D. Lowe

Sharpening



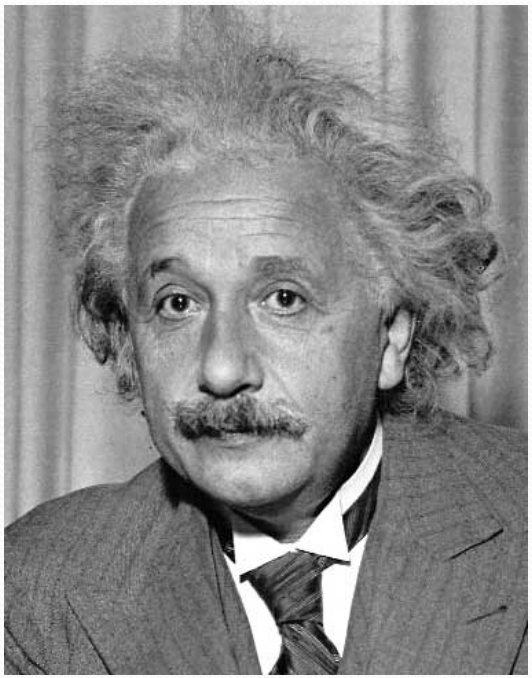
before



after

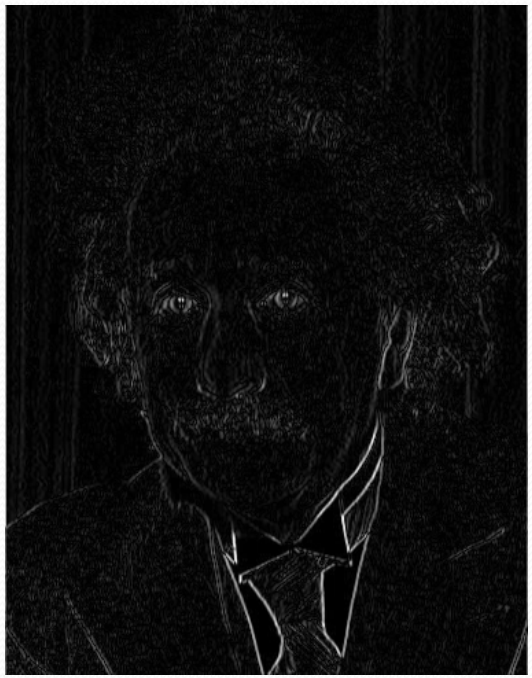
Source: D. Lowe

Other filters



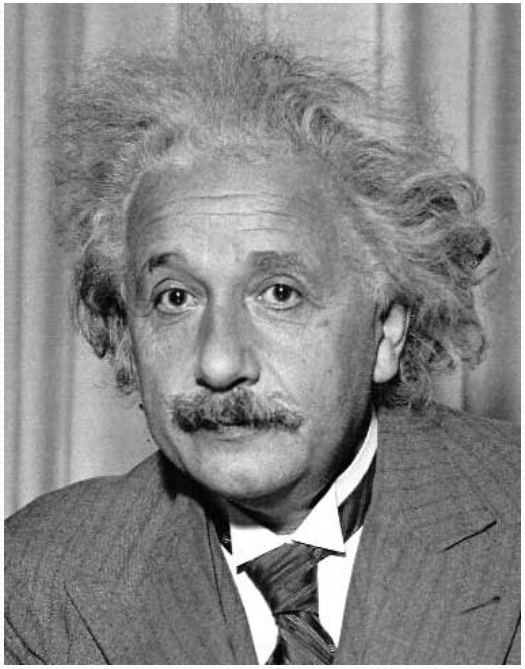
1	0	-1
2	0	-2
1	0	-1

Sobel



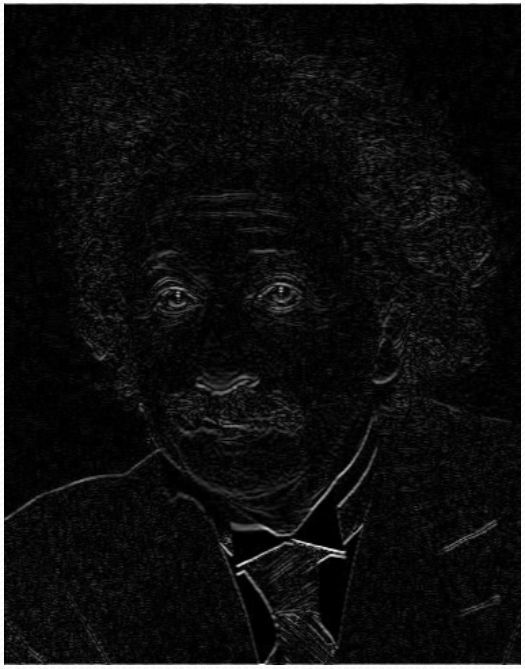
Vertical Edge
(absolute value)

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.

