



UAI

UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS



iUAI
UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERIA Y CIENCIAS

MDS 2019 PROCESAMIENTO DIGITAL DE IMÁGENES

PRIMEROS PASOS

Miguel Carrasco
miguel.carrasco@uai.cl
1 Semestre 2020

▶ OpenCV

- Ejemplo 1 | Lectura de una imagen
- Ejemplo 2 | Separación de canales
- Ejemplo 3 | Selección de un área de interés (ROI)
- Ejemplo 4 | Binarización

Lena Söderberg, 1972
Playboy Magazine



Lena



Canales
RGB



Transformación en
escala de grises

fuente: http://en.wikipedia.org/wiki/Lena_Soderberg

- ▶ Imágenes en escala de grises



lena.jpg

La imagen está compuesta por una matriz de píxeles. Cada píxel tiene un valor único entre [0 a 255]

▶ Lectura de imagen en OpenCV

```
import cv2
```



Módulo principal de OpenCV. Importa toda la biblioteca de funciones de openCV



▶ Lectura de imagen en OpenCV

```
import cv2  
  
img = cv2.imread('lena.png')
```



imread carga una
imagen en la memoria

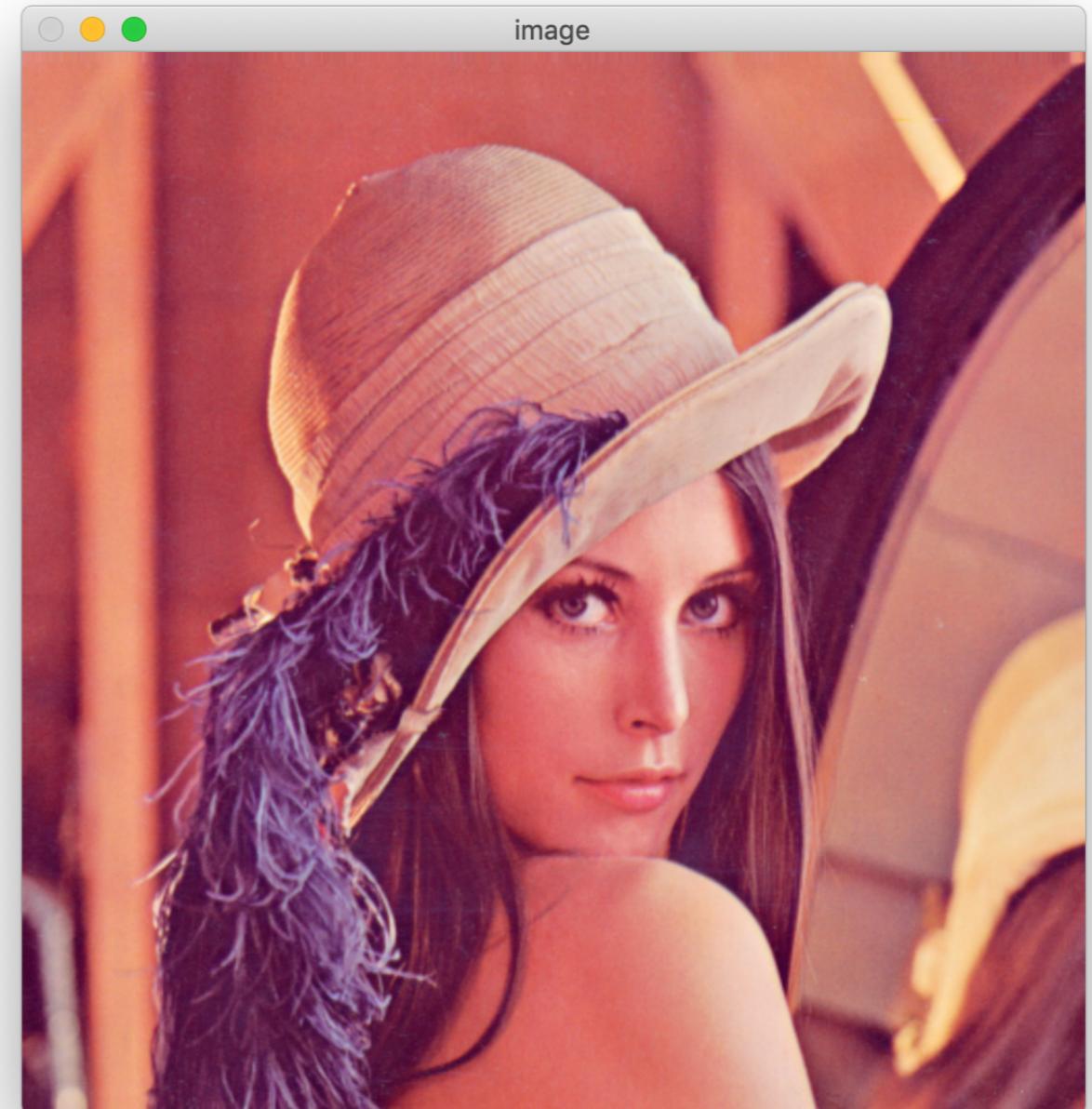


Debes descargar una
imagen (jpeg, png, bmp) en
la misma carpeta donde se
encuentre **tú script de**
Python

▶ Lectura de imagen en OpenCV

```
import cv2  
  
img = cv2.imread('lena.png')  
  
cv2.imshow('image',img)
```

imshow dibuja en la ventana con el nombre indicado en el primer parámetro



▶ Lectura de imagen en OpenCV

```
import cv2

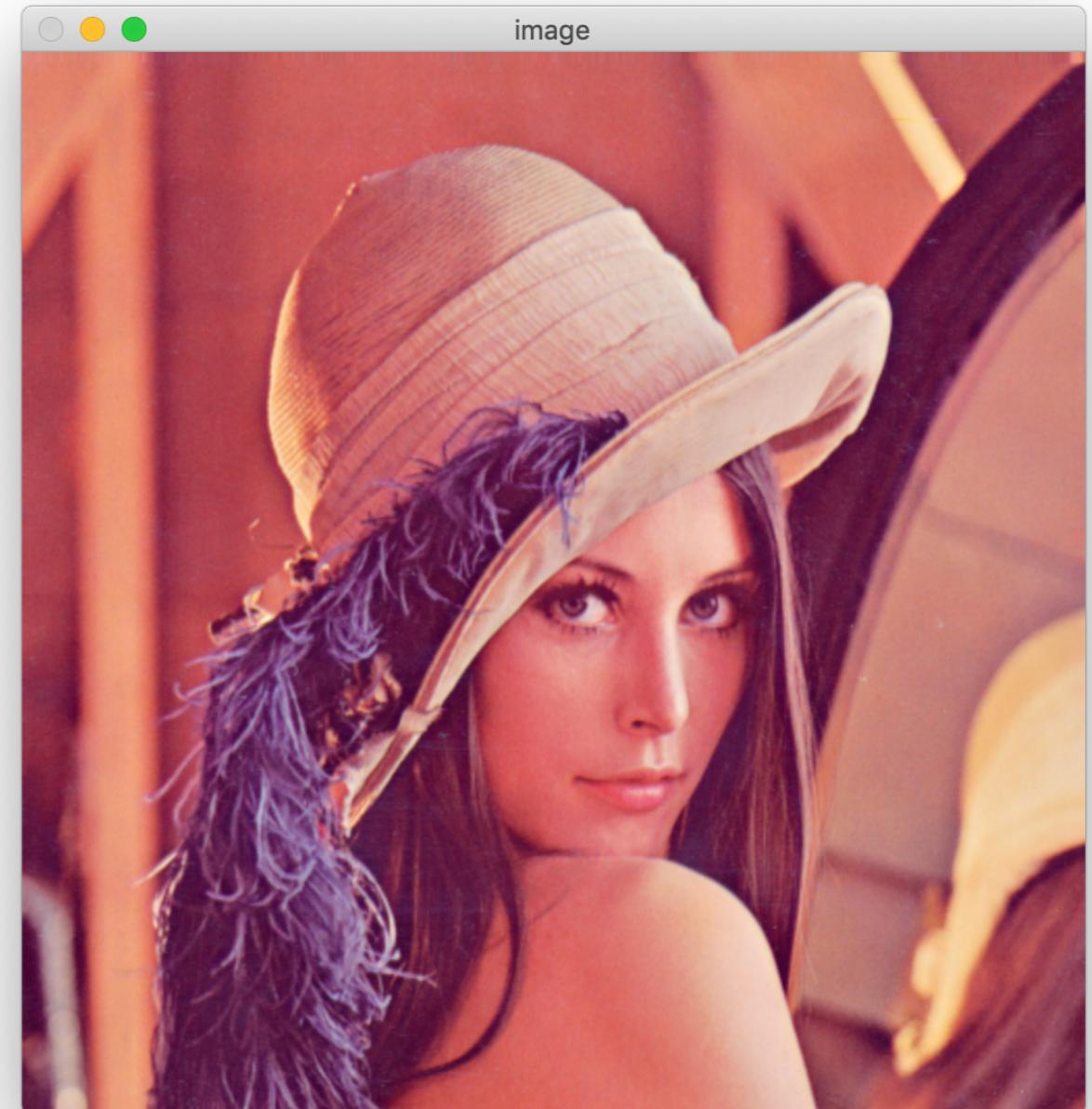
img = cv2.imread('lena.png')

cv2.imshow('image',img)

cv2.waitKey(0)
```



waitKey espera que sea presionada una tecla.



▶ Lectura de imagen en OpenCV

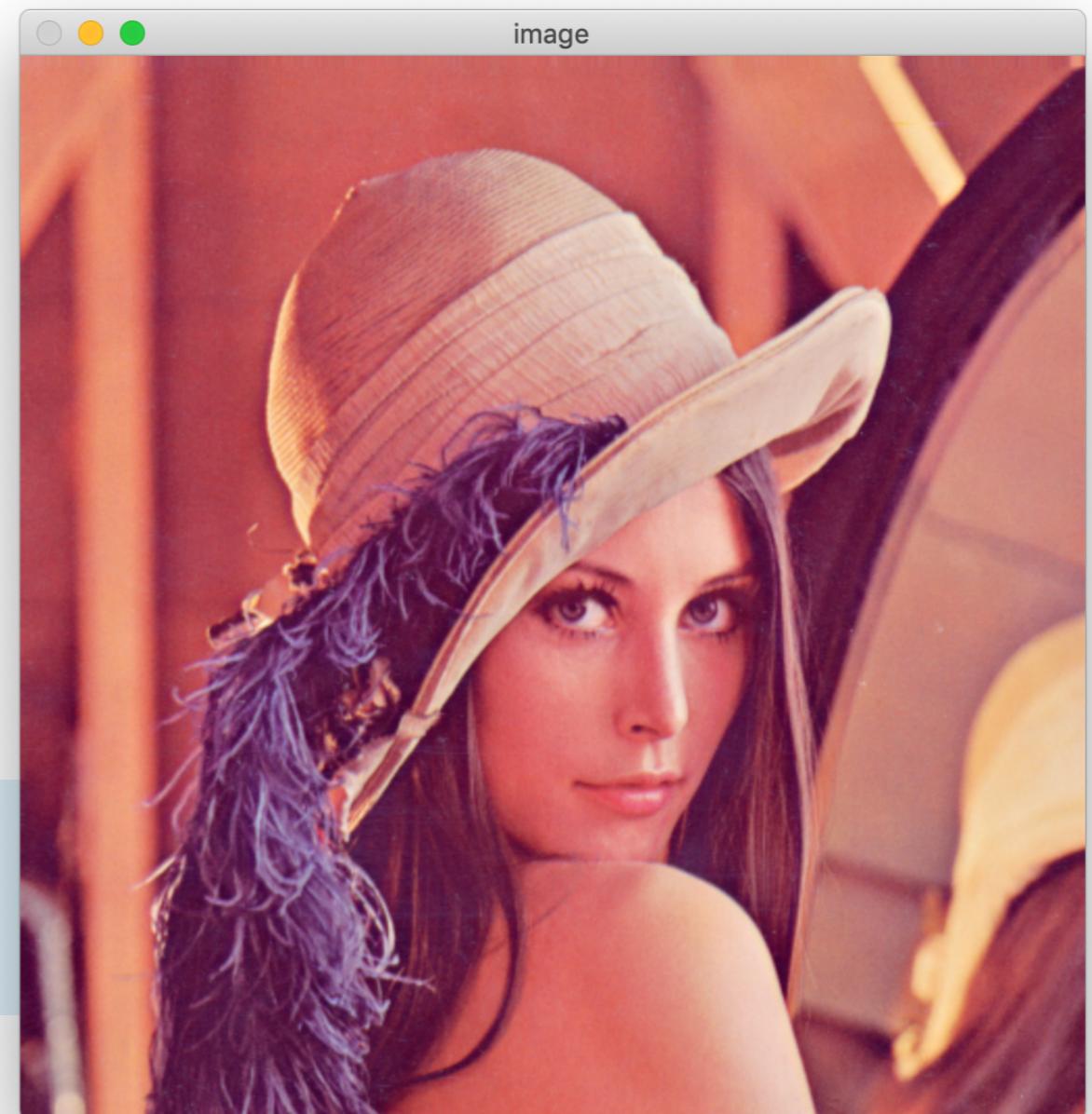
```
import cv2

img = cv2.imread('lena.png')

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```



cierra todas las ventanas de la aplicación



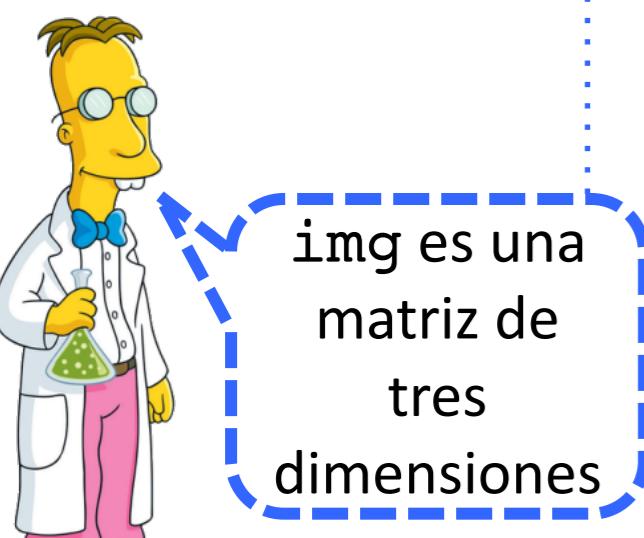
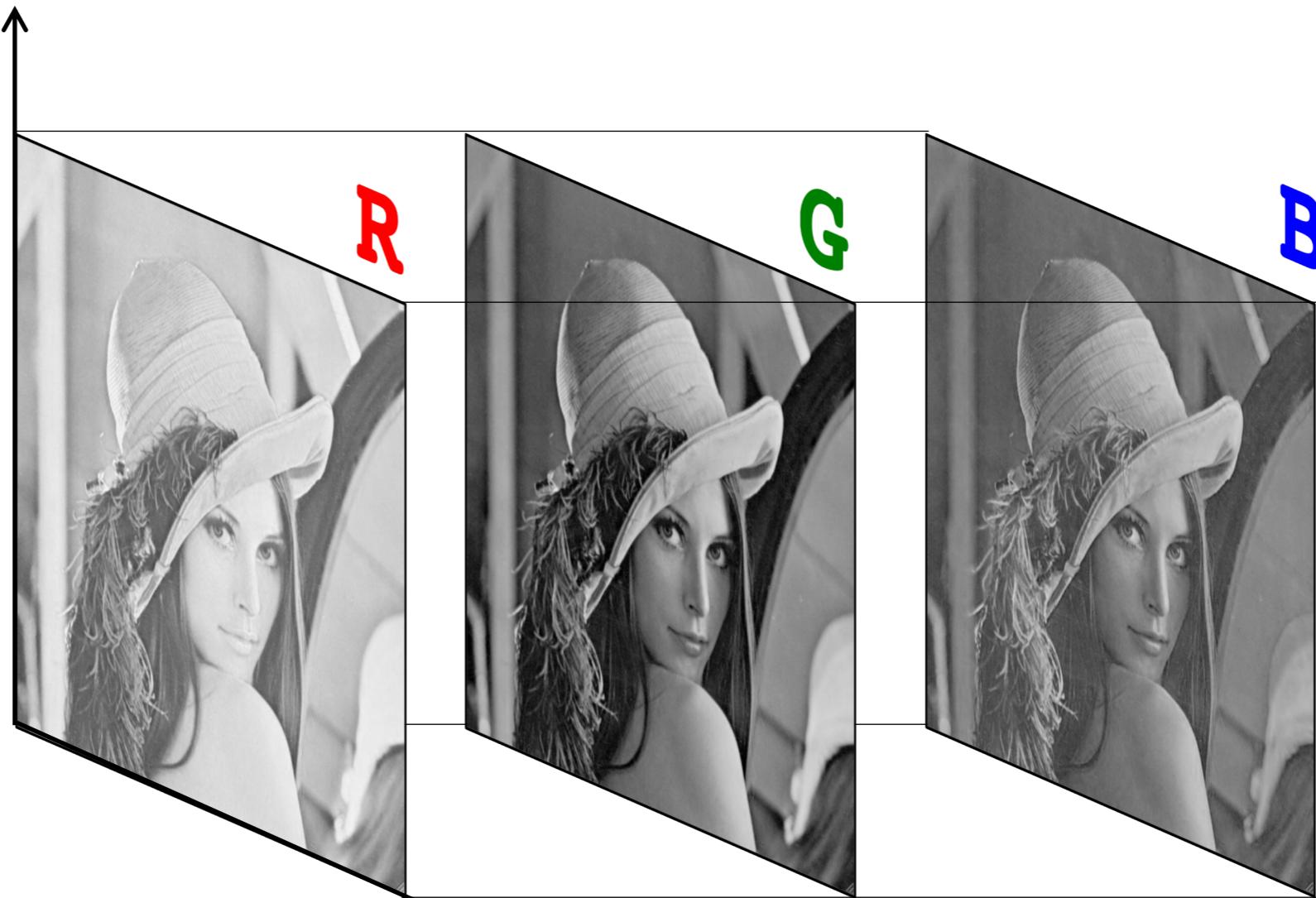
presiona cualquier tecla para cerrar la ventana

▶ OpenCV

- Ejemplo 1 | Lectura de una imagen
- Ejemplo 2 | Separación de canales
- Ejemplo 3 | Selección de un área de interés (ROI)
- Ejemplo 4 | Binarización

- ▶ Representación de cada canal RGB en la variable I

`img =`



img es una
matriz de
tres
dimensiones

- ▶ Representación de cada canal RGB

Mohammed Alim Khan (1880-1944).
Foto tomada por
Sergei Mikhailovich
Prokudin-Gorskii
(Wikipedia)

Imagen tomada en
1911 empleando tres
filtros Rojo, Verde y
Azul. (Ningún
procesamiento se ha
realizado)



▶ Separando un canal

```
import cv2

img = cv2.imread('lena.png')

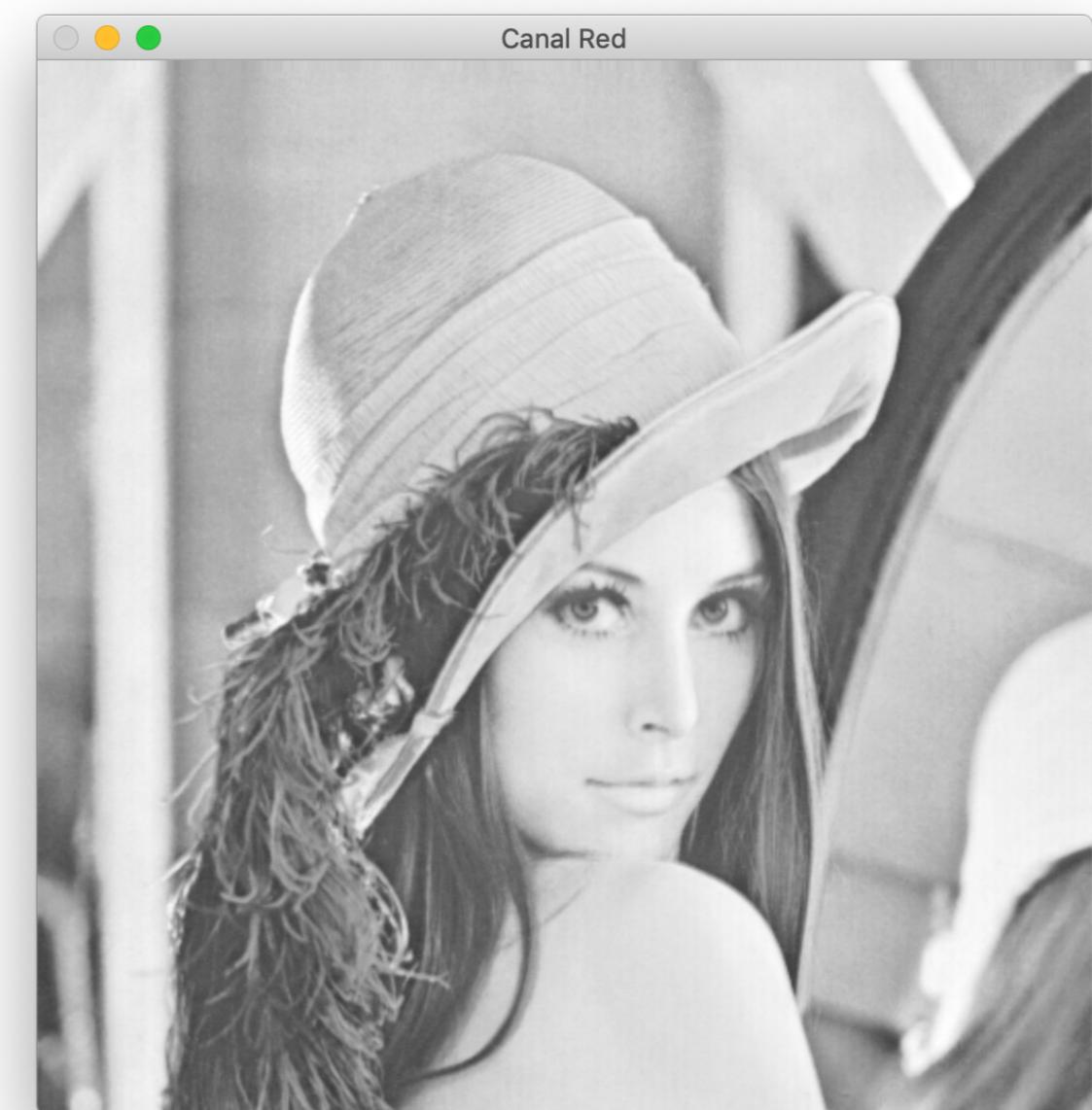
blue, green, red = cv2.split(img)

