Tarea 03: Reconocimiento facial en Python utilizando OpenCV

Alejandra Magalí López Miranda, 201600085[1, *]

[1] *Escuela de Ingenieria Mecanica Electrica, Facultad de ingenieria , Universidad de San Carlos, Guatemala.*

(Dated: 5 de agosto de 2025)

I. INTRODUCCIÓN

El reconocimiento facial funciona como un método de identificación biométrica, aprovechando características físicas únicas, en este caso, de la cara y la cabeza, para validar la identidad de un individuo basándose en su patrón biométrico y otros datos relevantes. En este trabajo se desarrolla un sistema de reconocimiento facial en Python utilizando la biblioteca OpenCV, implementando y comparando tres métodos: Eigenfaces, Fisherfaces y Local Binary Patterns Histograms (LBPH). Cada técnica emplea un enfoque distinto para la extracción y clasificación de características faciales, lo que permite evaluar su desempeño y precisión bajo diferentes condiciones. La práctica integra desde la captura de imágenes y el preprocesamiento, hasta el entrenamiento y validación de modelos, brindando un panorama general de las capacidades y limitaciones de cada método.

II. OBJETIVOS

A. General

- Implementar y evaluar un sistema de reconocimiento facial en Python utilizando OpenCV.

B. Específicos

* Capturar y preprocesar un conjunto de imágenes faciales para la construcción de la base de datos de entrenamiento.
* Implementar los algoritmos Eigenfaces, Fisherfaces y LBPH utilizando la biblioteca OpenCV.
* Comparar el desempeño de las tres técnicas en términos de precisión, tiempo de respuesta y tolerancia a variaciones de iluminación y expresión facial.

III. MARCO TEÓRICO

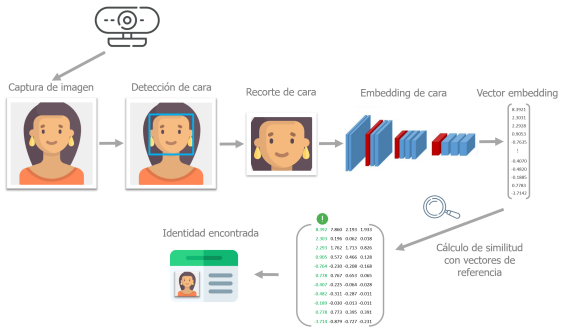
A. Reconocimiento facial

El reconocimiento facial es una forma de inteligencia artificial que imita una capacidad humana para reconocer caras humanas. Al igual que cuando un humano reconoce un rostro, el software de reconocimiento facial captura los rasgos faciales y crea un patrón de rasgos faciales que utiliza para identificar o agrupar un rostro.

B. Proceso de reconocimiento facial

- Se captura una imagen.
- El software de reconocimiento de imágenes con IA lee la geometría del rostro en la imagen observando la distancia entre los rasgos faciales clave, creando una plantilla facial única.
- La plantilla facial se compara con una base de datos de caras conocidas, almacenadas o disponibles.
- Se determina si se encuentra una coincidencia en la base de datos existente.

Figura 1: Diagrama del proceso de reconocimiento facial



C. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de código abierto que se utiliza para desarrollar aplicaciones de visión por computadora

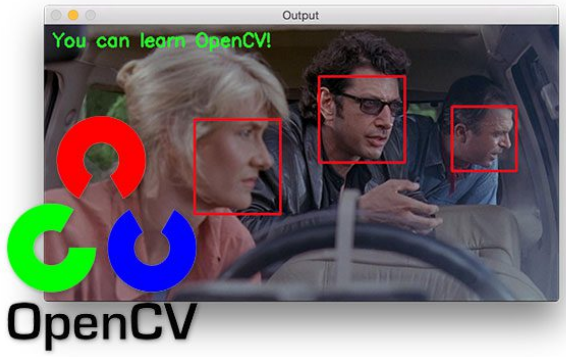
Su principal objetivo es facilitar el procesamiento de imágenes y videos en tiempo real, proporcionando un conjunto amplio de algoritmos que permiten realizar tareas como la detección de objetos, el reconocimiento facial, el análisis de movimiento y muchas otras.