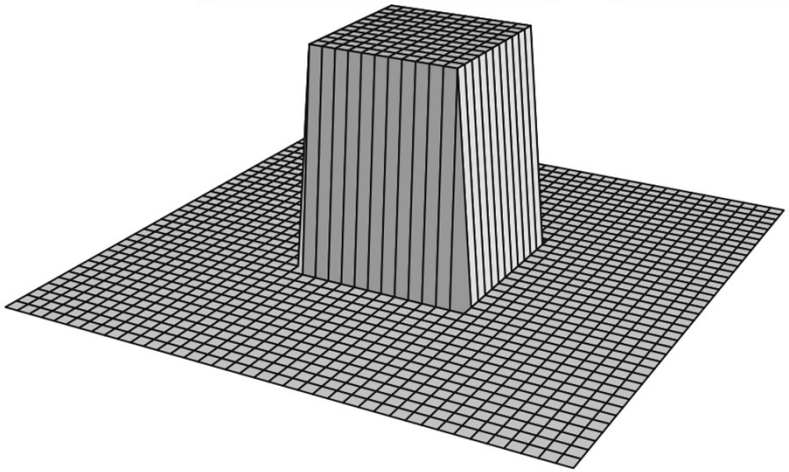
Propiedades de los filtros lineales

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

  Aplicar varios filtros en serie: $(\big((a \star b_1) \star b_2\big) \star b_3)$
- 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

