

Reconocimiento Facial

Primero se generó la estructura de carpetas y es “Script” para el análisis de reconocimiento facial del **“dataset_rostros”** que incluye la captura del dataset, carga/etiqueta y con los entrenamientos de los diferentes métodos de reconocimiento facial: Eigen, Fisher y LBPH.

Nombre	Estado	Fecha de modificación	Tipo	Tamaño
.env		12/08/2025 10:54 p. m.	Carpeta de archivos	
__pycache__		14/08/2025 12:18 p. m.	Carpeta de archivos	
Data		14/08/2025 12:14 p. m.	Carpeta de archivos	
modelos		14/08/2025 11:24 a. m.	Carpeta de archivos	
Muestras		14/08/2025 12:11 p. m.	Carpeta de archivos	
Persona_1		14/08/2025 11:24 a. m.	Carpeta de archivos	
Persona_2		14/08/2025 11:24 a. m.	Carpeta de archivos	
Persona_3		14/08/2025 11:24 a. m.	Carpeta de archivos	
utils		14/08/2025 11:24 a. m.	Carpeta de archivos	
captura_rostros.py		14/08/2025 12:13 p. m.	Archivo de origen ...	2 KB
carga_etiquetado.py		14/08/2025 11:04 a. m.	Archivo de origen ...	2 KB
deteccion_facial_real_time.py		14/08/2025 11:04 a. m.	Archivo de origen ...	2 KB
eigenFace_model.yml		14/08/2025 12:26 p. m.	Archivo de origen ...	48,036 KB
fisherFace_model.yml		14/08/2025 12:26 p. m.	Archivo de origen ...	1,152 KB
haarcascade_frontalface_default.xml		14/08/2025 11:51 a. m.	Microsoft Edge H...	909 KB
LBPHFace_model.yml		14/08/2025 12:26 p. m.	Archivo de origen ...	10,882 KB

Primero se genera el script para la captura de rostros desde la webcam nombrado **“captura_rostros”** se hace la detección con el clasificador (‘haarcascade_frontalface_default.xml’) detecta rostros con Haar y guarda recortes normalizados.

Después de genero un script para la captura de rostros desde una webcam(80) guardándolo como “captura_rostros.py” , que a su vez se corrio tre veces cambiando el nombre de la persona-A-B-C y usando Detección con el modelo Haar y guardar los recortes normalizado.

Luego generamos el script de carga y etiquetado de dataset, Genera imágenes en gris normalizadas, image de script:

```

carga_etiquetado.py > ...
1  import cv2
2  import os
3  import numpy as np
4
5  from captura_rostros import dataPath
6
7  dataPath = './Data'
8  peopleList = os.listdir(dataPath)
9  print('Lista de personas: ', peopleList)
10 labels = []
11 facesData = []
12 label = 0
13 for nameDir in peopleList:
14     personPath = dataPath + '/' + nameDir
15     print('Leyendo las imágenes')
16
17     for fileName in os.listdir(personPath):
18         print('Rostros: ', nameDir + '/' + fileName)
19         labels.append(label)
20         facesData.append(cv2.imread(personPath + '/' + fileName, 0))
21     label = label + 1
22
23 # Métodos para entrenar el reconocedor utilizando EigenFace, FisherFace y LBPH
24 face_recognizer_Eigen = cv2.face.EigenFaceRecognizer.create()
25 face_recognizer_Fisher = cv2.face.FisherFaceRecognizer.create()
26 face_recognizer_LBPHF = cv2.face.LBPHFaceRecognizer.create()
27
28 # Entrenando el reconocedor de rostros
29 print("Entrenando...")
30 face_recognizer_Eigen.train(facesData, np.array(labels))
31 face_recognizer_Fisher.train(facesData, np.array(labels))
32 face_recognizer_LBPHF.train(facesData, np.array(labels))
33
34 # Almacenando el modelo obtenido
35 face_recognizer_Eigen.write('eigenFace_model.yml')
36 face_recognizer_Fisher.write('fisherFace_model.yml')
37 face_recognizer_LBPHF.write('LBPHFace_model.yml')
38 print("Modelos almacenados...")

```

Después de este script por ultimo generamos el script de entrenamiento con los diferentes modelos (“EigenFaces, FisherFaces, LBPH”):

```

deteccion_facial_real_time.py > ...
1  import cv2
2  import os
3
4  from carga_etiquetado import face_recognizer_Fisher
5
6  # Ruta donde se almacenan las imágenes de entrenamiento
7  dataPath = './Data'
8  # Lista de imágenes en el directorio Data
9  imagePaths = os.listdir(dataPath)
10 print('imagePaths=', imagePaths)
11
12 # Creando el reconocedor de rostros
13 face_recognizer_Eigen = cv2.face.EigenFaceRecognizer.create()
14 face_recognizer_Fisher = cv2.face.FisherFaceRecognizer.create()
15 face_recognizer_LBPHF = cv2.face.LBPHFaceRecognizer.create()
16
17 # Leyendo el modelo
18 face_recognizer_Eigen.read('./eigenFace_model.yml')
19 face_recognizer_Fisher.read('./fisherFace_model.yml')
20 face_recognizer_LBPHF.read('./LBPHFace_model.yml')
21
22 # Usar la webcam (índice 0)
23 cap = cv2.VideoCapture(0)
24 # Cargando el clasificador de rostros
25 faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
26

```

```

while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = gray.copy()
    faces = faceClassif.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        rostro = auxFrame[y:y + h, x:x + w]
        rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC)
        result = face_recognizer_Eigen.predict(rostro)
        cv2.putText(frame, '{}'.format(result), (x, y - 5), 1, 1.3, (255, 255, 0), 1, cv2.LINE_AA)

        if result[1] < 5700:
            cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x, y - 25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        else:
            cv2.putText(frame, 'Desconocido', (x, y - 20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    cv2.imshow('frame', frame)
    k = cv2.waitKey(1)
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()

```

Comparación y análisis (qué observar)

- **LBPH:** Suele ser **más robusto** a cambios de iluminación y funciona bien con datasets pequeños; tiende a ser el más **estable** en tiempo real.
- **FisherFaces:** Bueno separando clases cuando hay buena variación y número similar de muestras por persona; puede ser sensible a iluminación.
- **EigenFaces:** Puede requerir más datos y uniformidad; sensible a iluminación y fondo.
- **Velocidad:** Los tres son rápidos en CPU; LBPH suele comportarse muy bien con webcam estándar.
- **Datos:** La **calidad** del recorte y la **consistencia** (tamaño, frontalidad, claridad) impactan más que el algoritmo.