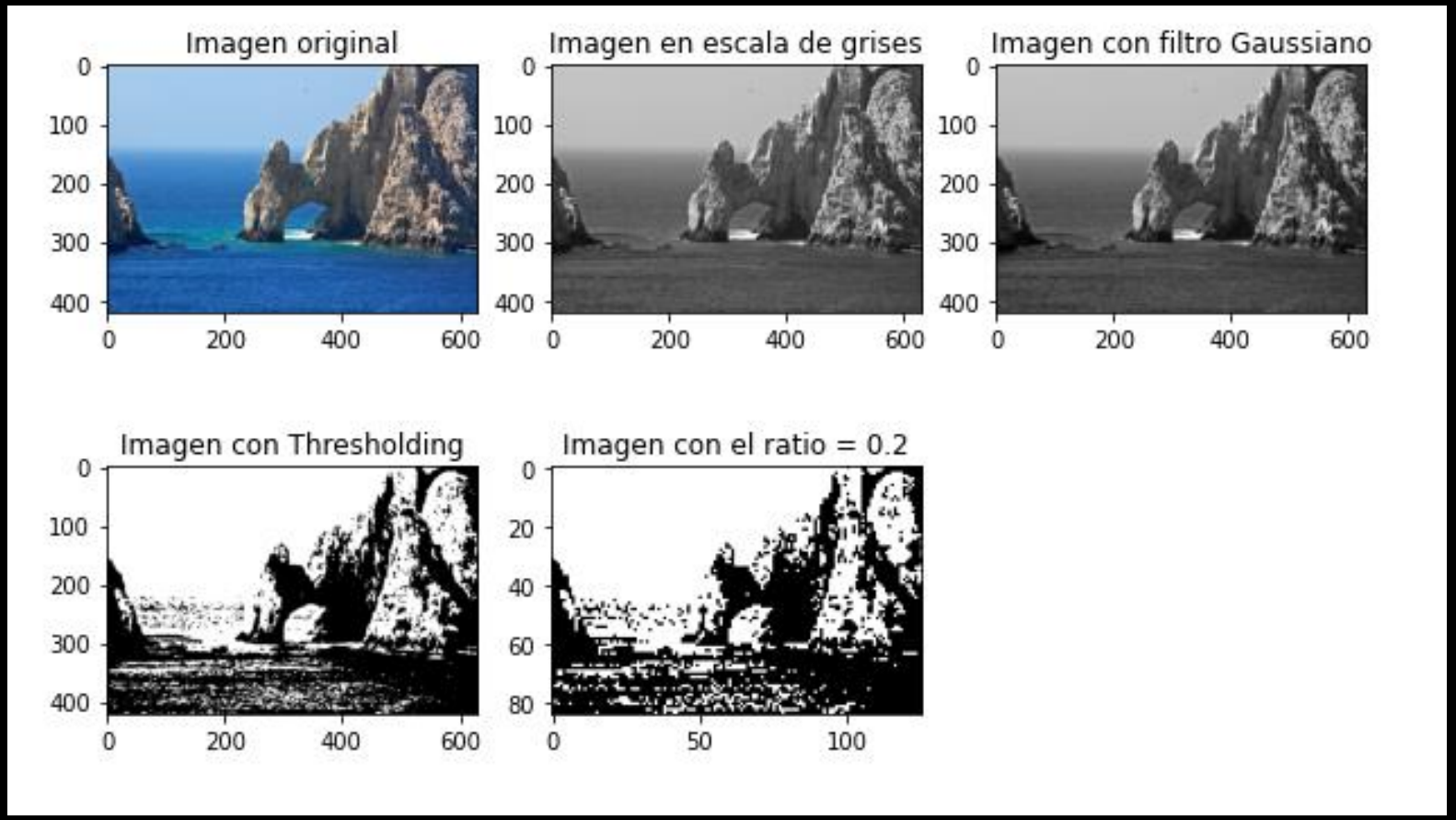
PROCESAMIENTO DE IMAGENES

Profesor: Enrique Camacho
enrique.camacho@gmail.com

PI - 02 - Manipulación de imágenes



Parte 1:

Conocer los principales pasos para manejo de imágenes con OpenCV

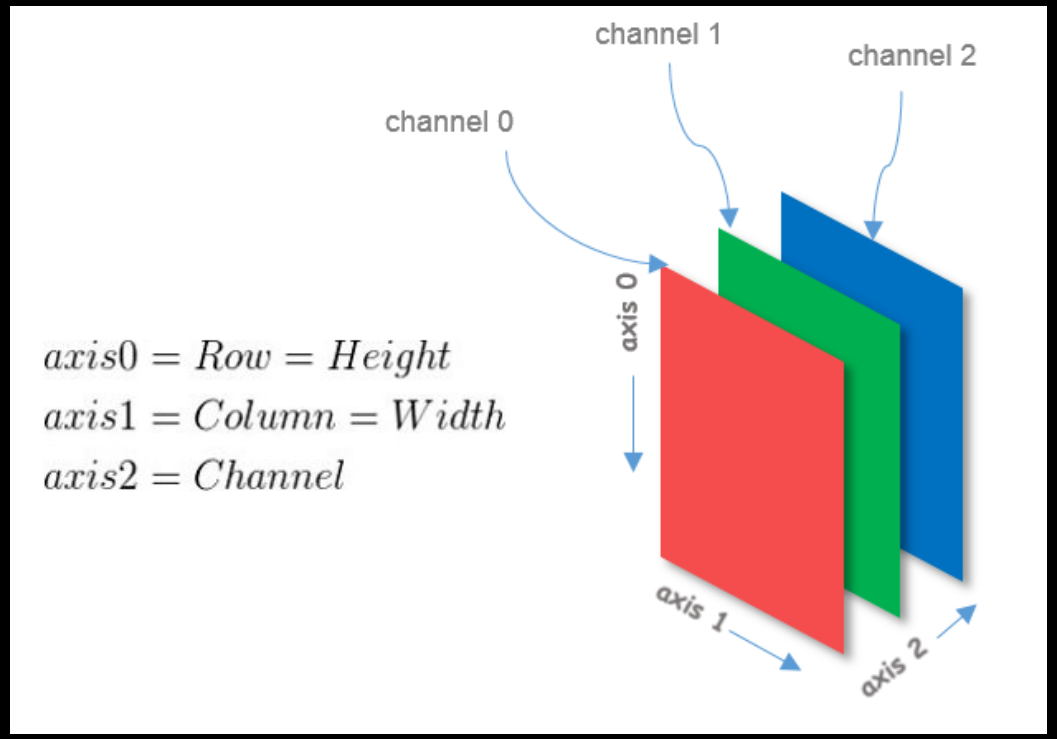
¿Qué entienden por visión por computadora?