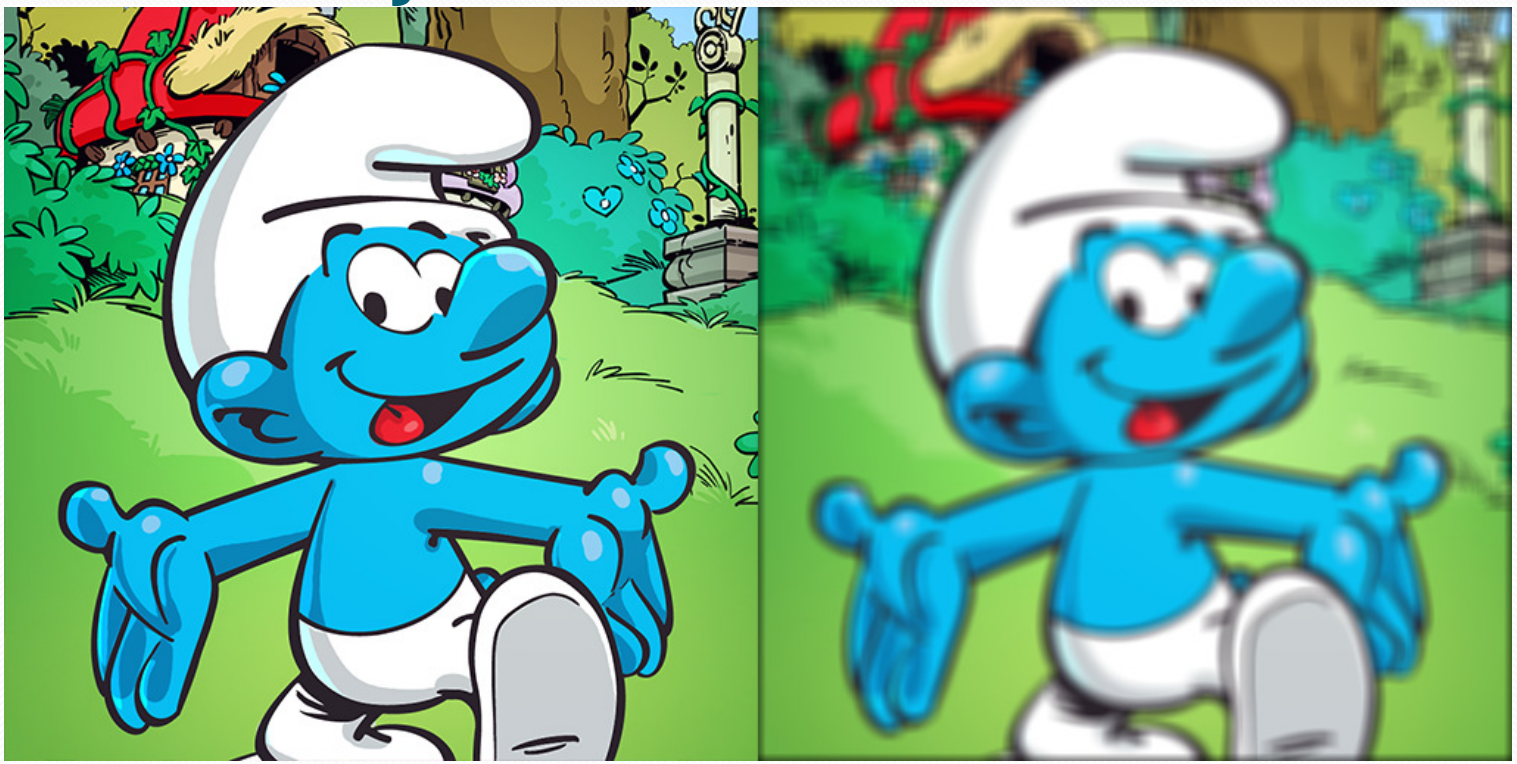


$$\frac{1}{9} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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

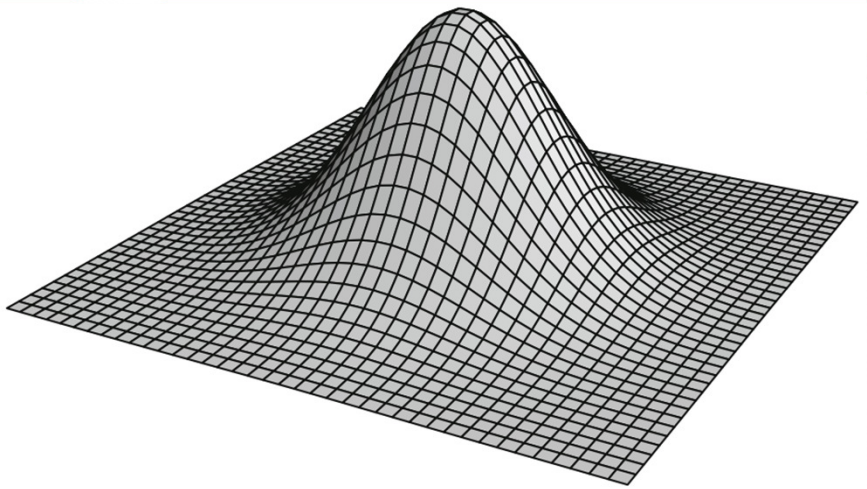
Filtro gaussiano

- 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



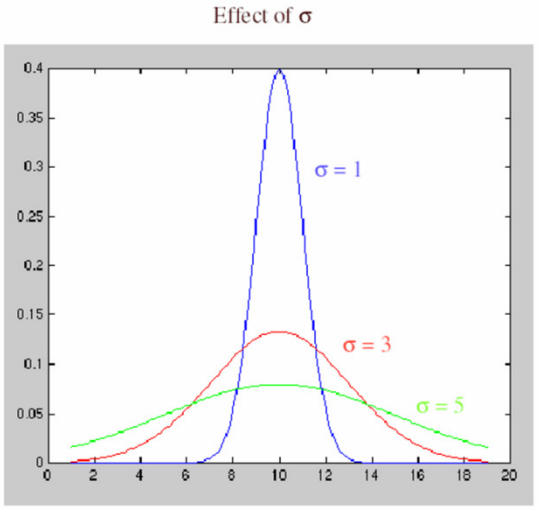
$$\frac{1}{57} \cdot \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 5 & 3 & 1 \\ 2 & 5 & 9 & 5 & 2 \\ 1 & 3 & 5 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

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.
- Se puede aplicar un filtro pequeño, repetir, y obtener el mismo resultado que con un kernel grande.
- 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

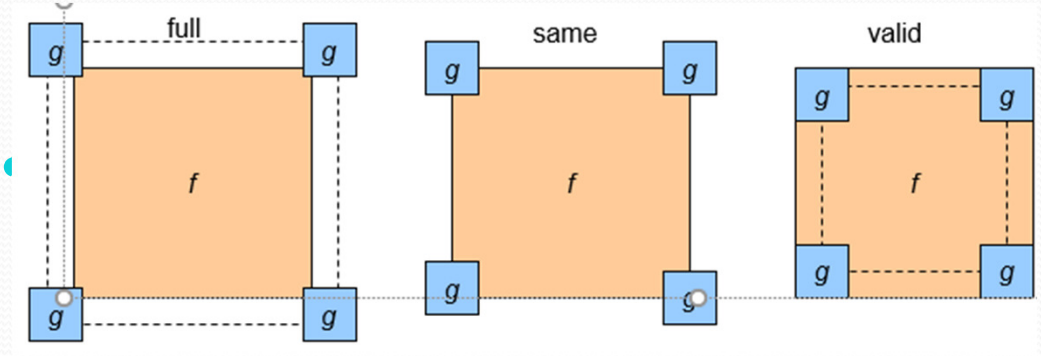
- ¿Qué hacer cerca de los bordes?
- El kernel del filtro se sale de la imagen.
- Se necesita extrapolar.
- Métodos:
- Cortar el filtro (color negro).
- Wrap around (envolver alrededor).
- Copiar el borde.
- Reflejar el borde.

Cuestiones prácticas

- Métodos en Python:
- Cortar el filtro (color negro) `convolve2d(f, g, boundary='fill', 0)`
- Wrap around `convolve2d(f, g, boundary='wrap')`
- Reflejar el borde `convolve2d(f, g, boundary='symm')`

Cuestiones prácticas

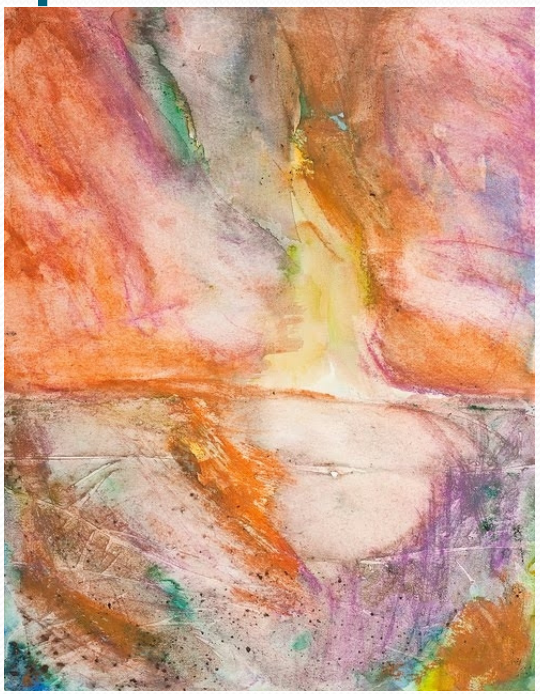
- Tamaño de la salida.
- Python: convolve2d(f, g, mode)
- mode='full': el tamaño de la salida es la suma de los tamaños de f y g.
- mode='same': el tamaño de la salida es el tamaño de f
- mode='valid': el tamaño de la salida es la diferencia de los tamaños de f y g.



Fuente: Derek Hoiem (UIUC)

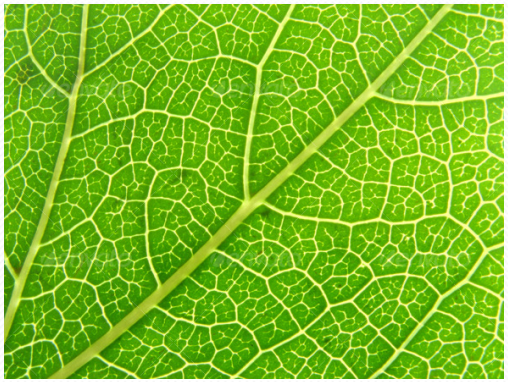
Filtro gaussiano con ruido sal y pimienta