cv2.imshow('canal R', red)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

El comando split permite separar los canals de una imagen.
El orden es BGR



► Separando un canal

```
import cv2

img = cv2.imread('lena.png')

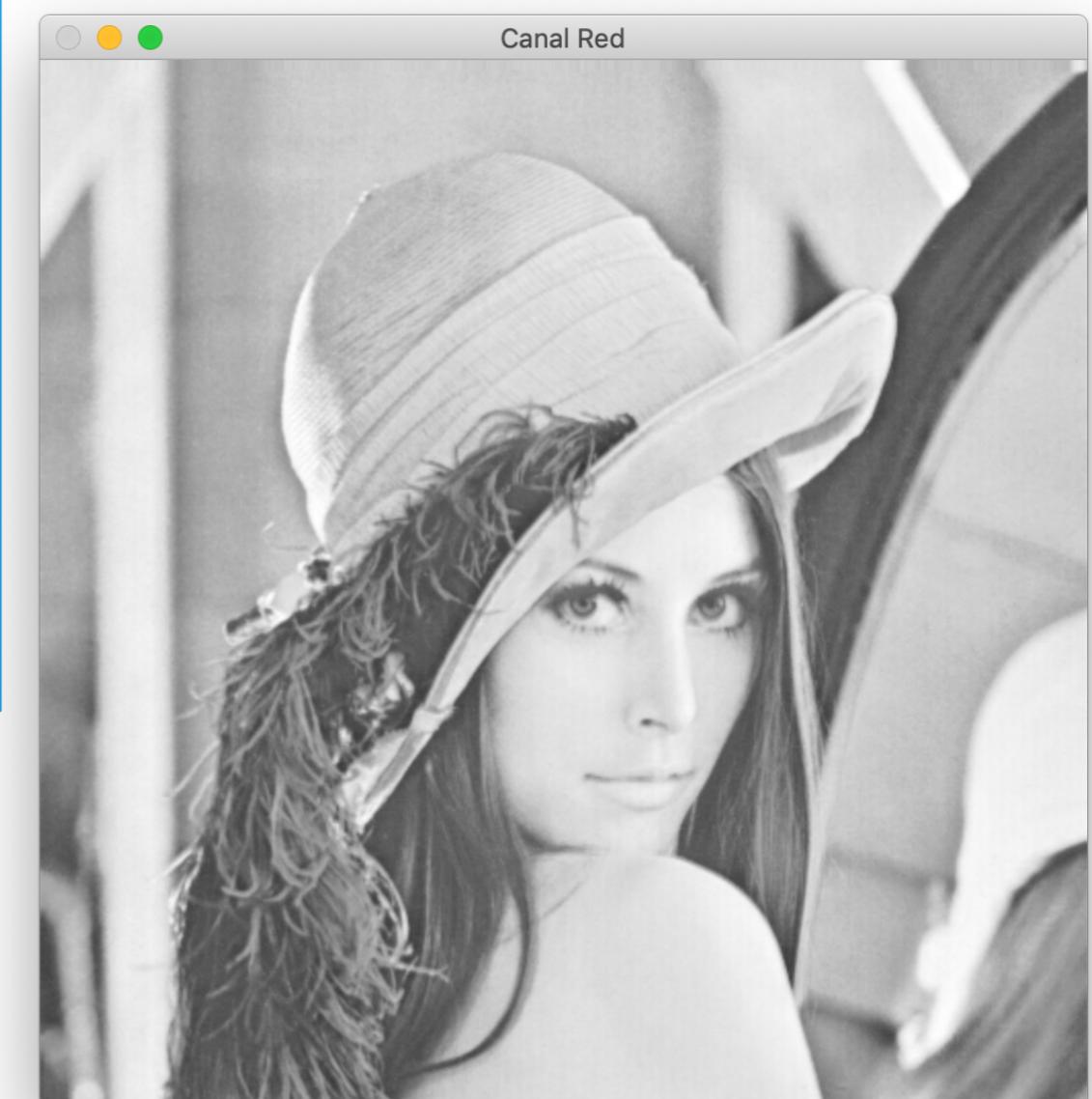
blue, green, red = cv2.split(img)

cv2.imshow('canal R', red)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

El comando split permite separar los canales de una imagen.
El orden es BGR



Ejercicio

Muestra una imagen de cada canal

- ▶ Analicemos la distribución de grises de un canal

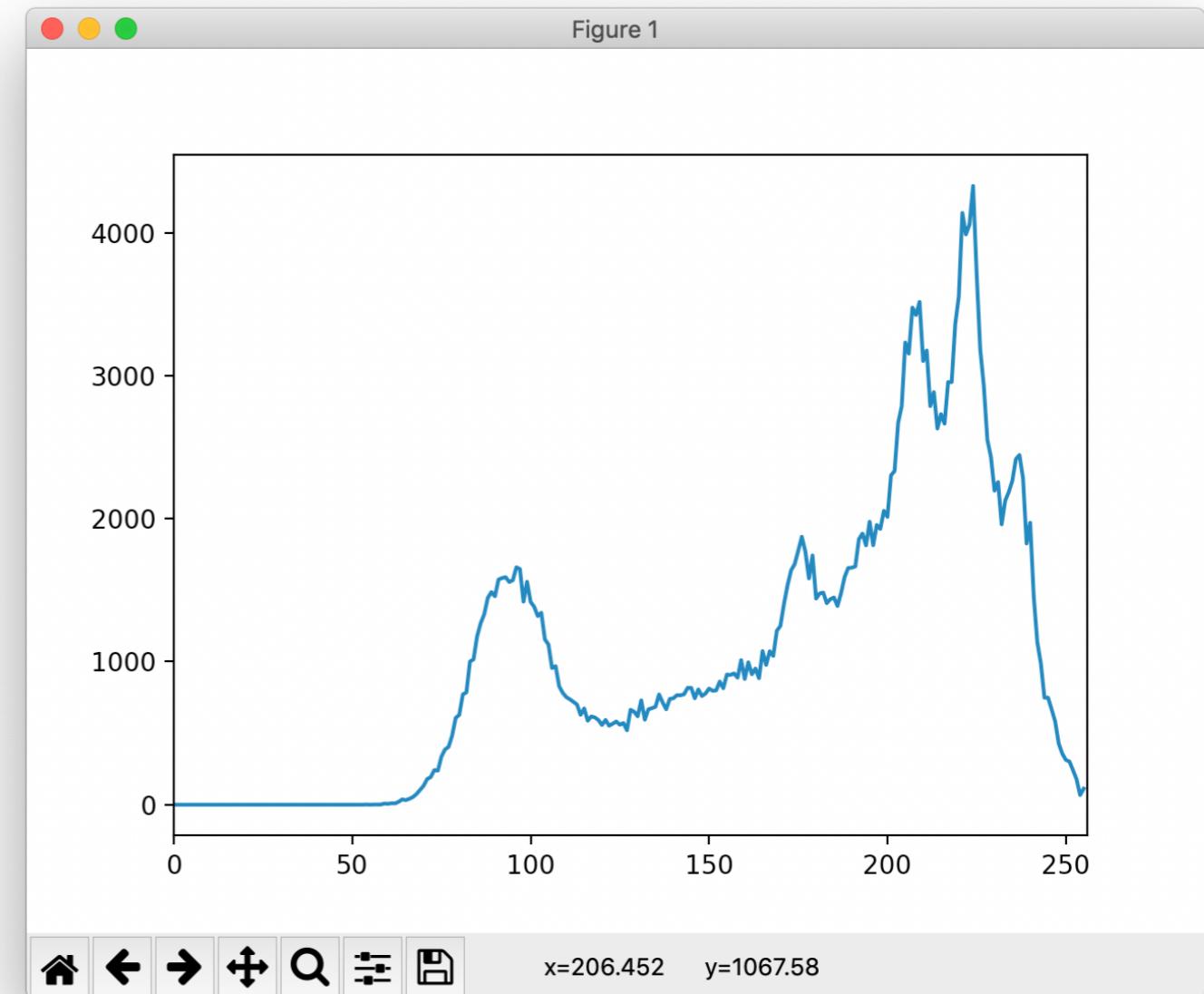
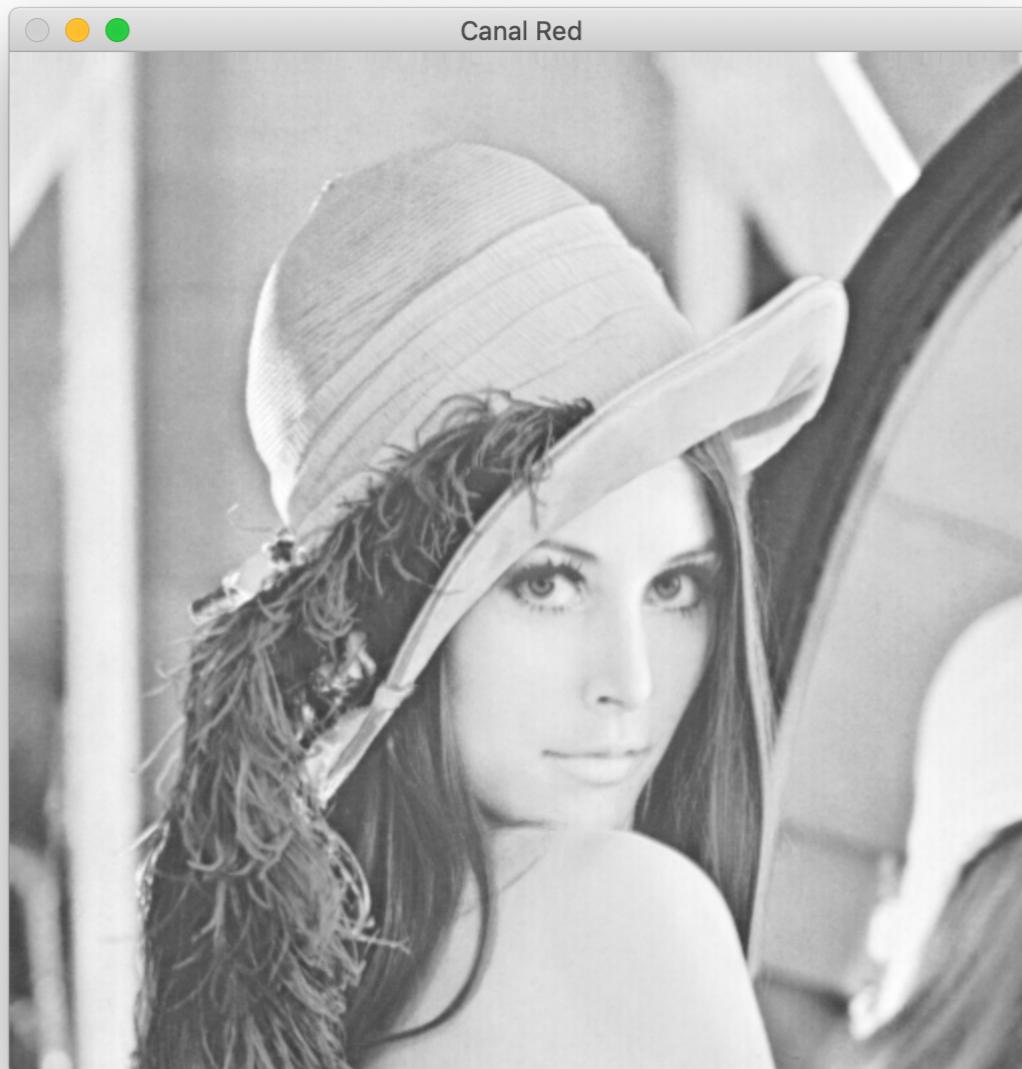
```
import cv2  
  
import matplotlib.pyplot as plt  
  
img = cv2.imread('lena.png')  
  
hist_red = cv2.calcHist([img], [2], None, [256], [0,256])  
  
plt.plot(hist_red)  
plt.xlim([0,256])  
plt.show()
```

La función calcHist nos permite calcular el histograma

The diagram consists of a horizontal blue line with five vertical green arrows pointing upwards from labels positioned below it. The labels are: "matriz" at the far left, "máscara" in the center, "niveles o bins" below the third arrow, and "rango" at the far right.

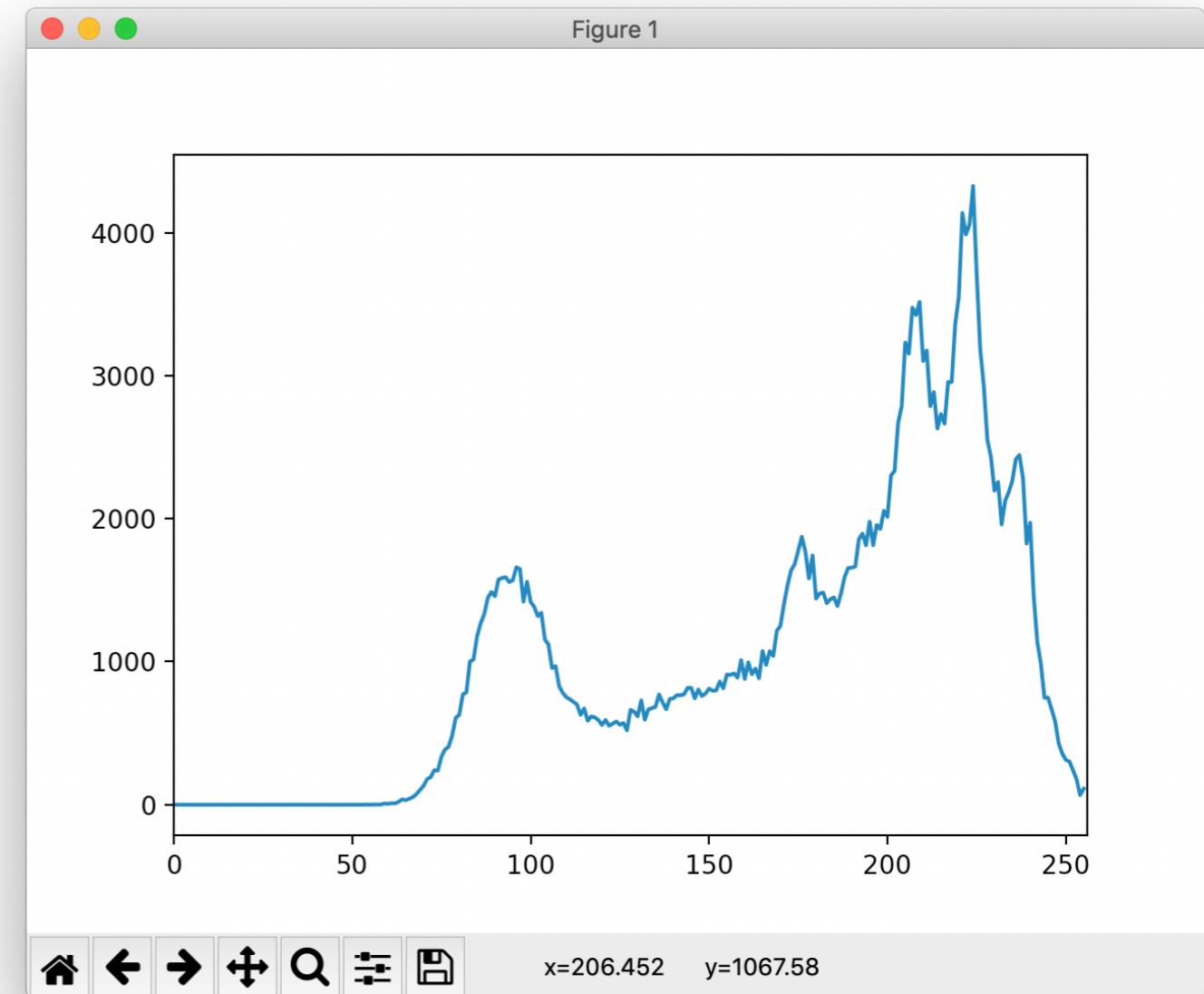
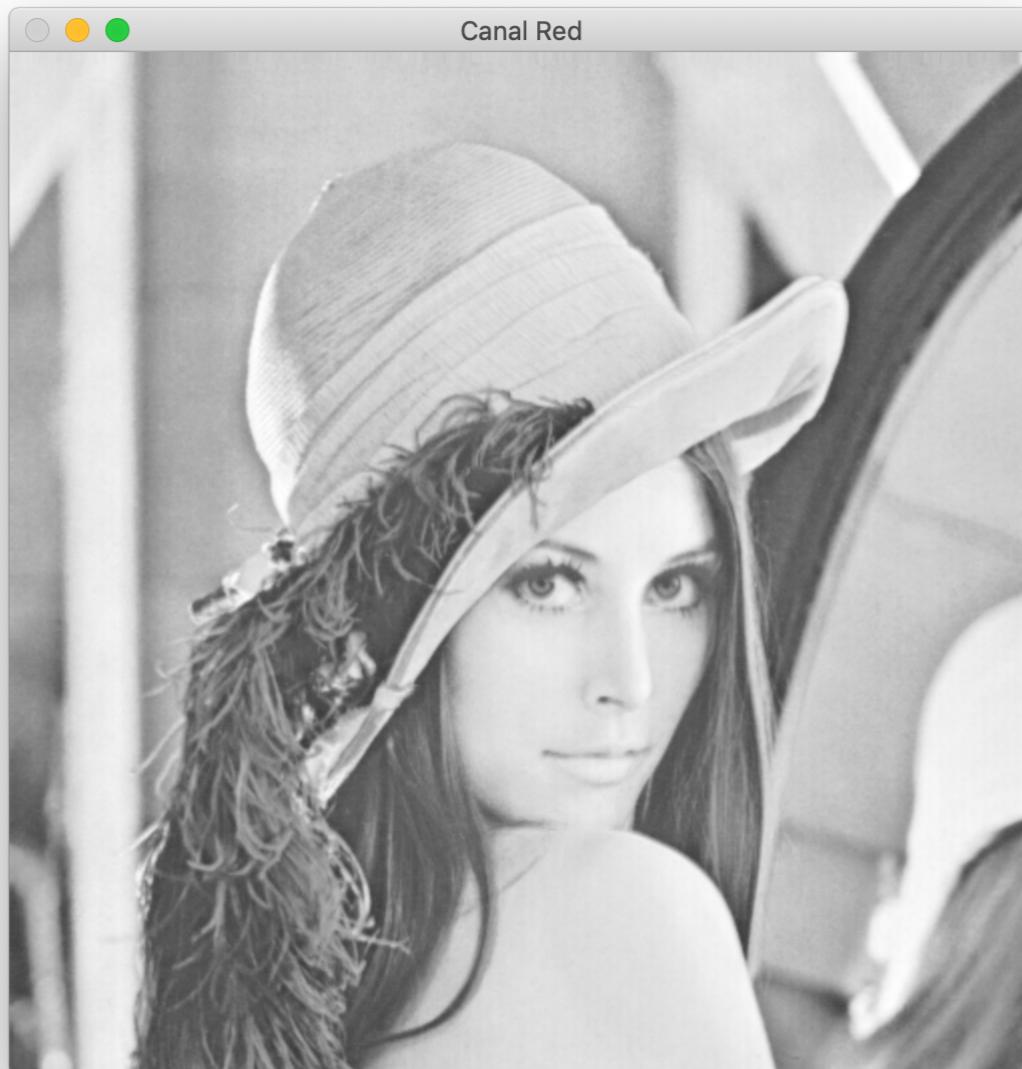
canal {0:blue, 1:green, 2:red}

- ▶ Analicemos la distribución de grises de un canal



¿qué representa esta curva del canal Rojo?, ¿será diferente para otros canales?

- ▶ Analicemos la distribución de grises de un canal



Ejercicio

Grafica para cada uno de los canals RGB su histograma

▶ OpenCV

- Ejemplo 1 | Lectura de una imagen
- Ejemplo 2 | Separación de canales
- Ejemplo 3 | Selección de un área de interés (ROI)
- Ejemplo 4 | Binarización



▶ Posiciones de los píxeles

```
import cv2

img = cv2.imread('lena.png')

dimensions = img.shape

height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]

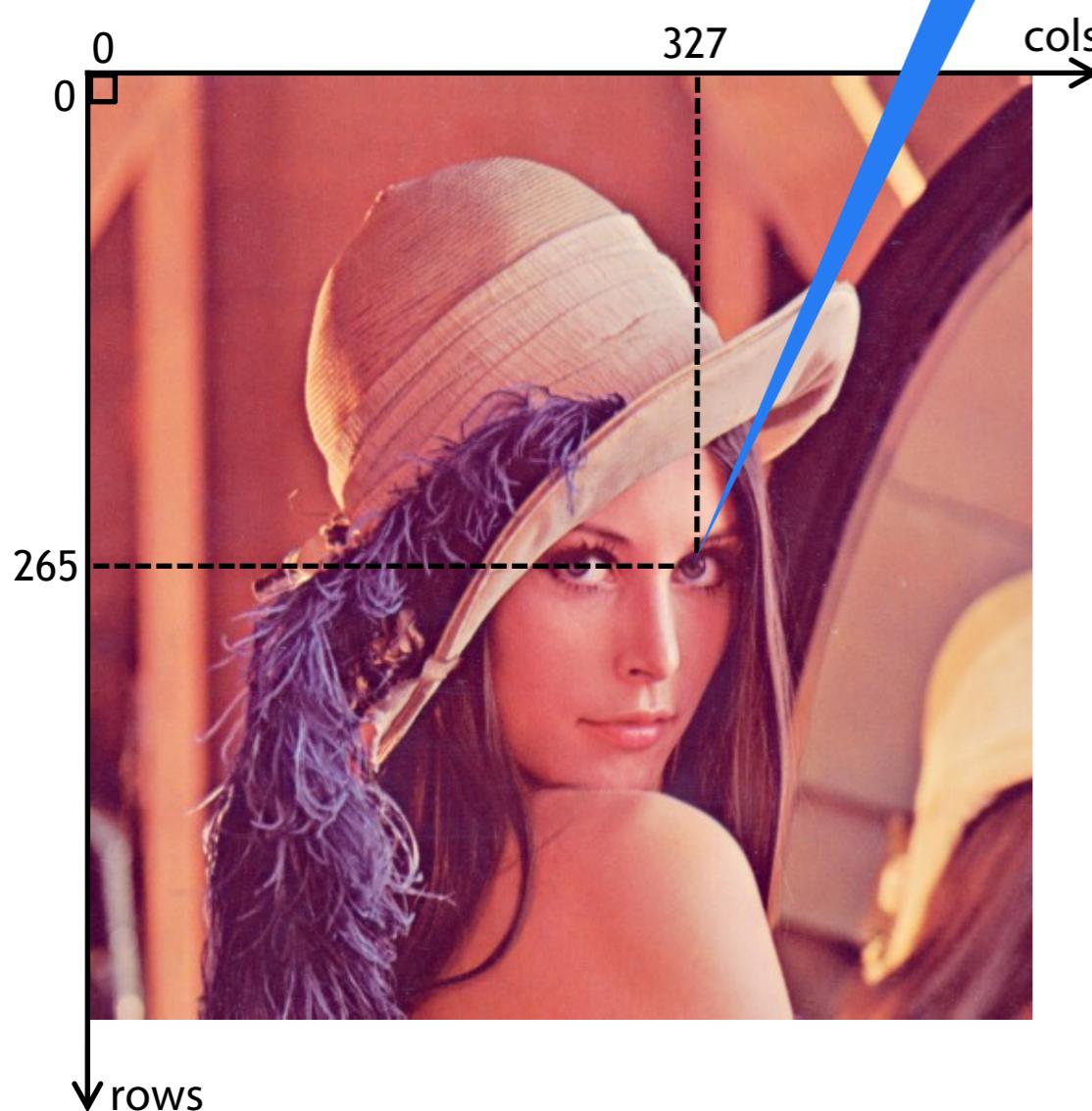
print('Image Dimension: ', dimensions)
print('Image Height : ', height)
print('Image Width : ', width)
print('Channels : ', channels)
```



el atributo shape nos permite determinar las dimensiones de la imagen



▶ Canales de una imagen



```
import cv2

img = cv2.imread('lena.png')

pixel = img[265,327,:]

print(pixel)
```

el operador : significa todas las dimensiones

output

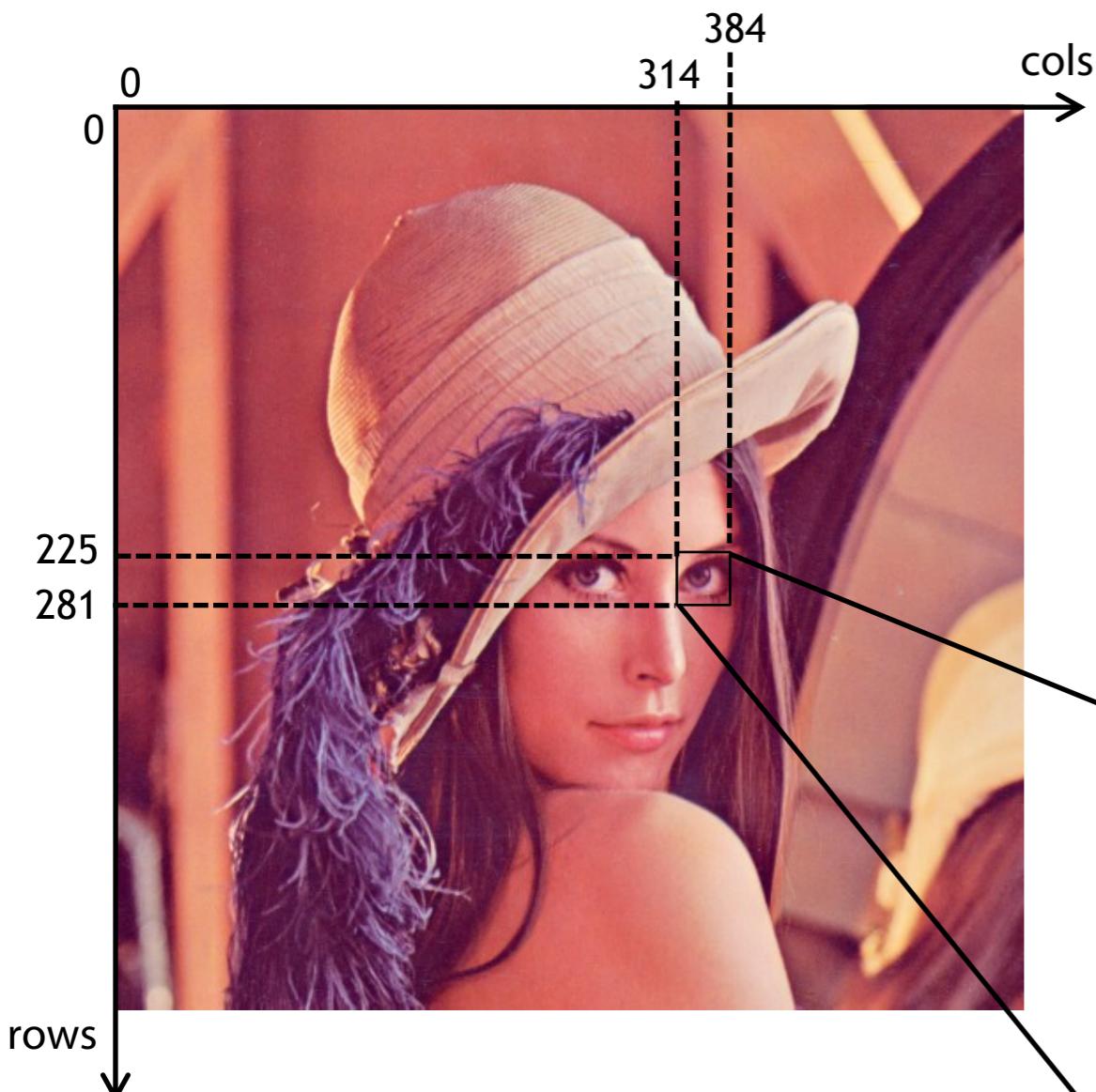
[67 16 88]

Canal B

Canal G

Canal R

▶ Sección de una imagen



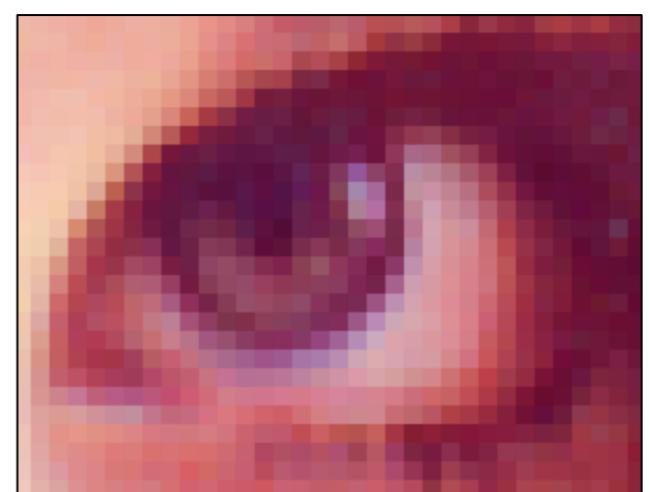
```
import cv2
img = cv2.imread('lena.png')

roi = img[255:281, 314:348, :]

cv2.imshow('image', roi)
cv2.waitKey(0)
```

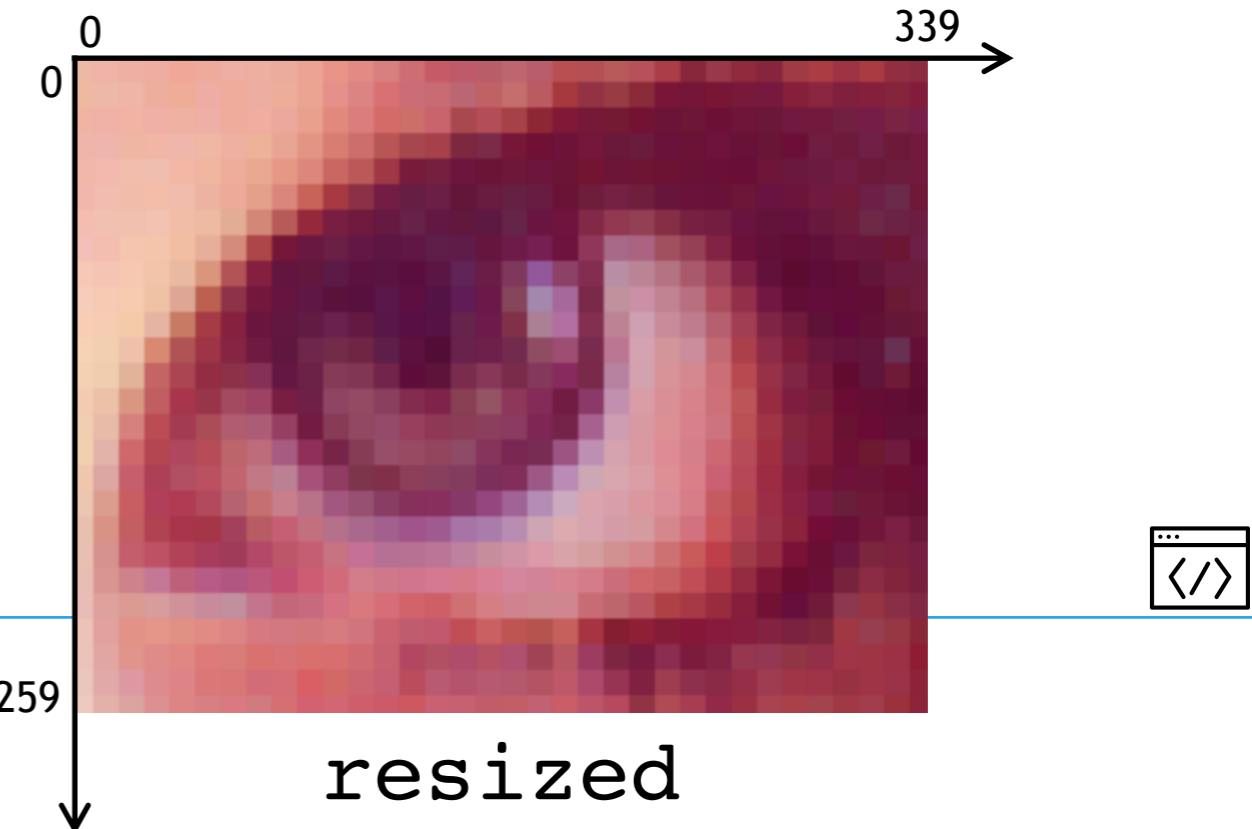


ROI (canal R)



ROI
(canal RGB)

- ▶ Resize de una imagen, o subimagen



