🖥️ Guía paso a paso: OpenCV + multicámara + detección facial LBPH

Para Windows y Linux

📦 1. Instalar Python

Windows 🪟

1. Ve a la web oficial de Python:
   🔗 https://www.python.org/downloads/windows/
2. Descarga el instalador (elige la versión más reciente, por ejemplo Python 3.11).
3. Durante la instalación:
   - Marca la casilla **Add Python 3.x to PATH**
   - Luego clic en **Install Now**
4. Verifica instalación abriendo **PowerShell** o **CMD** y ejecuta:

bash
CopiarEditar
```
python --version
```

Debería mostrar la versión instalada.

Linux 🐧 (Ubuntu/Debian)

1. Abre la terminal.
2. Actualiza paquetes:

bash
CopiarEditar
```
sudo apt update
sudo apt upgrade -y
```

3. Instala Python3 y pip:

bash
CopiarEditar
```
sudo apt install python3 python3-pip -y
```

4. Verifica instalación:

bash
CopiarEditar

```
python3 --version
pip3 --version
```

📦 2. Instalar OpenCV con contrib (módulo face incluido)

Este paso es importante porque la detección LBPH está en el módulo **contrib**.

Windows & Linux (con pip)

Abre tu terminal o PowerShell y ejecuta:

bash
CopiarEditar

```
pip install opencv-contrib-python numpy
```

- `opencv-contrib-python`: versión completa con módulos extra.
- `numpy`: requerido para procesamiento de imágenes.

🧰 3. Preparar dataset para reconocimiento facial

---

1. Crea carpeta llamada `dataset` dentro de tu proyecto.
2. Dentro de `dataset`, crea una carpeta por cada persona que quieres reconocer, con su nombre o alias:

CopiarEditar

```
dataset/
 ├─ juan/
 |    ├─ juan1.jpg
 |    ├─ juan2.jpg
 ├─ maria/
 |    ├─ maria1.jpg
 |    ├─ maria2.jpg
```

3. Usa fotos claras, de preferencia con el rostro centrado y en diferentes condiciones.
4. Puedes usar tu webcam para sacar fotos y guardarlas aquí.

🔨 4. Entrenar modelo LBPH con Python

Código: `entrenar_lbph.py`

python
CopiarEditar

```
import cv2
import os
import numpy as np
import pickle

def get_images_and_labels(dataset_path):
    faces = []
    labels = []
    label_map = {}
    current_label = 0

    for person_name in os.listdir(dataset_path):
        person_folder = os.path.join(dataset_path, person_name)
        if not os.path.isdir(person_folder):
            continue
        label_map[current_label] = person_name

        for image_name in os.listdir(person_folder):
            image_path = os.path.join(person_folder, image_name)
            img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
            if img is None:
                continue
            faces.append(img)
            labels.append(current_label)

        current_label += 1

    return faces, labels, label_map

dataset_path = 'dataset'  # Cambiar si tienes otro path

faces, labels, label_map = get_images_and_labels(dataset_path)
```

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
print("[INFO] Entrenando modelo...")
recognizer.train(faces, np.array(labels))
print("[INFO] Entrenamiento completado.")

recognizer.save("modelo_lbph.yml")

with open("label_map.pkl", "wb") as f:
    pickle.dump(label_map, f)
```

Cómo ejecutar el entrenamiento:

- Abre terminal / PowerShell.
- Ve a la carpeta donde guardaste `entrenar_lbph.py`.
- Ejecuta:

bash
CopiarEditar
`python entrenar_lbph.py`

- Saldrá:

  csharp
  CopiarEditar
  `[INFO] Entrenando modelo...`
  `[INFO] Entrenamiento completado.`

- Se crearán dos archivos:
  - `modelo_lbph.yml`
  - `label_map.pkl`

🖥️ 5. Código para servidor multicámara con detección facial y reconocimiento

---

Código: `servidor_multicamara.py`

python
CopiarEditar
```
import cv2
import socket
import threading
import numpy as np
```

```
import struct
import pickle

HOST = '0.0.0.0'
PORT = 9999

# Cargar modelo entrenado y etiquetas
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("modelo_lbph.yml")

with open("label_map.pkl", "rb") as f:
    label_map = pickle.load(f)

