Profesor: Francisco Javier Luna Rosas

Materia: Aprendizaje Inteligente

Alumnos:
Gustavo Benjamin II Pedraza Morales

Alejandro Pérez Velasco

Grupo: 6ºA

Proyecto final

Proyecto

El alumno deberá diseñar un modelo de aprendizaje (supervisado/o no supervisado) para detectar patrones relacionados con imágenes (por ejemplo, reconocimiento de firmas, reconocimiento de rostros, análisis de sentimientos utilizando el reconocimiento de emociones faciales, etc. (ver Figura 1).

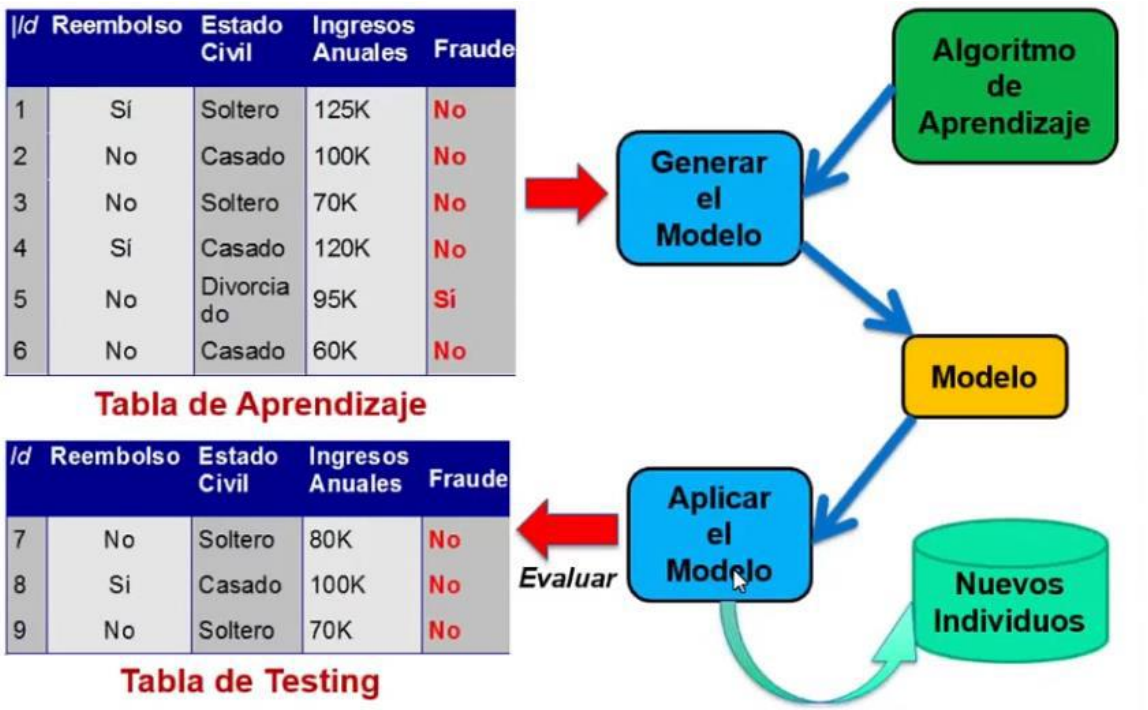


Figura 1. Modelo General para el Reconocimiento de Patrones, Utilizando Aprendizaje Supervisado con Entrenamiento Tabla Testing.

El alumno deberá diseñar el modelo de aprendizaje con las siguientes características:

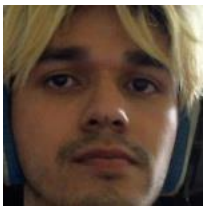
a) Dataset. Cuáles son las variables predictoras y cuáles son las variables a predecir.
b) Qué tipo de algoritmos se utilizan (explique el funcionamiento de los algoritmos, al menos tres algoritmos).
c) Qué tipo de entrenamiento se utiliza (Tabla Testing, LOOCV, K-Folds).
d) Realice análisis comparativo de máquinas de machine learning con aprendizaje supervisado (al menos tres máquinas), compare la precisión del modelo y su razón de error.

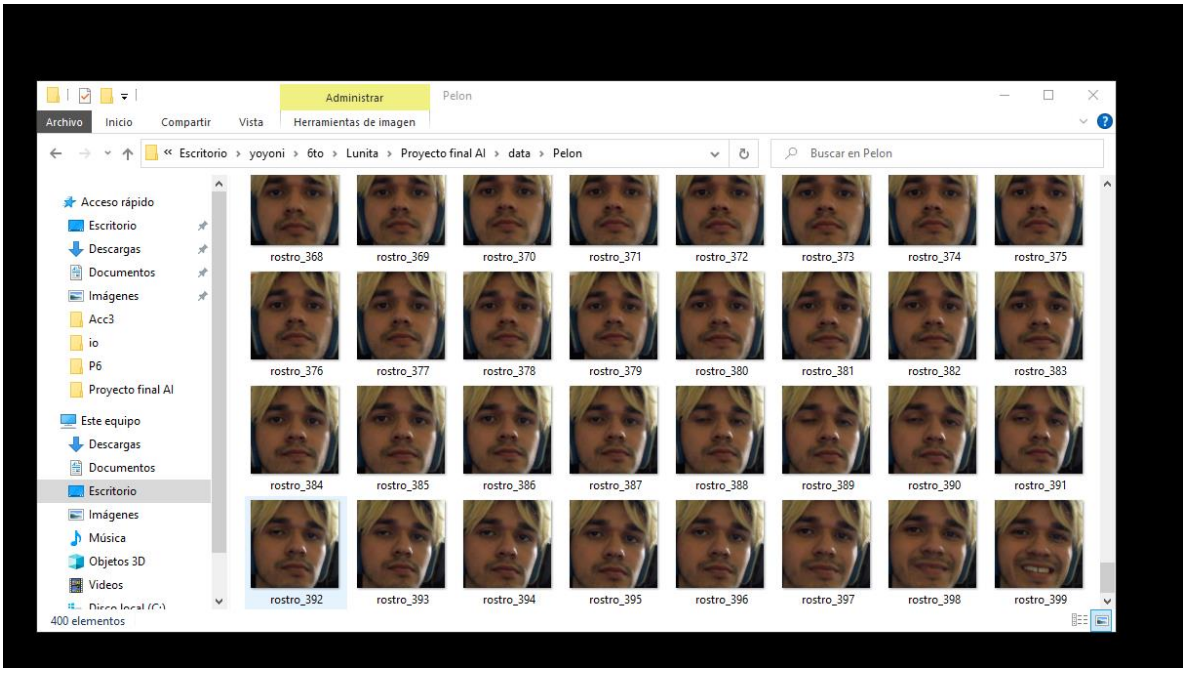
Dataset. Cuáles son las variables predictoras y cuáles son las variables a predecir.

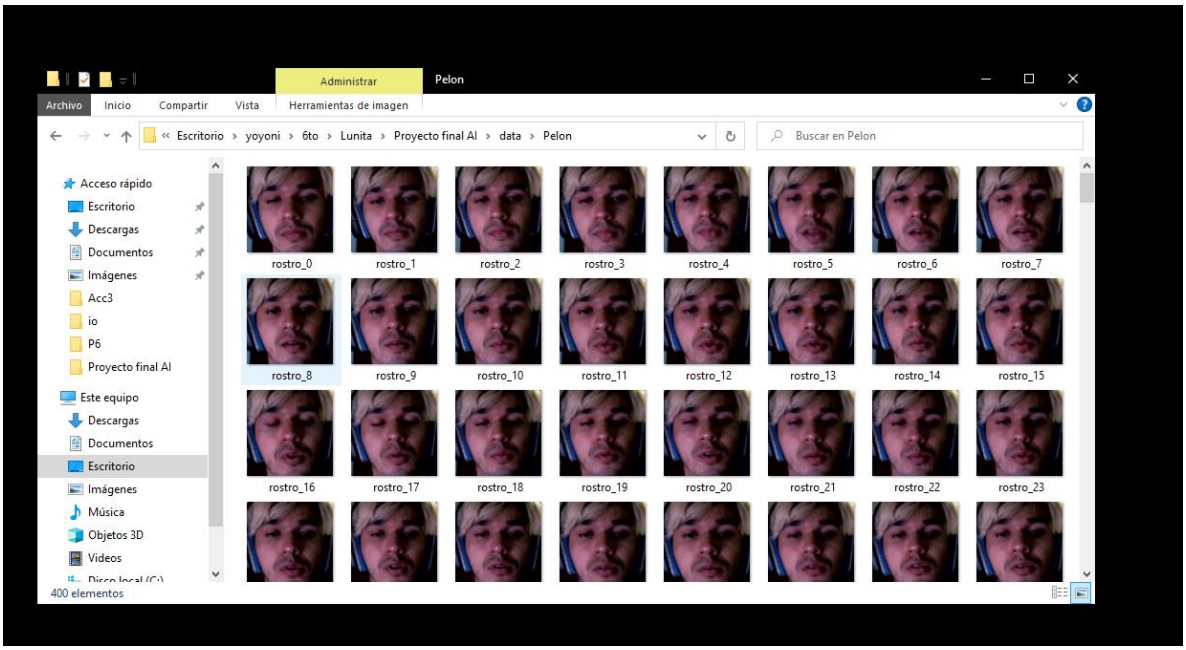
Para nuestro proyecto lo que realizaremos será la detección de rostros para el cual para fines de pruebas le dimos como entrada 3 videos de 2 sujetos de prueba distintos, de los cuales obtuvimos nuestro Dataset.