```
import cv2
img = cv2.imread('lena.png')
roi = img[255:281,314:348,:]

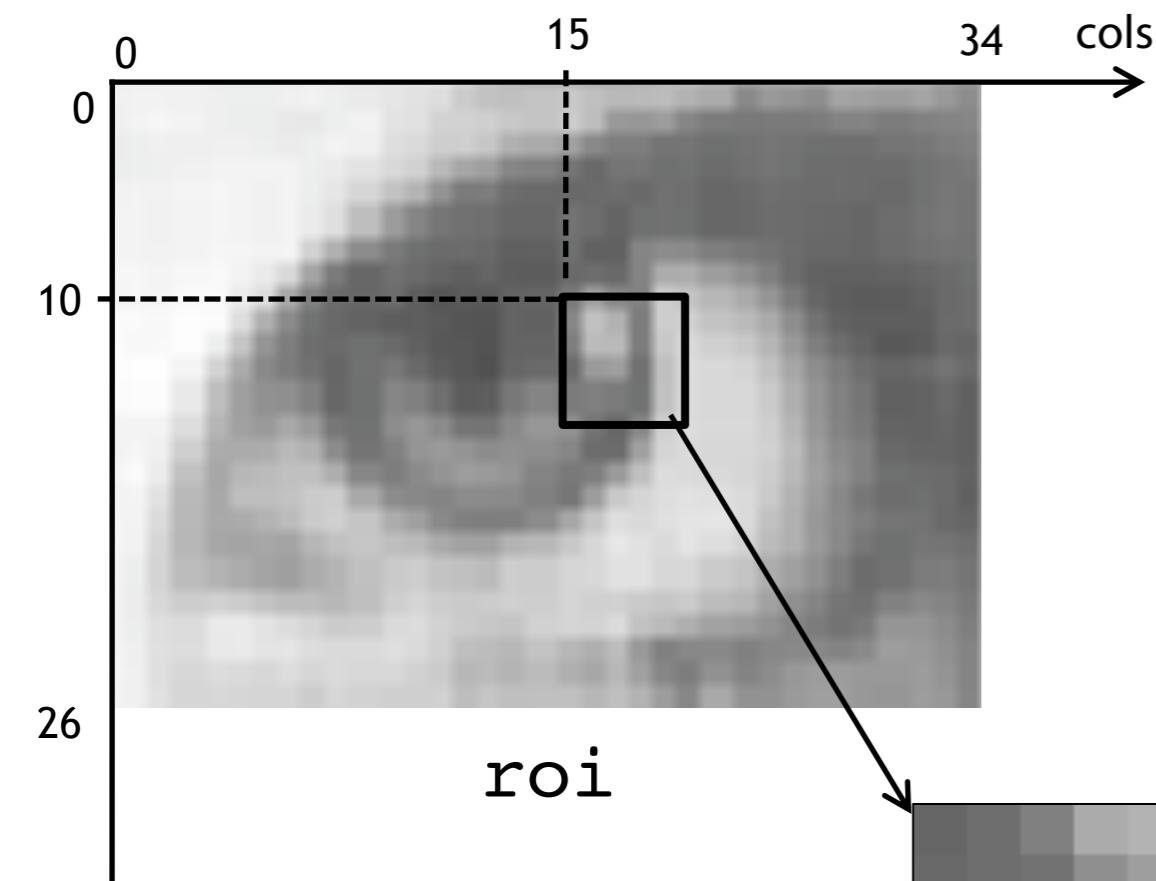
factor_times = 10
height = int(roi.shape[0] * factor_times)
width = int(roi.shape[1] * factor_times)
dim = (width, height)

resized= cv2.resize(roi, dim, interpolation=cv2.INTER_AREA)

cv2.imshow('image',resized)
cv2.waitKey(0)
```

el comando resize
modifica el tamaño de
una imagen

▶ Sección de una imagen



```
import cv2
img = cv2.imread('lena.png')
roi = img[255:281,314:348,2]

subroi = roi[10:20, 15:25]

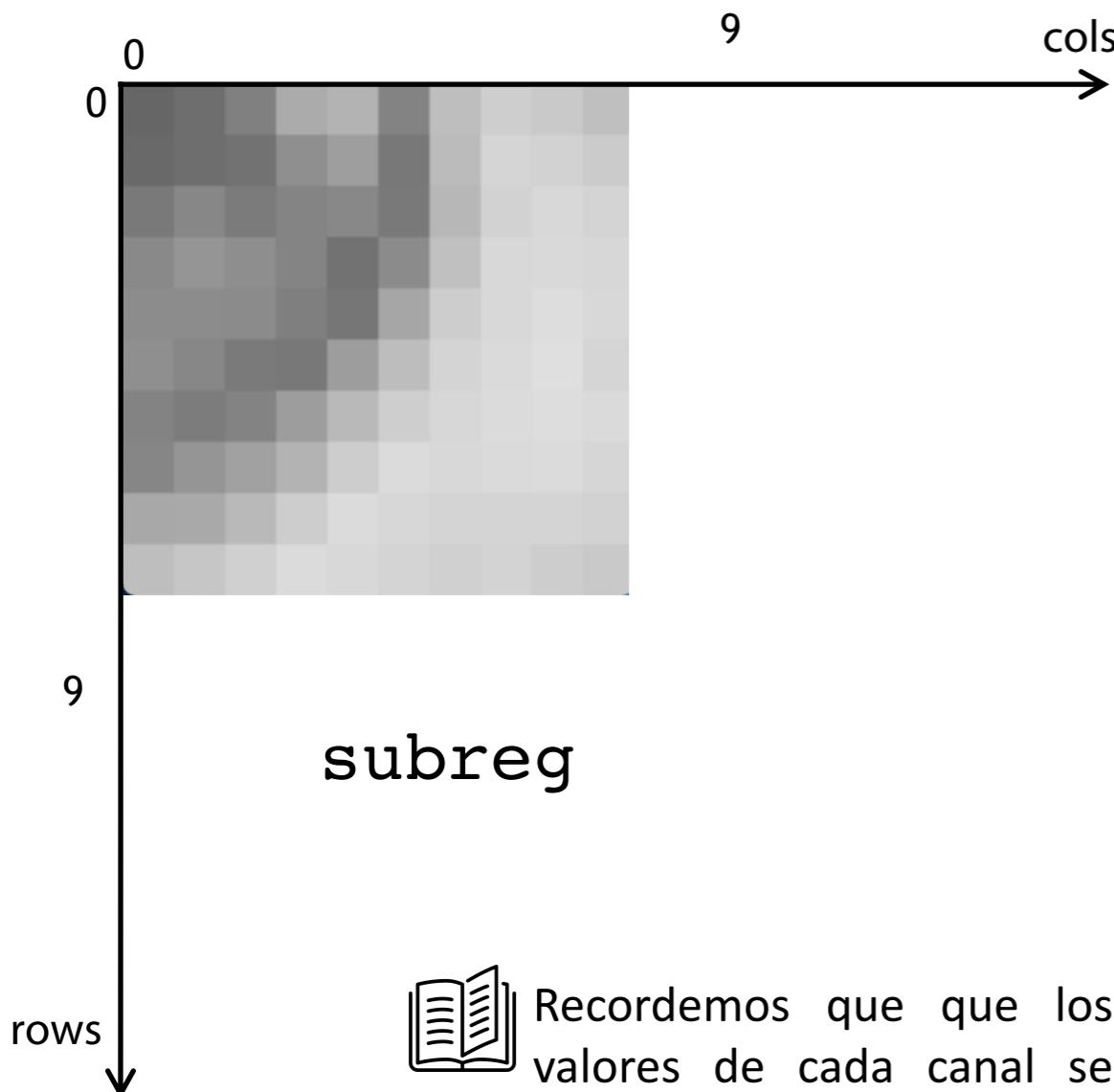
cv2.imshow('image', subroi)
cv2.waitKey(0)
```

Seleccionamos el canal a R

INICIO

FIN

▶ Valores en escala de grises



subreg



Recordemos que los valores de cada canal se encuentran en el rango de 0 a 255, que corresponde a un valor de 8 bits

```
import cv2
img = cv2.imread('lena.png')
roi = img[255:281,314:348,2]

subroi = roi[10:20, 15:25]
print(subroi)
```

output

```
[[102 110 128 171 179 131 190 206 201 191]
 [105 110 114 143 158 120 187 213 210 203]
 [121 135 123 132 136 121 183 210 216 212]
 [137 149 142 132 114 139 192 216 217 215]
 [140 140 139 127 118 166 205 216 221 216]
 [143 135 122 120 157 189 212 218 223 213]
 [131 124 131 157 185 206 215 219 221 218]
 [134 149 161 179 205 219 216 218 219 214]
 [168 169 185 205 219 215 212 212 212 210]
 [190 198 208 219 216 212 208 211 205 201]]
```

▶ OpenCV

- Ejemplo 1 | Lectura de una imagen
- Ejemplo 2 | Separación de canales
- Ejemplo 3 | Selección de un área de interés (ROI)
- Ejemplo 4 | Binarización



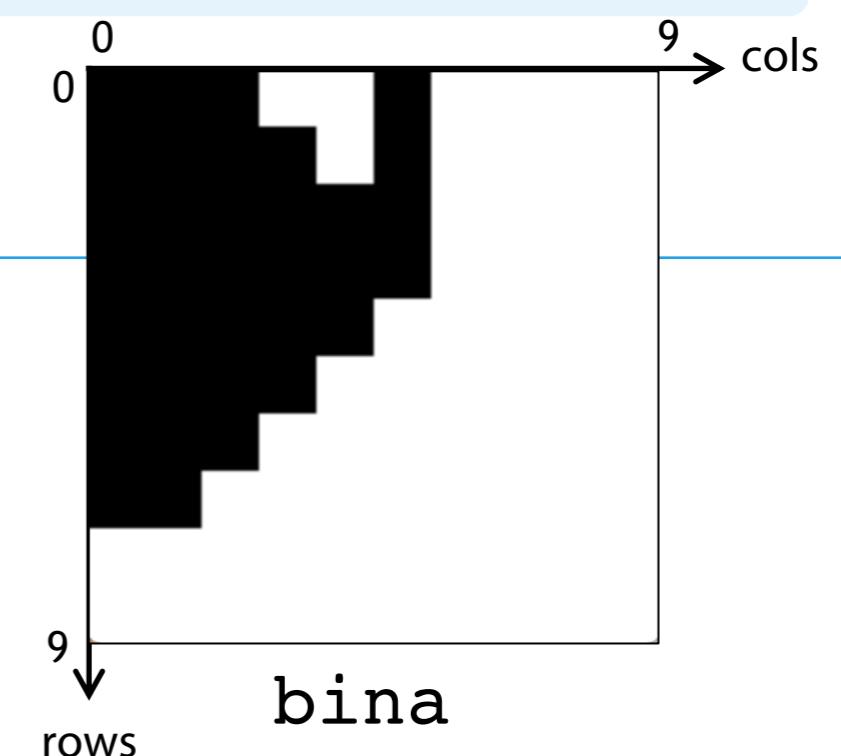
- ▶ Valores en escala binaria

```
import cv2
img = cv2.imread('lena.png')
roi = img[255:281,314:348,2]
subroi = roi[10:20, 15:25]

rt,bina = cv2.threshold(subroi,150,255,cv2.THRESH_BINARY)

cv2.imshow('image', bina)
cv2.waitKey(0)
```

La función threshold permite binarizar una imagen. Más adelante veremos más opciones de esta función

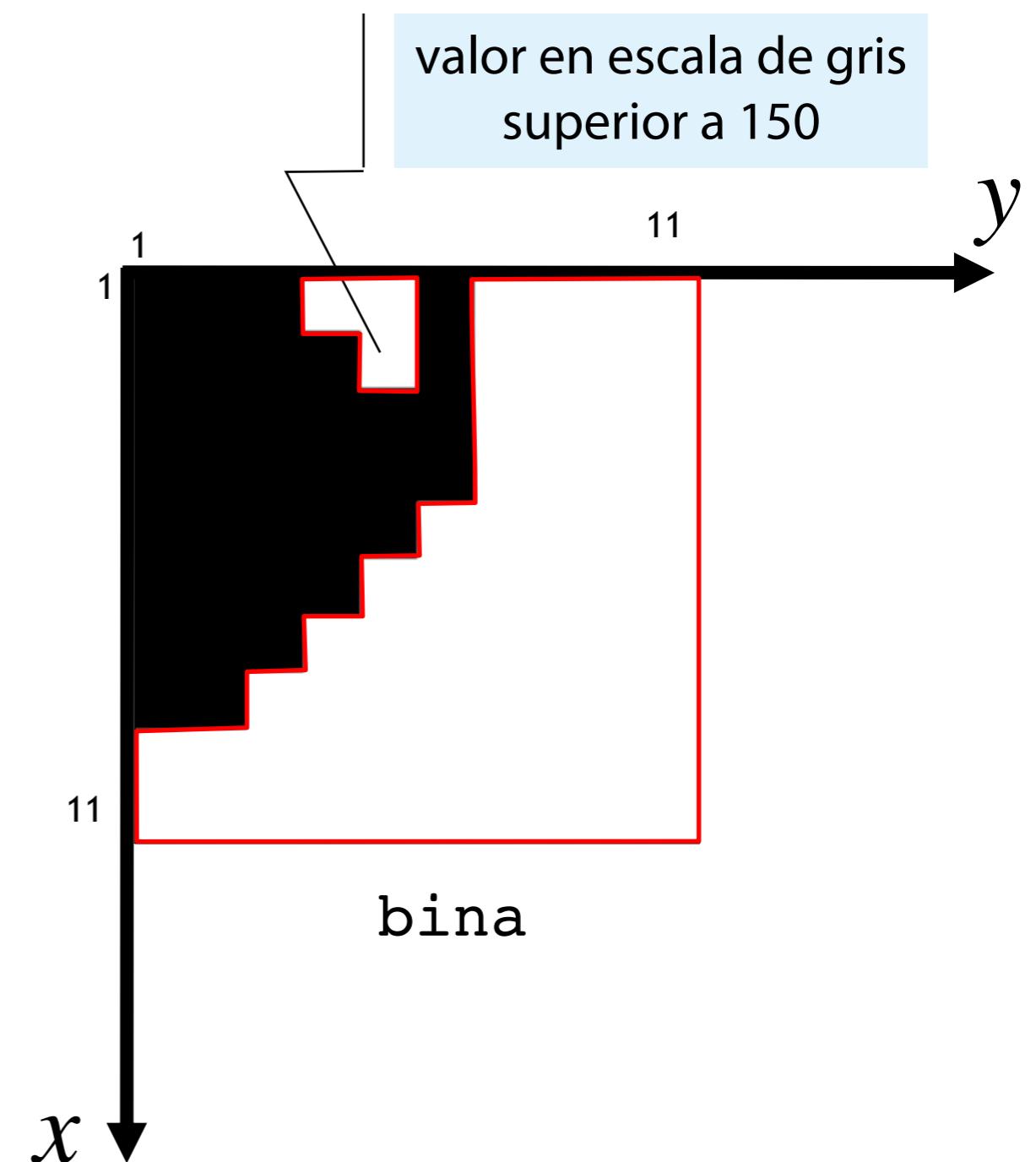
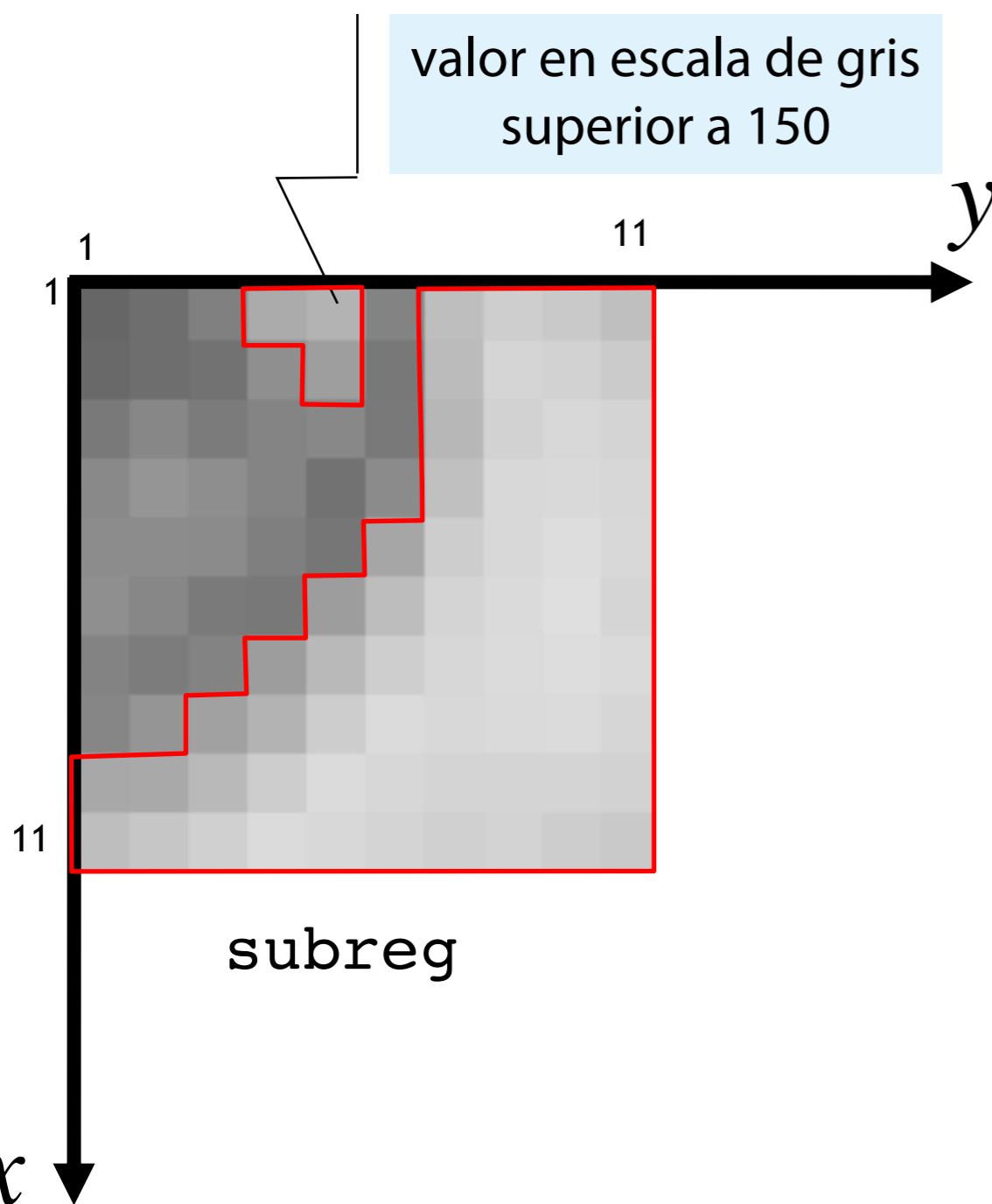


Ejercicio

Binarice el ojo derecho.

Seleccione el umbral que usted deseé

- Comparando dos imágenes



- ▶ Mejoramiento en el espacio
 - Transformaciones básicas en niveles de grises
 - Procesamiento de histogramas
 - Operaciones aritméticas y lógicas
 - Filtros espaciales (lineal y no lineal)

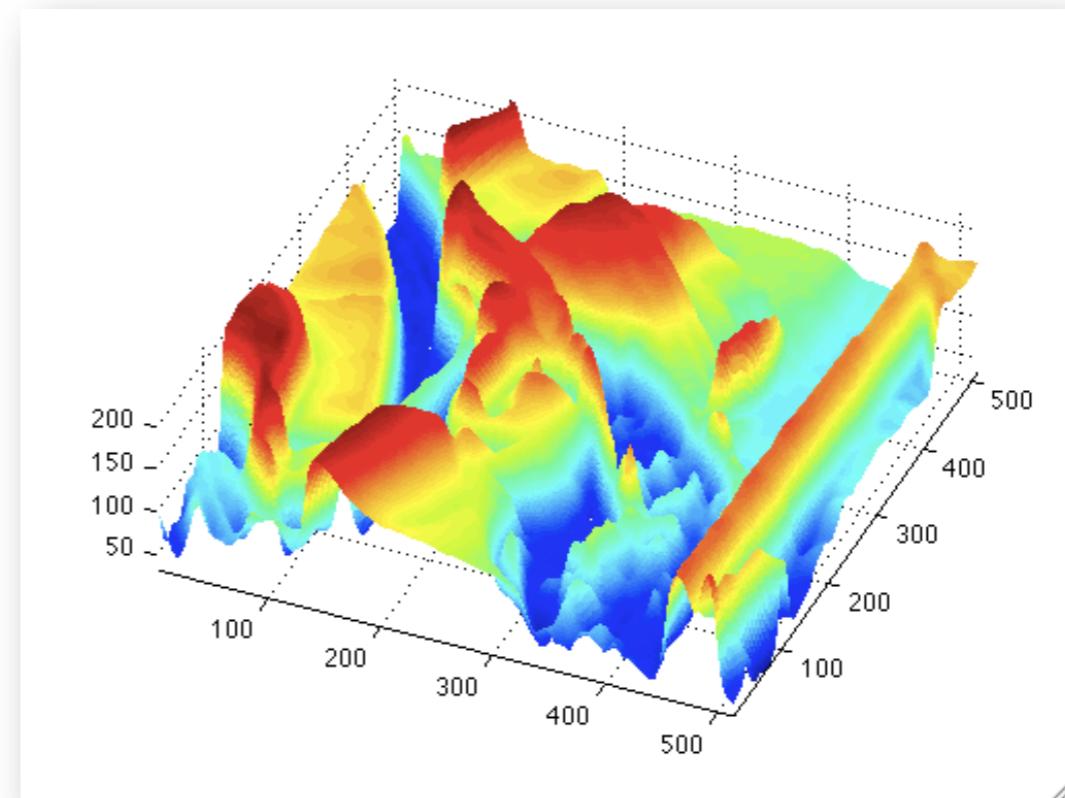
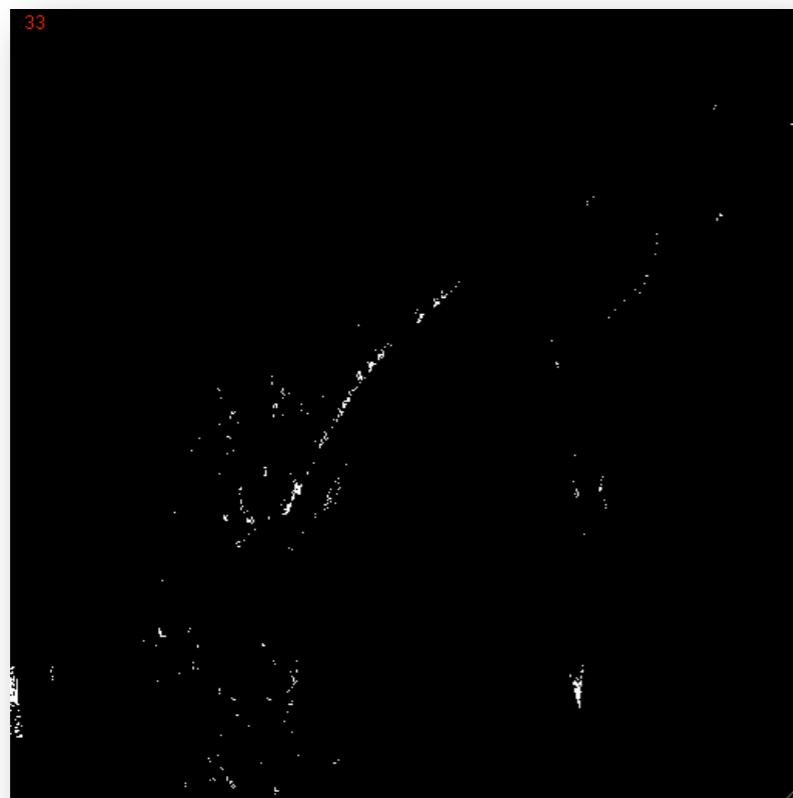


Mejoramiento en el espacio

Transformaciones en niveles de grises



Variemos el nivel de umbral de la imagen de Lena desde 0 a 255



▶ Inverso de una imagen



cameraman.jpg

```
import cv2

img = cv2.imread('cameraman.png')

cv2.imshow('image', img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



No olvides. Debes descargar una imagen (jpeg, png, bmp) en la misma carpeta donde se encuentre **tú script de Python**

Inverso de una imagen



cameraman.tif

Sabemos que los valores RGB van de [0 a 255], cómo calculamos su inverso?

```
import cv2  
  
img = cv2.imread('cameraman.png')  
  
print(img[0:5,0:5,0])
```

output

```
[[156 159 158 155 158]  
 [160 154 157 158 157]  
 [156 159 158 155 158]  
 [160 154 157 158 157]  
 [156 153 155 159 159]]
```

Objetivo:

Que el blanco sea negro y el negro sea blanco

▶ Inverso de una imagen



cameraman.jpg

```
import cv2

img = cv2.imread('cameraman.png')

neg = 255-img

cv2.imshow('image', neg)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

El valor 255 se resta a
cada valor de la matriz



▶ Inverso de una imagen

```
import cv2

img = cv2.imread('cameraman.png')

neg = 255-img

cv2.imshow('inverso', neg)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



Imagen negativa

- ▶ Inverso de una imagen

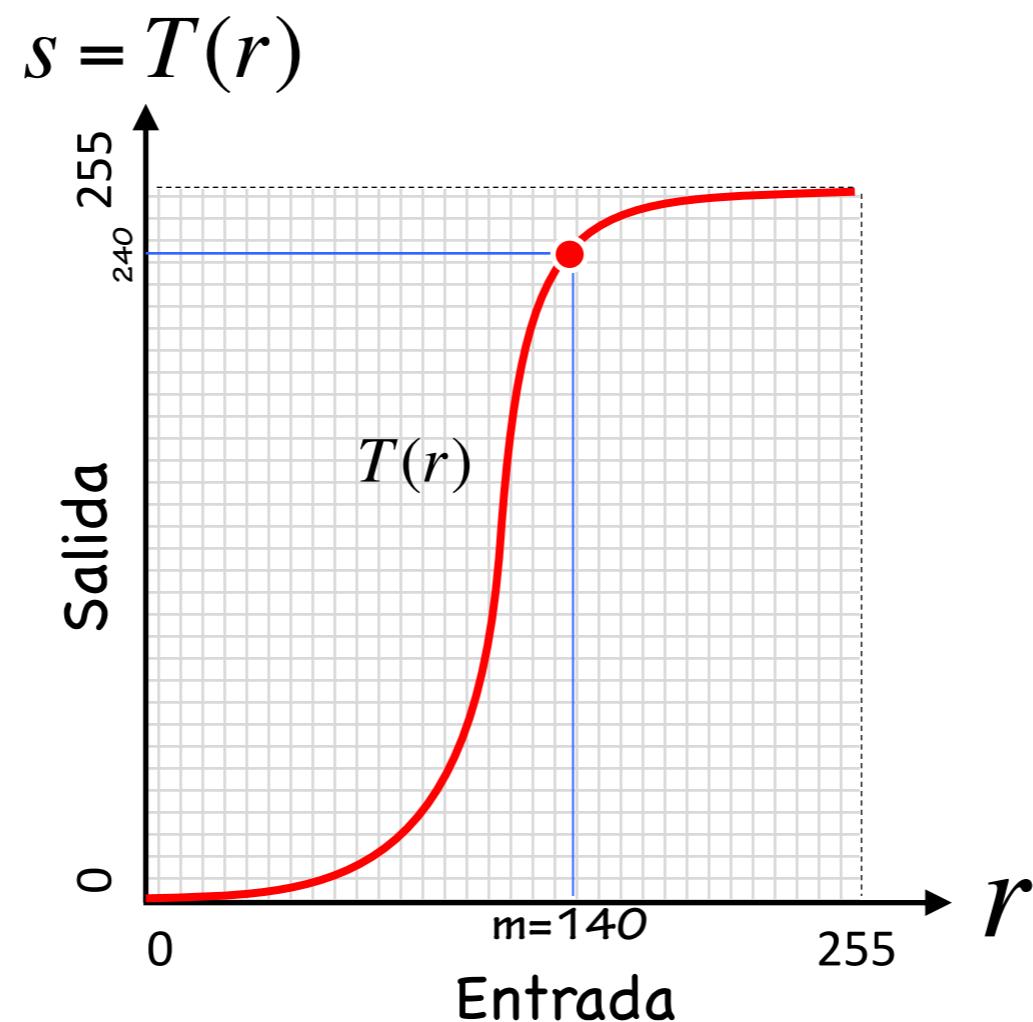


Imagen original

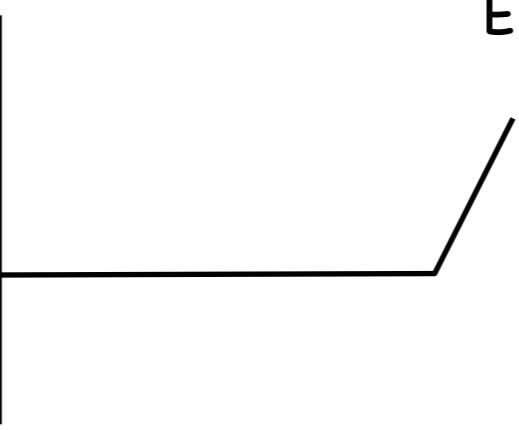


Imagen negativa

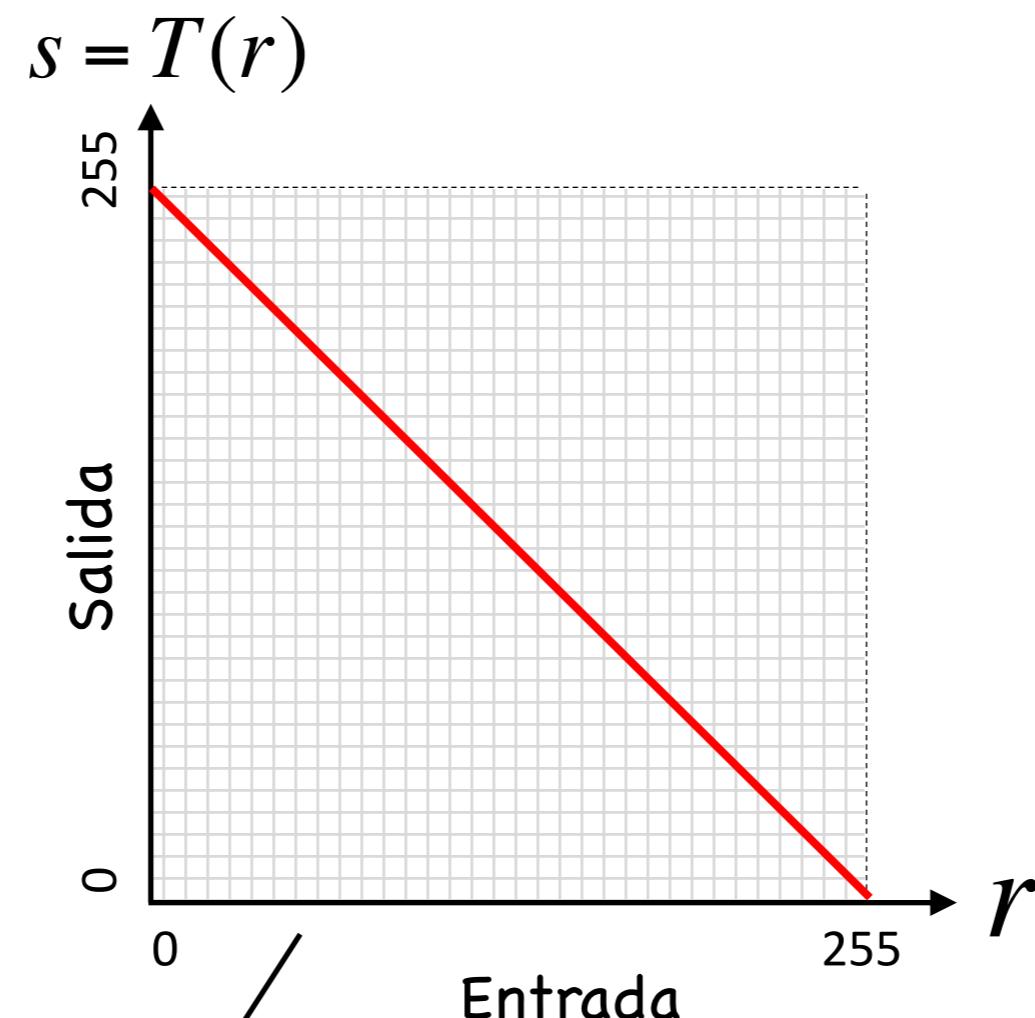
▶ Función de transformación



Supongamos que ingresa el valor de gris $m=140$. Empleando la curva $T(r)$ obtenemos el valor de salida 240



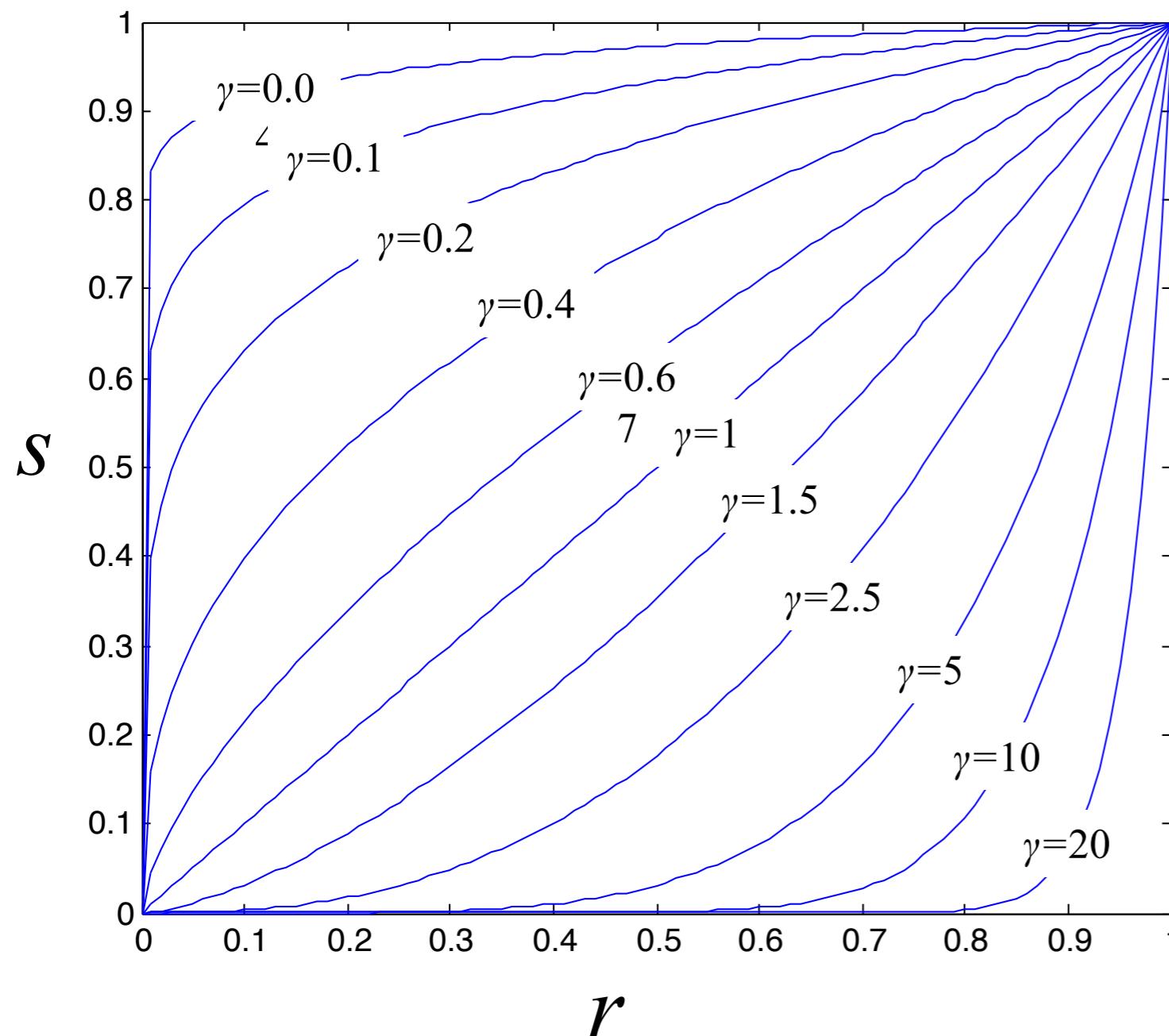
▶ Función de transformación



Qué imagen obtenemos al aplicar esta curva?



▶ Corrección gama

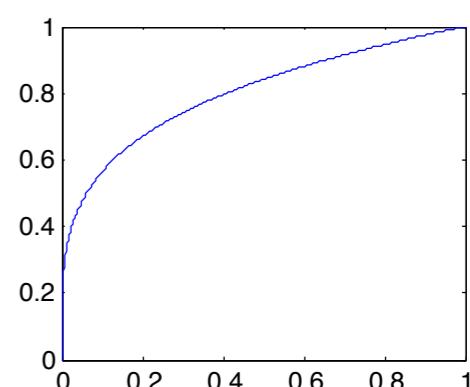
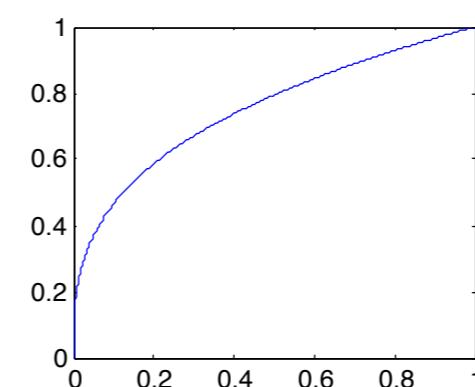
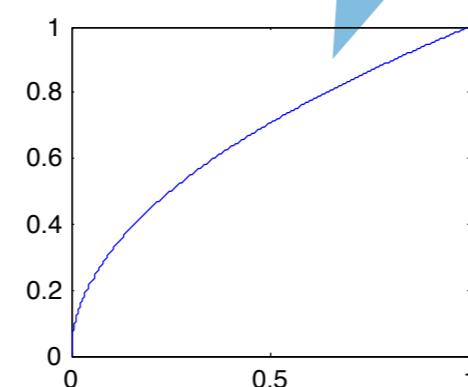
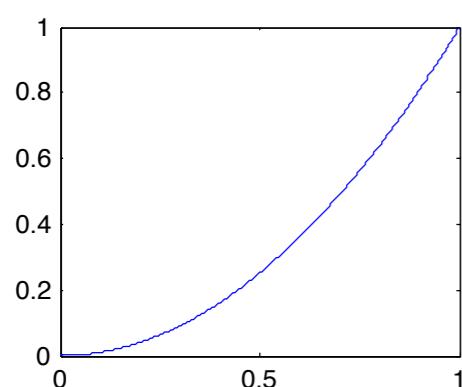


Ecuación de curva gama

$$S = r^\gamma$$

▶ Corrección gama

El efecto de corrección muestra cómo a mayor número de gama, la imagen se oscurece. Al contrario, cuando éste es menor a uno, las áreas oscuras se aclaran



$\gamma=2$

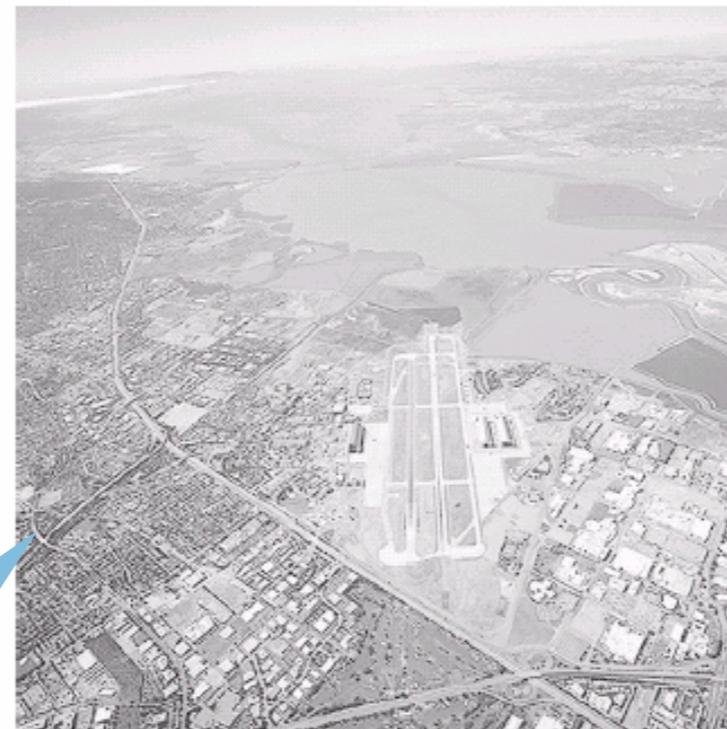
$\gamma=1/2$

$\gamma=1/3$

$\gamma=1/4$

▶ Corrección gama

$$\gamma = 1$$



Si la imagen es muy clara ¿qué valor de gama debemos emplear?

$$\gamma = 4$$



$$\gamma = 3$$

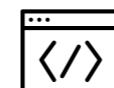


$$\gamma = 5$$



▶ Corrección gama

```
import cv2
import numpy as np
```



**librería de
manejo
numérico**

```
img = cv2.imread('cameraman.png')
```

```
output = gamma_correction(img, 5)
```

**llamado a la
función**

▶ Corrección gama

```
import cv2
import numpy as np

def gamma_correction(img, factor):
    img = img/255.0

img = cv2.imread('cameraman.png')

output = gamma_correction(img, 5)
```



función

el programa comienza a funcionar dentro de la función

llamado a la función

entregamos como parámetros la imagen, y el factor gamma

▶ Corrección gama

```
import cv2
import numpy as np

def gamma_correction(img, factor):
    img = img/255.0
    img = cv2.pow(img, factor)

img = cv2.imread('cameraman.png')

output = gamma_correction(img, 5)
```



función

el programa comienza a funcionar dentro de la función

llamado a la función

entregamos como parámetros la imagen, y el factor gamma

▶ Corrección gama

```
import cv2
import numpy as np

def gamma_correction(img, factor):
    img = img/255.0
    img = cv2.pow(img, factor)
    return np.uint8(img*255)

img = cv2.imread('cameraman.png')

output = gamma_correction(img, 5)
```



función

el programa comienza a funcionar dentro de la función

llamado a la función

entregamos como parámetros la imagen, y el factor gamma

▶ Corrección gama



```
import cv2
import numpy as np

def gamma_correction(img, factor):
    img = img/255.0
    img = cv2.pow(img, factor)
    return np.uint8(img*255)

img = cv2.imread('cameraman.png')

output = gamma_correction(img, 5)
```

función

Una vez que se alcanza `return`, se retorna el resultado

▶ Corrección gama



```
import cv2
import numpy as np

def gamma_correction(img, factor):
    img = img/255.0
    img = cv2.pow(img, factor)
    return np.uint8(img*255)

img = cv2.imread('cameraman.png')

output = gamma_correction(img, 5)

cv2.imshow('gamma', output)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

▶ Corrección gama

```
import cv2
import numpy as np

def gamma_correction(img, factor):
    img = img/255.0
    img = cv2.pow(img, factor)
    return np.uint8(img*255)
```

```
img = cv2.imread('cameraman.png')
```

```
output = gamma_correction(img, 5)
```

```
cv2.imshow('gamma', output)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```



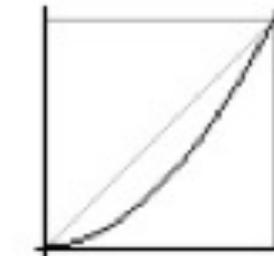
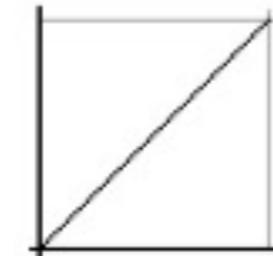
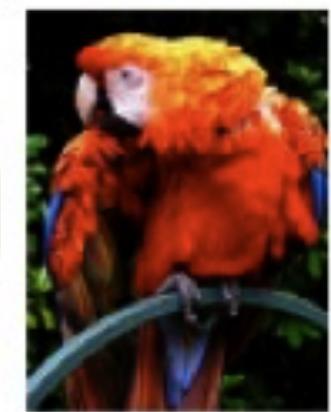
Ejercicio

Modifica el factor de la función gamma, para obtener una imagen más clara.

▶ Corrección gama



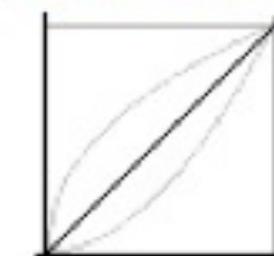
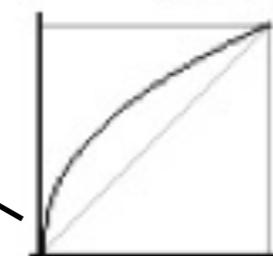
$$\gamma = 1.0$$



La corrección gama fue inicialmente desarrollada para compensar la relación input/output de un tubo CRT



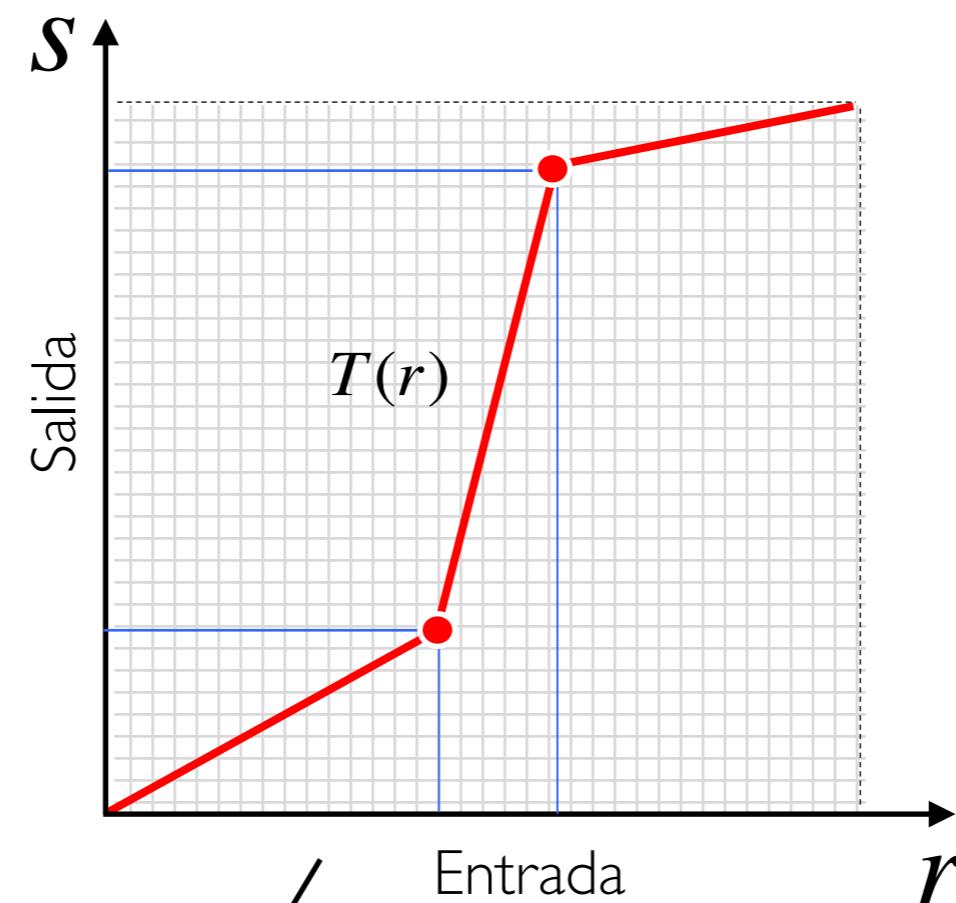
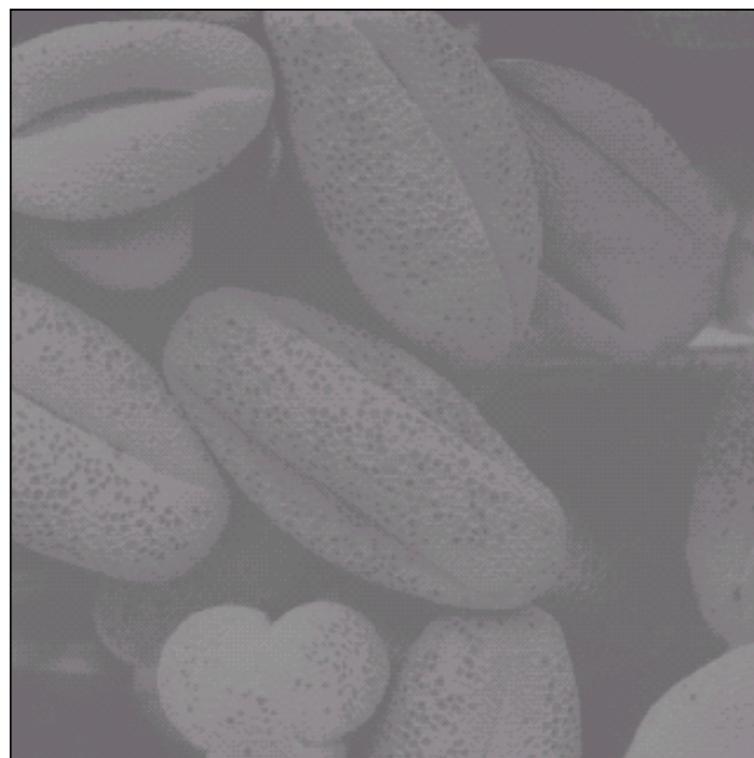
$$\gamma = 1/2.2$$



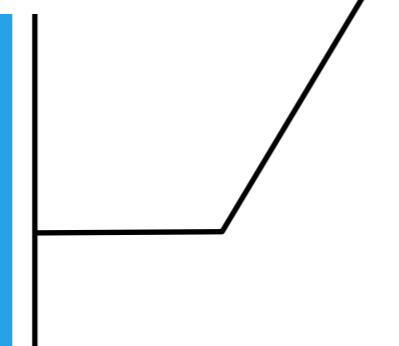
Teóricamente la imagen de captura de la cámara debiera ser proporcional a la imagen desplegada por el monitor

