Visión por Computador

Tema 3. Procesamiento de imagen

EII, GII. *Modesto Castrillón Santana*

Contenidos

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

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

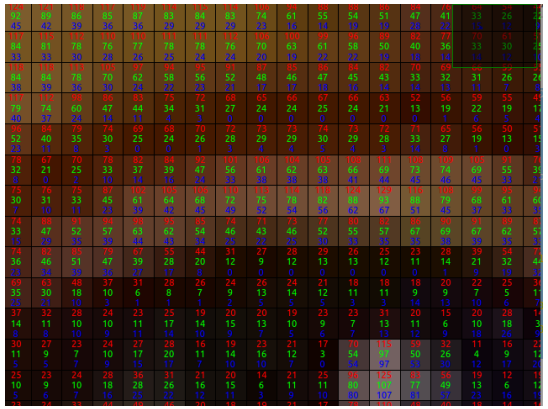
Puntuales

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

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

Imágenes discretas 2D

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

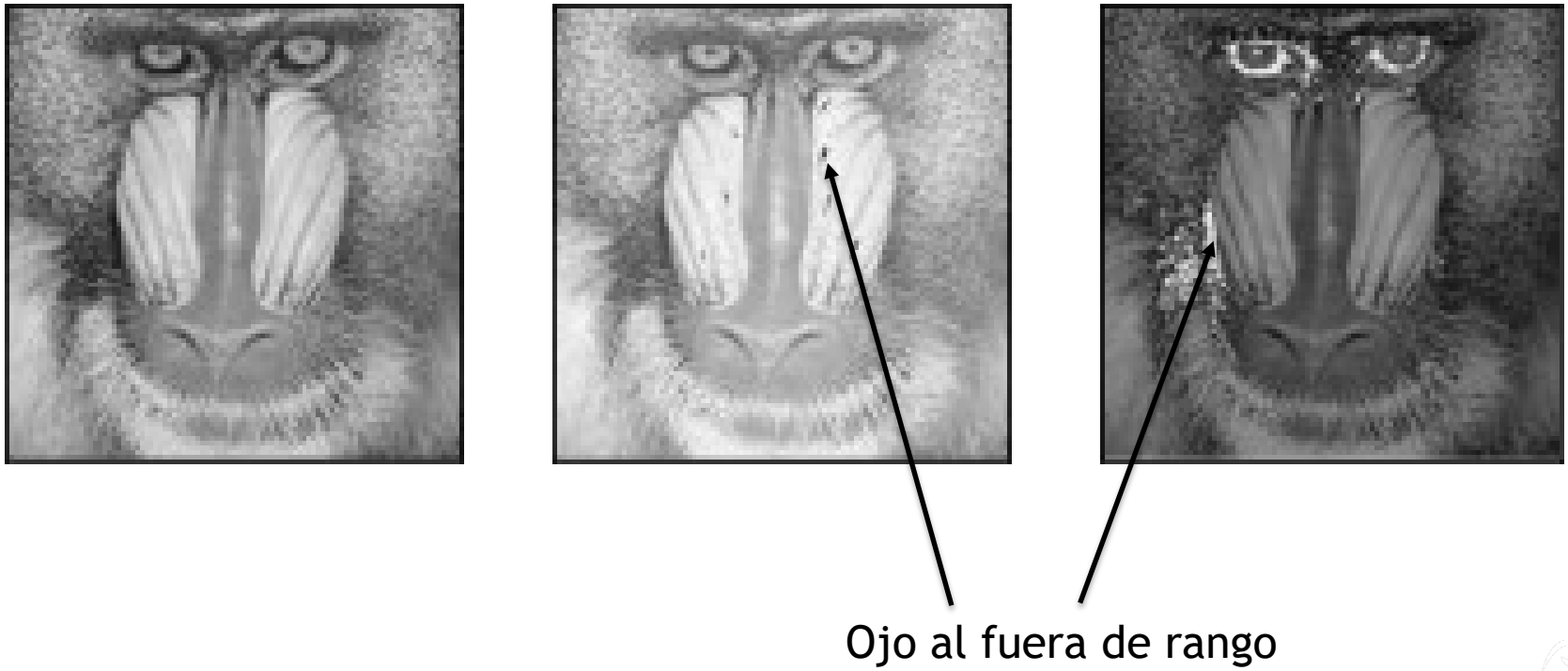


Aditivos/sustractivos

Sumar/restar valor constante

Aclaran u oscurecen

