



RECONOCIMIENTO DE PATRONES EN IMÁGENES TICS 585

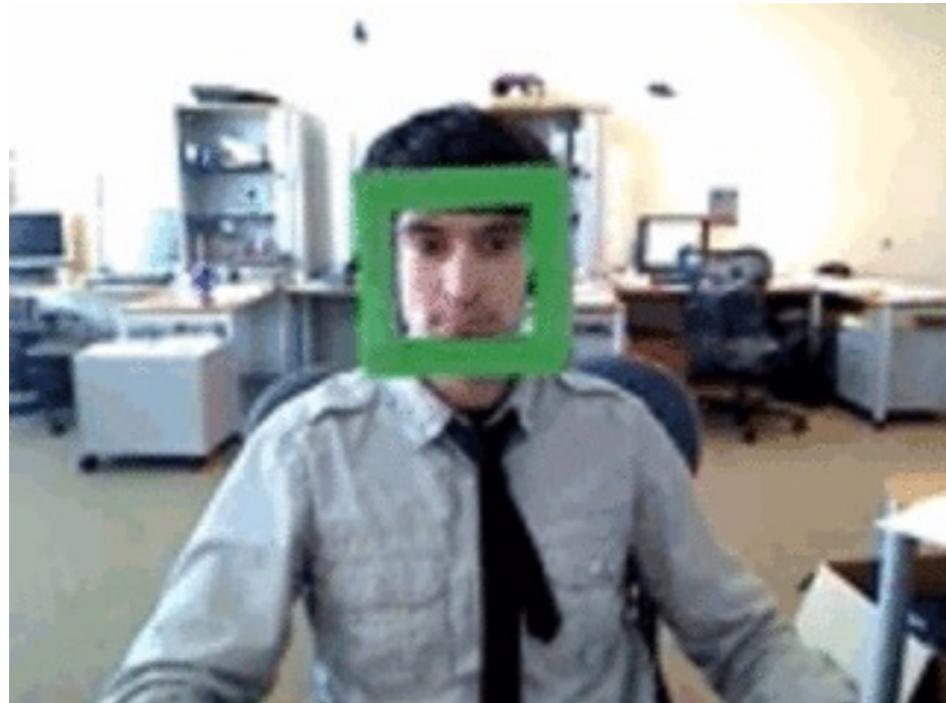
FACULTAD DE INGENIERÍA Y CIENCIAS
UNIVERSIDAD ADOLFO IBÁÑEZ

SEGUNDO SEMESTRE 2021

PROFESOR: MIGUEL CARRASCO

INTRODUCCIÓN A OPENCV

- Real time face recognition



Neutral	0 %
Happy	0 %
Surprise	0 %
Angry	0 %
Disgust	0 %
Fear	0 %
Sad	0 %

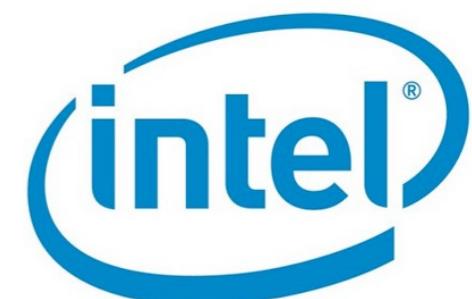
Status:
* Source: Webcam
* Player: Playing
* Face: Found
* Markers: Scale to face

Hint:
Press "Fit mask" to fit the

- OpenCV es una librería de visión por computador originalmente diseñada por INTEL en 1999.

Características

- Escrito en C/ C++ con más de 500 funciones
- Multiplataforma: Windows, Linux, MacOS, Android, IOS
- Libre para desarrollo e investigación



Funcionalidades

- Análisis de procesamiento de imágenes
- Análisis de estructuras
- Reconocimiento de patrones
- Momentos estadísticos
- Análisis de movimiento y tracking
- Reconstrucción 3D y Calibración
- Adquisición de imágenes e interfaz gráfica



- OpenCV es una librería de visión por computador originalmente diseñada por INTEL en 1999.

Algunos usos

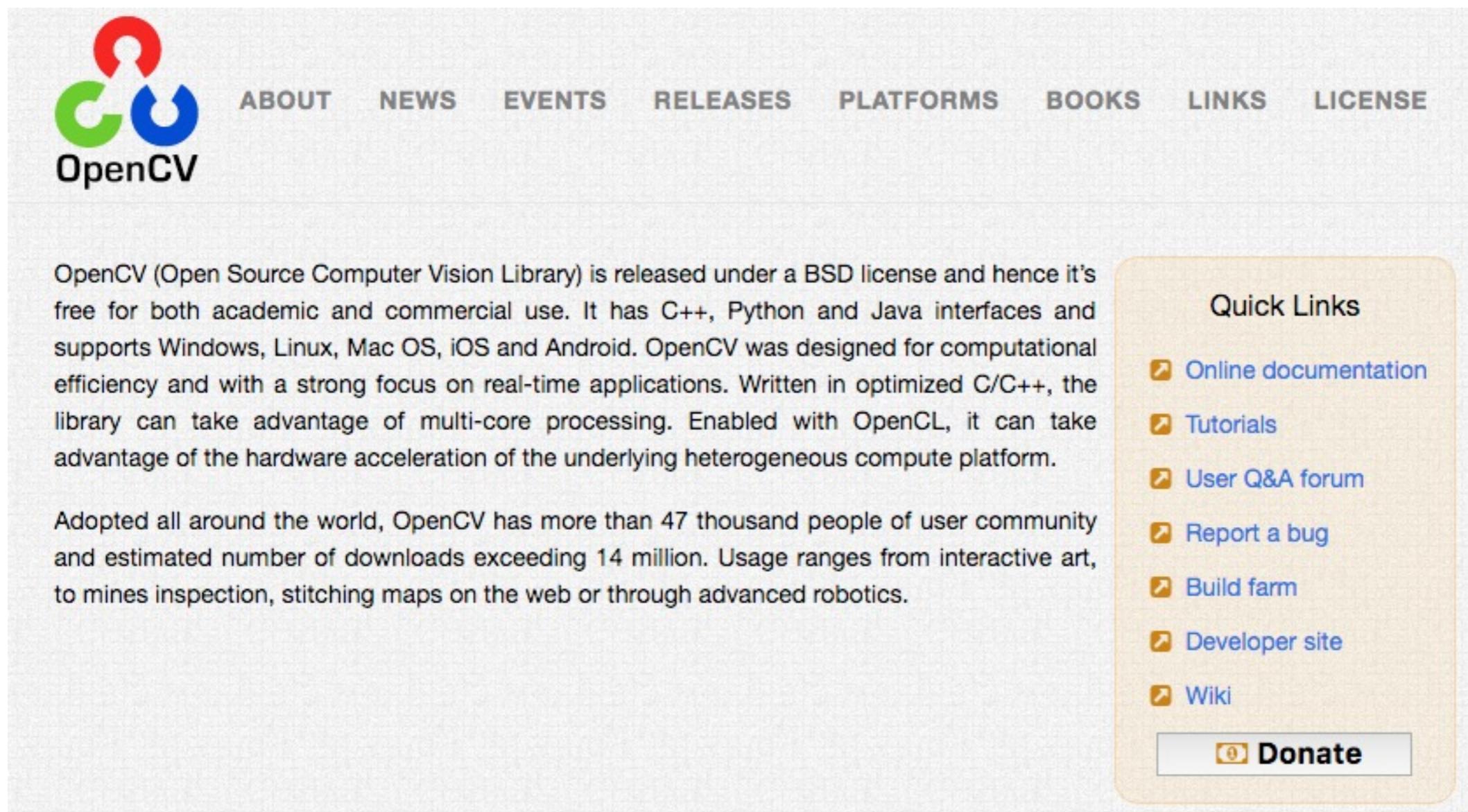
- Identificación y reconocimiento de objetos
- Reconocimiento de gestos y caras
- Interfaz humano-computador
- Procesamiento de imágenes
- Robótica
- Tracking
- Vigilancia
- Movimiento de estructuras
- Inspección industrial
- Análisis médico
- Modelación topográfica



- Página oficial con el proyecto

<http://opencv.org/>

Documentación en: <http://opencv.willowgarage.com/documentation/c/>



The screenshot shows the official OpenCV website. At the top, there is a navigation bar with the OpenCV logo on the left and links for ABOUT, NEWS, EVENTS, RELEASES, PLATFORMS, BOOKS, LINKS, and LICENSE. Below the navigation bar, there is a large text block about the library's history and features. To the right, there is a yellow sidebar titled "Quick Links" containing links to online documentation, tutorials, user forums, reporting bugs, building, developer sites, and a wiki. There is also a "Donate" button at the bottom of the sidebar.

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

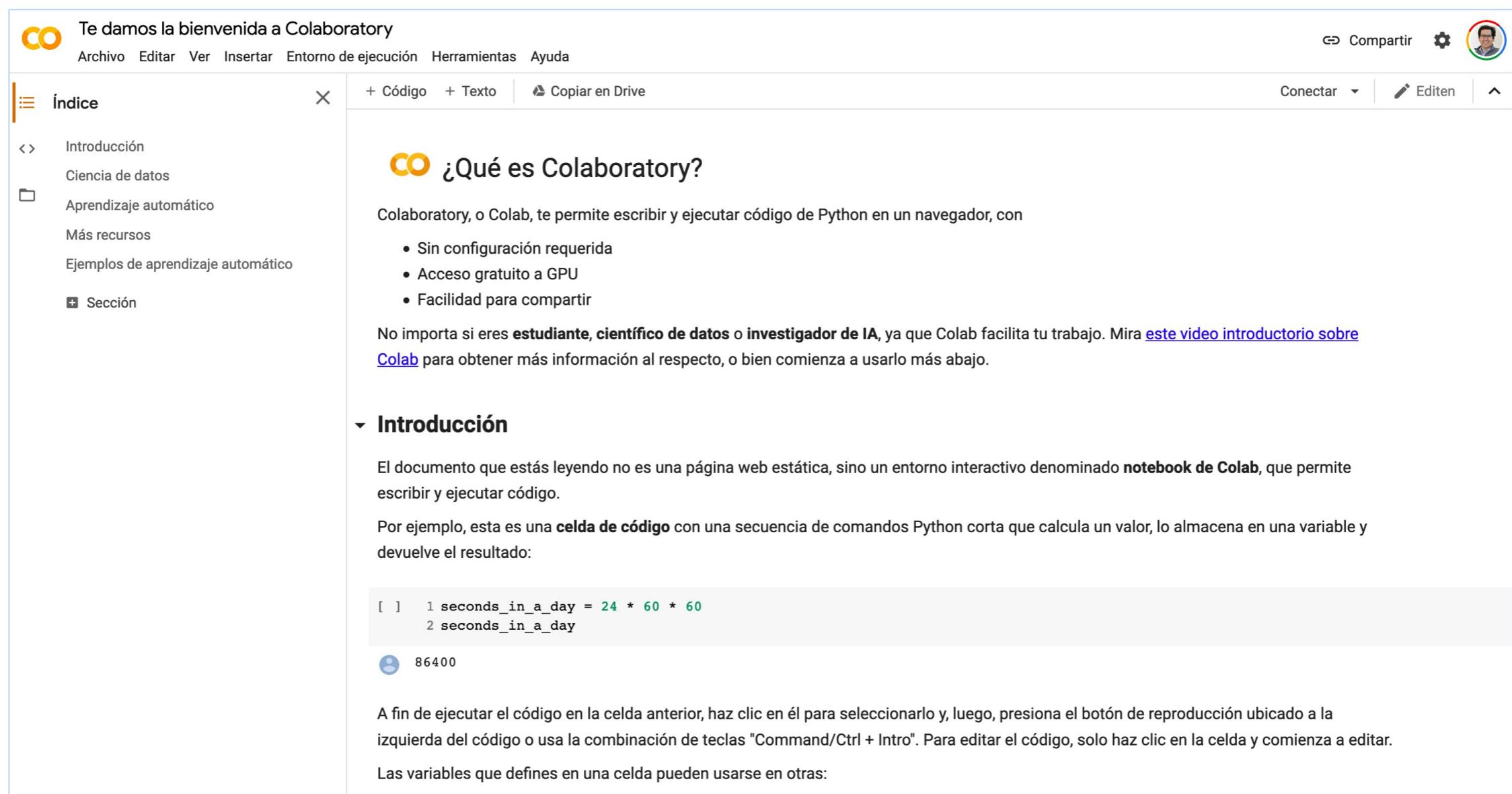
Quick Links

- Online documentation
- Tutorials
- User Q&A forum
- Report a bug
- Build farm
- Developer site
- Wiki

Donate

Ambientes de trabajo

online



The screenshot shows the Google Colaboratory interface. On the left, there's a sidebar with a tree view of notebooks: 'Índice' (Index), 'Introducción' (Introduction), 'Ciencia de datos' (Data Science), 'Aprendizaje automático' (Machine Learning), 'Más recursos' (More resources), 'Ejemplos de aprendizaje automático' (Machine learning examples), and '+ Sección' (New section). The main content area has a title '¿Qué es Colaboratory?' (What is Colaboratory?). It explains that Colaboratory or Colab allows writing and executing Python code in a browser. It lists benefits: no configuration required, free GPU access, and easy sharing. It also links to an introductory video. Below this, a section titled 'Introducción' (Introduction) discusses how it's an interactive notebook where you can write and run code. It shows a code cell with the following Python code:

```
[ ] 1 seconds_in_a_day = 24 * 60 * 60
2 seconds_in_a_day
```

It also shows a user icon with the number 86400. At the bottom, it says you can execute the code by clicking on the cell and pressing Command/Control + Enter.

► <https://colab.research.google.com/>

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

- Los tres canales RGB corresponden a la mezcla aditiva de los colores primarios.



Colores primarios:
Rojo-Verde-Azul

amarillo
(255,255,0)

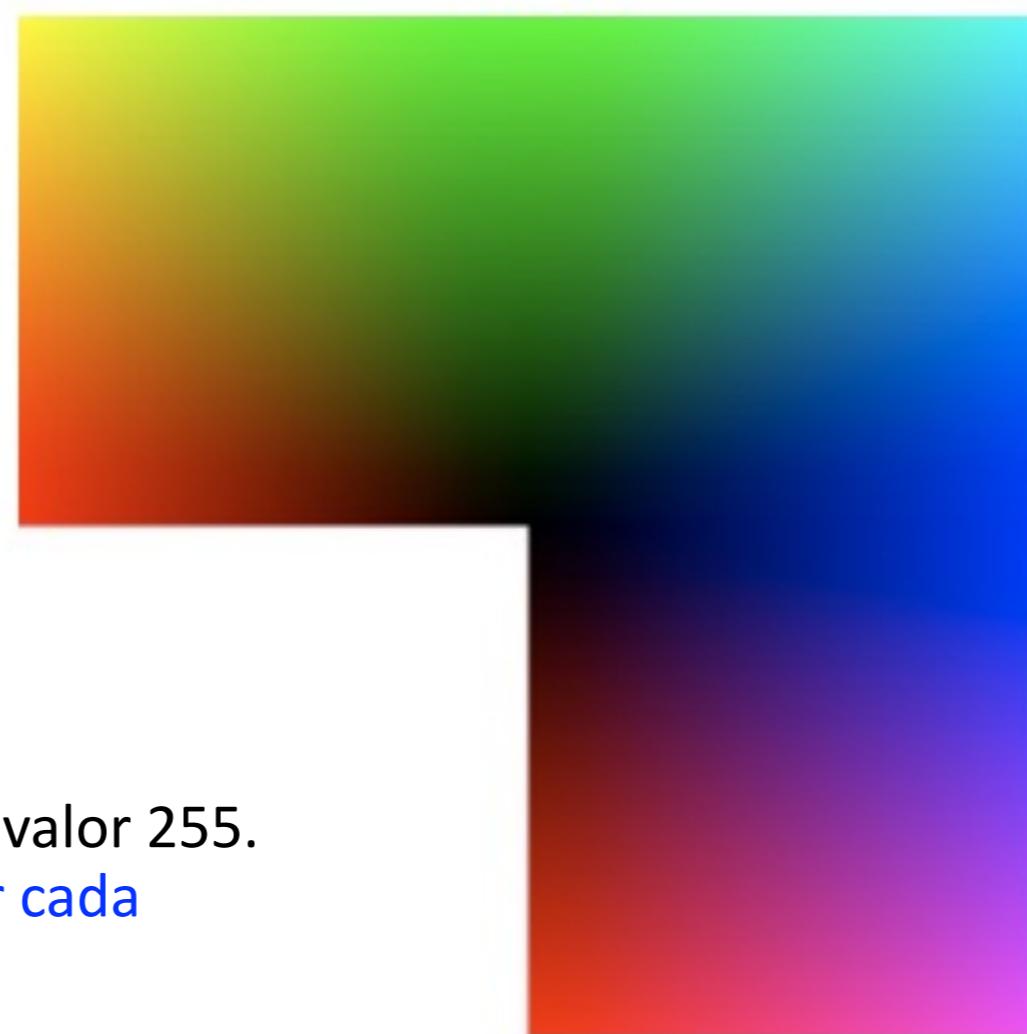
verde
(0,255,0)

cyan
(0,255,255)

rojo
(255,0,0)

azul
(0,0,255)

- Cada canal RGB va desde el valor 0 al valor 255. Por lo tanto, son necesarios **8 bits por cada canal**.
- La suma de los tres canales genera **24 bits**, o **16.78 millones de colores** ($2^{24} = 16.777.216$)

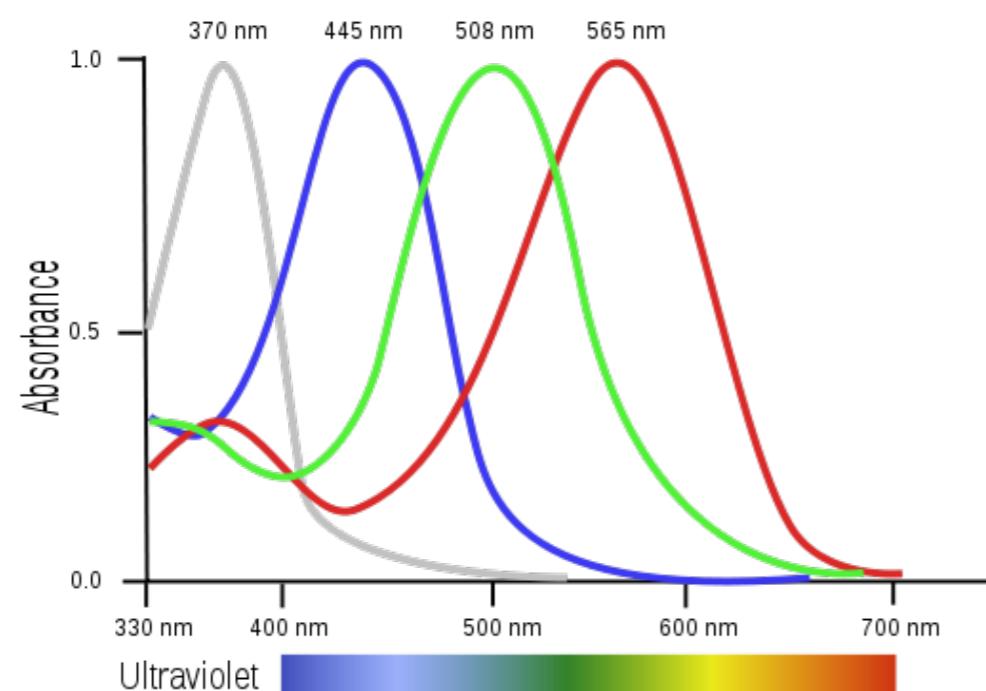


rojo
(255,0,0)

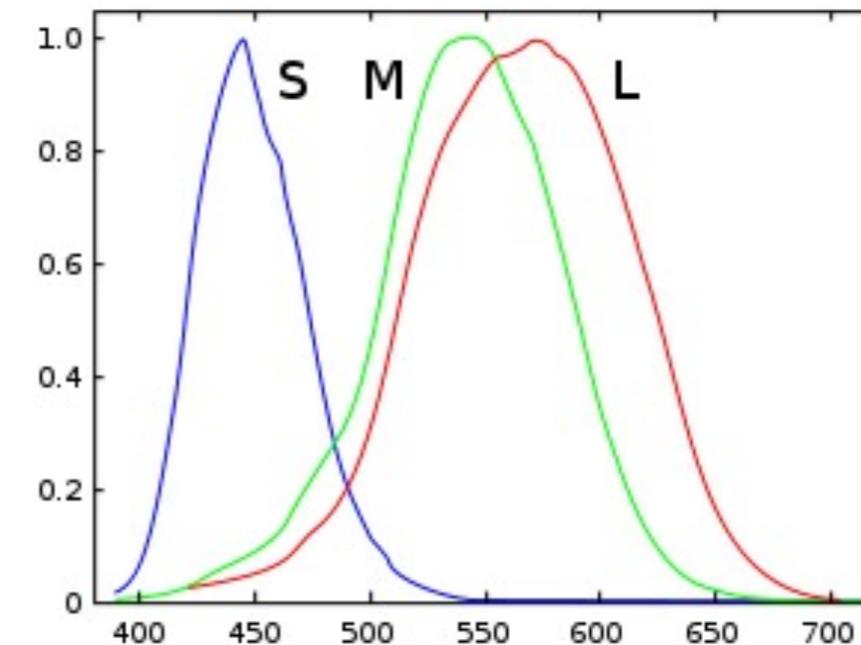
magenta
(255,0,255)

■ Tricromía y cuatricromía

- La discusión sobre cuantos colores vemos es amplia. En 1975 dos investigadores indicaron que el ser humano puede distinguir alrededor de 10 millones de colores (1)



Aves, insectos, arácnidos, anfibios reciben cuatro señales (cuatricromía), pudiendo ver la luz ultravioleta



Receptores de color del ojo humano corresponden a los colores RGB

- En el año 2001, Jameson *et al.* mostró que al menos un 50% de las mujeres y un 8% de los hombres reciben cuatro señales. Esto les permite diferenciar hasta 100 millones de colores (2).