Images from Wikipedia

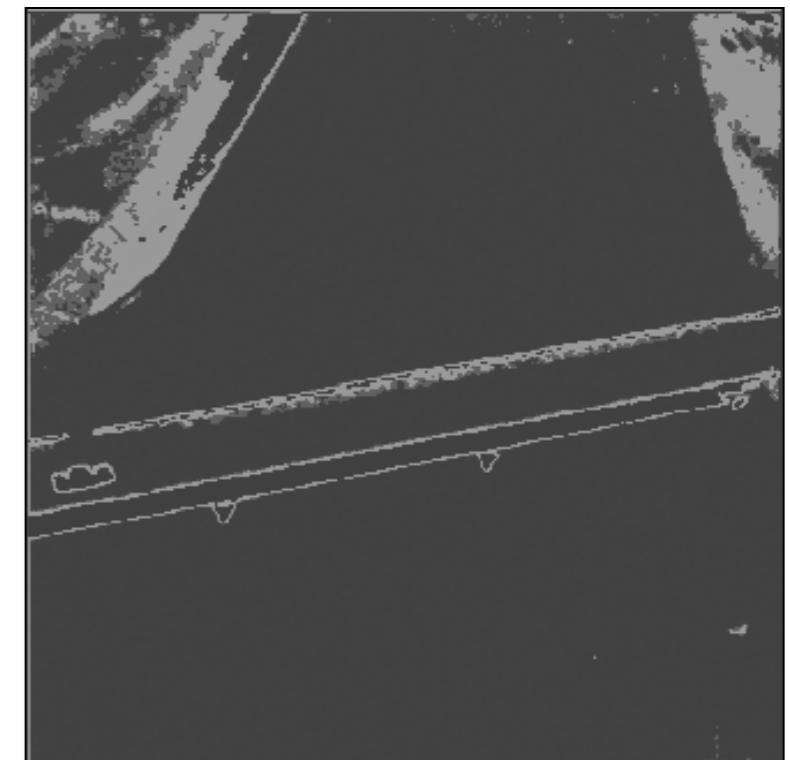
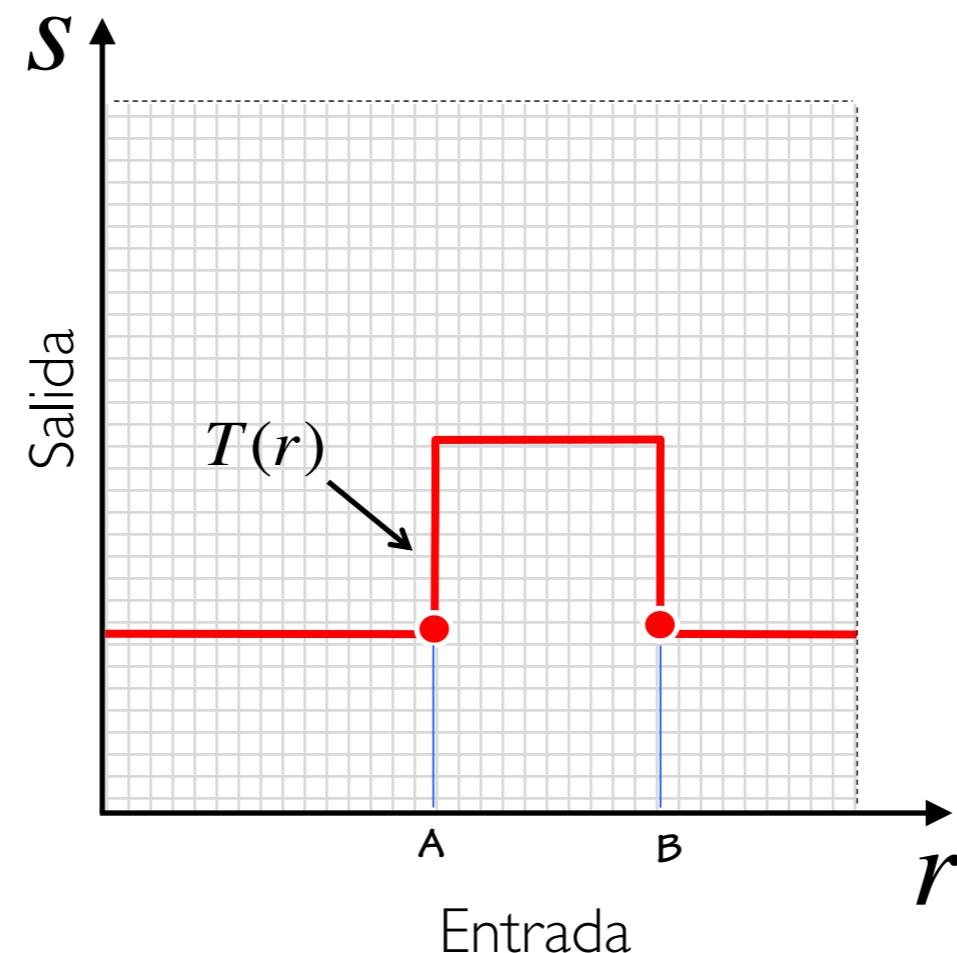
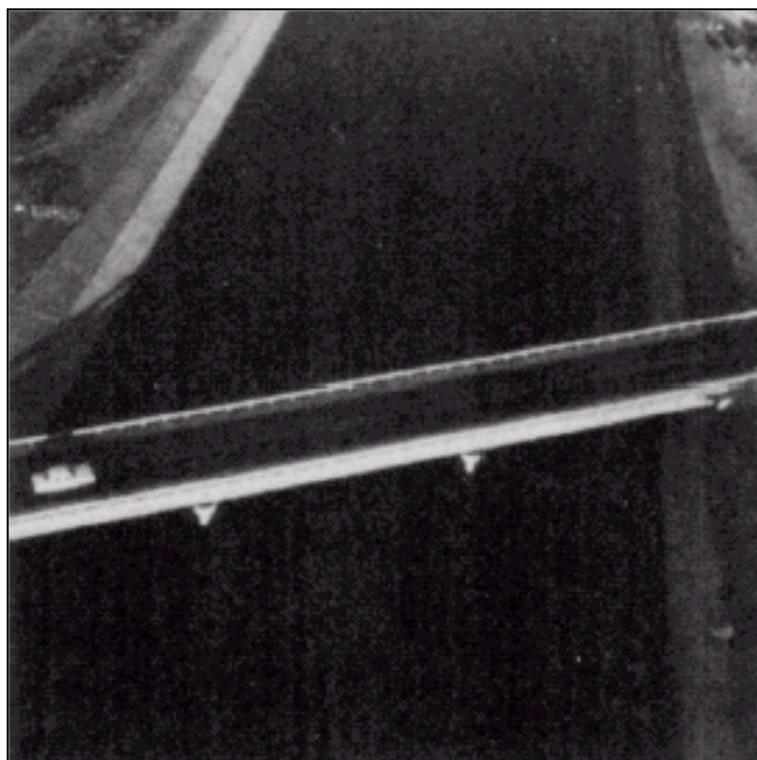
- ▶ Estiramiento de contraste (stretching)



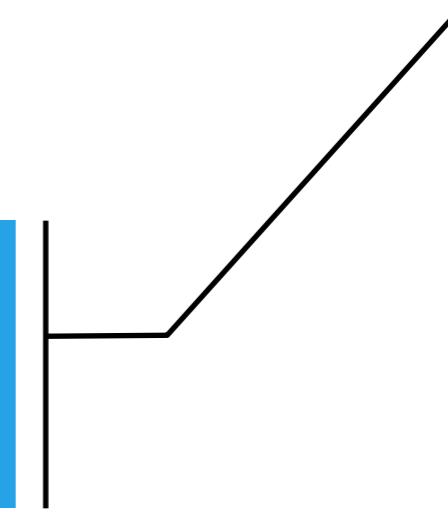
Al aplicar este estiramiento, ajustamos ciertos niveles de gris más que otros.



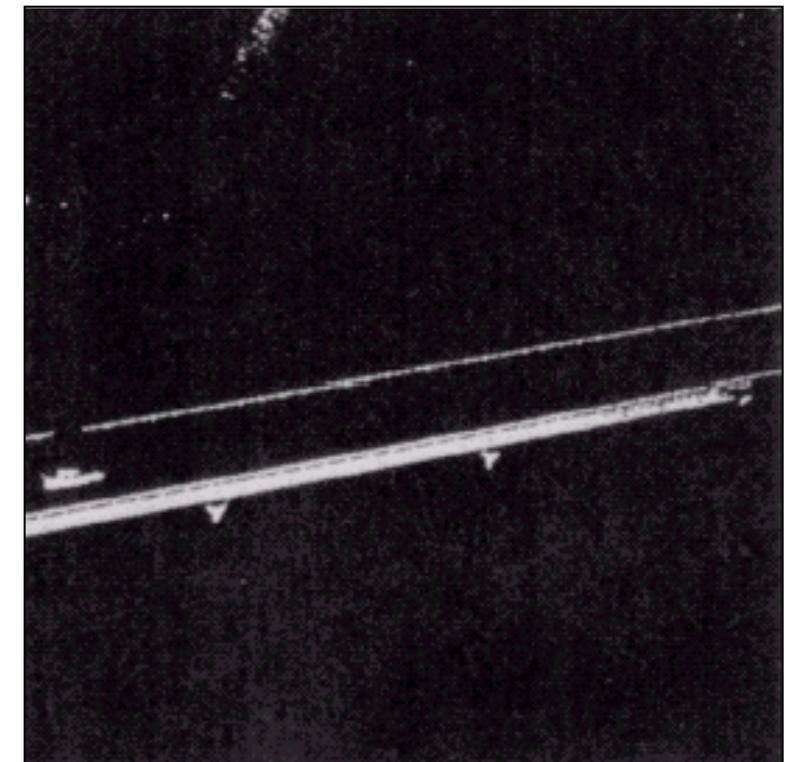
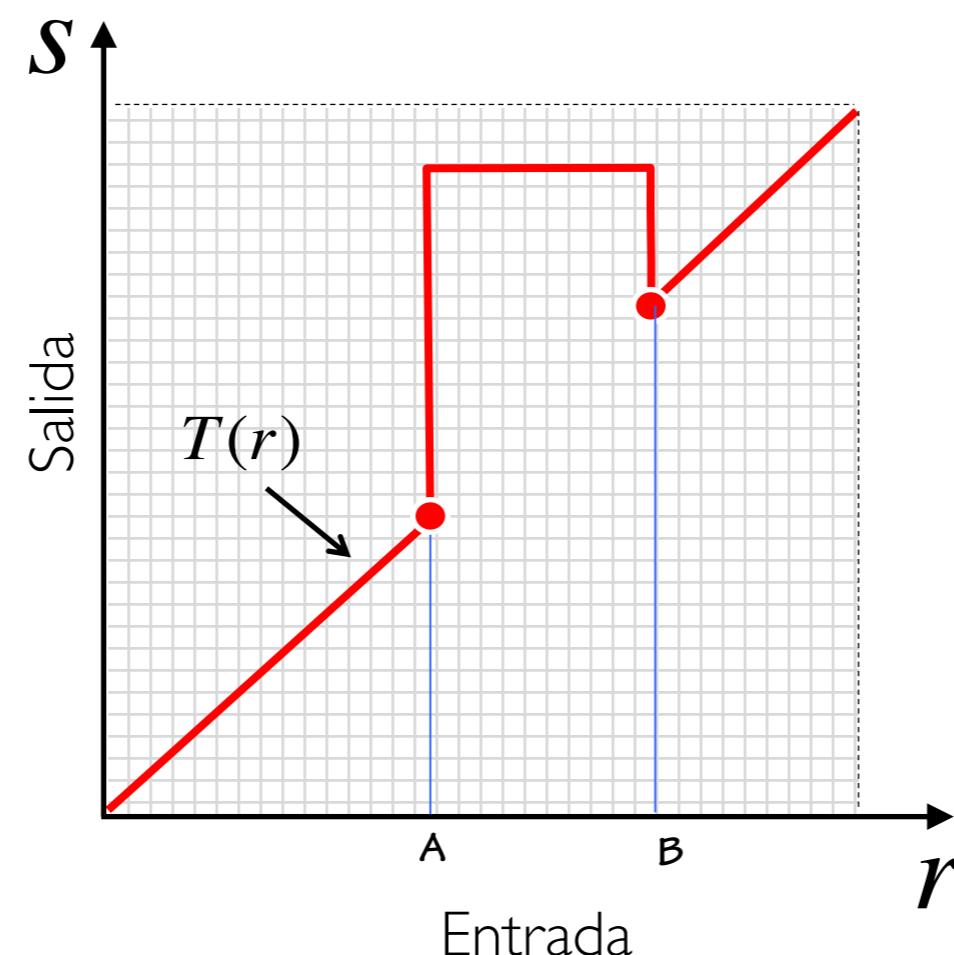
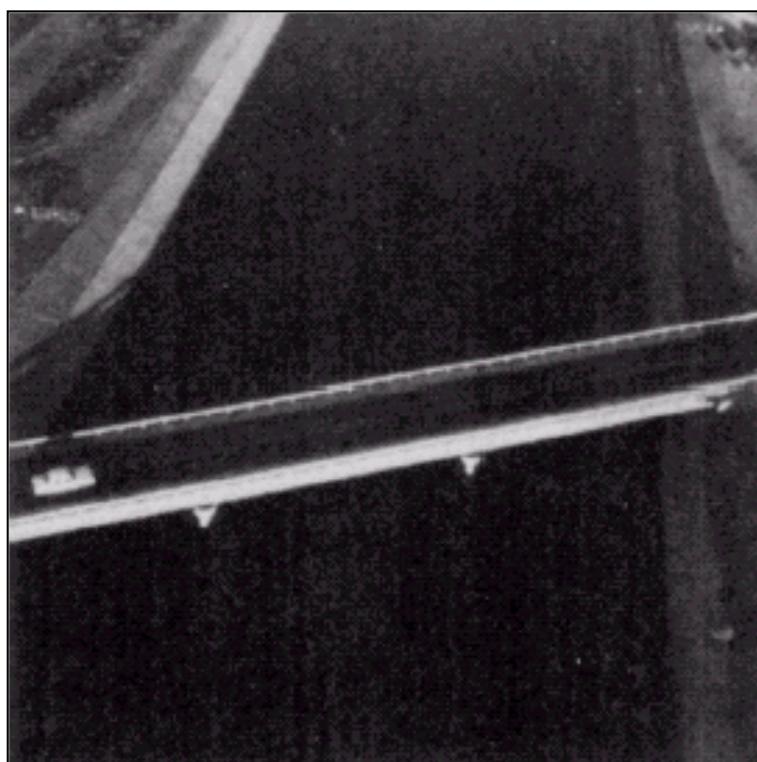
▶ Selección de contraste



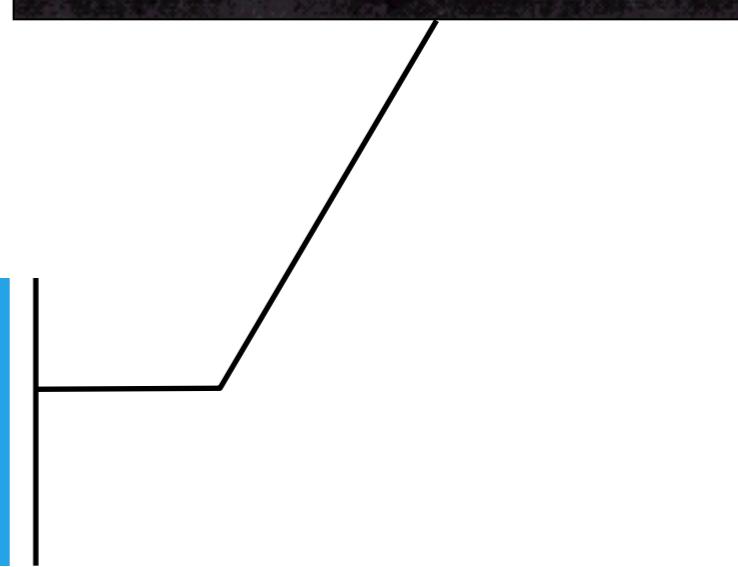
En este caso seleccionamos un rango [A-B] que destacamos. El resto lo reducimos.



▶ Selección de contraste



Esta transformación destaca el rango [A-B] y preserva el resto de los niveles



- ▶ Planos de bits: Consiste en dividir la imagen en planos de bits



cameraman.tif

uint8

2

255

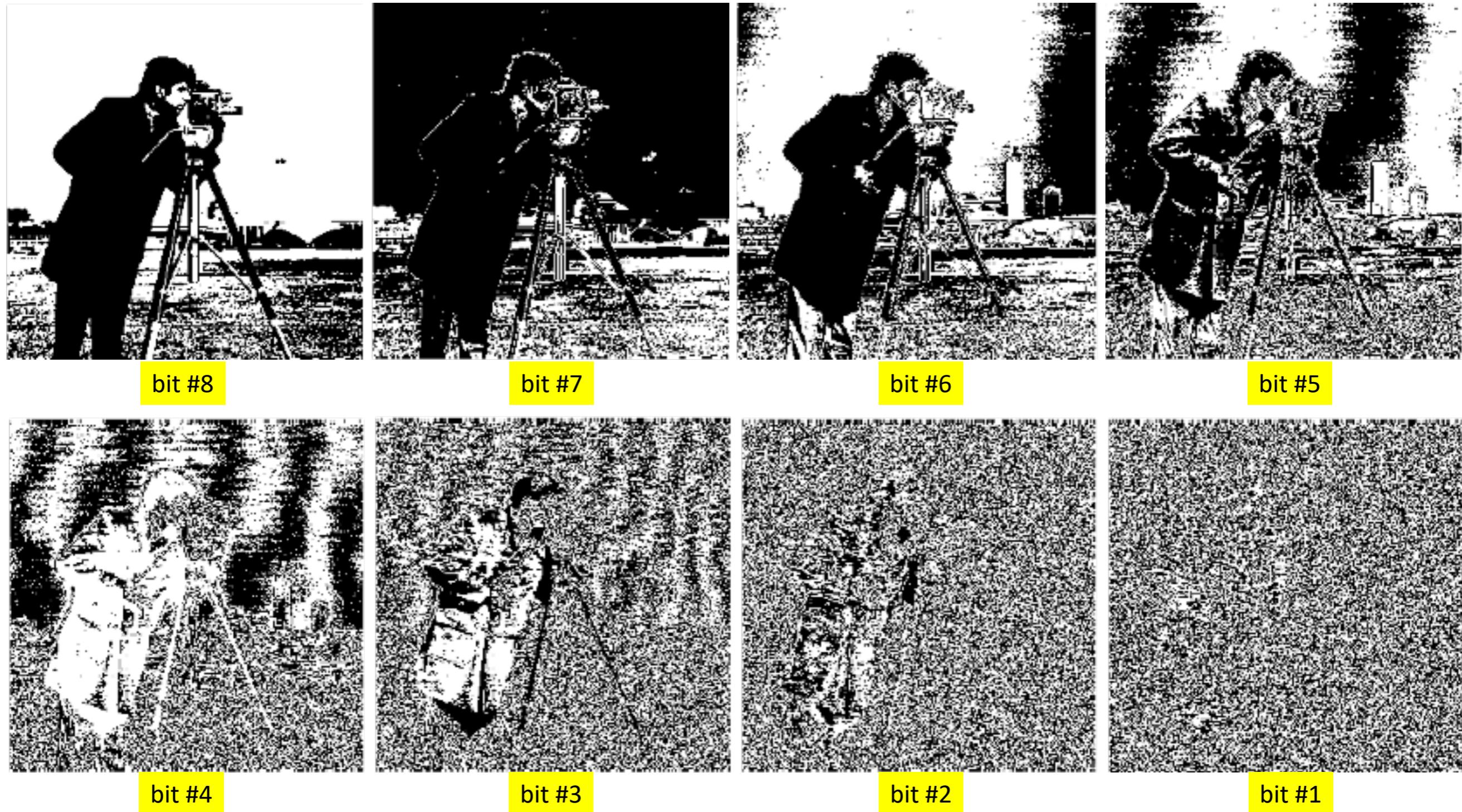
binario

0 0 0 0 0 0 1 0

1 1 1 1 1 1 1 1

bits #8
más significativo

▶ Planos de bits



▶ Planos de bits

```
import cv2
import numpy as np

img = cv2.imread('cameraman.png')
img = img[:, :, 0] #ocupamos en canal 0

for k in range(0, 7):
    plane = np.full((img.shape[0], img.shape[1]), 2 ** k, np.uint8)
```

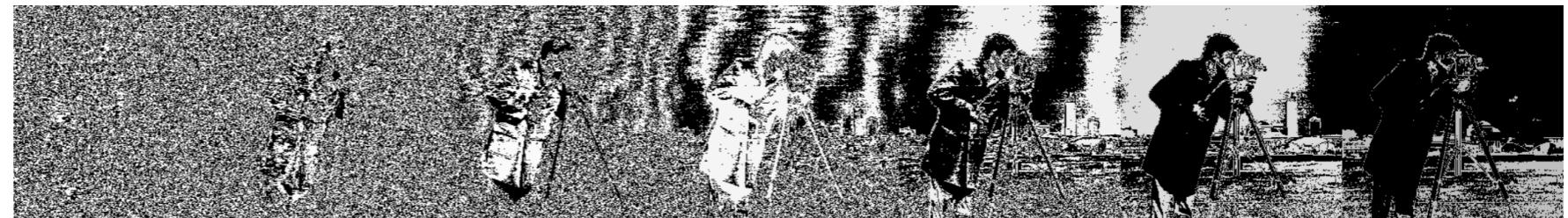
Esta función nos permite crear una matriz del mismo tamaño de la imagen, con un valor específico de nivel de gris

dimensión (alto, ancho)

valor deseado

tipo de datos

▶ Planos de bits



```
import cv2
import numpy as np

img = cv2.imread('cameraman.png')
img = img[:, :, 0] #ocupamos en canal 0

for k in range(0, 7):
    plane = np.full((img.shape[0], img.shape[1]), 2 ** k, np.uint8)

    res = plane & img
    x = res*255

    cv2.imshow("plane", x)
    cv2.waitKey()

cv2.destroyAllWindows()
```

Multiplicamos cada nivel de gris por la imagen, o realizamos una operación binaria AND

▶ Mejoramiento en el espacio

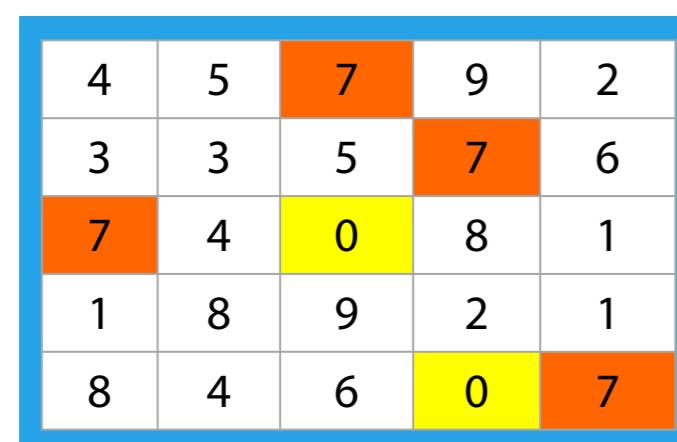
- Transformaciones básicas en niveles de grises
- Procesamiento de histogramas
- Operaciones aritméticas y lógicas
- Filtros espaciales (lineal y no lineal)

- Ecualización
- Especificación



▶ ¿Qué es el histograma en imágenes?

- El histograma de una imagen contiene el mismo número de píxel según un nivel de gris. Es decir, es una distribución acumulada de los tonos (valores de cada píxel) de una imagen.



5x5

Supongamos una
imagen con 10
niveles de gris (0-9)

¿Cuántos píxeles de nivel
 i hay en la imagen?

Píxel	Número
0	2
1	3
2	2
3	2
4	3
5	2
6	2
7	4
8	3
9	2
Σ	25

▶ ¿Qué es el histograma en imágenes?

- En OpenCV existe una función para determinar el histograma de una imagen y conocer su distribución.

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('cameraman.png')
hst = cv2.calcHist([img], [2], None, [256], [0,256])

