## EXPLICACIÓN DETALLADA

### import cv2, import numpy as np

* cv2: Biblioteca **OpenCV** para visión por computadora.
* numpy: Para trabajar con arrays numéricos (usado para definir rangos de color).

### cv2.VideoCapture(0)

Inicia la **captura de video** desde la cámara predeterminada (índice 0).

### Verificación de cámara

python

Copiar código

if not camara.isOpened():

Comprueba si la cámara se abrió correctamente. Si no, se detiene el programa.

### DETECCIÓN DE COLOR EN HSV

python

Copiar código

color\_bajo = np.array([0, 120, 70])

color\_alto = np.array([10, 255, 255])

Este rango corresponde al **color rojo**, pero puede ajustarse para detectar otros colores (por ejemplo, azul, verde, amarillo, etc.).

#### ¿Por qué HSV?

* HSV (Hue, Saturation, Value) es más **robusto** que BGR para detectar colores, especialmente con diferentes iluminaciones.
* Hue: el tono (0 a 180 en OpenCV).
* Saturation: qué tan puro es el color.
* Value: brillo del color.

### Bucle principal

python

Copiar código

while True:

ret, frame = camara.read()

Captura cada cuadro de la cámara.

### Conversión a HSV

python

Copiar código

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

Convierte el frame de **BGR (por defecto en OpenCV)** a **HSV**, necesario para aplicar el rango de color.

### Crear máscara binaria

python

Copiar código

mascara = cv2.inRange(hsv, color\_bajo, color\_alto)

* Crea una imagen en blanco y negro:
  + Píxeles dentro del rango: **blanco (255)**.
  + Píxeles fuera del rango: **negro (0)**.

### Detección de contornos

python

Copiar código

contornos, \_ = cv2.findContours(...)

Encuentra los **contornos blancos** en la máscara.

### Dibujo sobre la imagen

python

Copiar código

for contorno in contornos:

area = cv2.contourArea(contorno)

if area > 500:

x, y, w, h = cv2.boundingRect(contorno)

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.putText(frame, f"Area: {int(area)}", (x, y - 10), ...)

* Calcula el **área** del contorno.
* Si es mayor a 500 (filtro para eliminar ruido pequeño), se:
  + Dibuja un **rectángulo verde** alrededor del objeto detectado.
  + Muestra el área numérica sobre la imagen.

### Mostrar ventanas

python

Copiar código

cv2.imshow("Camara", frame)

cv2.imshow("Mascara", mascara)

* Camara: Imagen original con detecciones.
* Mascara: Solo las regiones detectadas del color deseado.

### Salida con 'q'

python

Copiar código

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

Presiona la tecla **q** para salir del programa.

### Limpieza final

python

Copiar código

camara.release()

cv2.destroyAllWindows()

* Libera la cámara.
* Cierra todas las ventanas de OpenCV.

## ¿QUÉ HACE EL PROGRAMA EN LA PRÁCTICA?

1. Detecta **un color específico** en tiempo real (ej. rojo).
2. Dibuja **rectángulos** alrededor de los objetos detectados.
3. Muestra el **área** de esos objetos.
4. Te permite **visualizar** tanto la cámara como la máscara binaria.
5. Sirve como base para proyectos de:
   * Seguimiento de objetos por color.
   * Control de robots por colores.
   * Interfaces físicas usando papeles de colores.