**Actividad 4: Detección de Rostros con OpenCV**

**Objetivo**

Aprender a detectar rostros humanos en tiempo real usando la cámara web, con ayuda de OpenCV y los clasificadores Haar Cascade. Este ejercicio es fundamental en el campo de la visión artificial, especialmente para sistemas de seguridad, reconocimiento biométrico o interfaces interactivas.

**¿Qué es un clasificador Haar Cascade?**

Un clasificador Haar Cascade es un algoritmo entrenado para detectar objetos específicos en imágenes. Fue introducido por Paul Viola y Michael Jones, y funciona a través del aprendizaje automático. Este proceso requiere:

* **Imágenes positivas** (donde aparece el objeto a detectar).
* **Imágenes negativas** (sin el objeto).

Una vez entrenado, el clasificador puede escanear nuevas imágenes y detectar patrones similares a los del entrenamiento. OpenCV proporciona varios clasificadores preentrenados, incluyendo detección de rostros, ojos, sonrisas, matrículas, entre otros.

**Conceptos clave utilizados en esta actividad**

|  |  |
| --- | --- |
| **Concepto** | **Descripción** |
| CascadeClassifier | Clase que carga un archivo XML con el modelo Haar Cascade preentrenado. |
| detectMultiScale() | Método para detectar objetos de diferentes tamaños en una imagen. |
| scaleFactor | Define la reducción de la imagen en cada escala de análisis. |
| minNeighbors | Cuántos vecinos debe tener un rectángulo para que sea considerado válido. |
| cv2.cvtColor(..., COLOR\_BGR2GRAY) | Convierte una imagen de color a escala de grises. |

**Requisitos previos**

* Python (versión 3.8 o superior).
* OpenCV (pip install opencv-python).
* Archivo haarcascade\_frontalface\_default.xml (se encuentra en OpenCV por defecto).

**Código Python completo**

import cv2

# Cargar el clasificador Haar Cascade para detección de rostros

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

# Iniciar la cámara

camara = cv2.VideoCapture(0)

if not camara.isOpened():

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

exit()

print("Presiona 'q' para salir")

while True:

ret, frame = camara.read()

if not ret:

print("Error al capturar imagen")

break

# Convertir a escala de grises

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

# Detección de rostros en la imagen

rostros = clasificador\_rostro.detectMultiScale(

gris, scaleFactor=1.1, minNeighbors=5

)

# Dibujar rectángulos verdes alrededor de los rostros

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

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

# Mostrar el resultado

cv2.imshow("Detección de Rostros", frame)

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

break

# Liberar recursos

camara.release()

cv2.destroyAllWindows()

**Explicación detallada del código**

**1. Importar librerías**

import cv2

Importamos la librería de OpenCV, que proporciona funciones para procesamiento de imágenes y visión por computadora.

**2. Cargar el clasificador Haar Cascade**

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

Este modelo XML está entrenado para identificar rostros frontales. Se encuentra en la carpeta de datos de OpenCV.

**3. Iniciar la cámara**

camara = cv2.VideoCapture(0)

Abre la cámara predeterminada del sistema. 0 indica la primera cámara conectada.

**4. Capturar imagen y convertir a escala de grises**

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

La detección se realiza en escala de grises porque es más rápida y eficaz al reducir la complejidad de la imagen (solo un canal en lugar de tres).

**5. Detección de rostros**

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

* scaleFactor: controla el tamaño de la imagen en cada paso (más pequeño permite detectar rostros de diferentes tamaños).
* minNeighbors: controla cuántas detecciones similares necesita para considerarse un rostro real.

**6. Dibujar rectángulos en la imagen**

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

Dibuja un borde verde (color RGB: 0, 255, 0) alrededor del rostro detectado. El grosor de la línea es de 2 píxeles.

**7. Mostrar imagen y salir con 'q'**

cv2.imshow("Detección de Rostros", frame)