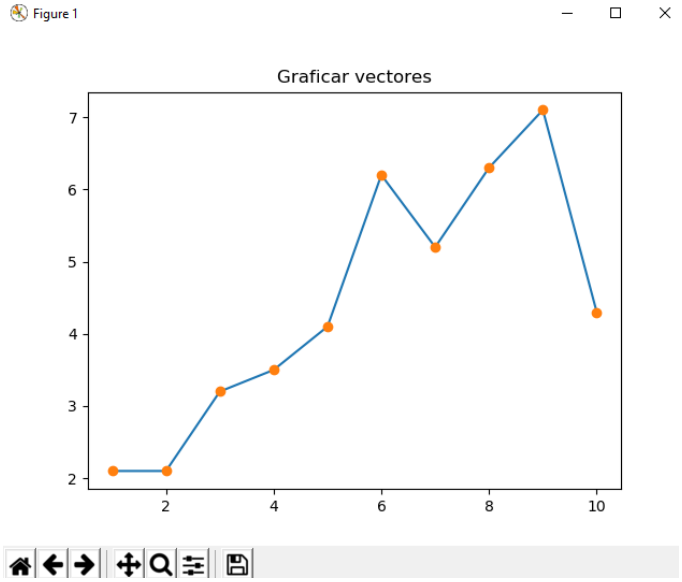
Tratamiento de imágenes usando geometría Vectorial

Ing. Francisco Alejandro Medina

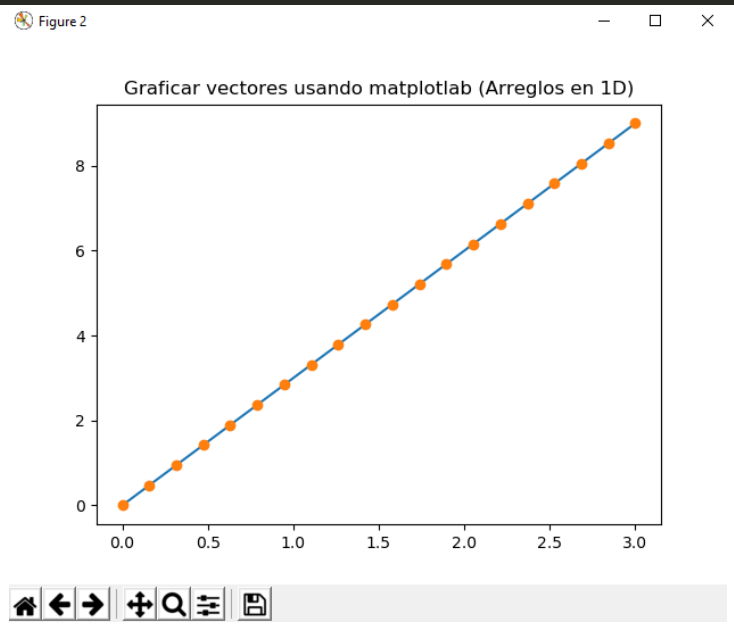
Graficar vectores

```python
import math
import numpy as np
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
y = [2.1, 2.1, 3.2, 3.5, 4.1, 6.2, 5.2, 6.3, 7.1, 4.3];
plt.figure()
plt.plot(x, y)       # gráfica con línea continua
plt.title("Graficar vectores")
plt.plot(x, y, 'o')  # gráfica con línea punteada

```



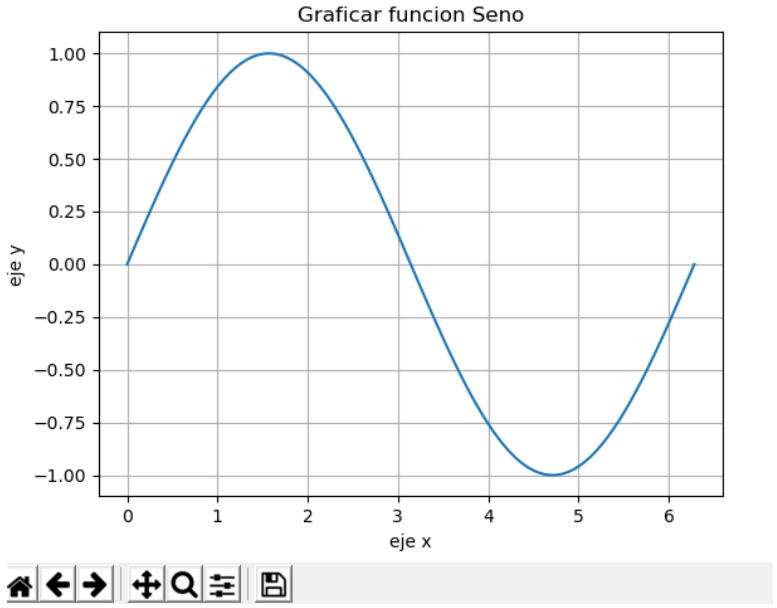
Vectores usando linspace

```
12  #-------------------------------------
13  #Graficar vectores usando matplotlab (Arreglos en 1D)
14  x = np.linspace(0, 3, 20)
15  y = np.linspace(0, 9, 20)
16  plt.figure()
17  plt.title("Graficar vectores usando matplotlab (Arreglos en 1D)")
18  plt.plot(x, y)       # gráfica con línea continua
19  plt.plot(x, y, 'o')  # gráfica con línea punteada
20
```



Graficar una función

```
21  #-----------------------------------------------------
22  xx = np.linspace(0, 2 * math.pi, 100)
23  yy = np.sin(xx);
24  plt.figure()
25  plt.title("Graficar funcion Seno")
26  plt.plot(xx, yy)        # gráfica con línea continua
27  plt.grid(True)
28  plt.xlabel("eje x")
29  plt.ylabel("eje y")
30  plt.show()              # <-- mostrar la gráfica
```



Crear nuestra propia función Seno

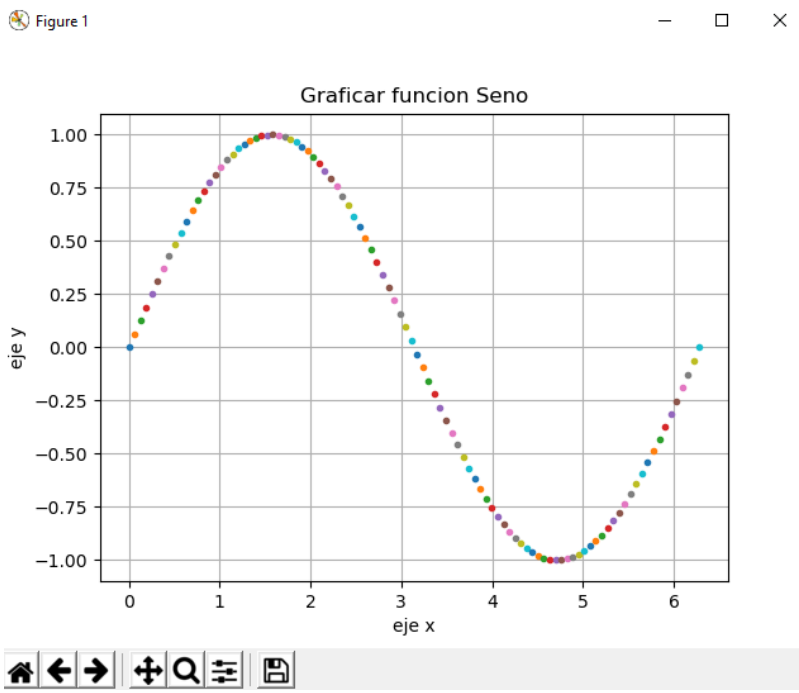
```
1  import math
2  import numpy as np
3  import matplotlib.pyplot as plt
4  #-----------------------------------
5  def factorial(n):
6      if n==0 or n==1:
7          resultado=1.
8      elif n>1:
9          resultado=n*factorial(n-1)
10     else:
11         return "NaN"
12     return resultado
13 #-----------------------------------
```

Crear nuestra propia función Seno

```
14  def misin(x):
15      suma=0
16      termino=0
17      n=0
18      aportemin=0.00001
19      while (True):
20          termino=((-1)**n)/(factorial(2*n+1))*x**(2*n+1)
21          suma=suma + termino
22          n=n+1
23          if (math.fabs(termino)<aportemin):
24              break
25      return(suma)
26  #-----------------------------------
```

Crear nuestra propia función Seno

```
26  #-------------------------------------
27  plt.figure()
28  plt.title("Graficar funcion Seno")
29  plt.grid(True)
30  plt.xlabel("eje x")
31  plt.ylabel("eje y")
32  x = np.linspace(0, 2 * math.pi, 100)
33  i=0
34  while i < 100:
35      y = misin(x[i])
36      plt.plot(x[i], y,'.' )
37      i=i+1
38  plt.show()
```



Ejemplo 1

```
1  # Creando matrices y visualizándolas como imágenes
2  # Las imágenes se codifican como matrices. En particular,
3  # las imágenes de intensidad o escala de grises se codifican
4  # como una matriz de dos dimensiones, donde cada número
5  # representa la intensidad de un pixel.
6
7  # Pero eso significa que cualquiera de estas matrices que
8  # generamos se puede visualizar como una matriz. Para visualizar
9  # imágenes, usamos el módulo pyplot de la librería matplotlib.
10
11
12 import matplotlib.pyplot as plt
13
14 #configuracion necesaria de pyplot para ver las imagenes en escala de grises
15 plt.rcParams['image.cmap'] = 'gray'
16
17 #tambien importamos numpy ya que lo usamos para crear y manipular matrices
18 import numpy as np
```