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

def client_thread(conn, addr, client_id):
    print(f"[SERVIDOR] Cliente conectado {addr}, ID: {client_id}")
    data_buffer = b''
    payload_size = struct.calcsize(">L")

    try:
        while True:
            while len(data_buffer) < payload_size:
                packet = conn.recv(4096)
                if not packet:
                    print(f"[SERVIDOR] Cliente {client_id} desconectado")
                    return
                data_buffer += packet

            packed_msg_size = data_buffer[:payload_size]
            data_buffer = data_buffer[payload_size:]
            msg_size = struct.unpack(">L", packed_msg_size)[0]

            while len(data_buffer) < msg_size:
                packet = conn.recv(4096)
```

```
                if not packet:
                    print(f"[SERVIDOR] Cliente {client_id} desconectado")
                    return
                data_buffer += packet

            frame_data = data_buffer[:msg_size]
            data_buffer = data_buffer[msg_size:]

            frame = cv2.imdecode(np.frombuffer(frame_data, np.uint8),
cv2.IMREAD_COLOR)
            if frame is None:
                print(f"[SERVIDOR] Cliente {client_id} frame corrupto")
                continue

            # Detectar y reconocer caras
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)

            for (x, y, w, h) in faces:
                roi_gray = gray[y:y+h, x:x+w]
                id_, conf = recognizer.predict(roi_gray)

                if conf < 70:
                    name = label_map.get(id_, "Desconocido")
                    color = (0, 255, 0)
                else:
                    name = "Desconocido"
                    color = (0, 0, 255)

                cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
                cv2.putText(frame, f"{name} ({int(conf)})", (x, y-10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, color, 2)

            cv2.imshow(f'Cliente {client_id}', frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                print("[SERVIDOR] Cierre solicitado")
```

```
                break

    finally:
        conn.close()
        cv2.destroyWindow(f'Cliente {client_id}')
        print(f"[SERVIDOR] Cliente {client_id} desconectado")

def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((HOST, PORT))
    server.listen(5)
    print(f"[SERVIDOR] Escuchando en {HOST}:{PORT}")

    client_id = 0
    try:
        while True:
            conn, addr = server.accept()
            client_id += 1
            threading.Thread(target=client_thread, args=(conn, addr, client_id),
daemon=True).start()
    except KeyboardInterrupt:
        print("[SERVIDOR] Servidor detenido")
    finally:
        server.close()
        cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```

📡 6. Código cliente para enviar vídeo (desde cámara)

Código: `cliente_simple.py`

python
CopiarEditar

```
import cv2
import socket
```

```
import struct

SERVER_IP = 'IP_DEL_SERVIDOR'  # Cambia esto por la IP real
SERVER_PORT = 9999

def main():
    cap = cv2.VideoCapture(0)  # Cambiar si tienes más cámaras

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((SERVER_IP, SERVER_PORT))

    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break

            ret, buffer = cv2.imencode('.jpg', frame, [cv2.IMWRITE_JPEG_QUALITY,
80])
            if not ret:
                break

            data = buffer.tobytes()
            sock.sendall(struct.pack(">L", len(data)) + data)

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

    finally:
        cap.release()
        sock.close()

if __name__ == '__main__':
    main()
```

🛠️ 7. Cómo usar los scripts paso a paso

1️⃣ Entrena el modelo con tu dataset

bash
CopiarEditar
```
python entrenar_lbph.py
```

2️⃣ Ejecuta el servidor (máquina que recibe vídeo y hace detección)

bash
CopiarEditar
```
python servidor_multicamara.py
```

3️⃣ Ejecuta uno o varios clientes (máquinas que envían vídeo)

- Cambia la IP en `cliente_simple.py` para que apunte al servidor.
- Ejecuta:

bash
CopiarEditar
```
python cliente_simple.py
```

4️⃣ Visualiza la ventana con vídeo y detección

- En la máquina servidor verás ventanas con el vídeo de cada cliente.
- Caras detectadas y nombres reconocidos si están en el dataset.
- Para cerrar alguna ventana, pulsa `q`.

📝 Consejos finales

- Para cámaras IP, cambia `VideoCapture(0)` en el cliente a la URL RTSP o HTTP que provea la cámara.
- Ajusta el umbral de confianza en el servidor (`conf < 70`) para mejorar precisión.
- Usa buena iluminación para mejorar detección y reconocimiento.
- En caso de problemas con OpenCV en Windows, verifica que instalaste `opencv-contrib-python` y no solo `opencv-python`.