- (1) Judd, Deane B.; Wyszecki, Günter (1975). Color in Business, Science and Industry.
- (2) Jameson, K. A., Highnote, S. M., & Wasserman, L. M. (2001). "Richer color experience in observers with multiple photopigment opsin genes" (PDF). Psychonomic Bulletin and Review 8 (2): 244–261

Ejemplo 1 | Lectura de una imagen

Lena Söderberg, 1972
Playboy Magazine



Lena



Canales
RGB



Transformación en
escala de grises

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

losing lena <https://www.nature.com/articles/s41565-018-0337-2>



fuente: <https://vimeo.com/372265771>

- Imágenes en escala de grises



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  
import matplotlib.pyplot as plt
```



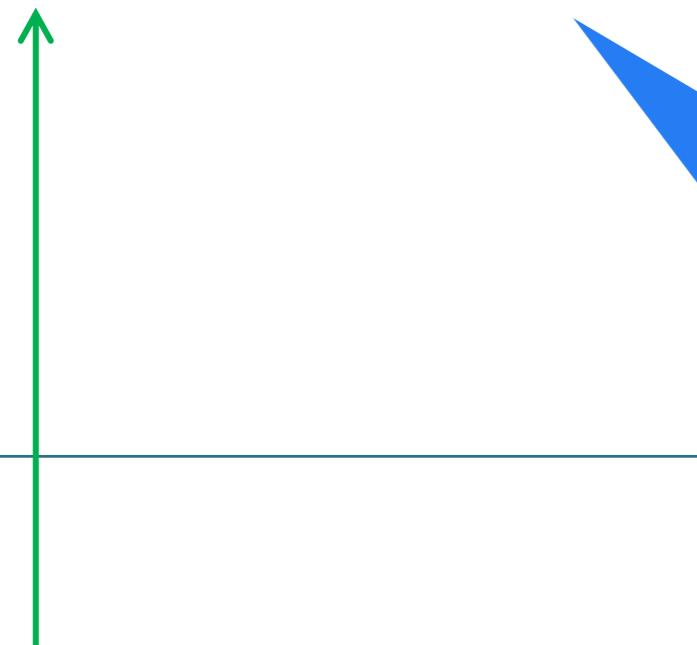
Módulo para
generar un
despliegue de
gráficos



Lectura de imagen en OpenCV

```
import cv2
import matplotlib.pyplot as plt

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



imread carga
una imagen en la
memoria



imagen original > *lena.jpg*



matriz: La imagen queda
almacenada en esta
variable



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
import matplotlib.pyplot as plt

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

plt.imshow(img)
```



imshow crea una ventana con la imagen contenida en ella

Ingresamos como parámetro la información de la imagen anteriormente leída

Lectura de imagen en OpenCV

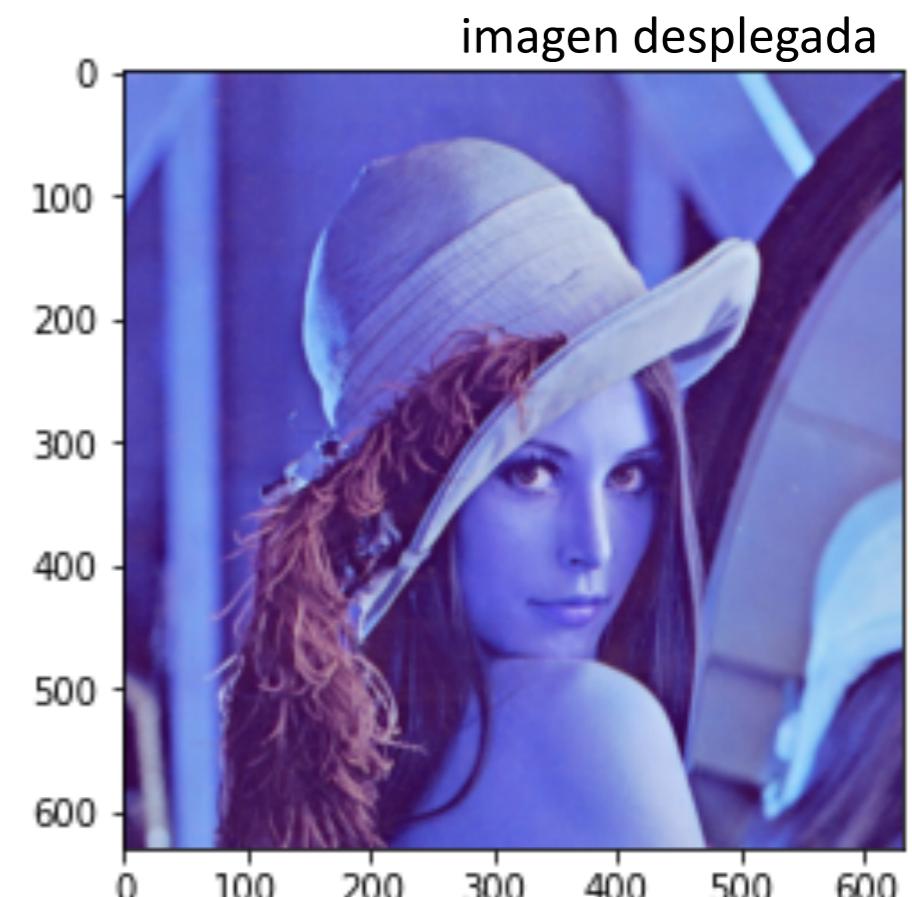
```
import cv2
import matplotlib.pyplot as plt

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

plt.imshow(img)

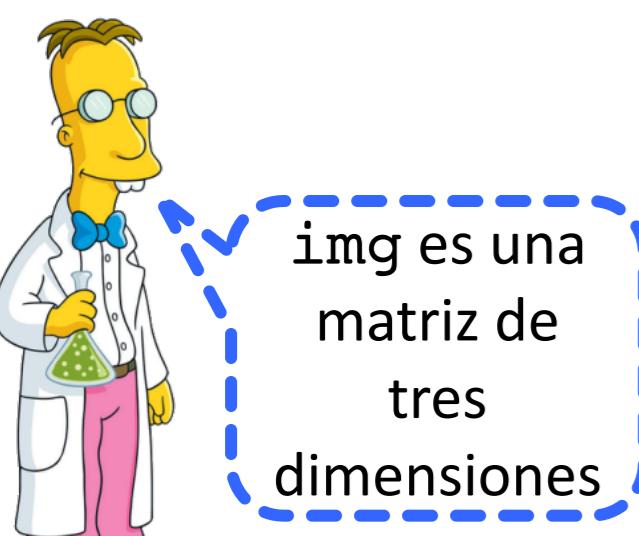
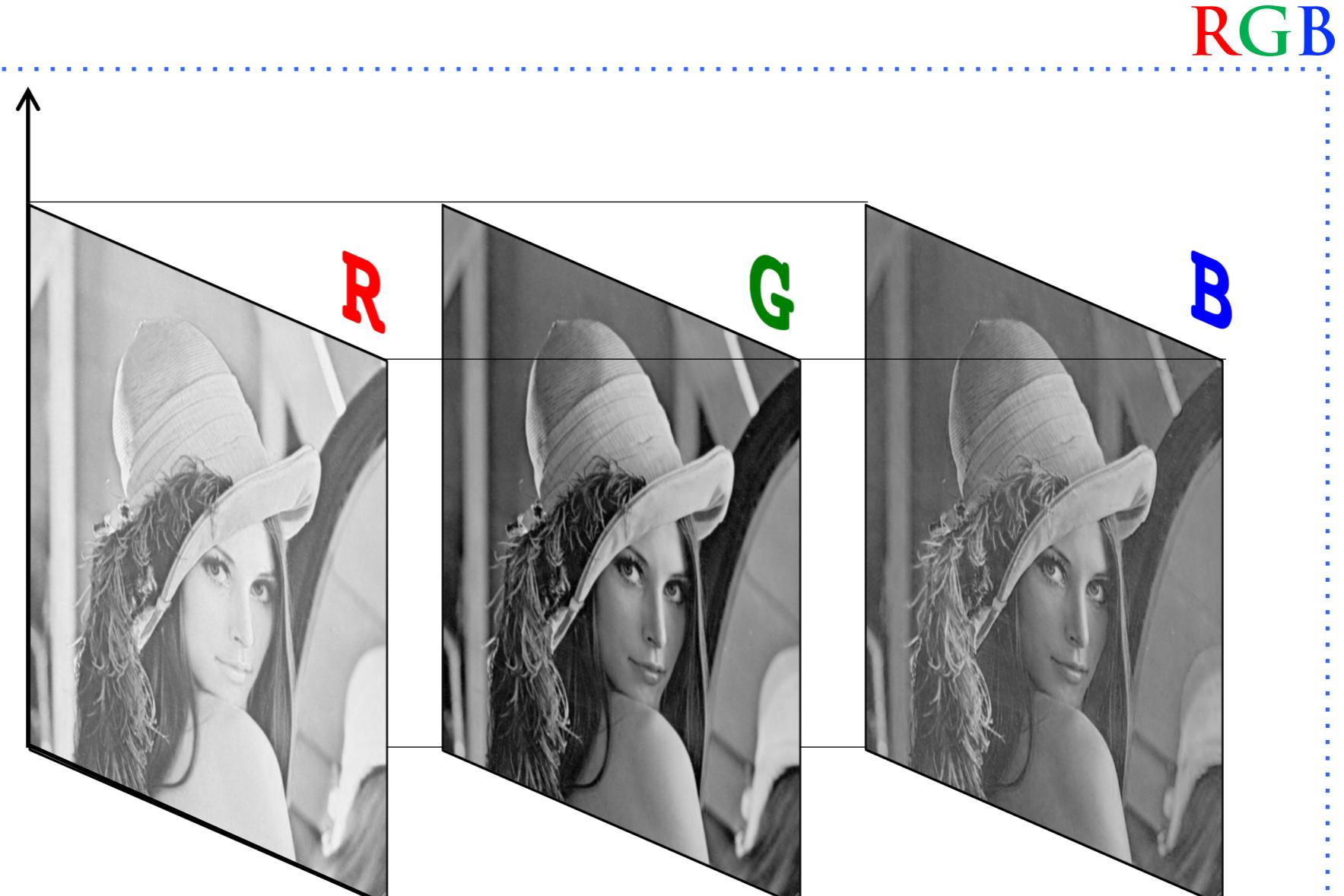
plt.show()
```

show() despliega el resultado en la pantalla según lo generado en los pasos anteriores



- Representación de cada canal RGB en la variable img

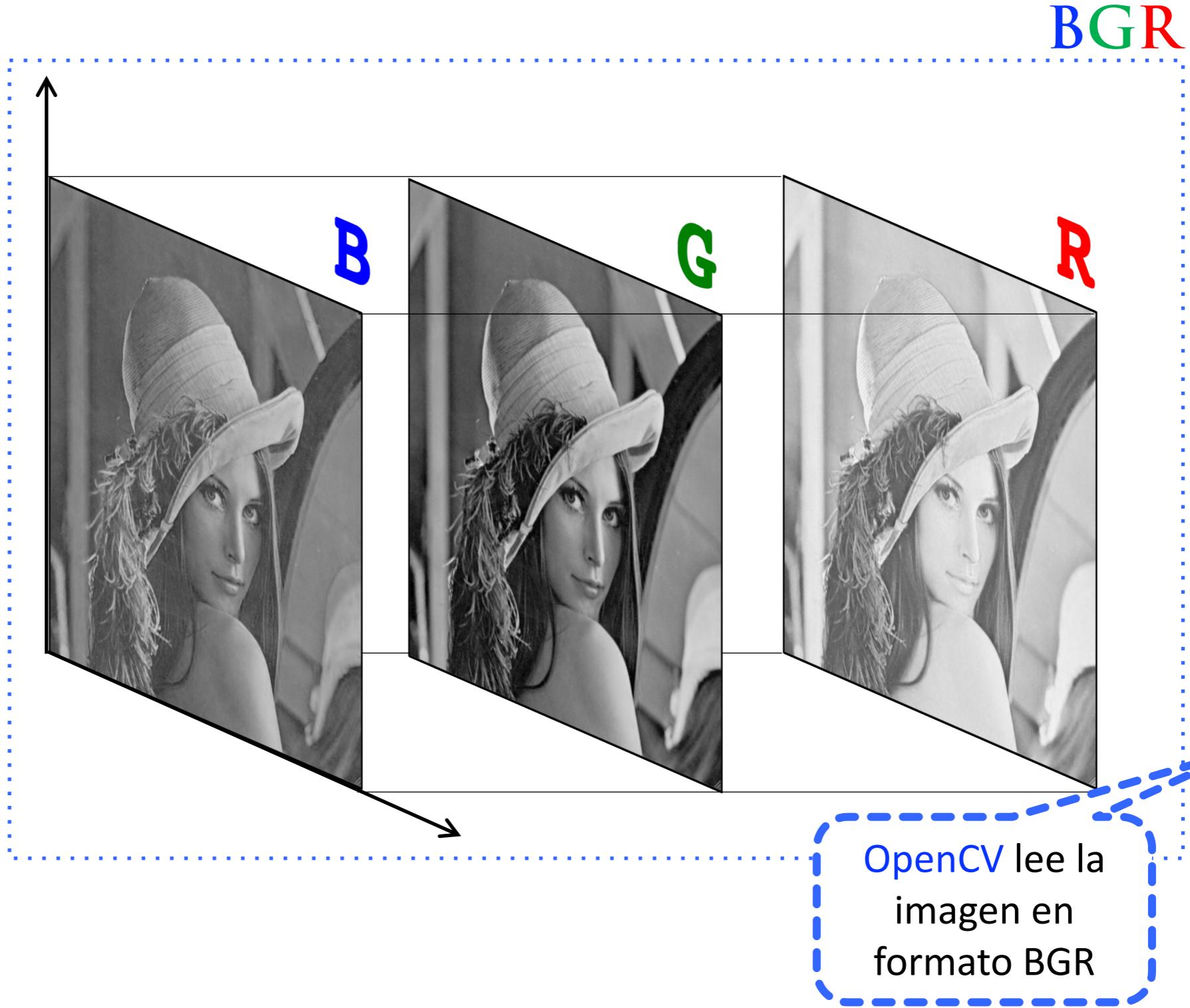
img =



img es una
matriz de
tres
dimensiones

- Representación de cada canal BGR en la variable img

img =



Lectura de una imagen en OpenCV

```
import cv2
import matplotlib.pyplot as plt

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

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

cvtColor()
modifica el espacio
de color de la
imagen original a
otro espacio



cv2.COLOR_BGR2RGB
cv2.COLOR_BGR2RGBA
cv2.COLOR_BGR2XYZ
cv2.COLOR_BGR2YCR_CB
cv2.COLOR_BGR2YCrCb
cv2.COLOR_BGR2YUV
cv2.COLOR_BGR2YUV_I420
cv2.COLOR_BGR2YUV_IYUV
cv2.COLOR_BGR2YUV_YV12
cv2.COLOR_BGR5552BGR
cv2.COLOR_BGR5552BGRA
cv2.COLOR_BGR5552GRAY



OpenCV ofrece múltiples transformaciones a otros espacios de color

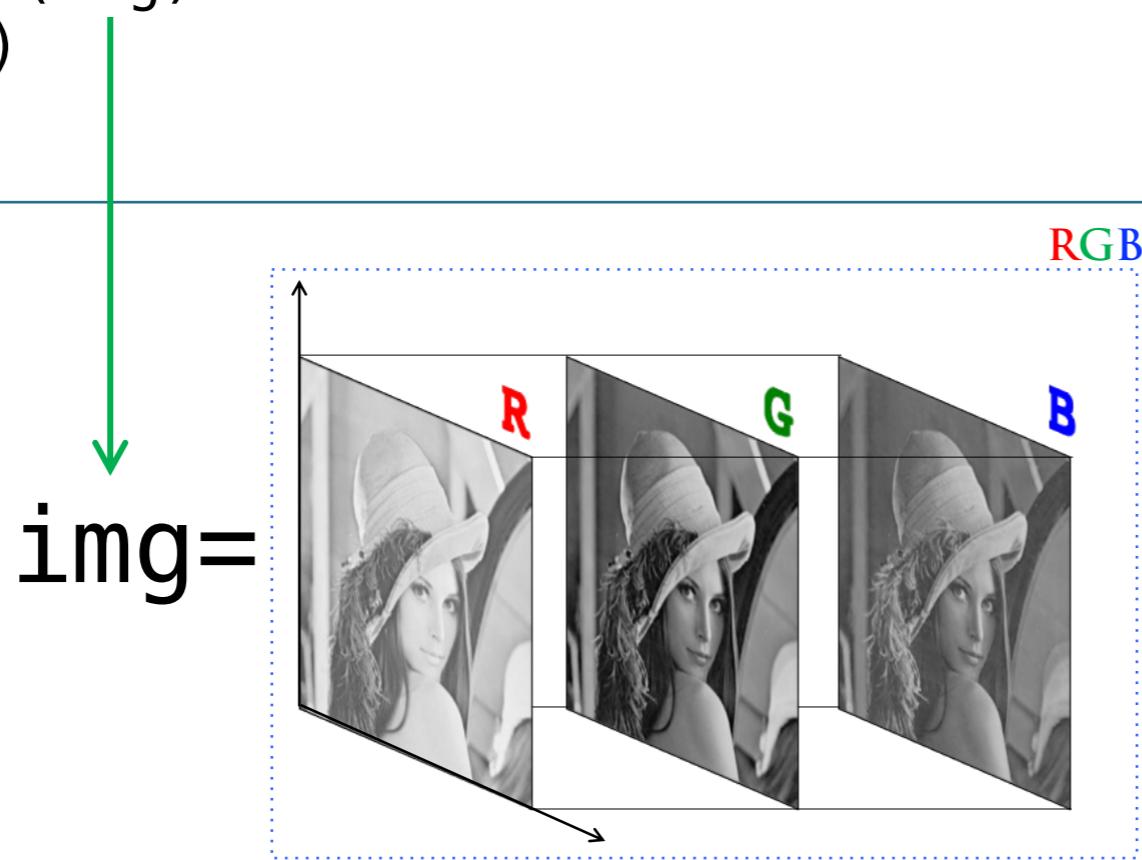
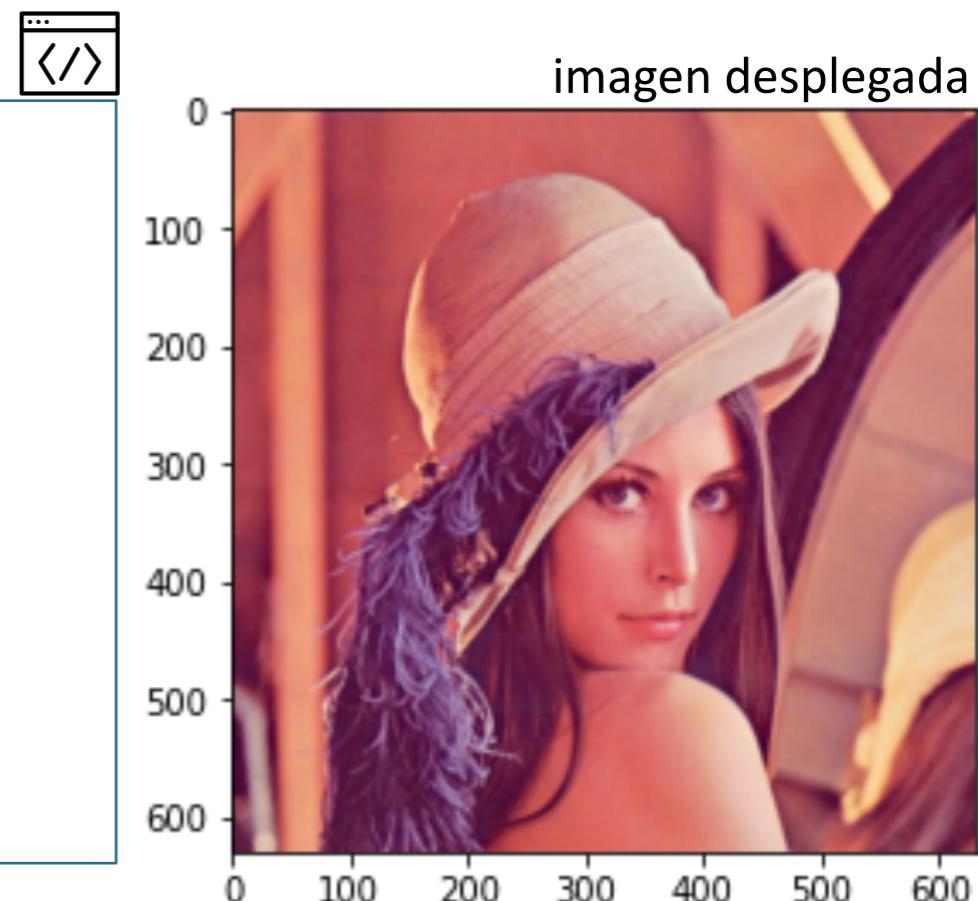
- Lectura de una imagen en OpenCV

```
import cv2
import matplotlib.pyplot as plt

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

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img)
plt.show()
```