Ejemplo 1

```
23  # Una matriz de ceros.
24  imagen_negra = np.zeros(size)
25
26  #visualizamos la matriz
27  #Se ve como una imagen negra, ya que todos los elementos (pixeles) tienen intensidad 0
28
29  plt.imshow(imagen_negra,vmin=0,vmax=1)
30  plt.figure()
31  # (es necesario indicar vmin y vmax para que pyplot sepa que el minimo es 0 y el maximo 1)
32  # (solo imagenes escala de grises)
33
34  # IMAGEN BLANCA
35  # Una matriz de unos.
36  imagen_blanca = np.ones(size)
37
38  #visualizamos la matriz
39  #Se ve como una imagen blanca, ya que todos los elementos (pixeles) tienen intensidad 1
40
41  plt.imshow(imagen_blanca,vmin=0,vmax=1)
42  #creamos otra figura para mostrar la imagen (sino el proximo imshow sobreescribe al anterior)
43  plt.figure()
```

Ejemplo 1

```
45  # IMAGEN GRIS
46  # Una matriz con valor 0.5 en todos sus elementos
47  imagen_gris = np.ones(size)*0.5
48
49  #visualizamos la matriz
50  #Se ve como una imagen gris, ya que todos los elementos (pixeles) tienen intensidad 0.5
51  plt.imshow(imagen_gris,vmin=0,vmax=1)
52
53  # IMAGEN GRIS OSCURO
54  # Una matriz con valor 0.2 en todos sus elementos
55  imagen_gris_oscuro = np.ones(size)*0.2
56
57  #visualizamos la matriz
58  #Se ve como una imagen gris, ya que todos los elementos (pixeles) tienen intensidad 0.5
59  plt.imshow(imagen_gris_oscuro,vmin=0,vmax=1)
60
61  #creamos otra figura para mostrar la imagen (sino el proximo imshow sobreescribe al anterior)
62  plt.figure()

64  # IMAGEN ALEATORIA
65  # Una matriz con valor aleatorio
66  imagen_aleatoria = np.random.rand(size[0],size[1])
67
68  #visualizamos la matriz
69  #Se ve como una imagen gris, ya que todos los elementos (pixeles) tienen intensidad 0.5
70  plt.imshow(imagen_aleatoria,vmin=0,vmax=1)
71  plt.show()
```

La Imagen

Una imagen es una representación esculpida, pintada o fotografiada de un objeto o escena. Actualmente estas pueden ser representadas sobre un monitor electrónico como una pantalla Led.

Imagen Digital

- Una imagen digital es una celda compuesta por unos elementos llamados píxeles, que son los componentes más pequeños de una imagen digital.

- Cada píxel es un espacio en la memoria de la computadora donde se almacena un numero y este número representa la definición del color y el brillo de una parte de la imagen. Cada píxel puede definir un color solamente y el numero de píxeles define la cantidad de información que contiene una imagen.

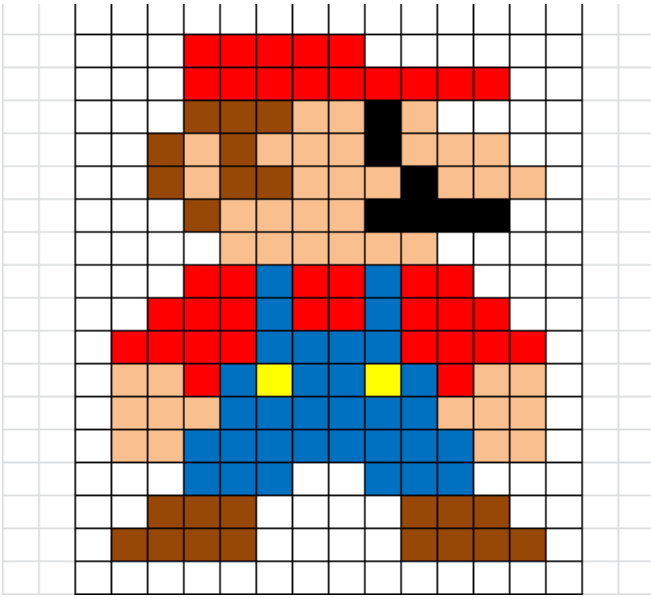
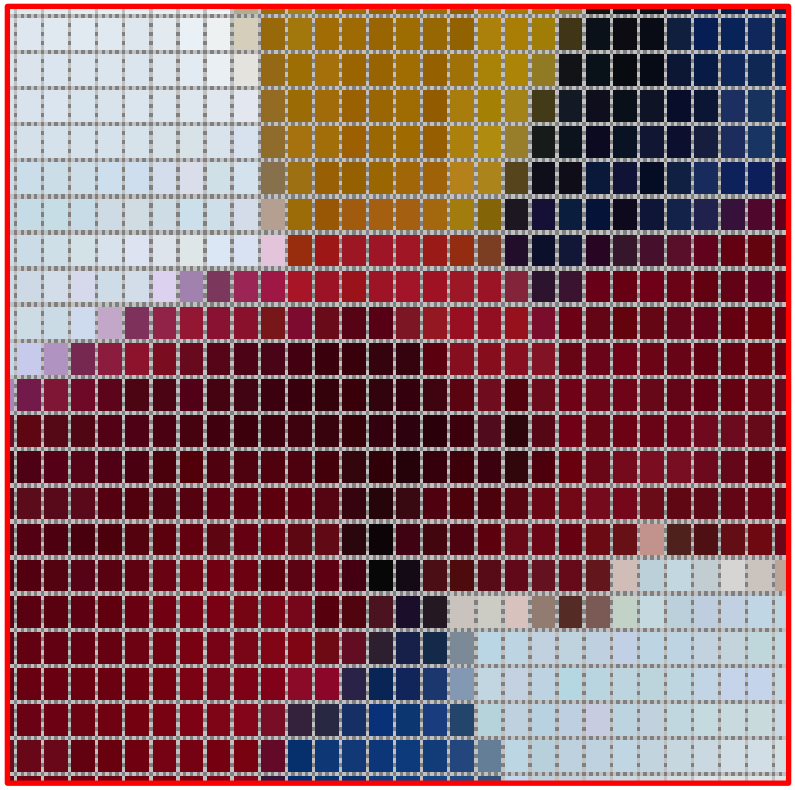


Imagen Digital

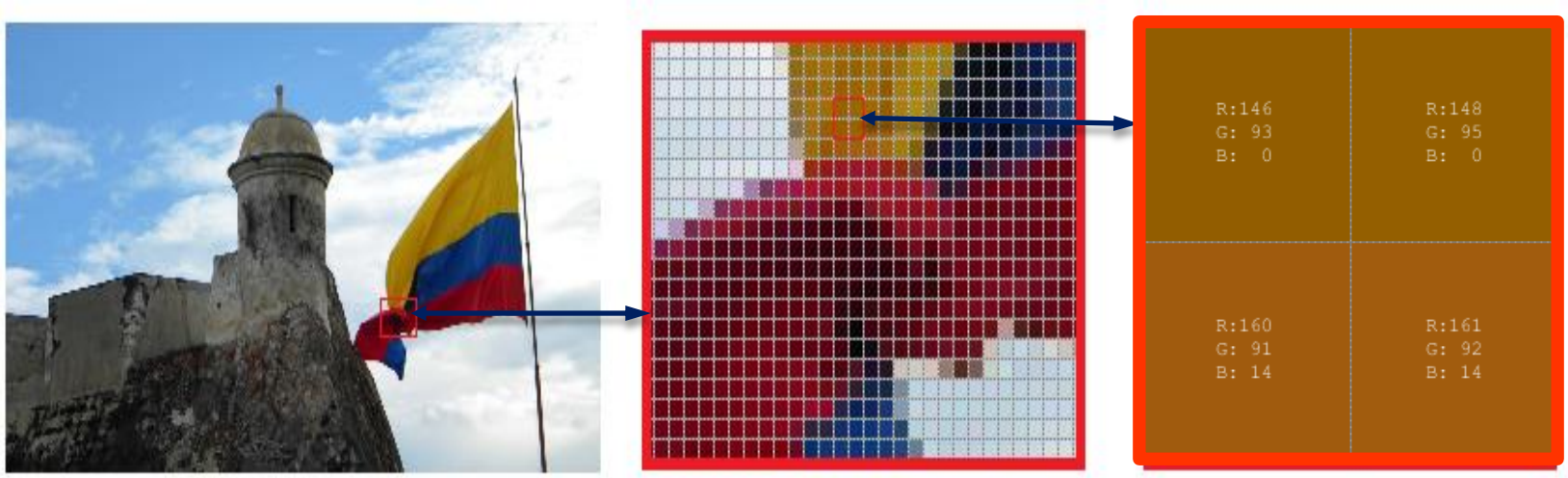


Una imagen digital es una celda compuesta por unos elementos llamados píxeles, que son los componentes más pequeños de una imagen digital.

Imagen Digital

Cada cuadro de la imagen se denomina pixel (*picture element*)

Una imagen a color tiene 3 valores de color por cada pixel.



Una imagen en blanco y negro tiene solo un valor de color por cada pixel.

Pixels

- Una imagen digital, *I*, es un arreglo de una regilla 2D de puntos discretos distribuidos espacialmente de forma uniforme, {*p* = (*r,c*)}, dentro de un grupo de valores enteros positivos, {*I*(*p*)}, o un grupo de valores vectoriales, *por ejemplo.,* {[*R G B*]T(*p*)}*.*
- Para cada punto formado por una fila y una columna le corresponde un valor de I.
- El par (*p, I*(*p*)) se le denomina “pixel” (*picture element*).

p = (r,c) r es la fila y c es la columna donde se localiza el pixel.

I(p) = I(r,c) es el valor del pixel en el punto p.

Si I(p) es un solo número, entonces I es una imagen monocromática.

si I(p) es un vector (Usualmente de orden 3) entonces I es una imagen a color.