plt.plot(hst)
plt.show()
```



Para ocupar la biblioteca
matplotlib, debes instalarla

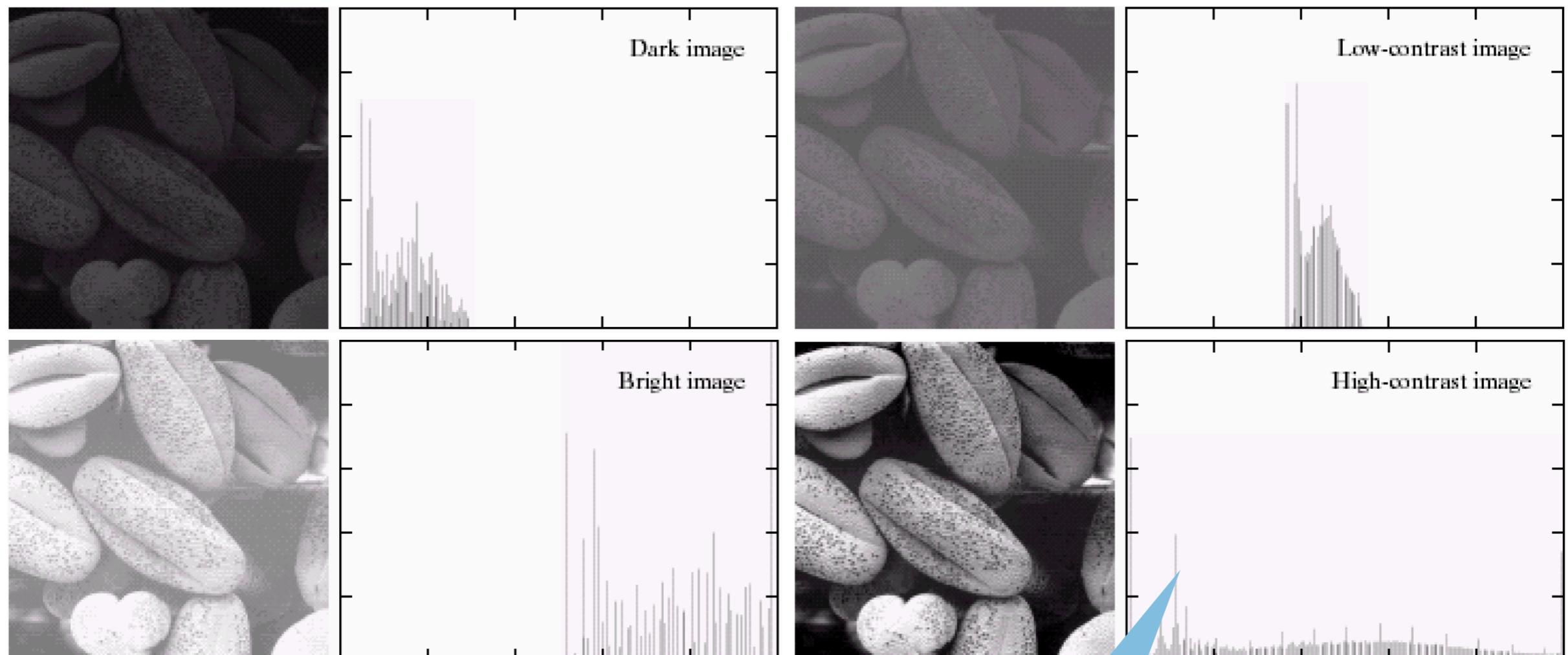
```
pip install matplotlib
```

¿Qué píxeles existen
en mayor cantidad?



Solo si ocupas anaconda `conda install matplotlib`

▶ Interpretación del histograma



A partir de la información de un histograma podemos ver el brillo y contraste de una imagen

▶ Ejemplo de ecualización



Imagen original

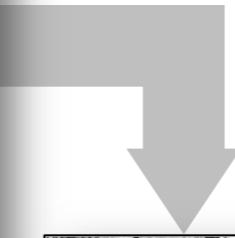
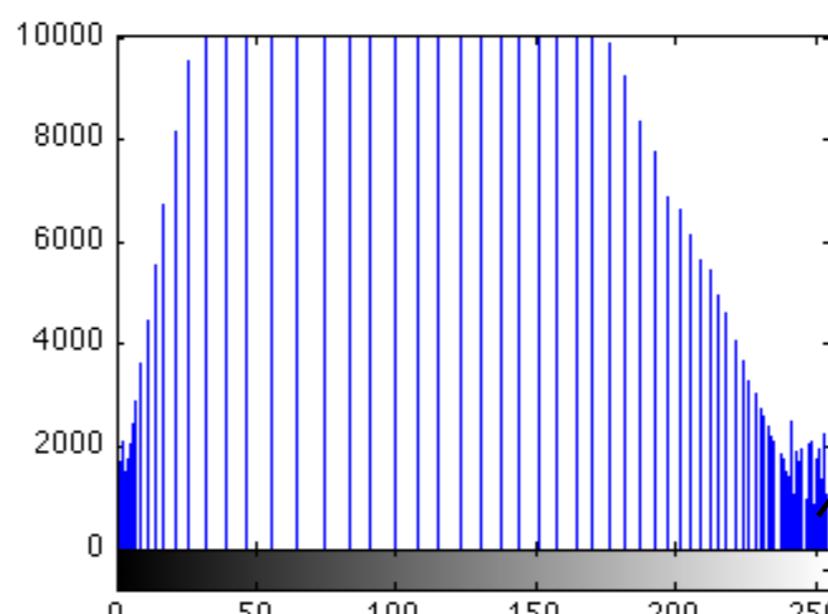
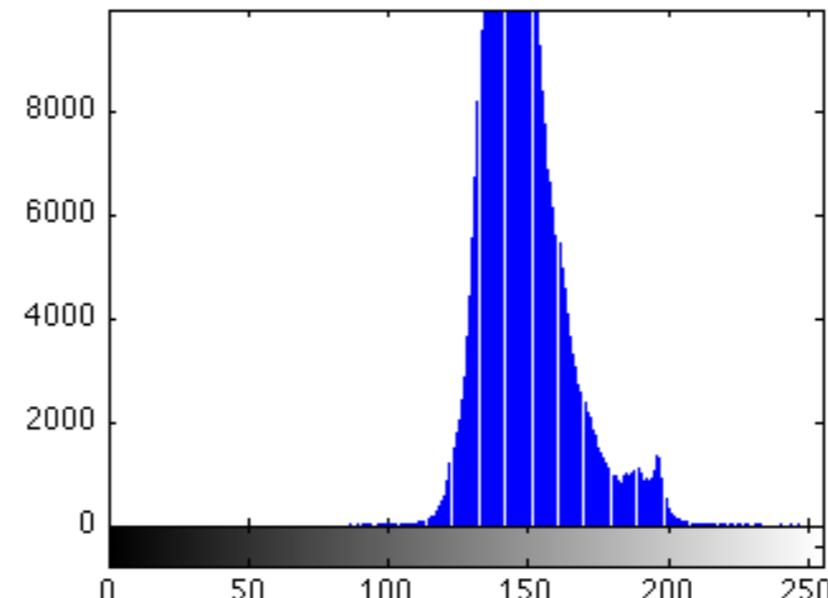


Imagen ecualizada



La ecualización nos permite modificar el histograma original y distribuirlo en uno nuevo.

► Ejemplo de ecualización

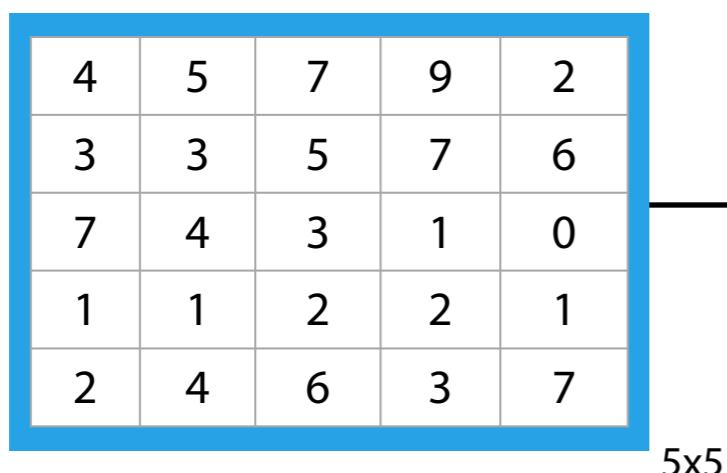


El histograma ecualizado distribuye los niveles de grises.

- ▶ Pasos para la ecuación de una imagen (a color en cada canal)

1) Crear el histograma

Sea $p(i)$ la distribución de cada píxel i en la imagen



Píxel	p
0	1
1	4
2	4
3	4
4	3
5	2
6	2
7	4
8	0
9	1
Σ	25

▶ Pasos para la ecualización

2) Determinar la CDF del histograma

La CDF es la *cumulative distribution function* o suma acumulada de la distribución

Píxel	p	cdf
0	1	1
1	4	5
2	4	9
3	4	13
4	3	16
5	2	18
6	2	20
7	4	24
8	0	24
9	1	25
Σ	25	

$$cdf(i) = \sum_{j=0}^i p(j)$$

Recuerde que p tiene la suma de cada nivel.

- ▶ Pasos para la ecualización

3) Calcular la CDF mínima

Píxel	p	cdf
0	1	1
1	4	5
2	4	9
3	4	13
4	3	16
5	2	18
6	2	20
7	4	24
8	0	24
9	1	25
Σ	25	

$$cdf_{\min} = \min(cdf(i))$$

- ▶ Pasos para la ecualización

4) Determinar el número de cuadros de la imagen

Píxel	P	cdf
0	1	1
1	4	5
2	4	9
3	4	13
4	3	16
5	2	18
6	2	20
7	4	24
8	0	24
9	1	25
Σ	25	

4	5	7	9	2
3	3	5	7	6
7	4	3	1	0
1	1	2	2	1
2	4	6	3	7

5x5 —→ 25

- ▶ Pasos para la ecualización

5) Calcular la nueva transformación $h(v)$

Píxel	P	cdf
0	1	1
1	4	5
2	4	9
3	4	13
4	3	16
5	2	18
6	2	20
7	4	24
8	0	24
9	1	25
Σ	25	

$$h(v) = \text{round} \left(\frac{cdf(v) - cdf_{\min}}{(m \times n) - cdf_{\min}} \times (L - 1) \right)$$

En una imagen real,
existen 256 niveles
(0-255). → **L=256**



- Pasos para la ecualización

6) Calcular la nueva transformación $h(v)$

Píxel	P	cdf	h
0	1	1	0
1	4	5	2
2	4	9	3
3	4	13	5
4	3	16	6
5	2	18	6
6	2	20	7
7	4	24	9
8	0	24	9
9	1	25	9
Σ	25		

Ejemplo: Calculemos
 $h(6)$

$$h(v) = \text{round} \left(\frac{cdf(v) - cdf_{\min}}{(m \times n) - cdf_{\min}} \times (L - 1) \right)$$

$$h(6) = \text{round} \left(\frac{20 - 1}{25 - 1} \times 9 \right) = \text{round} \left(\frac{19 \times 9}{24} \right) = 7$$

- ▶ Pasos para la ecualización

7) Calcular la nueva imagen a partir de la distribución $h(v)$

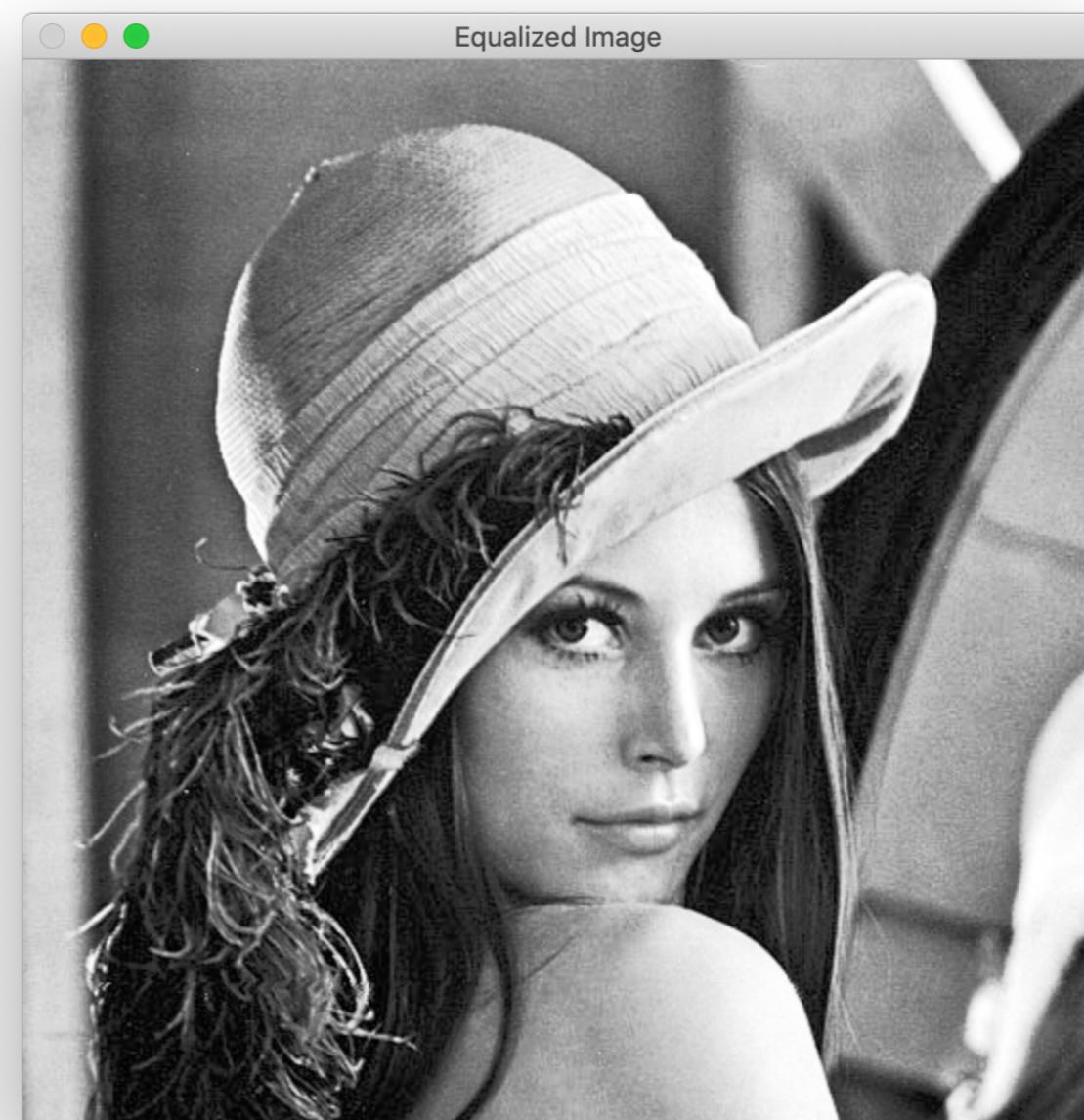
Píxel	P	cdf	h
0	1	1	0
1	4	5	2
2	4	9	3
3	4	13	5
4	3	16	6
5	2	18	6
6	2	20	7
7	4	24	9
8	0	24	9
9	1	25	9
Σ	25		

4	5	7	9	2
3	3	5	7	6
7	4	3	1	0
1	1	2	2	1
2	4	6	3	7



6	6	9	9	3
5	5	6	9	7
9	6	5	2	0
2	2	3	3	2
3	6	7	5	9

▶ Ejemplo de ecualización



El histograma
ecualizado distribuye
los niveles de grises.

- ▶ Códigos del ejemplo anterior

```
import cv2
img = cv2.imread('lena.png')

src = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

dst = cv2.equalizeHist(src)