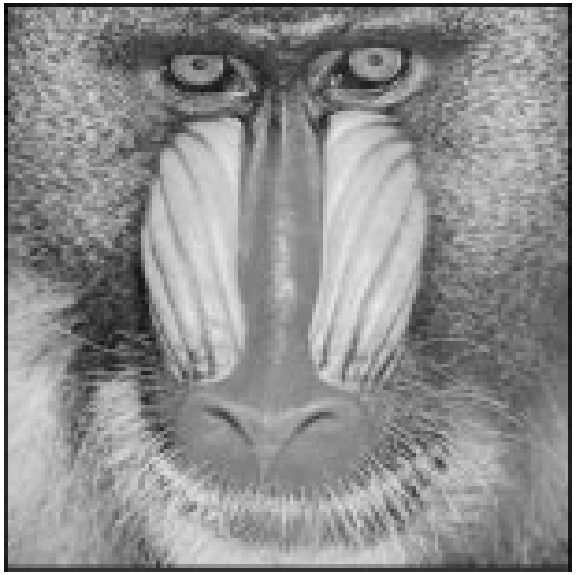
$$g(i, j) = T(f(i,j)) = f(i,j) + k$$

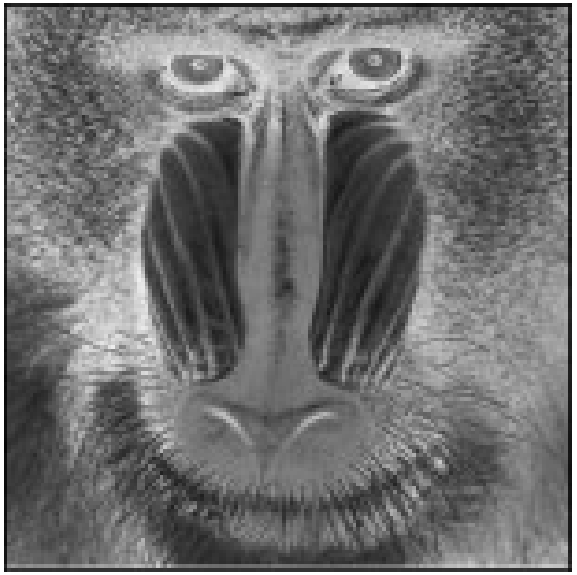


Inversión

Invertir valores

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





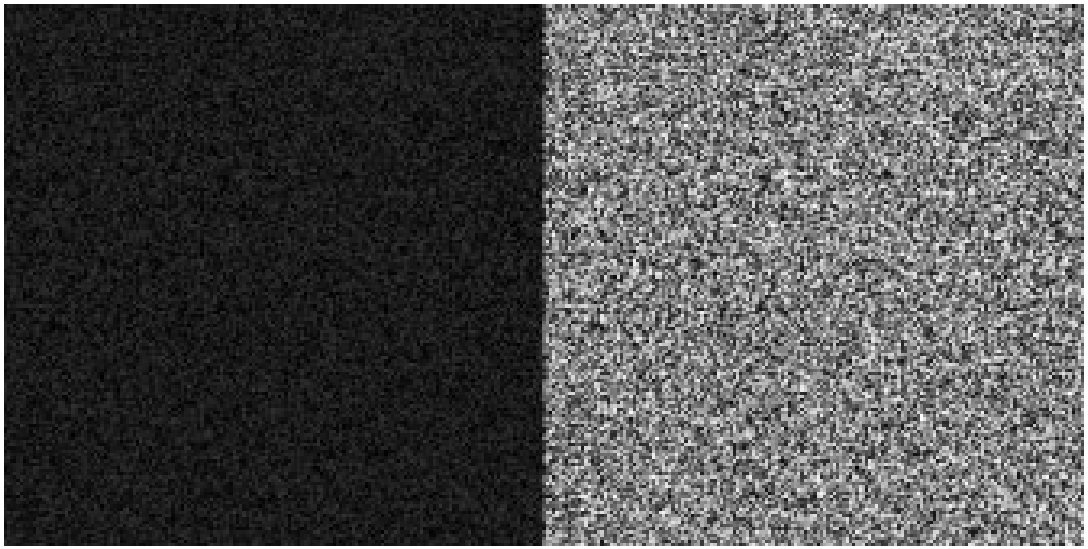
Producto

Multiplicar por valor

Aumenta (>1) o disminuye (<1) el contraste

Ojo al rango

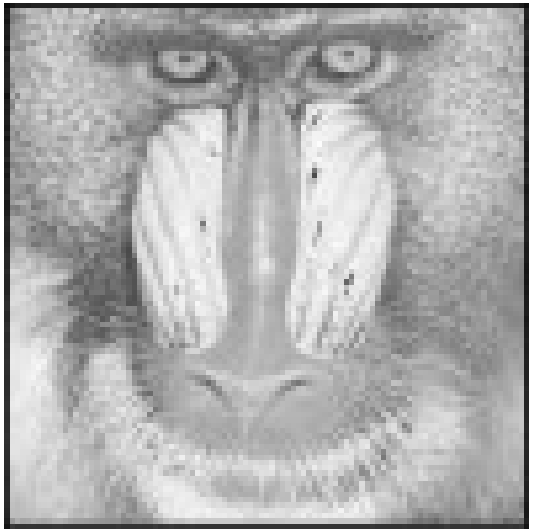
$$g(i, j) = T(f(i,j)) = f(i,j)*a$$



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)*

$0 <= \alpha <= 1$

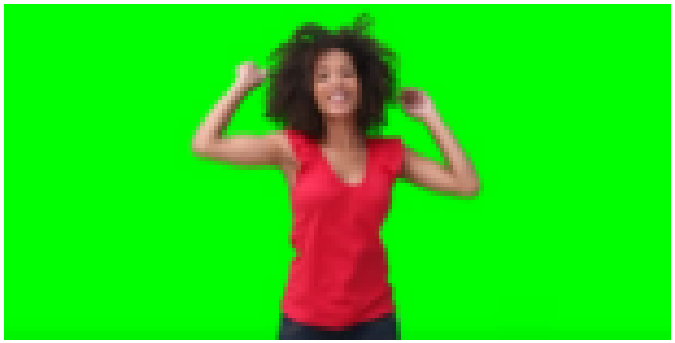
$$g(i, j) = \alpha * f_1(i,j) + (1-\alpha) * f_2(i,j)$$