Recuerde, una imagen a color está compuesta por 3 o más matrices



- 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
import matplotlib.pyplot as plt

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

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

al separar los canales de la imagen, podemos asignarlos a variables independientes

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

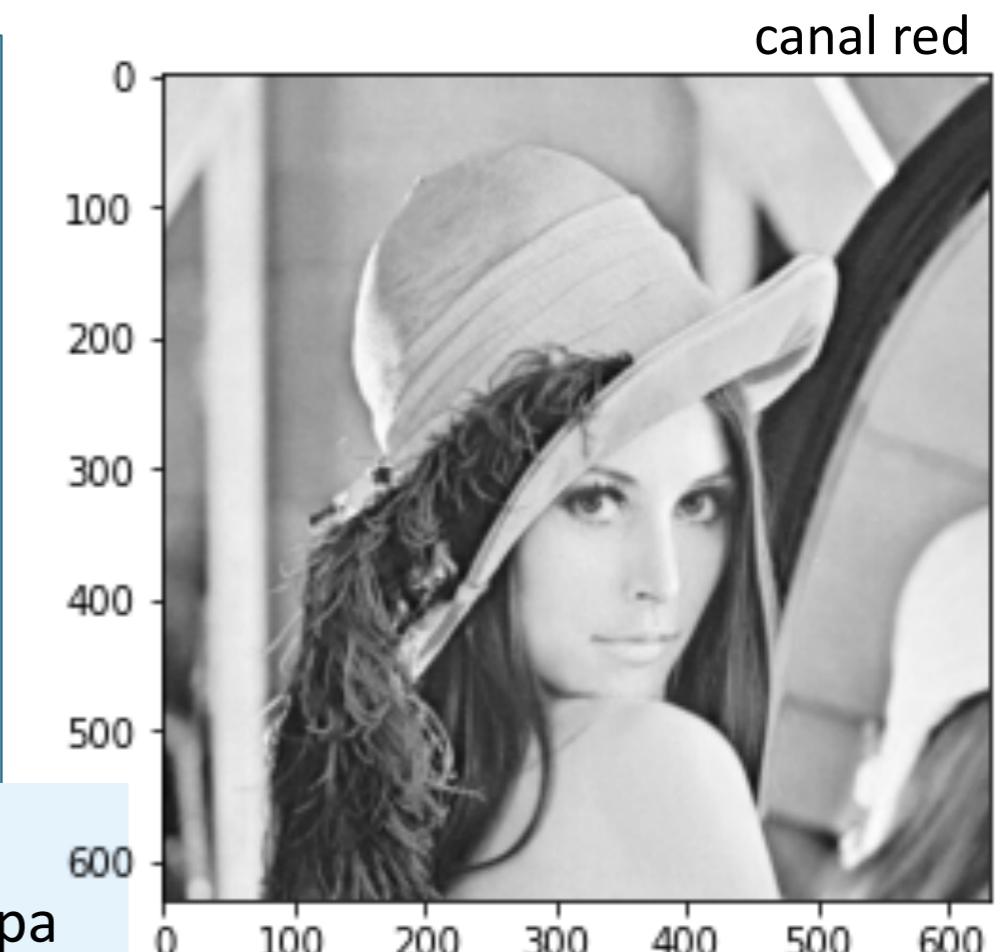
■ Separando un canal

```
import cv2
import matplotlib.pyplot as plt

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

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

plt.imshow(red, cmap="gray")
plt.show()
```



es posible
cambiar el mapa
de color a través
de la opción cmap



Ejercicio

Muestra una imagen de cada canal

- Analicemos la distribución de grises de un canal



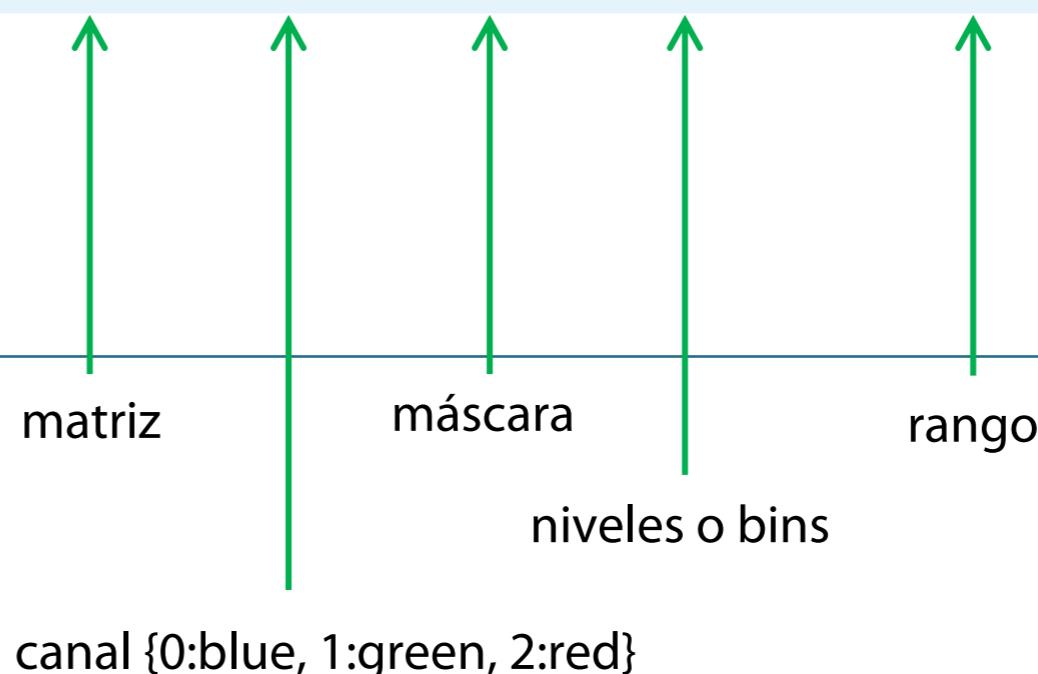
```
import cv2
import matplotlib.pyplot as plt

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

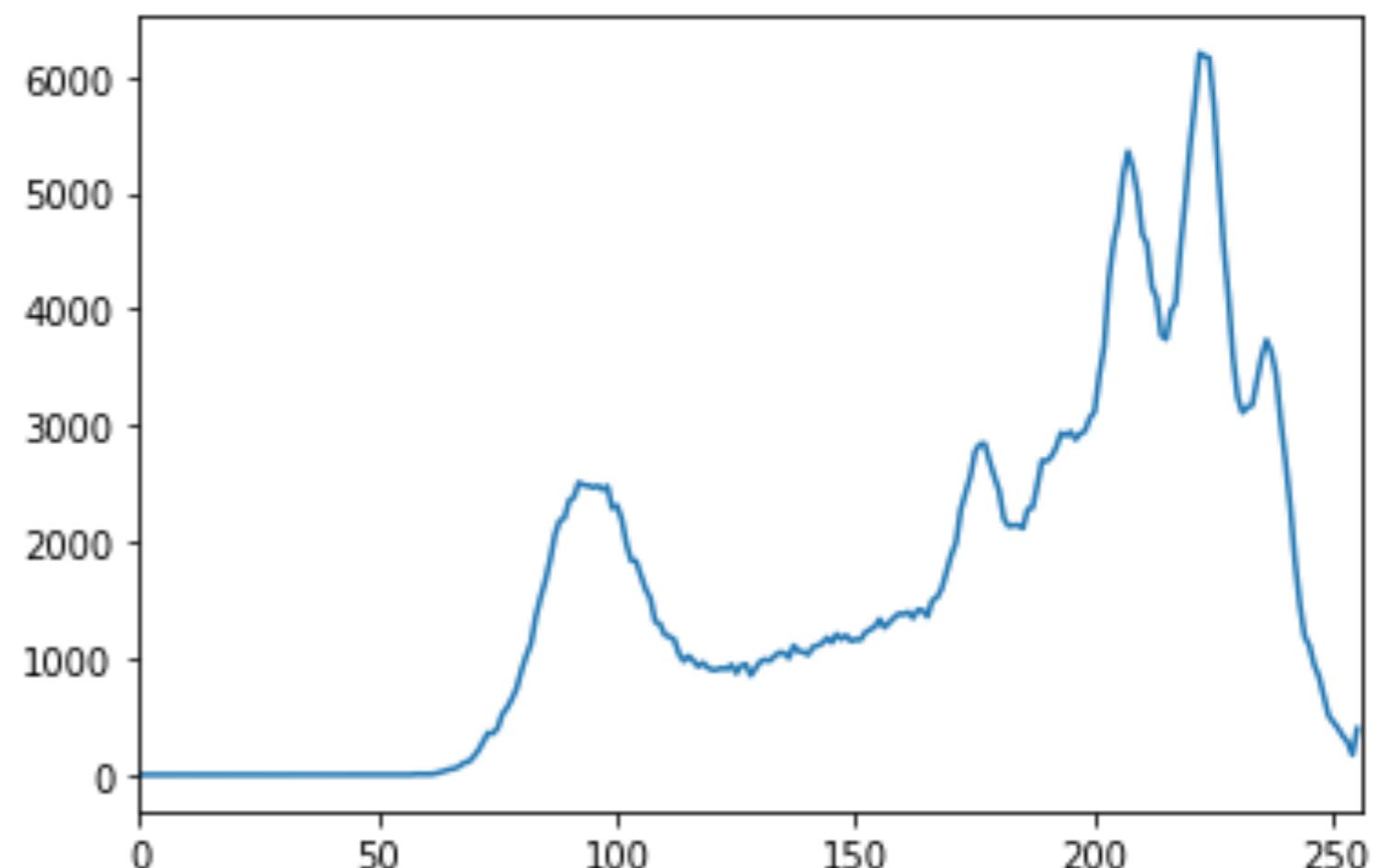
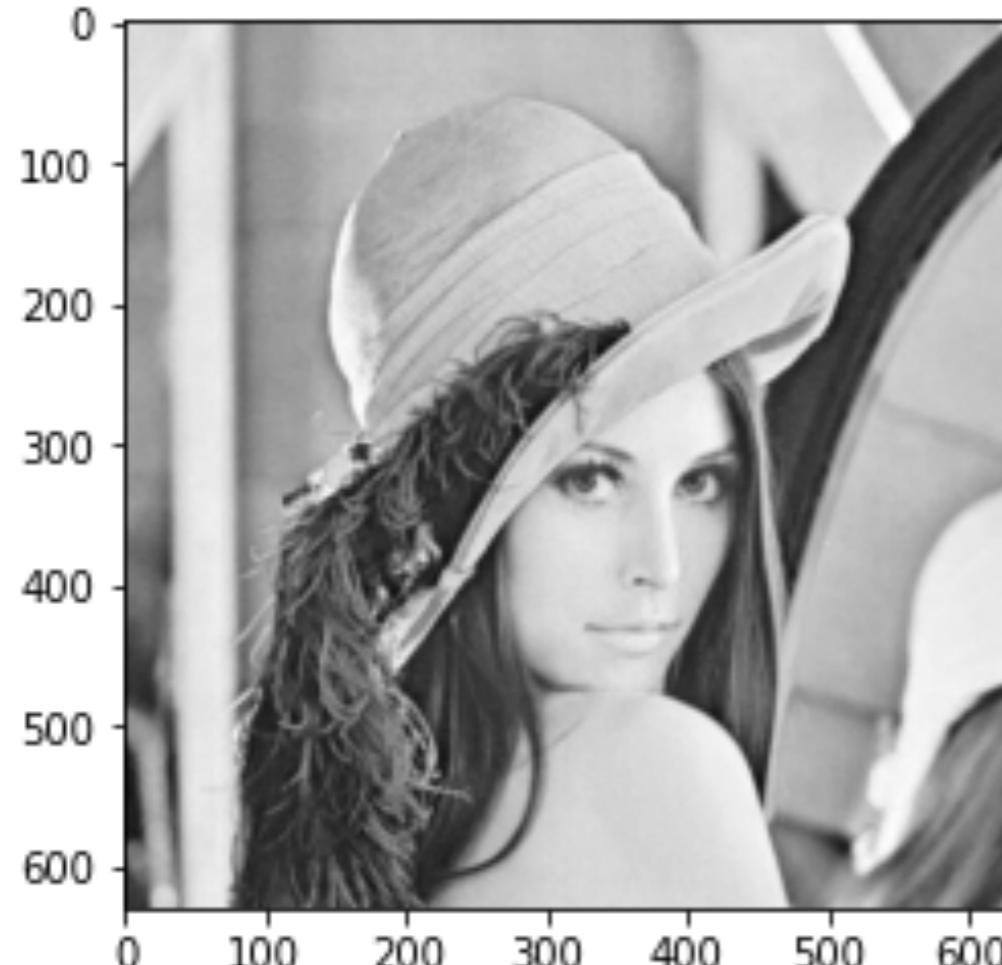
```
hist_red = cv2.calcHist([img], [2], None, [256], [0,256])
```

```
plt.plot(hist_red)
plt.xlim([0,256])
plt.show()
```

La función calcHist nos permite calcular el histograma

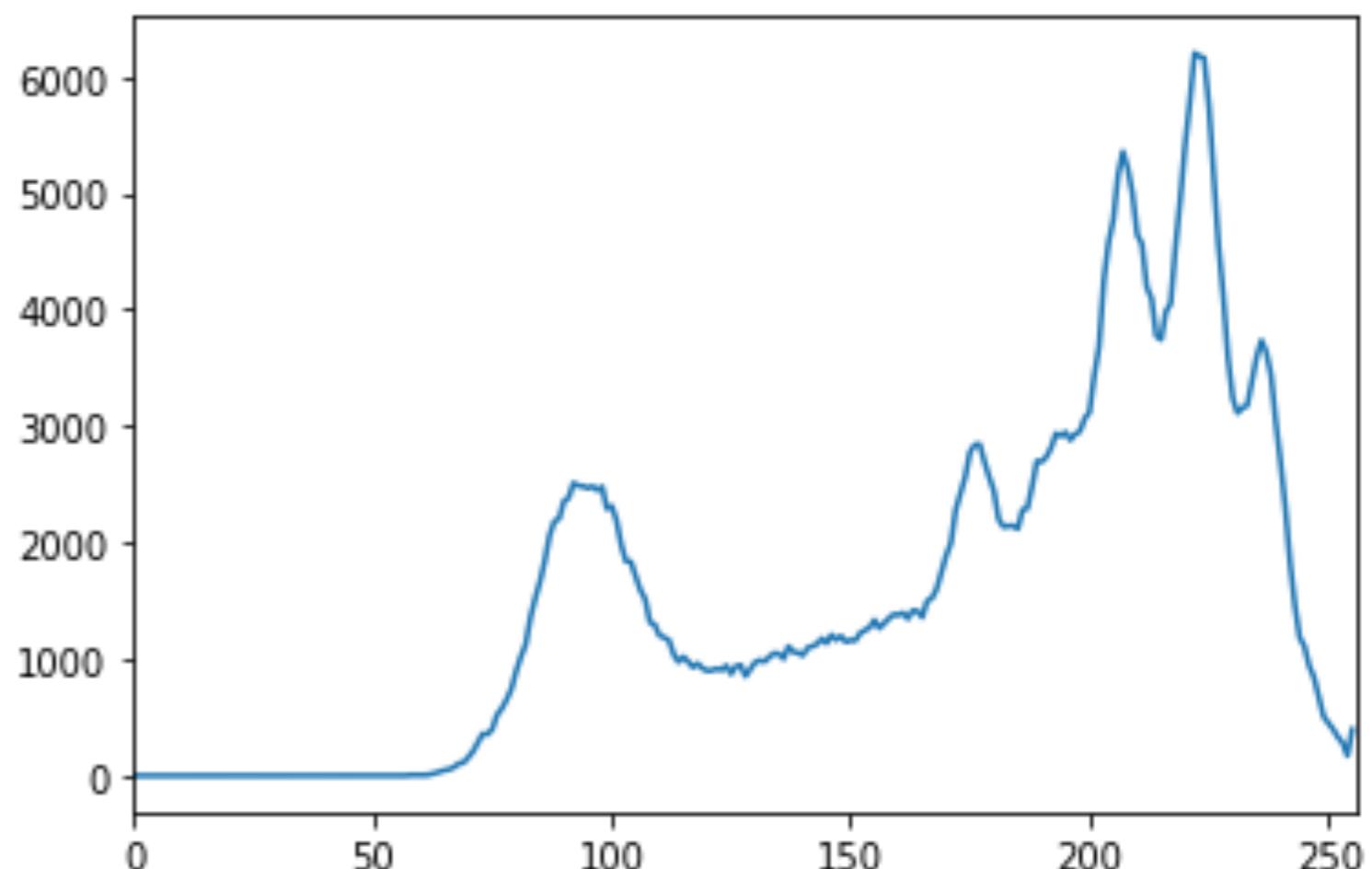
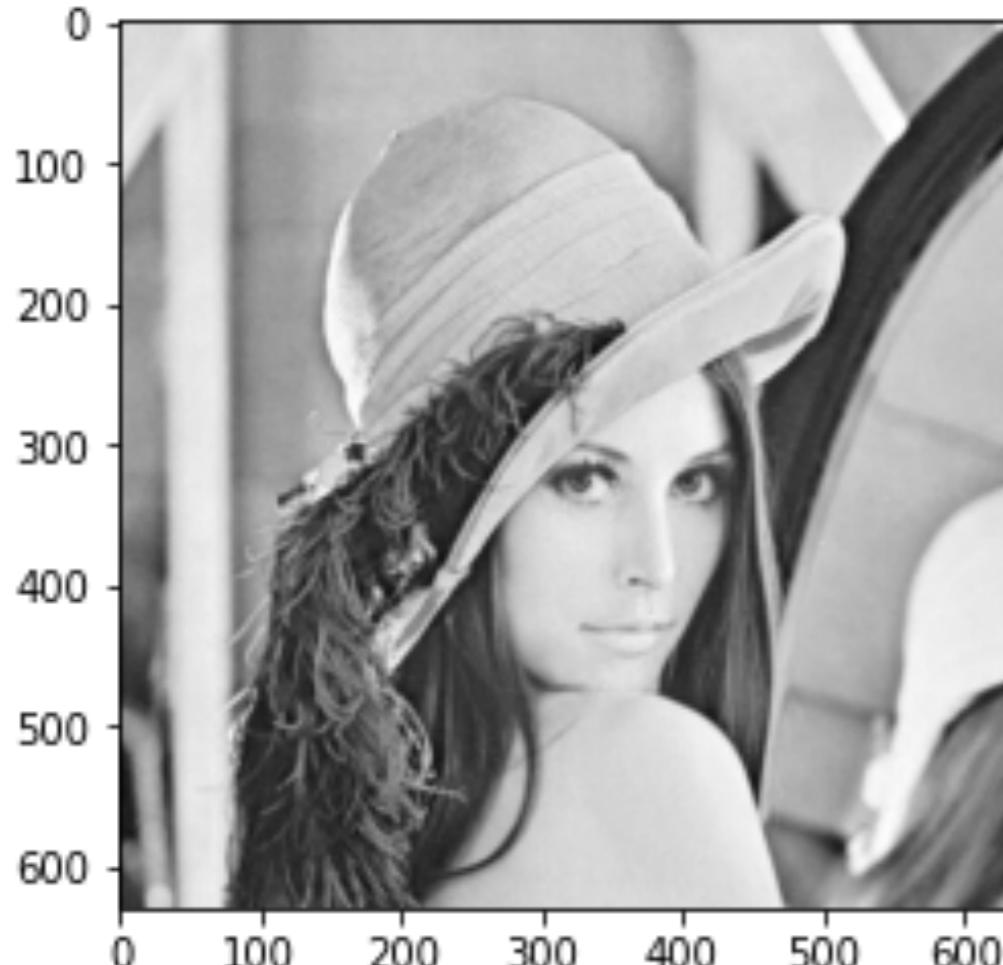


- ▶ Analicemos la distribución de grises de un canal



¿Qué representa esta curva del canal Rojo?,
¿Será diferente para otros canales?

- ▶ Analicemos la distribución de grises de un canal



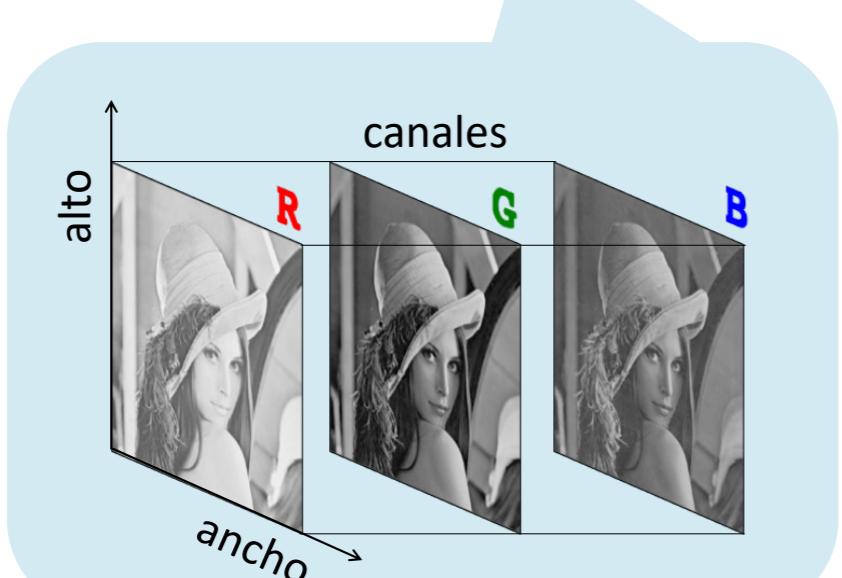
Ejercicio

Grafique para cada uno de los canales 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
```