cv2.imshow('Source image', src)
cv2.imshow('Equalized Image', dst)
cv2.waitKey()
```

función de
ecualización

▶ Mejoramiento en el espacio

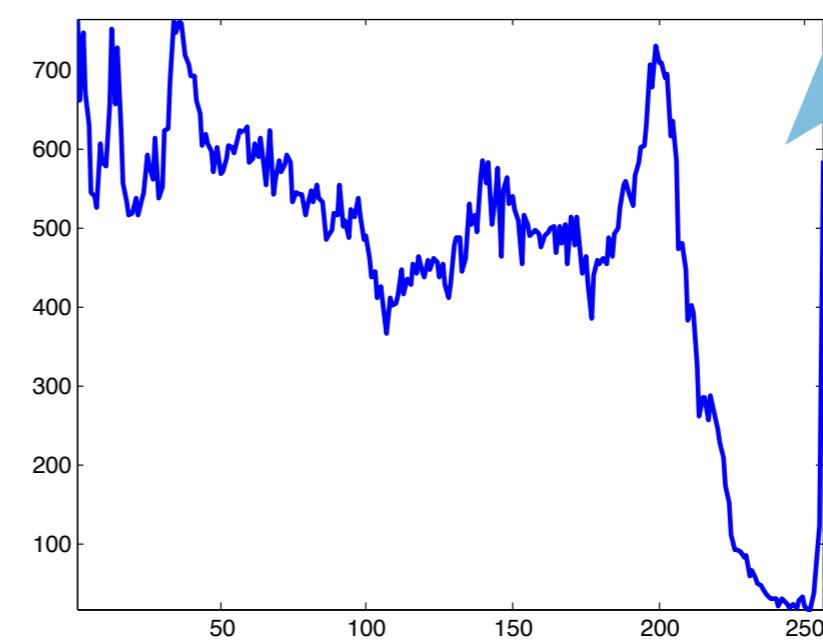
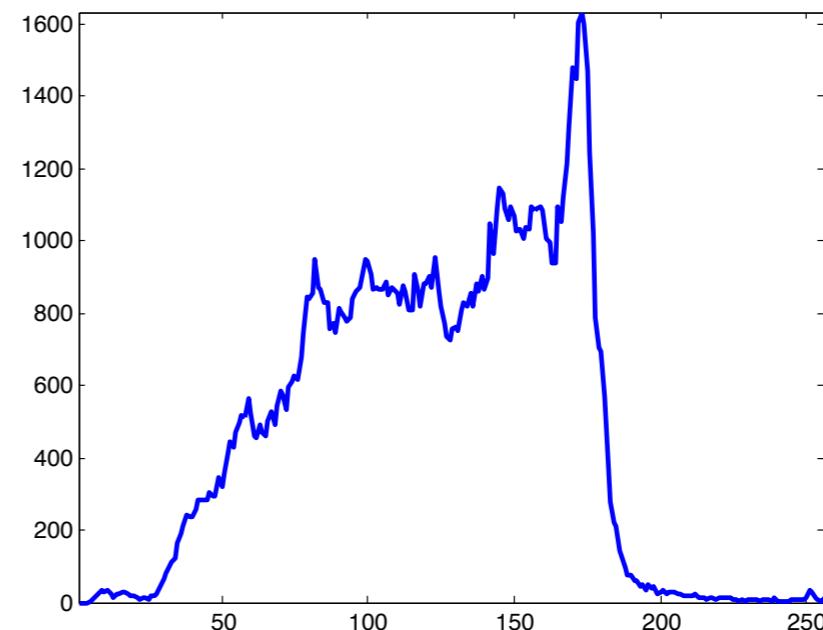
- Transformaciones básicas en niveles de grises
- Procesamiento de histogramas
- Operaciones aritméticas y lógicas
- Filtros espaciales (lineal y no lineal)

- Ecualización
- Especificación



▶ Especificación de histograma

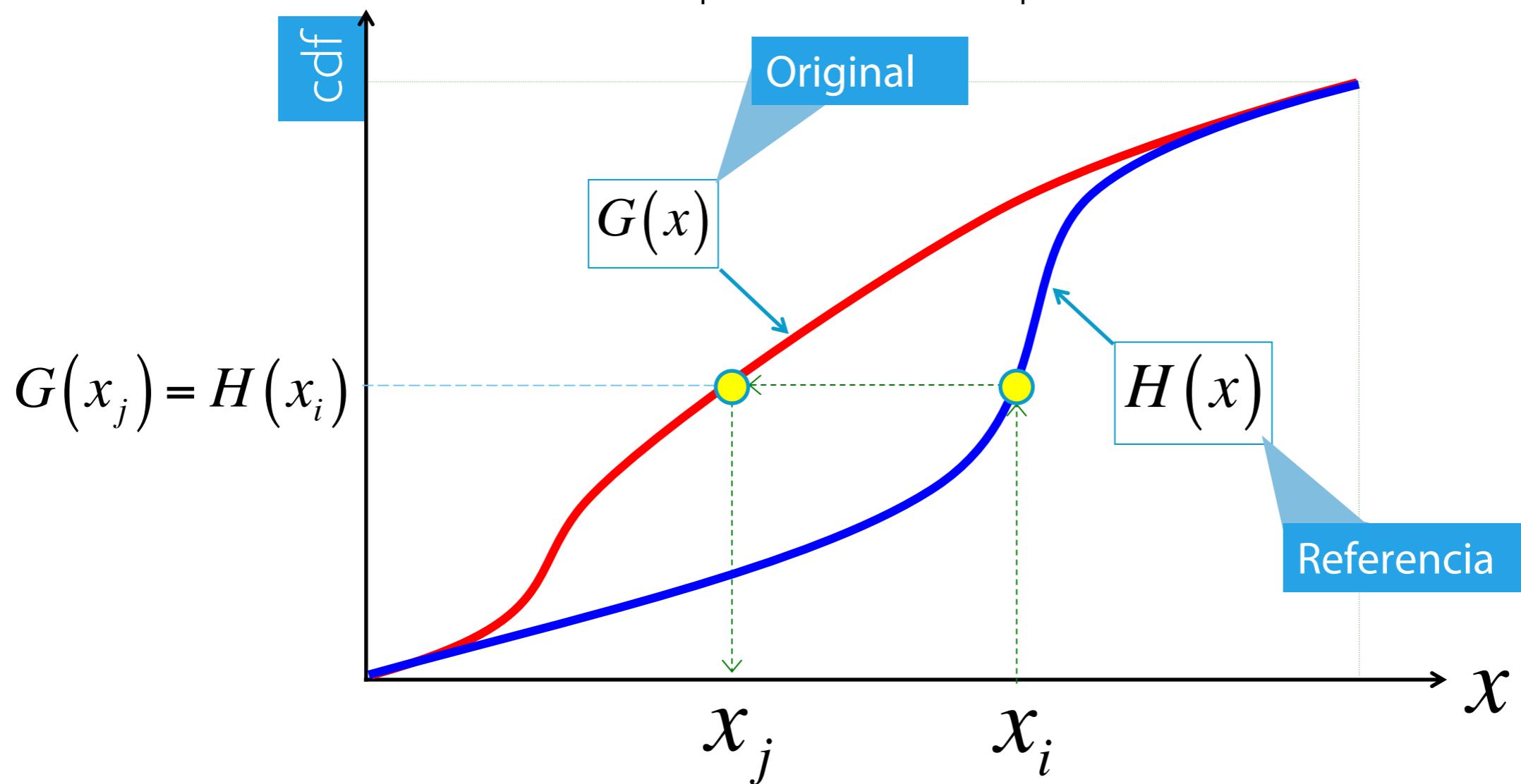
- Consiste en aplicar el histograma de una imagen a otra a través de una función o curva específica.



Aplicar la forma
del histograma de
la superior a la
imagen inferior

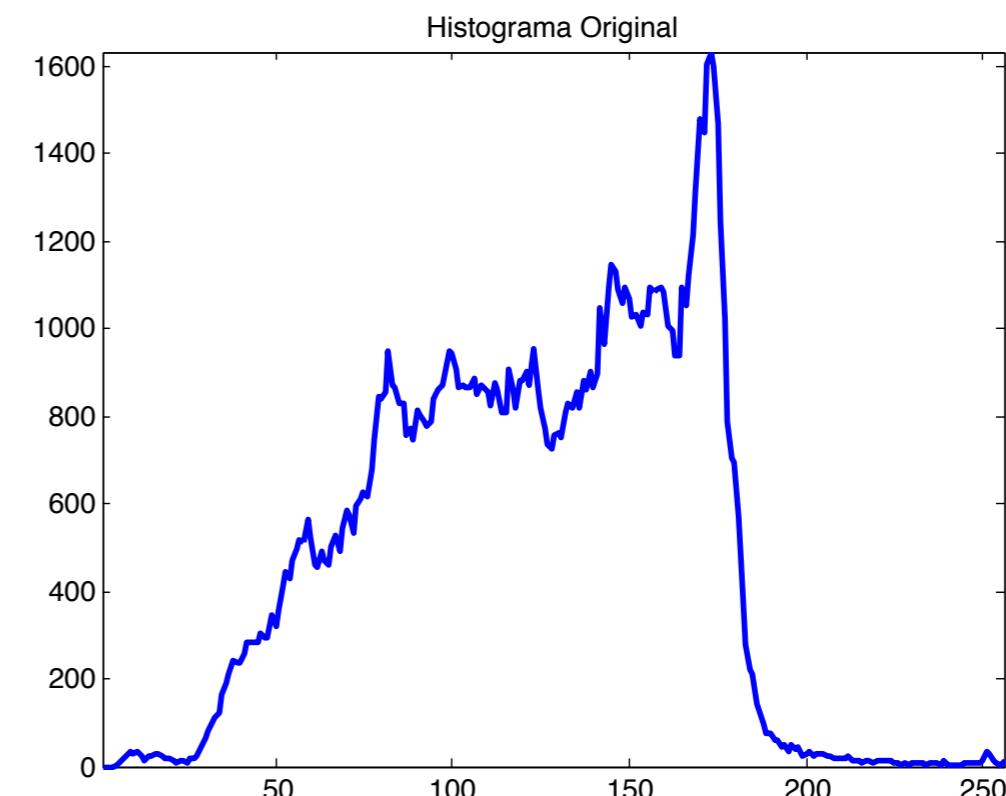
▶ Especificación de histograma

- Consiste en aplicar el histograma de una imagen a otra a través de una función o curva específica.
- En la práctica buscamos que: $|G(x_j) - H(x_i)| \rightarrow 0$



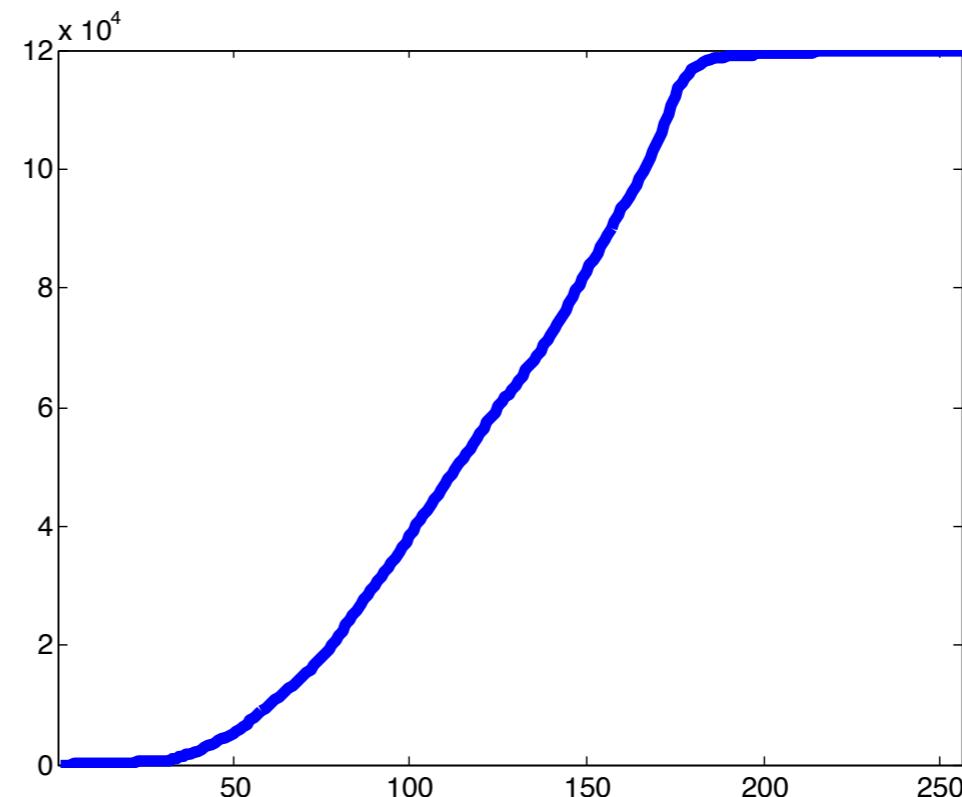
- ▶ Pasos para la especificación de una imagen (a color en cada canal)

- 1) Determine el histograma de la imagen de **referencia**



- ▶ Pasos para la especificación de una imagen (a color en cada canal)

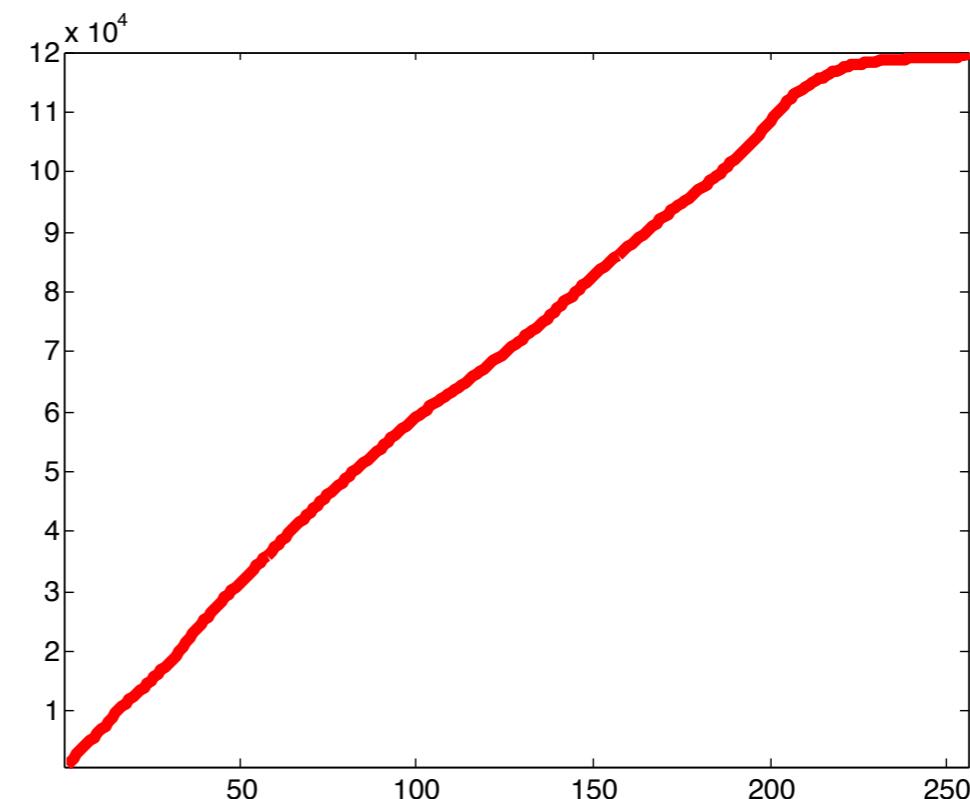
2) Calcule la CDF del histograma de referencia



Vamos a emplear el histograma de esta imagen.

- ▶ Pasos para la especificación de una imagen (a color en cada canal)

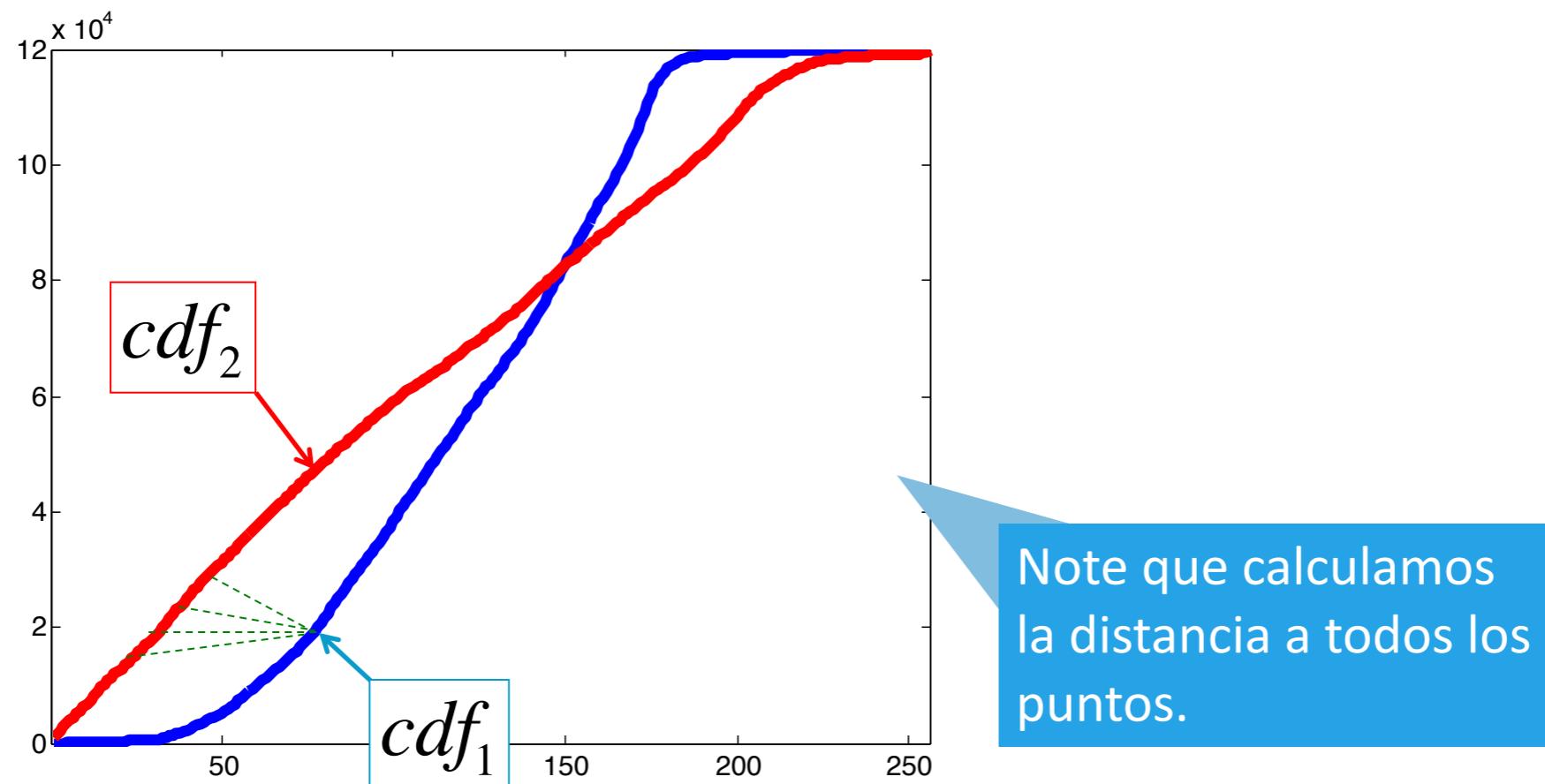
3) Calcule la CDF del histograma a especificar



Vamos a modificar el
histograma de esta
imagen

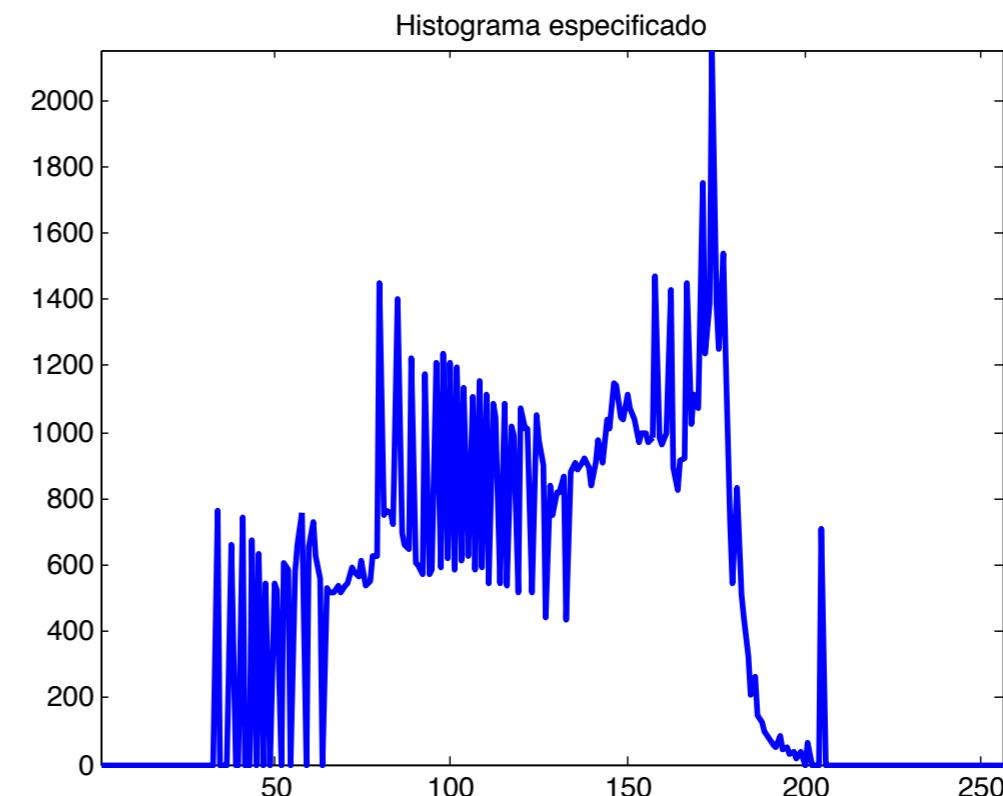
- ▶ Pasos para la especificación de una imagen (a color en cada canal)

4) Busque el valor más cercano a cero entre la CDF1 y la CDF2



- ▶ Pasos para la especificación de una imagen (a color en cada canal)

5) Desplegamos el resultado y calculamos el nuevo histograma



- ▶ Pasos para la especificación de una imagen (a color en cada canal)

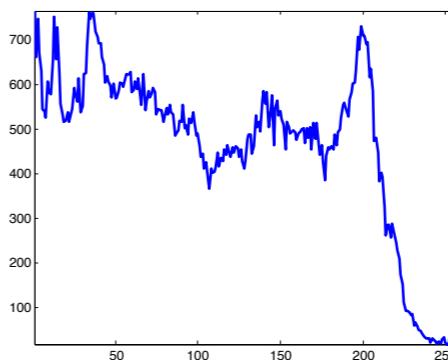
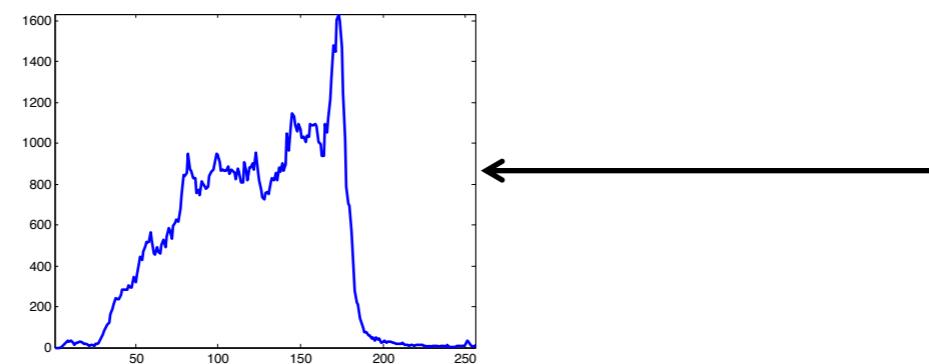
Busque las diferencias



Imagen de referencia



Imagen a especificar



Queremos
cambiar este
histograma

- ▶ Pasos para la especificación de una imagen (a color en cada canal)

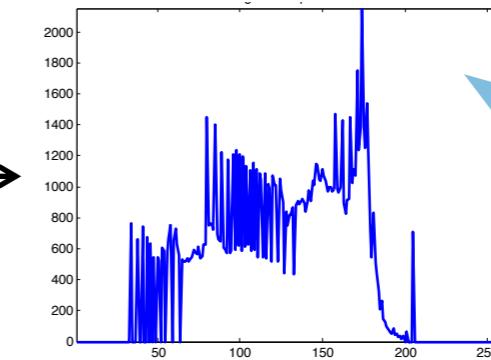
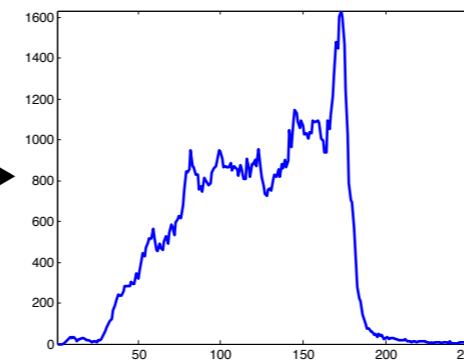
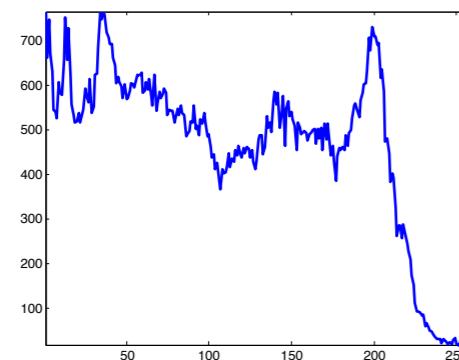
Busque las diferencias



Imagen de referencia



Resultado de la especificación



Histograma
resultante
especificado

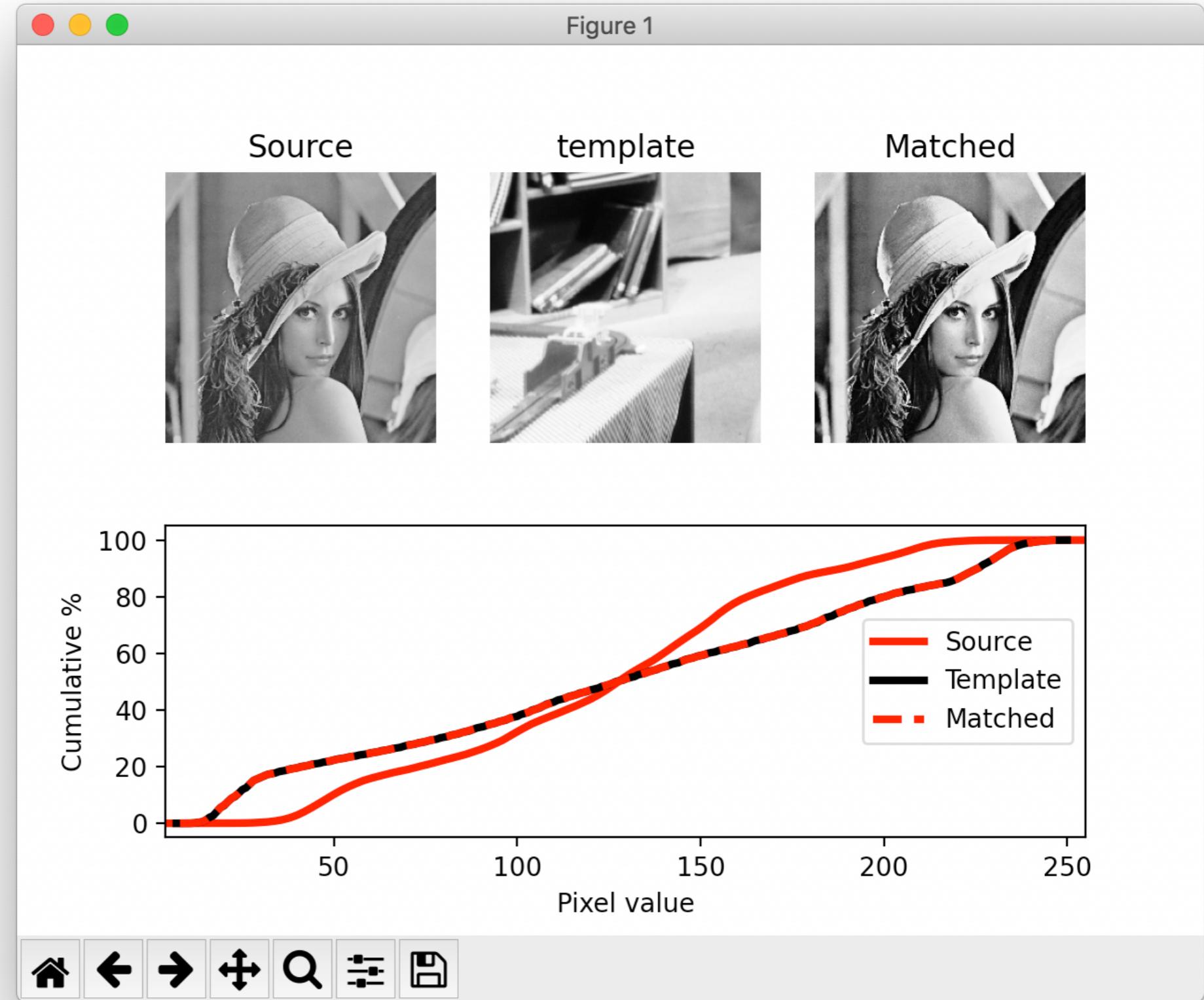


Bajar código en
webcursos