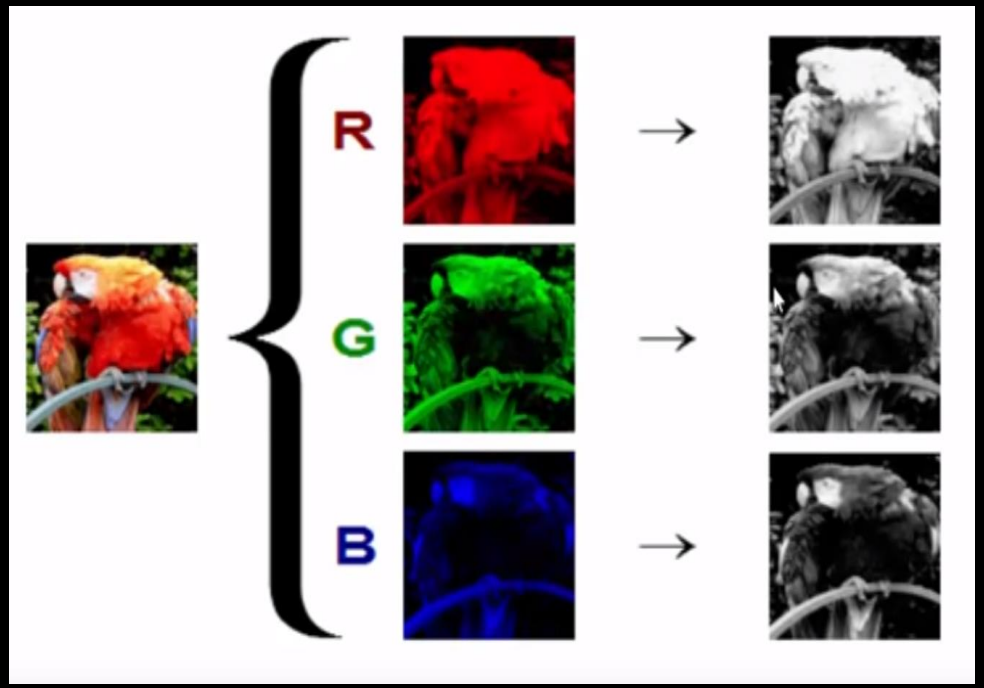
Visión por computadora

Es lo que permite ver y procesar datos visuales como nosotros los humanos

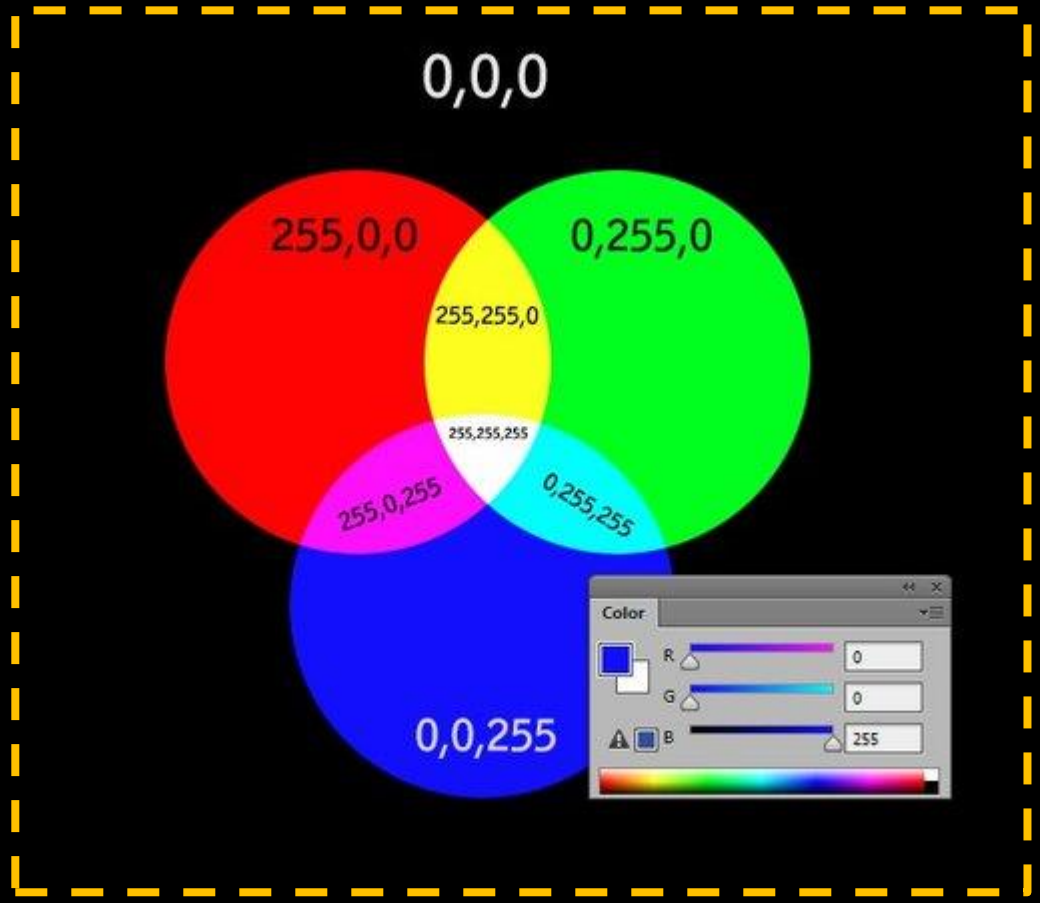
Canales de colores RGB

- Una imagen digital normalmente está compuesta por tres canales Rojo, Verde y Azul.





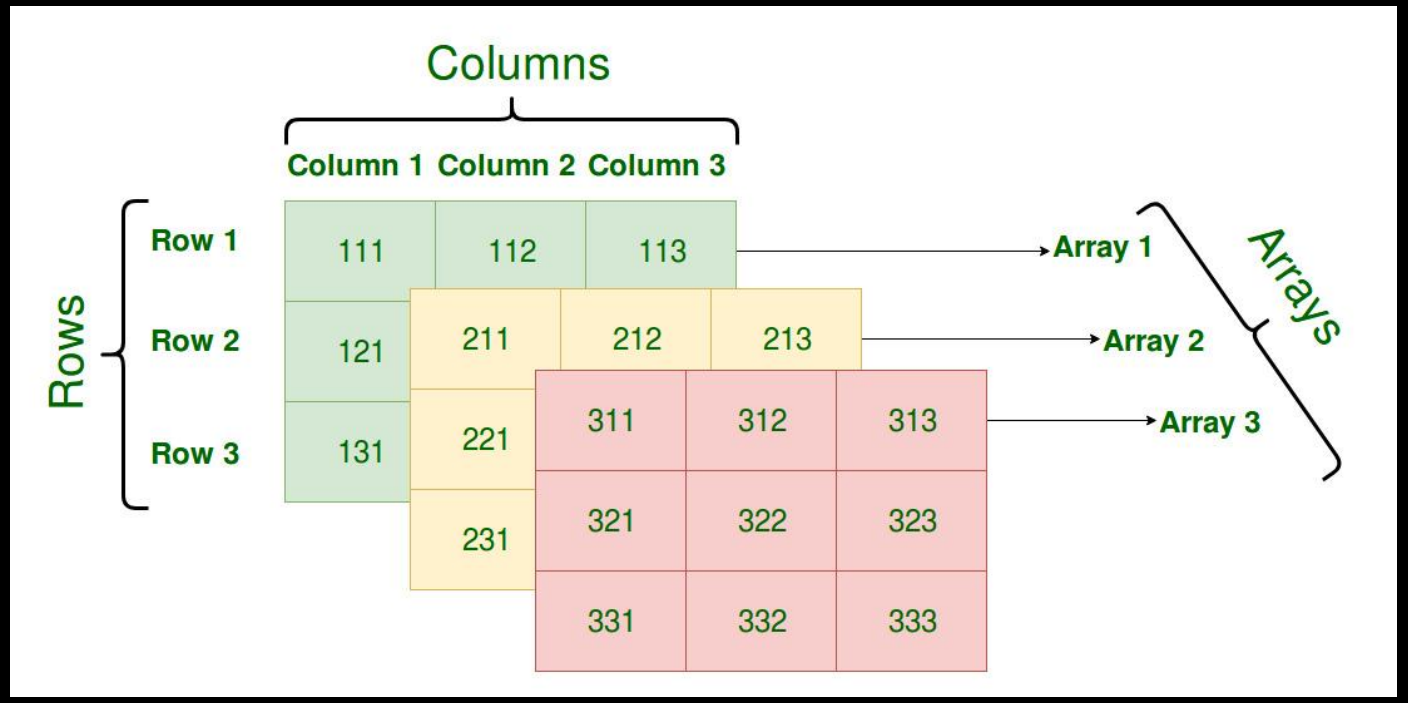
- En la escala RGB cada uno de los colores es representado por un entero entre 0 a 255, que indica cuanto de ese color tiene.
- La tupla (red,green,blue) representa el color
- Negro es (0,0,0) y Blanco (255,255,255)
- OpenCV almacena los canales en el formato (BGR)



- **Black:** (0,0,0)
- **White:** (255,255,255)
- **Red:** (255,0,0)
- **Green:** (0,255,0)
- **Blue:** (0,0,255)
- **Aqua:** (0,255,255)
- **Fuchsia:** (255,0,255)
- **Maroon:** (128,0,0)
- **Navy:** (0,0,128)
- **Olive:** (128,128,0)
- **Purple:** (128,0,128)
- **Teal:** (0,128,128)
- **Yellow:** (255,255,0)

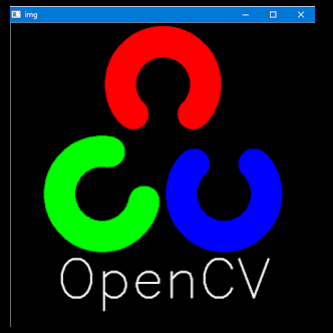
Imagen digital

- La librería OpenCV nos permite manipular las imágenes como un arreglo de 3 dimensiones, donde cada canal es un arreglo de dos dimensiones.



- *Los arreglos son el tipo de conjuntos de datos que maneja la librería numpy.*
- *Las matrices son un ejemplo de arreglos de numpy.*

Open CV

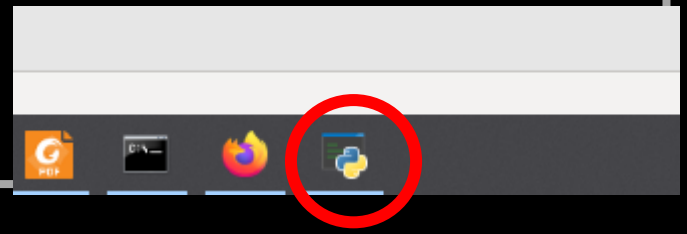


- Librería de software libre desarrollada por Intel, mantenido en la actualidad por la empresa de robótica Willow Garage
- OpenCv es multiplataforma ya que tiene versiones para diferentes sistemas operativos.
- Corre en C++, C, Phyton y Java.
- Cuenta con más de 500 funciones integradas
- Su primera versión estable fue liberada en 2006.

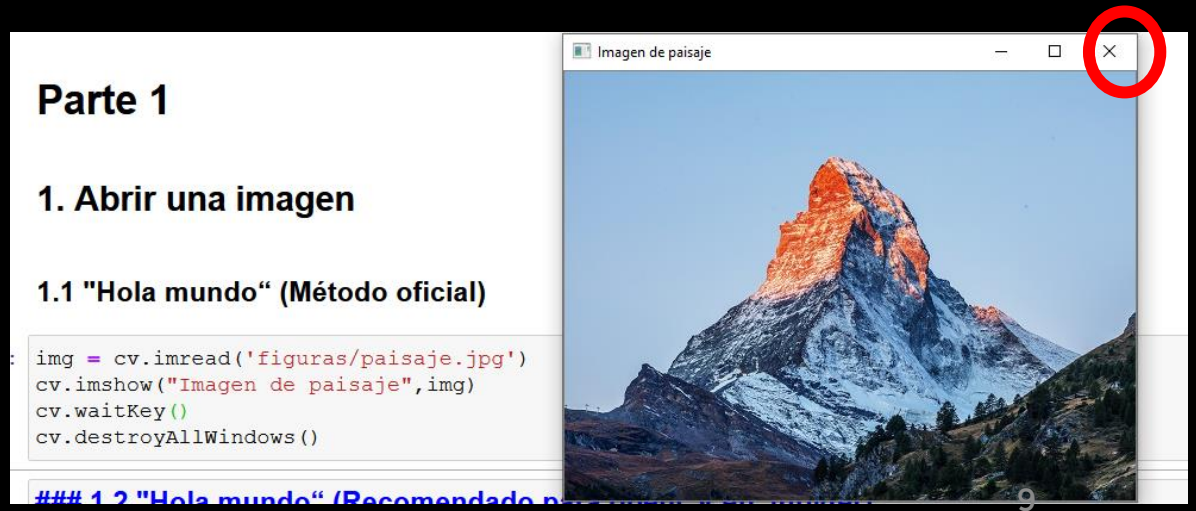
Abrir un archivo: "Hola mundo“ (El oficial)

```
import cv2 as cv

img = cv.imread('figuras/paisaje.jpg')
cv.imshow("Imagen de paisaje",img)
cv.waitKey()
cv.destroyAllWindows()
```



Tener cuidado porque abre una ventana emergente y el kernel queda en estado de espera “[*]” hasta que se cierre la imagen.

