INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

“Practica 6-Conversion de imágenes a otros modelos de color”

-Tinoco Videgaray Sergio Ernesto

Grupo: 4BV1

Materia: Procesamiento digital de imágenes

12/05/22

En el mundo de las imágenes a color existen distintos modelos de colores que nos permiten representar y mapear el comportamiento de los colores dentro de imágenes.

Un ejemplo de estos modelos son el modelo RGB que se utiliza universalmente en el mundo digital para representar los 3 colores primarios: Rojo, Verde y Azul por sus siglas en inglés (Red, Green, Blue).

De igual forma existe el modelo CMY (Cian, Magenta, Yellow) el cual nos muestra otra forma de representar los colores por medio de estos 3, dicho modelo es muy utilizado en el mundo de la imprenta.

Finalmente tenemos el modelo HSI (Hue, Saturation, Intensity) que nos permite representar los colores en 3 distintos módulos: Matiz, Saturación e Intensidad, muy similar a la forma en la que el ojo humano captura los colores. Dicho modelo es comúnmente utilizado en el mundo de los programas de diseño.

En esta practica se van a desarrollar 2 conversiones de imágenes del modelo RGB a los anteriormente mencionados.

- Desarrollo:

Para el algoritmo de esta práctica se va a trabajar con el lenguaje de programación Python.

Imagen a convertir:

Primeramente, se van a importar las bibliotecas correspondientes:

Se incorpora la librería SQRT para trabajar con radicales.

Se importa la librería Math para funciones trigonométricas.

Se importa el paquete PIL para trabajar con imágenes.

De igual forma se importa el paquete Numpy para trabajar con matrices.

Y finalmente la biblioteca matplot para los gráficos resultantes.

```
from cmath import sqrt
import math
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
```

Al iniciar el programa se carga la imagen original dentro de un objeto de tipo imagen.
Se generan 2 objetos, en el primero se carga la imagen en niveles de gris y en el segundo se carga la imagen en una matriz.

```
img=Image.open("peces.jpg")   #Se carga la imagen original
imgGray=img.convert("L")    #Se convierte a escala de grises
gray=np.asarray(imgGray)    #Se genera una matriz con la imagen en
niveles de gris
```

Se extraen los 3 canales RGB en 3 variables distintas de tipo imagen.

```
r,g,b=img.split()
```

Se forman 3 matrices con esos niveles de RGB.

```
r=np.asarray(r)
g=np.asarray(g)
b=np.asarray(b)
```

Se obtienen las dimensiones de la imagen por medio de las matrices.

```
alto=r.shape[0]
ancho=r.shape[1]
```

- Conversión al modelo CMY:

Se define una función convertirCMY que recibe como parámetros los valores RGB de la imagen, así como las dimensiones de la misma:
En dicha función se recorren las 3 matrices RGB:
En cada valor de las matrices se va a obtener el valor complemento de esa casilla, es decir el valor negativo de la matriz 255-r.
Finalmente se retornan las matrices convertidas al modelo CMY.

```
def convertirCMY(r,g,b,alto,ancho):
    i=0
    while(i<alto):
        j=0
        while(j<ancho):
```

```
            r[i][j]=255-(r[i][j])
            g[i][j]=255-(g[i][j])
            b[i][j]=255-(b[i][j])
            j+=1
        i+=1
    return r,g,b
```

Se invoca a la función para convertir a CMY mandando como parametros 3 copias de las matrices RGB y se almacena el resultado en 3 variables C, M, Y.

```
c,m,y=convertirCMY(np.copy(r),np.copy(g),np.copy(b),alto,ancho)
```

Posteriormente se crea el primer grafico para el modelo CMY:

```
fig=plt.figure(figsize=(15,18))  #Tamaño de cada subgrafica
fig.set_size_inches(12, 8)  #Tamaño de la grafica en pulgadas
```

Se añaden los 3 canales CMY en un subgrafico.