el atributo shape nos permite determinar las dimensiones de la imagen



■ Posiciones de los píxeles

```
import cv2

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

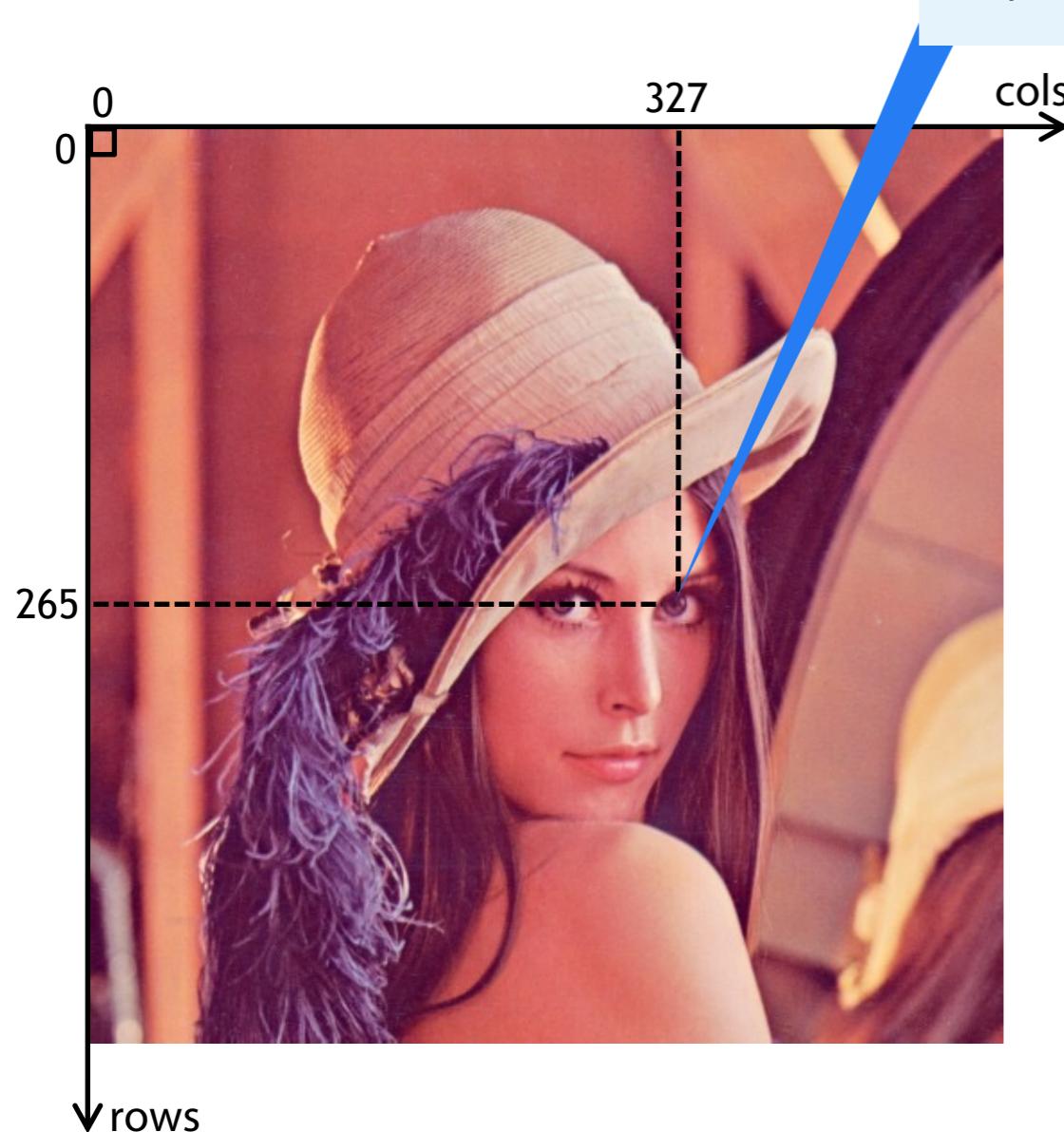
dimensions = img.shape

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)
```



- Canales de una imagen



los valores RGB de dicho
pixel son [56 20 80]

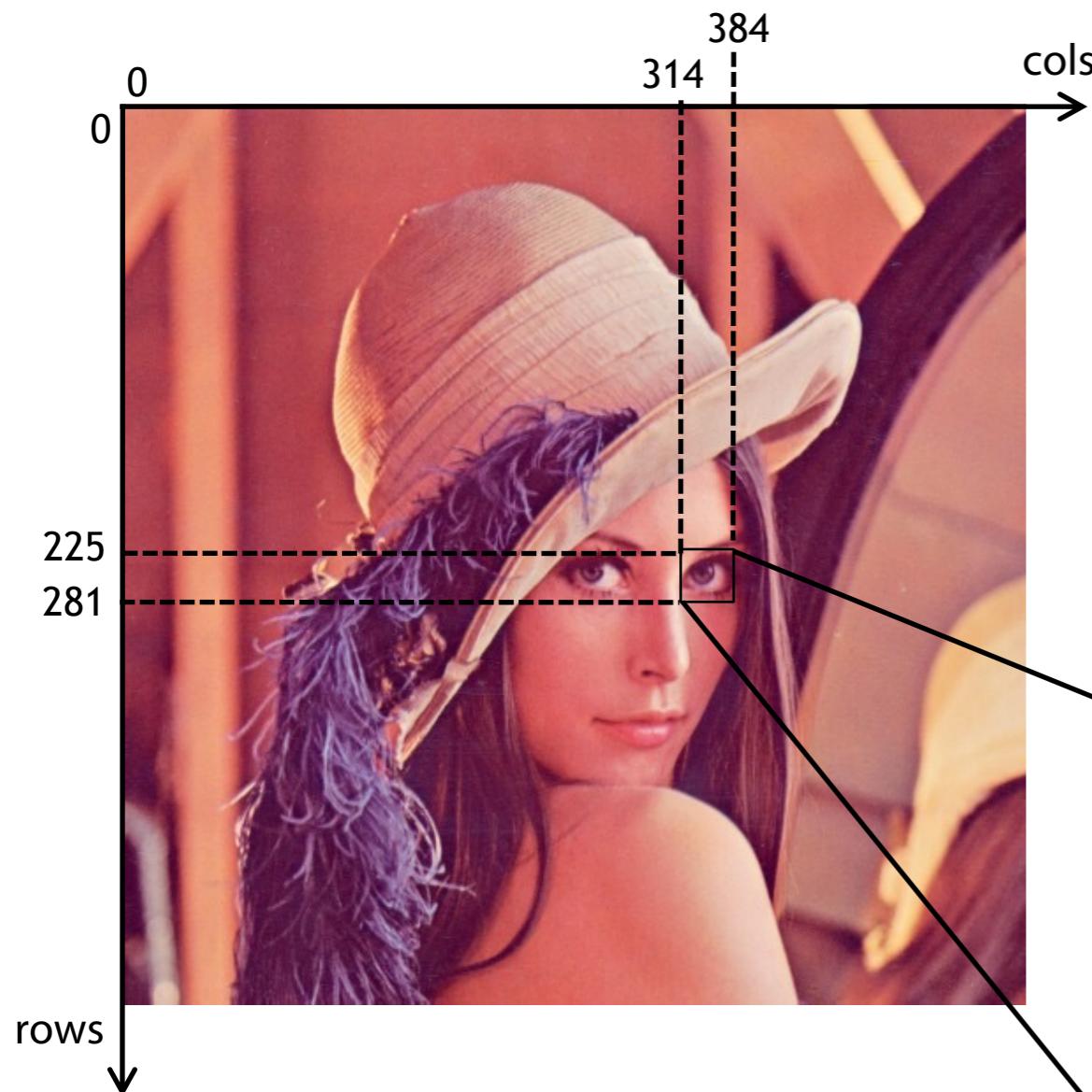
```
import cv2  
  
img = cv2.imread('lena.png')  
  
pixel = img[265,327,:]  
  
print(pixel)
```

el operador :
significa todas las
dimensiones

output
[67 16 88]

Canal B Canal G Canal R

- Sección de una imagen



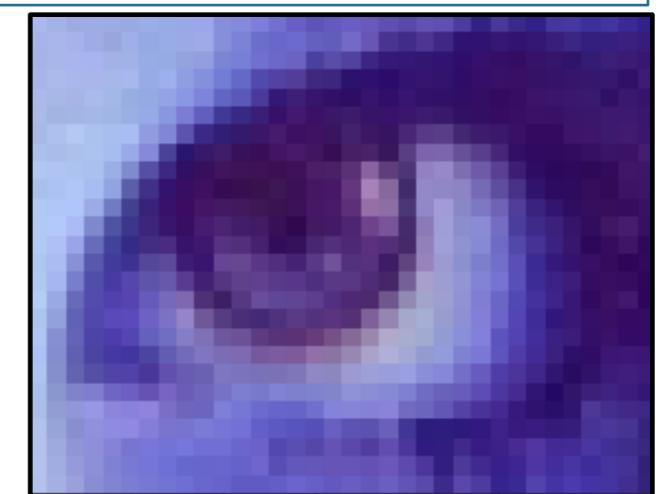
```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('lena.png')

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

plt.imshow(roi)
plt.show()
```



ROI (canal R)



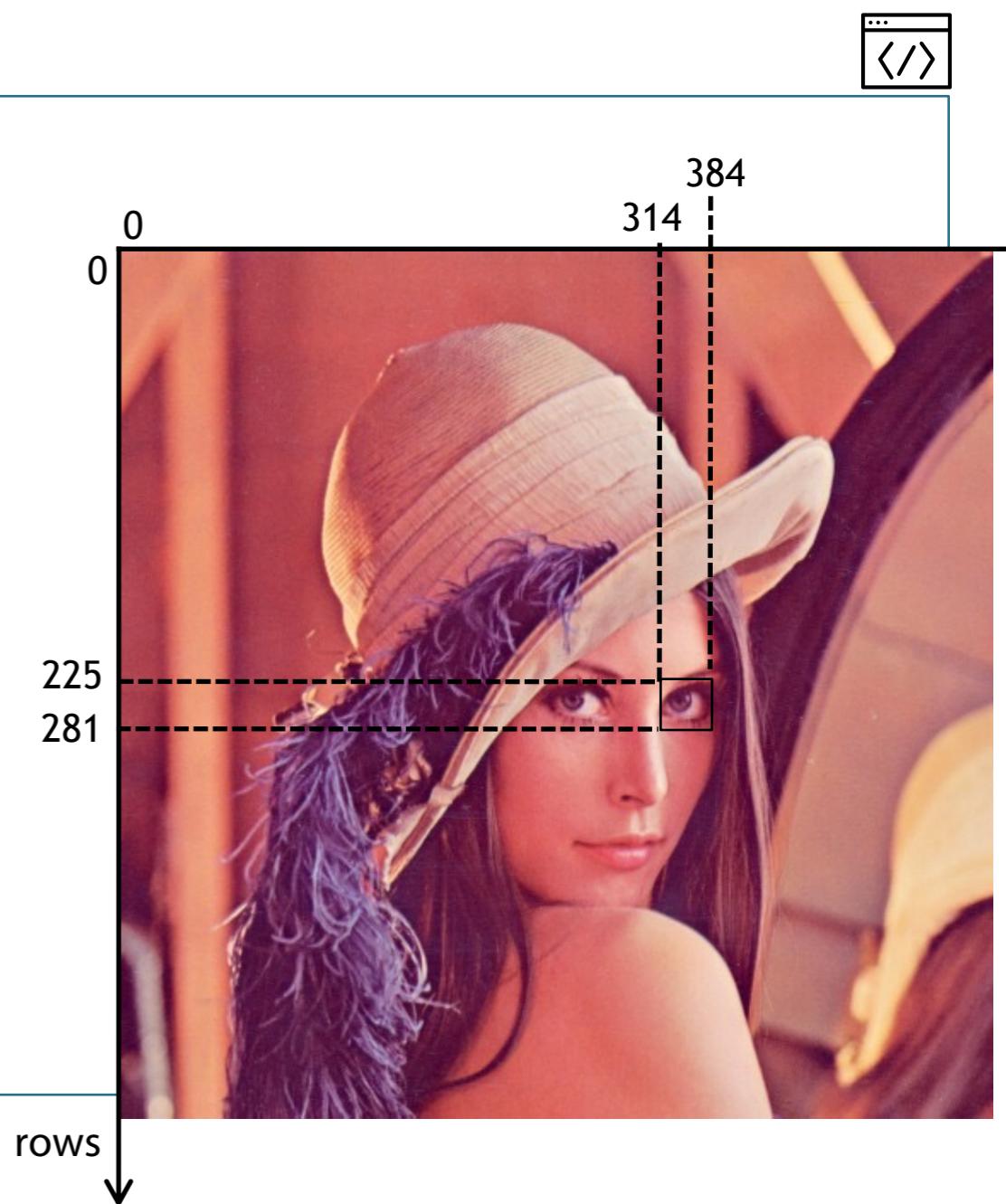
ROI
(canal BGR)

- Resize de una imagen, o subimagen

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('lena.png')
roi = img[255:281,314:348,[2,1,0]]
```

Podemos reordenar
el orden de los
canales por esta vía



- Resize de una imagen, o subimagen

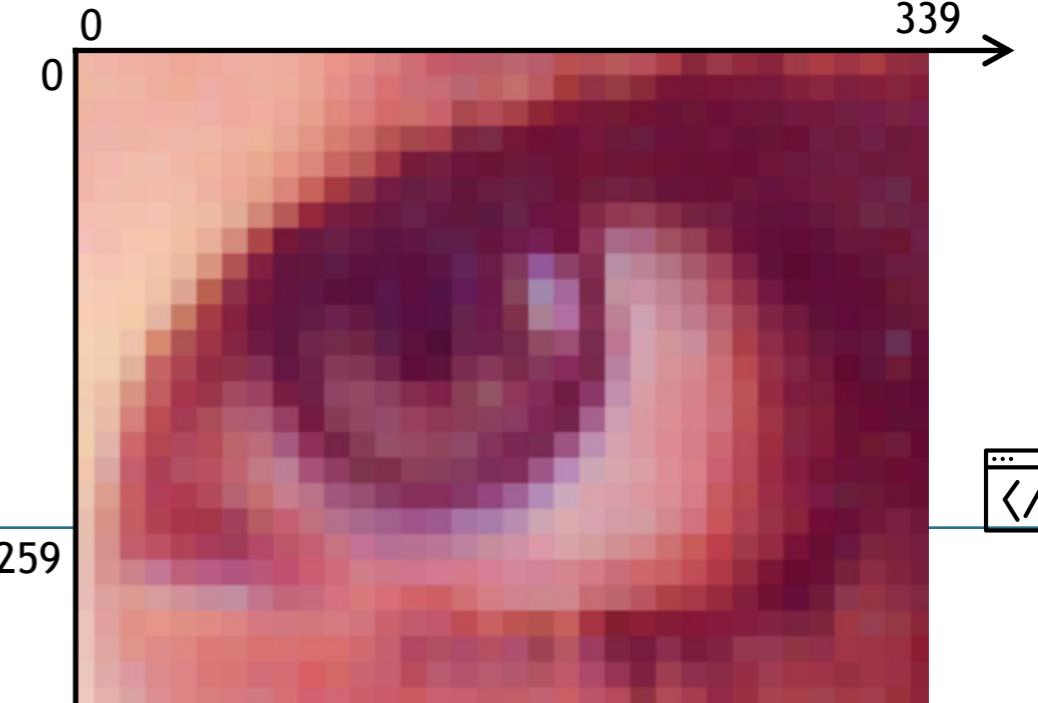


```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('lena.png')
roi = img[255:281,314:348,[2,1,0]]

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

generemos una
nueva dimensión
como una escala del
tamaño original



- Resize de una imagen, o subimagen

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('lena.png')
roi = img[255:281,314:348,[2,1,0]]