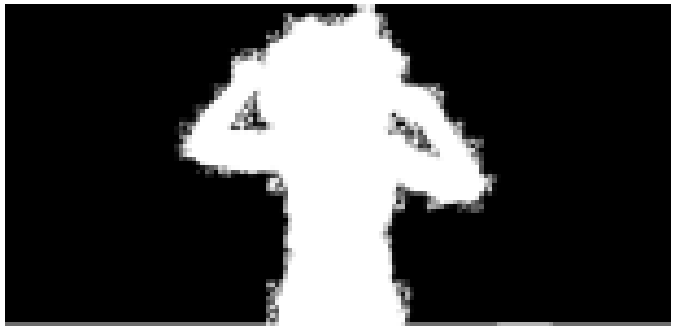
Combinación

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

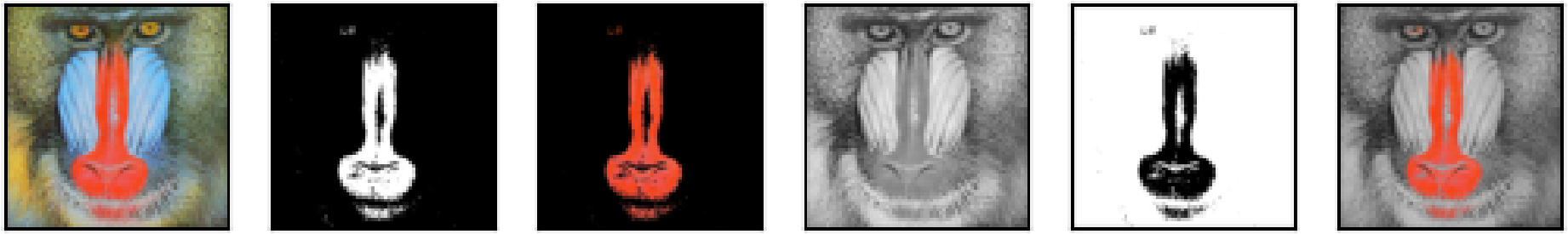
```
imgrgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
imghsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#Máscara rango rojo y extracción desde imgrgb
mask = cv2.inRange(imghsv, (0, 120, 70), (10, 255, 255))
imgmasked = cv2.bitwise_and(imgrgb, imgrgb, mask=mask)
#Máscara 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:

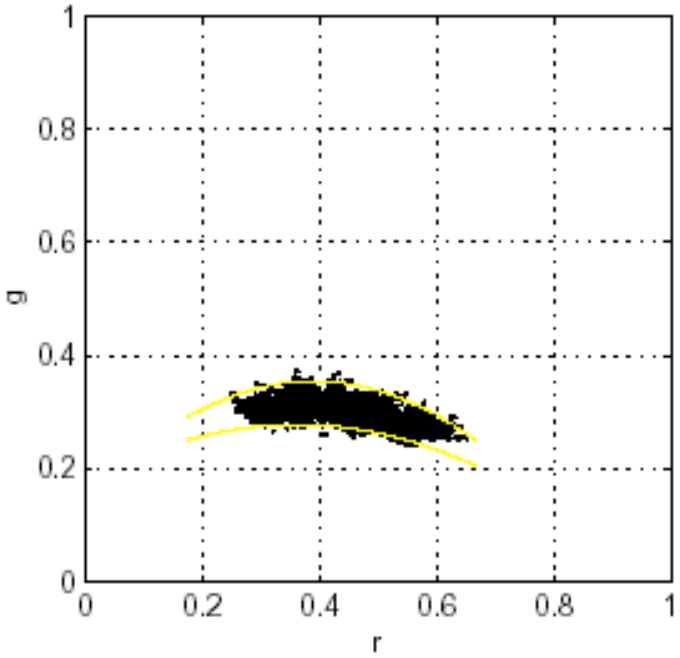
```
col = np.uint8([[[0,255,0 ]]])
hsv_col = cv2.cvtColor(col,cv2.COLOR_BGR2HSV)
print(hsv_col)
```