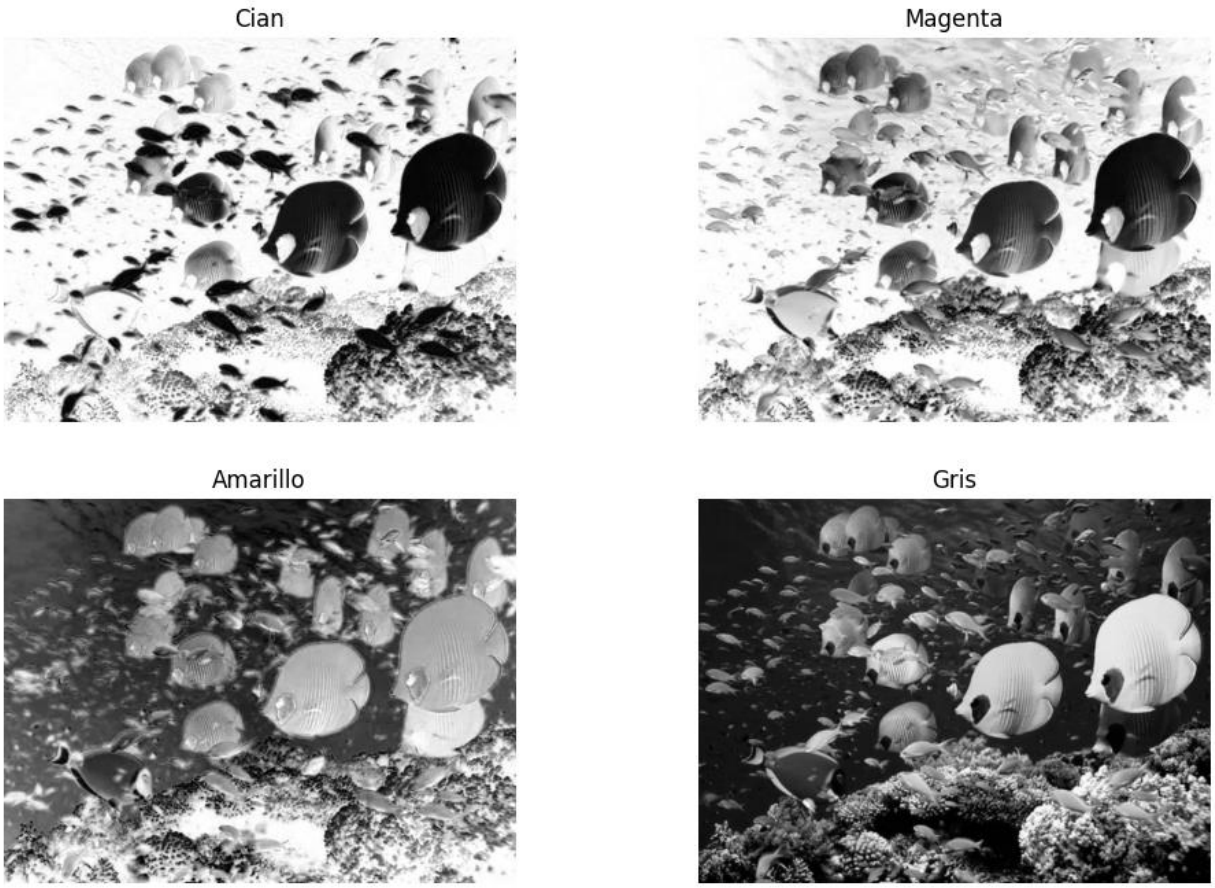
```
#Añado el subgrafo del canal Cian
fig.add_subplot(2,2,1)
plt.imshow(c,cmap="gray")
plt.title("Cian")
plt.axis("off")

#Añado el subgrafo del canal Magenta
fig.add_subplot(2,2,2)
plt.imshow(m,cmap="gray")
plt.title("Magenta")
plt.axis("off")

#Añado el subgrafo del canal Amarillo
fig.add_subplot(2,2,3)
plt.imshow(y,cmap="gray")
plt.title("Amarillo")
plt.axis("off")

#Añado el subgrafo del canal Gray
fig.add_subplot(2,2,4)
```

```
plt.imshow(gray,cmap="gray")
plt.title("Gris")
plt.axis("off")
```

Finalmente se muestra el grafico resultante con los 4 subgrafos.

```
plt.show()
```

Resultados de la primera conversión:



- Conversión a modelo HSI:

Primeramente, se define una función convertirHSI que recibe como parámetros los niveles RGB normalizados y las dimensiones de la imagen original.
Se obtiene el valor de theta aplicando una función de arco coseno de la variable argumento para posteriormente convertir el valor a grados.

En caso de que el valor B sea menor o igual que el valor de G se asigna el valor de theta directamente a la matriz. Y en caso contrario se obtiene el complemento y se asigna a la matriz.

Para la saturación se resta a 1 la media armónica normalizada y se multiplica por el valor mínimo de R, G, B y se asigna a la matriz de G.
Para el caso de la intensidad se obtiene la media aritmética de los valores RGB y se asignan a la matriz de B.

