**Detección de Color Rojo en Tiempo Real con OpenCV**

---

# 1. Introducción

Este código implementa un sistema básico de detección de color en tiempo real utilizando la cámara del dispositivo. Específicamente, detecta regiones rojas en el video mediante el uso del espacio de color HSV y operaciones de enmascaramiento. Esta técnica es muy común en tareas de seguimiento de objetos, reconocimiento de señales o segmentación por color.

---

# 2. Explicación Línea por Línea

```
import cv2
import numpy as np
```

- **cv2:** Importa la biblioteca OpenCV, utilizada para el procesamiento de imágenes y video.
- **numpy:** Importa NumPy para operaciones numéricas, como la definición de rangos de colores.

```
cap = cv2.VideoCapture(0)
```

- Inicia la captura de video desde la **cámara predeterminada** (generalmente la webcam integrada).

```
while(1):
```

- Inicia un bucle **infinito** que se repetirá hasta que se presione la tecla ESC. Este bucle procesa cada fotograma capturado por la cámara.

```
  _, frame = cap.read()
```

- Lee un nuevo **fotograma** de la cámara. `frame` contiene la imagen actual, mientras que `_` guarda el valor de éxito de la captura (no usado).

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

- Convierte el fotograma del espacio de color **BGR (azul, verde, rojo)** a **HSV (tono, saturación, valor)**, más adecuado para la detección de colores.

```
lower_red = np.array([0,150,150])
 upper_red = np.array([20,250,250])
```

- Define los **rangos inferior y superior del color rojo** en el espacio HSV.
- Hue (tono) entre 0 y 20, saturación y valor en rangos altos para asegurar que sea un rojo fuerte.

```
mask = cv2.inRange(hsv, lower_red, upper_red)
```

- Crea una **máscara binaria** donde:
    - Los píxeles dentro del rango especificado se vuelven **blancos (255)**.
    - Los píxeles fuera del rango se vuelven **negros (0)**.

```
res = cv2.bitwise_and(frame, frame, mask=mask)
```

- Realiza una **operación AND a nivel de bits** entre el fotograma y sí mismo, aplicando la máscara.
- Como resultado, se conservan solo las partes de la imagen que corresponden al color rojo.

```
cv2.imshow('frame', frame)
```

- Muestra el **fotograma original**.

```
cv2.imshow('mask', mask)
```

- Muestra la **máscara binaria**, indicando en blanco las zonas rojas detectadas.

```
cv2.imshow('res', res)
```

- Muestra el **resultado final**, es decir, sólo las zonas rojas del video original.

```
k = cv2.waitKey(5) & 0xFF
```

- Espera **5 milisegundos** para detectar si se ha presionado una tecla. Se aplica una máscara binaria `0xFF` para asegurar compatibilidad entre sistemas.

```
if k == 27:
    break
```

- Si se presiona la tecla **ESC (código ASCII 27)**, el bucle se rompe, finalizando la ejecución.

```
cv2.destroyAllWindows()
```

- Cierra todas las **ventanas de visualización** abiertas por OpenCV.

```
cap.release()
```

- **Libera la cámara**, permitiendo que otros programas puedan acceder a ella.

---

# 3. Conceptos Generales

## Espacio de Color HSV

- **Hue (tono):** Representa el color puro (ej. rojo, azul).
- **Saturation (saturación):** Intensidad del color.
- **Value (valor):** Brillo de la imagen.

Este modelo es más adecuado que BGR para la detección de colores, ya que separa información de color y brillo.

## Detección de Colores con Máscaras

- Las máscaras binarizan las imágenes en función de condiciones específicas (en este caso, rangos de color), facilitando tareas como segmentación, seguimiento o conteo de objetos.

---

# 4. Conclusión

Este código demuestra una implementación práctica y sencilla de detección de color en tiempo real. Utiliza el espacio HSV para mejorar la robustez de la detección, y muestra los resultados en tiempo real, lo que lo hace ideal para aplicaciones como reconocimiento de señales, seguimiento de objetos o interacción con el entorno mediante visión artificial.