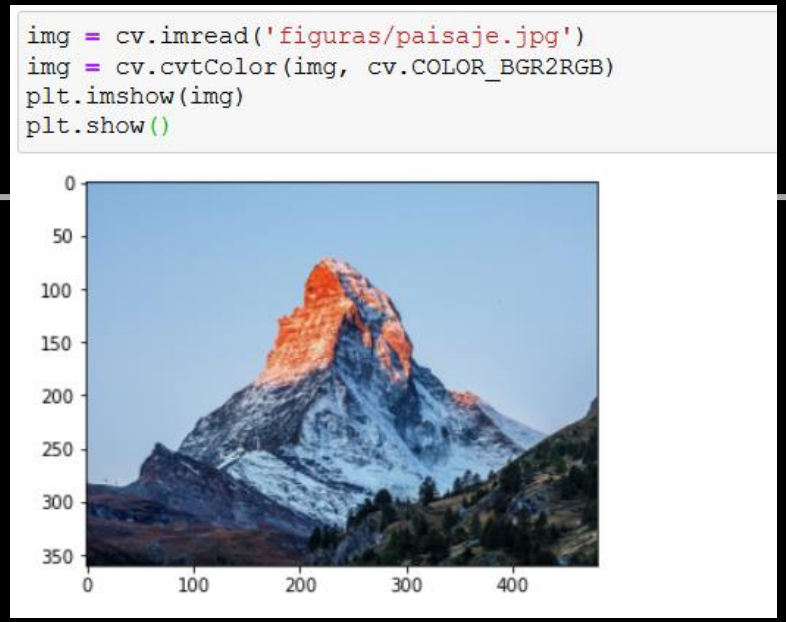


Abrir un archivo: "Hola mundo“ (Recomendado para openCV en Jupyter)

```
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv

img = cv.imread('figuras/paisaje.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
plt.show()
```

Esta opción es la que más estaremos utilizando, y usaremos la biblioteca matplotlib para desplegar la imagen



shape: propiedades básicas

- El siguiente código muestra como ver las propiedades de una imagen: alto, ancho y numero de canales

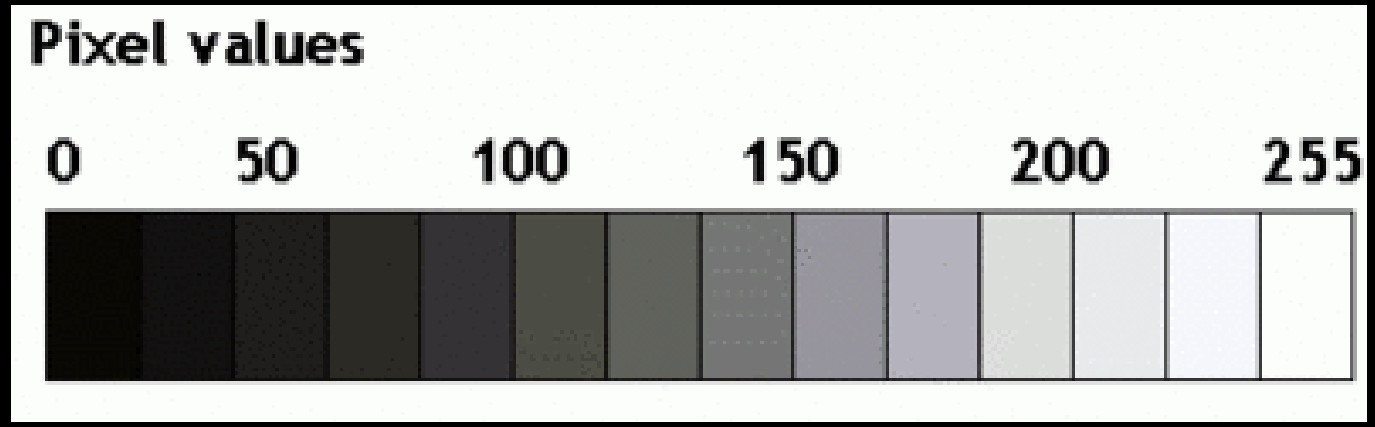
```
img = cv.imread('figuras/paisaje.jpg')
print("Función shape",img.shape)
print("alto:",img.shape[0],"pixeles")
print("ancho:", img.shape[1],"pixeles" )
print("canales: ", img.shape[2])
```

```
: img = cv.imread('figuras/paisaje.jpg')
  print("Función shape",img.shape)
  print("alto:",img.shape[0],"pixeles")
  print("ancho:", img.shape[1],"pixeles" )
  print("canales: ", img.shape[2])

Función shape (360, 480, 3)
alto: 360 pixeles
ancho: 480 pixeles
canales:  3
```

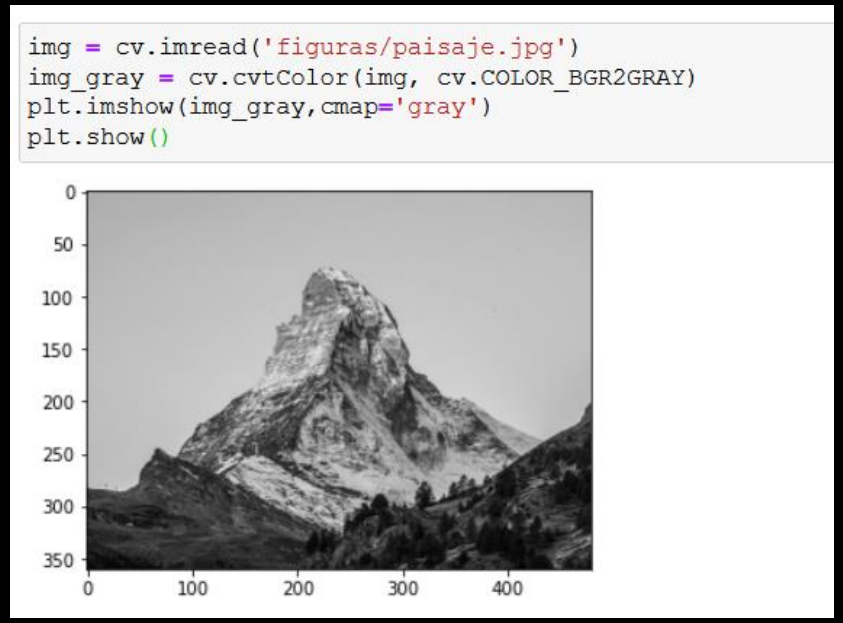
Escala de grises

- Muchas veces para simplificar el manejo de imágenes será necesario convertirla a escala de grises.
- El valor 0 es el negro
- EL valor 255 es el blanco



Conversión a escala de grises en openCV

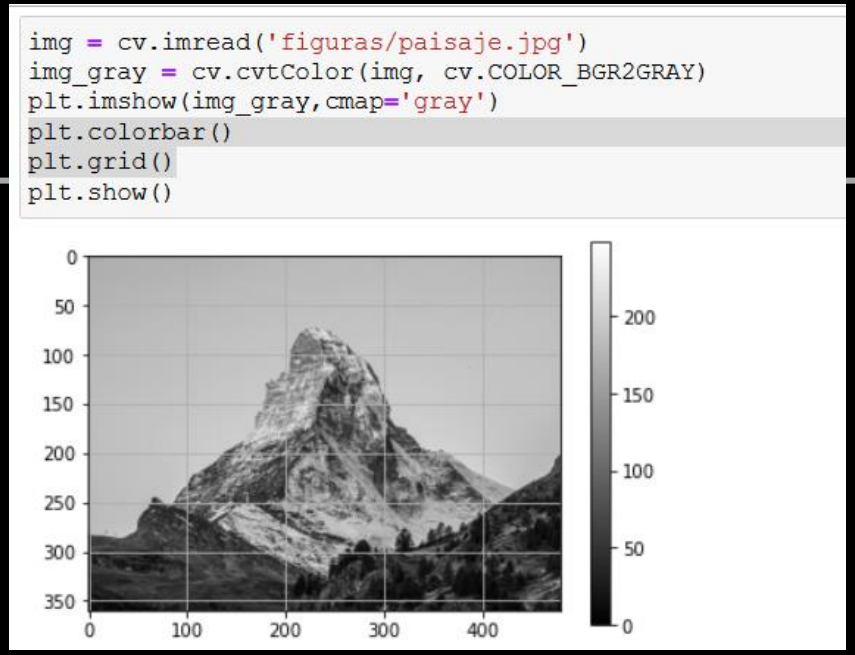
```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap='gray')
plt.show()
```



Grid y barra de escala de grises

```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap='gray’)
plt.colorbar()
plt.grid()
plt.show()
```

- Algunas veces para entender la imagen se puede agregar un grid así como la escala de colores.



Dimensiones de una imagen en escala de grises

- Cuando hemos convertido la imagen a escala de grises, desaparecen los canales RGB y solo nos queda un arreglo de dos dimensiones

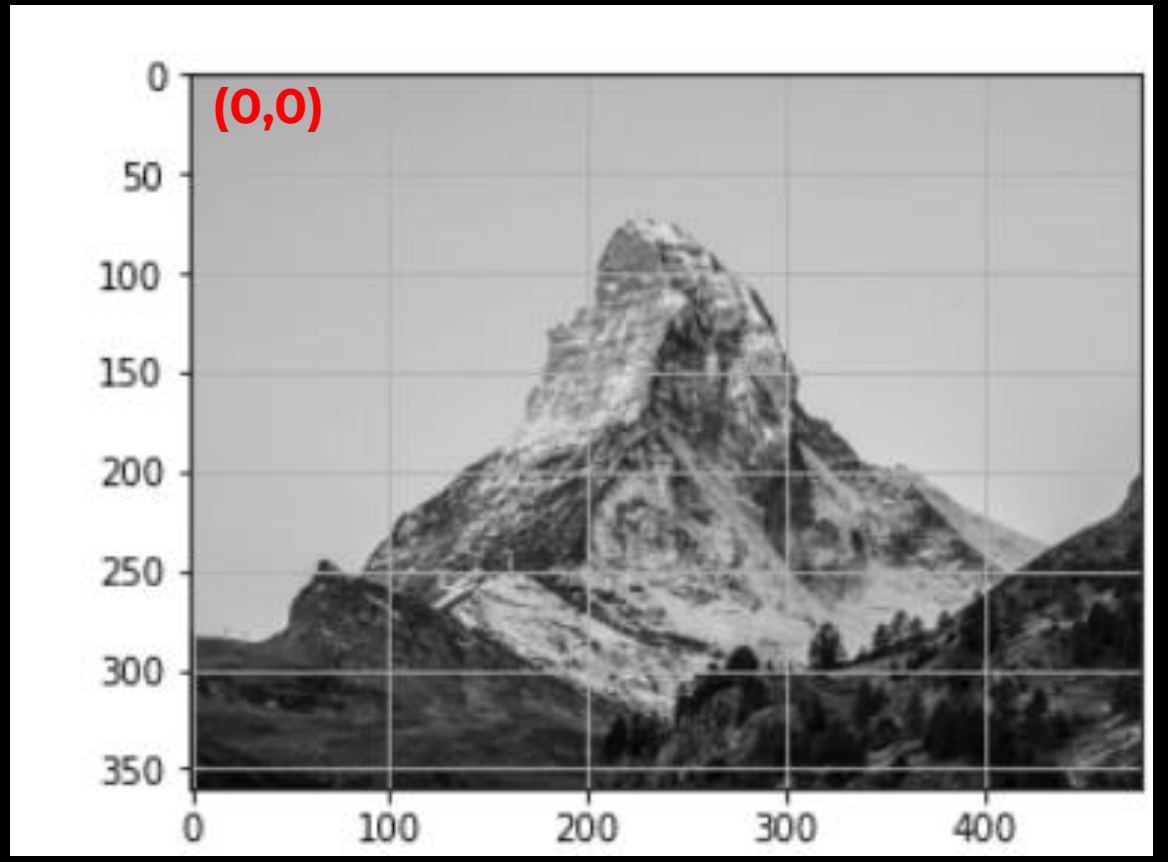
```
print("Función shape",img_gray.shape)
print("alto:",img_gray.shape[0],"pixeles")
print("ancho:", img_gray.shape[1],"pixeles" )
```

```
print("Función shape",img_gray.shape)
print("alto:",img_gray.shape[0],"pixeles")
print("ancho:", img_gray.shape[1],"pixeles" )
```

```
Función shape (360, 480)
alto: 360 pixeles
ancho: 480 pixeles
```

