Visión por Computador

Tema 3. Procesamiento de imagen

EII, GII. *Modesto Castrillón Santana*

Contenidos

- Operadores
- Puntuales
- Filtrado
- Pirámides
- Geométricos
- Transformadas

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

Operadores

Transforma una, f, o varias imágenes, f_i, en otra g

$$g=T(f(x))$$
$$g=T(f_1(x), \ldots, f_n(x))$$

- Píxel/puntuales
- Locales (filtrado)
- Geométricos

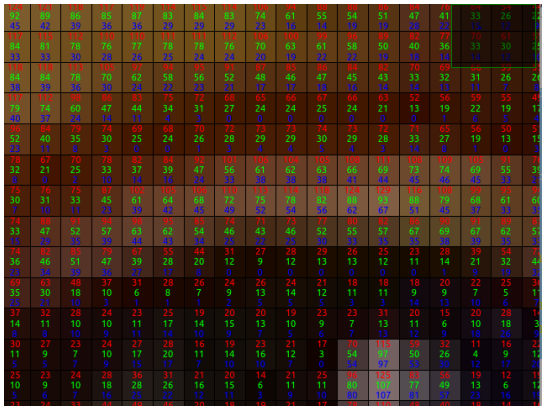
Puntuales

Dependiente del brillo/color del propio píxel
Misma operación para todos

$$\forall x \in D: g(x)=T(f(x))$$

Imágenes discretas 2D

$$g(i, j) = T(f(i,j))$$

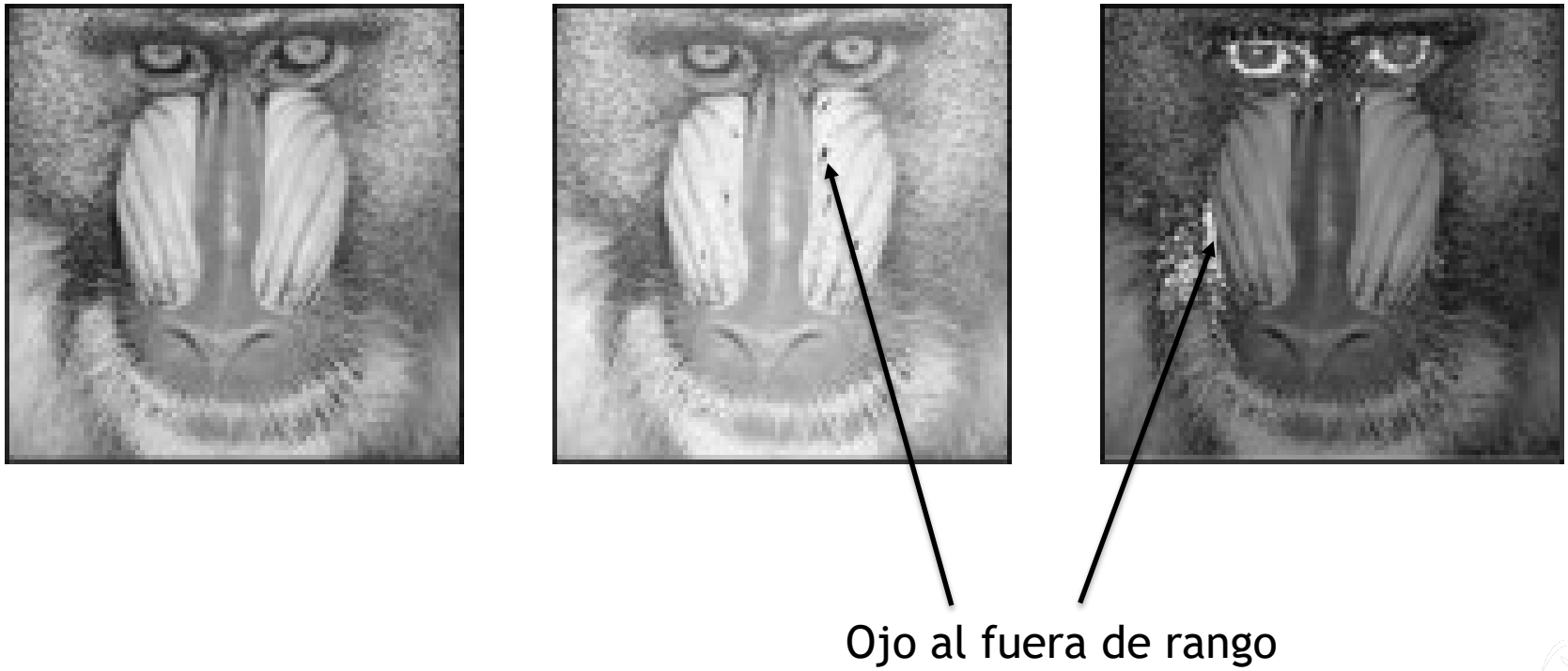


Aditivos/sustractivos

Sumar/restar valor constante

Aclaran u oscurecen

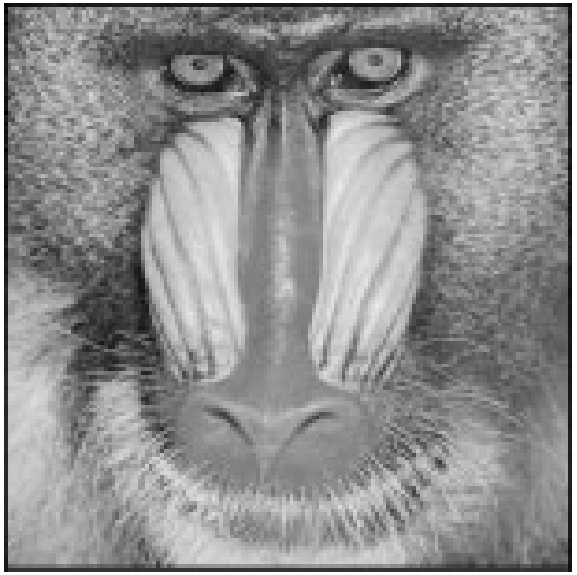
$$g(i, j) = T(f(i,j)) = f(i,j) + k$$

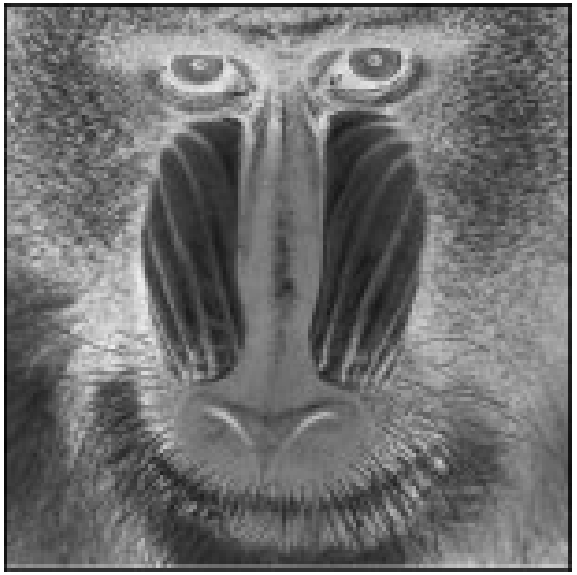


Inversión

Invertir valores

$$g(i, j) = T(f(i,j)) = 255 - f(i,j)$$

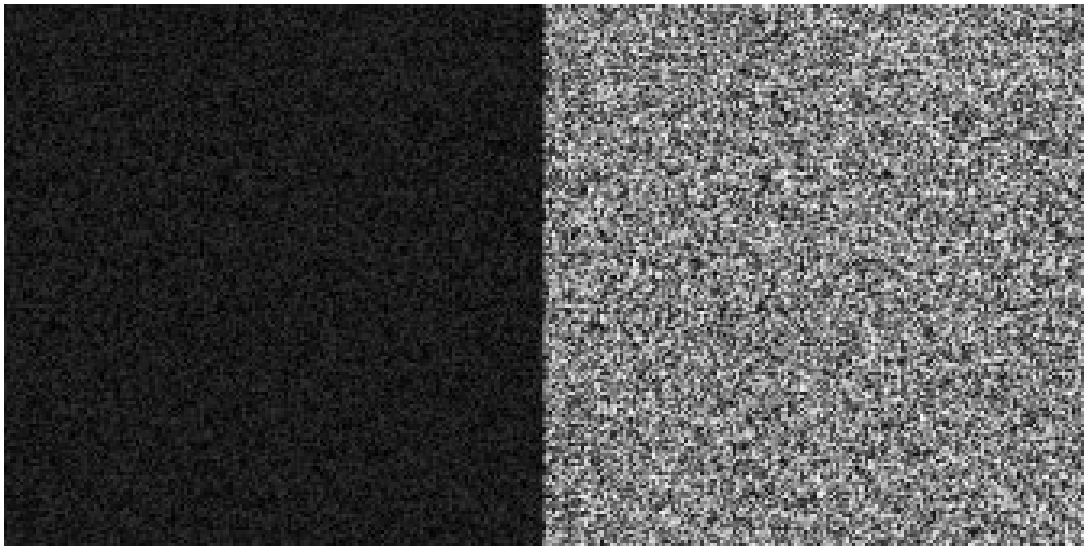




Producto

Multiplicar por valor
Aumenta (>1) o disminuye (<1) el contraste
Ojo al rango

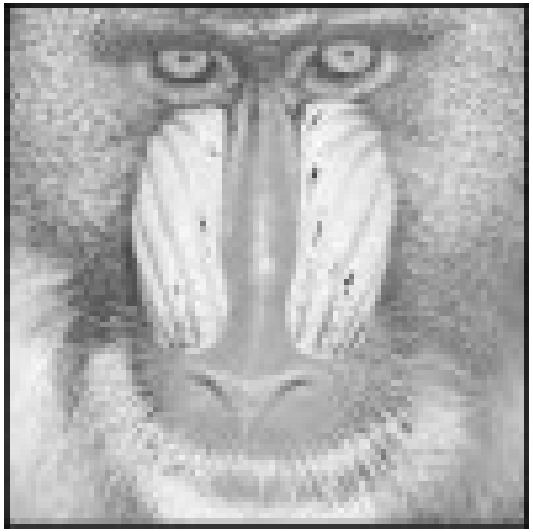
$$g(i, j) = T(f(i,j)) = f(i,j)*a$$



Combinación

Sumar dos imágenes
Promedio, divide entre 2

$$g(i, j) = (f_1(i,j) + f_2(i,j))/2$$



Combinación

Ponderar suma (α-*blending)*

$0 <= \alpha <= 1$

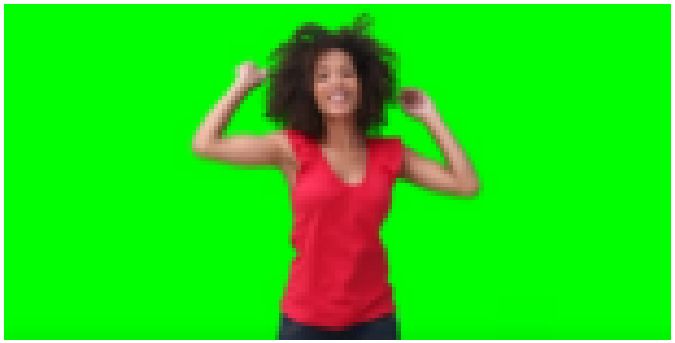
$$g(i, j) = \alpha * f_1(i,j) + (1-\alpha) * f_2(i,j)$$

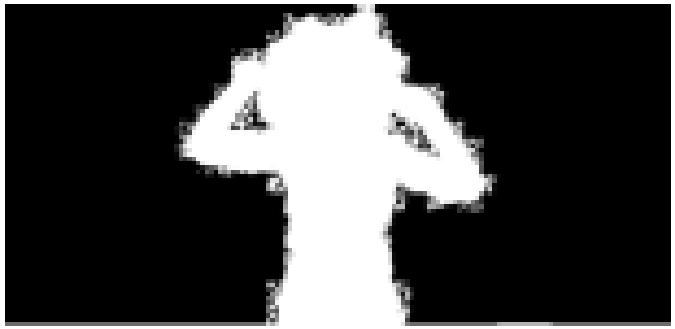
Combinación

Transparencia por píxel
Canal *alpha*
α como imagen

$$g(i, j) = \alpha (i, j) * f_1(i,j) + (1-\alpha (i, j)) * f_2(i,j)$$

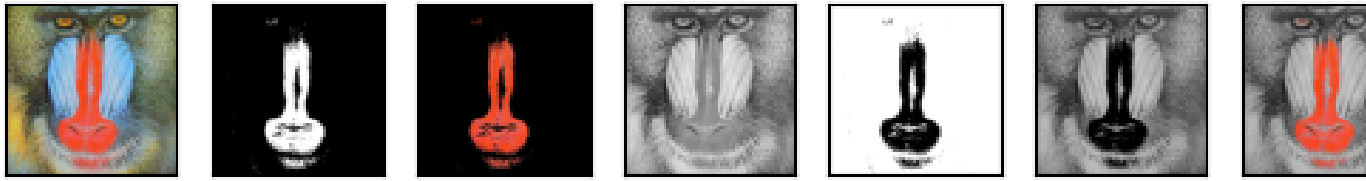
Valor 1 para píxeles opacos y 0 para los transparentes
Ejemplo elimina zona verde (RGB [0 255 0])





Combinación

Por planos de color
Figura y fondo
Rango color



```
imgrgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
imghsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#Máscara rango rojo y extracción desde imgrgb
mask = cv2.inRange(imghsv, (0, 120, 70), (10, 255, 255))
imgmasked = cv2.bitwise_and(imgrgb, imgrgb, mask=mask)
#Máscara inversa (sin rojo) y extracción desde gris
mask_inv = cv2.bitwise_not(mask)
background = cv2.bitwise_and(gris, gris, mask = mask_inv)
#Convierte a tres canales y combina
background = cv2.merge([background,background,background])
added_img = cv2.add(imgmasked, background)
```

Color

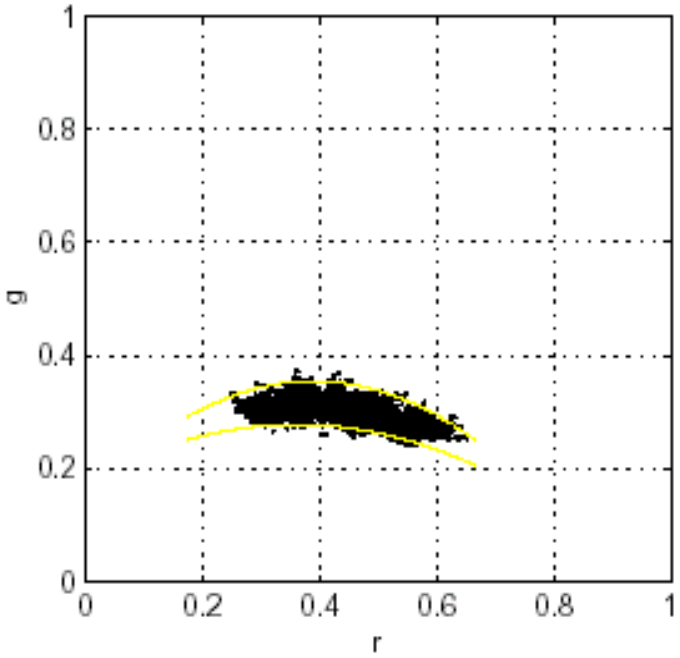
Rangos de color
RGB no perceptualmente uniforme
Conversión entre espacios de color
RGB o BGR a HSV
Convertir un color concreto:

```
col = np.uint8([[[0,255,0 ]]])
hsv_col = cv2.cvtColor(col,cv2.COLOR_BGR2HSV)
print(hsv_col)
```

Tono definido con rango del valor h

Color

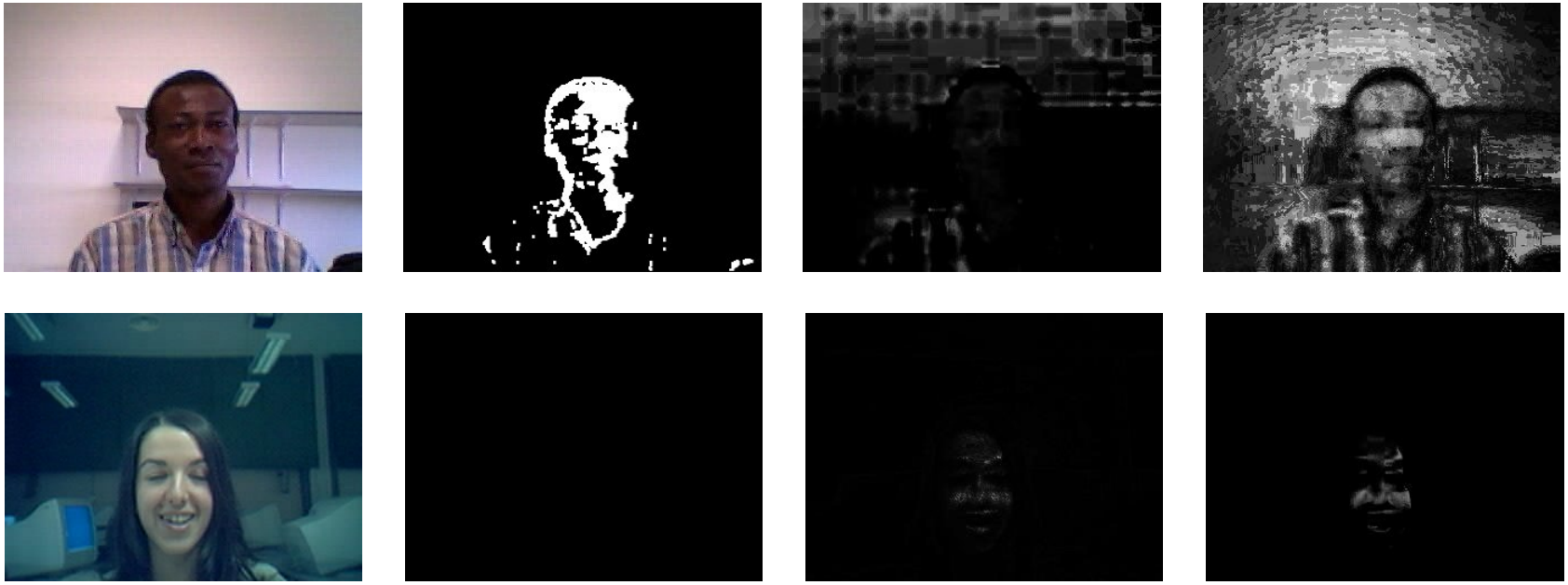
[Batallita]
Color piel
rojo-verde normalizado (R/(R+G+B) y G/(R+G+B))



M. Störring et al. Physics-based modelling of human skin colour under mixed illuminants. Robotics and Autonomous Systems, 2001.