De cada video de 30 segundos, realizamos captura de 300 imágenes las cuales utilizaremos como variables predictoras, y agregamos 100 de captura en tiempo real, en total tenemos 700 fotos que conforman nuestro Dataset con el cual haremos funcionar el programa para la detección del sujeto.

Nuestras variables a predecir serían los rostros de los sujetos de prueba que deseamos identificar, y se basa en la carpeta que nosotros denominamos como “Pelon”.







Qué tipo de algoritmos se utilizan (explique el funcionamiento de los algoritmos, al menos tres algoritmos).

EigenFaces

El problema con la representación de la imagen que se nos da es su alta dimensionalidad. Las imágenes bidimensionales p × q en escala de grises abarcan un espacio vectorial m = pq-dimensional, por lo que una imagen con 100 × 100 píxeles ya se encuentran en un espacio de imagen de 10,000 dimensiones. La pregunta es: ¿son todas las dimensiones igualmente útiles para nosotros? Solo podemos tomar una decisión si hay alguna variación en los datos, por lo que lo que buscamos son los componentes que representan la mayor parte de la información. El análisis de componentes principales (PCA) fue propuesto independientemente por Karl Pearson (1901) y Harold Hotelling (1933) para convertir un conjunto de variables posiblemente correlacionadas en un conjunto más pequeño de variables no correlacionadas. La idea es que un conjunto de datos de alta dimensión a menudo se describe mediante variables correlacionadas y, por lo tanto, solo unas pocas dimensiones significativas representan la mayor parte de la información. El

método PCA encuentra las direcciones con la mayor variación en los datos, llamadas componentes principales.

FisherFaces

El análisis de componentes principales (APC), que es el núcleo del método de caras propias, encuentra una combinación lineal de características que maximiza la varianza total en los datos. Si bien esta es claramente una forma poderosa de representar datos, no considera ninguna clase y, por lo tanto, se puede perder mucha información discriminativa al desechar componentes. Imagine una situación en la que la variación en sus datos es generada por una fuente externa, que sea la luz. Los componentes identificados por un APC no contienen necesariamente ninguna información discriminativa en absoluto, por lo que las muestras proyectadas se manchan juntas y una clasificación se vuelve imposible.

El análisis discriminante lineal realiza una reducción de dimensionalidad específica de clase y fue inventado por el gran estadístico Sir R. A. Fisher. Lo utilizó con éxito para clasificar flores en su artículo de 1936 El uso de múltiples medidas en problemas taxonómicos. Para encontrar la combinación de características que separa mejor entre clases, el Análisis discriminante lineal maximiza la proporción de la dispersión entre clases y dentro de las clases, en lugar de maximizar la dispersión general. La idea es simple: las mismas clases deben agruparse muy juntas, mientras que las diferentes clases están lo más alejadas posible entre sí en la representación de dimensiones inferiores. Esto también fue reconocido por Belhumeur, Hespanha y Kriegman por lo que aplicaron un Análisis Discriminante al reconocimiento facial.

LBPHFaces

Histogramas de patrones binarios locales

Es una técnica descriptiva simple pero altamente efectiva para la clasificación de objetos dentro de la visión por computador que filtra los píxeles adyacentes

mediante consideraciones determinadas y obtiene un valor binario representativo. Debido a su elevada capacidad discriminatoria, constituye una aproximación usual para la solución de multitud de problemas. Probablemente una de sus características más importantes es la robustez de su invariante ante variaciones lumínicas.

Esta técnica proporciona un operador de análisis de textura que se define como una medida de la textura en una escala de grises invariante, derivado de una definición general de textura mediante vecinos locales. La forma actual del operador LBP es muy diferente de su versión básica: la definición original se extiende a un conjunto de vecinos arbitrarios circulares, y se han desarrollado nuevas versiones de este, sin embargo, la idea principal es la misma: un código binario que describe el patrón de la textura local que es construido por el umbral de un conjunto de vecinos por el valor de gris de su centro.

Qué tipo de entrenamiento se utiliza (Tabla Testing, LOOCV, K-Folds).