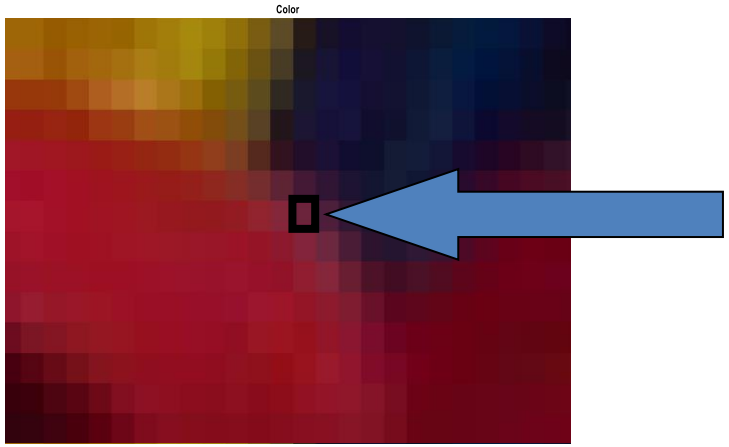
Lectura de imágenes

```python
# Lectura de imágenes
# Para leer imágenes utilizaremos el módulo io de la librería skimage.
# En ese módulo tenemos la función imread para leer imágenes de un
# archivo(y también la función imsave para escribirlas a un archivo).

import matplotlib.pyplot as plt
import numpy as np
#   Debe instalar la libreria scikit-image de python
#   https://scikit-image.org/docs/dev/install.html
from skimage import io
image=io.imread("logoutp.jpg")/255.0 # imread lee las imagenes con los pixeles
                                     # codificados como enteros en el rango 0-255.
                                     # Por eso la convertimos a flotante y en el rango 0-1
print("- Dimensiones de la imagen:")
print(image.shape)
plt.imshow(image,vmin=0,vmax=1)
plt.show()
```

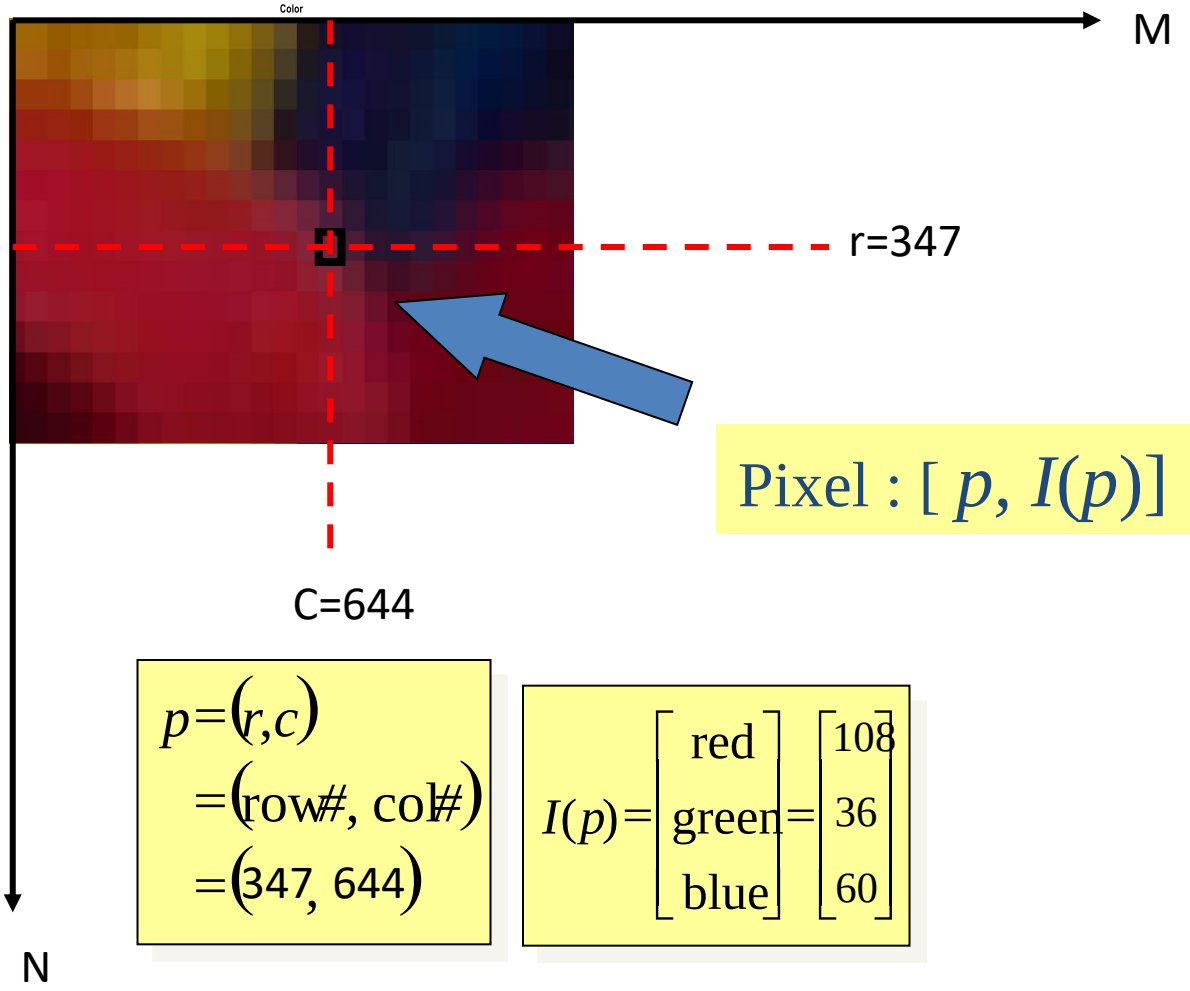
Pixels

Localización de pixel: $p = (r , c)$

Valor de pixel: $I(p) = I(r , c)$



Pixel : $[p, I(p)]$

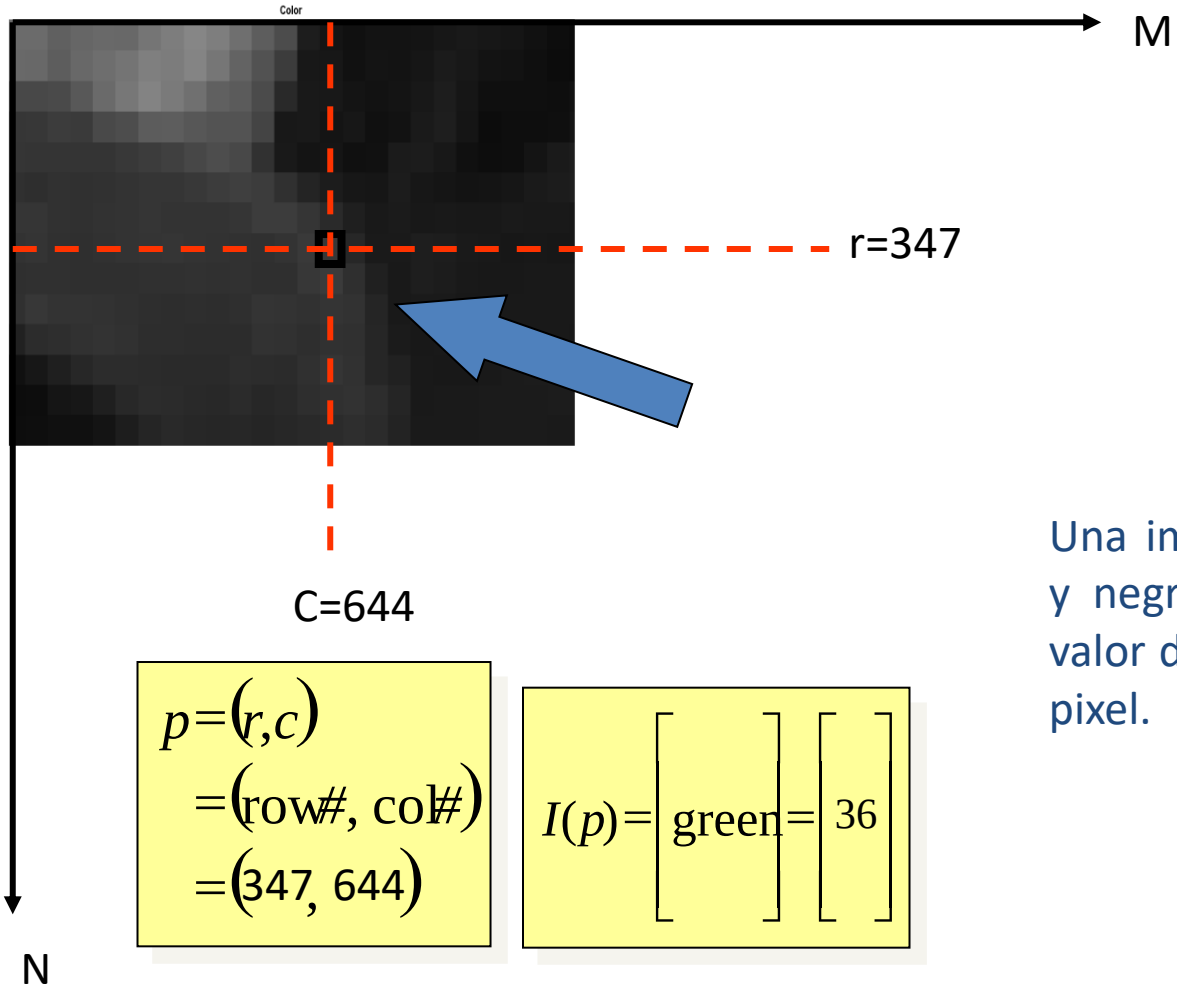


Pixel : $[\, p, I(p)]$

$$p = (r, c) = (\text{row\#}, \text{col\#}) = (347, 644)$$

$$I(p) = \begin{bmatrix} \text{red} \\ \text{green} \\ \text{blue} \end{bmatrix} = \begin{bmatrix} 108 \\ 36 \\ 60 \end{bmatrix}$$

Una imagen a color tiene 3 valores de color por cada pixel.



$$p = (r, c) = (\text{row\#}, \text{col\#}) = (347, 644)$$

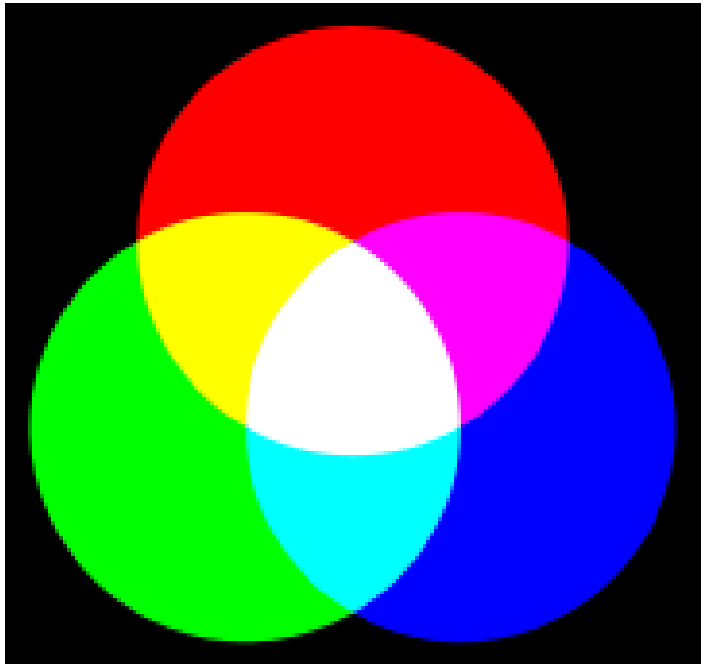
$$I(p) = \begin{bmatrix} \text{green} \end{bmatrix} = \begin{bmatrix} 36 \end{bmatrix}$$

Una imagen en blanco y negro tiene solo un valor de color por cada pixel.

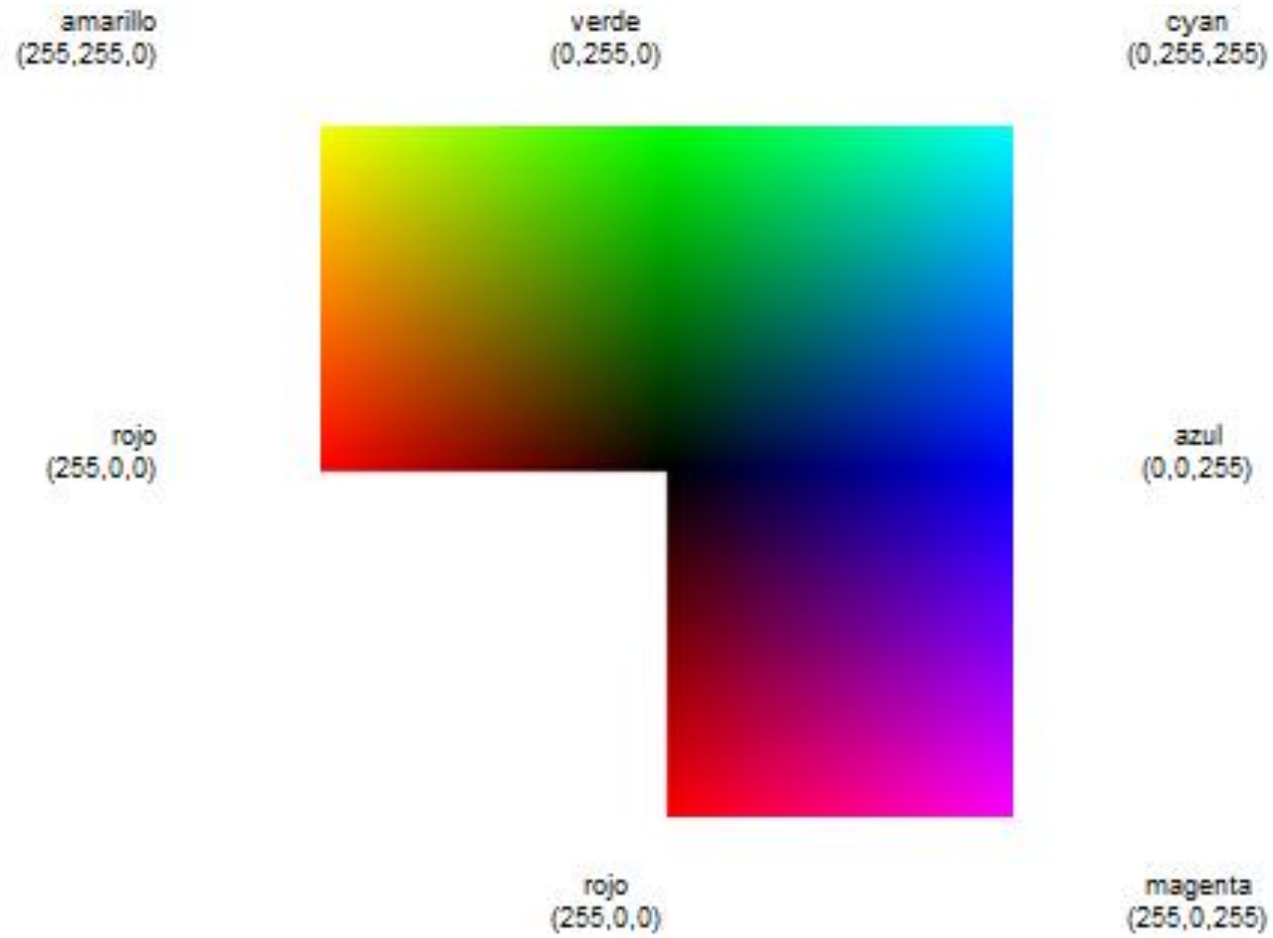
Componentes de color

- El RGB (Red, Green, Blue) representa a los colores en función de la intensidad de los colores primarios.
- Es posible representar un color mediante la adición de los tres colores luz primarios.

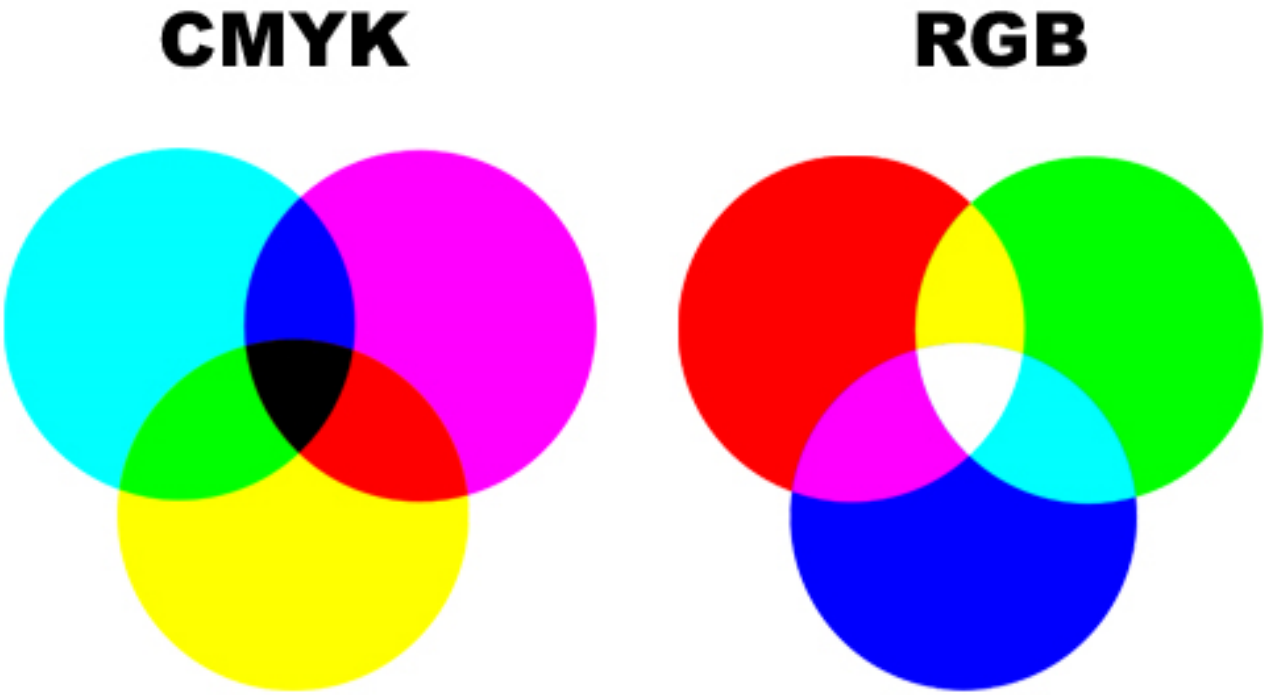
0	no interviene en la mezcla
255	aporta más intensidad.