Uno de los grandes atractivos de OpenCV es su compatibilidad con varios lenguajes de programación, lo que

* 2816733570108@ingenieria.usac.edu.gt

lo hace accesible a un mayor número de desarrolladores que programan en Python, C++, Java y MATLAB; aunque Python es el lenguaje más utilizado gracias a su simplicidad y a la extensa cantidad de recursos disponibles en la comunidad.

Figura 2: OpenCV

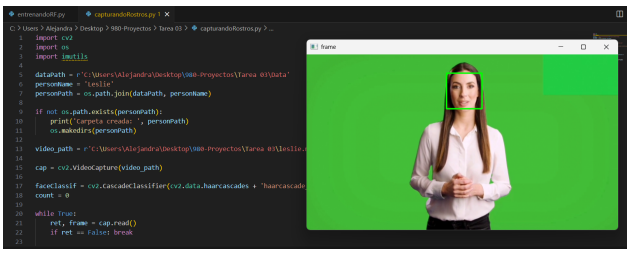


IV. RESULTADOS

A. Capturando Rostros

Este script utiliza la biblioteca OpenCV para capturar fotogramas desde un archivo de video (leslie.mp4 o Alejandra.mp4), y asi detectar rostros en cada fotograma mediante el clasificador $haarcascade_frontalface_default.xml$ y guardar las imágenes recortadas de cada rostro en una carpeta específica (Data/Leslie o Data/Alejandra). Antes de guardar, cada rostro se redimensiona a 150×150 píxeles. El bucle de captura continúa hasta que se presiona la tecla ESC o se hayan almacenado 600 imágenes. Esta etapa corresponde a la fase de creación de la base de datos facial, necesaria para entrenar posteriormente los algoritmos de reconocimiento como Eigenfaces, Fisherfaces o LBPH.

Figura 3: Capturando Rostros



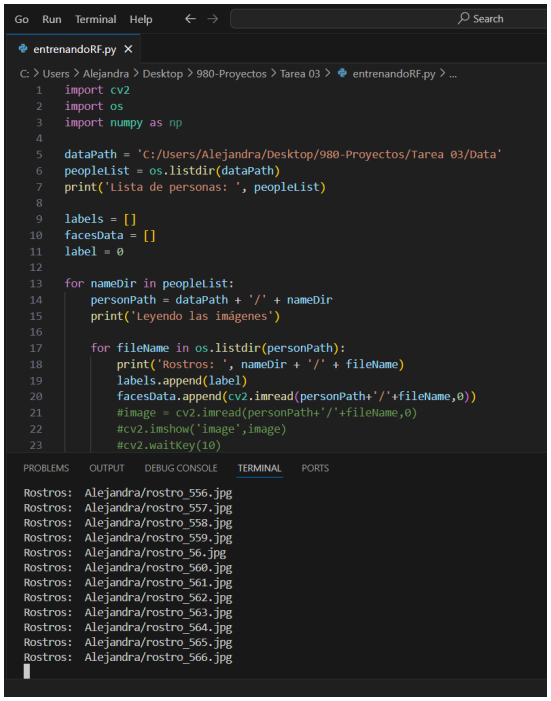
B. Entrenamiento

En esta paso se importaron librerías como cv2, os, numpy para trabajar con imágenes, archivos y datos numéricos.

Se obtiene la lista de carpetas en dataPath, donde cada carpeta corresponde a una persona diferente. Se inicializa listas para guardar las etiquetas (labels) y las imágenes (facesData). Recorre cada carpeta (persona) y luego cada imagen dentro de esa carpeta: Asigna una etiqueta numérica (label) a todas las imágenes de esa persona. Y lee las imágenes en escala de grises y las guarda en facesData.

Crea el reconocedor facial, en este caso LBPH, aunque podría ser Eigenfaces o Fisherfaces.Entrena el modelo usando las imágenes (facesData) y sus etiquetas (labels). Guarda el modelo entrenado en un archivo .xml para usarlo después en el reconocimiento.

Figura 4: Entrenamiento



C. Reconocimiento facial

Se obtiene los nombres de carpetas en dataPath, que corresponden a las personas entrenadas.

Se crea el reconocedor facial (en este caso FisherFace), se carga el modelo entrenado desde el archivo .xml guardado previamente.Abre la fuente de video y carga el clasificador Haar Cascade para detectar rostros en las imágenes.