Tono definido con rango del valor h



Color

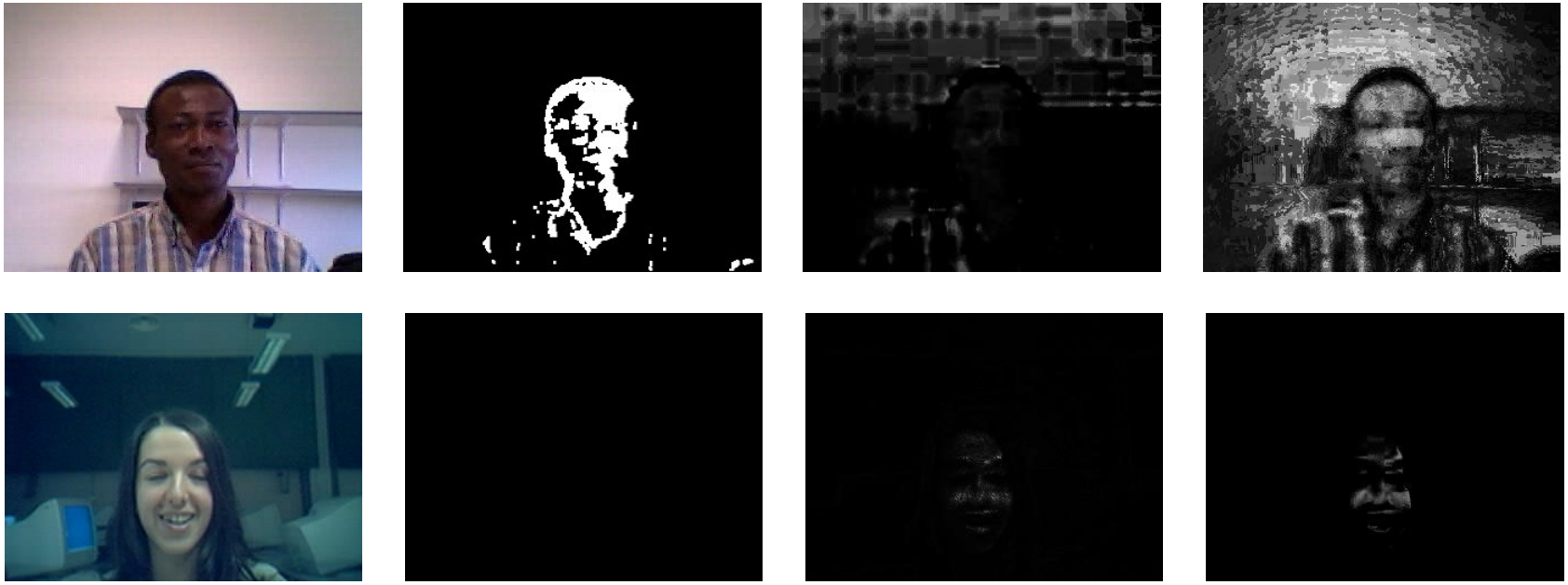
[Batallita]
Color piel
rojo-verde normalizado (R/(R+G+B) y G/(R+G+B))



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

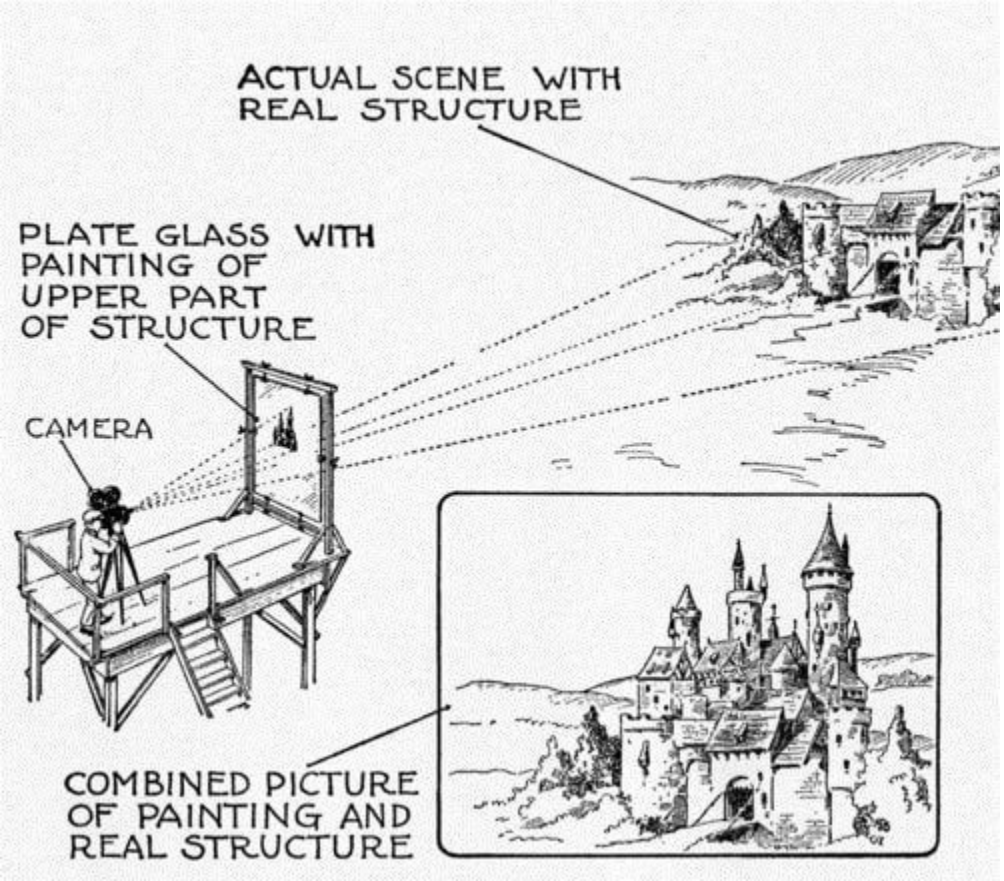


Esquema simple | Jones and Rehg [1] | Kruppa [2]

[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

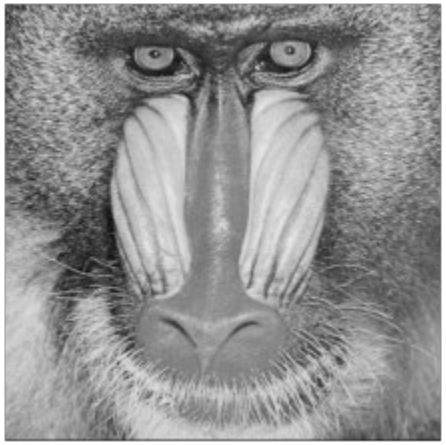
Actualidad: digital

Figura/fondo

Umbralizado
Asignar valor aplicando condicional

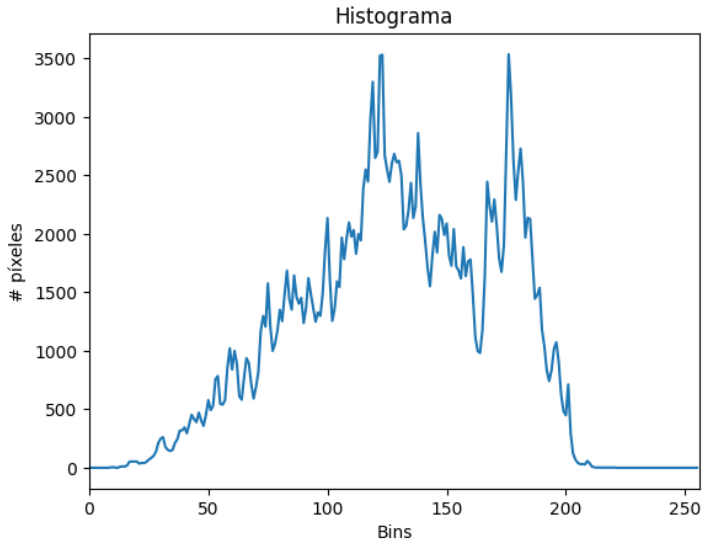
Figura/fondo

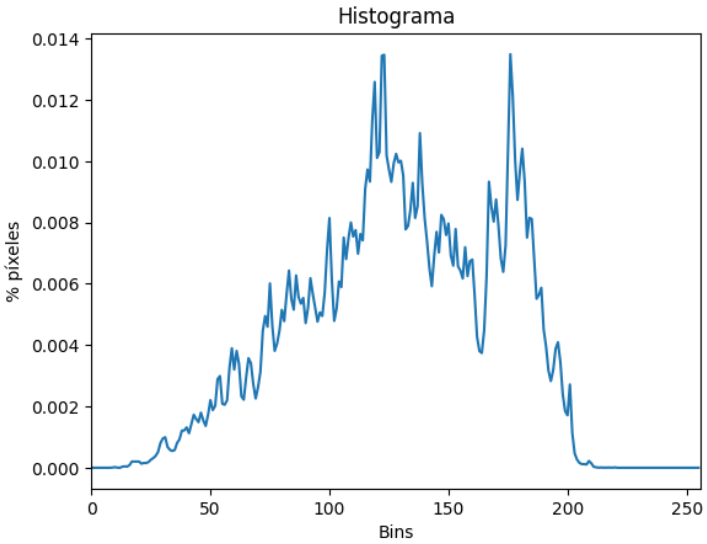
Recordando histograma
Distribución intensidades píxeles