Componentes de color

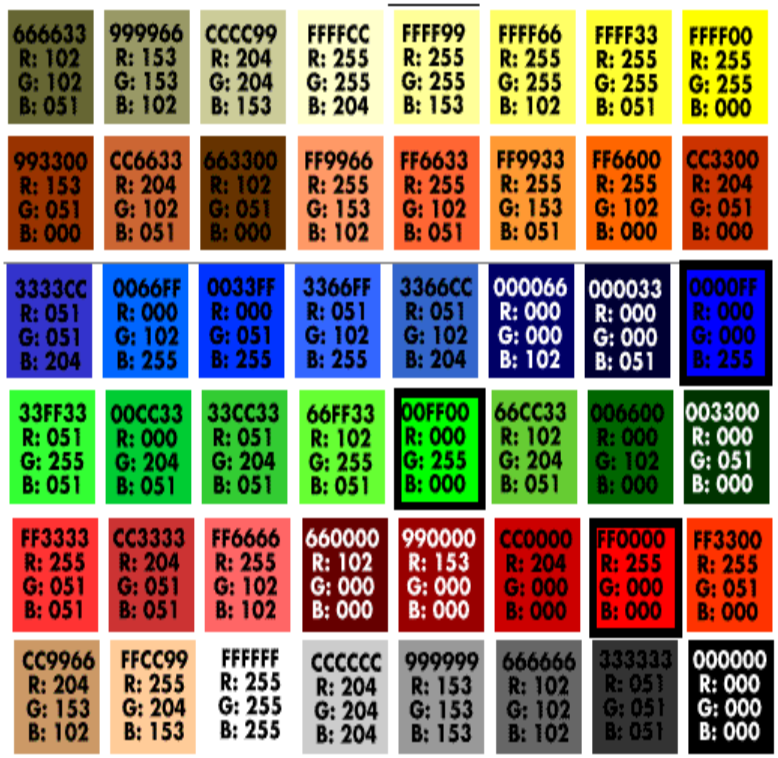


Comparación RGV y CMYK



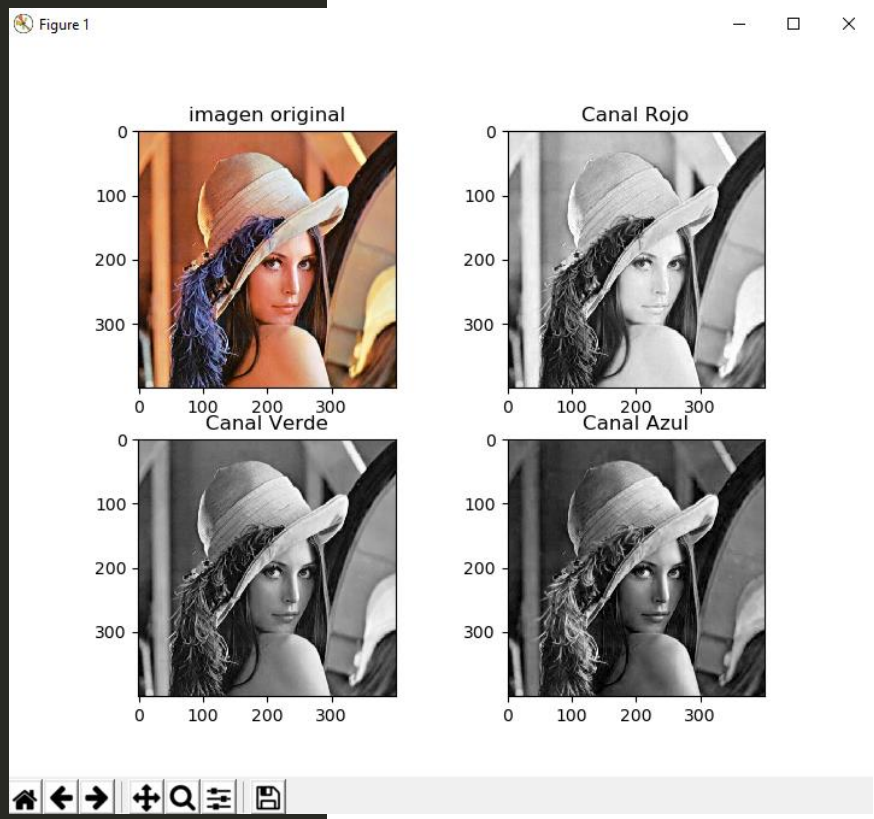
RGB

Cualquier color puede ser representado mediante la combinación de los colores rojo, verde y azul, cada uno en diferente proporción.
La combinación RGB estándar indica 256 niveles por cada canal, es decir por cada color rojo, verde o azul. (256x256x256=16,777,216).

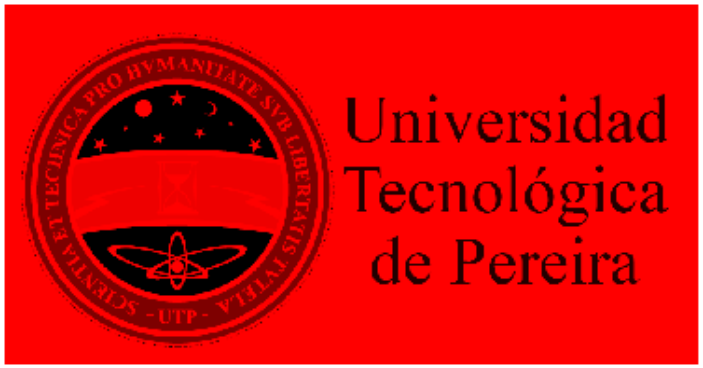


Canales RGB

```python
#   Debe instalar la libreria scikit-image de python
#   https://scikit-image.org/docs/dev/install.html
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['image.cmap'] = 'gray'
from skimage import io
image_rgb=io.imread("lena.jpg")/255.0
plt.subplot(221)
plt.imshow(image_rgb)
plt.title("imagen original")
plt.subplot(222)
plt.imshow(image_rgb[:,:,0])
plt.title("Canal Rojo")
plt.subplot(223)
plt.imshow(image_rgb[:,:,1])
plt.title("Canal Verde")
plt.subplot(224)
plt.imshow(image_rgb[:,:,2])
plt.title("Canal Azul")
plt.show()

print("- Dimensiones de la imagen:")
print(image_rgb.shape)
```



Descomposición RGB



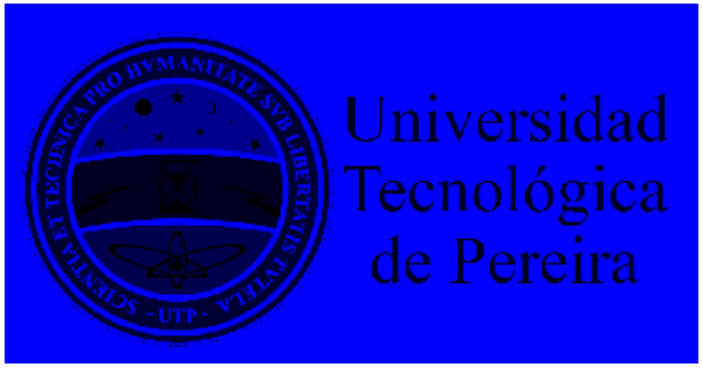
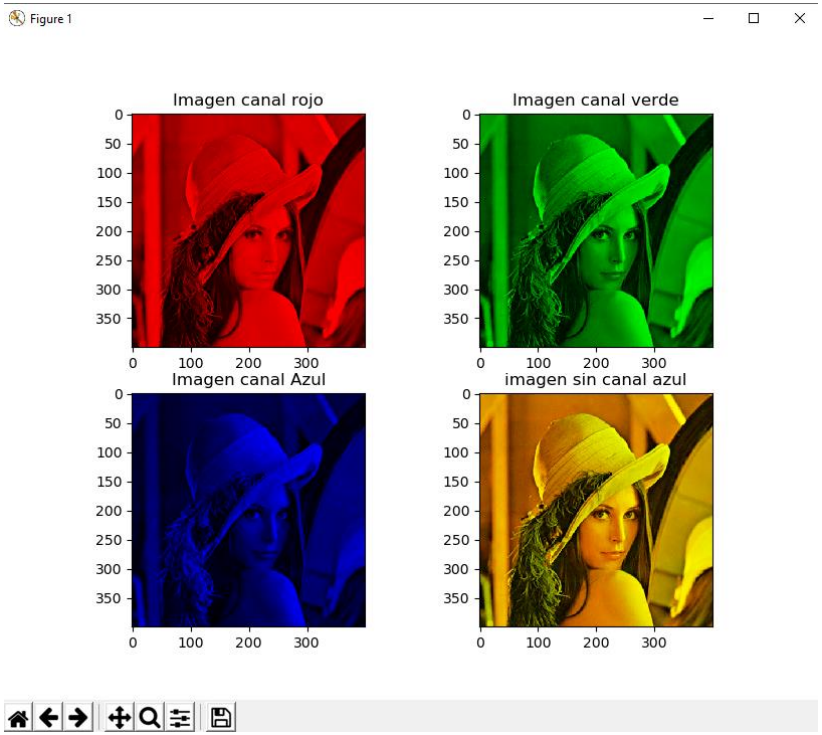


Imagen en los canales RGB

```python
import matplotlib.pyplot as plt
import numpy as np
#   Debe instalar la libreria scikit-image de python
#   https://scikit-image.org/docs/dev/install.html
from skimage import io

image_rgb=io.imread("lena.jpg")/255.0
print("- Dimensiones de la imagen:")
print(image_rgb.shape)
# creo una copia de la imagen para preservar la original
image_red=np.copy(image_rgb)
image_red[:,:,1]=0
image_red[:,:,2]=0
print(image_red)
plt.figure()
plt.subplot(221)
plt.title("Imagen canal rojo")
plt.imshow(image_red)
```

Imagen en los canales RGB

```
19
20  image_green=np.copy(image_rgb)
21  image_green[:,:,0]=0
22  image_green[:,:,2]=0
23  plt.subplot(222)
24  plt.title("Imagen canal verde")
25  plt.imshow(image_green)
26
27  image_blue=np.copy(image_rgb)
28  image_blue[:,:,0]=0
29  image_blue[:,:,1]=0
30  plt.subplot(223)
31  plt.title("Imagen canal Azul")
32  plt.imshow(image_blue)
33
34  image_red_green=np.copy(image_rgb)
35  image_red_green[:,:,2]=0
36  plt.subplot(224)
37  plt.title("imagen sin canal azul")
38  plt.imshow(image_red_green)
39
40  plt.show()
```



