**¿Qué hace este código?**

Este script:

* Abre la cámara,
* Captura video en tiempo real,
* Detecta rostros,
* Dentro de cada rostro detecta ojos,
* Dibuja rectángulos sobre ellos y les pone etiquetas.

**Explicación línea por línea**

**1. Importación de la librería OpenCV**

python

Copiar código

import cv2

* Importa la biblioteca de visión por computador OpenCV.

**2. Cargar clasificadores Haar**

python

Copiar código

clasificador\_rostros = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

clasificador\_ojos = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

* Se cargan **dos clasificadores Haar Cascade**:
  + haarcascade\_frontalface\_default.xml: detecta rostros frontales.
  + haarcascade\_eye.xml: detecta ojos dentro de una región.
* Estos archivos .xml contienen datos entrenados para identificar patrones específicos.

**3. Inicializar la cámara**

python

Copiar código

cam = cv2.VideoCapture(0)

if not cam.isOpened():

print("No se pudo acceder a la cámara")

exit()

* Abre la cámara predeterminada del sistema (0).
* Si no se puede abrir, muestra error y termina.

**4. Instrucciones al usuario**

python

Copiar código

print("Presiona 'q' para salir")

* Informa cómo finalizar el programa.

**5. Bucle de procesamiento de video**

python

Copiar código

while True:

ret, frame = cam.read()

* Bucle infinito que captura cuadros de la cámara.
* ret: indica si la captura fue exitosa.
* frame: imagen del cuadro actual.

**6. Verificar si se capturó correctamente**

python

Copiar código

if not ret:

print("Error al capturar imagen")

break

* Si falla la captura, se imprime mensaje de error y termina.

**7. Conversión a escala de grises**

python

Copiar código

gris = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

* Convierte la imagen a escala de grises.
* Los clasificadores Haar funcionan sobre imágenes en blanco y negro.

**8. Detección de rostros**

python

Copiar código

rostros = clasificador\_rostros.detectMultiScale(gris, scaleFactor=1.1, minNeighbors=5)

* Detecta **rostros** en la imagen gris.
* scaleFactor=1.1: reduce la imagen en cada iteración.
* minNeighbors=5: cuántos vecinos debe tener un rectángulo para considerarse rostro.

**9. Dibujar rostros y detectar ojos dentro de cada uno**

python

Copiar código

for (x, y, w, h) in rostros:

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

cv2.putText(frame, "Rostro", (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

* Por cada rostro detectado:
  + Se dibuja un **rectángulo verde**.
  + Se escribe la palabra "Rostro" encima.

**10. Definir región de interés (ROI) para buscar ojos**

python

Copiar código

rostro\_gris = gris[y:y + h, x:x + w]

rostro\_color = frame[y:y + h, x:x + w]

* Se recorta el rostro para:
  + rostro\_gris: usarlo como entrada para detectar ojos.
  + rostro\_color: dibujar los ojos detectados en color.

**11. Detección de ojos dentro del rostro**

python

Copiar código

ojos = clasificador\_ojos.detectMultiScale(rostro\_gris, scaleFactor=1.1, minNeighbors=10)

* Se detectan ojos **solo dentro del área del rostro**.
* minNeighbors=10: aumenta la precisión y reduce falsos positivos.

**12. Dibujar rectángulos sobre los ojos**

python

Copiar código

for (ox, oy, ow, oh) in ojos:

cv2.rectangle(rostro\_color, (ox, oy), (ox + ow, oy + oh), (255, 0, 0), 2)

cv2.putText(rostro\_color, "Ojo", (ox, oy - 5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 0, 0), 1)

* Se dibujan **rectángulos azules** sobre los ojos.
* Se etiqueta con "Ojo" cada uno.

**13. Mostrar la imagen procesada**

python

Copiar código

cv2.imshow("Rostros y Ojos Detectados", frame)

* Muestra el cuadro en una ventana en tiempo real.

**14. Salir con 'q'**

python

Copiar código

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

break

* Escucha si el usuario presiona 'q' para salir.

**15. Limpieza final**

python

Copiar código

cam.release()

cv2.destroyAllWindows()

* release(): cierra la cámara.
* destroyAllWindows(): cierra ventanas de OpenCV.

**Resultado visual**

* ✅ Rostros: rectángulo verde con etiqueta "Rostro".
* ✅ Ojos: rectángulo azul con etiqueta "Ojo", solo dentro del área del rostro.

**¿Qué puedes modificar o mejorar?**

| **Quieres hacer esto...** | **Cambia esto...** |
| --- | --- |
| Detectar ojos más fácilmente | Baja minNeighbors=6 para ojos |
| Filtrar por tamaño mínimo de rostro | Agrega condicional if w > X and h > Y: |
| Guardar las caras detectadas en imágenes | Usa cv2.imwrite() dentro del bucle |
| Agregar detección de sonrisas u otras cosas | Usa otro clasificador como haarcascade\_smile.xml |
| Usar cámara IP o externa | Cambia cv2.VideoCapture(0) por su URL o índice |