Universidad Autónoma de San Luis Potosí
Facultad de ingeniería
Tratamiento de Imágenes
Practica 3
Nombre Práctica: Espacios de Color y Segmentación de Color
Nombre del Alumno: Manuel Ramírez Galván

Fecha: 11/02/2025

Procedimiento

3.1. Diseñe un programa en Python en el cual se cumplan los siguientes requisitos:

- Cargue una imagen del disco duro.
- Realice una conversión a un espacio de color diferente a BGR.
- La imagen original y su conversión deben de ser mostradas en ventanas distintas, cada una con su respectivo nombre.
- Las ventanas serán mostradas hasta que se presione le tecla q

Resultados

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

img_rgb = cv2.imread("C:/Users/HUAWEI/Desktop/L TRATAMIENTO/Manual/Practica 3 -

img_cmy = 1- (img_rgb/255)

mostrar = True
while mostrar:
    cv2.imshow("Imagen RGB", img_rgb)
    cv2.imshow("Imagen CMY", img_cmy)
    k = cv2.waitKey(0)
    if k == ord('q'):
        mostrar = False
        cv2.destroyAllWindows()

```

Imagen 1.- Código Ejercicio 1

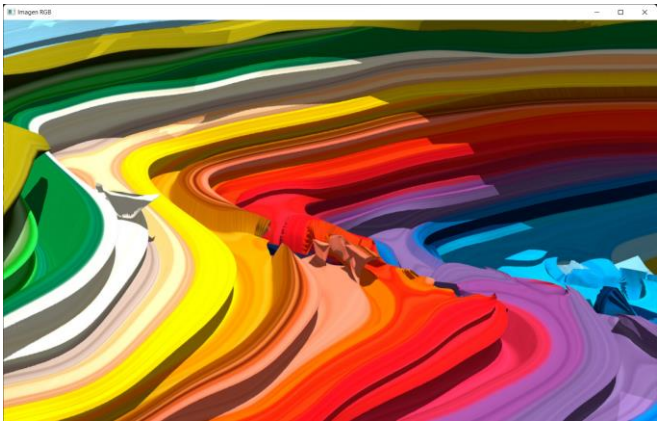

Imagen 2.- Imagen RGB

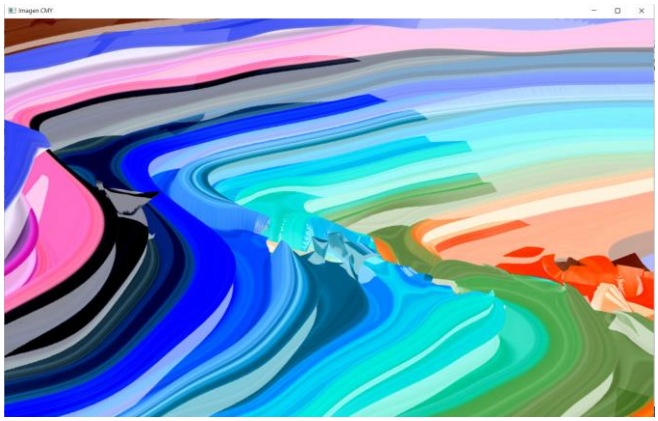

Imagen 3.- Imagen CMY

Procedimiento

3.2. Diseñe un programa en Python en el cual se cumplan los siguientes requisitos:

- Cargue una imagen del disco duro.
- Realice una conversión a un espacio de color diferente a BGR.
- Dividir la imagen convertida en sus respectivos canales.
- Aplicar una operación de mejora de la imagen (suma, resta, etc.) a un canal de la imagen convertida.
- Converger los canales en una sola imagen multicanal.
- Mostrar las diferencias entre la imagen convertida y la imagen mejorada en la misma ventana.

Resultados

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

img_rgb = cv2.imread("C:/Users/HUAWEI/Desktop/L TRATAMIENTO/Manual/Practica 3 - Espacios de color

img_hsv = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2HSV)

h, s, v = cv2.split(img_hsv)

h = cv2.add(v, 30)

img_hsv = cv2.merge([h, s, v])
img_new = cv2.cvtColor(img_hsv, cv2.COLOR_BGR2RGB)

mostrar = True
while mostrar:
    plt.figure(figsize=(1, 4))
    cv2.imshow("Imagen Original", img_rgb)
    cv2.imshow("Imagen Modificada", img_new)
    k = cv2.waitKey(1)
    if k == ord('q'):
        mostrar = False

cv2.destroyAllWindows()
```