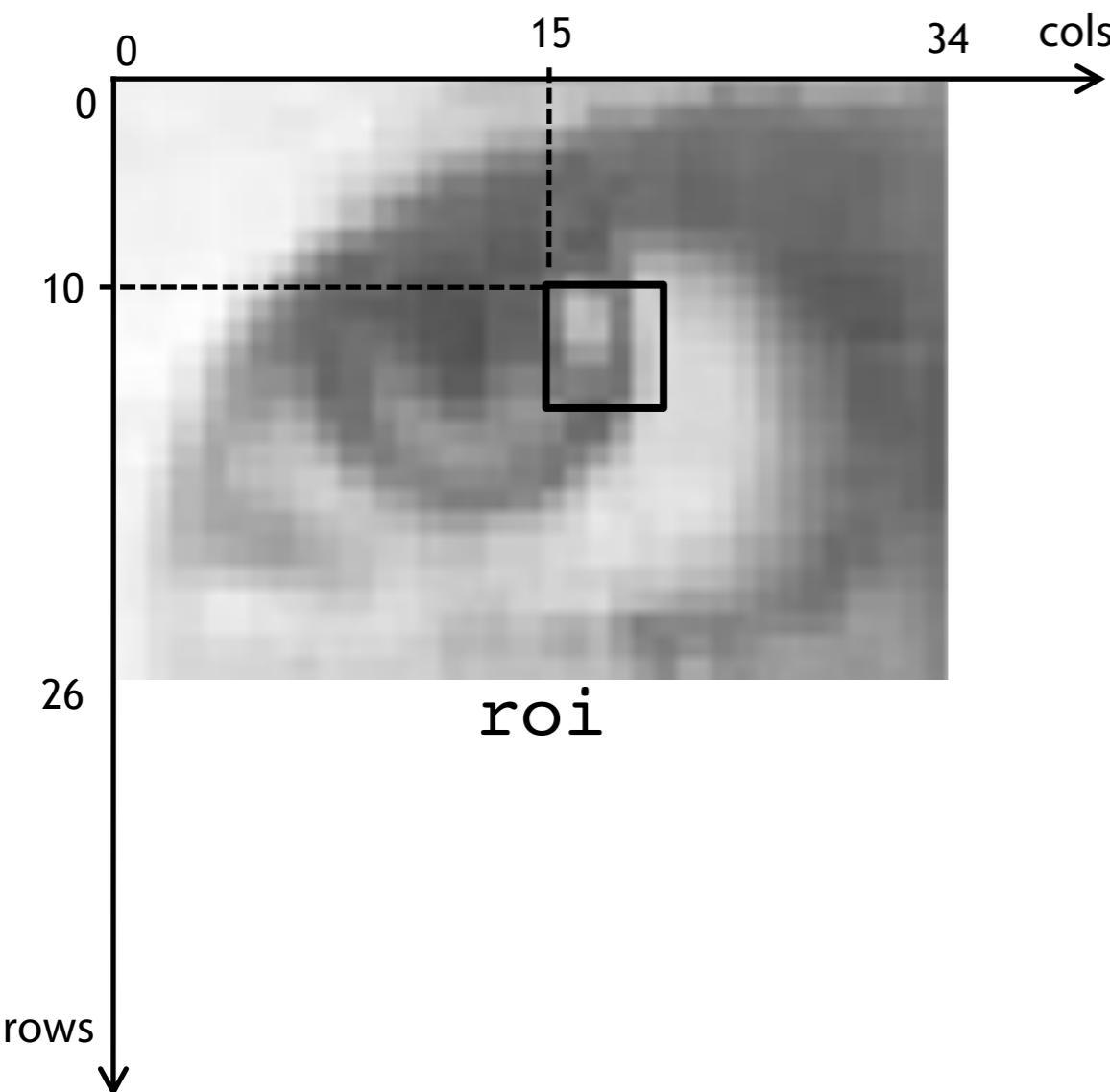
factor_times = 10
height = int(roi.shape[0] * factor_times)
width = int(roi.shape[1] * factor_times)
dim = (width, height)

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

plt.imshow(resized)
plt.show()
```

el comando `resize()` modifica el tamaño de una imagen

- Sección de una imagen



```
import cv2
import matplotlib.pyplot as plt

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

roi = img[255:281,314:348,2]

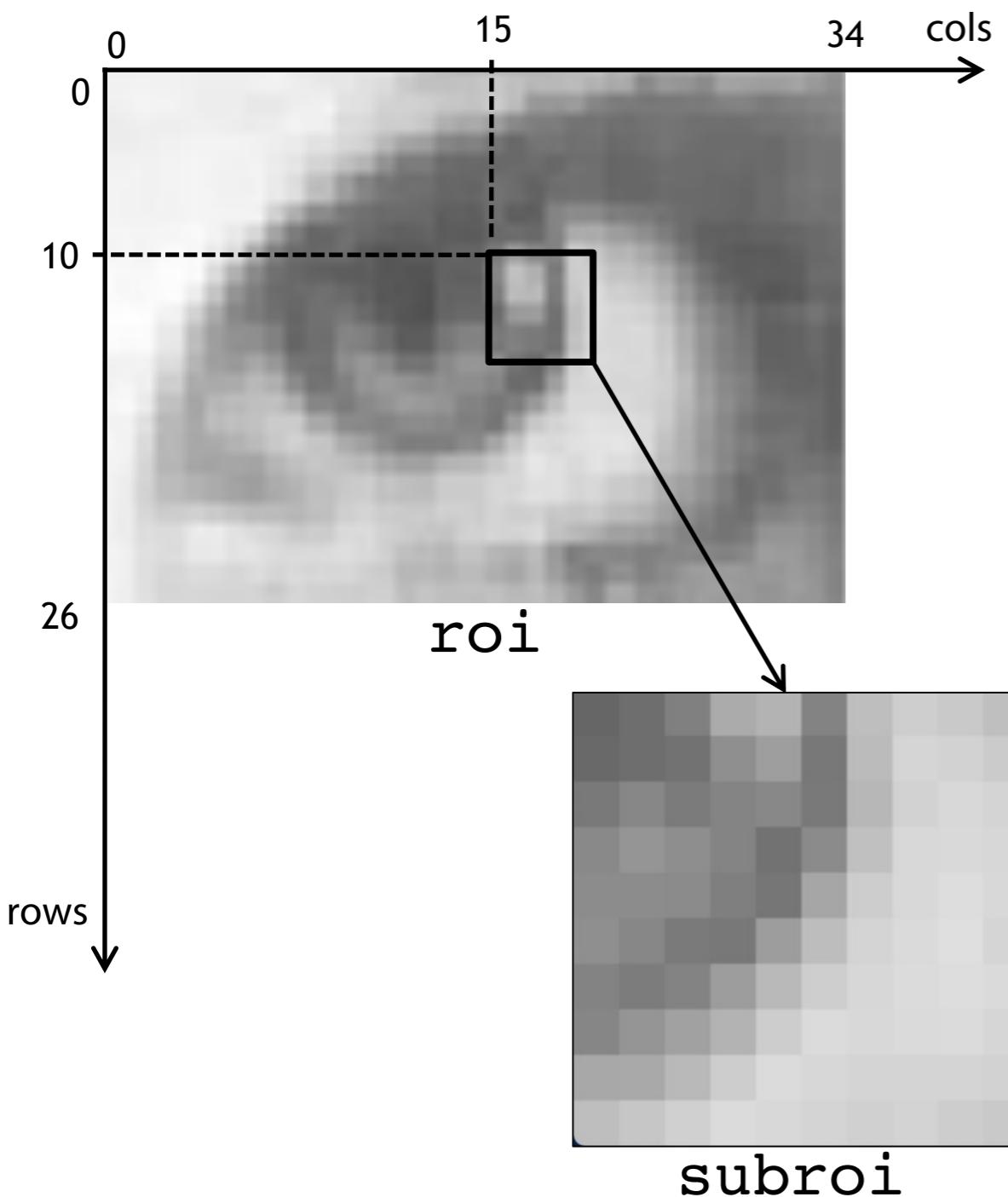
subroi = roi[10:20, 15:25]
```

INICIO FIN INICIO

Seleccionamos el canal a R

The code shows how to extract a region of interest (ROI) from a grayscale image named 'lena.png'. It then extracts a subregion from the ROI. The subregion is highlighted with a blue rounded rectangle. Red arrows point from the text labels 'INICIO' and 'FIN' to the start and end indices in the subroi assignment. A green arrow points from the text 'Seleccionamos el canal a R' to the index '2' in the line 'roi = img[255:281,314:348,2]'. The code uses NumPy-style indexing to select the second channel (blue).

- Sección de una imagen



```
import cv2
import matplotlib.pyplot as plt

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

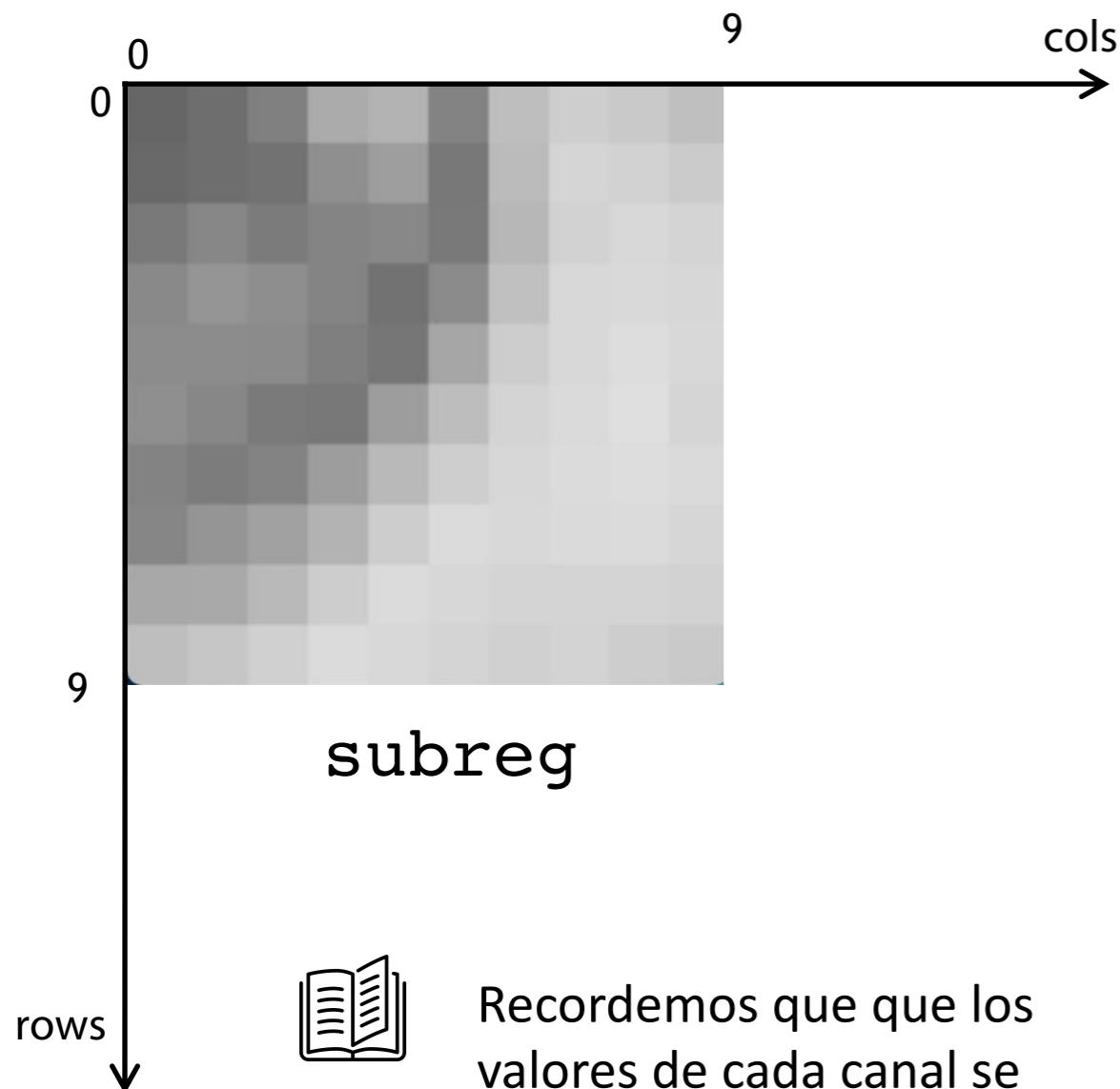
roi = img[255:281,314:348,2]

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

plt.imshow(subroi, cmap="gray")
plt.show()
```

</>

- Valores en escala de grises



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
import matplotlib.pyplot as plt

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

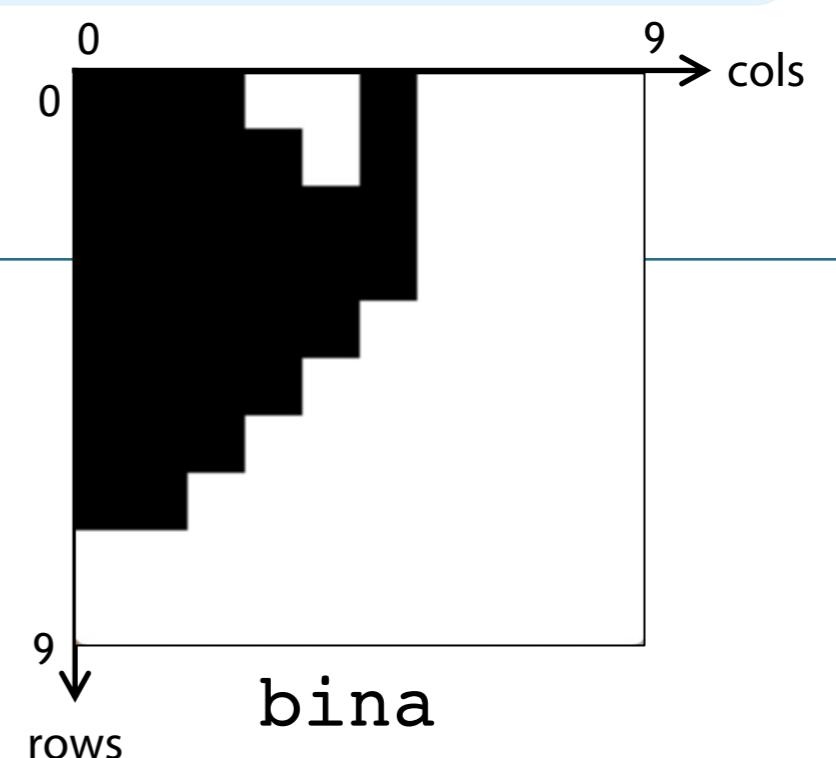
```
import cv2
import matplotlib.pyplot as plt

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)

plt.imshow(bina, cmap="gray")
plt.show()
```

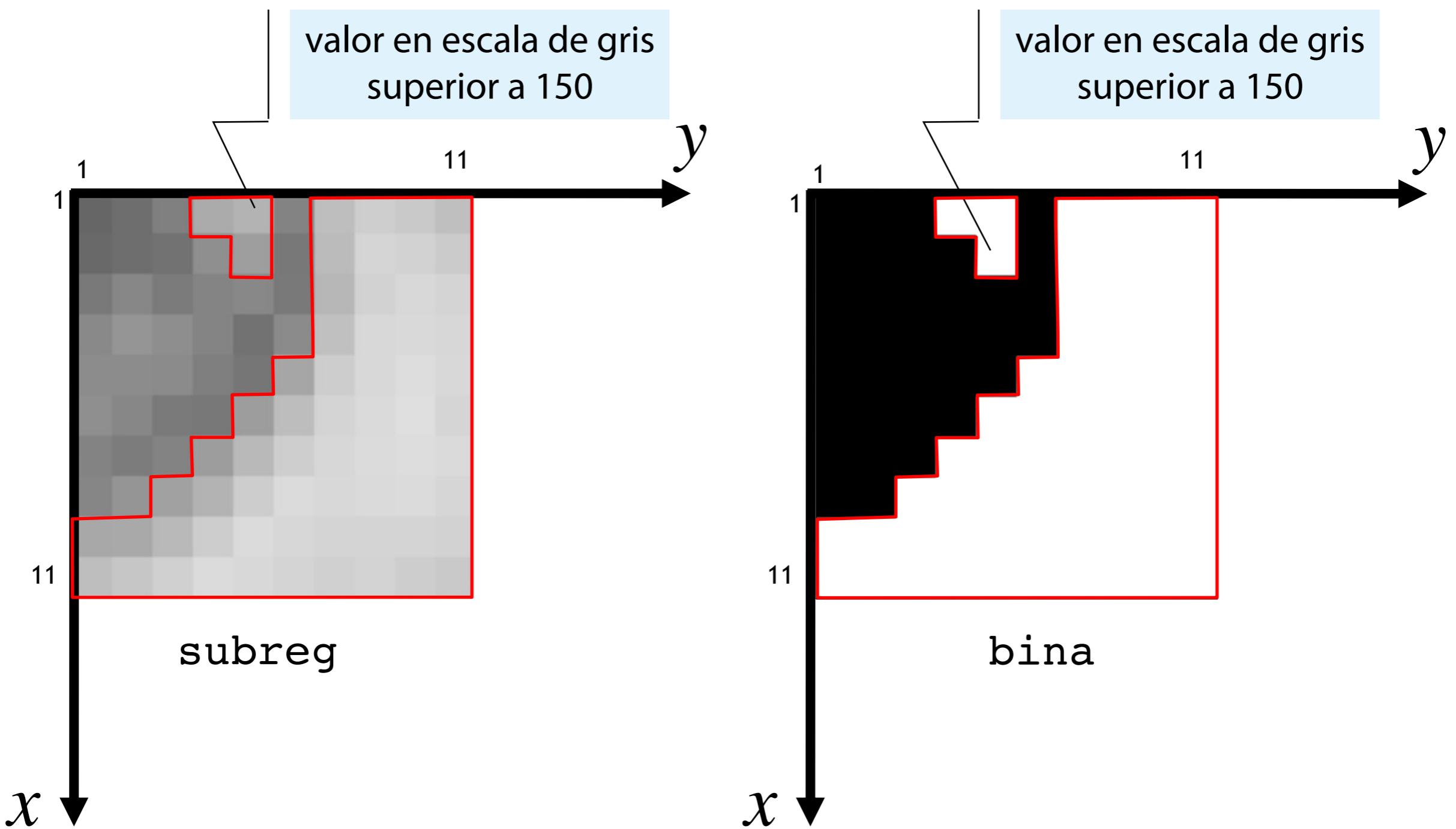
La función threshold permite binarizar una imagen. Más adelante veremos más opciones de esta función



Ejercicio

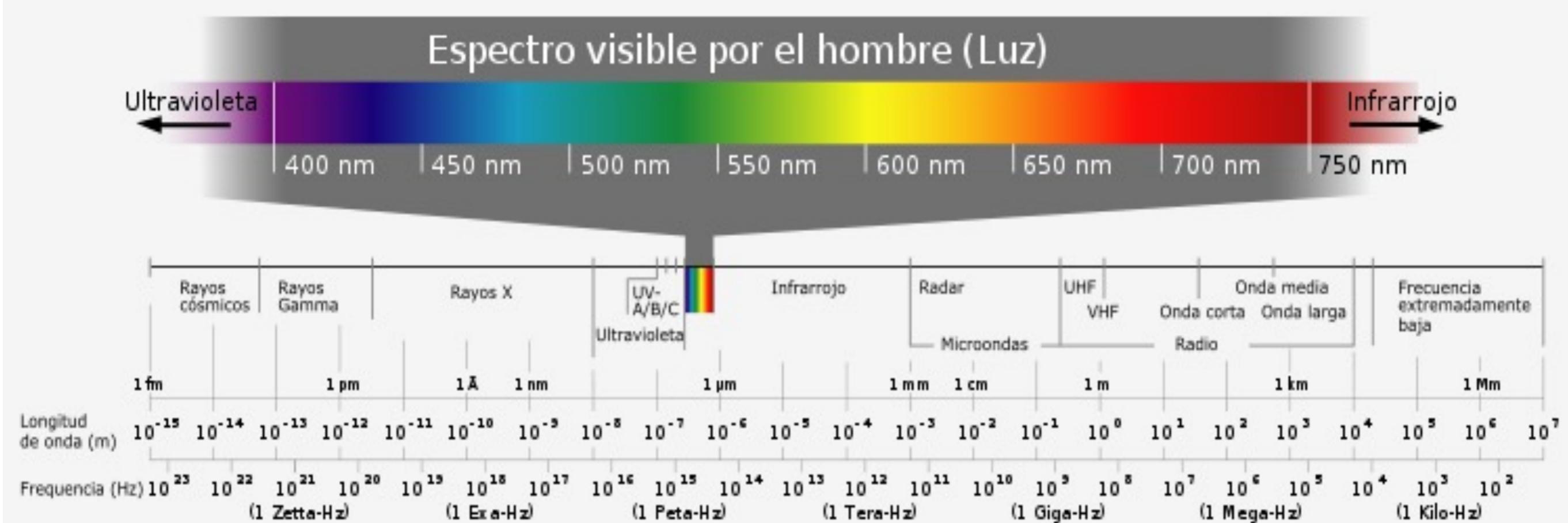
Binarice el ojo derecho. Seleccione el umbral que usted desee

- Comparando dos imágenes



- OpenCV
- Modelos del color

- La luz visible es una radiación electromagnética que es percibida por el ojo humano.



■ Definición

- Flujo radiante: es la medida de la potencia de radiación electromagnética



➤ 1 ampolleta de 100 watt ilumina 1000 lumen.

- Es la energía medida por unidad de tiempo.
- Su unidad internacional es el *Watt*.

$$W = V \cdot A = \frac{J}{S}$$

(joule) (seg)

potencia voltaje corriente

■ Definición

- Flujo luminoso: es la medida de la potencia luminosa percibida por el ojo



➤ 1 Lux equivale a la luz de la Luna llena a gran altitud en latitud tropical

- Su unidad internacional es el *Lumen (lm)*.

$$lm = lx \cdot m^2 = cd \cdot sr$$

lumen lux metro candela estereoradián

A diagram showing the definition of a lumen. It consists of a central equation $lm = lx \cdot m^2 = cd \cdot sr$ enclosed in a dashed circle. Arrows point from the words "lumen", "lux", "metro", "candela", and "estereoradián" to their respective variables in the equation: lx, sr, m², cd, and sr.

➤ 1 ampolleta de 100 watt ilumina 1000 lúmenes.

Definición

- Luminancia: es la densidad angular y superficial de flujo luminoso que incide, atraviesa o emerge de una superficie siguiendo una dirección determinada.



Su unidad internacional es el L_v .

$$L_v = \frac{cd}{m^2} = \frac{d^2 F}{dS \cdot d\Omega \cdot \cos\theta}$$

➤ 1 candela equivale a la sexagésima parte de la luz emitida por un centímetro cuadrado de platino puro en estado sólido a la temperatura de su punto de fusión (2046 K).

CPGM, 1954

F : Flujo luminoso
 S : Elemento de superficie
 Ω : Elemento de ángulo sólido
 Θ : Ángulo entre la normal de la superficie y la dirección considerada

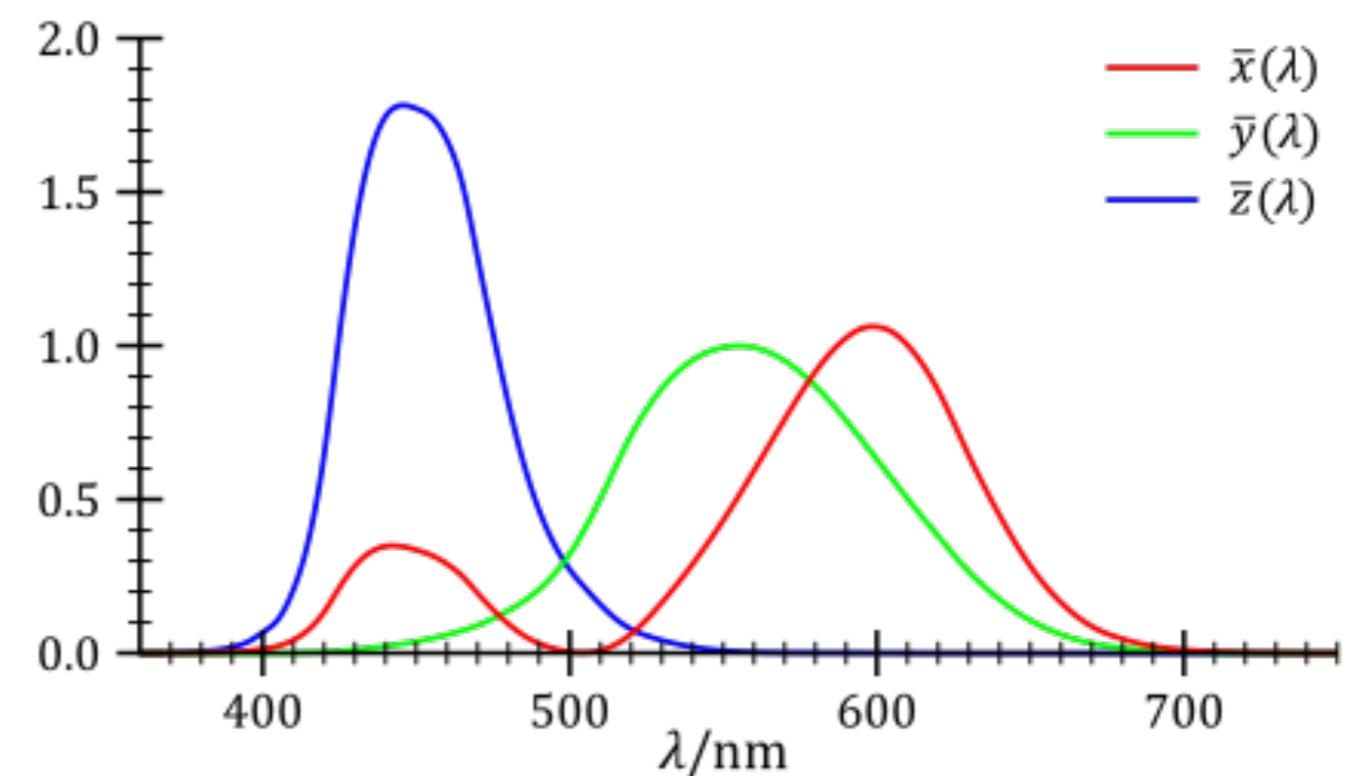
■ CIE XYZ

- El primer sistema de color formal se conoce como CIE XYZ. Fue diseñado por la CIE en 1931 con el fin de modelar matemáticamente el espacio del color.

$$X = \int_0^{\infty} I(\lambda) \cdot \bar{x}(\lambda) d\lambda$$

$$Y = \int_0^{\infty} I(\lambda) \cdot \bar{y}(\lambda) d\lambda$$

$$Z = \int_0^{\infty} I(\lambda) \cdot \bar{z}(\lambda) d\lambda$$



- Los valores X-Y-Z representan (simulan) los estímulos de la visión humana.

Funciones de color según longitud de onda

CIE Yyx

- Para facilitar el proceso de visualización, el canal Y fue deliberadamente designado como el canal de iluminancia. De esta forma es posible presentar un espacio de colores usando dos componentes (x,y)

$$Y = Y$$

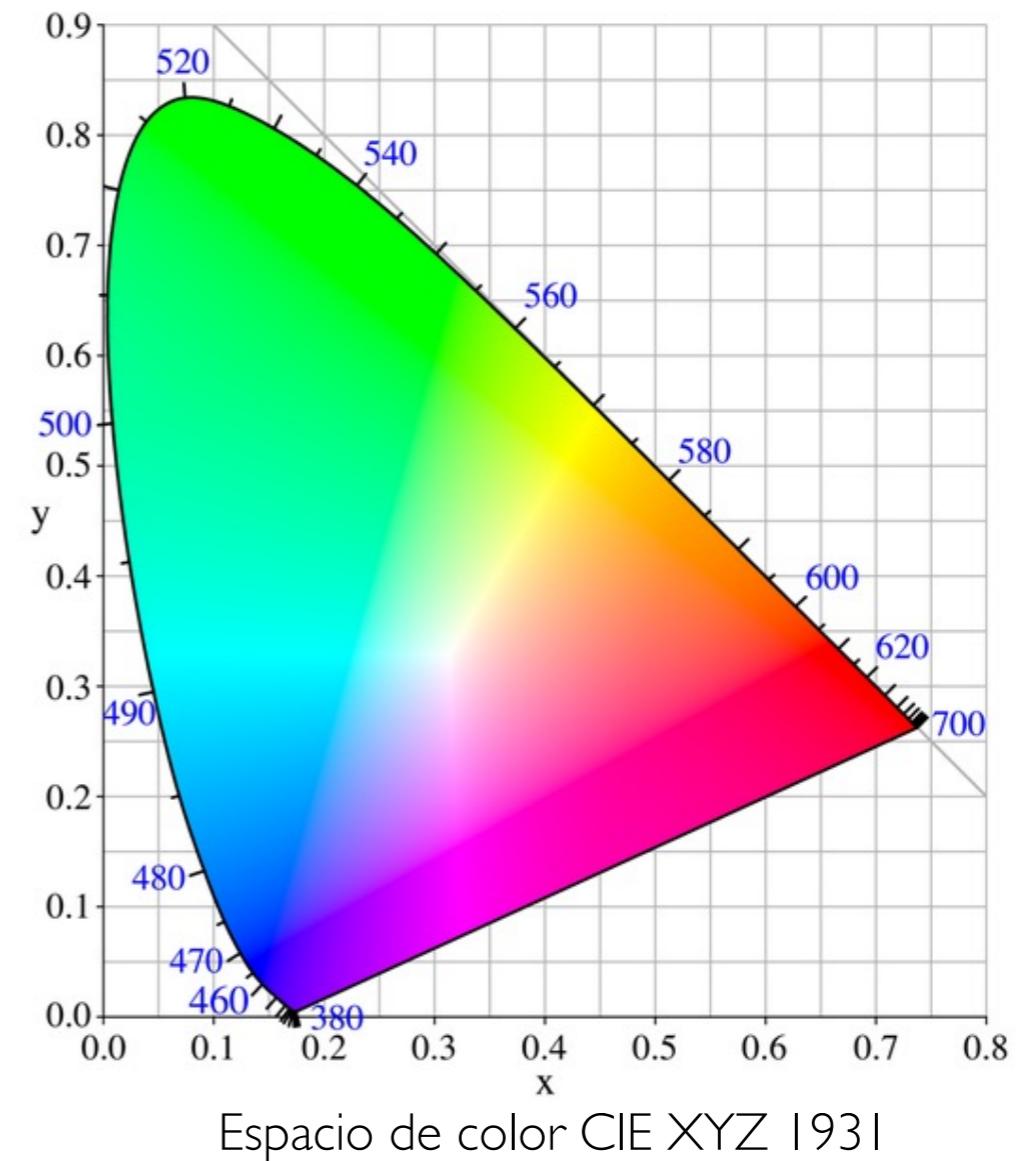
$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

- La transformada inversa es posible calcularla de la siguiente forma

$$X = \frac{Y}{y} \cdot x$$

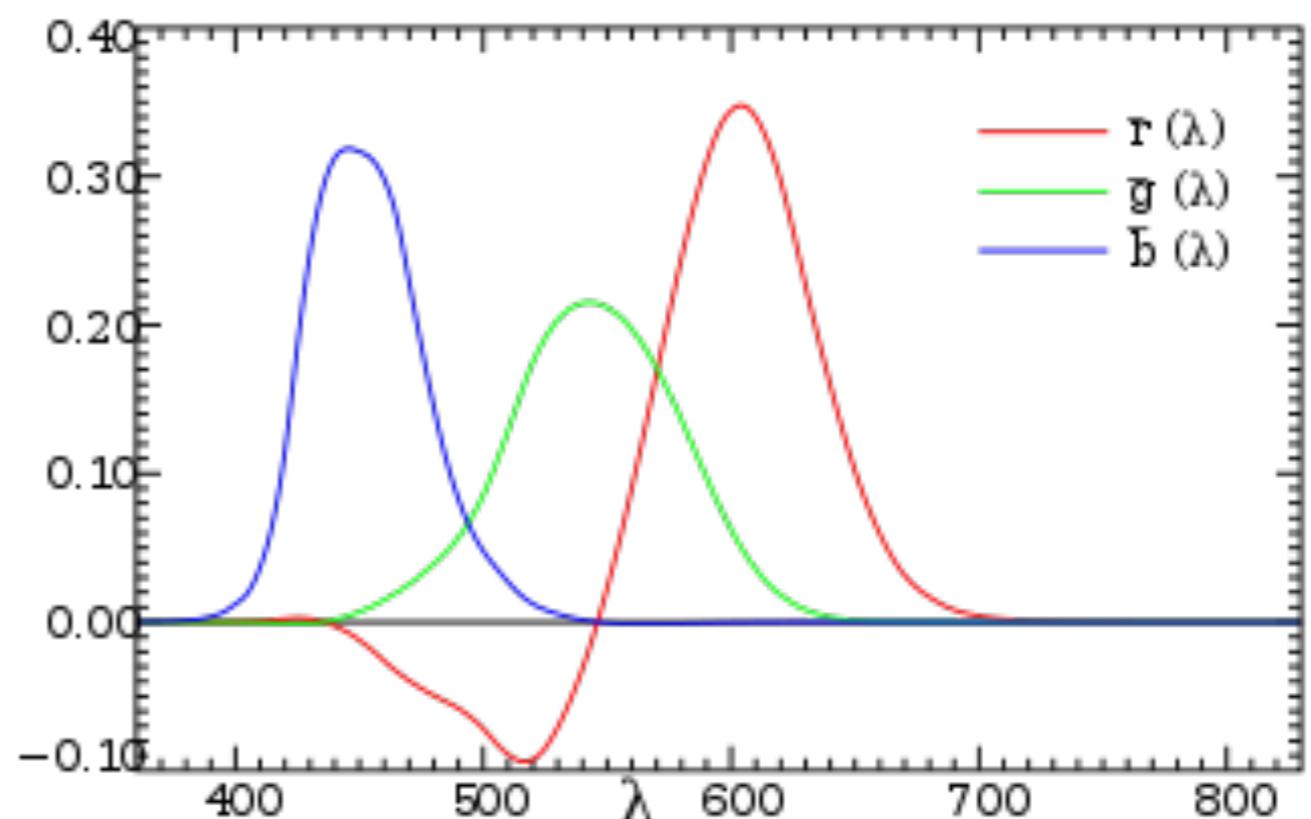
$$Z = \frac{Y}{y} \cdot (1 - x - y)$$



CIE RGB

- El espacio CIE RGB fue diseñado a través de serie de experimentos realizados en 1920 por los investigadores David Wright and John Guild que permitieron construir el estándar CIE XYZ de 1931

$$R = \int_0^{\infty} I(\lambda) \cdot r(\lambda) d\lambda$$
$$G = \int_0^{\infty} I(\lambda) \cdot g(\lambda) d\lambda$$
$$B = \int_0^{\infty} I(\lambda) \cdot b(\lambda) d\lambda$$

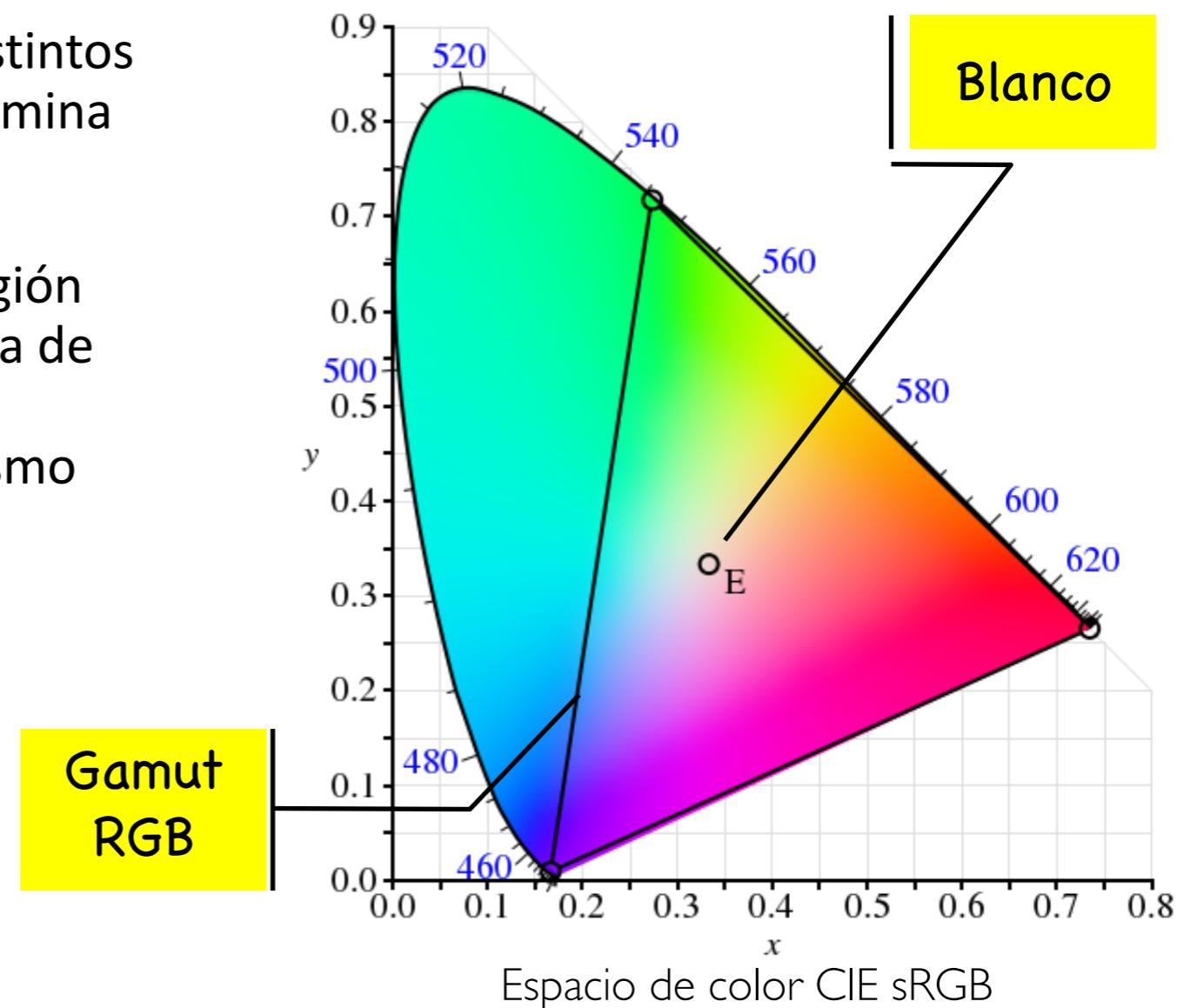


- Los valores de la función de color fueron construidos por medio de una pantalla circular de 2 grados.

Funciones de color según longitud de onda

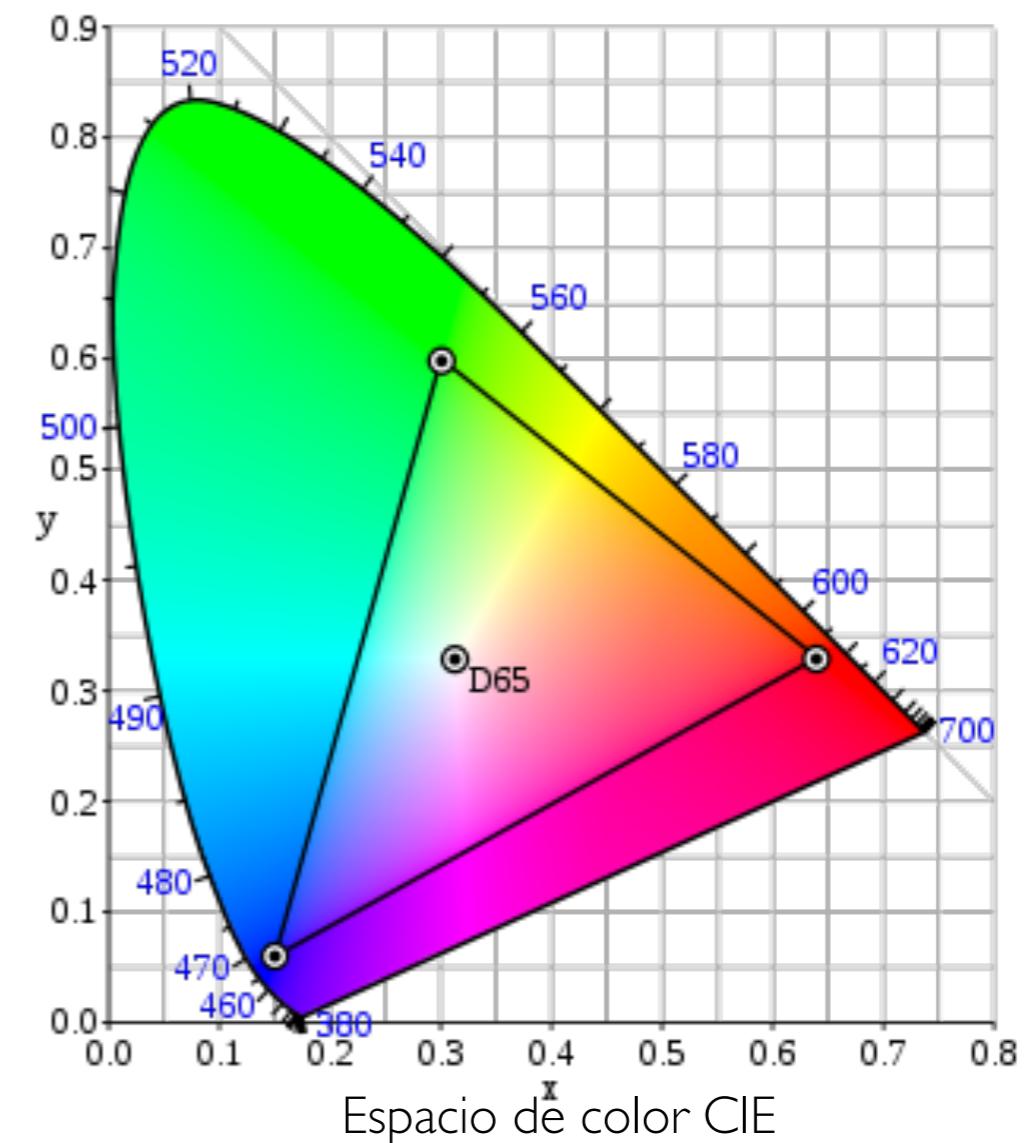
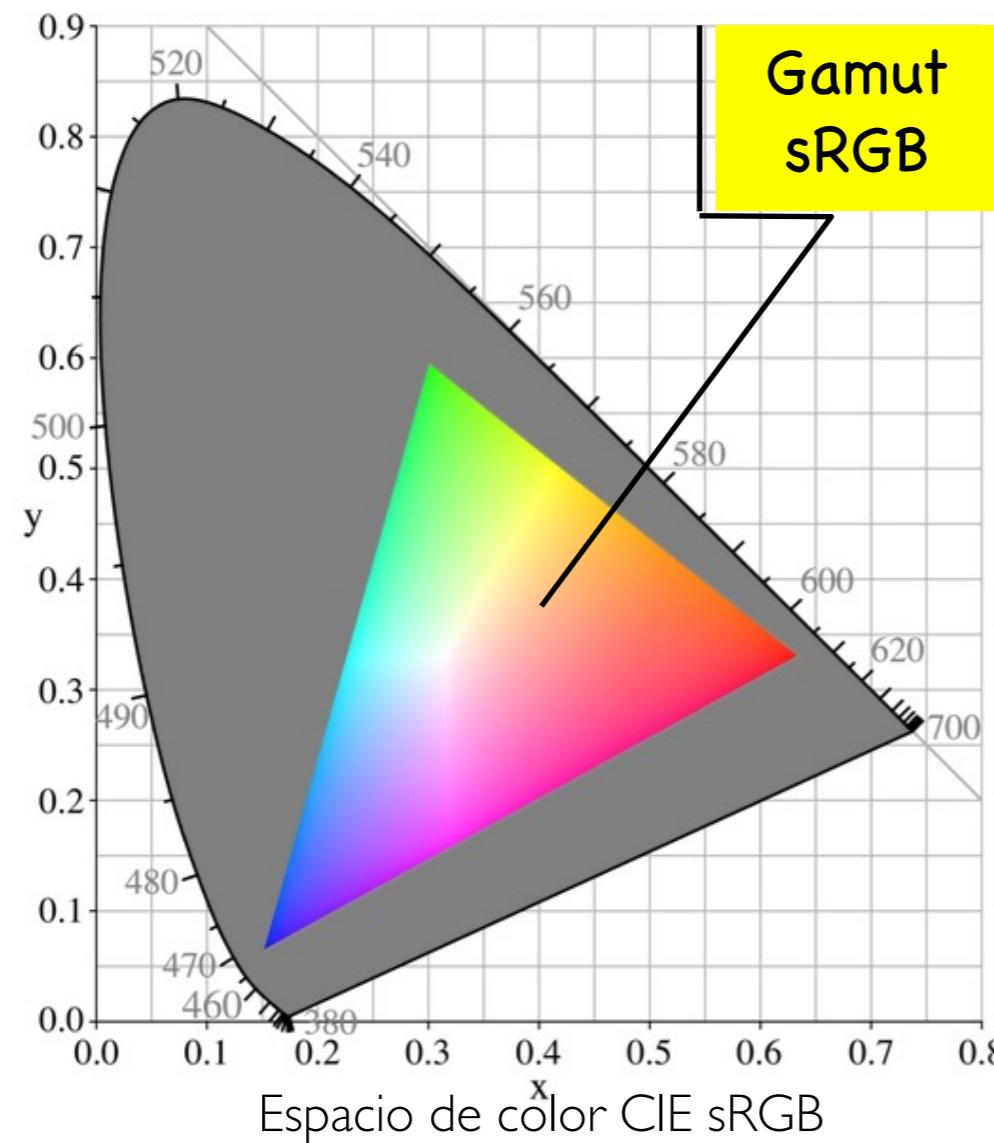
CIE RGB

- El espacio CIE RGB fue diseñado a través de serie de experimentos realizados en 1920 por los investigadores David Wright and John Guild. Esto permitió al CIE determinar el estándar CIE XYZ de 1931.
- El espacio de color que los distintos dispositivos emplean se denomina **gamut**.
- **Gamut** corresponde a una región específica dentro de una gama de colores. De esta forma si dos dispositivos comparten el mismo gamut, ambos podrán ver los mismos colores.



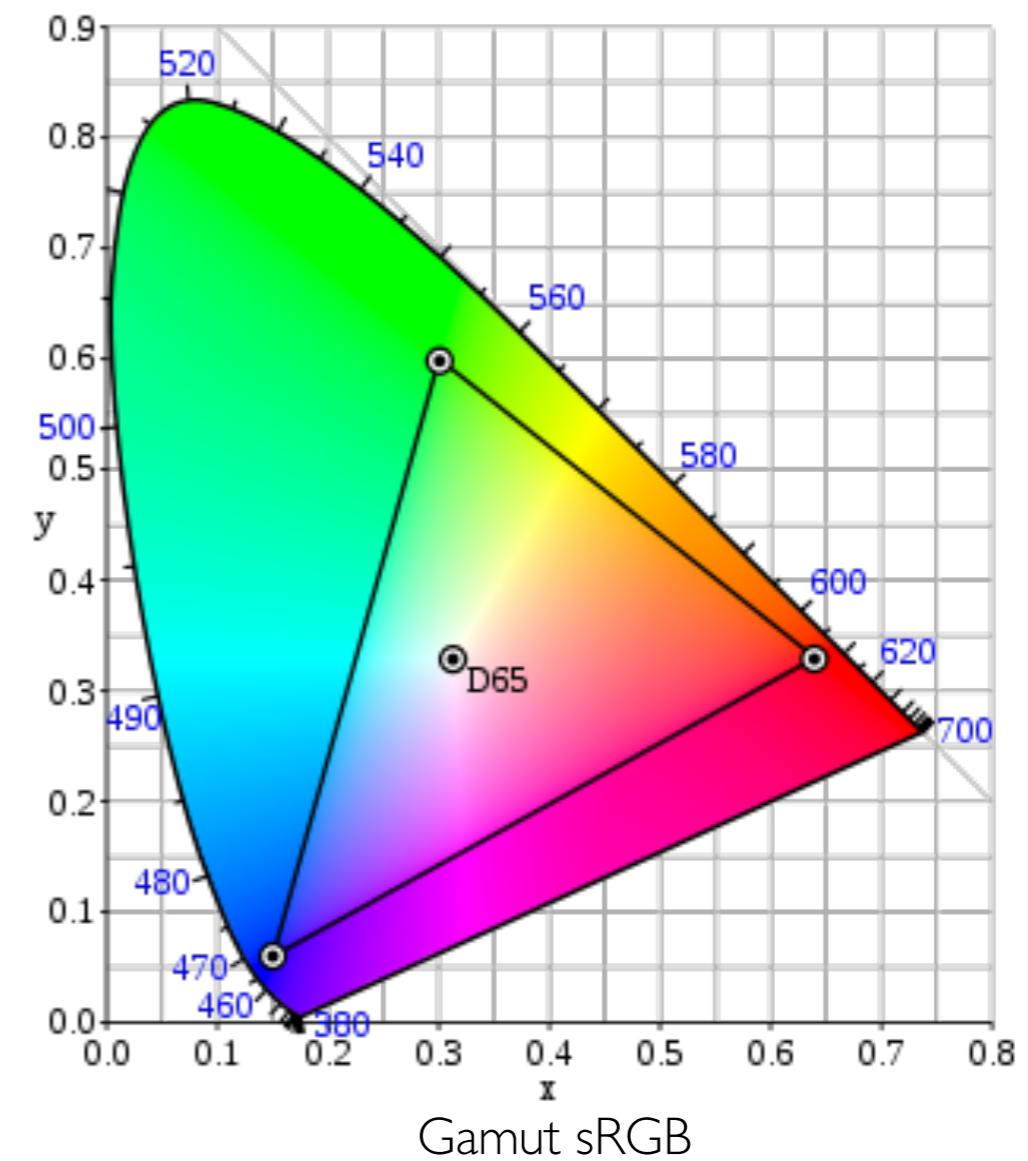
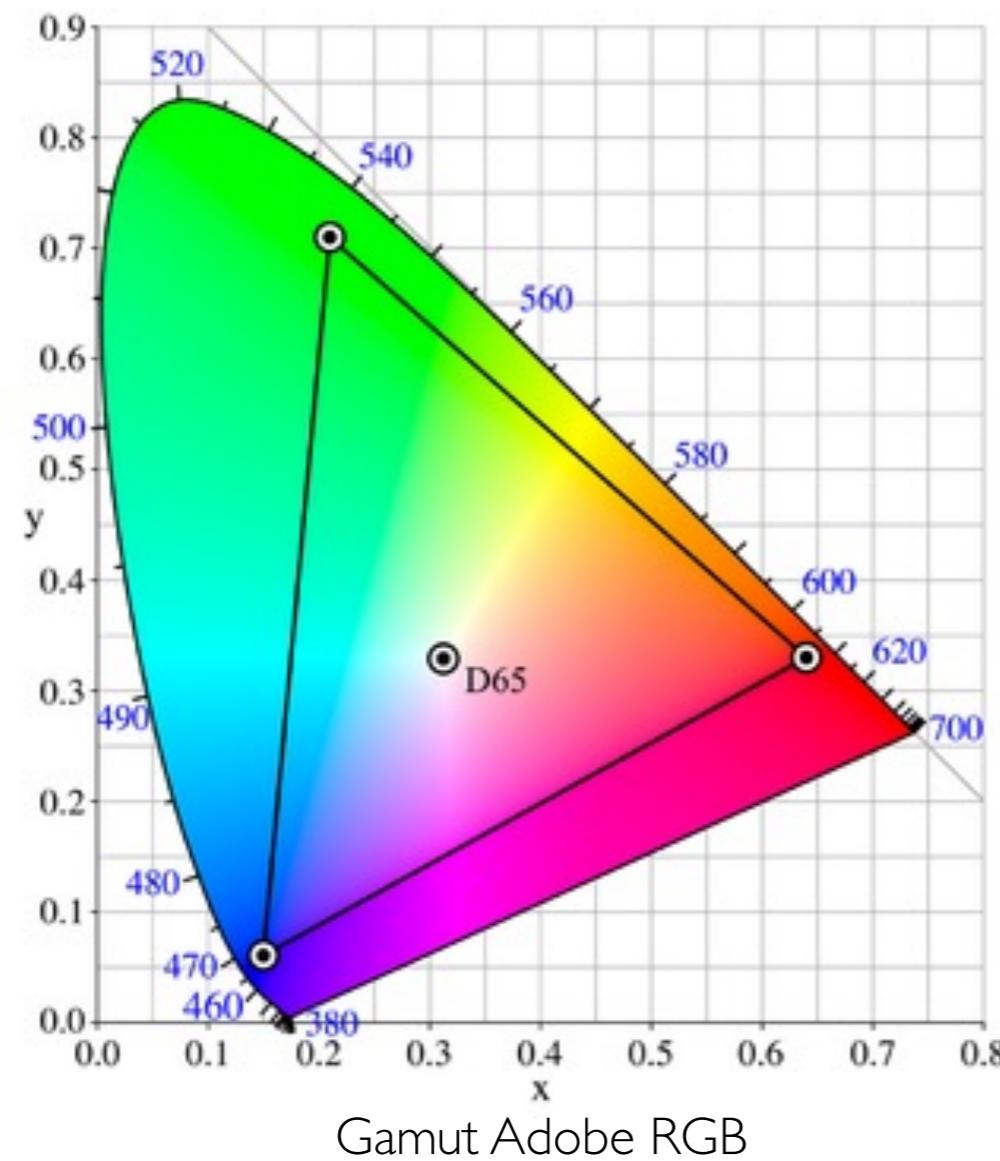
Gamut sRGB

- Para que los dispositivos (cámaras, scaners, impresoras, monitores) cumplan un estándar en cuanto al color utilizado, en 1996, Hewlett-Packard y Microsoft presentaron el modelo sRGB.



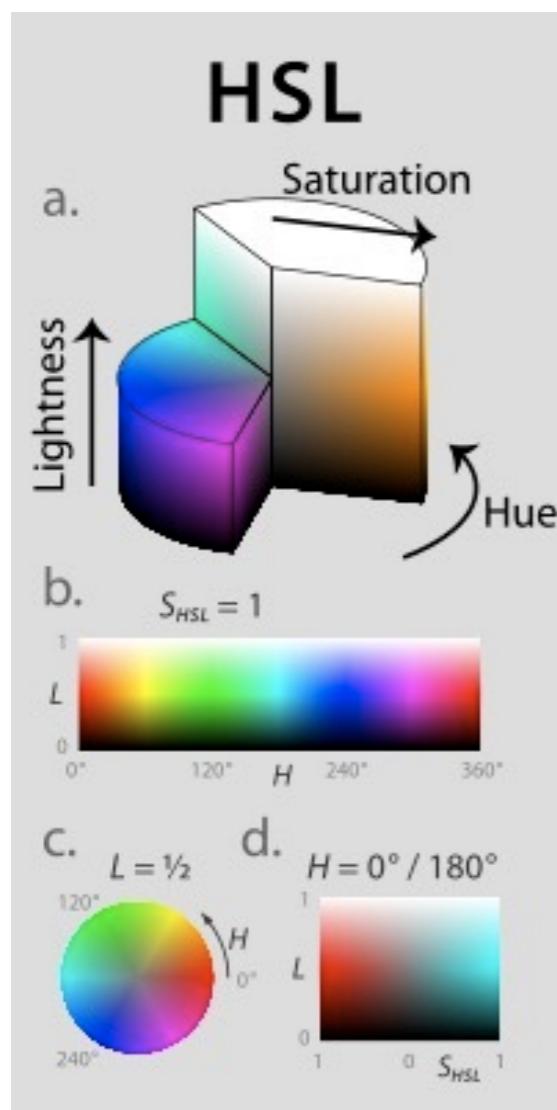
Gamut Adobe RGB

- Debido a la limitada capacidad de colores del sistema sRGB, la empresa Adobe presentó en 1998 un gamut más amplio, permitiendo mejorar la calidad de las impresiones CMYK.



■ HSV y HSL

- Los modelos de color HSV y HSL corresponden a una representación en coordenadas cilíndricas del modelo de color RGB. A diferencia de los modelos perceptuales XYZ, estos dos modelos no relacionan la complejidad de la apariencia del color

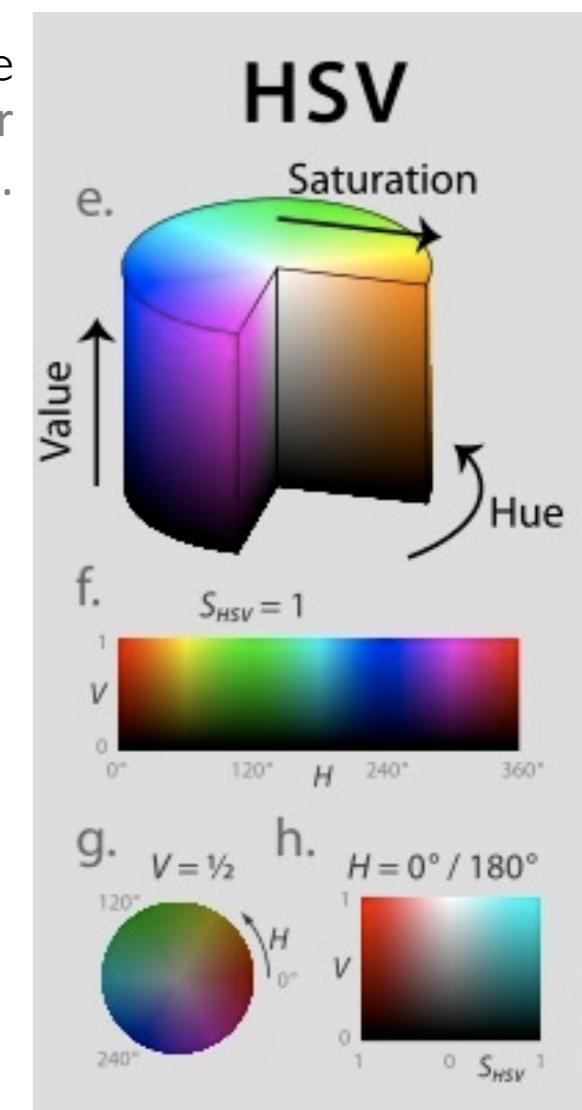


HSL: Hue-Saturation-Lightness
Fue desarrollado por la
empresa Textronik en 1979.



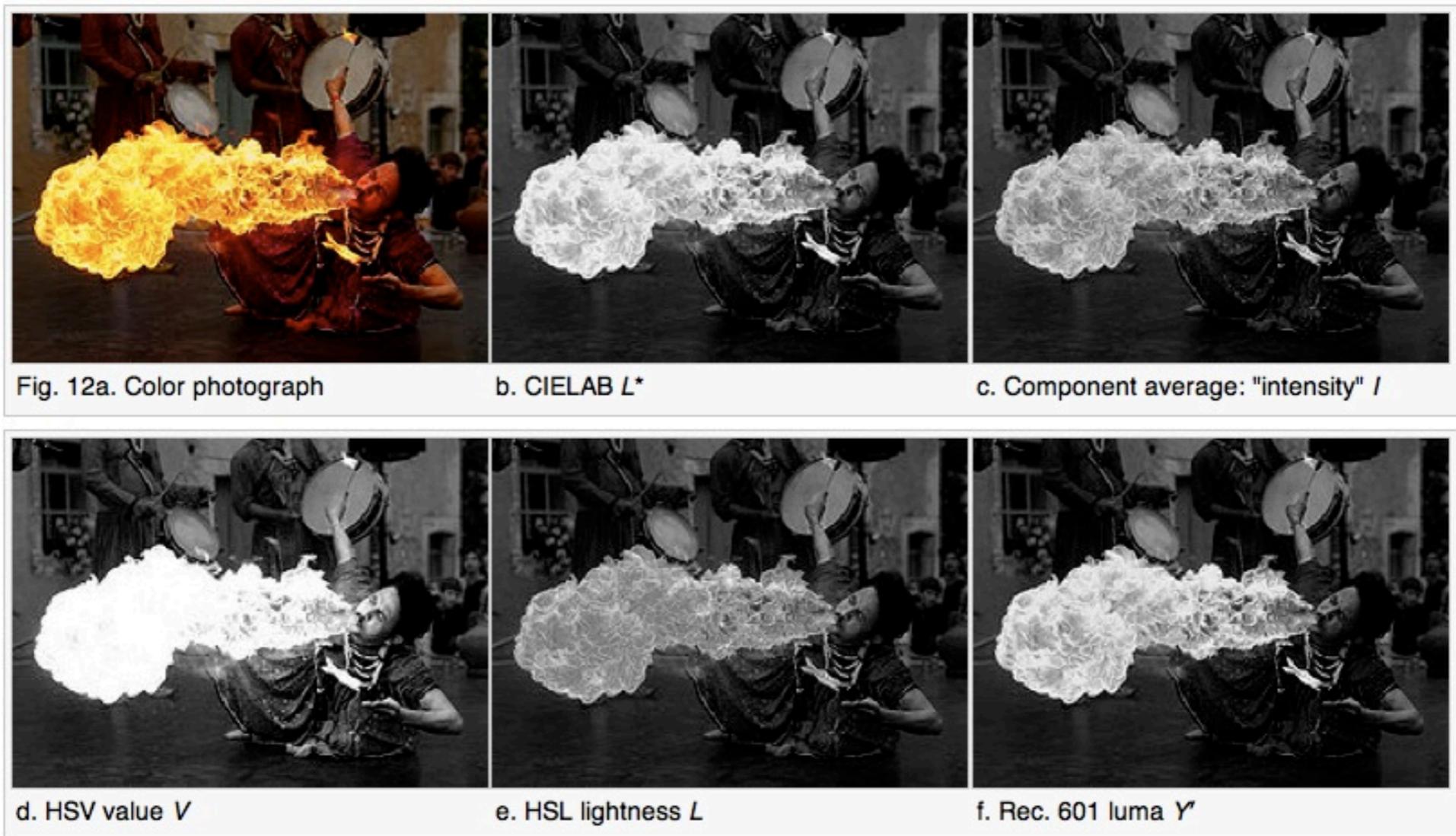
HSV: Hue-Saturation-Value
Fue desarrollado en 1978 por
Alvy Ray Smith en Xerox.

En ambos modelos, el ángulo
alrededor del eje vertical
corresponde al color (hue)



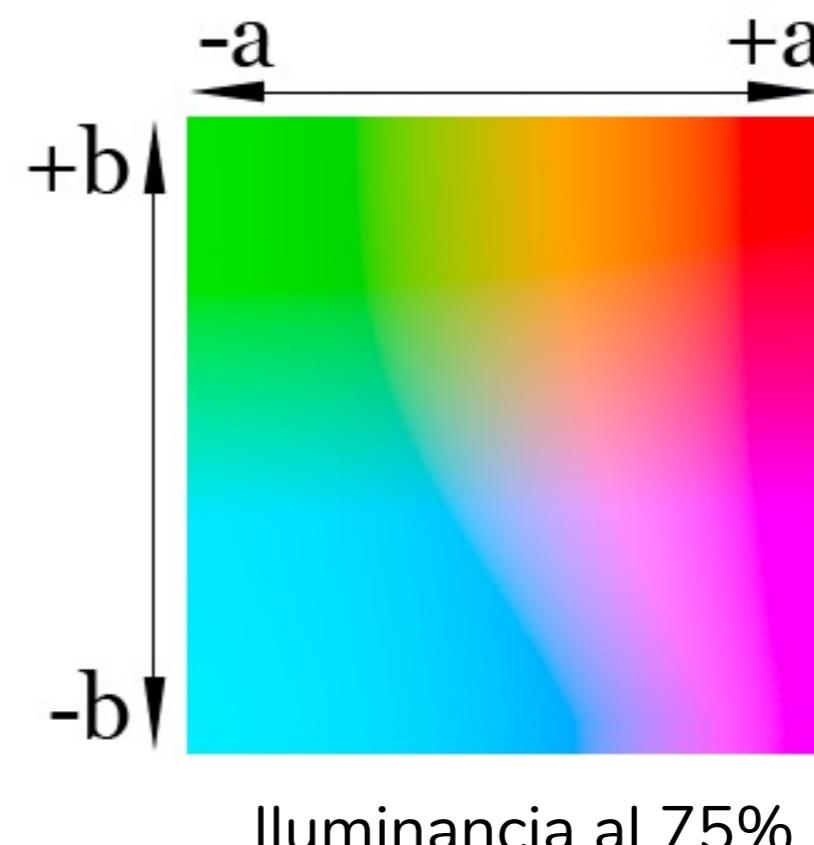
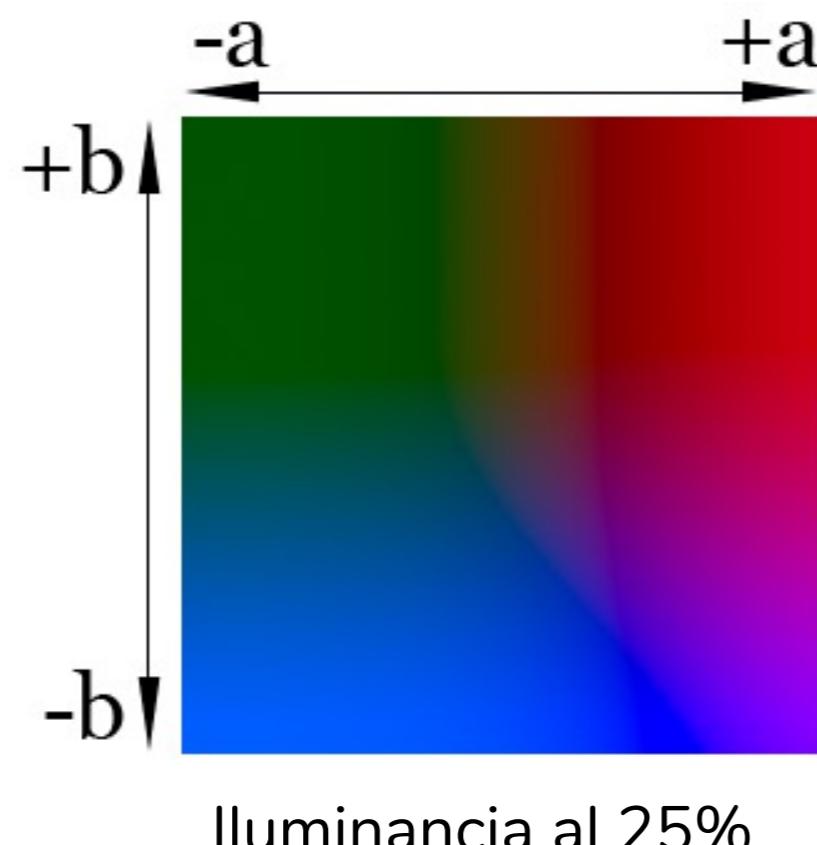
Ventajas

- Los modelos de color HSL y HSV son muy utilizados en algoritmos de visión por computador y procesamiento de imágenes. Esto se debe a que dichas transformaciones permiten separar en otra forma las relaciones presentes en el modelo RGB



■ CIE L*a*b

- El mayor problema del espacio CIE XYZ es que la distribución de colores no es perceptualmente uniforme. Por ello, la CIE presentó en 1976 un nuevo estándar conocido como CIE L*a*b.
- L es la luminancia ($L=0$ negro) ($L=100$ blanco). Los valores a y b pueden ir desde -100 a +100

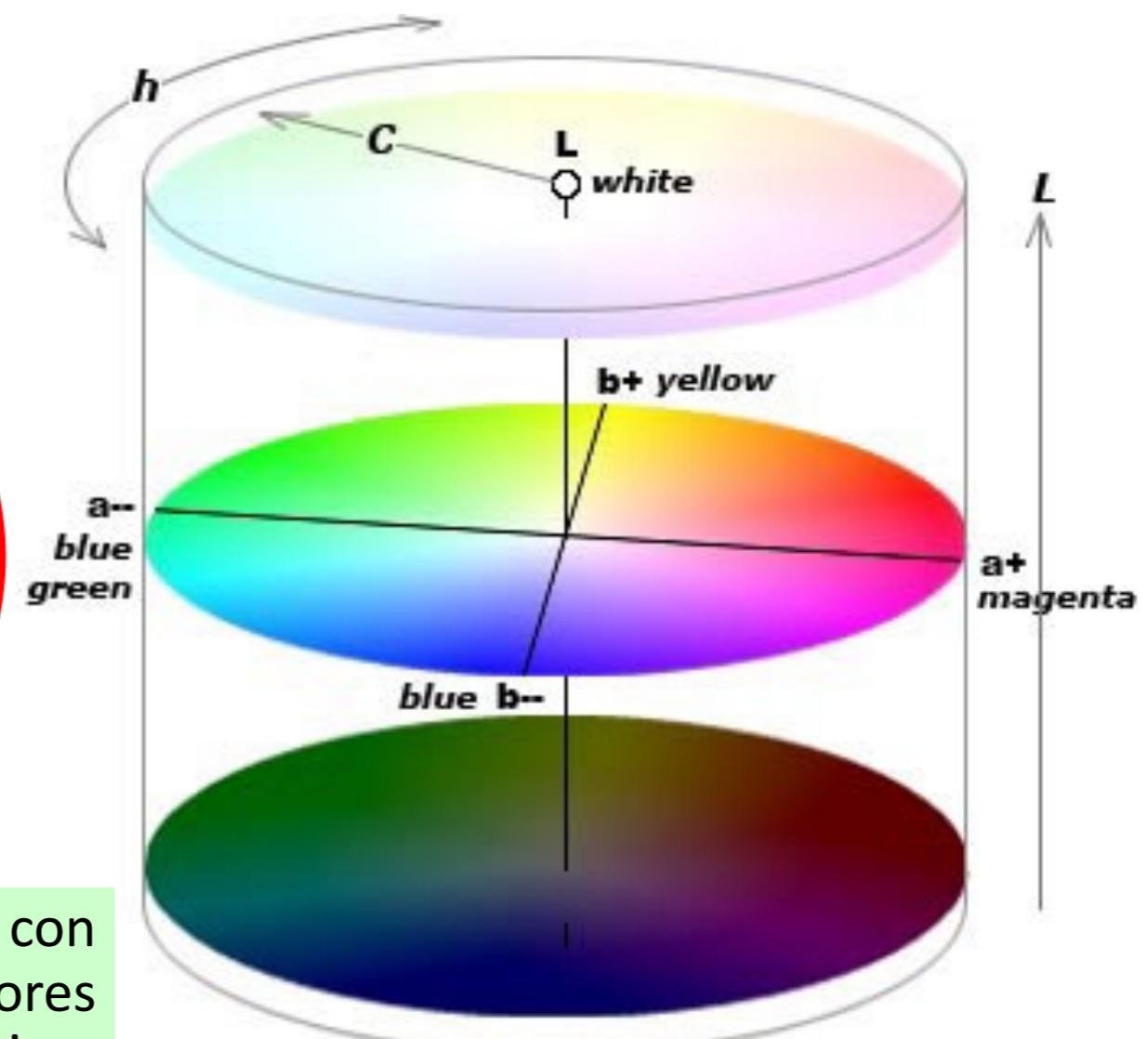


CIE L*a*b

- El mayor problema del espacio CIE XYZ es que la distribución de colores no es perceptualmente uniforme. Por ello, la CIE presentó en 1976 un nuevo estándar conocido como CIE L*a*b*.
- Los colores **a*** y **b*** son correspondientes con la regla de los colores opuestos



Esta teoría es correspondiente con la distribución de los receptores (conos) en la retina.



Espacio de color CIE LAB

- CIE L*a*b

- El mayor problema del espacio CIE XYZ es el limitado salto entre un color y otro ya que la diferencia no es lineal. Por ello, la CIE presentó en 1976 un nuevo estándar conocido como CIE L*a*b

$$L = 116 \cdot (Y/Y_0)^{1/3} - 16$$

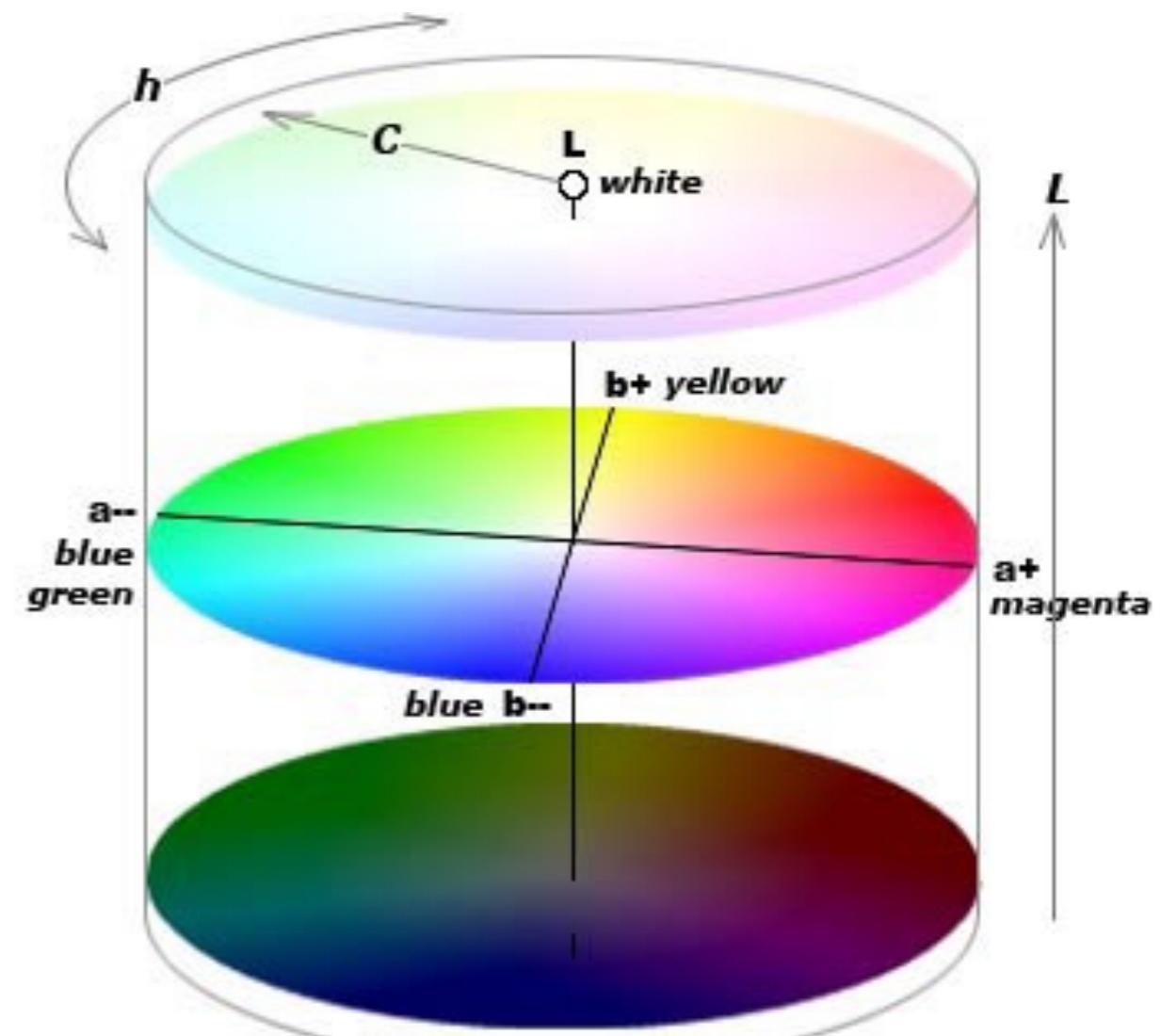
$$a = 500 \left[(X - X_0)^{1/3} - (Y - Y_0)^{1/3} \right]$$

$$b = 200 \left[(Y - Y_0)^{1/3} - (Z - Z_0)^{1/3} \right]$$

$$C = \sqrt{(a^2 + b^2)}$$
Croma

$$h = \tan^{-1} \left(\frac{b}{a} \right)$$
Saturación

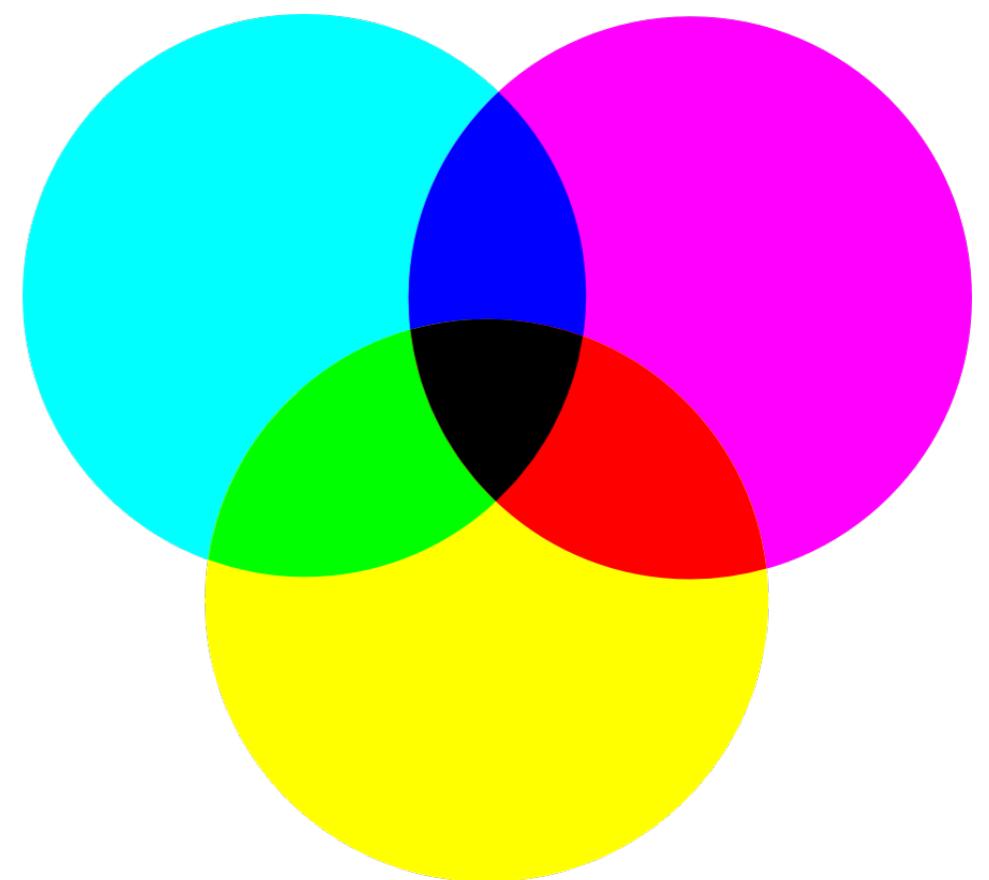
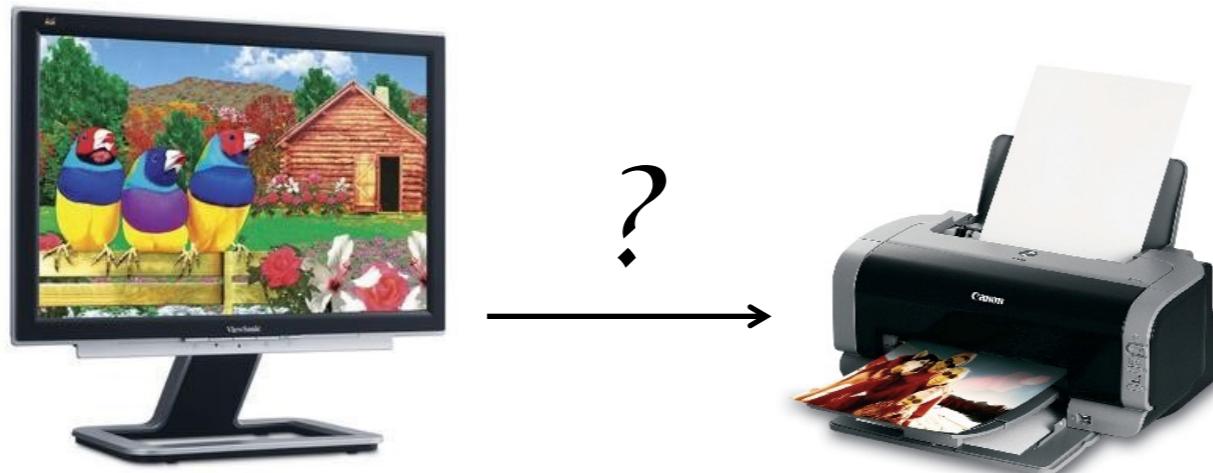
- $X_0 Y_0 Z_0$ es un valor de referencia del blanco.



Espacio de color CIE LAB

■ Impresión (CMYK) / Visualización (RGB)

- Normalmente los dispositivos de impresión presentan una gama de colores distinta a los monitores o cámaras digitales. Esto se debe a que la tinta impresa presenta un modelo de color subtractivo.
- El problema consiste en ¿cómo imprimir una imagen de un monitor a una impresora sin perder calidad?



Colores primarios CMYK
Cyan-Magenta-Amarillo-Negro

■ Impresión (CMYK) / Visualización (RGB)

- No existe una forma única de cambiar los colores ya que cada modelo posee su propia espacio de color Gamut. Esto significa que con algoritmos distintos obtendremos diferentes impresiones.

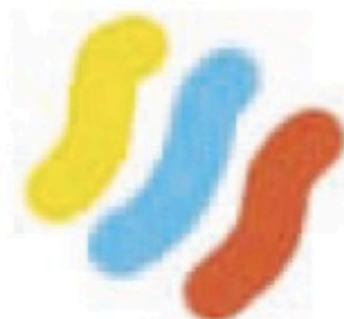


Durante el proceso de conversión, existen colores en la paleta RGB que no están presentes en la CMYK. Para ello, estos colores deben ser convertidos a través de algoritmos de conversión.



rgb colors

(what you see on screen)



cmyk colors

(printing inks will do this)



rgb colors

(what you see on screen)



cmyk colors

(printing inks will do this)

- Ejemplo del proceso de separación en canales CMYK



Algunas impresoras simulan el negro al unir los canales CMY. Esto genera una menor calidad de imagen

Gamut



Los triángulos de color que muestra la paleta de colores corresponden a la gama (gamut) que muestra dicho sistema.

