

06/05/2025

SEGUNDO PARCIAL

PRÁCTICA 6

Maestro:

Mauricio Alejandro Cabrera
Arellano

Alumno:

Orozco Dueñas Roberto Eduardo
6 - G

Ingeniería Mecatrónica

OBJETIVO:

Objetivo: Encontrar colores específicos o remover colores específicos. (Green Screen)
Hacer el filtro de rojo, verde y azul.

CÓDIGO:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # --- Rango para ROJO ---
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red, upper_red)
    res_red = cv2.bitwise_and(frame, frame, mask=mask_red)

    # --- Rango para VERDE ---
    lower_green = np.array([36, 50, 70])
    upper_green = np.array([89, 255, 255])
    mask_green = cv2.inRange(hsv, lower_green, upper_green)
    res_green = cv2.bitwise_and(frame, frame, mask=mask_green)

    # --- Rango para AZUL ---
    lower_blue = np.array([94, 80, 2])
    upper_blue = np.array([126, 255, 255])
    mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
    res_blue = cv2.bitwise_and(frame, frame, mask=mask_blue)

    # Mostrar todo
    cv2.imshow('Original', frame)
    cv2.imshow('Red', res_red)
    cv2.imshow('Green', res_green)
    cv2.imshow('Blue', res_blue)

    # Salir con ESC
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

COMENTARIOS:

En esta práctica al principio no tuve ningún problema, cuanto tuvo nomás que compilarse el color verde, ya después cuando necesite que se aplicara el rojo y azul tuve problemas para poder hacerlo, pero le pedí a chat que me ayudara y me diera una explicación del por qué de cada cosa, aquí está la explicación:

explicame el por que de cada cosa en el filtro rojo y azul

ChatGPT Plus

¡Perfecto! Vamos a **explicar paso a paso** cómo funcionan los **filtros de color rojo y azul** en el código, usando el espacio de color **HSV** (Hue, Saturation, Value), que es mucho mejor para detectar colores que el tradicional BGR.

¿Por qué usar HSV?

El espacio **HSV** separa la información del color (tono) de la iluminación, lo que facilita la detección de colores sin que nos afecte tanto la luz o las sombras.

- **Hue (H)** = matiz o tono del color (0 a 179 en OpenCV).
 - Rojo \approx 0 o 179 (¡está en los extremos!)
 - Verde \approx 60
 - Azul \approx 120
 - **Saturation (S)** = qué tan puro o intenso es el color (0 a 255).
 - **Value (V)** = qué tan brillante es el color (0 a 255).
-

Filtro para Rojo

python

CopiarEditar

```
lower_red = np.array([0, 120, 70])  
upper_red = np.array([10, 255, 255])
```

¿Qué significa?

- **Hue** entre **0 y 10** → detecta el rojo en el extremo inicial del círculo HSV.

- **Saturation** entre **120 y 255** → solo queremos rojos bien marcados, no colores grisáceos o muy pálidos.
- **Value** entre **70 y 255** → ignoramos rojos muy oscuros.

Nota: El color rojo es especial porque **aparece dos veces en el círculo HSV** (en 0 y en 179), por eso para un filtro completo de rojo se usan **dos rangos** normalmente (este es solo uno de ellos). Ejemplo para detectar todo el rojo:

```
python
CopiarEditar
# Rojo claro
lower_red1 = np.array([0, 120, 70])
upper_red1 = np.array([10, 255, 255])

# Rojo oscuro
lower_red2 = np.array([170, 120, 70])
upper_red2 = np.array([180, 255, 255])
```

Y luego combinas las dos máscaras.

Filtro para Azul

```
python
CopiarEditar
lower_blue = np.array([94, 80, 2])
upper_blue = np.array([126, 255, 255])
```

¿Qué significa?

- **Hue** entre **94 y 126** → abarca los tonos de azul en HSV.
 - **Saturation** desde **80** → evita los azules muy pálidos o grises.
 - **Value** desde **2** → acepta incluso tonos muy oscuros de azul.
-

¿Qué hace **cv2.inRange(hsv, lower, upper)**?

Crea una **máscara**: una imagen en blanco y negro donde:

- **Blanco (255)** = píxeles que están **dentro del rango de color** (lo que queremos).
- **Negro (0)** = píxeles fuera del rango (lo que ignoramos).

🎯 ¿Qué hace `cv2.bitwise_and(frame, frame, mask=...)`?

Aplica la **máscara al cuadro original**, mostrando solo **la parte de la imagen donde el color coincide**.

🔄 Resultado final

- **Filtro rojo** → detecta objetos de color rojo, como una playera, una flor, etc.
- **Filtro azul** → detecta objetos azules, como una pluma, una tapa, un folder, etc.

RESULTADOS:



