## ¿Qué hace este código?

Este script abre la cámara, captura imágenes en tiempo real, y detecta **rostros humanos** en cada cuadro usando un **clasificador Haar Cascade preentrenado**. Si detecta un rostro, dibuja un rectángulo verde alrededor de él.

## Explicación paso a paso

### 1. Importación de la librería

python

Copiar código

import cv2

* Se importa la biblioteca **OpenCV**, utilizada para visión por computador.

### 2. Cargar el clasificador de rostros

python

Copiar código

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

* Carga un **modelo Haar Cascade** preentrenado para detectar **rostros frontales**.
* El archivo XML (haarcascade\_frontalface\_default.xml) contiene los patrones que el algoritmo usa para encontrar rostros.
* cv2.data.haarcascades te da la ruta correcta del archivo, no hace falta escribir la ruta completa.

### 3. Inicializar la cámara

python

Copiar código

cam = cv2.VideoCapture(0)

if not cam.isOpened():

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

exit()

* VideoCapture(0) abre la **cámara por defecto** (puedes probar 1 si tienes más de una).
* isOpened() comprueba si la cámara funciona correctamente. Si no, se detiene el programa.

### 4. Instrucciones para el usuario

python

Copiar código

print("Presiona 'q' para salir")

* Mensaje que indica cómo cerrar el programa.

### 5. Bucle de captura de video

python

Copiar código

while True:

ret, frame = cam.read()

* Bucle infinito que captura imágenes en tiempo real.
* ret es True si la captura fue exitosa.
* frame es la imagen capturada.

### 6. Verificar si se capturó la imagen

python

Copiar código

if not ret:

print("Error al capturar imagen")

break

* Si ret es False, ocurre un error y se termina el programa.

### 7. Conversión a escala de grises

python

Copiar código

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

* Convierte la imagen de color a **escala de grises**.
* Los clasificadores Haar Cascade funcionan mejor con imágenes en blanco y negro.

### 8. Detección de rostros

python

Copiar código

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

#### Explicación de parámetros:

* gris: imagen en escala de grises sobre la que buscar rostros.
* scaleFactor=1.1: cuanto más cercano a 1.0, más precisa y lenta la detección. Indica cuánto se reduce la imagen en cada escala.
* minNeighbors=5: cuántos vecinos necesita un rectángulo para considerarse una detección válida. Más vecinos = menos falsos positivos, pero también puede perder detecciones pequeñas.
* Resultado: una lista de tuplas (x, y, w, h) donde:
  + x, y: coordenadas del vértice superior izquierdo del rostro.
  + w, h: ancho y alto del rectángulo.

### 9. Dibujar rectángulos sobre los rostros detectados

python

Copiar código

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

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

* Por cada rostro detectado, se dibuja un **rectángulo verde** alrededor en el frame original.

### 10. Mostrar la imagen procesada

python

Copiar código

cv2.imshow("Rostros Detectados", frame)

* Muestra en una ventana la imagen con los rostros marcados.

### 11. Salir si se presiona 'q'

python

Copiar código

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

break

* Espera 1 milisegundo por una tecla.
* Si presionas 'q', rompe el bucle.

### 12. Limpieza

python

Copiar código

cam.release()

cv2.destroyAllWindows()

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

## ¿Qué puedes modificar?

| **Quieres hacer esto...** | **Cambia esto...** |
| --- | --- |
| Detectar más rápido (menos preciso) | scaleFactor=1.2 o mayor |
| Detectar rostros más pequeños | minNeighbors=3 o menor |
| Dibujar etiquetas con texto | Usa cv2.putText(...) dentro del for |
| Guardar una imagen si se detecta un rostro | Usa cv2.imwrite("rostro.jpg", frame) |
| Grabar el video con rostros detectados | Usa cv2.VideoWriter |