Operador no dependiente de posición en imagen sino en histograma

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



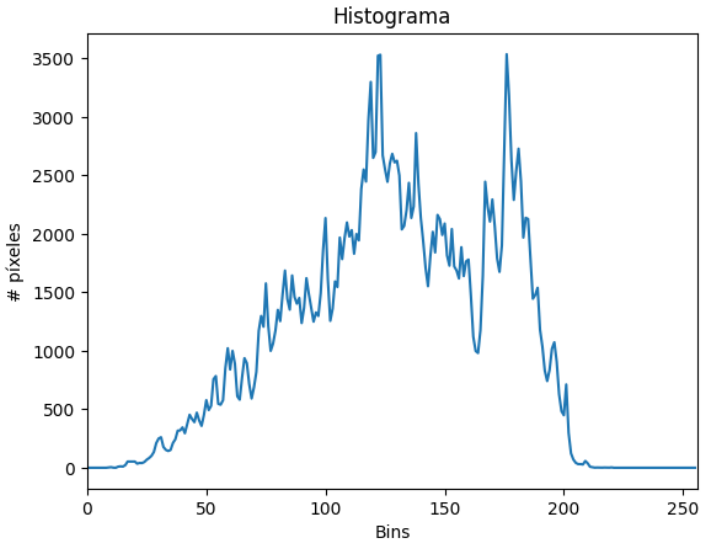


Figura/fondo

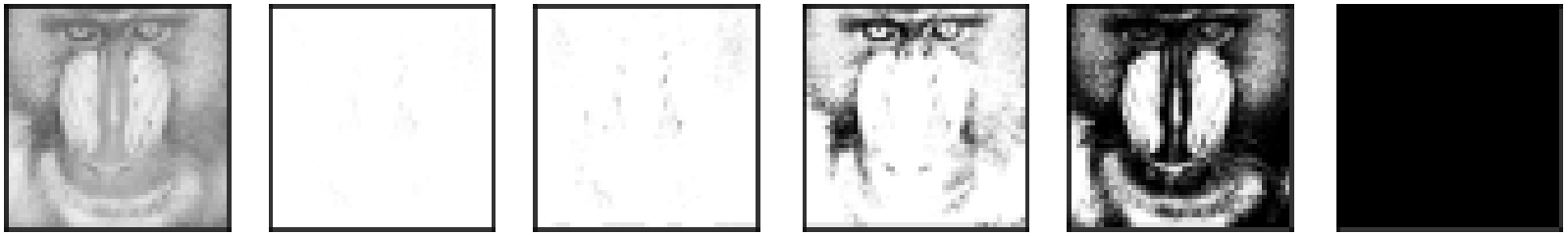
Umbralizado

Imagen binaria según umbral

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



Variando el umbral

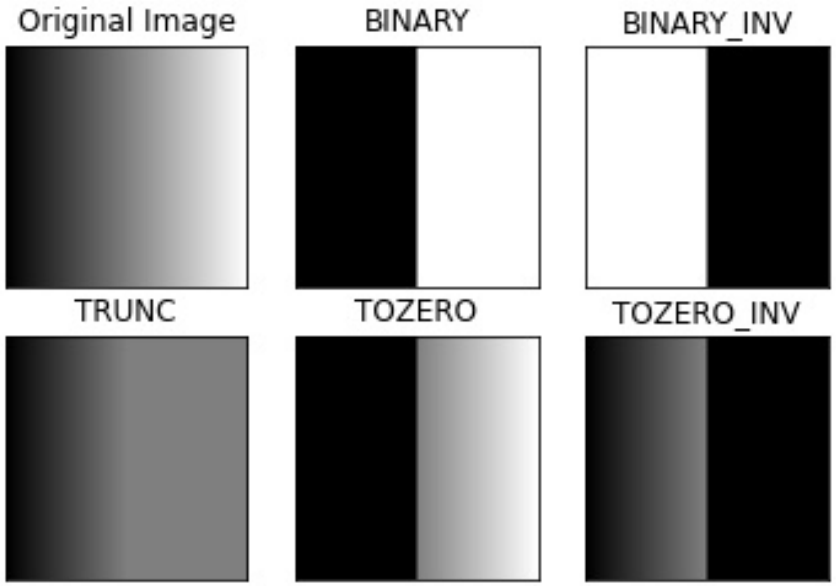


Figura/fondo

Umbralizado

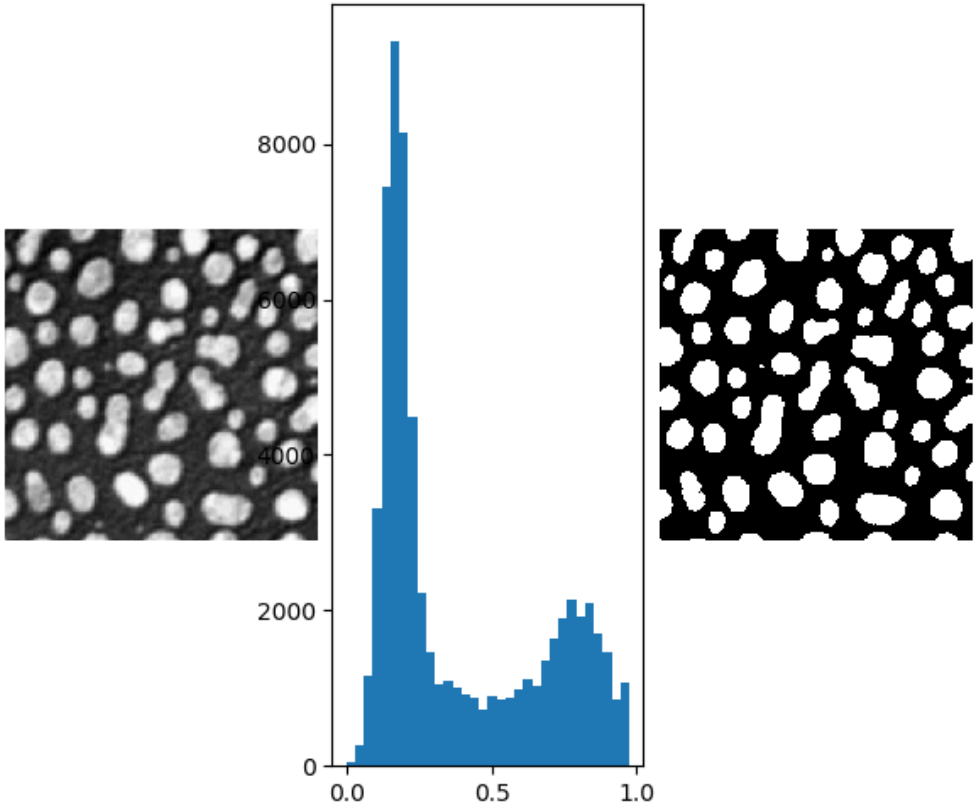
Opciones `cv2.threshold`

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



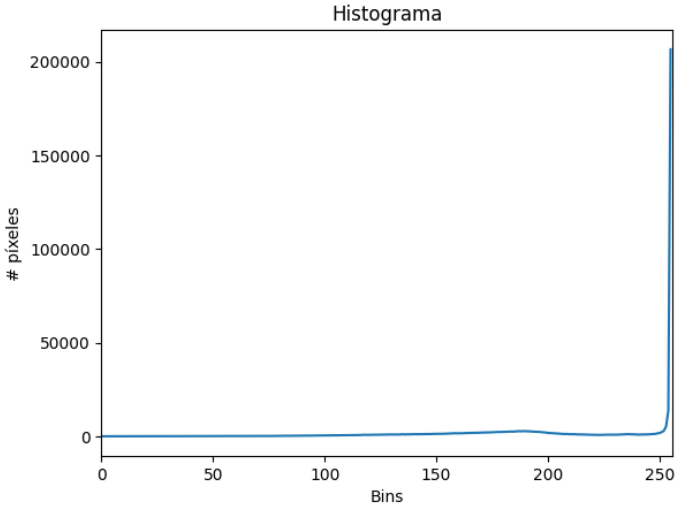
Figura/fondo

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)
```