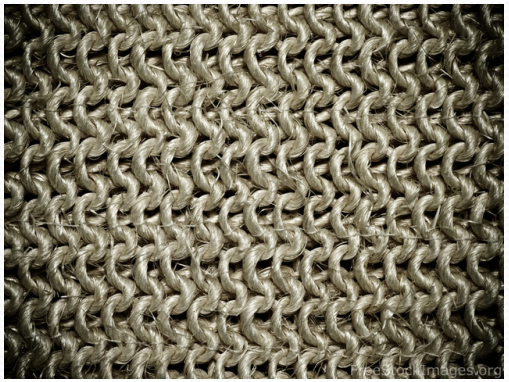
Aplicación: representación de texturas



Fuente: https://www.robots.ox.ac.uk/~vgg/data/dtd/

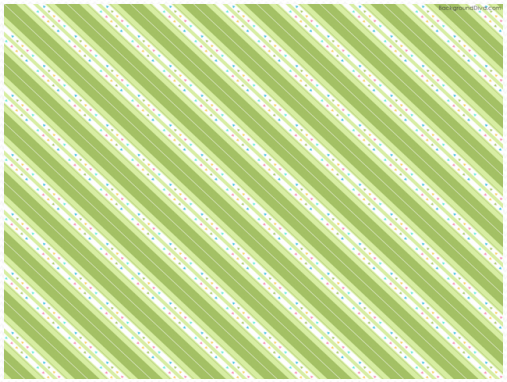
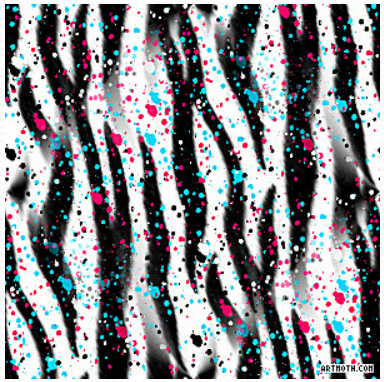
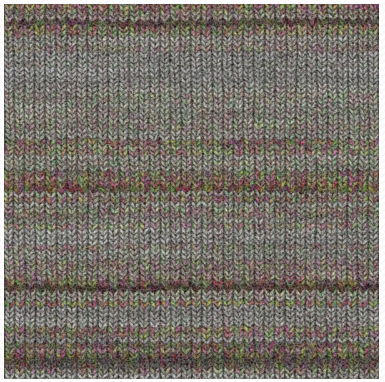
Textura y material





Fuente: https://www.robots.ox.ac.uk/~vgg/data/dtd/

Textura y orientación

Fuente: https://www.robots.ox.ac.uk/~vgg/data/dtd/

Textura y escala

Fuente: https://www.robots.ox.ac.uk/~vgg/data/dtd/

¿Qué es una textura?

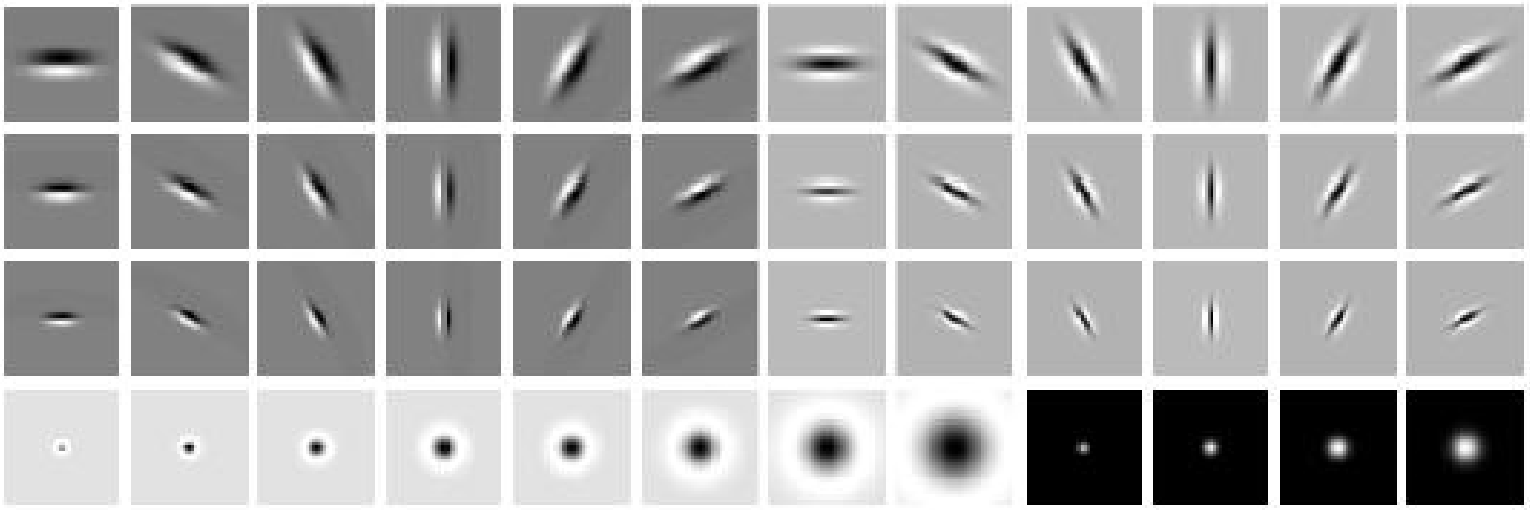
- Patrones regulares o estocásticos causados por abultamientos, canales, y/o marcas.