Descomposición CMYK

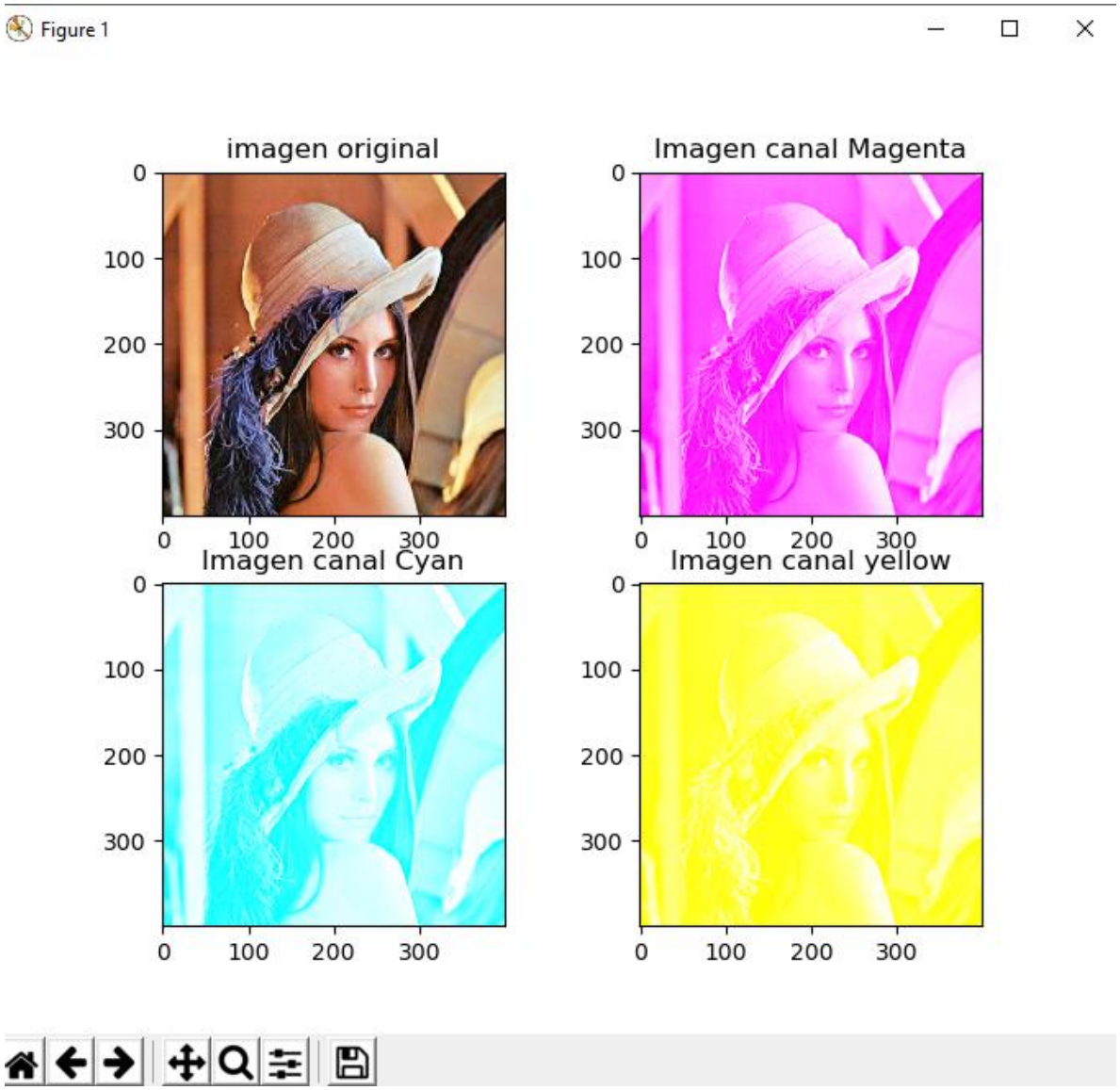
Imagen en los canales CMYK

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  #   Debe instalar la libreria scikit-image de python
4  #   https://scikit-image.org/docs/dev/install.html
5  from skimage import io
6
7  factor=255
8  image_mcyk=io.imread("lena.jpg")/255.0
9  print("- Dimensiones de la imagen:")
10 print(image_mcyk.shape)
11 filas=image_mcyk.shape[0]
12 columnas=image_mcyk.shape[1]
13 print (filas)
14 print (columnas)
15
16 plt.figure()
17 plt.subplot(221)
18 plt.imshow(image_mcyk,vmin=0,vmax=1)
19 plt.title("imagen original")
20
```

Imagen en los canales CMYK

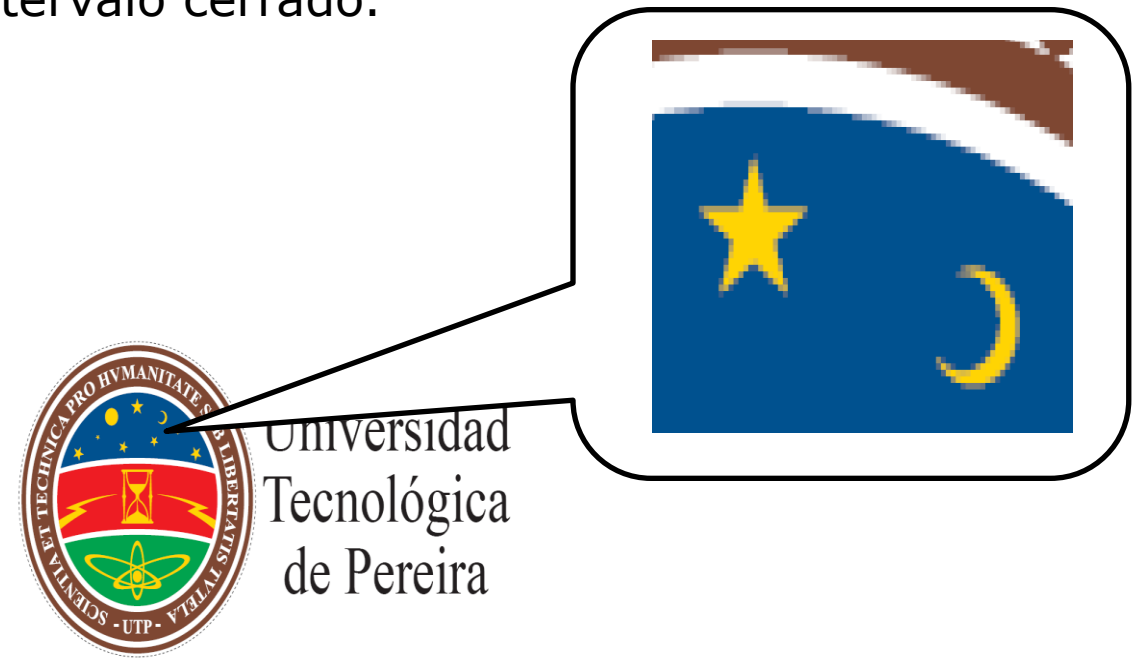
```
21  image_magenta=np.copy(image_mcyk)
22  image_magenta[:,:,0]=np.ones((filas,columnas))*factor
23  image_magenta[:,:,2]=np.ones((filas,columnas))*factor
24  plt.subplot(222)
25  plt.title("Imagen canal Magenta")
26  plt.imshow(image_magenta)
27
28  image_cyan=np.copy(image_mcyk)
29  image_cyan[:,:,1]=np.ones((filas,columnas))*factor
30  image_cyan[:,:,2]=np.ones((filas,columnas))*factor
31  plt.subplot(223)
32  plt.title("Imagen canal Cyan")
33  plt.imshow(image_cyan)
34
35  image_yellow=np.copy(image_mcyk)
36  image_yellow[:,:,0]=np.ones((filas,columnas))*factor
37  image_yellow[:,:,1]=np.ones((filas,columnas))*factor
38  plt.subplot(224)
39  plt.title("Imagen canal yellow")
40  plt.imshow(image_yellow)
41  plt.show()
```

Imagen en los canales CMYK



Representación de Imágenes

Una imagen se puede representar una imagen (RGB) formaremos una matriz de dimensiones m x n x q, con elementos vectores en donde cada vector estará compuesto o constituido por 3 componentes (canales RGB), con valores contenidos en los enteros de 0 a 255 en un intervalo cerrado.



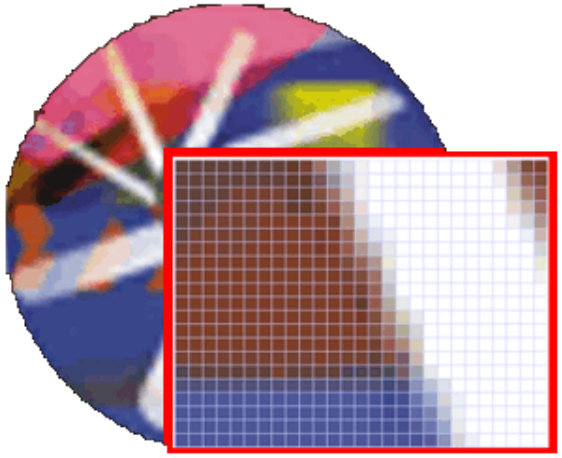
Universidad Tecnológica de Pereira

Representación en formato de mapa de bits

El píxel (picture element), es el elemento básico de la imagen. Los colores se obtienen en un conjunto basado en los colores primarios rojo, verde, y azul (RGB).

- Ofrecen bordes dentados.
- Resolución pobre.
- Tratan la información de texto como datos de imagen.

- Es imposible escalar la imagen sin distorsión.
- La resolución está dada por el número de píxeles por pulgada (ppi) o puntos por pulgada (dpi).



Formato RAW

- Es un formato que contiene la totalidad de los datos de la imagen captada por el sensor digital de la cámara.
- Es un formato de cada fabricante con iniciativas como OpenRAW (o DNG) que trata de homogeneizar el formato que no han triunfado.
- Almacena al menos 8 bits por color (rojo,verde y azul) aunque la mayoría de las cámaras réflex digitales almacenan 12 bits por color.
- Siempre se puede retornar al original, y no sufre ningún tipo de pérdida de calidad en su manipulación.

Formato TIFF (Tagged Image File Format)

- Es un formato de almacenamiento sin pérdidas de alta calidad.
- Son archivos muy pesados, pero ofrecen algoritmos de compresión sin pérdidas que consiguen reducir su nivel de espacio.

Formato JPG (Joint Photographers Experts Group)

- Es el formato más popular tanto en Internet como para imprimir.
- Es un formato con compresión con pérdida de calidad, pero los archivos finales ocupan muy poco espacio.
- Utiliza 8 bits por color.

- El grado de compresión se puede elegir, pero agrega artefactos. La profundidad de color normalmente es de 8 bits.

Formato GIF (*Graphic Interchange Format*)

- Es un formato de "Mapa de Bits" en el que en cada imagen, hay una tabla que indica los colores que le representan en la imagen.
- GIF tiene una profundidad de color de 8 bits, de 256 colores
- Usa un sistema de compresión sin pérdida llamado LZW (*Lemple - Zif - Wellch*, el mismo que utiliza el ZIP).
- LZW está patentado, por lo que todos los programas editores de software que usaban imágenes GIF deben pagarle regalías a Unisys, la compañía propietaria de los derechos
- Fue desarrollado por *Compuserve.*

Formato PNG (*Portable Network Format*)

- Aparece como respuesta a los problemas del formato GIF (los legales y técnicos).
- Este es un formato totalmente libre, de tal forma que cualquiera puede implementarlo en sus programas o usarlo libremente sin pagar derechos a nadie.

JPG format

PNG format

GIF format

TIF format

BMP format

RAW format

Suma y Resta de Matrices

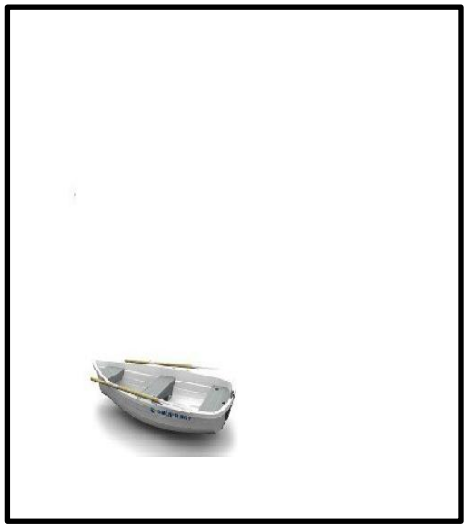
Consiste en sumar o restar imágenes unas con otras:

$$IMC(i,j)=IMA(i,j)\pm IMB(i,j)$$

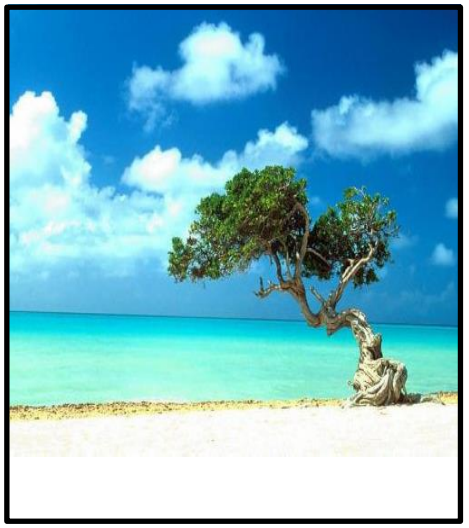
Muy utilizadas cuando se busca el movimiento de un objeto dentro de una escena. Se comparan dos imágenes correlativas en el tiempo mediante la substracción de una de ellas con la otra. Si algún objeto se ha movido se detecta inmediatamente porque aparecerá en la imagen resultante. Si la escena está completamente quieta, la imagen resultante será cero.

Ejemplo Suma y Resta de Matrices

Bote



Mar