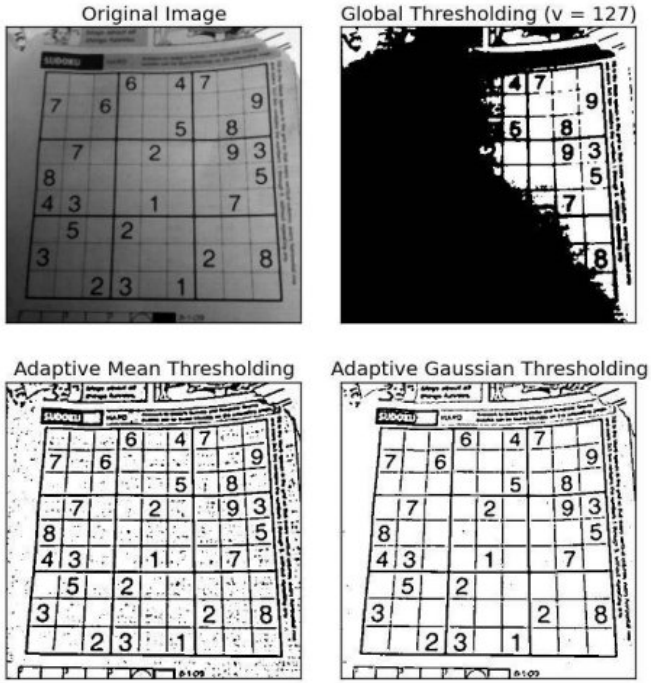
Figura/fondo

Separación de figura y fondo

Fijo

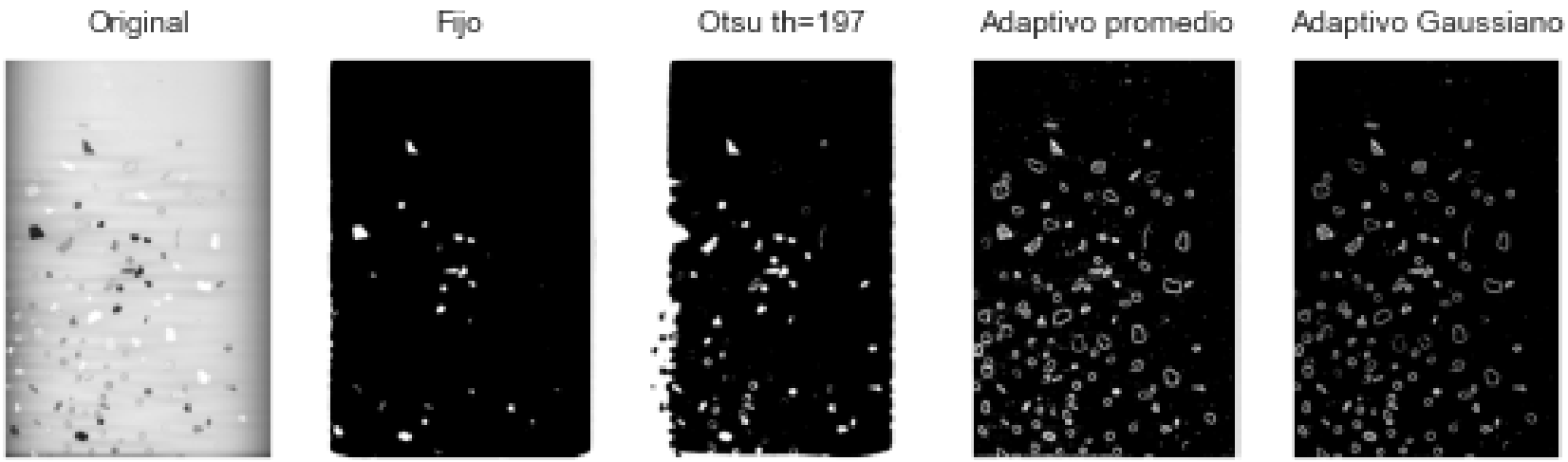
Adaptativo por vecindad, opciones `cv.adaptiveThreshold`