Sistema de coordenadas

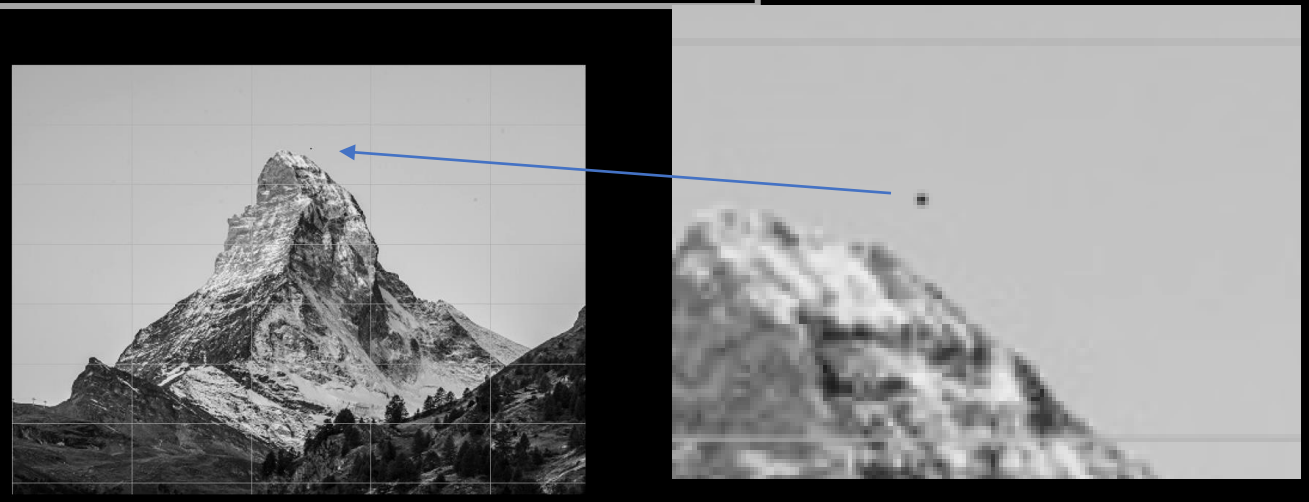
- El sistema tiene como origen la esquina izquierda comenzando con (0,0)



Cambiar el color a un pixel

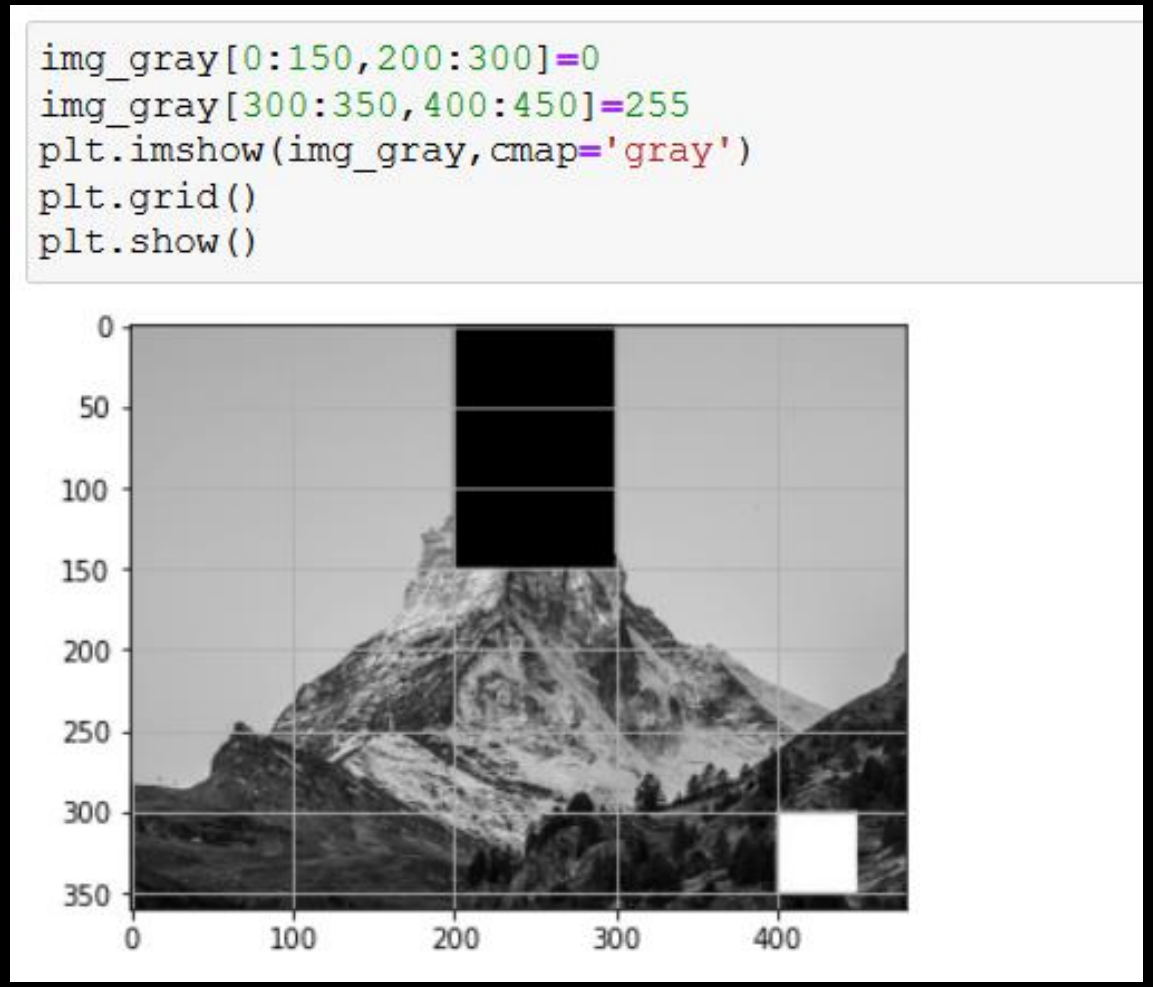
- Asignar al pixel (0,0) y al (70,250) el color negro

```
img_gray[0,0]=0
img_gray[70,250]=0
plt.imshow(img_gray,cmap='gray')
plt.grid()
plt.show()
```



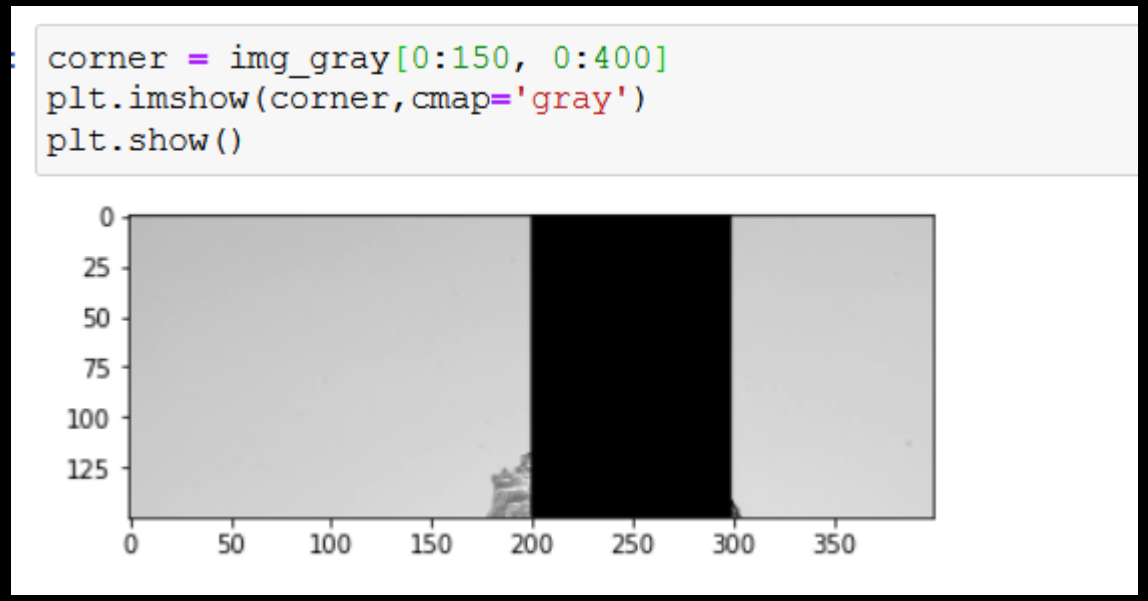
Cambiando zonas a negro y blanco

```
img_gray[0:150,200:300]=0
img_gray[300:350,400:450]=255
plt.imshow(img_gray,cmap='gray')
plt.grid()
plt.show()
```