Color

[Batallita]
Color piel
rojo-verde normalizado (R/(R+G+B) y G/(R+G+B))
70 individuos en condiciones diversas

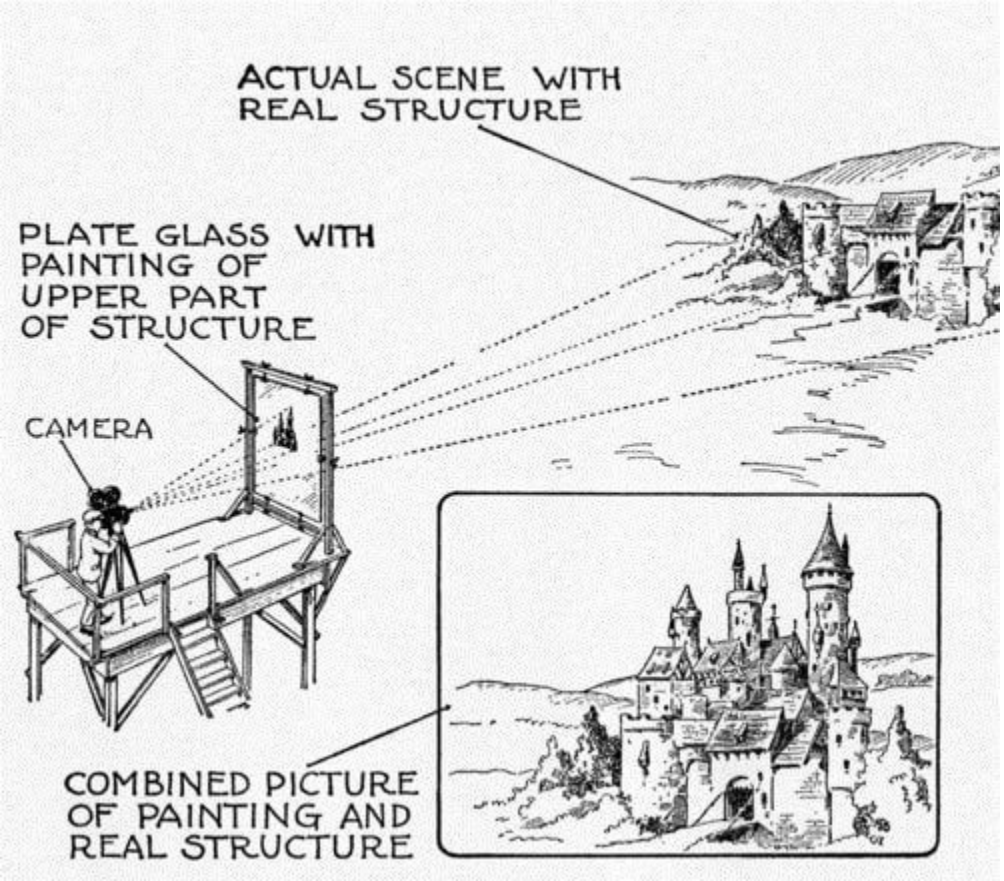


Esquema simple | Jones and Rehg [1] | Kruppa [2]

[1] Statistical Color Models with Application to Skin Detection (1999) Michael J. Jones and James M. Rehg. International Journal of Computer Vision
[2] H. Kruppa, M. A. Bauer, B. Schiele: Skin Patch Detection in Real-World Images. DAGM-Symposium 2002

Figura/fondo

Composición, croma, *matting*
Efectos especiales en el cine antes de los píxeles
Matting o *matte painting*, primeros tiempo sobre vidrio



Figura/fondo

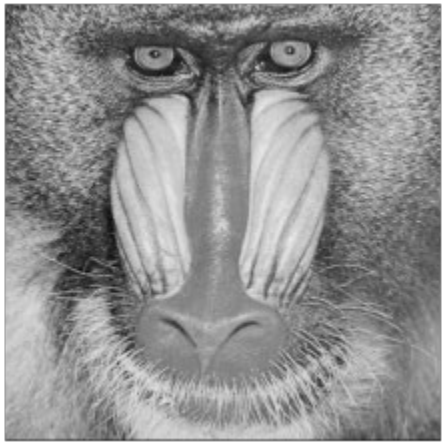
Actualidad: digital

Figura/fondo

Umbralizado
Asignar valor aplicando condicional

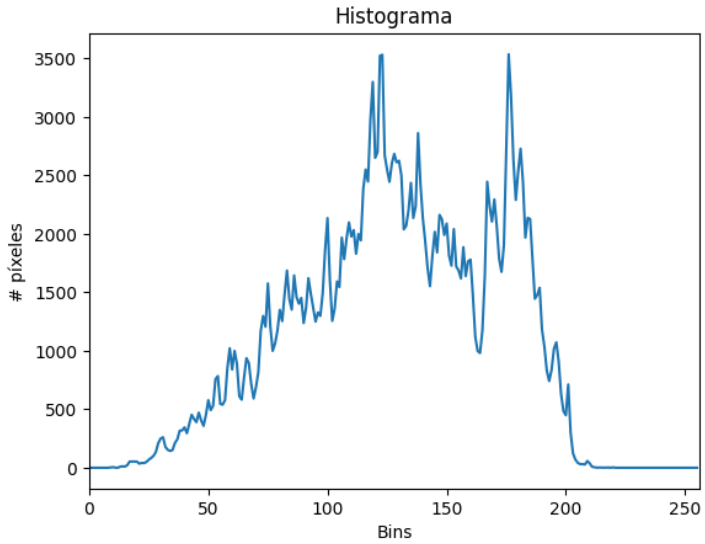
Figura/fondo

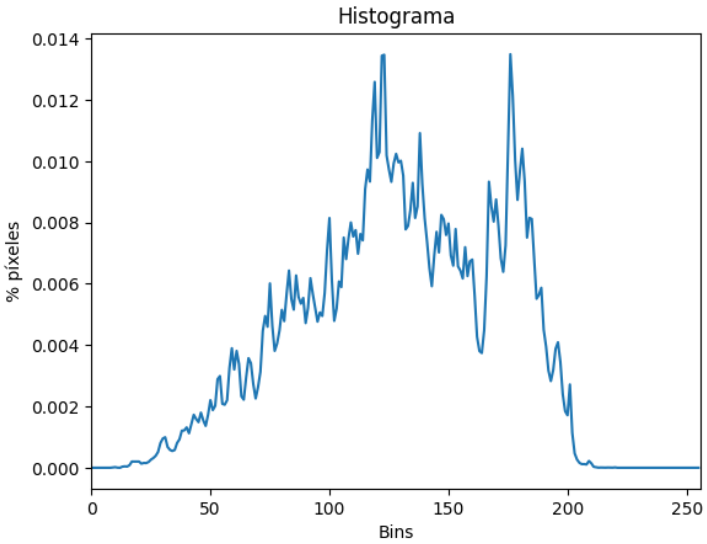
Recordando histograma
Distribución intensidades píxeles



Operador no dependiente de posición en imagen sino en histograma

$$g(i,j))=T(h(f(i,j)))$$



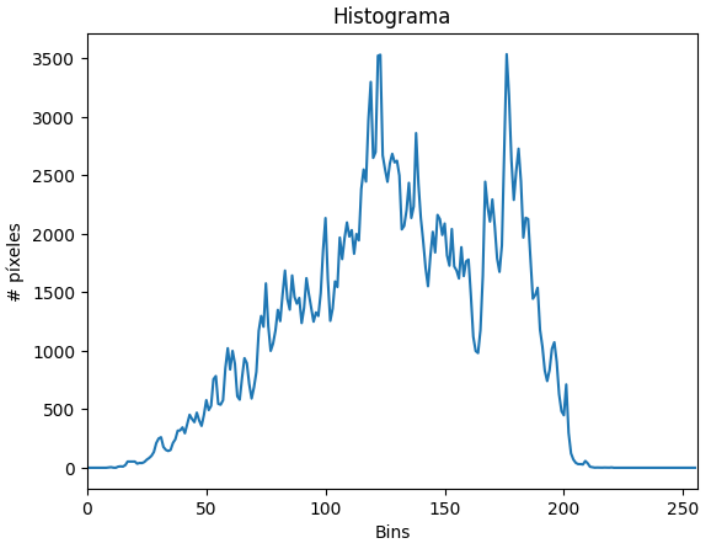


Figura/fondo

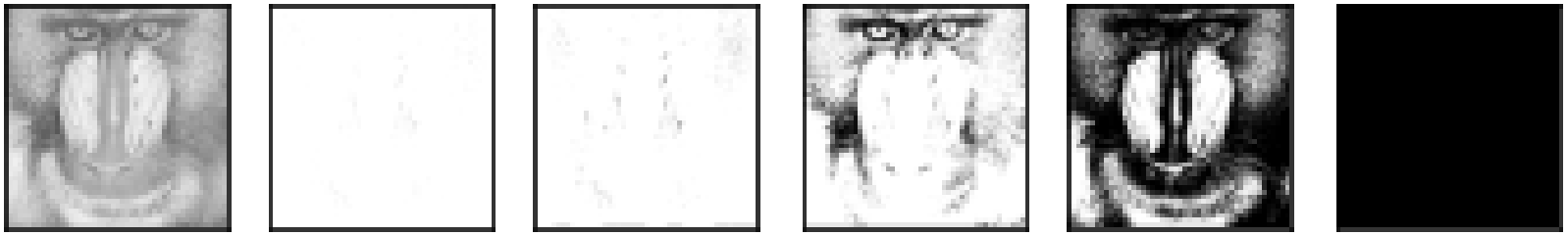
Umbralizado

Imagen binaria según umbral

$$1 \text{ si } f(i,j)>t$$



Variando el umbral

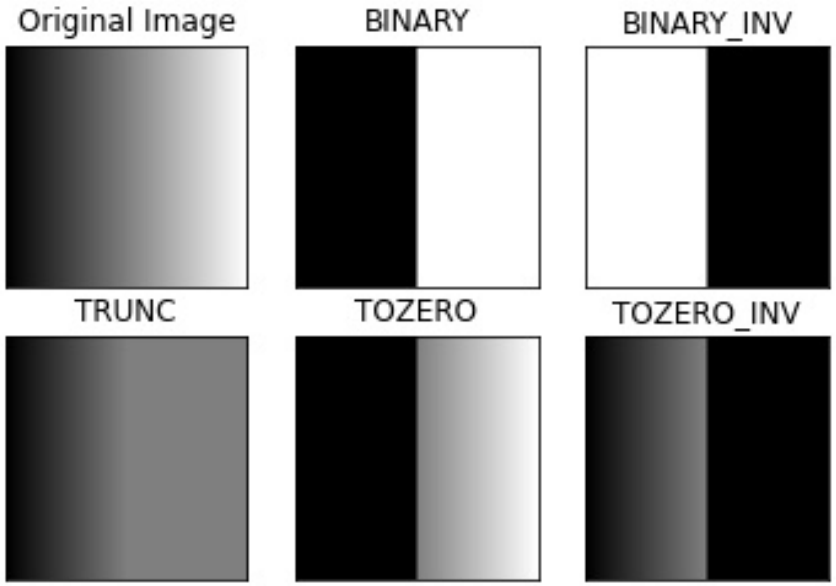


Figura/fondo

Umbralizado

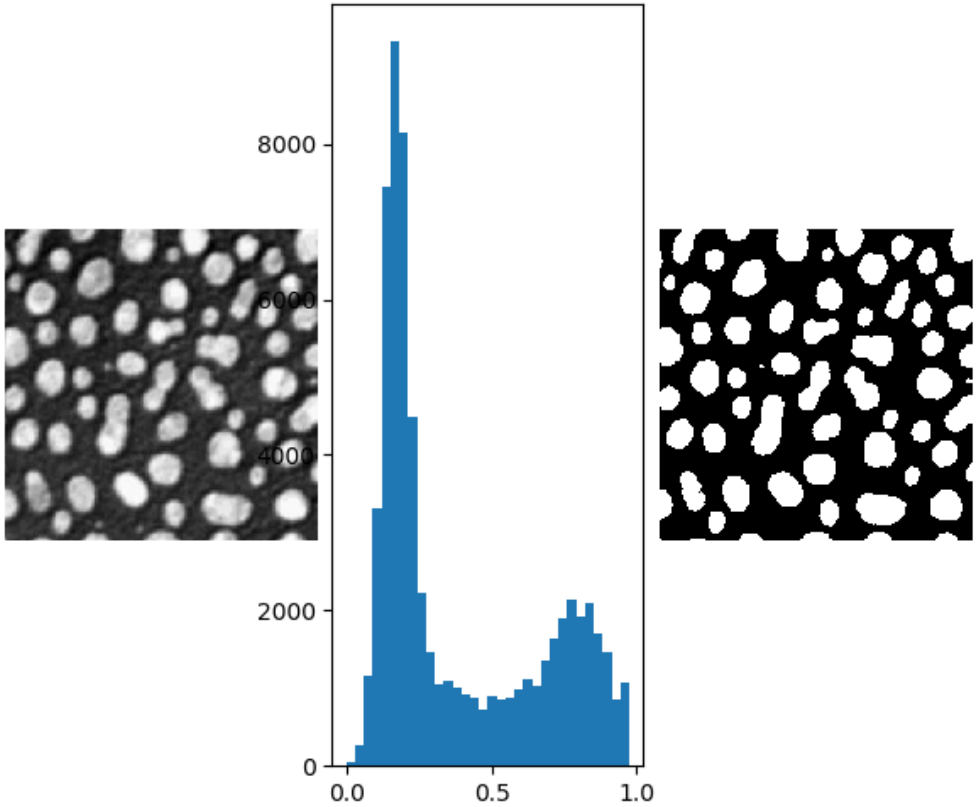
Opciones `cv2.threshold`

- **`cv.THRESH_BINARY`**
- **`cv.THRESH_BINARY_INV`**
- **`cv.THRESH_TRUNC`**
- **`cv.THRESH_TOZERO`**
- **`cv.THRESH_TOZERO_INV`**



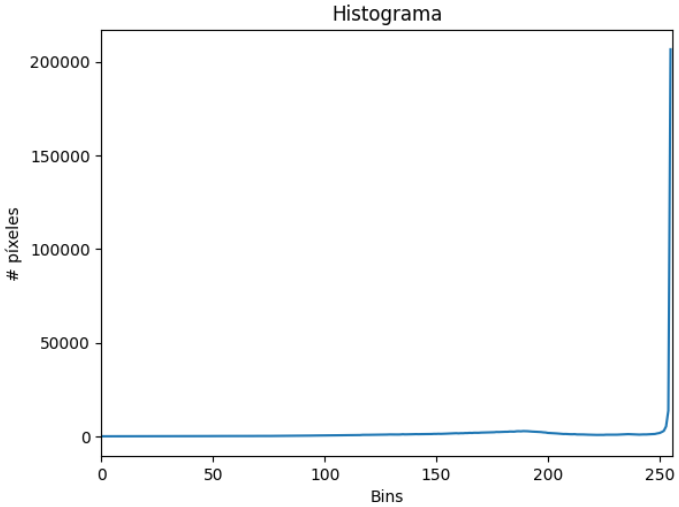
Figura/fondo

Separación de figura y fondo
Umbralizado para separar objetos del fondo



Figura/fondo

Separación de figura y fondo
Umbral fijo
Selección umbral automatizada (p.e. Otsu)



