CENTRO DE ENSEÑANZA TECNICA INDUSTRIAL

Practica 006

Filtros de color HSV – RGB – YUV en video

Gonzalez Yañez Leonardo Isaac

29/04/2025

Objetivo

Aplicar filtros de color utilizando los espacios de color **HSV, RGB y YUV** para **detectar o eliminar colores específicos** (como rojo, verde y azul) en tiempo real mediante captura de video. Esta técnica es fundamental en aplicaciones como *Green Screen*, segmentación de objetos o seguimiento de color.

Fundamento teórico

Espacios de color:

- **RGB** (Red, Green, Blue): es el modelo estándar en imágenes digitales, pero sensible a la iluminación.
- **HSV** (Hue, Saturation, Value): separa la información de tono (color) de la intensidad, facilitando la detección de colores específicos incluso con variaciones de luz.
- **YUV**: separa luminancia (Y) y crominancia (U, V), útil en compresión y transmisión de video.

Filtros:

Se utilizan **máscaras** generadas con `cv2.inRange()` para aislar ciertos rangos de color. Luego se aplica esa máscara a la imagen original usando `cv2.bitwise_and()` para visualizar solamente el color filtrado.

Desarrollo / Código utilizado (HSV)

```
python
CopiarEditar
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Filtro rojo
    lower_red1 = np.array([0, 100, 100])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([160, 100, 100])
    upper_red2 = np.array([179, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red1, upper_red1) | cv2.inRange(hsv,
lower_red2, upper_red2)

    # Filtro verde
    lower_green = np.array([35, 100, 100])
    upper_green = np.array([85, 255, 255])
```

```
    mask_green = cv2.inRange(hsv, lower_green, upper_green)

    # Filtro azul
    lower_blue = np.array([100, 100, 100])
    upper_blue = np.array([130, 255, 255])
    mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

    # Aplicar máscaras
    red_result = cv2.bitwise_and(frame, frame, mask=mask_red)
    green_result = cv2.bitwise_and(frame, frame, mask=mask_green)
    blue_result = cv2.bitwise_and(frame, frame, mask=mask_blue)

    # Mostrar ventanas
    cv2.imshow('Original', frame)
    cv2.imshow('Rojo', red_result)
    cv2.imshow('Verde', green_result)
    cv2.imshow('Azul', blue_result)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

Resultados esperados

- Se muestra en tiempo real la imagen original y tres ventanas más:
  - Una con el color **rojo** detectado.
  - Una con el color **verde** detectado.
  - Una con el color **azul** detectado.
- Se pueden usar objetos físicos con esos colores para verificar si el filtro funciona.

Conclusiones

- El uso del espacio **HSV** facilita la segmentación de colores sin depender tanto de la iluminación como en RGB.
- Esta técnica es clave en aplicaciones como eliminación de fondo (*chroma key*), rastreo de objetos por color y procesamiento de imagen en visión artificial.
- Al aprender a definir rangos adecuados de color, se logra un filtrado más preciso.