Cortando una parte de la imagen

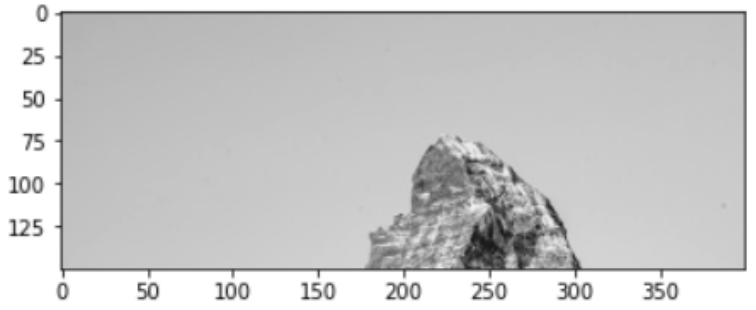
```
corner = img_gray[0:150, 0:400]
plt.imshow(corner,cmap='gray')
plt.show()
```



Cortando una parte de la imagen

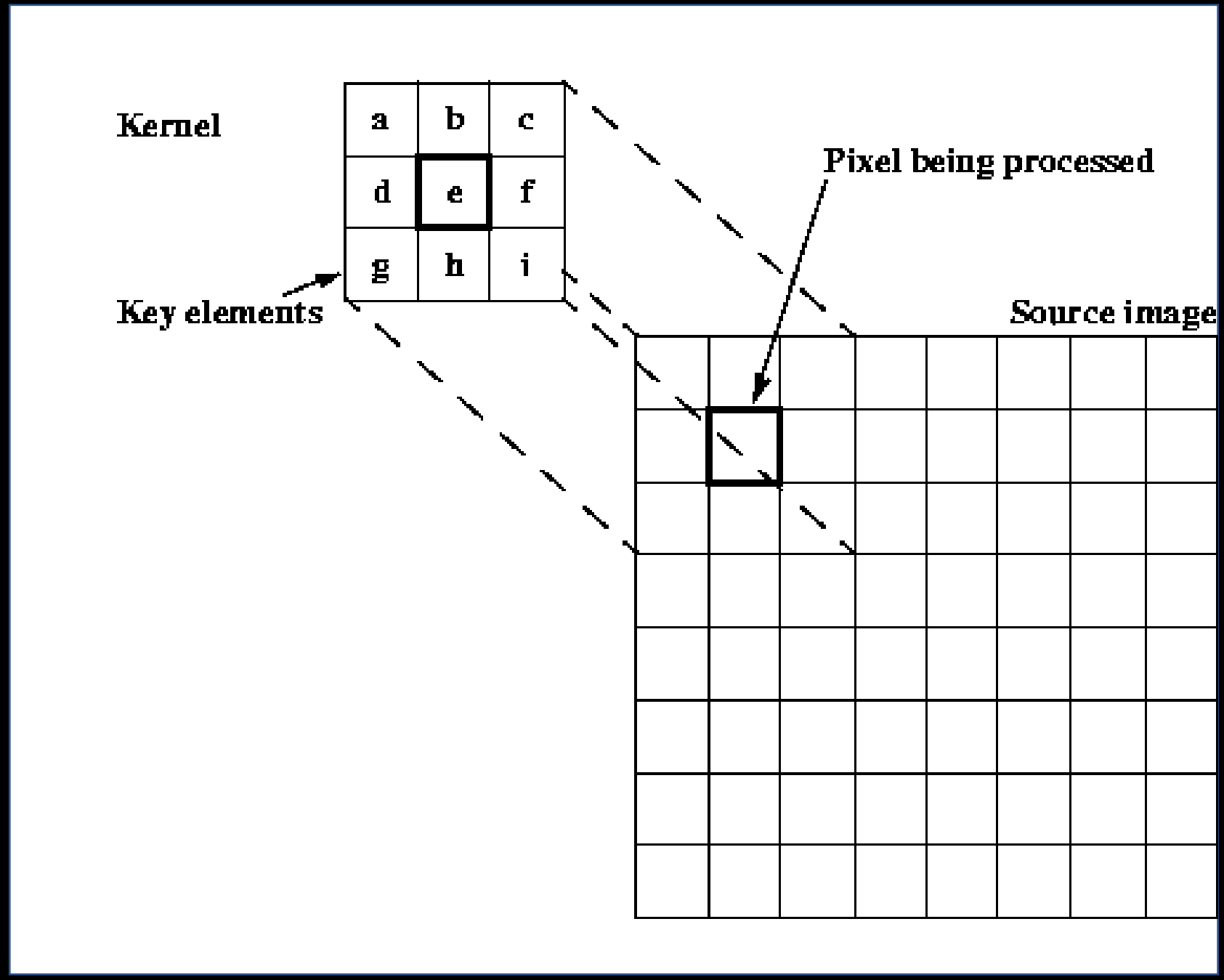
```
#Cargando nuevamente la imagen
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
corner = img_gray[0:150, 0:400]
plt.imshow(corner,cmap='gray')
plt.show()
```

```
#Cargando nuevamente la imagen
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
corner = img_gray[0:150, 0:400]
plt.imshow(corner,cmap='gray')
plt.show()
```



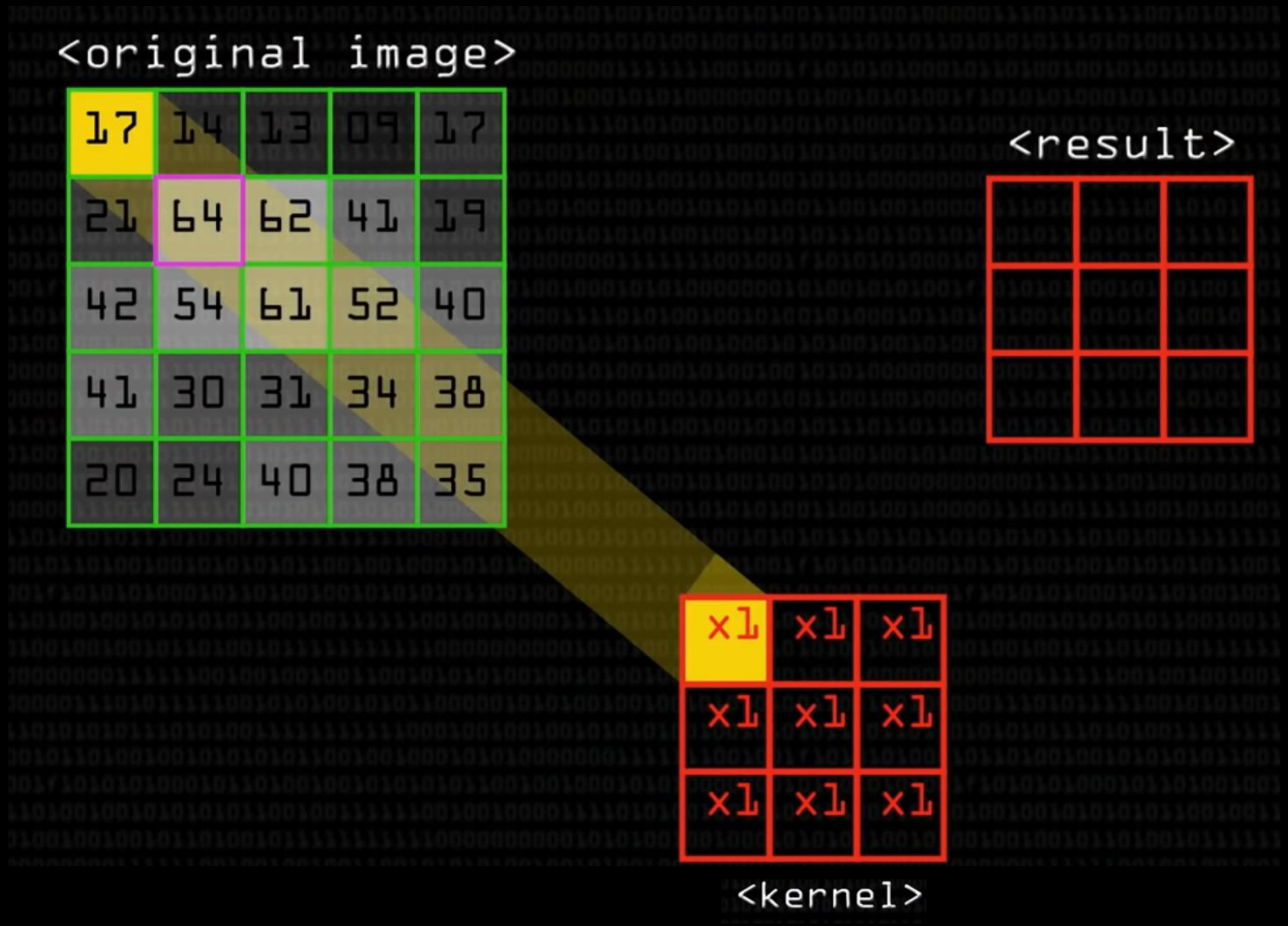
Filtros

- Transforman imágenes en nuevas imágenes
- Los más comunes son Medio, Gaussiano y Mediana.