Finalmente se retornan las matrices R, G, B con la conversión al modelo HSI.

```
def convertirHSI(r,g,b,alto,ancho):
    i=0
    while(i<alto):
        j=0
        while(j<ancho):
            R=r[i][j]
            G=g[i][j]
            B=b[i][j]

         arg=((1/2)*((R-G)+(R-B)))/(sqrt((pow((R-G),2))+(R-B)*(G-B)))
            theta=math.acos(arg)
            theta*=(180/math.pi)        #Conversion Radianes a Grados
            if(B<=G):
                r[i][j]=theta/360       #Matiz
            else:
                r[i][j]=(360-theta)/360

            g[i][j]=1-((3/(R+G+B)))*(min([R,G,B]))   #Saturacion

            b[i][j]=(R+G+B)/3   #Intensidad

            j+=1
        i+=1
    return np.trunc(r*255),np.trunc(g*255),np.trunc(b*255)
```

Se invoca a la función para convertir a HSI mandándole como parámetros los canales RGB normalizados y las dimensiones de la imagen cargada:

Se cargan los valores en 3 variables H, S, I:

```
h,s,i=convertirHSI(np.copy(r)/255,np.copy(g)/255,np.copy(b)/255,alto,
ancho)
```

Se declara un objeto de tipo figure donde se definen los tamaños de cara subgrafica:

```
fig=plt.figure(figsize=(15,18))  #Tamaño de cada subgrafica
fig.set_size_inches(12, 8)  #Tamaño de la grafica en pulgadas
```

Se añaden las subgraficas al objeto figure:

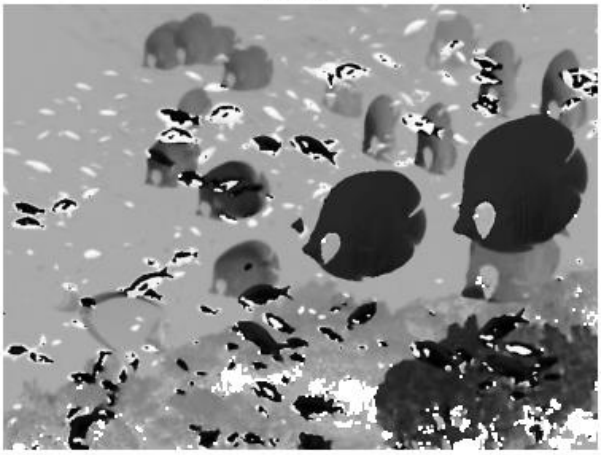
```
#Añado el subgrafo del canal Red
fig.add_subplot(2,2,1)
plt.imshow(h,cmap="gray")
plt.title("Matiz")
plt.axis("off")

#Añado el subgrafo del canal Green
fig.add_subplot(2,2,2)
plt.imshow(s,cmap="gray")
plt.title("Saturacion")
plt.axis("off")

#Añado el subgrafo del canal Blue
fig.add_subplot(2,2,3)
plt.imshow(i,cmap="gray")
plt.title("Intensidad")
plt.axis("off")

#Añado el subgrafo del canal Gray
fig.add_subplot(2,2,4)
plt.imshow(gray,cmap="gray")
plt.title("Gris")
plt.axis("off")
```

Finalmente se muestra el grafico resultante con los 4 subgrafos.

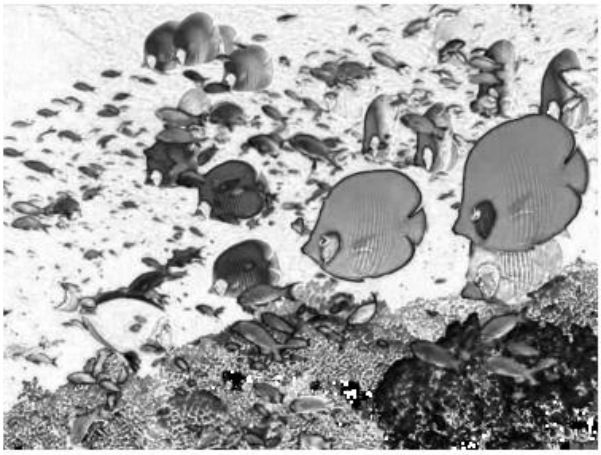
```
plt.show()
```

Resultados de la conversión:

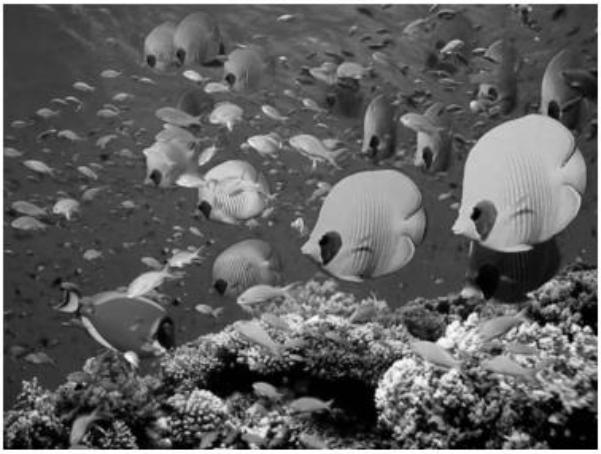
Matiz



Saturacion



Intensidad



Gris