Ejercicio

Descargue dos imágenes desde internet y genere una especificación con dos nuevas imágenes



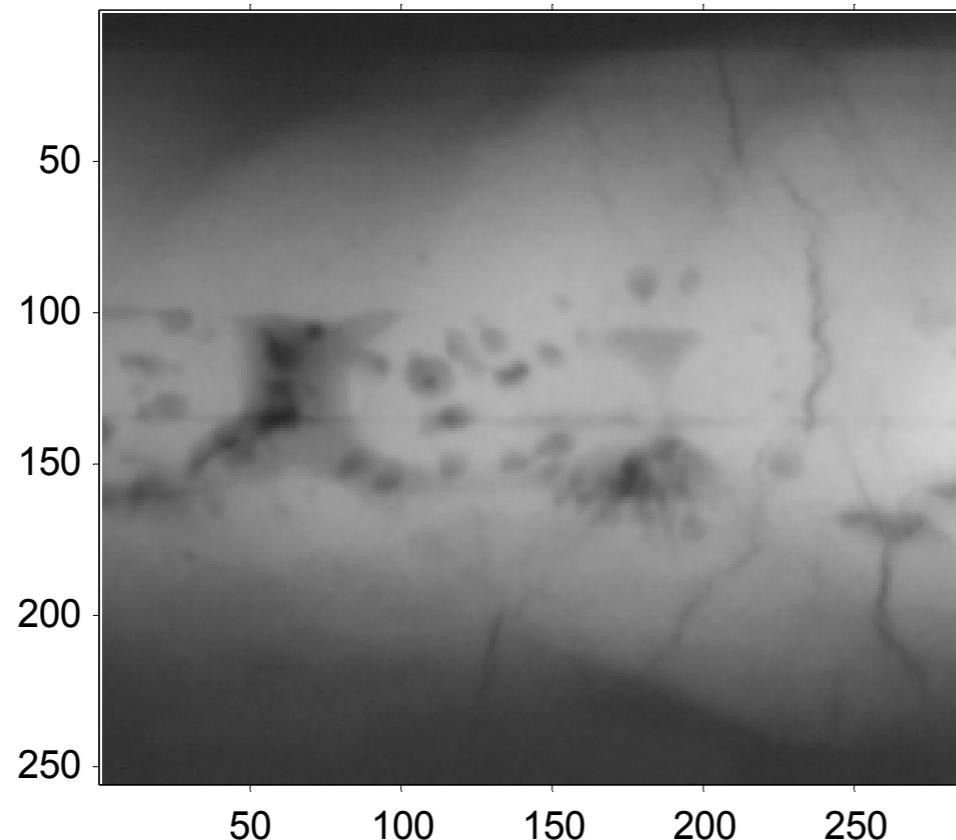
▶ Mejoramiento en el espacio

- Transformaciones básicas en niveles de grises
- Procesamiento de histogramas
- Operaciones aritméticas y lógicas

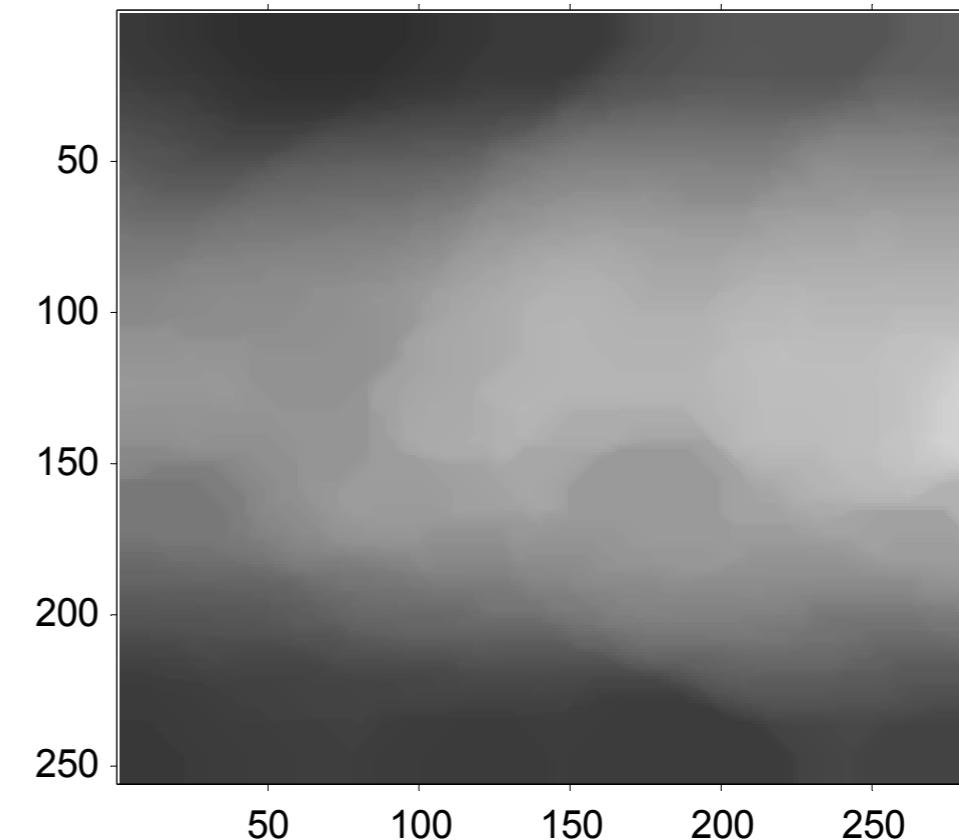


▶ Resta de imágenes

- La resta de imágenes consiste en restar una imagen con otra píxel por píxel.



I = grisA.tif



J = grisB.tif

$$P(x, y) = I(x, y) - J(x, y)$$

▶ Resta de imágenes

- La resta de imágenes consiste en restar una imagen con otra píxel por píxel.

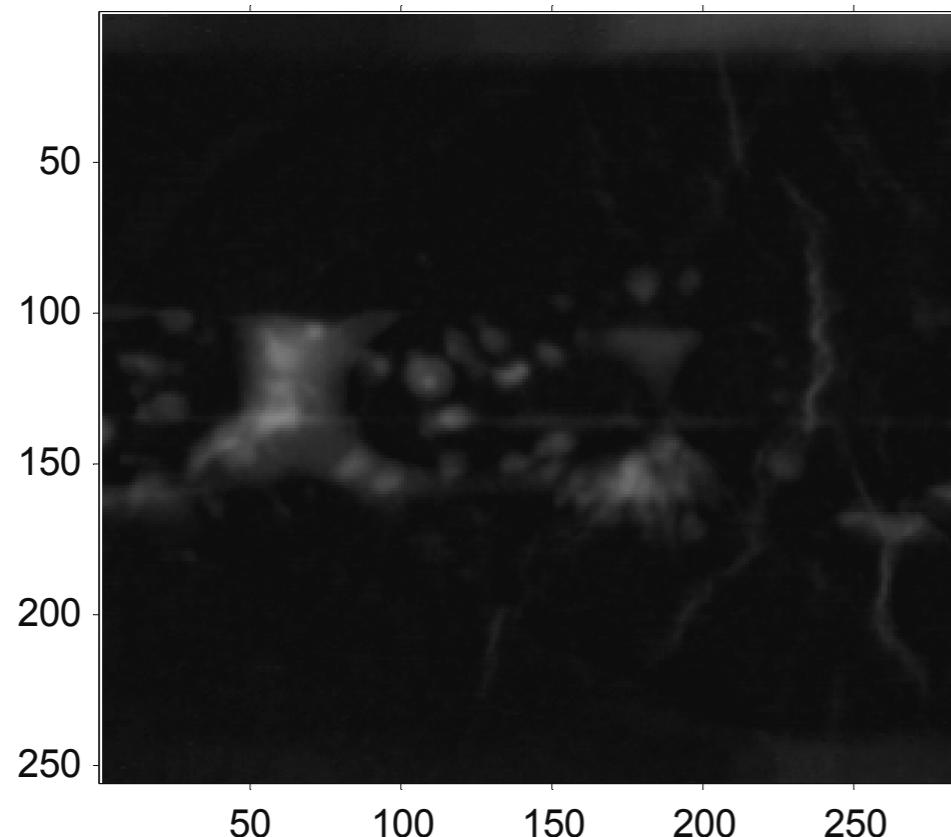
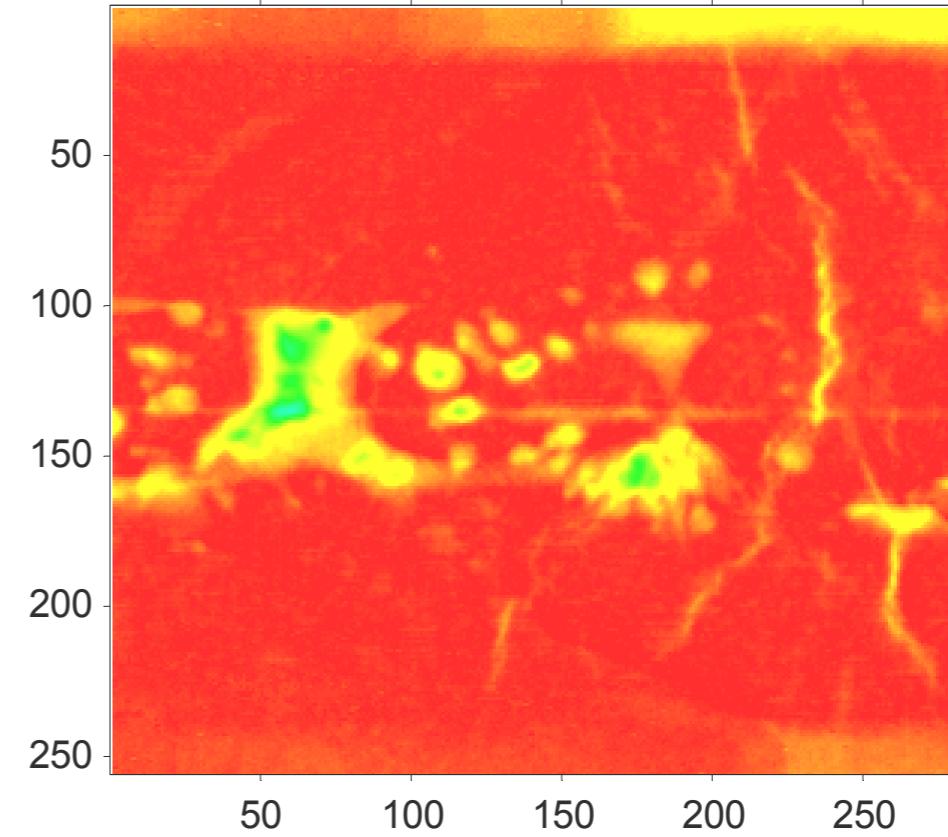


Imagen diferencia



Diferencia en Pseudocolor

$$\rightarrow P(x, y) = I(x, y) - J(x, y)$$

- ▶ Promedio de las imágenes

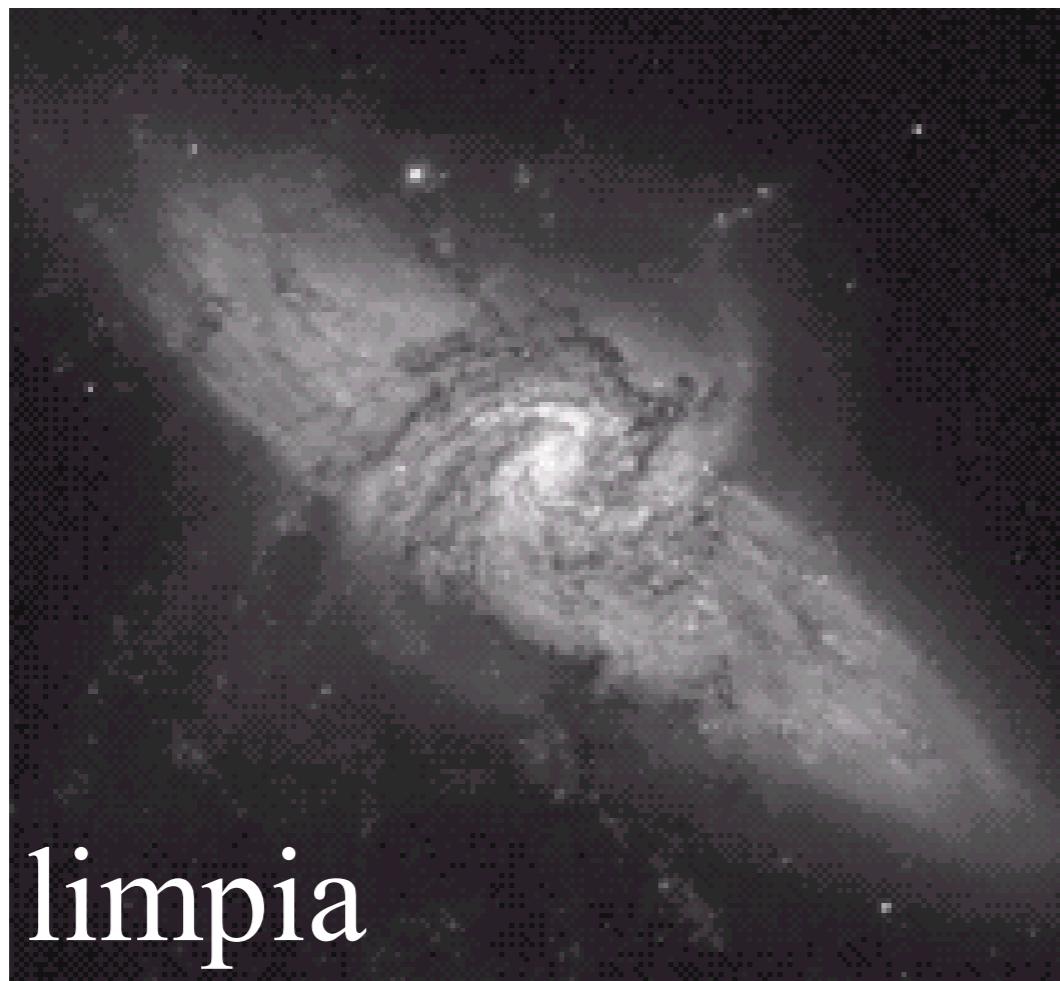
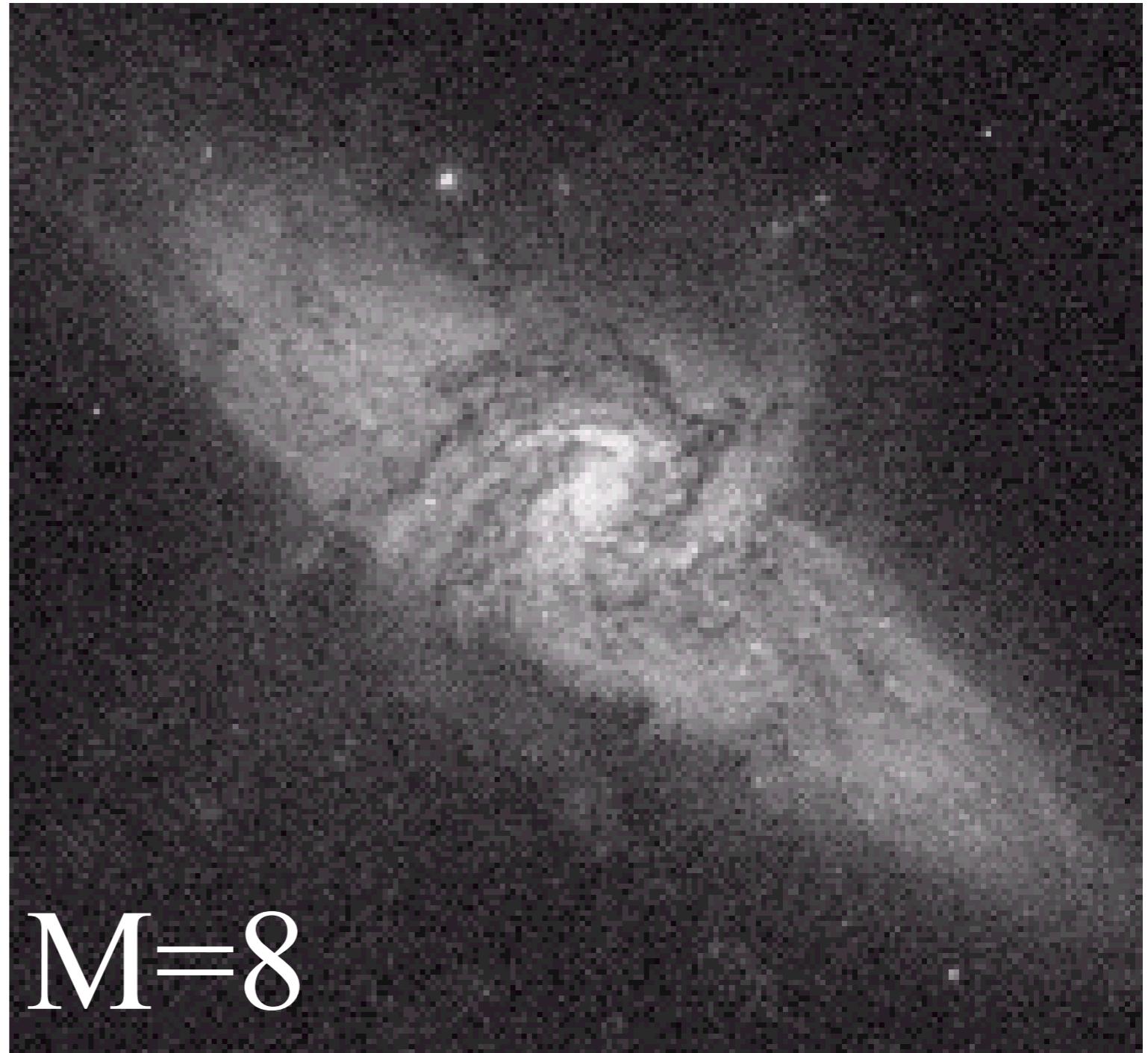
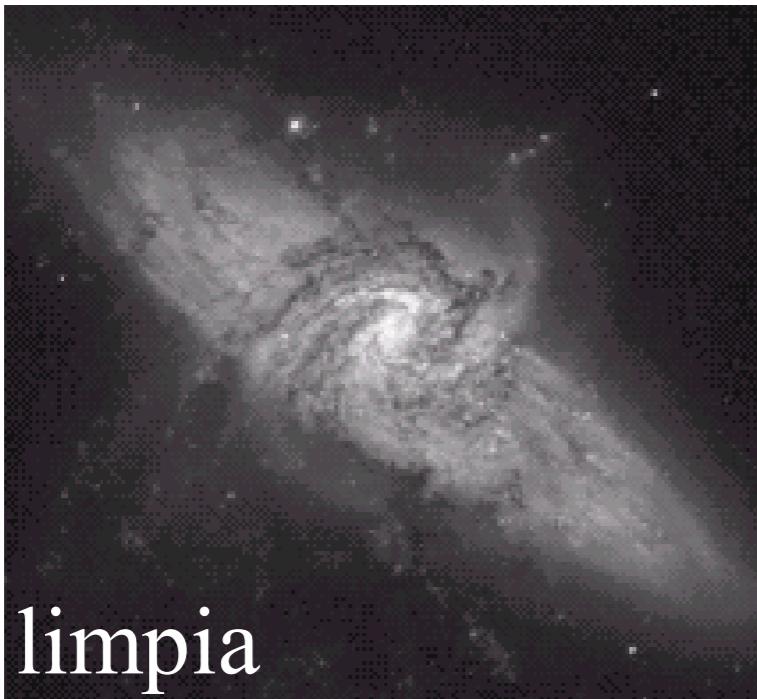


Imagen objetivo

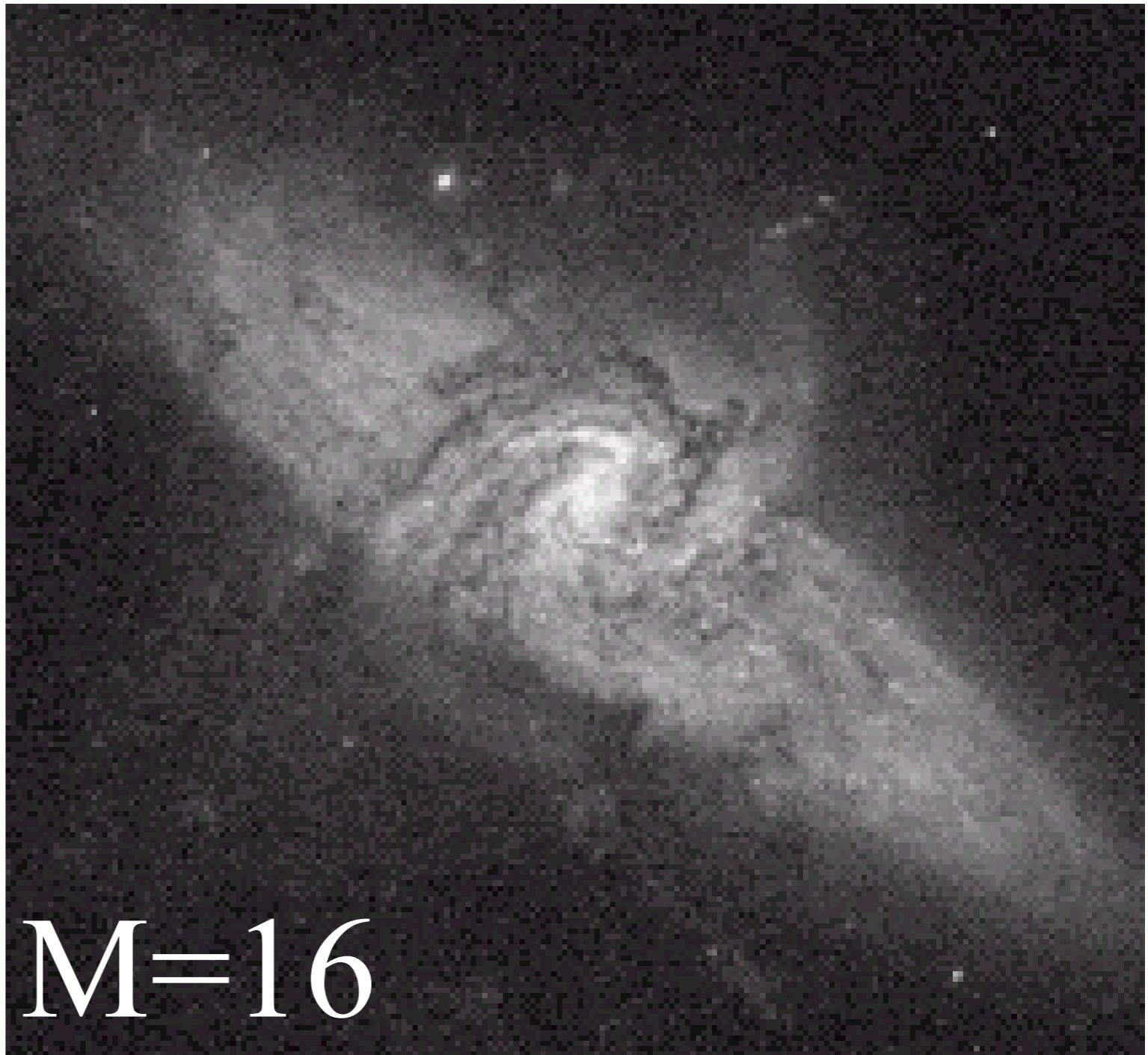


Suponga que tiene una imagen con ruido. Usted desea una imagen libre de ruido

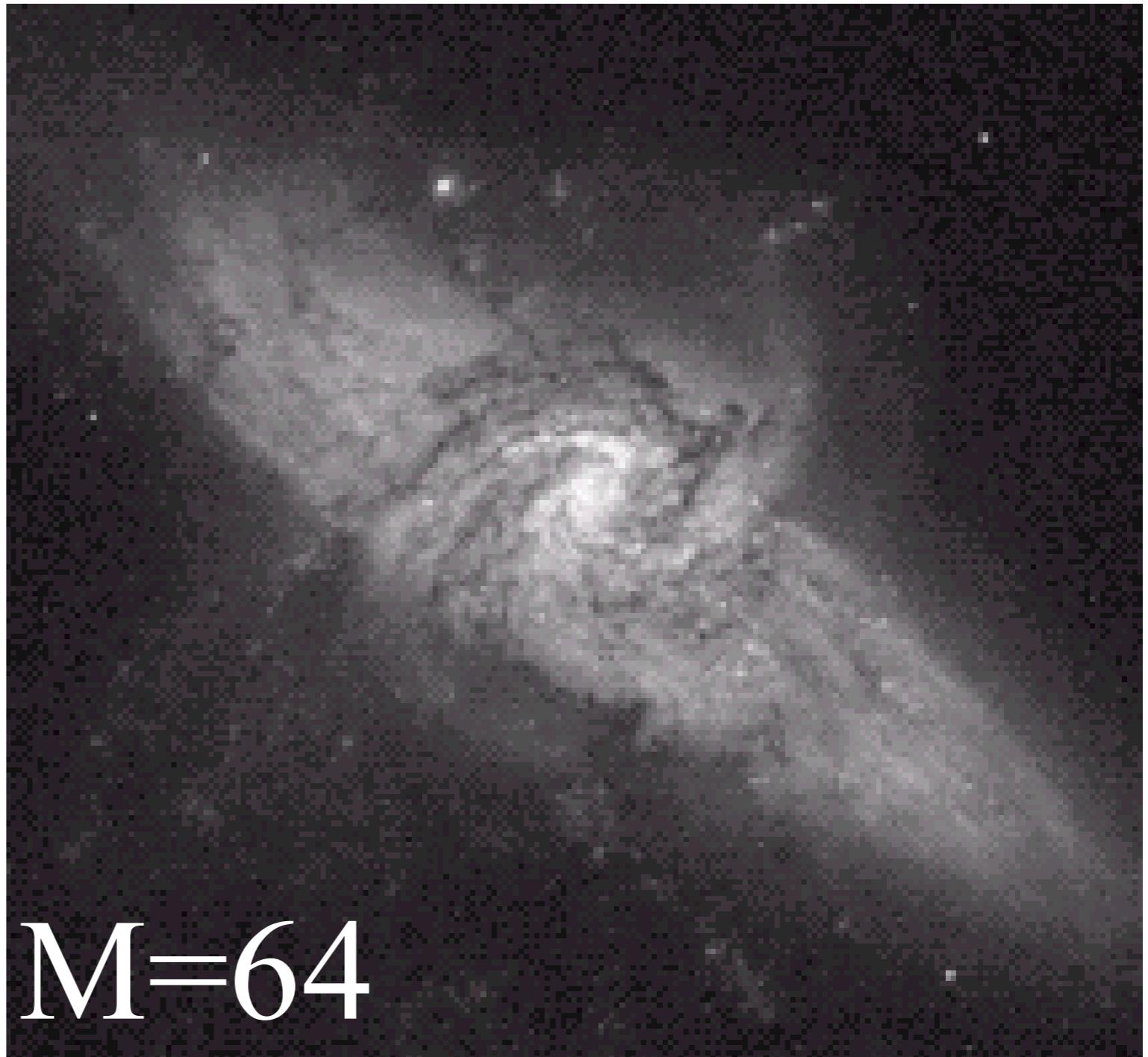
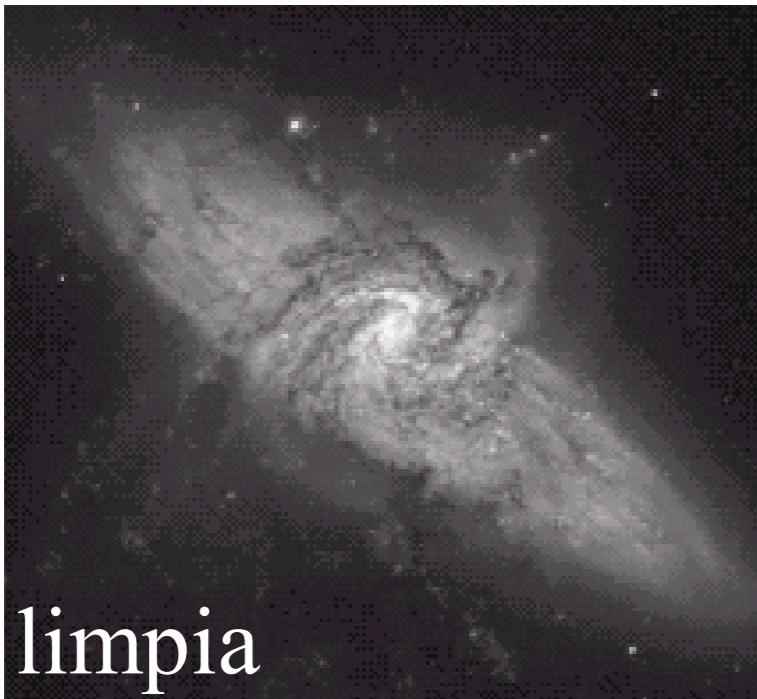
- ▶ Promedio de las imágenes



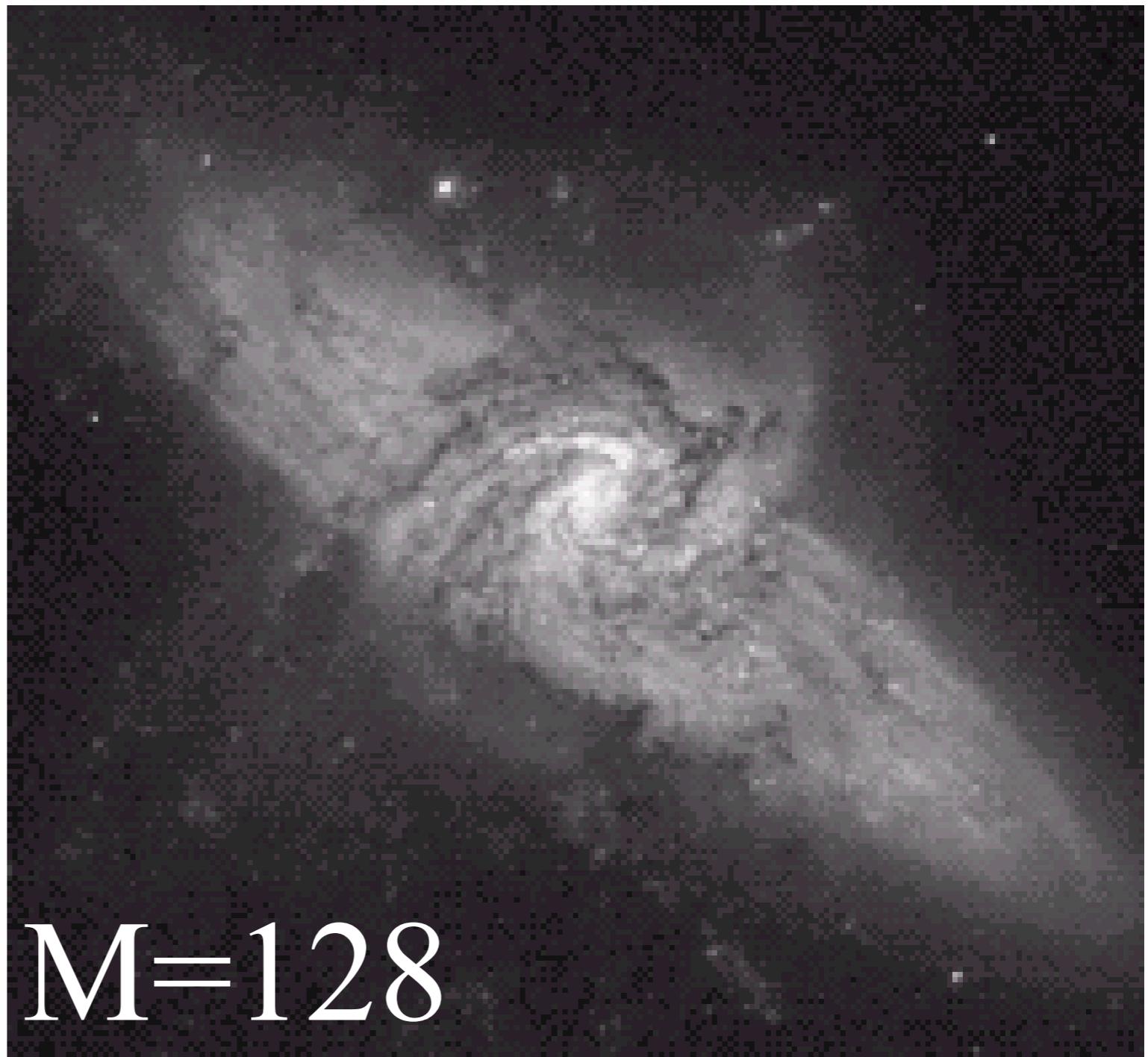
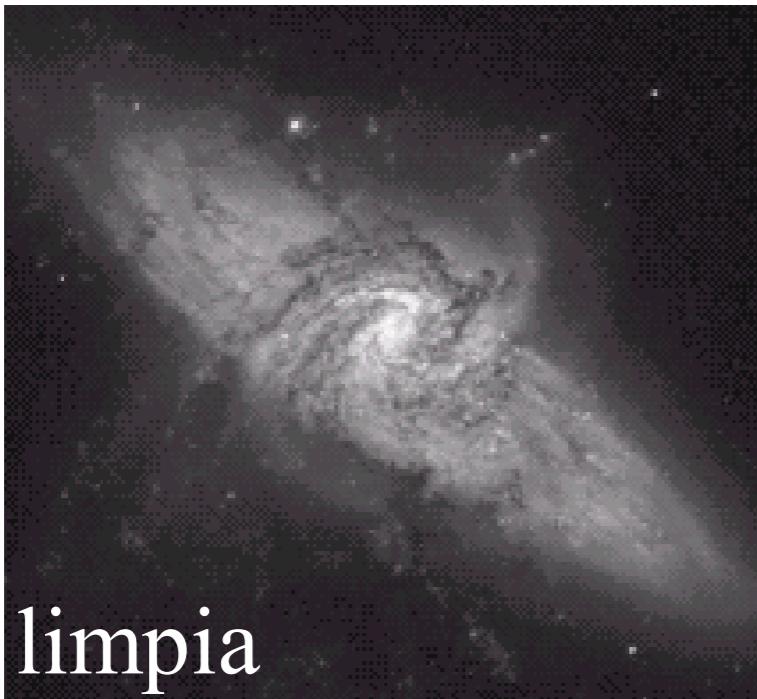
- ▶ Promedio de las imágenes



- ▶ Promedio de las imágenes



- ▶ Promedio de las imágenes



- Promedio de las imágenes

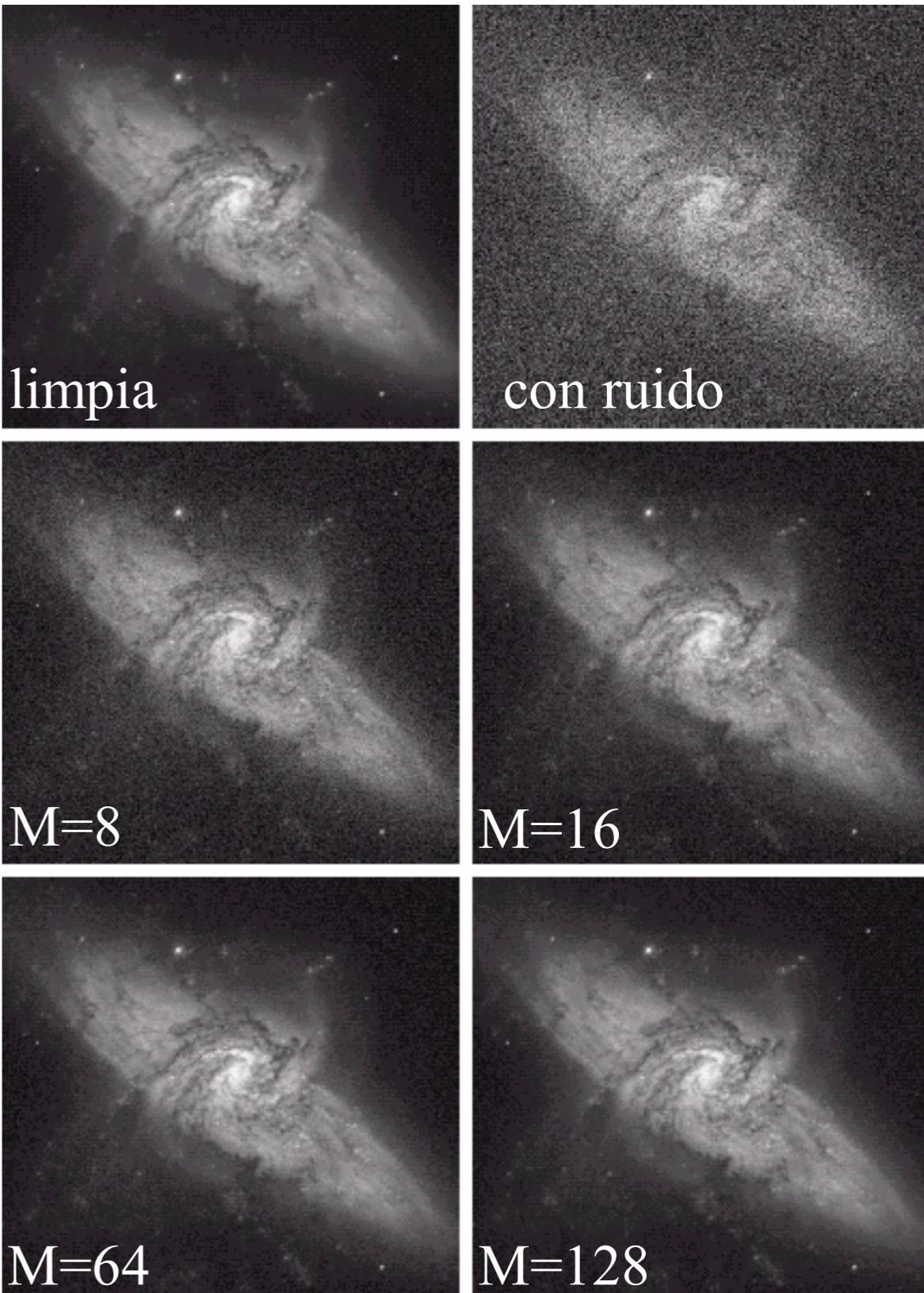
$$g_i(x, y) = f(x, y) + \eta(x, y)$$

Ruido no correlacionado con media cero.

Teorema del límite central: El promedio de variables aleatorias tiene norma 1 y media cero

$$\bar{g}(x, y) = \frac{1}{M} \cdot \sum_{i=1}^M g_i(x, y)$$

Imagen promedio.
Mientras más, disminuye el ruido. ¿Por qué?



▶ Suma de dos imágenes

```
import cv2

lena= cv2.imread('lena.png')
gray1 = cv2.cvtColor(lena, cv2.COLOR_BGR2GRAY)

barbara= cv2.imread('barbara_gray.bmp')
gray2 = cv2.cvtColor(barbara, cv2.COLOR_BGR2GRAY)

output = cv2.add(gray1,gray2)

cv2.imshow('output', output)
cv2.waitKey(0)
```



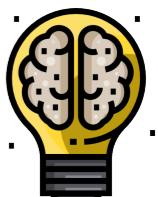
+



=



suma



Pregunta
¿Por qué se ve más clara la imagen resultante?

▶ Suma de dos imágenes

```
import cv2

lena= cv2.imread('lena.png')
gray1 = cv2.cvtColor(lena, cv2.COLOR_BGR2GRAY)

barbara= cv2.imread('barbara_gray.bmp')
gray2 = cv2.cvtColor(barbara, cv2.COLOR_BGR2GRAY)

output = cv2.addWeighted(gray1, 0.3, gray2, 0.7, 0)

cv2.imshow('output', output)
cv2.waitKey(0)
```

suma
Ponderada



+



=