Filtro medio

Elementos=17,14,13,21,64,62,42,54,6



Suma=17+14+13+21
+64+62+42+54+61

Suma=348

Promedio=348/9=38.66

Promedio = 39

<original image>

17	14	13	09	17
21	64	62	41	19
42	54	61	52	40
41	30	31	34	38
20	24	40	38	35

<result>

17	14	13
21	64	62
42	54	61

x1	x1	x1
x1	x1	x1
x1	x1	x1

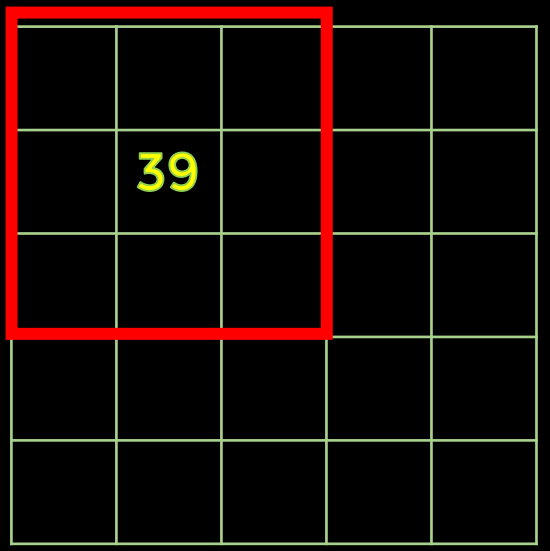
<kernel>

Suma=348

Promedio=348/9=38.66

Promedio = 39

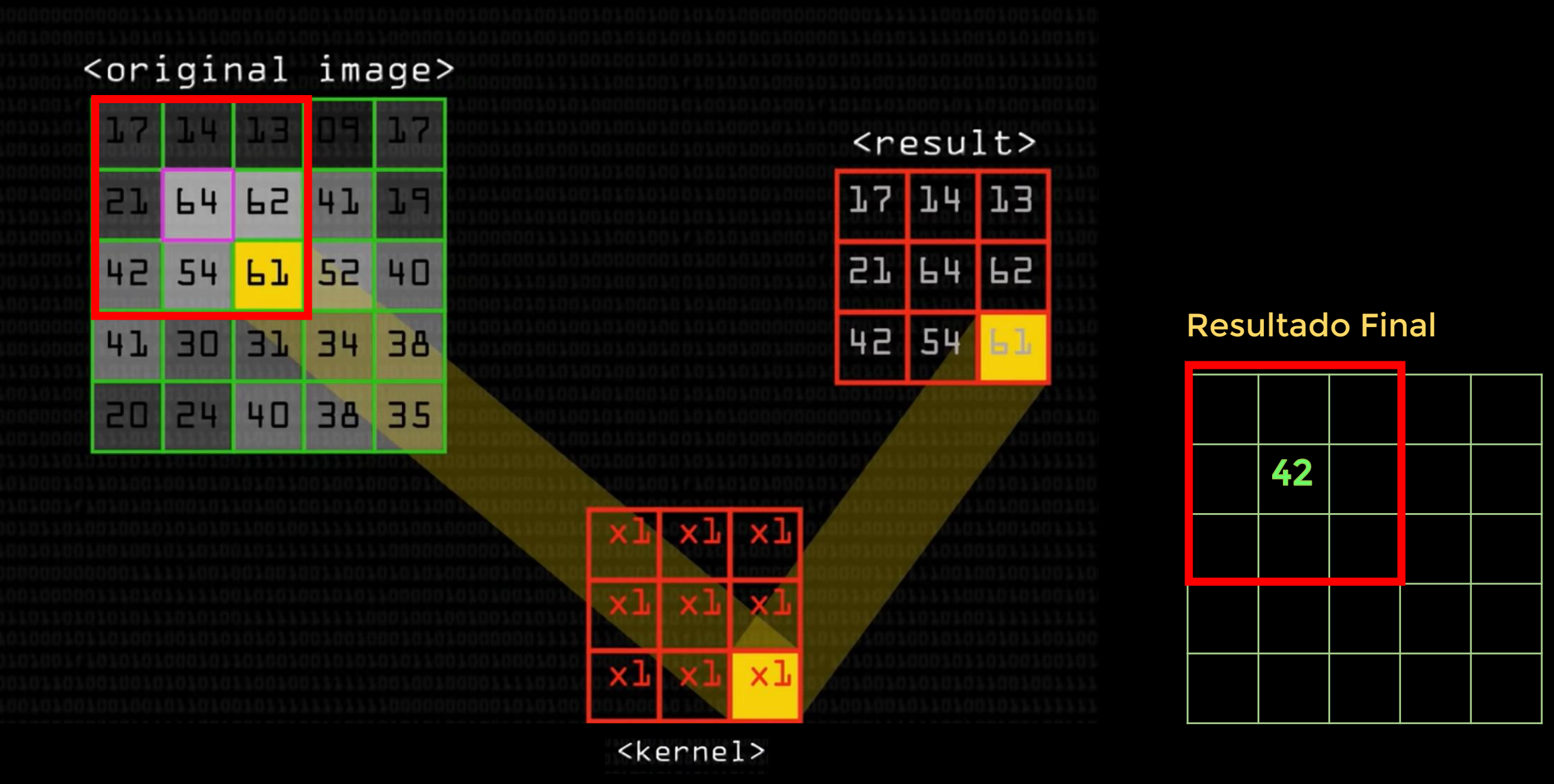
Resultado Final



Filtro mediana

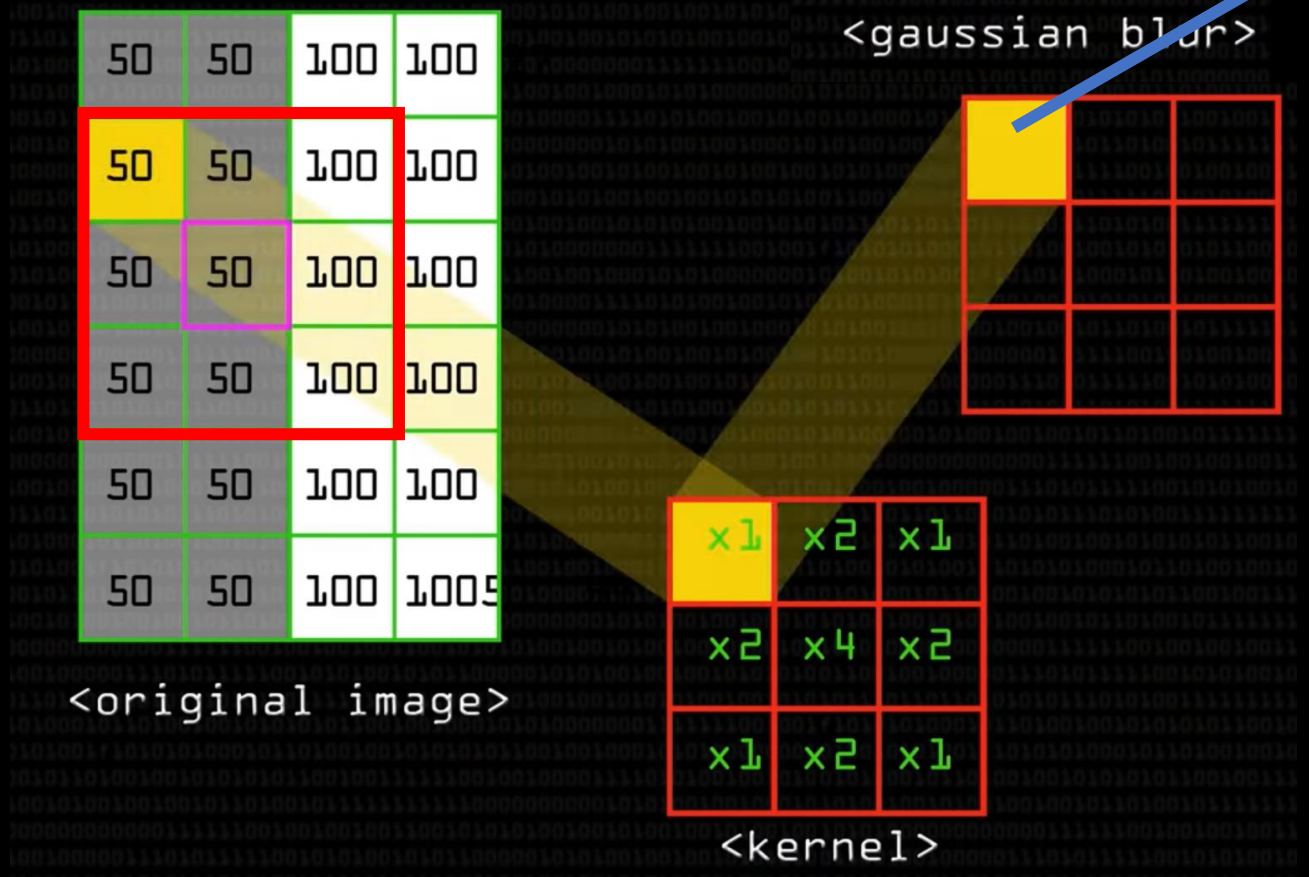
Elementos=17,14,13,21,64,62,42,54,61

Elementos ordenados de mayor a menor=13,14,17,21,42,54,6



Resultado Final

Filtro Gauss



Suma=
50*1 + 50*2+ 100*1 +
50*2 + 50*4 + 100*2 +
50*1 + 50*2 + 100*1
= 1000

Suma=1000

Promedio=1000/16=62.5

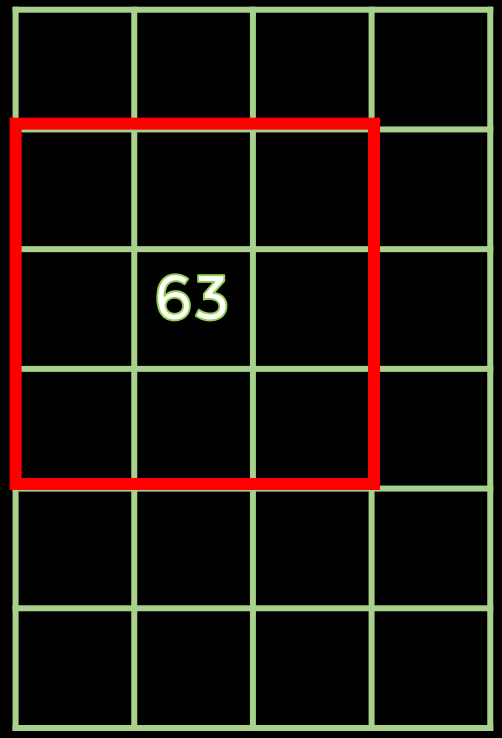
Promedio = 63

<original image>

<gaussi

50	100	100
100	200	200
50	100	100

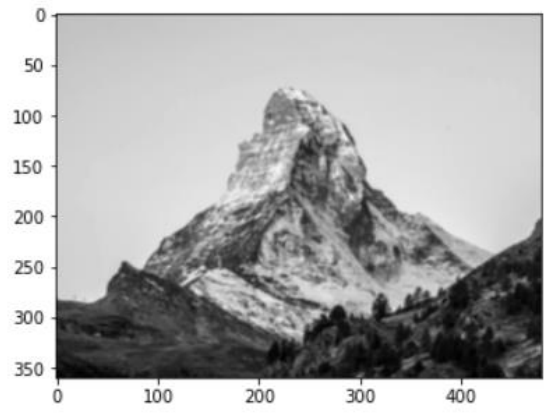
<kernel>



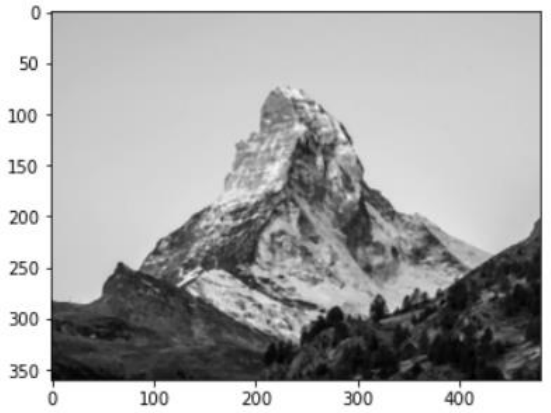
Diferentes filtros

```
filtro_media=cv.blur(img_gray,(3,3) )
filtro_mediana=cv.medianBlur(img_gray ,3)
filtro_gauss=cv.GaussianBlur(img_gray ,(3,3),0)
```