```
R=Mar*factor+(1-
factor)*Bote;
```

Ejemplo Suma y Resta de Matrices

Imágenes Fusionadas

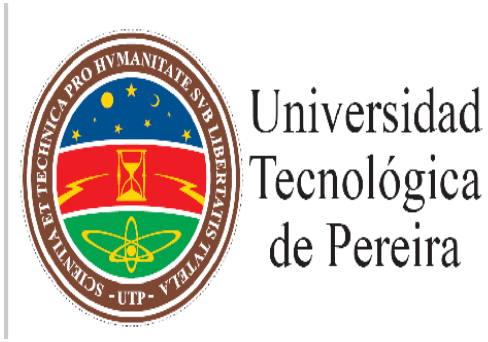
Ejemplo Suma y Resta de Matrices

Imágenes Fusionadas y Ecualizadas*

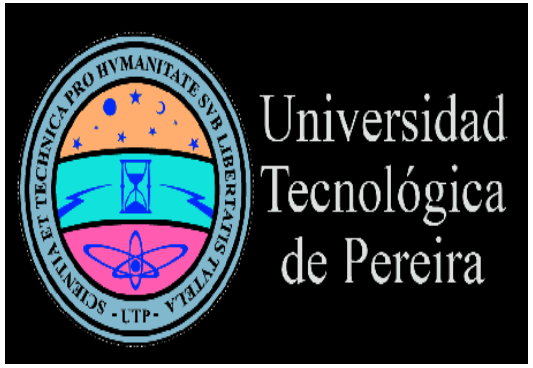
Inversión Color Matlab

Esta operación consiste en invertir cada canal RGB a su negativo. Por ejemplo, en una película fotográfica en la cual se plasman los colores invertidos de la imagen real. Esto es que el blanco pasa a ser negro, el azul a amarillo, verde a magenta y rojo a cyan.
La utilidad de este filtro, se encuentra en la digitalización de películas fotográficas.

$I_{invertida}=255-I_{original}$



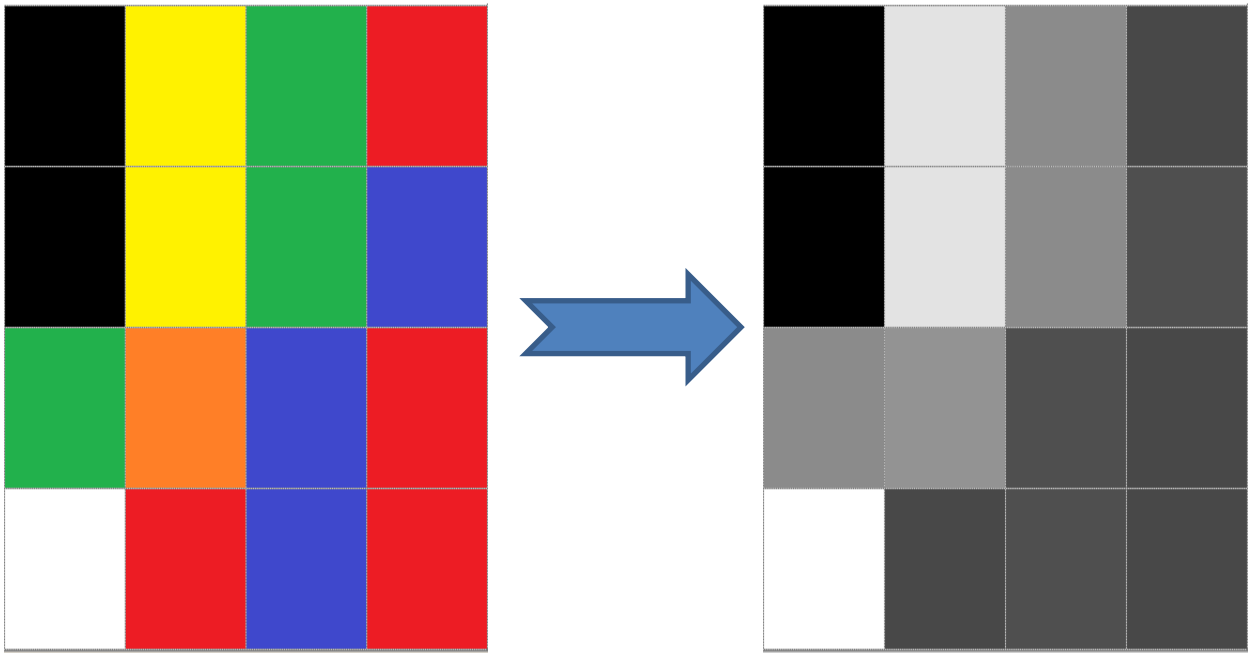
$I_{original}$



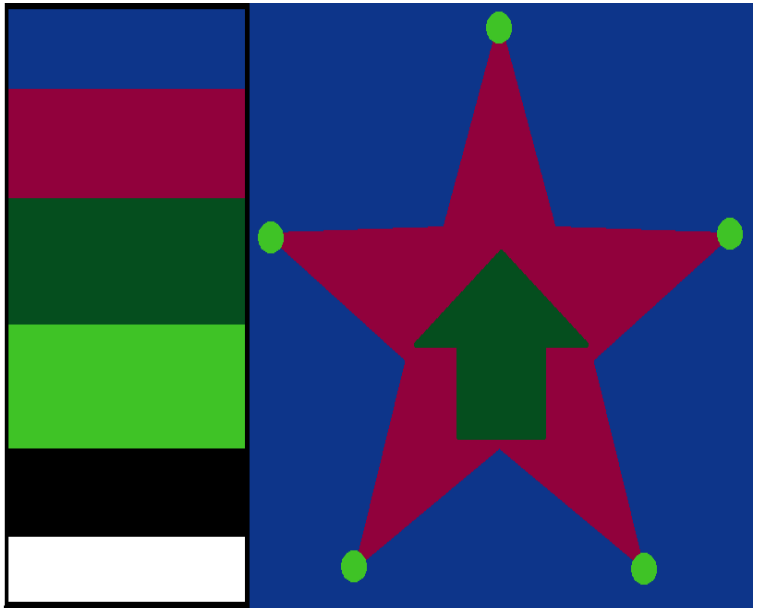
$I_{invertida}$

Escala de Grises

Una imagen en escala de grises es un arreglo matricial de dos dimensiones que aporta información de la intensidad de la luz presente para cada punto de la imagen.

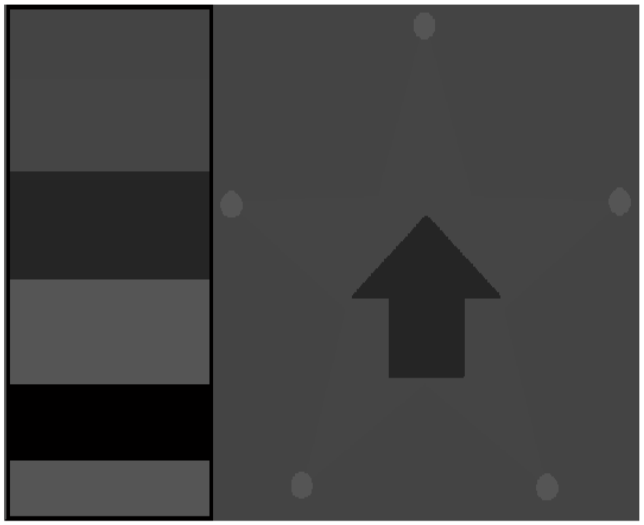


Conversión a escala de Grises (prueba)



Técnica de promedio (Average)

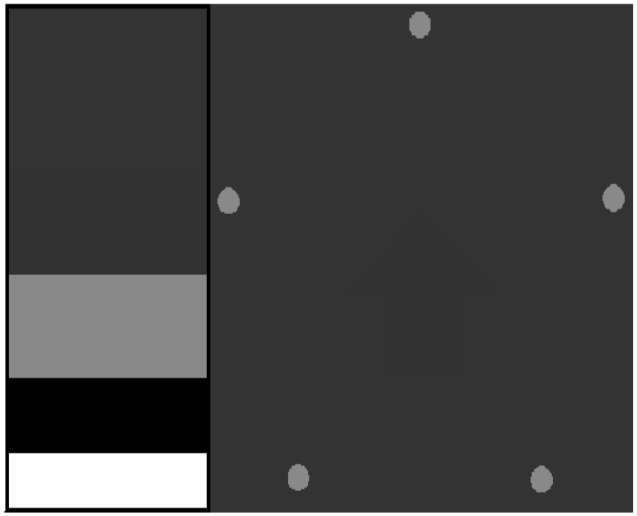
La forma más simple de lograrlo es mediante la suma de las componentes RGB de cada capa pixel a pixel y dividirles por la cantidad (3).



$$Grey(i,j) = \frac{(R(i,j)+G(i,j)+B(i,j))}{3}$$

Técnica de Luminosidad (Luminosity)

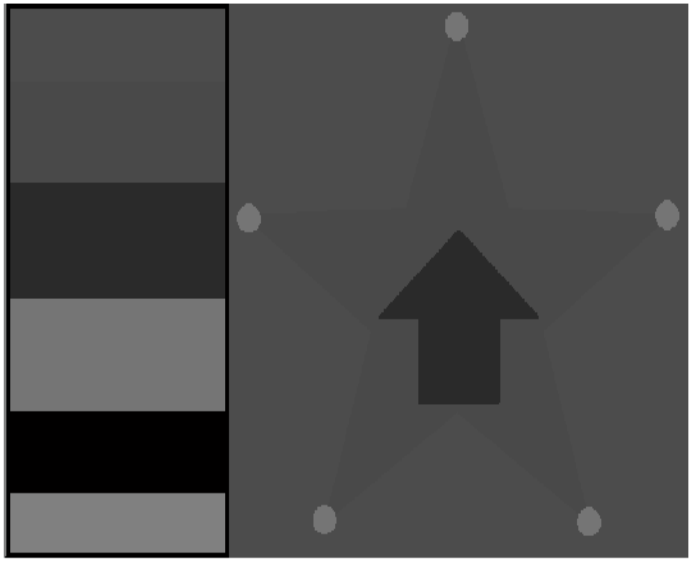
Este método es una versión más sofisticada del método del promedio. Los valores de cada color presentan un valor teniendo en cuenta la percepción humana. El ojo humano es más sensible al verde que a los otros colores, por lo que el porcentaje de este es mayor. Estos valores fueron establecidos a través de la recomendación *Rec. 601 NTSC* por la *International Telecommunication Union – Radiocommunications;* sector que le ha hecho un estándar mundial para la televisión a color con compatibilidad a blanco y negro.



$$Grey(i,j) = 0.299 * R(i,j) + 0.587 * G(i,j) + 0.114 * B(i,j)$$

Técnica de La tonalidad (Midgray)

En el modelo HSL (del inglés *Hue*, *Saturation*, *Lightness* con su equivalencia en español Tonalidad, Saturación, Luminancia o intensidad) la tonalidad se define como parte de la media de las componentes de color máxima y mínima.



$$Grey(i,j) = \frac{(Max[R(i,j);G(i,j),B(i,j)]+Min([R(i,j),G(i,j),B(i,j)])}{2}$$

Conversión de color a escala de grises

- **Forma 1:** Esta conversión se realiza calculando un equivalente “E” formado a partir de los tres planos de la imagen a color. En su forma mas sencilla se establece este equivalente como el promedio, es decir:

$$E(x,y) = \frac{R(x,y) + G(x,y) + B(x,y)}{3}$$

Conversión de color a escala de grises

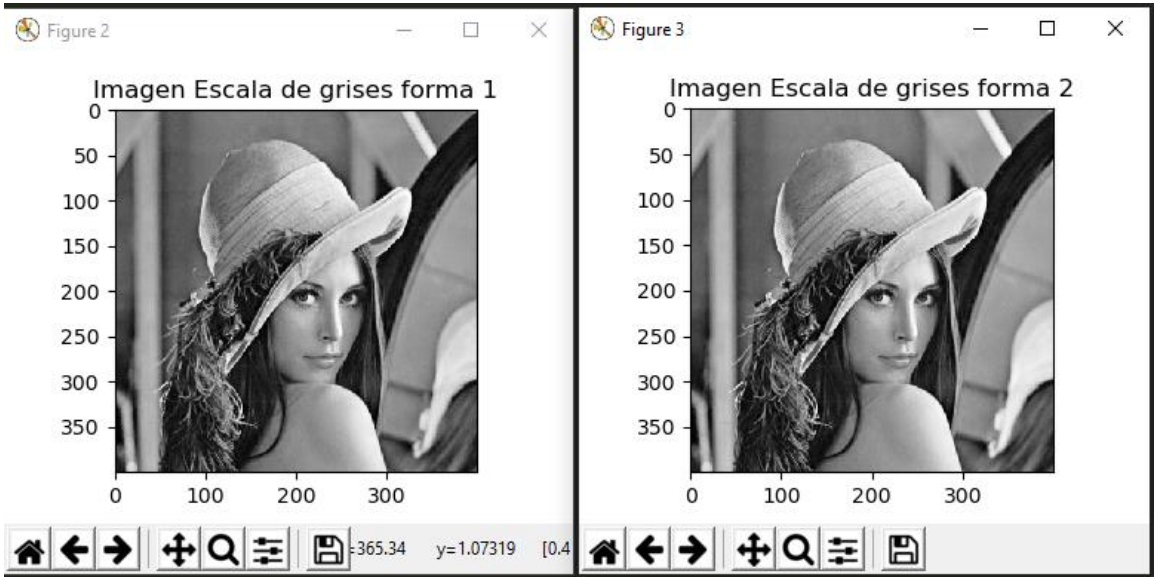
- **Forma 2:** La subjetiva iluminación, propia del modelo RGB hace que utilizando el promedio las imágenes con un valor grande en la componente de rojo y/o verde tengan una apariencia obscura. El efecto contrario sucede donde el contenido del plano azul es grande, mostrando en su versión a escala de grises una apariencia muy clara. Con el objetivo de solventar esto se considera como una mejor aproximación calcular una combinación lineal de todos los planos, definida como:

$$E_{lin}(x, y) = W_R R(x, y) + W_G G(x, y) + W_B B(x, y)$$

Conversión de color a escala de grises

Donde WR, WG y WB son los coeficientes que definen la transformación, los cuales de acuerdo al criterio utilizado en la TV para señales a color se consideran como:

WR=0.299 WG=0.587 WB=0.114



Escala de grises