Figura/fondo

Separación de figura y fondo

Fijo vs. Otsu

```
# Umbralizado invertido, se asume objetos en blanco
-,imth1 = cv2.threshold(gris,200,255,cv2.THRESH_BINARY_INV)
# Umbral con método de Otsu
-,imth2 = cv2.threshold(gris,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

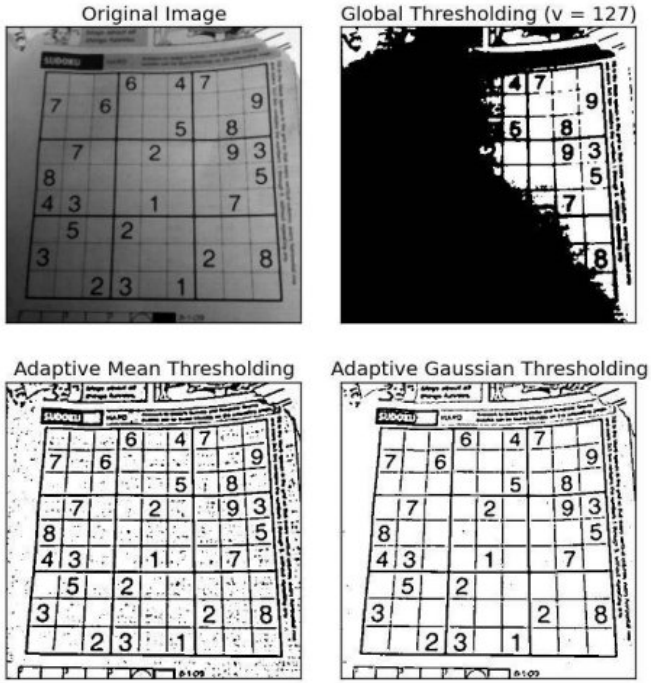
Figura/fondo

Separación de figura y fondo

Fijo

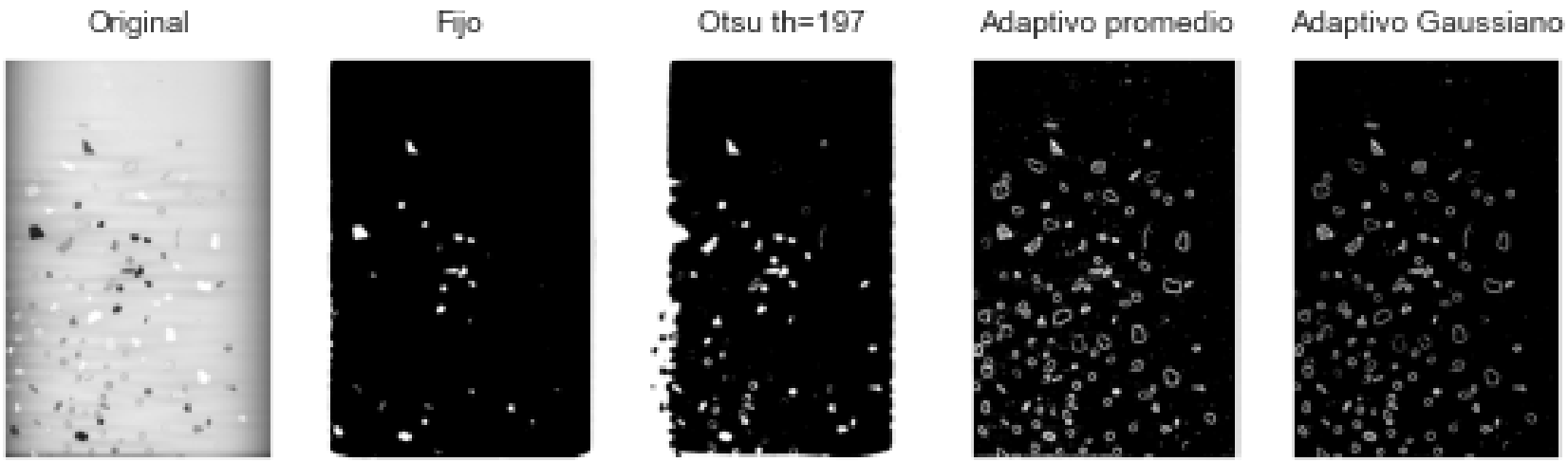
Adaptativo por vecindad, opciones `cv.adaptiveThreshold`

- **`cv.ADAPTIVE_THRESH_MEAN_C`**
- **`cv.ADAPTIVE_THRESH_GAUSSIAN_C`**



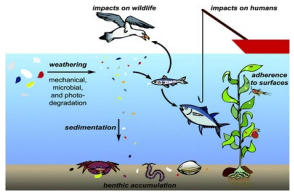
Figura/fondo

Separación de figura y fondo
Comparativa con OpenCV



Figura/fondo

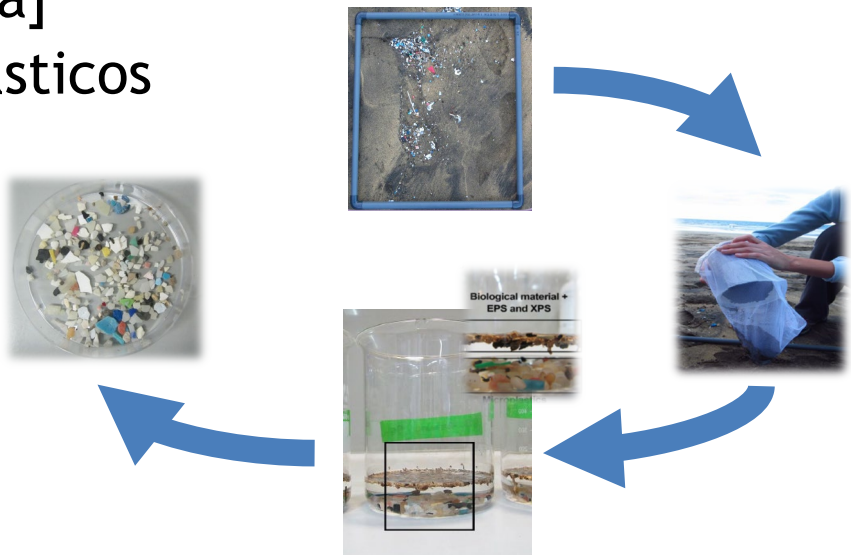
[Batallita]
Microplásticos



J. Lorenzo et al. *SMACC: A System for Microplastics Automatic Counting and Classification* *IEEE Access 2020*
J. Lorenzo et al. *Deep learning approach for automatic microplastics counting and classification* STOTEN 2021

Figura/fondo

[Batallita]
Microplásticos



Fragmento

Línea

Orgánico

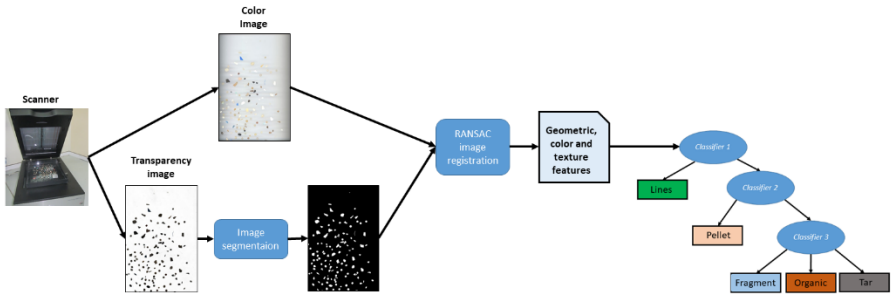
Pellet

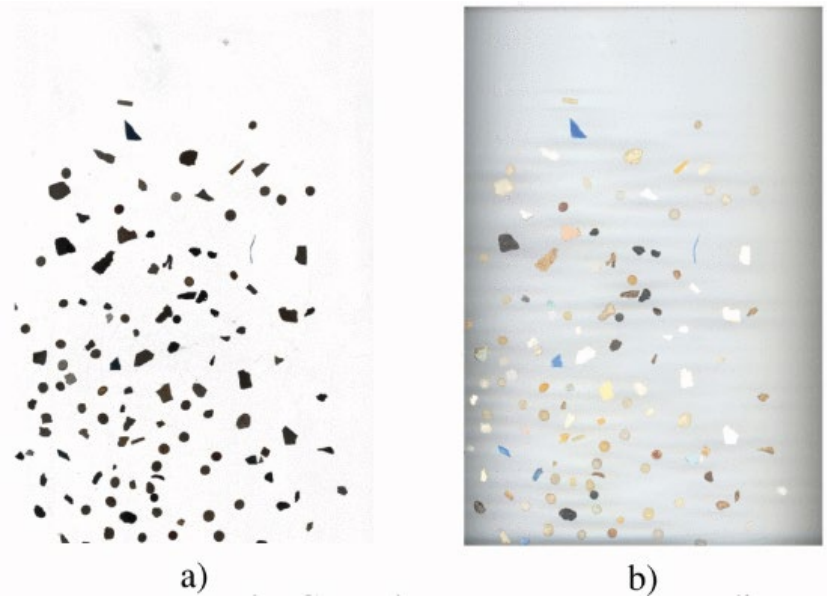
Alquitrán

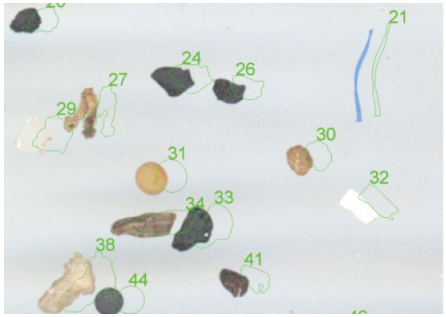
J. Lorenzo et al. *SMACC: A System for Microplastics Automatic Counting and Classification* *IEEE Access 2020*
J. Lorenzo et al. *Deep learning approach for automatic microplastics counting and classification* STOTEN 2021

Figura/fondo

[Batallita]
Microplásticos







J. Lorenzo et al. *SMACC: A System for Microplastics Automatic Counting and Classification* *IEEE Access 2020*
J. Lorenzo et al. *Deep learning approach for automatic microplastics counting and classification* STOTEN 2021

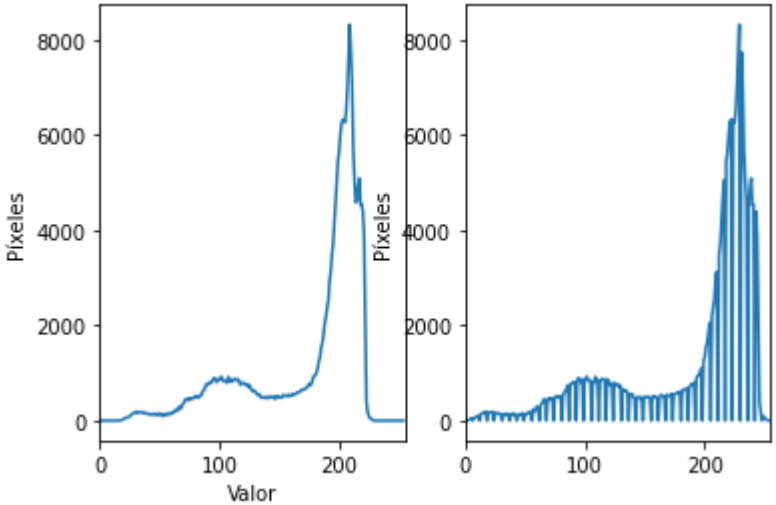
Figura/fondo

[Batallita]
Actualmente otros esquemas y escenarios

Contraste

Ajustar al rango máximo

$$g(i,j) = \frac{f(i,j) - f_{min}}{f_{max} - f_{min}}$$



Lo aplica `matplotlib.imshow` al escoger `cmap='gray'`

Contraste

Ecualizado del histograma

Preprocesamiento habitual

Mejora contraste

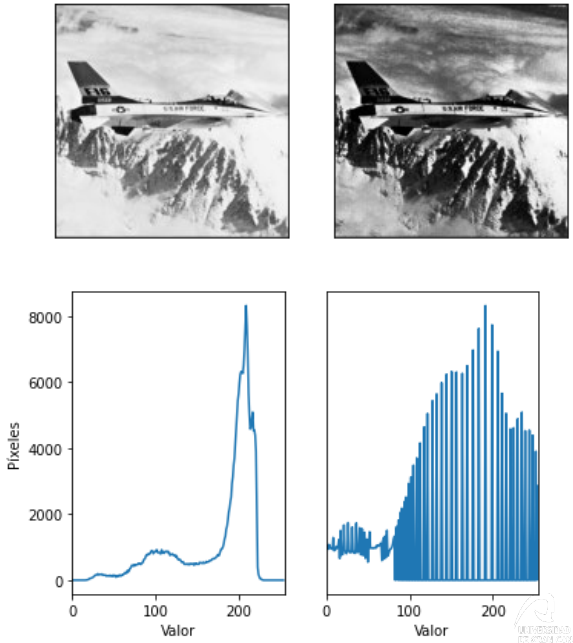
Pretende invarianza frente a cambios iluminación

Mapear la intensidad (0<=I<=255) para obtener histograma plano

Histograma, *h(I)*, de imagen de N píxeles, integra la distribución

$$c(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = c(I-1) + \frac{1}{N}h(I)$$

Obtiene el percentil, *f(I)=c(I)*



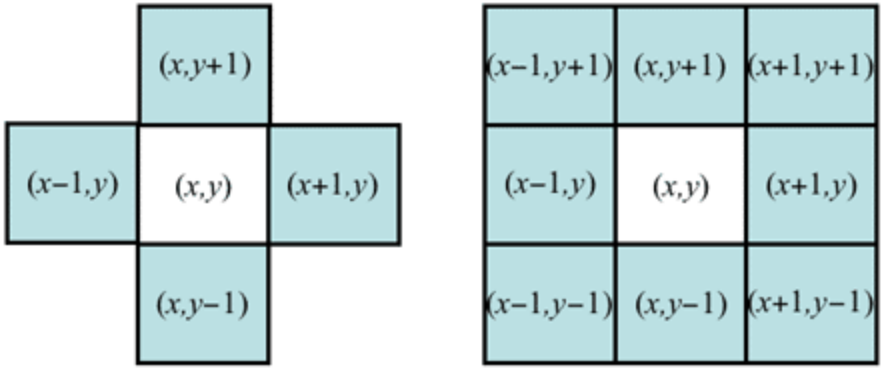
Filtros

Ecualizado del histograma adaptativo usa vecindad
Operadores locales o de vecindad: filtros

Gran importancia:

- Mejora de imagen: ruido, contraste, redimensión, etc.
- Extracción de información: textura, bordes, esquinas, etc.
- Detección de patrones

Filtrado lineal, suma ponderada de vecinos



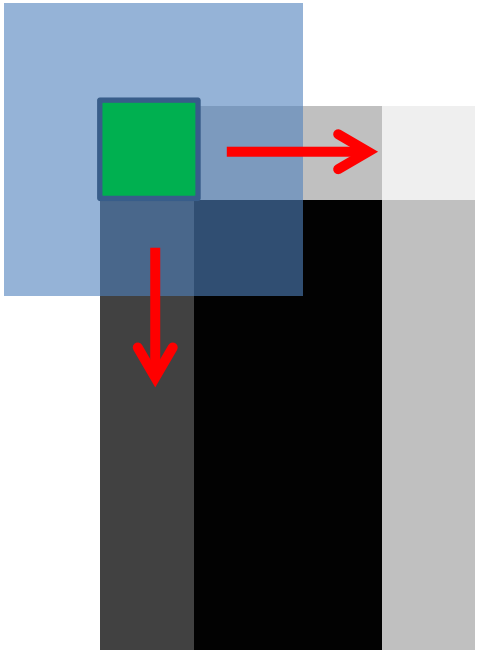
4-neighbourhood 8-neighbourhood

Filtros

Filtrado lineal
Función calculada en vecindad con pesos fijos

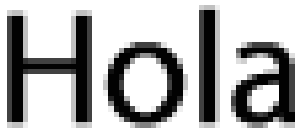
$$g(i,j) = \sum_{n,m} k(n,m) f(i+n, j+m)$$

k kernel o *impulse response*
k se desplaza sobre imagen
Respuesta para cada píxel

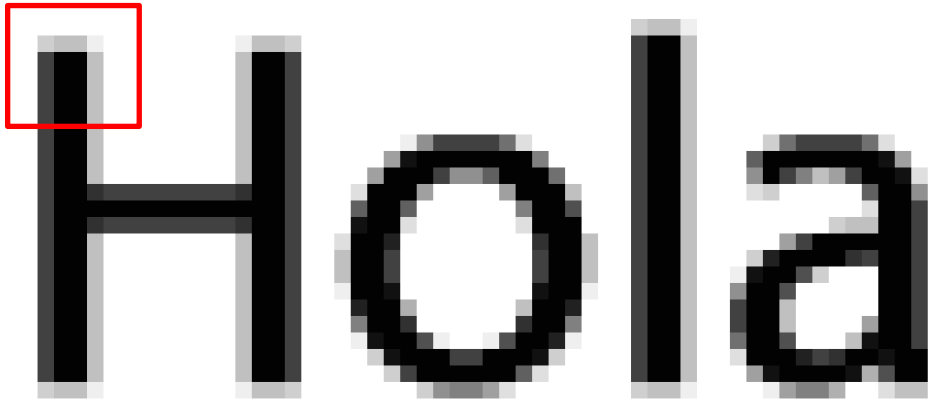


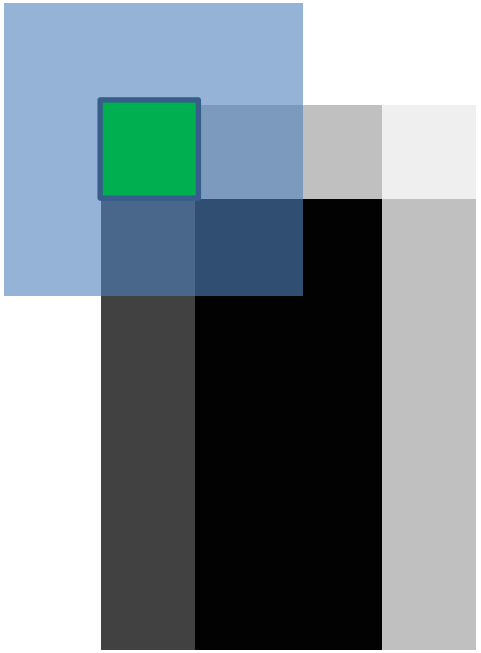
Filtros

Vecindad
Ejemplo 8 vecinos



Zoom





Filtros

Tonos de gris

$$f(x, y)$$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

Filtros

Filtro de caja

$$g(i,j) = \sum_{n,m} k(n,m) f(i+n, j+m)$$

$$n, m = [-1,0,1]$$

Kernel o máscara de convolución del filtro

$k(n,m)$

$^1/_9$

1	1	1
1	1	1
1	1	1

Filtros

$k(n,m)$

$1/9$

1	1	1
1	1	1
1	1	1

$g(x,y)$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

$f(x,y)$

Filtros

$k(n,m)$

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

$$g(1,1) = \sum_{n,m} k(n,m) f(1+n, 1+m) \quad \text{donde} \quad n,m=[-1,0,1]$$

$g(x,y)$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

$f(x,y)$

Filtros

$k(n,m)$

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

$$g(1,1) = \sum_{n,m} k(n,m) f(1+n, 1+m) \quad \text{donde} \quad n,m=[-1,0,1]$$

$g(x,y)$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

$f(x,y)$

	194					

Filtros

$k(n,m)$

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

$$g(1,2) = \sum_{n,m} k(n,m) f(1+n, 2+m) \quad \text{donde} \quad n,m=[-1,0,1]$$

$g(x,y)$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

$f(x,y)$

	193	158				

Filtros

$k(n,m)$

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

$$g(1,3) = \sum_{n,m} k(n,m) f(1+n, 3+m) \quad \text{donde} \quad n,m=[-1,0,1]$$

$g(x,y)$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

$f(x,y)$

	193	158	176			

Filtros

$k(n,m)$

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

$$g(i,j) = \sum_{n,m} k(n,m) f(i+n, j+m) \quad \text{donde} \quad n,m=[-1,0,1]$$

$g(x,y)$

255	255	255	255	255	255	255
255	207	192	192	239	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255
255	65	2	2	192	255	255

$f(x,y)$

	193	158	176	211	246	
	144	107	112	176	239	
	107	23	65	150	234	
	107	23	65	150	234	
	107	23	65	150	234	

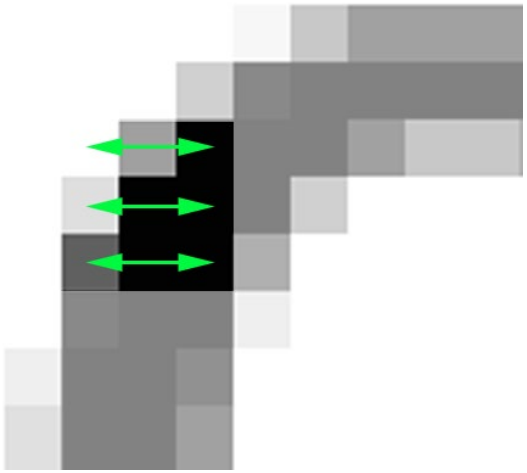
$k(n,m)$

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

Filtros

$$g(i,j) = \sum_{n,m} k(n,m) f(i+n, j+m)$$ donde n,m=[-1,0,1]

¿Por qué los pesos suman 1?
Promedio vecindad
El resultado suaviza los contornos



Filtros

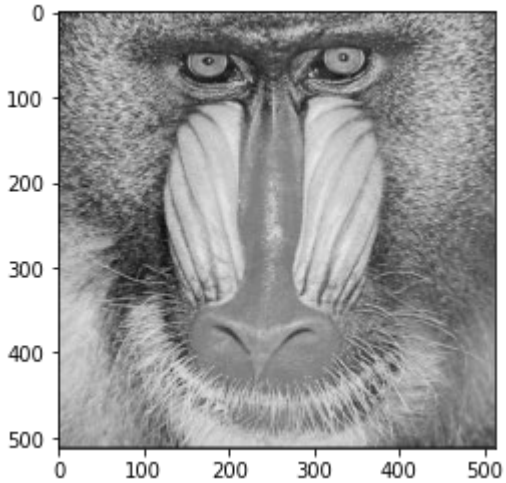
¿Qué pasaría al aplicar?

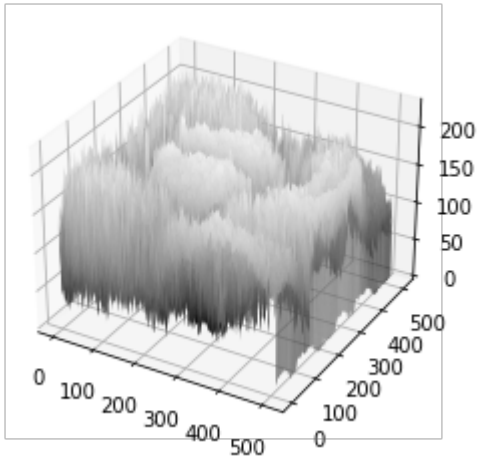
0	0	0
0	1	0
0	0	0

0	0	0
0	0	1
0	0	0

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

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





Filtros

Posibilidades definiendo coeficientes del *kernel*:

- Suavizado
- Bordes
- Eliminar ruido
- Mejora de la imagen
- Restauración
- Redimensionar
- Dilatación
- Binaria
- Función distancia
- …

Original Automatic Enhancement

Filtros

[Batallita]
RUNNERS/ILUSOS

O. Santana et al. Evaluating the Impact of Low-light Image Enhancement Methods on Runner Re-identification in the Wild. ICPRAM23
O. Santana et al. Applying deep learning image enhancement methods to improve person re-identification. Neurocomputing, 2024

Filtros

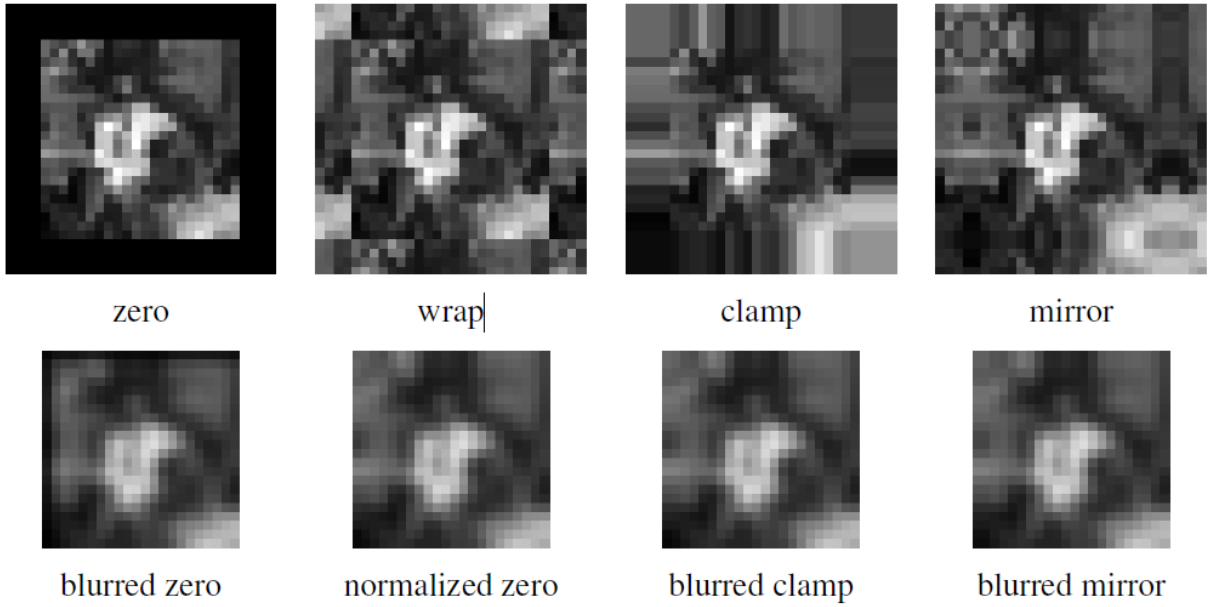
[Batallita]
RUNNERS/ILUSOS

O. Santana et al. Evaluating the Impact of Low-light Image Enhancement Methods on Runner Re-identification in the Wild. ICPRAM23
O. Santana et al. Applying deep learning image enhancement methods to improve person re-identification. Neurocomputing, 2024

Filtros

Efectos de borde (*padding*)

- Cero
- Constante
- Reflejo
- Repetición



Szeliski

Filtros

Dado *k*

Correlación 2D $g = f \otimes k$

$$g(i,j) = \sum_{n=-h}^{h} \sum_{m=-h}^{h} k(n,m) f(i+n, j+m)$$