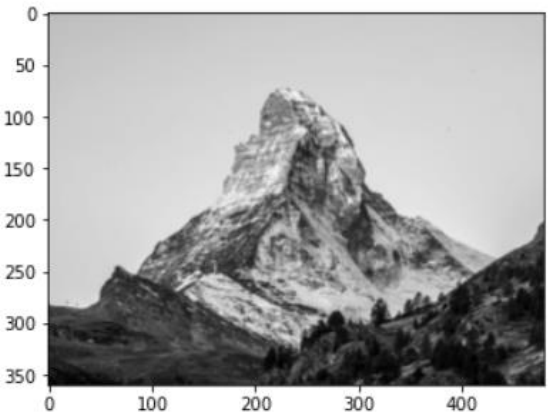
```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_fmedia = cv.blur(img_gray,(3,3) )
plt.imshow(img_fmedia,cmap='gray')
plt.show()
```




```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_fmediana =cv.medianBlur(img_gray ,3)
plt.imshow(img_fmediana,cmap='gray')
plt.show()
```



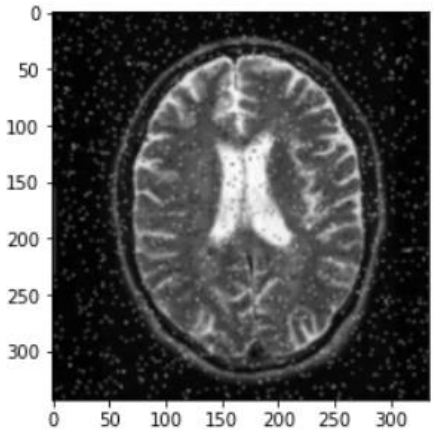

```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
plt.imshow(img_gauss,cmap='gray')
plt.show()
```



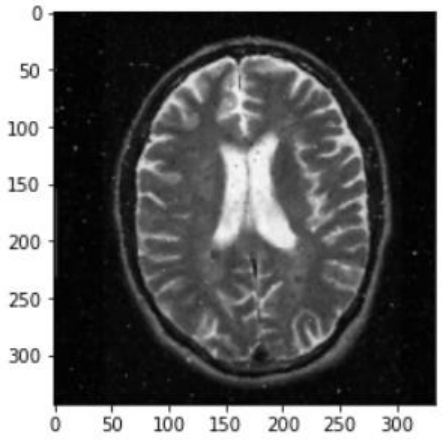
Aplicación del filtro de mediana para quitar el ruido de puntos blancos

Imagen original

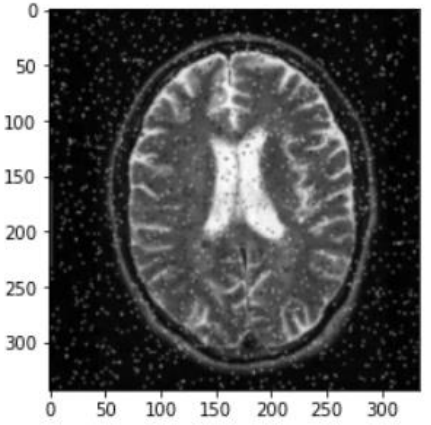
```
img = cv.imread('figuras/cerebro.png')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_fmedia = cv.blur(img_gray,(3,3) )
plt.imshow(img_fmedia,cmap='gray')
plt.show()
```




```
img = cv.imread('figuras/cerebro.png')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_fmediana =cv.medianBlur(img_gray ,3)
plt.imshow(img_fmediana,cmap='gray')
plt.show()
```

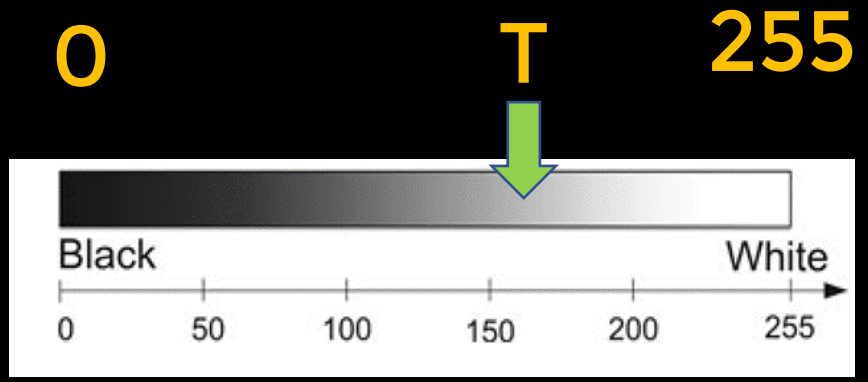


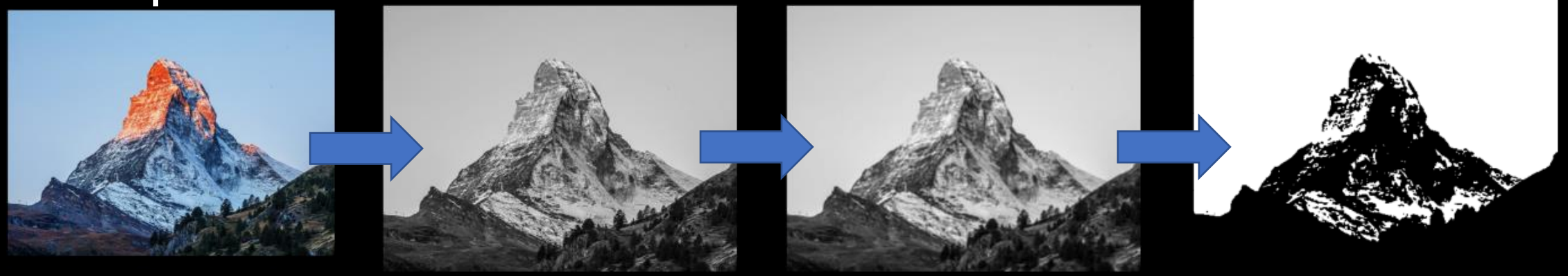

```
img = cv.imread('figuras/cerebro.png')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
plt.imshow(img_gauss,cmap='gray')
plt.show()
```



Thresholding

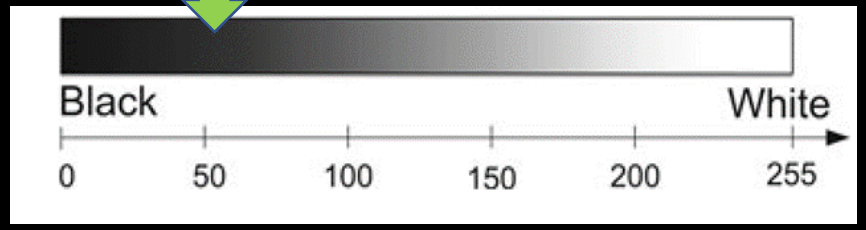
- Thresholding es la binarización de una imagen (conversión a blanco/negro)
- Los pasos son los siguientes
  1. Cargar la imagen
  2. Convertir a escala de grises
  3. Aplicar el filtro Gaussiano
  4. Aplicar la binarización





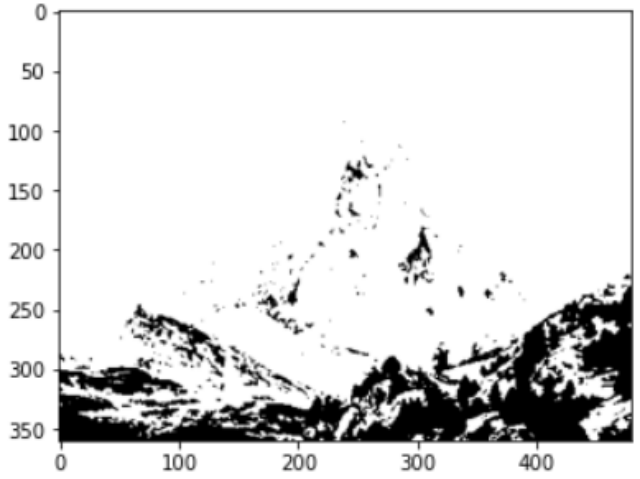
T=50

0 T=50 255



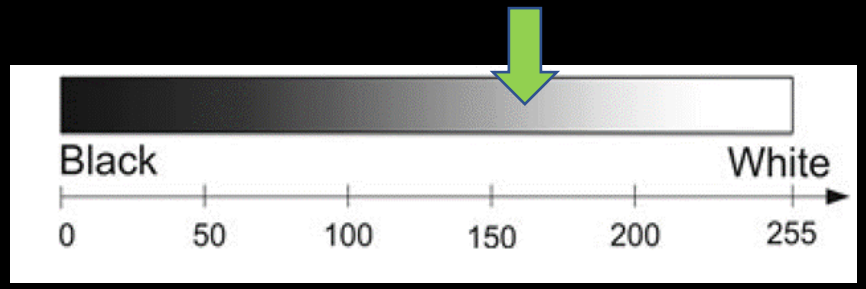
Los valores mayores a 50 se eliminan

```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
thr, img_thr= cv.threshold(img_gauss ,50 ,255,cv.THRESH_BINARY)
plt.imshow( img_thr,cmap='gray')
plt.show()
```