El bucle de captura de video convierte el cuadro a escala de grises, detecta rostros en el cuadro. Para cada

rostro detectado lo recorta y redimensiona a 150×150 píxeles y lo envía al modelo para predecir la persona (predict).

Se muestra en pantalla el nombre si la predicción está por debajo de un umbral (en este caso, <500 para FisherFace), o "Desconocido" si no coincide.Dibuja un rectángulo verde (reconocido) o rojo (desconocido).

Figura 5: Alejandra

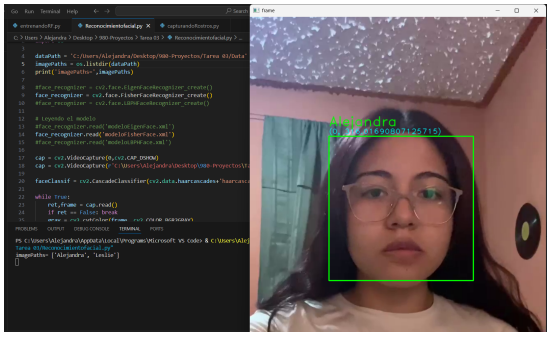


Figura 6: Leslie

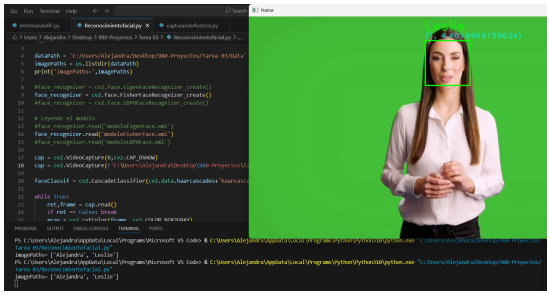
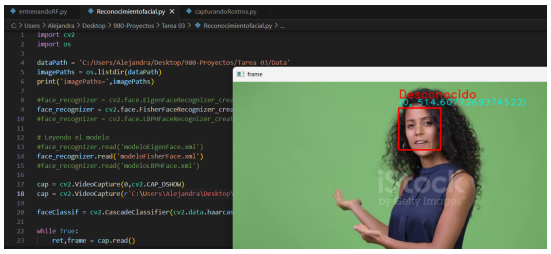


Figura 7: Desconocido



V. CONCLUSIONES

* Durante las pruebas se identificó que el algoritmo FisherFace, requiere obligatoriamente de al menos imágenes de dos personas diferentes para poder calcular las diferencias entre clases. Cuando se intenta ejecutar con datos de una sola persona, el algoritmo no encuentra variaciones discriminantes relevantes, lo que provoca que lance un error. Esto resalta la importancia de seleccionar el método de reconocimiento facial considerando el tamaño y la diversidad de la base de datos, ya que, en casos de entrenamiento con un único individuo, es más adecuado utilizar algoritmos como LBPH o Eigenfaces que no presentan esta limitación.

* En las pruebas se observó que el modelo identificaba erróneamente a personas distintas como (Alejandra), generando falsos positivos. La causa principal fue un valor de umbral alto en el proceso de predicción, lo que impedía diferenciar correctamente entre rostros similares y los registrados en el entrenamiento. Reducir este valor permitió que el sistema rechace con mayor precisión a individuos que no forman parte de la base de datos.

* En el sistema de reconocimiento facial, Leslie no es reconocida correctamente ni por LBPH ni por Eigenfaces, a pesar de tener 600 imágenes, y esto pudo deberse a que el dataset de Leslie carece de suficiente variación en iluminación, ángulos y expresiones.

VI. ANEXOS

Link de repositorio: `https://github.com/AlejandraMLM/980-Proyectos.git`

[1] Reconocimiento Facial | Python - OpenCV. Disponible en: `https://omes-va.com/reconocimiento-facial-python-opencv/`

[2] Reconocimiento facial. Disponible en: `https://www.miteksystems.com/es/blog/reconocimiento-facial-que-es-usos`

[3] Proceso de reconocimiento facial. Disponible en: `https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-face-recognition`

[4] Reconocimiento facial con deep learning y python. Disponible en: https://cienciadedatos.net/documentos/py34-reconocimiento-facial-deeplearning-python

[5] OpenCV. Disponible en: https://openwebinars.net/blog/opencv-introduccion-y-su-rol-en-la-vision-por-computadora/