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



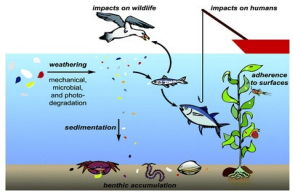
Figura/fondo

Separación de figura y fondo
Comparativa con OpenCV



Figura/fondo

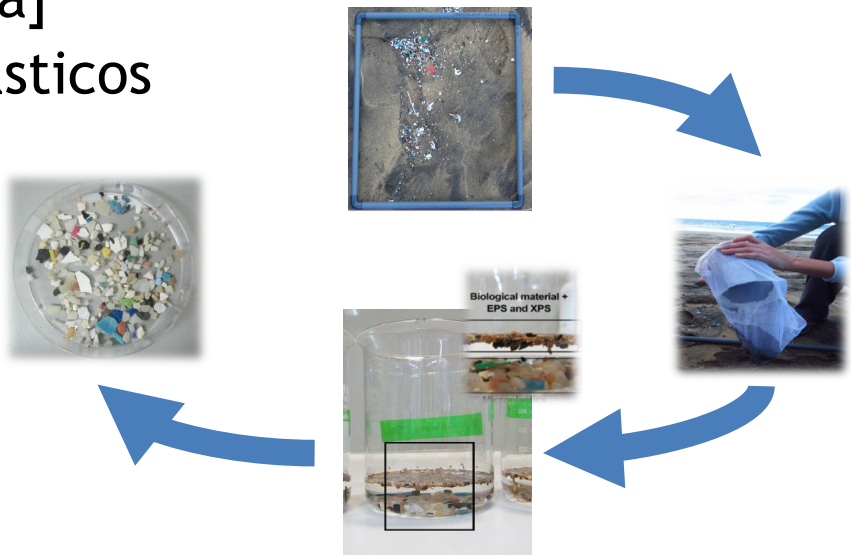
[Batallita]
Microplásticos



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

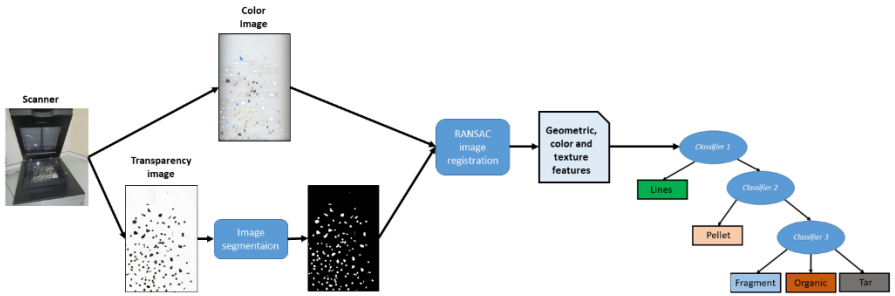
Pellet

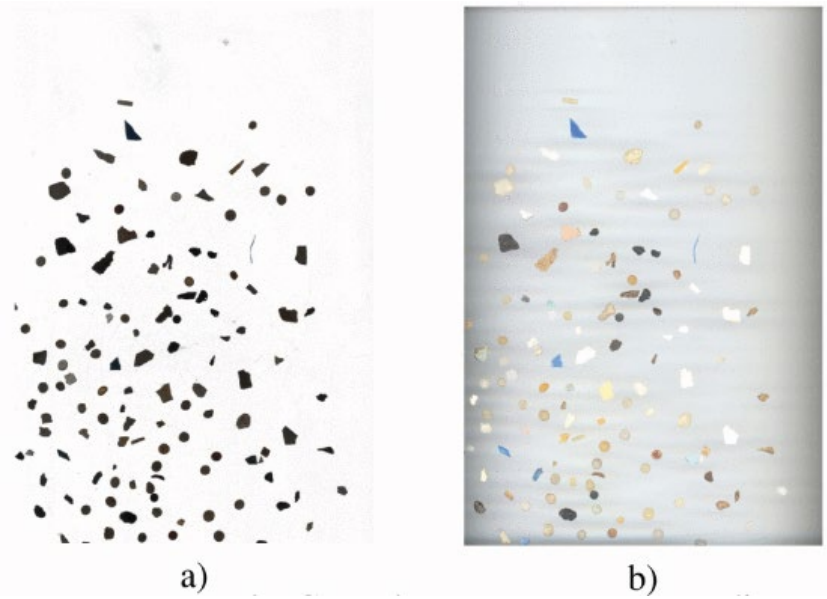
Alquitrán

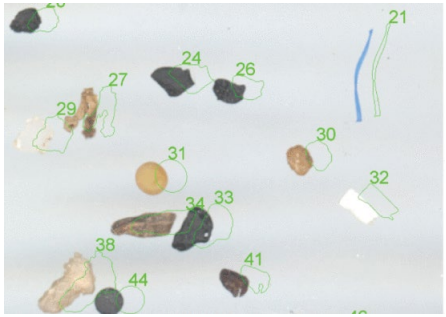
J. Lorenzo et al. *SMACC: A System for Microplastics Automatic Counting and Classification* *IEEE Access 2020*
J. Lorenzo et al. *Deep learning approach for automatic microplastics counting and classification* STOTEN 2021

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

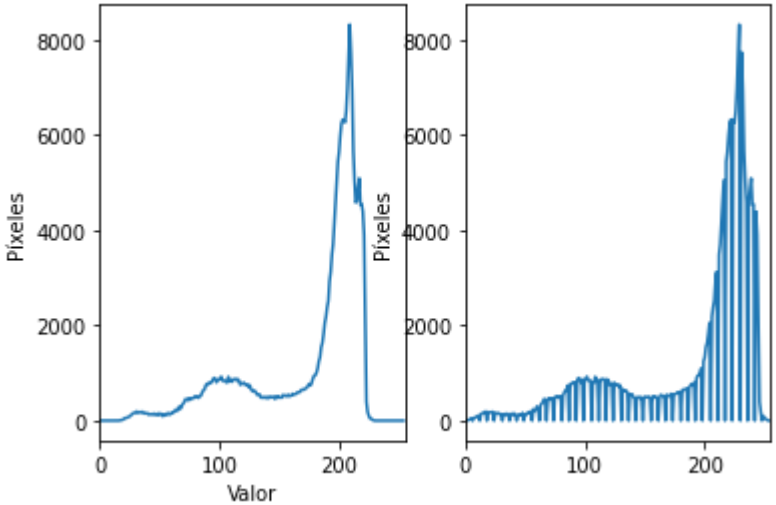
Figura/fondo

[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

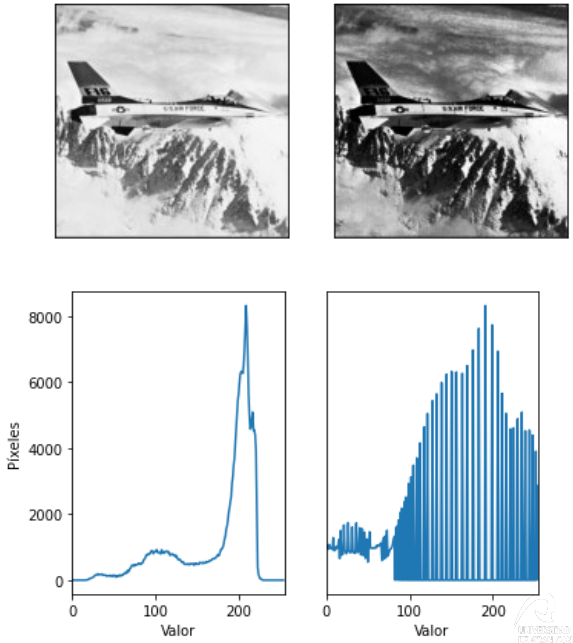
Pretende invarianza frente a cambios iluminación