¿Cómo se pueden representar texturas?

- Calculando las respuestas de blobs y bordes en diversas orientaciones y escalas.

Representación con bancos de filtros

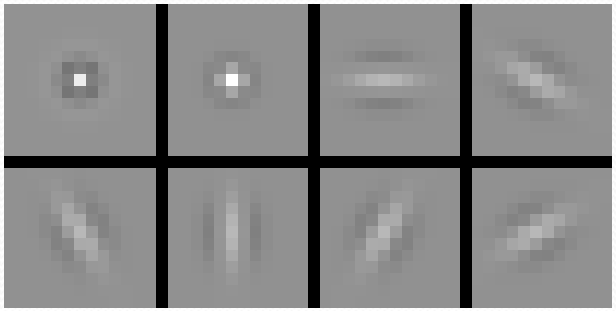
The Leung-Malik (LM) Filter Bank

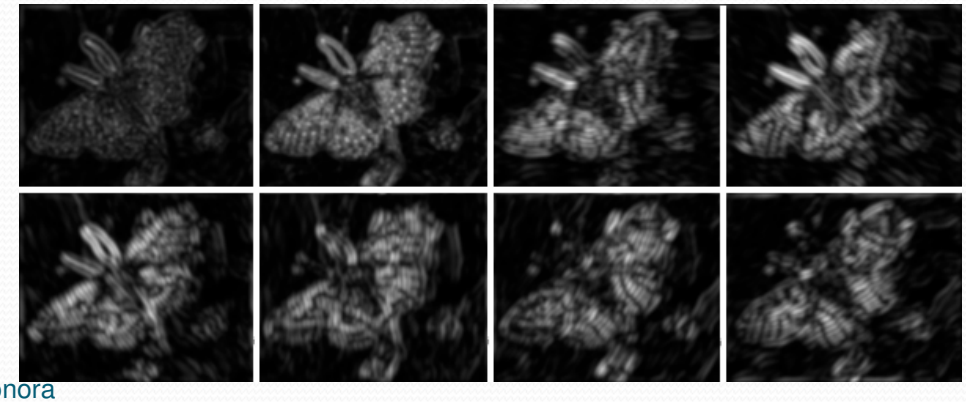


Fuente: https://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Bancos de filtros

- Procesar la imagen con cada filtro y guardar las respuestas.