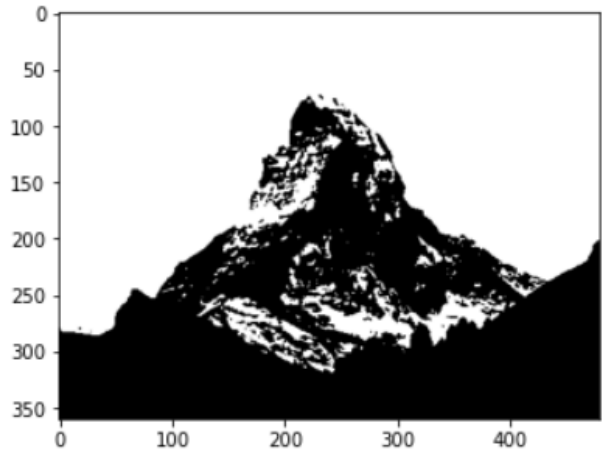
T=160

0 T=160 255



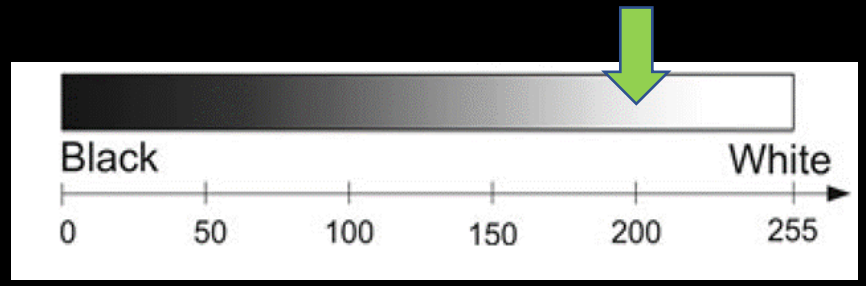
Los valores mayores a 160 se eliminan

```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
thr, img_thr= cv.threshold(img_gauss ,160,255,cv.THRESH_BINARY)
plt.imshow( img_thr,cmap='gray')
plt.show()
```



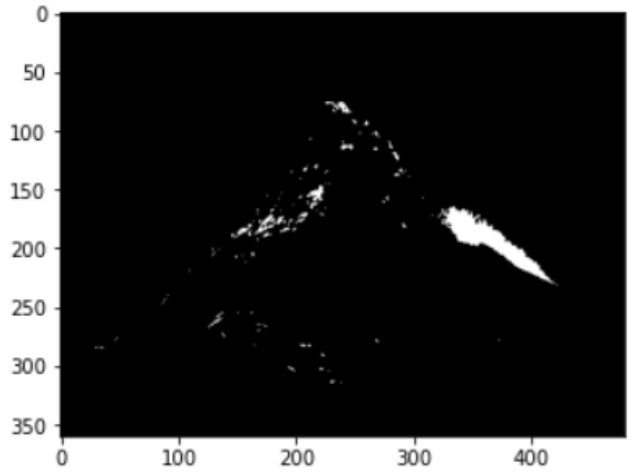
T=200

0 T=200255



Los valores mayores a 200 se eliminan

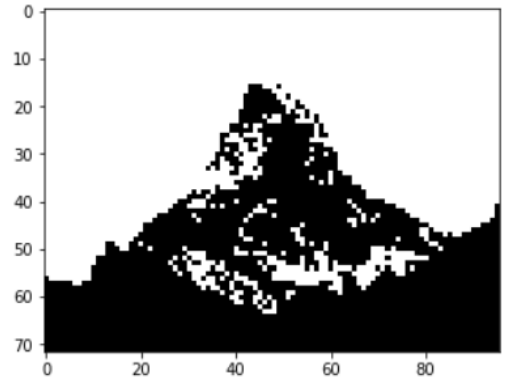
```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
thr, img_thr= cv.threshold(img_gauss ,200 ,255,cv.THRESH_BINARY)
plt.imshow( img_thr,cmap='gray')
plt.show()
```



Resize o cambio de tamaño

```
#Cargando nuevamente la imagen
alto=img.shape[0]
ancho=img.shape[1]
ratio=0.2
img_r = cv.resize(img_thr,(int(ancho*ratio),int(alto*ratio)), interpolation=cv.INTER_NEAREST)
plt.imshow( img_r,cmap='gray')
plt.show()
```

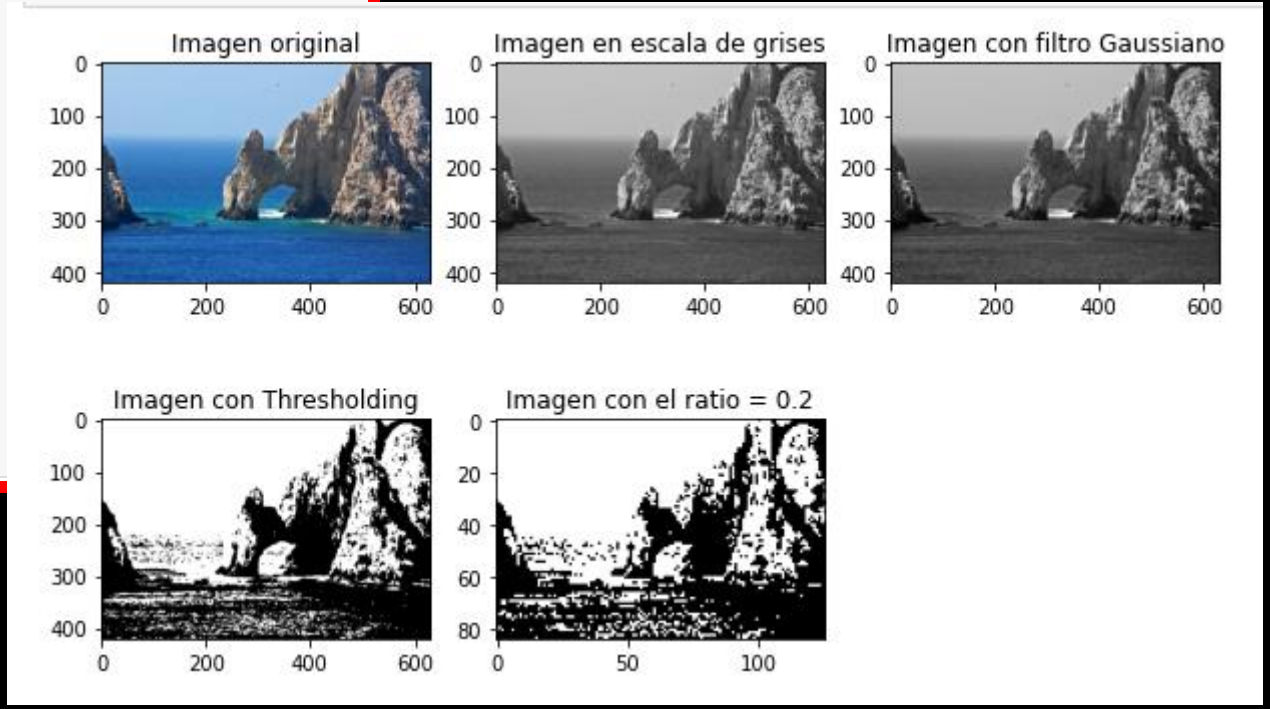
```
img = cv.imread('figuras/paisaje.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
thr, img_thr= cv.threshold(img_gauss ,160 ,255,cv.THRESH_BINARY)
alto=img.shape[0]
ancho=img.shape[1]
ratio=0.2
img_r = cv.resize(img_thr,(int(ancho*ratio),int(alto*ratio)), interpolation=cv.INTER_NEAREST)
plt.imshow( img_r,cmap='gray')
plt.show()
```



Todo el proceso

```
img = cv.imread('figuras/paisaje.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
thr, img_thr= cv.threshold(img_gauss ,160 ,255,cv.THRESH_BINARY)
alto=img.shape[0]
ancho=img.shape[1]
ratio=0.2
img_r = cv.resize(img_thr,(int(ancho*ratio),int(alto*ratio)), interpolation=cv.INTER_NEAREST)

plt.figure(figsize=(10,6))
plt.subplot(2,3,1)
plt.imshow(img)
plt.title("Imagen original")
plt.subplot(2,3,2)
plt.imshow(img_gray,cmap='gray')
plt.title("Imagen en escala de grises")
plt.subplot(2,3,3)
plt.imshow(img_gauss, cmap='gray')
plt.title("Imagen con filtro Gaussiano")
plt.subplot(2,3,4)
plt.imshow(img_thr, cmap='gray')
plt.title("Imagen con Thresholding")
plt.subplot(2,3,5)
plt.imshow(img_r, cmap='gray')
plt.title("Imagen con el ratio = " + str(ratio))
plt.show()
```



Parte II:

Operaciones aritméticas en imágenes

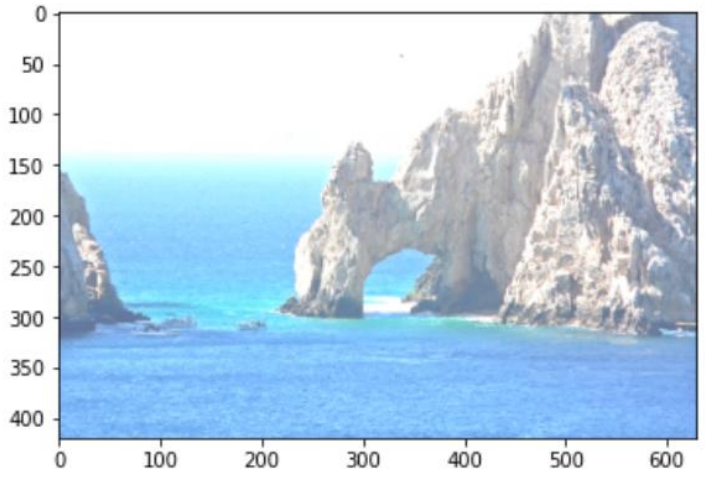
Aumentar la intensidad

```
M = np.ones(img.shape, dtype="uint8") * 100
added = cv.add(img, M)
plt.imshow(added)
```

Aumentar la intensidad

```
# Aumentar las intensidades de píxeles en nuestra imagen en 100
M = np.ones(img.shape, dtype="uint8") * 100
added = cv.add(img, M)
plt.imshow(added)
```

```
<matplotlib.image.AxesImage at 0x22faf662ef0>
```



Disminuir la intensidad

```
M = np.ones(img.shape, dtype="uint8") * 50
subtracted = cv.subtract(img, M)
plt.imshow(subtracted)
```

Disminuir la intensidad

```
M = np.ones(img.shape, dtype="uint8") * 50
subtracted = cv.subtract(img, M)
plt.imshow(subtracted)
```

```
<matplotlib.image.AxesImage at 0x22faf701cf8>
```

Alta y baja intensidad

```
plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.imshow(img)
plt.title("Imagen original")
plt.subplot(1,3,2)
plt.imshow(added )
plt.title("Imagen con alta intensidad")
plt.subplot(1,3,3)
plt.imshow(subtracted )
plt.title("Imagen con baja intensidad")
plt.show()
```

Evidencias de la tarea

- Seguir la misma notebook utilizando una imagen de algún sitio arqueológico

- Una vez terminado el ejercicio, subirlo a la plataforma, asegúrese de colocar su nombre y apellido:
- PI_02_Nombre-Apellido.ipynb

NOTA: No subir archivos comprimidos