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)*



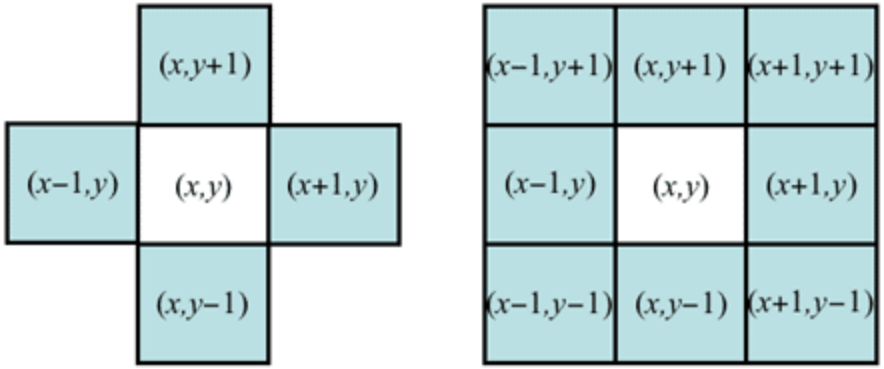
Filtros

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



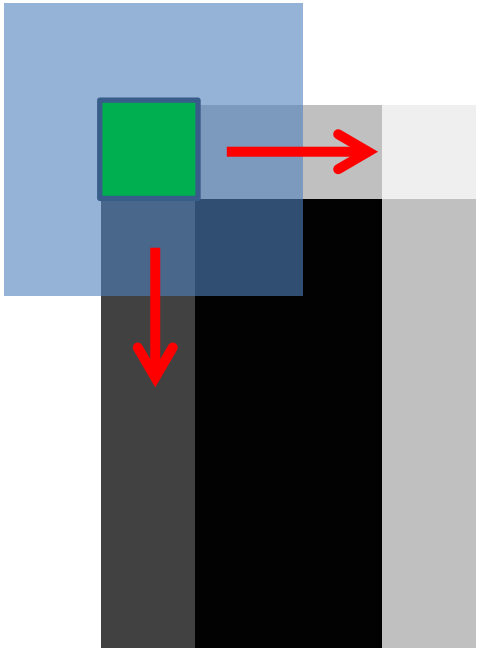
4-neighbourhood 8-neighbourhood

Filtros

Filtrado lineal
Función calculada en vecindad con pesos fijos

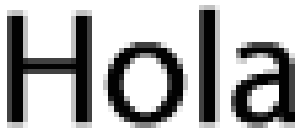
$$g(i,j) = \sum_{n,m} k(n,m) f(i+n, j+m)$$

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

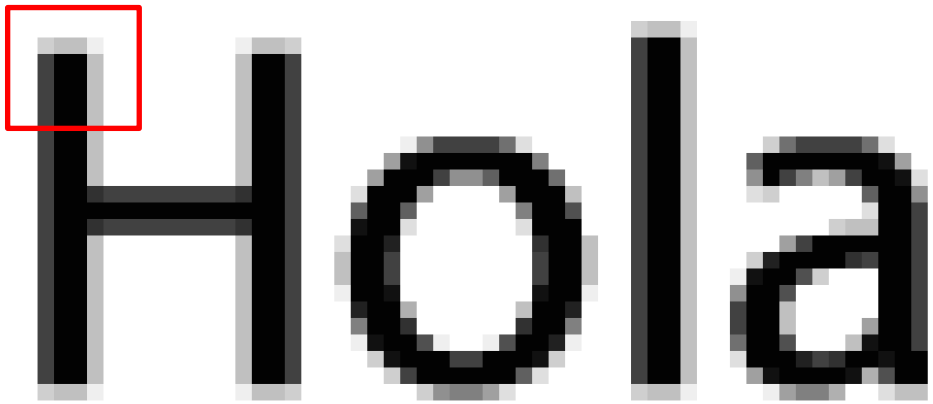


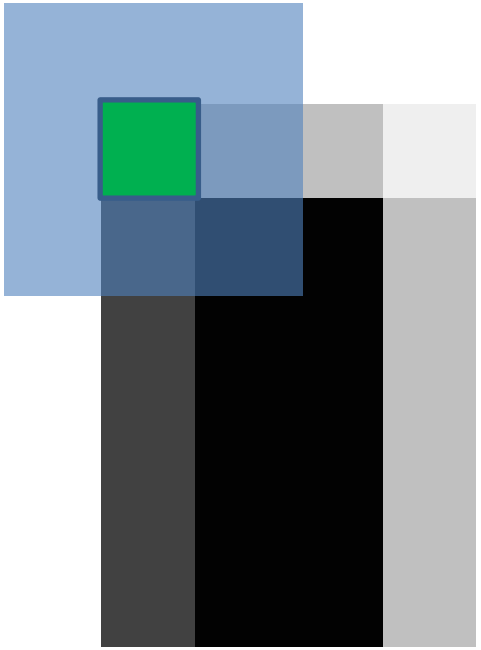
Filtros

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	

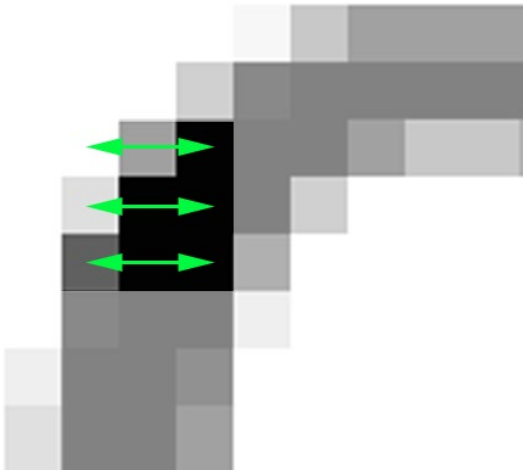
$k(n,m)$

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

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

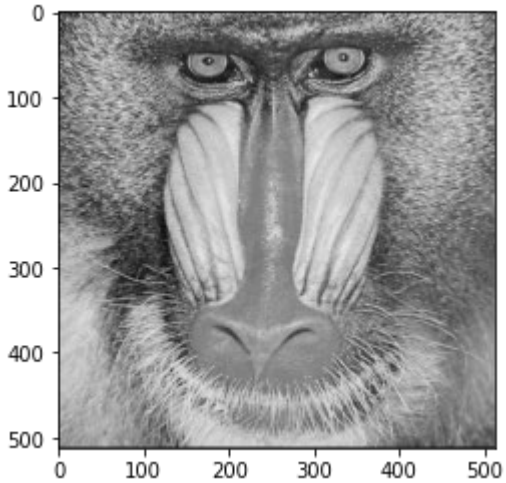
¿Qué pasaría al aplicar?

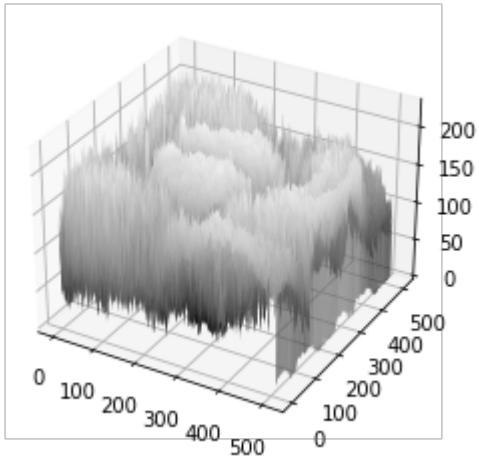
0	0	0
0	1	0
0	0	0

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