Imagen 4.- Código Ejercicio 2

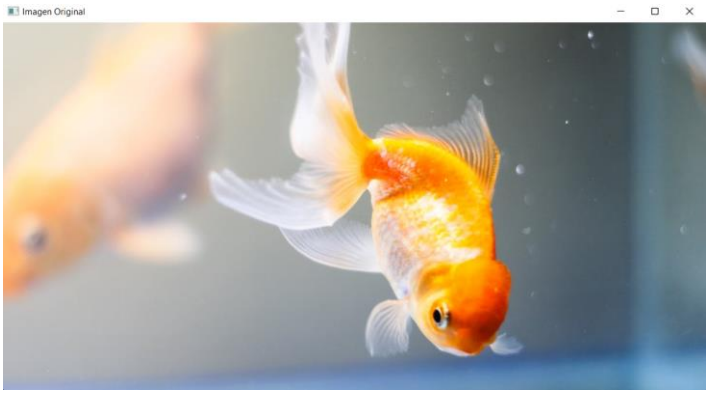

Imagen 5.- Imagen a Color

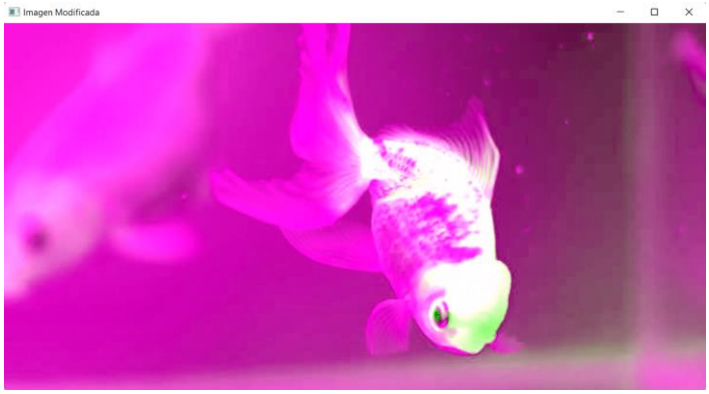

Imagen 6.- Imagen Convertida

Procedimiento

3.3. Diseñe un programa en Python en el cual se cumplan los siguientes requisitos:

- Cargue una imagen a color del disco duro.
- Use una función callback para realizar segmentación de color en base al color que se le esté haciendo click en.
- Determine los criterios de segmentación más óptimos para las imágenes.
- Se debe mostrar la imagen segmentada en su propia ventana.

Resultados

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os

tol = 7
showing = False

def mouseFunc(evento, x, y, flags, img):
    global showing
    if evento == cv2.EVENT_LBUTTONDOWN:
        hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        os.system("cls")
        mat, sat, val = hsv[y, x]
        print(f'[{mat}, {sat}, {val}]')
        if (mat - tol) >= 0 and (mat + tol) <= 255:
            lower = np.array((mat - tol, 10, 50), np.uint8)
            upper = np.array((mat + tol, 255, 255), np.uint8)
            mascara = cv2.inRange(hsv, lower, upper)
        else:
            lower = np.array((0, 10, 50), np.uint8)
            upper = np.array((0 + tol, 255, 255), np.uint8)
            bin_img = cv2.inRange(hsv, lower, upper)

            lower = np.array((255 - tol, 10, 50), np.uint8)
            upper = np.array((255 + tol, 255, 255), np.uint8)
            bin2_img = cv2.inRange(hsv, lower, upper)