```
h = scipy.signal.correlate2d(k,f)
```

Similitud entre *k* y la zona de la imagen
Mayor valor, más similar o mayor encaje

Filtros

Dado *k*

Convolución 2D $g = f * k$

$$g(i,j) = \sum_{n=-h}^{h} \sum_{m=-h}^{h} k(n,m) f(i-n, j-m)$$

```
h = scipy.signal.convolve2d(k,f)
```

Refleja *k* antes de correlar (cambio de signo)
Efecto de una señal sobre otra, combina en una tercera

Filtros

Dado *k*

Correlación 2D $g = f \otimes k$

$$g(i,j) = \sum_{n=-h}^{h} \sum_{m=-h}^{h} k(n,m) f(i+n, j+m)$$

Convolución 2D $g = f * k$

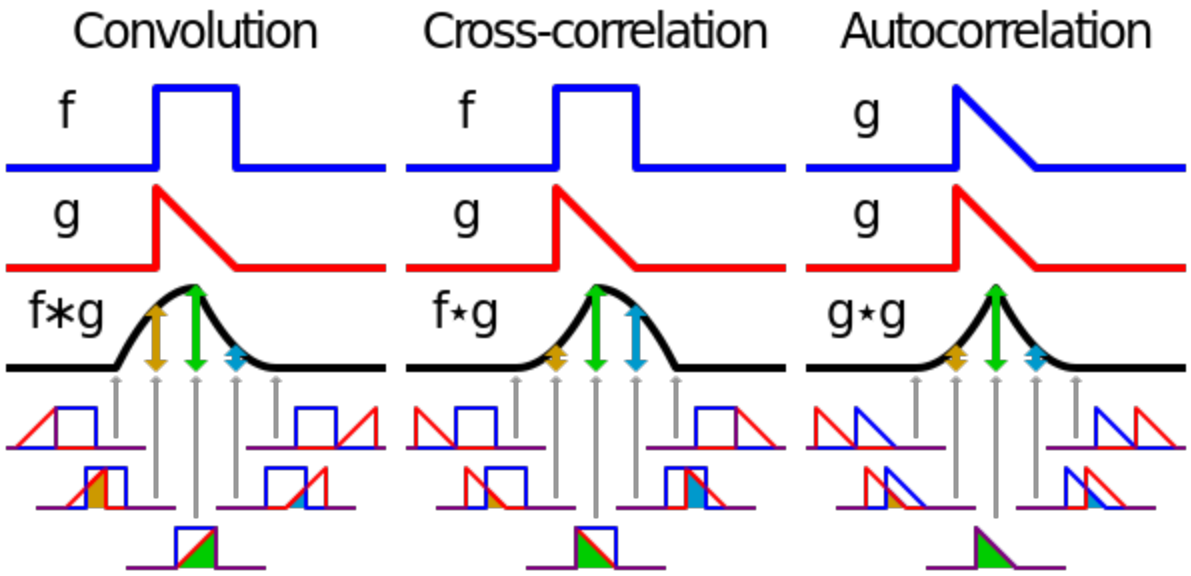
$$g(i,j) = \sum_{n=-h}^{h} \sum_{m=-h}^{h} k(n,m) f(i-n, j-m)$$

Similares, la convolución refleja *k* en *x* e *y* antes de operar

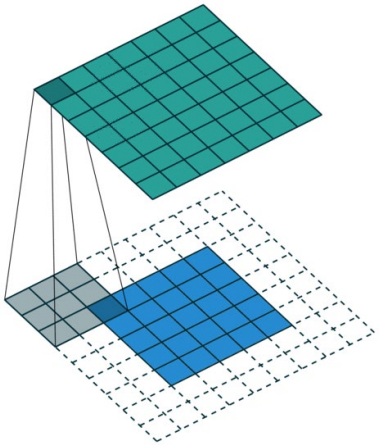
Equivalentes si *k* es simétrico (geométricamente)

Filtros

Correlación vs. convolución



Filtros

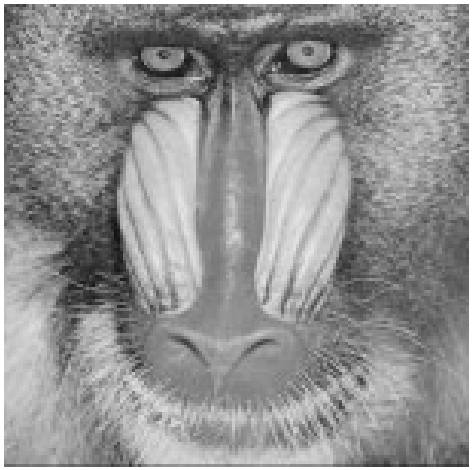


Convolución 2D $g = f * k$

```
k = [[-1, -2, -1], [0,0,0], [1,2,1]]

g_conv = convolve2d(gris, kernel, mode="same", boundary="symm")
```

Original



Convolución

Filtros

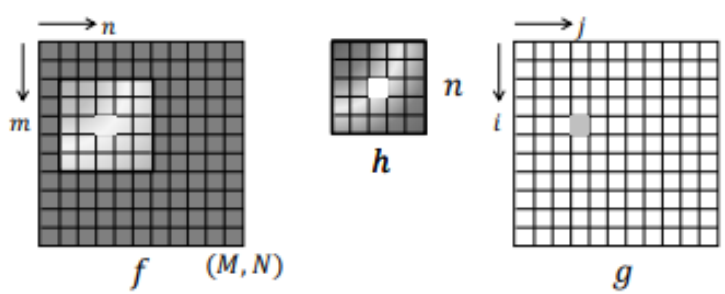
Propiedades convolución:

- Conmutativa $a * b = b * a$
- Asociativa (**no la correlación**) $a * (b * c) = (a * b) * c$
- Identidad 1D $e = [0, 0, 1, 0, 0]$, $a * e = a$

Simplifican operaciones, permite combinar filtros

$$f[i,j] \longrightarrow \boxed{h[i,j]} \longrightarrow g[i,j]$$

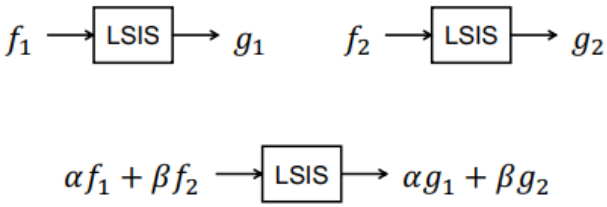
$$g[i,j] = \sum_{m=1}^{M} \sum_{n=1}^{N} f[m,n] \underbrace{h[i-m, j-n]}_{\text{"Mask," "Kernel," "Filter"}} \qquad \boxed{1}$$



S. Nayar

Filtros

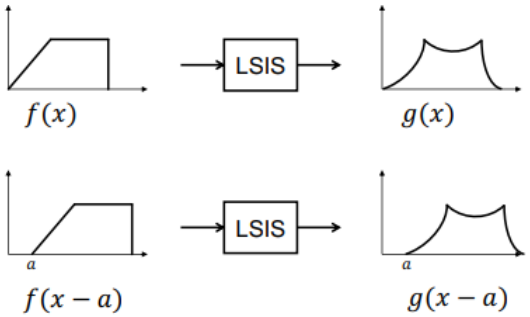
LSIS (*linear-shift invariant system*)
Convolución/correlación
Principio de superposición e invarianza al desplazamiento



Combinaciones de entradas equivalentes a combinaciones en la salida

$\alpha * f_1(i,j) + \beta * f_2(i,j) \qquad \alpha * g_1(i,j) + \beta * g_2(i,j)$

P.e., desplazar la entrada, equivale a desplazar la salida:



$f(x,y)+v \qquad g(x,y)+v$

S. Nayar

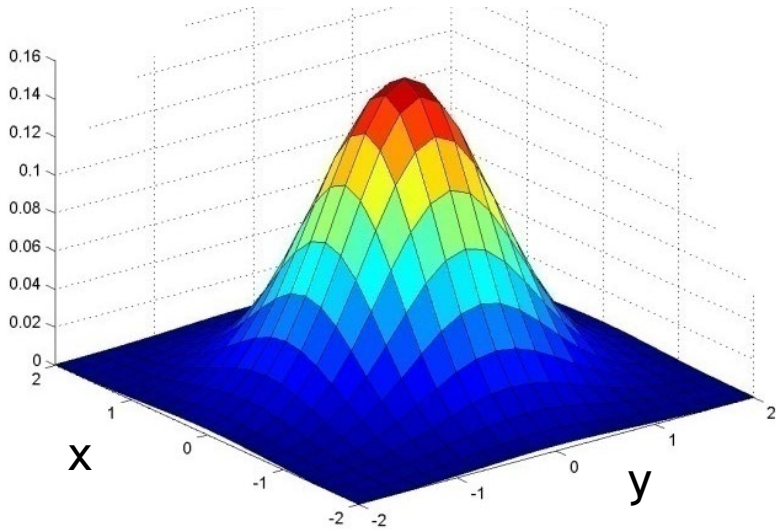
Filtros

Gaussiano

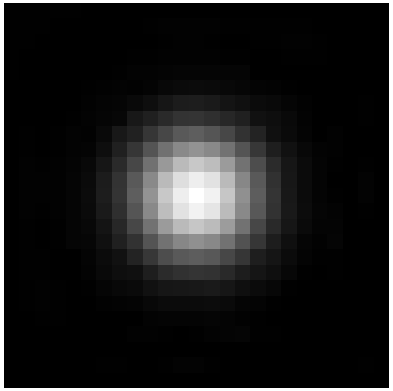
Pesado por cercanía

Control de forma con σ

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



Vista superior



y

x

Filtros

Suavizado (`cv2.filter2D cv2.GaussianBlur`)