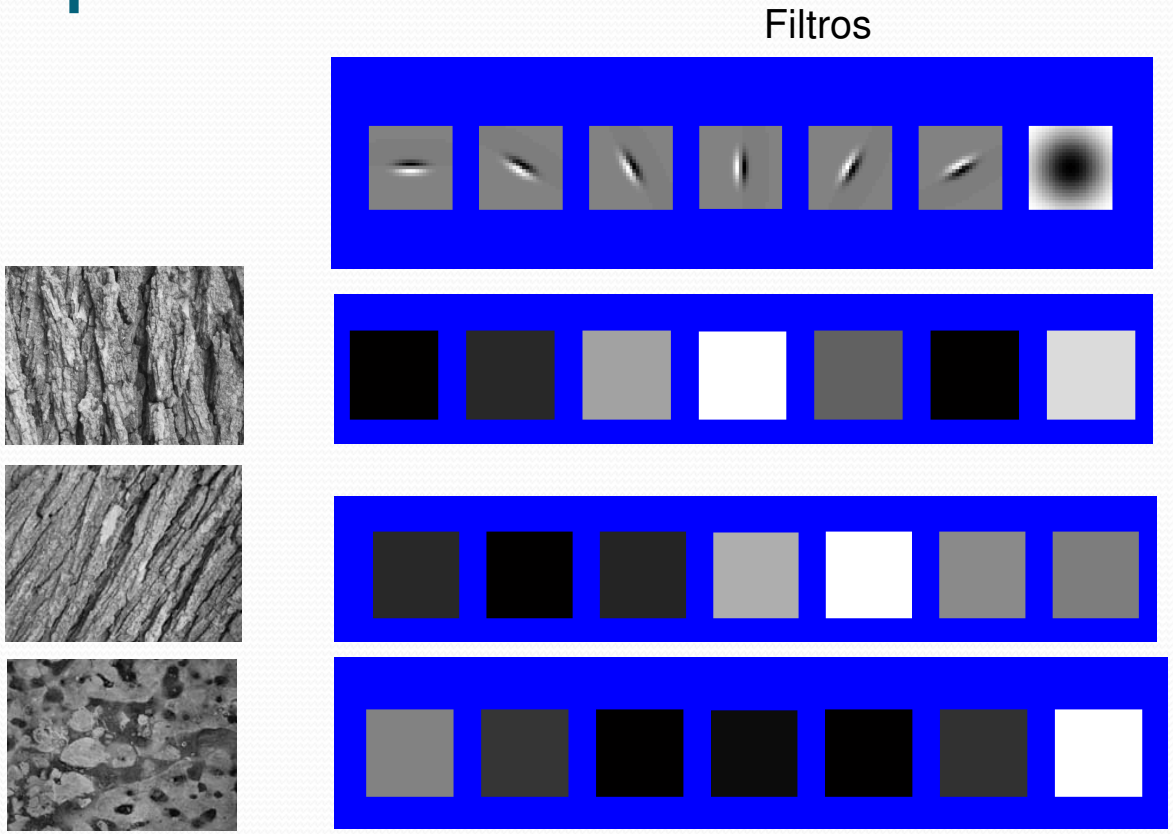


Fuente: Derek Hoiem (UIUC)

¿Cómo se pueden representar texturas?

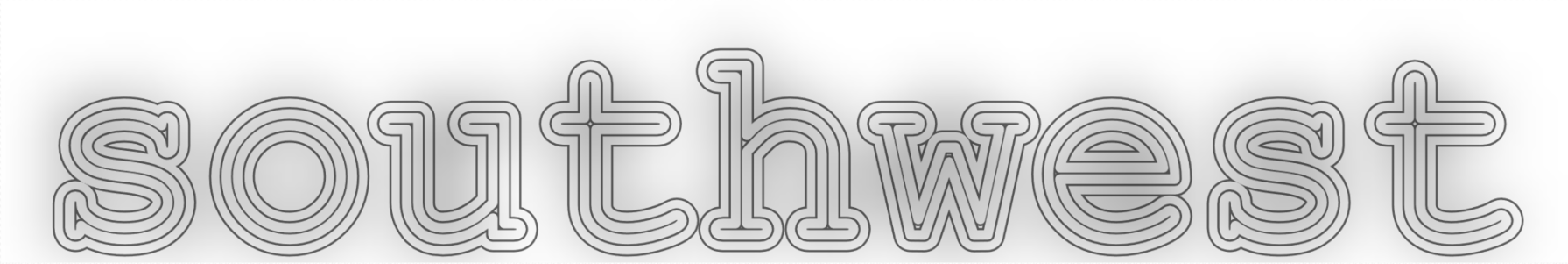
- Calculando las respuestas de blobs y bordes en diversas orientaciones y escalas.
- Registrando estadísticas simples (p.e. media, desviación standard) de las respuestas absolutas del filtro.