            mascara = cv2.bitwise_or(bin_img, bin2_img)

        res = cv2.bitwise_and(img, img, mask = mascara)
        cv2.imshow("Segmentacion", res)
        showing = True
    if evento == cv2.EVENT_RBUTTONUP:
        if showing:
            os.system("cls")
            print("Imagen Original")
            cv2.destroyWindow("Segmentacion")

def main():
    img = cv2.imread("C:/Users/HUAWEI/Desktop/L TRATAMIENTO/Manual/P
    cv2.namedWindow('Colores')
    cv2.setMouseCallback('Colores', mouseFunc, img)

    while True:
        cv2.imshow('Colores', img)
        k = cv2.waitKey(1)
        if k == ord('q'):
            break
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

Imagen 7.- Código Ejercicio 3

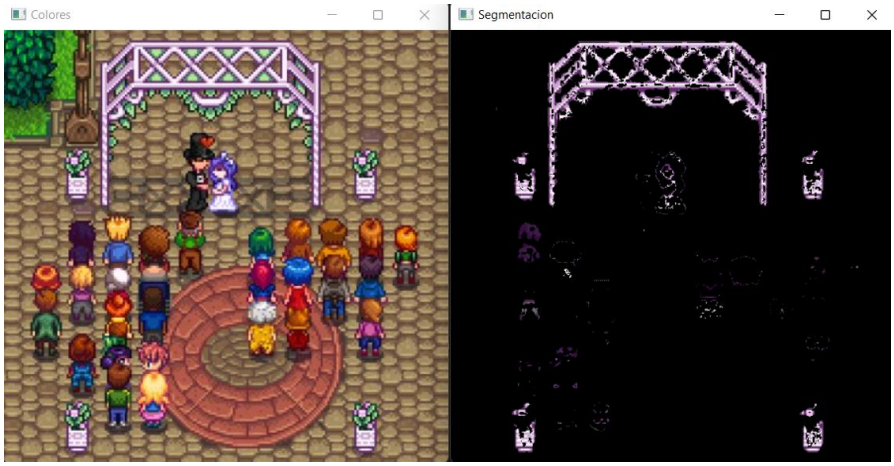

Imagen 8.- Imagen Original e Imagen Segmentada

Comprensión

1. **¿Qué es un modelo de color?**

Es la especificación de un sistema de coordenadas tridimensionales en el que cada color queda representado por un único punto. Los modelos de color facilitan la especificación de los colores de forma normalizada.

2. **¿Qué es el espacio Lab y a que se asemeja?**

Es un modelo de color diseñado para ser perceptualmente uniforme, para representar los colores de manera similar como los percibe el ojo humano. Es independiente de dispositivos como pantallas, impresoras o cámaras.

Tiene 3 componentes, L (Luminosidad), A (Canal Verde-Rojo), B (Canal Azul-Amarillo).

3. **¿Qué argumentos contiene la función setMouseCallback?**

Cv2.setMouseCallback(ventana, función, parámetros)

Ventana: Nombre de la ventana donde se capturan los eventos del mouse.

Función: Nombre de la función que maneja los eventos del mouse.

Parámetros: Parámetros adicionales que se pueden pasar a la función de manejo de eventos. En esta práctica, la imagen.

Conclusiones

Se pueden hacer la conversión de color entre espacios de colores, esto con formulas que hacen la conversión o con funciones que ya lo hacen.

La función split() sirve para separar los canales de una imagen y esto para poder hacer modificaciones a uno o mas de ellos para la mejora de la imagen.

La función merge() sirve para unir los canales separados y poder visualizar la imagen que posiblemente se modificó.

Se puede hacer que por medio de eventos, por ejemplo el click del mouse, se haga una modificación en la imagen, por ejemplo en esta práctica el seleccionar el color en una parte en específico de una imagen y visualizar un rango de colores cercano a este seleccionado, mostrando la imagen solo con el rango del color.