```
[[0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]]
```

```
[[0.00390625 0.015625   0.0234375  0.015625   0.00390625]
 [0.015625   0.0625     0.09375    0.0625     0.015625  ]
 [0.0234375  0.09375    0.140625   0.09375    0.0234375 ]
 [0.015625   0.0625     0.09375    0.0625     0.015625  ]
 [0.00390625 0.015625   0.0234375  0.015625   0.00390625]]
```

Original

Filtro de caja 5x5

Gaussiana 5x5

Filtros

Gaussiano
Elimina altas frecuencias, suaviza

Aplicar dos veces G_σ equivale a aplicar una vez $G_{\sigma/2}$
Expresable como producto de dos funciones, en *x* y en *y*

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}}$$

Es un **filtro separable**

Filtros

Separables

Kernel n*n, convolución n^2 operaciones por píxel
Componentes horizontal y vertical por separado
Acelera, son 2*n operaciones

$$K = v\,h^T$$

$$K = \sum \sigma_i u_i v_i^T$$

¿Cómo saber si lo es?
El filtro 2D puede convertirse en el producto de filtros 1D

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Filtros

Separables

Patrón [1 2 1] conocido como binomial

Aproxima la Gaussiana en aritmética entera

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

(a) box, $K = 5$ (b) bilinear (c) “Gaussian” (d) Sobel (e) corner

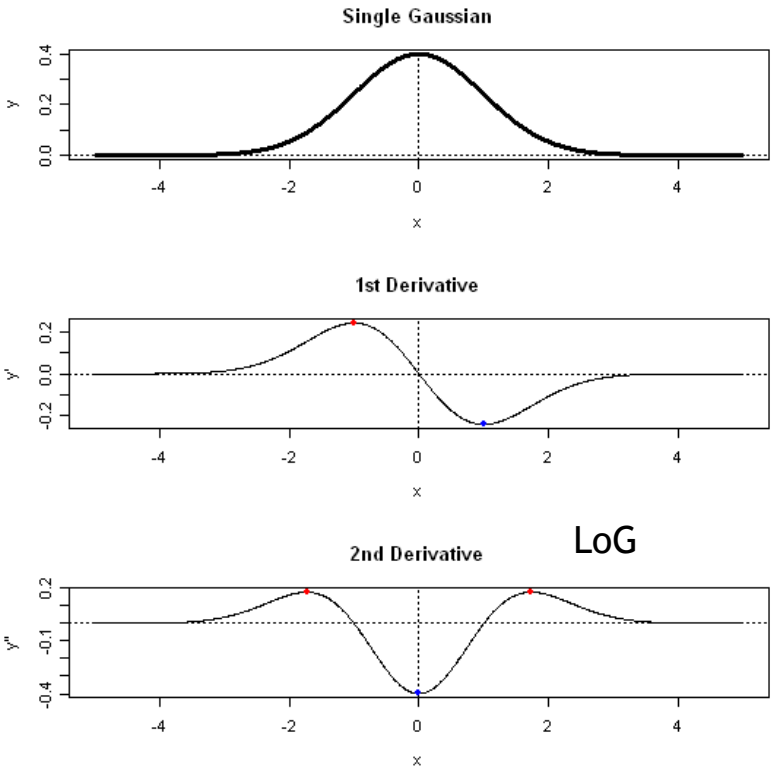
Szeliski

Filtros

Filtros más complejos aplican primero Gaussiana
Luego calculan primera o segunda derivada
Denominados paso banda, quitan bajas y/o altas frecuencias

Ejemplos:

- Sobel
- Laplaciana



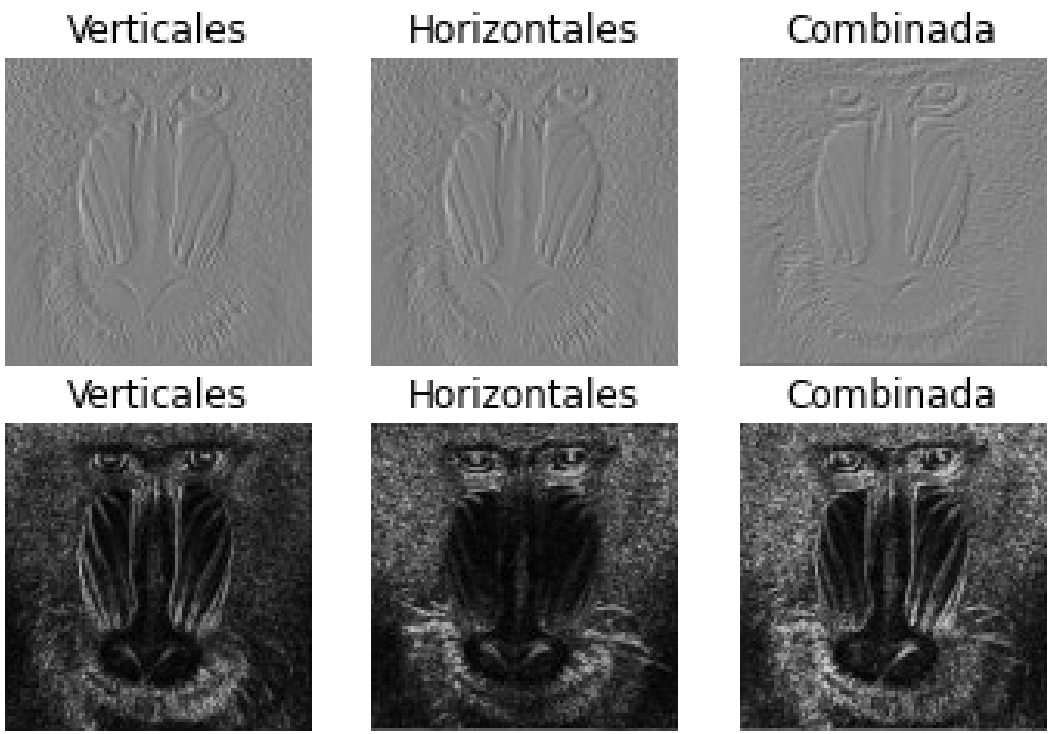
Filtros

$k(k,l)$

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

Sobel

Aproxima filtro direccional
Gaussiana previa y luego filtro direccional
Resultado con posibles valores negativos
Desplazamiento y ajuste para visualizar



Filtros

$k(k,l)$

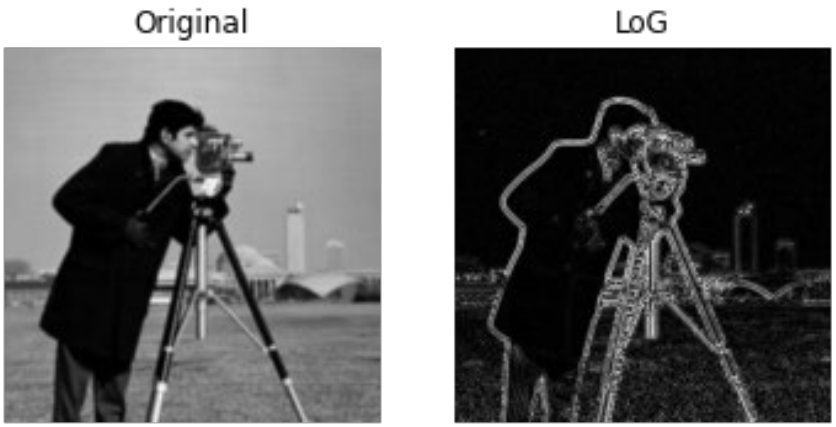
0	1	0
1	-4	1
0	1	0

Laplaciana

Segunda derivada

Gaussiana previa, se denomina filtro LoG

Mayor respuesta a bordes relevantes localmente



(a) (b) (c)

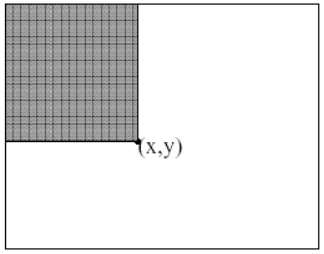
Szeliski

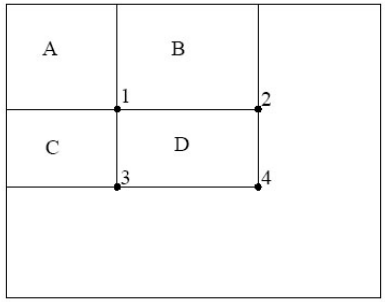
Filtros

Imagen integral

Convolución con filtros de caja de forma repetitiva
Precalcula la tabla de áreas sumadas (*summed area table*)
Cómputo eficiente (línea de barrido)

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





31	2	4	33	5	36
12	26	9	10	29	25
13	17	21	22	20	18
24	23	15	16	14	19
30	8	28	27	11	7
1	35	34	3	32	6

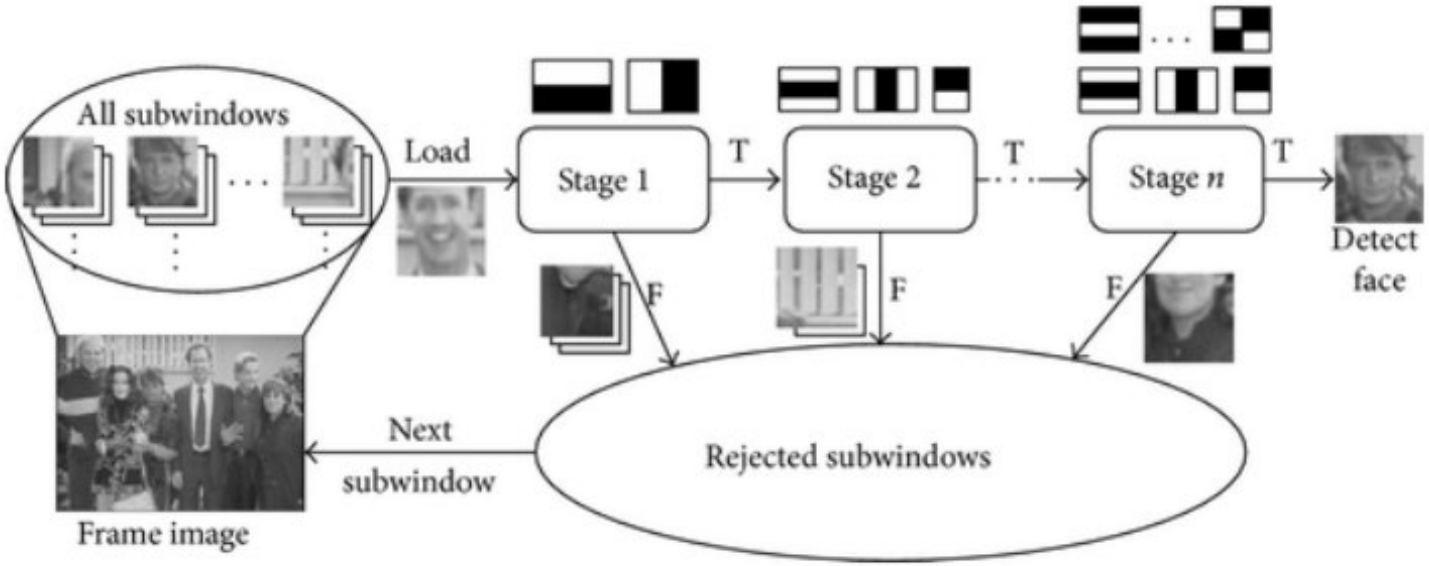
31	33	37	70	75	111
43	71	84	127	161	222
56	101	135	200	254	333
80	148	197	278	346	444
110	186	263	371	450	555
111	222	333	444	555	666

15 + 16 + 14 + 28 + 27 + 11 =
101 + 450 – 254 – 186 = 111

Filtros

Imagen integral

Detector de Viola Jones (2001)



Cascade structure for Haar classifiers

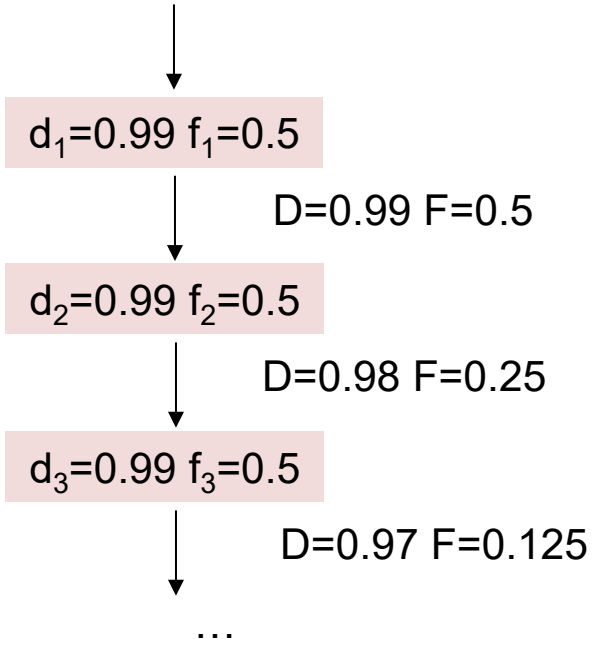
Filtros

Imagen integral

Detector de Viola Jones

Ventana deslizante

Cascada de clasificadores “débiles”



Filtros no lineales

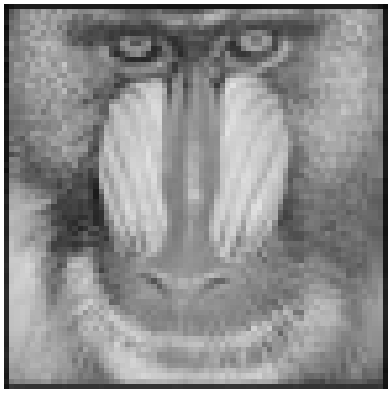
Mediana

No convolutivo

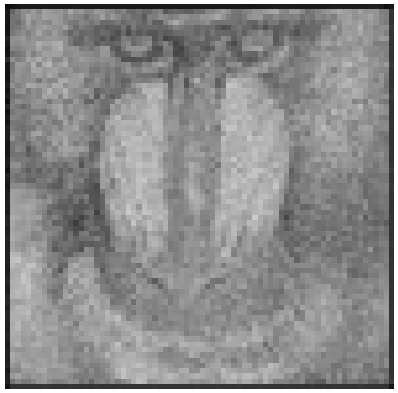
Mejor respuesta con ruido *sal y pimienta* que suavizado

Ordena valores vecindad del píxel

Asigna el valor central de lista ordenada

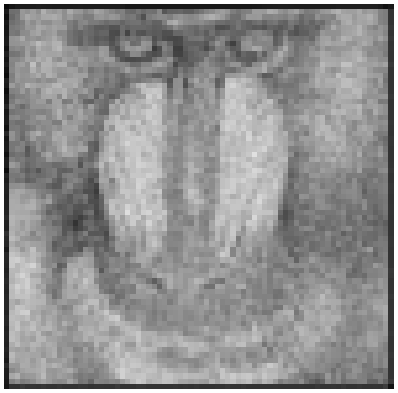


Original



Ruido

Mediana



Promedio 5x5

Filtros no lineales

Ruido sal y pimienta

Los 9 valores ordenados del píxel destacado

{2, 2, 2, 2, 2, 2, 2, 2, 255}

$f(x, y)$

255	255	255	255	255	255	255
255	2	2	2	2	255	255
255	2	2	2	2	255	255
255	2	2	2	2	255	255
255	2	255	2	2	255	255
255	2	2	2	2	255	255
255	2	2	2	2	255	255

$g(x, y)$

	255	2	2	255	255	
	2	2	2	2		
	2	2	2	2		
	2	2	2	2		
	2	2	2	2		

Filtros no lineales

Ruido sal y pimienta

Los 9 valores ordenados del píxel destacado

{2, 2, 2, 2, 2, 2, 2, 2, 255}