Respuestas medias absolutas



Imágenes híbridas

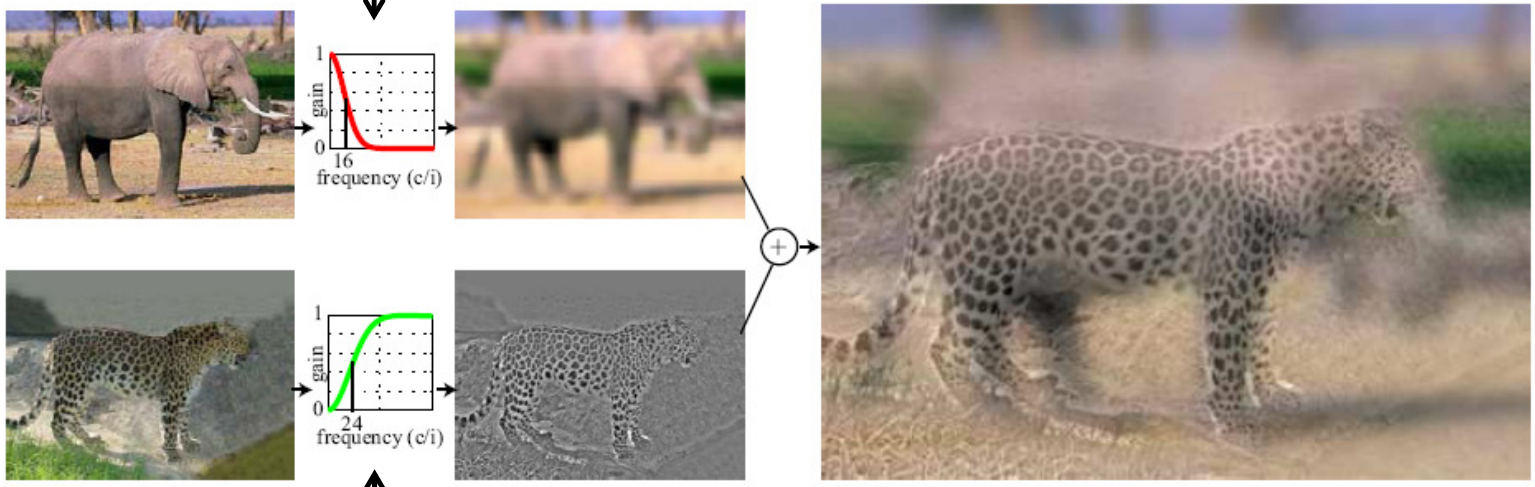
- **Imagen híbrida**. Imagen que se percibe de dos maneras diferentes, dependiendo de la distancia de visualización-



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

Imágenes híbridas

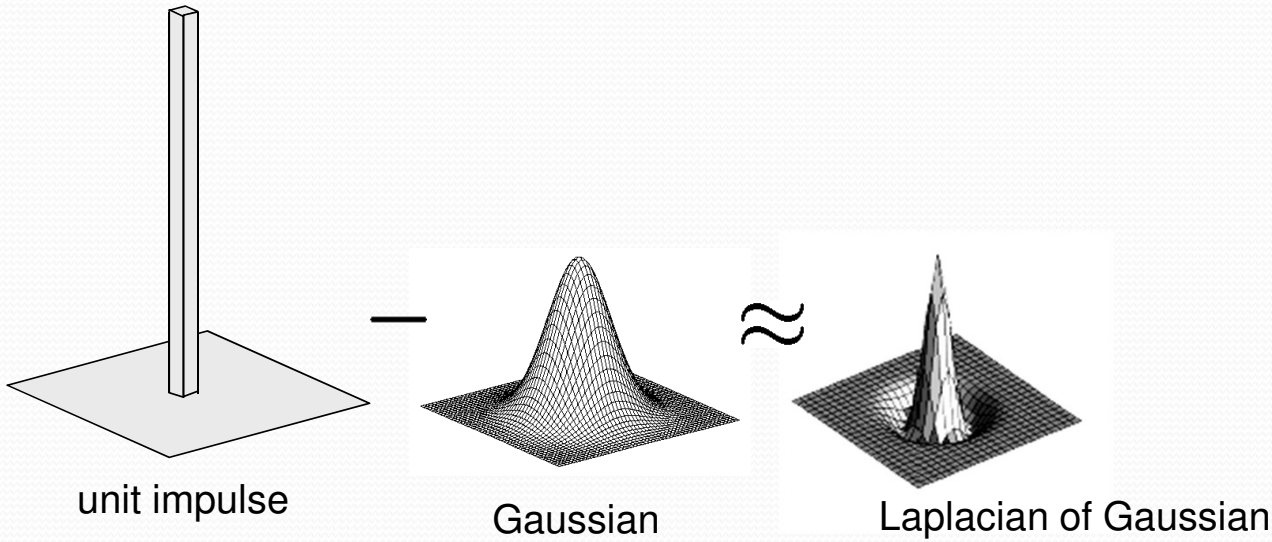
Gaussian Filter!



Laplacian Filter!

Fuente: Derek Hoiem (UIUC)

Imágenes híbridas



Fuente: Derek Hoiem (UIUC)

Resumen

- Una imagen es una matriz de números.
- 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 bordes.