```
2  import matplotlib.pyplot as plt
3  import numpy as np
4  from skimage import io
5  plt.rcParams['image.cmap'] = 'gray'
6  image=io.imread("lena.jpg")/255.0
7  image_gris=np.copy(image)
8  print("- Dimensiones de la imagen:")
9  print(image_gris.shape)
10 print("Datos del vector" , image_gris )
11 plt.figure()
12 plt.title("Imagen Original")
13 plt.imshow(image_gris )
14 # Forma 1 ------------------------------------
15 R=image_gris[:,:,0]
16 print("Vector R" , R )
17 G=image_gris[:,:,1]
18 print("Vector G" , G )
19 B=image_gris[:,:,2]
20 print("Vector B" , R )
21 image_gris=((R+G+B)/3)
22 plt.figure()
23 plt.title("Imagen Escala de grises forma 1")
24 plt.imshow(image_gris)
```

Escala de grises

```
26  # Forma 2 ---------------------------------------
27  image_gris=np.copy(image)
28  R=image_gris[:,:,0]
29  print("Vector R" , R )
30  G=image_gris[:,:,1]
31  print("Vector G" , G )
32  B=image_gris[:,:,2]
33  print("Vector B" , R )
34  image_gris=0.2989 *  R + 0.5870 * G + 0.1140 * B
35  plt.figure()
36  plt.title("Imagen Escala de grises forma 2")
37  plt.imshow(image_gris)
38
39  plt.show()
```