$f(x, y)$

255	255	255	255	255	255	255
255	2	2	2	2	255	255
255	2	2	2	2	255	255
255	2	2	2	2	255	255
255	2	255	2	2	255	255
255	2	2	2	2	255	255
255	2	2	2	2	255	255

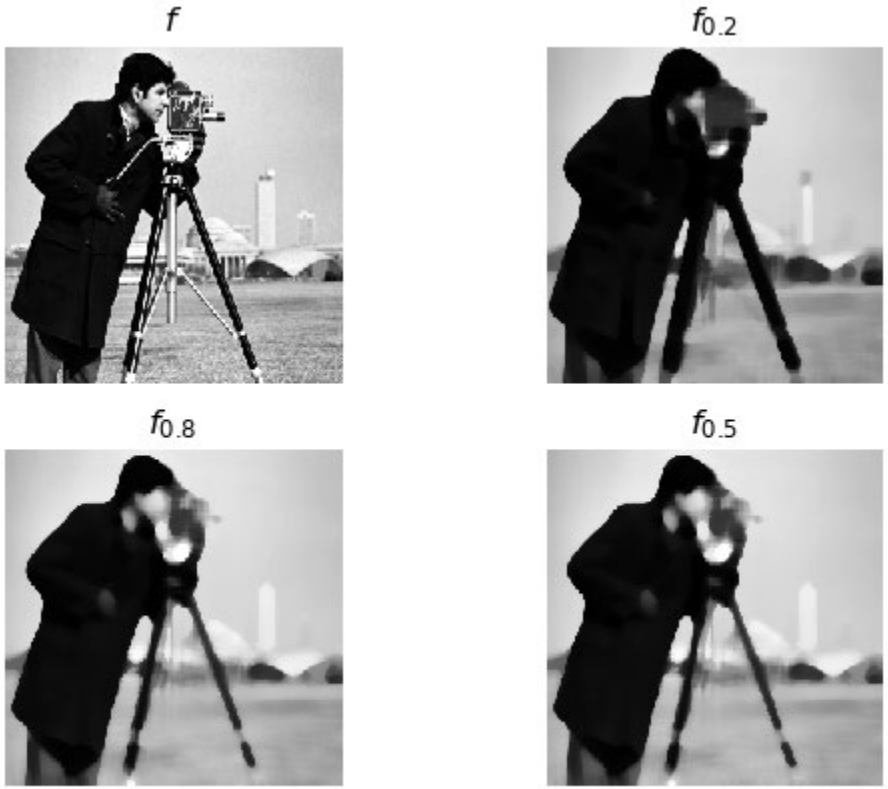
$g(x, y)$

	255	2	2	255	255	
	2	2	2	2		
	2	2	2	2		
	2	2	2	2		
	2	2	2	2		

Filtros no lineales

Filtrado por percentil

Generaliza la mediana (percentil 0.5)
Ordena vecindad y toma el valor de la posición del percentil

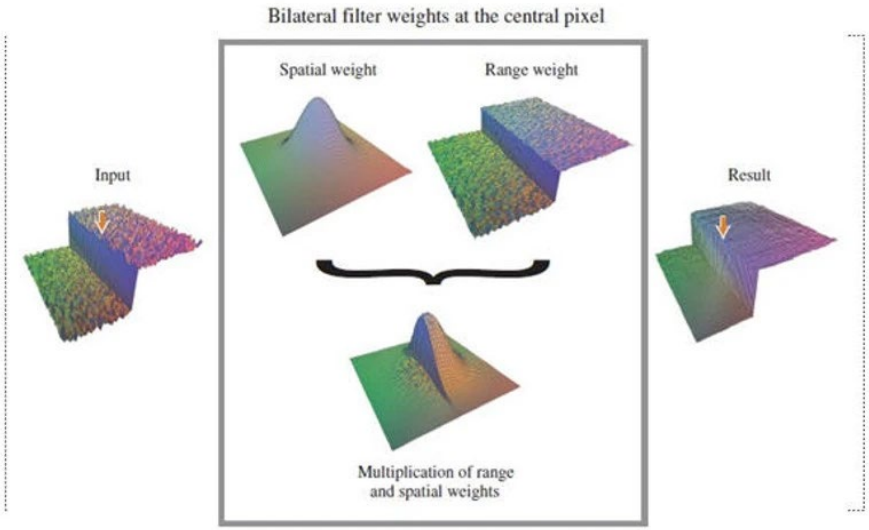


R. van den Boomgard

Filtros no lineales

Bilateral

Mantener bordes, eliminar ruido
Gaussiana pesa por distancias
Suaviza bordes
Eliminar valores que difieren del píxel central (*outliers*)
Bilateral pesa por diferencias del tono del píxel
Preserva bordes, suaviza zonas planas



Filtros no lineales

Bilateral

Original | Gaussian $\sigma_s = 2$ | Bilateral $\sigma_s = 2, \sigma_b = 10$

S. Nayar

Imagen binaria

Filtros no lineales frecuentes sobre imágenes binarias

Aplican operaciones de umbralizado

$$\theta(f,t) = \begin{cases} 1 & si \;\; f > t \\ 0 & resto \end{cases}$$

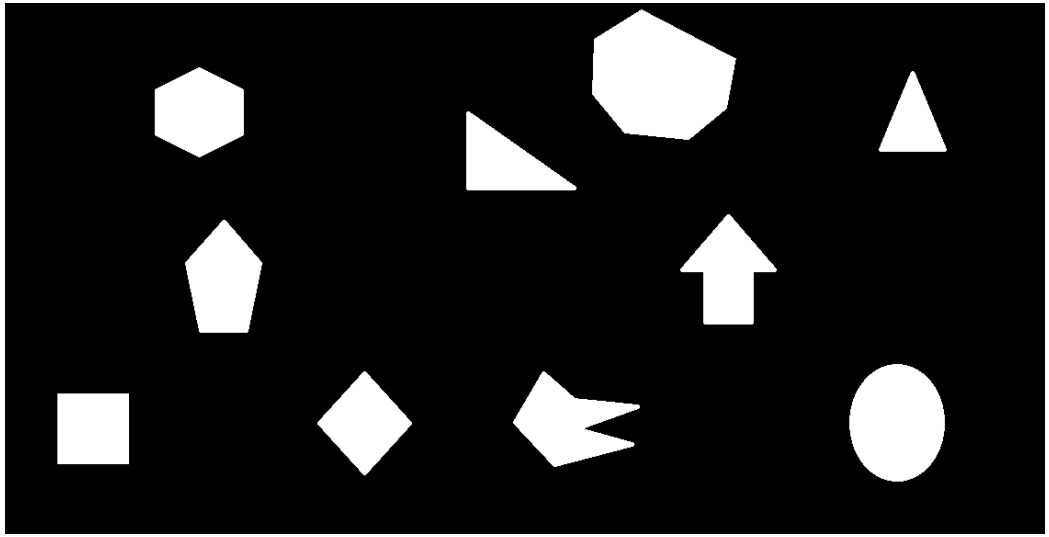


Imagen binaria

Operaciones **morfológicas**

Habituales, cambian la forma

Convolución de imagen con elemento estructural *s* (SxS)

$$c = f \otimes s$$

Umbraliza *c* (contar número de 1) con segundo parámetro

b) Dilatación (*dilate*) $\theta(c\ ,\ 1)$

c) Erosión (*erode*) $\theta(c\ ,\ S)$

d) Mayoría (*maj*) $\theta(c\ ,\ S/2)$

e) Apertura (*open*) $dilate(erode(f,S),S)$

f) Cierre (*close*) $erode(dilate(f,S),S)$

(a) (b) (c) (d) (e) (f)

Szeliski

Imagen binaria

Transformada distancia

Distancia a conjunto de puntos en imagen binaria

Definida una distancia entre píxeles (p.e. euclídea, Manhattan)

$$d_M(k,l) = |k| + |l| \qquad d_e(k,l) = \sqrt{k^2 + l^2}$$

La transformada distancia es el mínimo entre el píxel y fondo

$$D(i,j) = \min_{k,l:b(k,l)=0} d(i-k, j-l)$$

Imagen binaria

Transformada distancia

0	0	0	0	1	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

(a)

0	0	0	0	1	0	0
0	0	1	1	2	0	0
0	1	2	2	3	1	0
0	1	2	3			

(b)

0	0	0	0	1	0	0
0	0	1	1	2	0	0
0	1	2	2	3	1	0
0	1	2	2	1	1	0
0	1	2	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

(c)

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

(d)

(e)

(g)

Szeliski

El valor de la transformada crece al alejarse del fondo

Crea “esqueleto” de la forma donde se calcula

Píxeles equidistantes a dos bordes

Imagen binaria

Componentes conectadas

Tras segmentar

Regiones de píxeles adyacentes (horizontal o verticalmente)

Estadísticos para caracterizar:

- Área (M_{00})
- Perímetro
- Centroide

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad \bar{y} = \frac{M_{01}}{M_{00}}$$

- Ejes mayor y menor
- Circularidad E_{min}/E_{max}
- Momentos

$$M_{ij} = \sum_x \sum_y x^i y^j f(x,y)$$

(f)

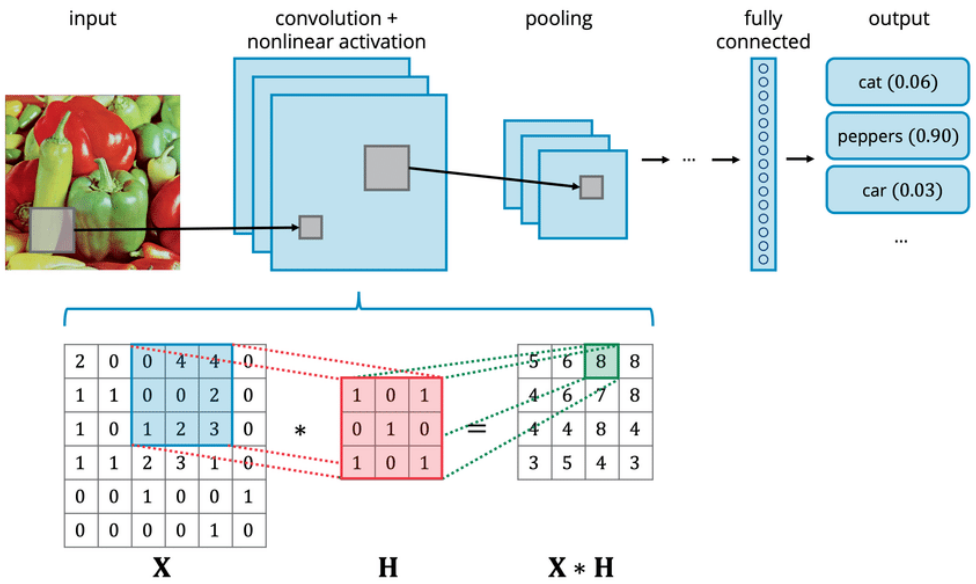
(h)

Szeliski

CNNs

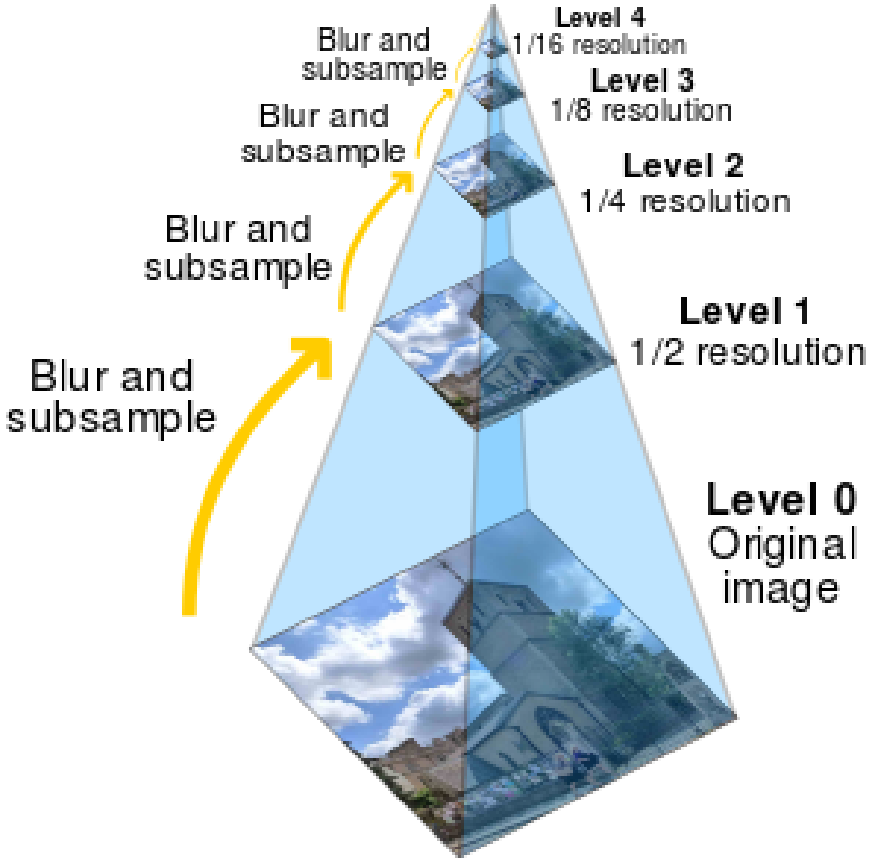
Filtros lineales para procesamiento básico

Convolución como operación básica de las CNNs
Características útiles del conjunto de imágenes



Pirámides

Transformaciones que no mantienen tamaño
Subir y bajar resolución

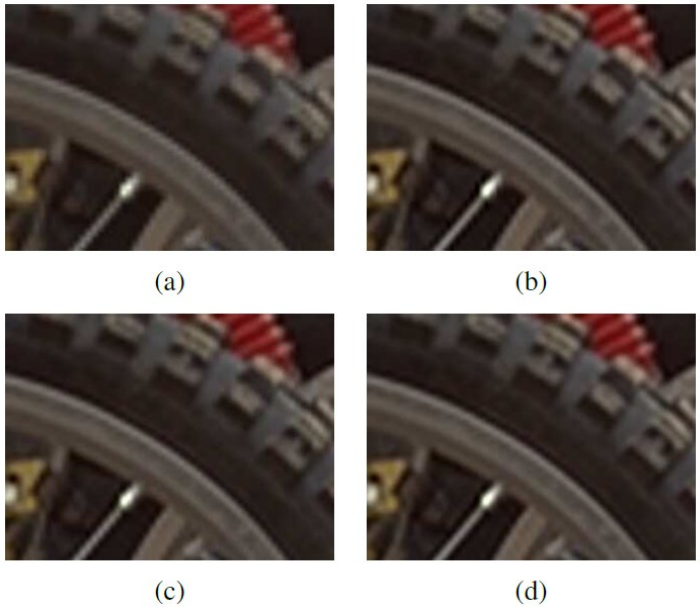


Srinath Sridhar

Pirámides

Aumentar resolución (*upscaling*)
Utilizar *kernel* que permita interpolar (*r* factor escalado)

$$g(i,j) = \sum_{n,m} k(n,m) f(i - rn, j - rm)$$



Szeliski

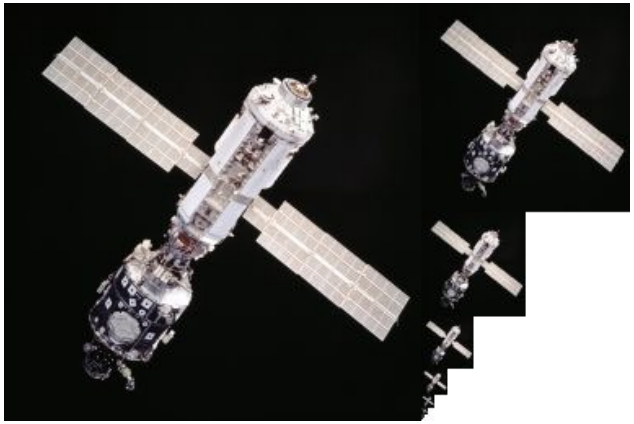
Pirámides

Reducir resolución (*downscaling*)
Convolución con filtro paso alto para eliminar *aliasing*

$$g(i,j) = \sum_{n,m} k(n,m) f(ri - n, rj - m)$$

Filtros binomial, p.e. [1 2 1], cúbico, coseno, 9/7 jpeg, etc.

Mip-mapping representación multiresolución



Pirámides

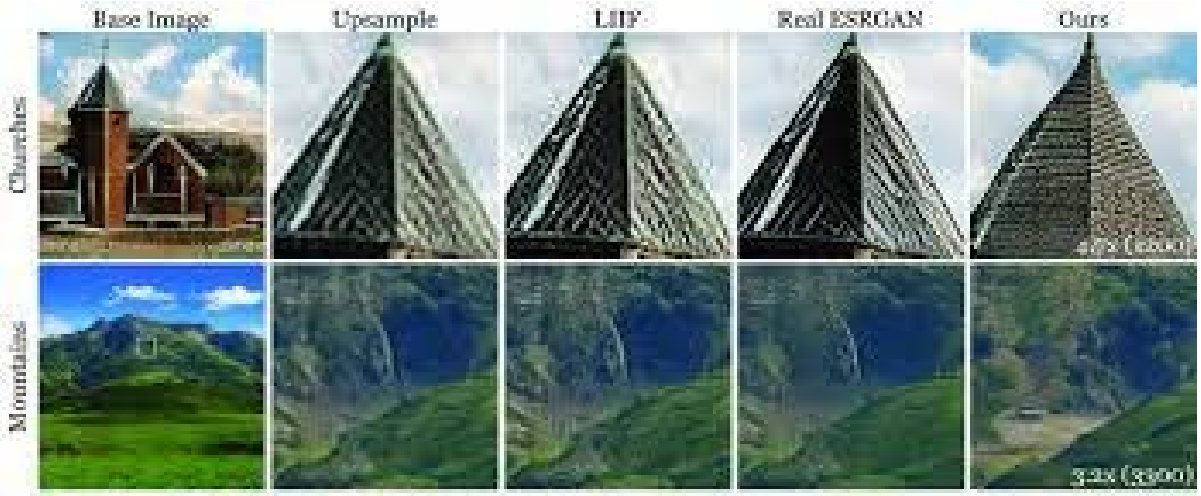
OpenCV

```
cv2.resize(src, dsize[, dst[, fx[, fy[, inter]]]])
```

- `INTER_NEAREST` Interpolación vecino más cercano
- `INTER_LINEAR` Bilineal (por defecto)
- `INTER_AREA` Relación de áreas. Para *downscaling*
- `INTER_CUBIC` Bicúbica
- `INTER_LANCZOS4` Interpolación Lanczos

Pirámides

Superresolución



Local Implicit Image Function, CVPR 21

Geométricos

No transformar el rango de la imagen sino el dominio

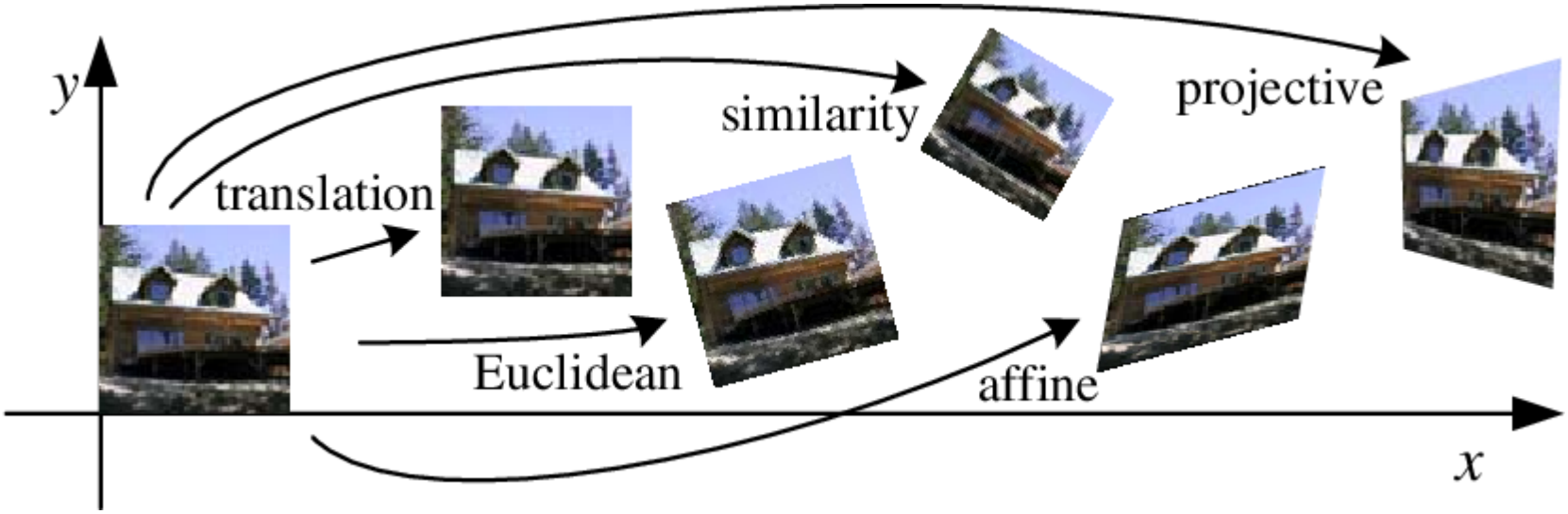
$$g=f(T(x))$$

Más allá de cambios de resolución:

- Paramétricas
- Basadas en mallas

Geométricos

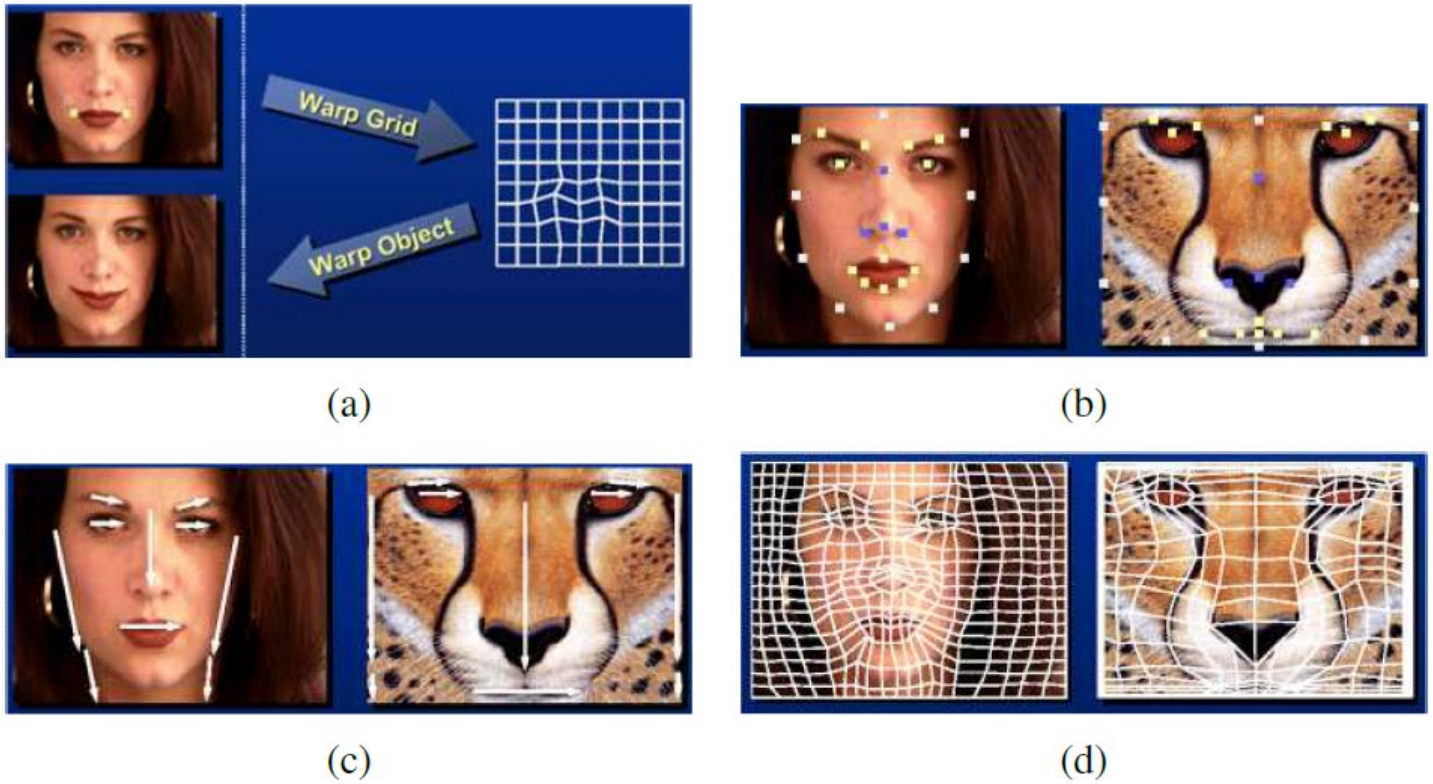
Paramétricas
Controlados por un número de parámetros globales



Szeliski

Geométricos

Basadas en mallas
Control local



Szeliski

Transformadas

Series de Fourier

Cualquier función periódica se descompone en la suma ponderada de infinitas sinusoidales de distinta frecuencia.

$$\sum_{n=1}^{\infty}(a_n \cos(nt) + b_n \sin(nt))$$

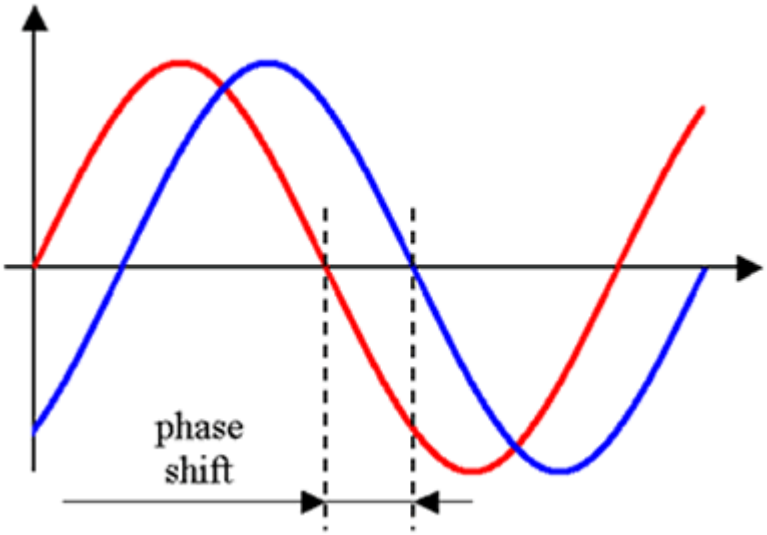
Transformadas

Fase desplaza los valores de coseno a seno

Descomposición en funciones seno

$$\sum_{n=1}^{N} (a_n \sin(nx + \emptyset_n))$$

Fase



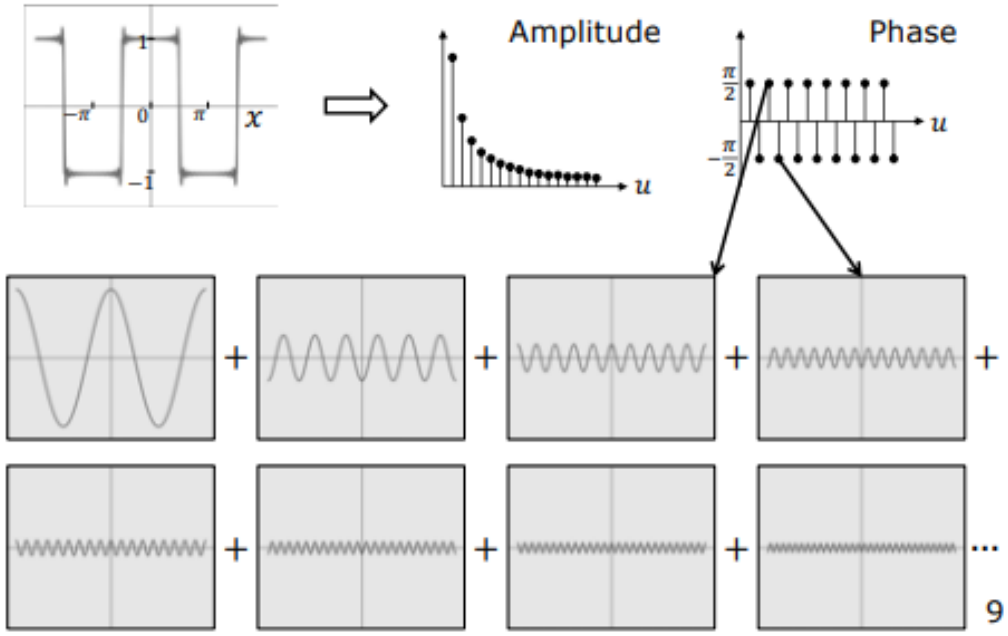
Transformadas

Transformada de Fourier

Mapea las respuestas de amplitud y fase a cada frecuencia

Representación alternativa a cualquier señal

Espectro de frecuencias



S.Nayar

Transformadas

Señales no periódicas, asume período infinito
Imágenes y filtros tienen su transformada de Fourier
Dominio frecuencial
Filtros del dominio espacial:

1. Aplicar transformada de Fourier
2. Trabajar en dominio frecuencial
3. Aplicar inversa transformada de Fourier

Fourier Transform:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx$$

x: space

u: frequency

$e^{i\theta} = \cos\theta + i\sin\theta$

$i = \sqrt{-1}$

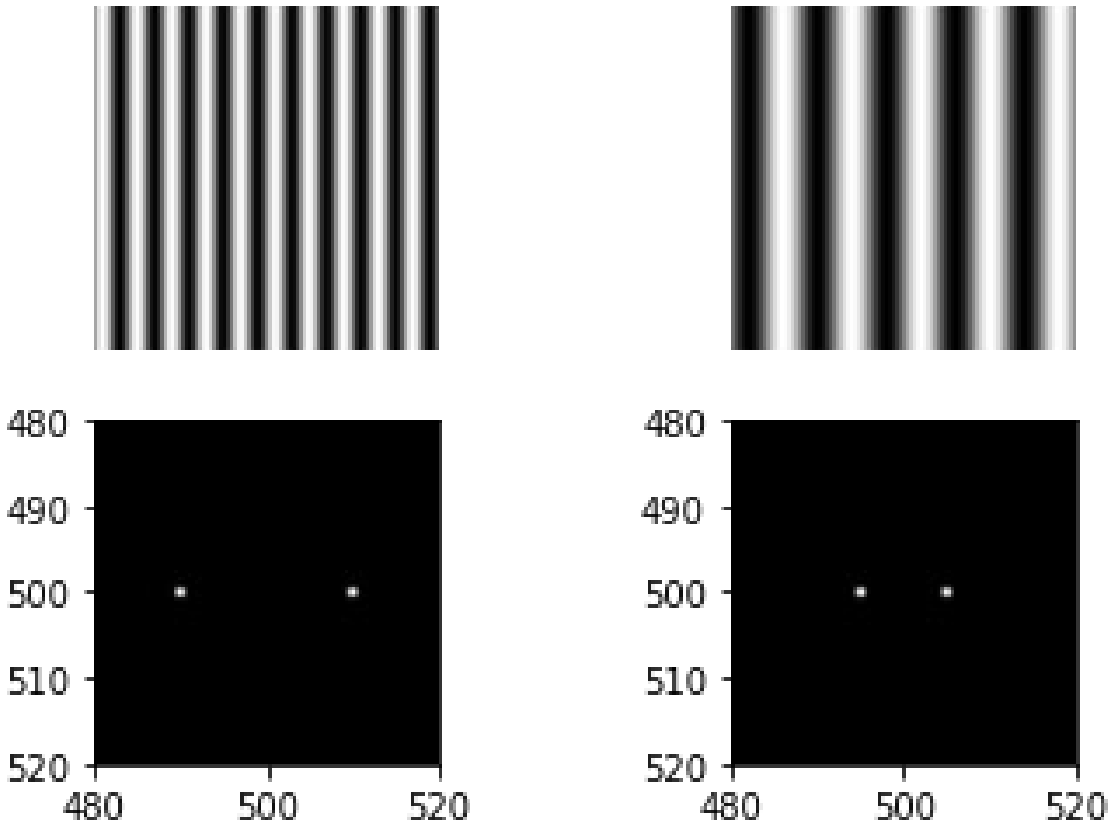
Inverse Fourier Transform:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} du$$

S.Nayar

Transformadas

Visualización transformada de Fourier de una imagen
Magnitud y frecuencia
Origen en el centro

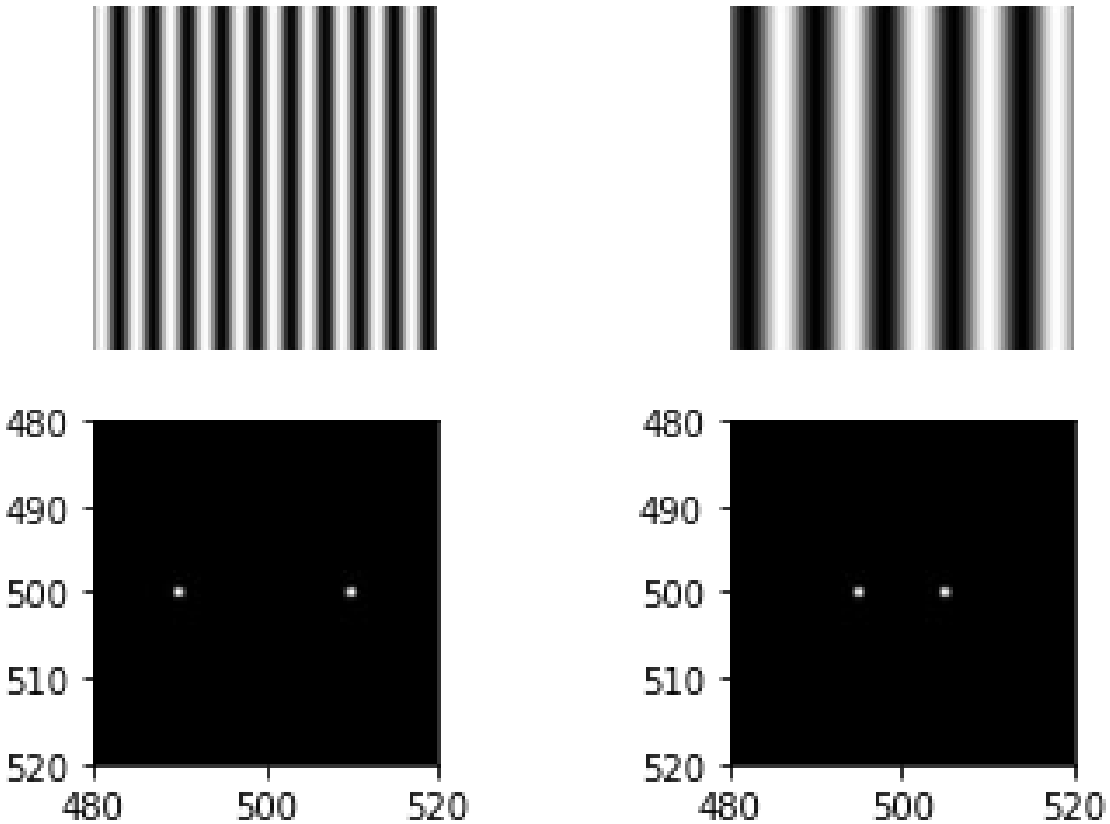


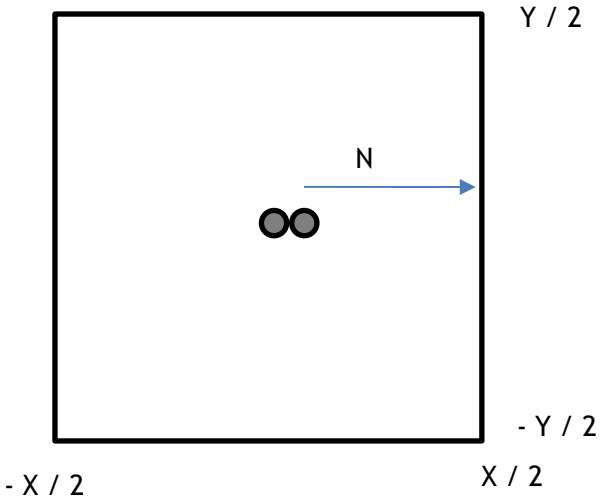
Transformadas

Variación de la frecuencia de la imagen de entrada

$a_0/2$ media de la señal

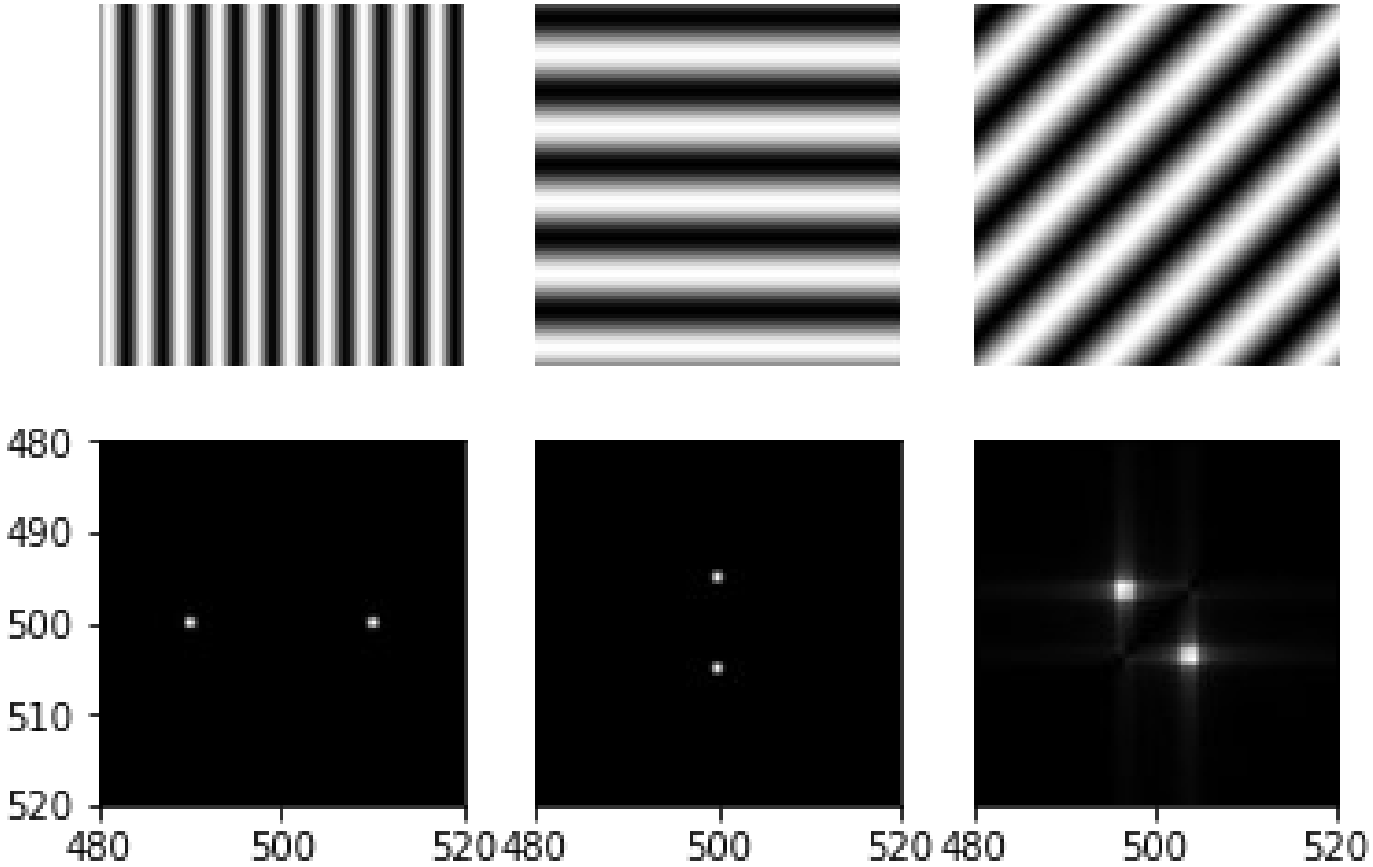
$$\frac{a_0}{2} + \sum_{n=1}^{N} (a_n \sin(nx + \emptyset_n))$$





Transformadas

Variación de frecuencia y orientación



Transformadas

Cálculo complejo

Optimizado, la transformada rápida de Fourier (FFT)

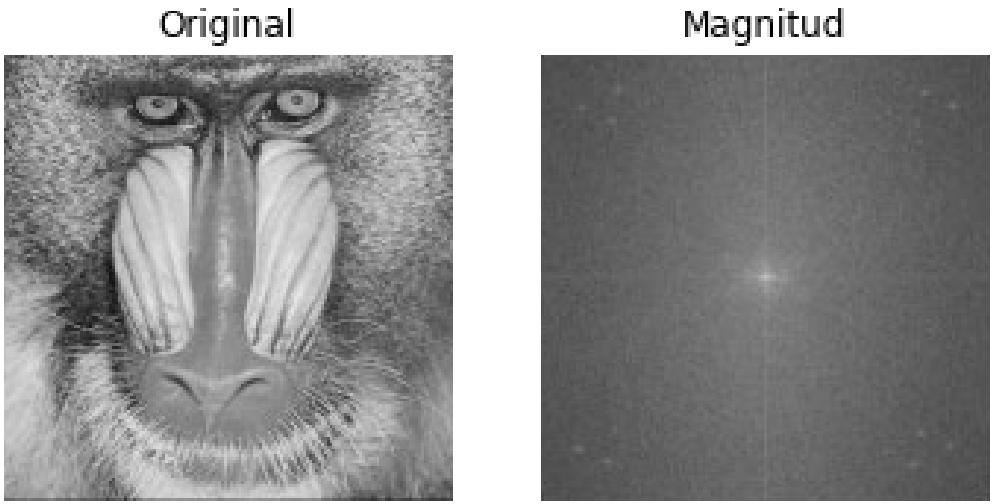
Funciones disponibles en numpy, OpenCV, etc.

La convolución de dos señales en dominio espacial equivale al producto en dominio de frecuencia

Más conveniente trabajar en el espacio de frecuencias con filtros grandes

Transformadas

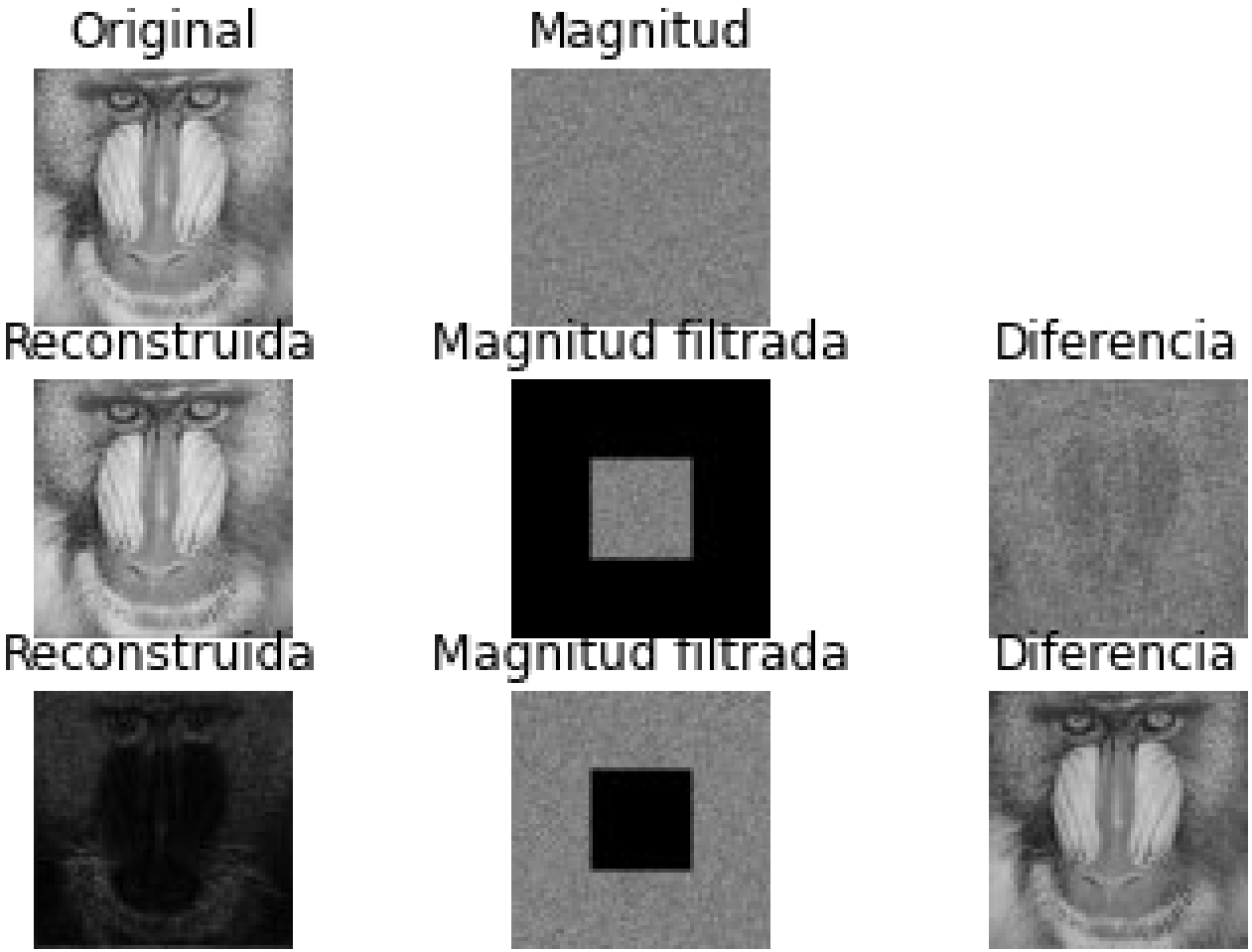
Visualización imagen natural



Transformadas

Filtrado/edición en el dominio de frecuencias

Elimina en el espacio de frecuencias las altas frecuencias



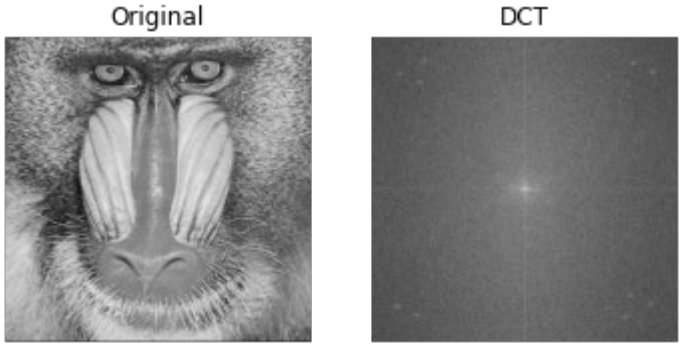
Transformadas

Transformada del coseno discreta (DCT)

Variante, base de funciones coseno

Para compresión de imágenes

Aproxima descomposición de Karhunen-Loéve de estadísticos de imágenes (PCA análisis de componentes principales)



$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1.$$

Transformadas

Convolución en dominio espacial equivale a producto en frecuencial

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

kernels grandes, interesante usar FFT y operar en frecuencias

Pensar en espacio de frecuencias

Aplicaciones en compresión (jpeg)

Bibliografía

- S. K. Nayar, First Principles of Computer Vision
- R. Klette, *Concise Computer Vision*, Springer, 2014 (online ULPGC)
- R. Szeliski. Computer Vision: Algorithms and Applications, 2nd ed. 2022
- R. van den Boomgaard. *Image Processing and Computer Vision*, 2017