La validación cruzada es principalmente utilizada para estimar la habilidad de un modelo en información que no ha visto. Esto es, utilizar un data sample limitado para estimar como es que el modelo funcionara en general para hacer predicciones sobre datos que no fueron utilizados en el entrenamiento.

Utilizamos K-Fold, debido a nuestro dataset, que tiene un tamaño aproximado de 700 imágenes y a la hora de evaluar lo usaríamos en data que no se usó en el entrenamiento.

En la validación cruzada de k-iteraciones, los datos de muestra se dividen en K subconjuntos, uno de estos subconjuntos se utiliza como datos de prueba y el resto como datos de entrenamiento. El proceso de k-fold es repetido durante las k-iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Y, por último, realizamos la media aritmética de los resultados de cada iteración para obtener un único resultado.

Procedimiento general:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Realice análisis comparativo de máquinas de machine learning con aprendizaje supervisado (al menos tres máquinas), compare la precisión del modelo y su razón de error.

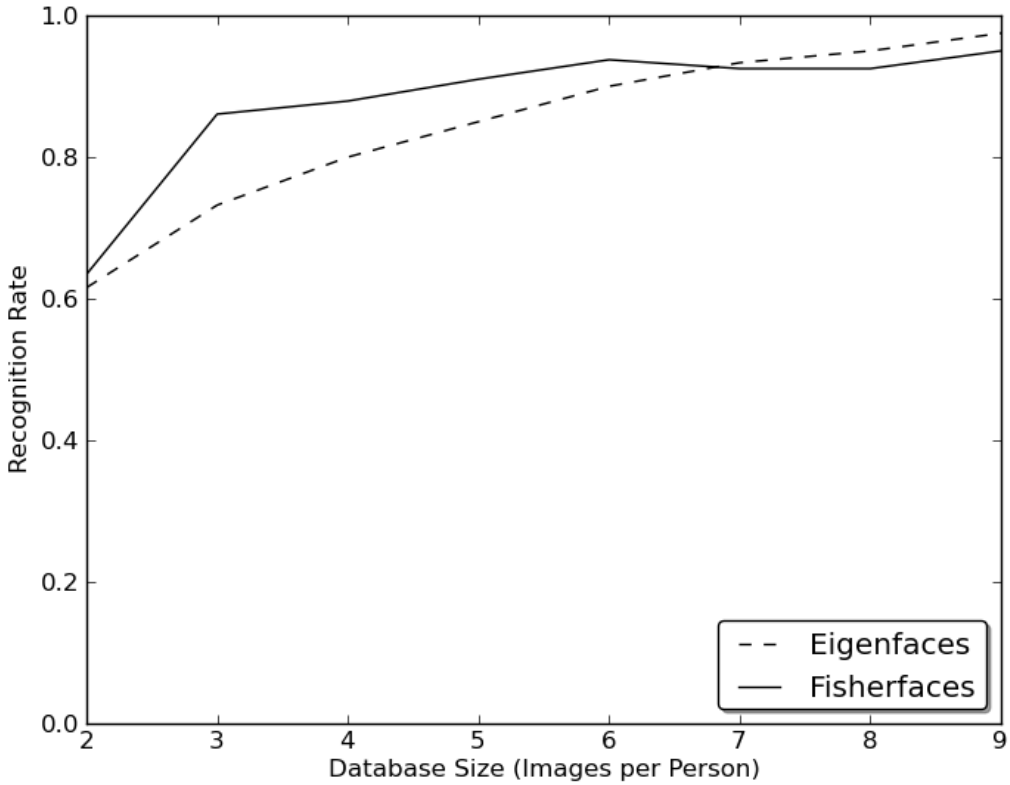
Eigenfaces y Fisherfaces tienen un enfoque holistico para el reconocimiento facial. Tratan los datos como un vector situado en algún lugar del espacio de una imagen de alta-dimension.

Un espacio de alta dimensión no es bueno, por eso, identificamos espacios de menor dimensión, donde probablemente la información que obtengamos sea más útil.

El enfoque en Eigenfaces es maximizar la dispersión total, que nos puede llevar a problemas si la varianza es generada por una fuente externa, por que los componentes con una varianza máxima sobre todas las clases no son necesariamente útiles para la clasificación.

Ahora bien, en la vida real nuestra información es imperfecta, entonces no podemos garantizar la composición de luces perfecta en todas las imágenes, y si solo hay una imagen por persona, nuestros estimados de covarianza posiblemente estén mal, y por lo tanto nuestro reconocimiento.

El método Eigenfaces presento un accuracy del 96% mientras que Fisherfaces rondaba por los 94%: (tabla en siguiente página)

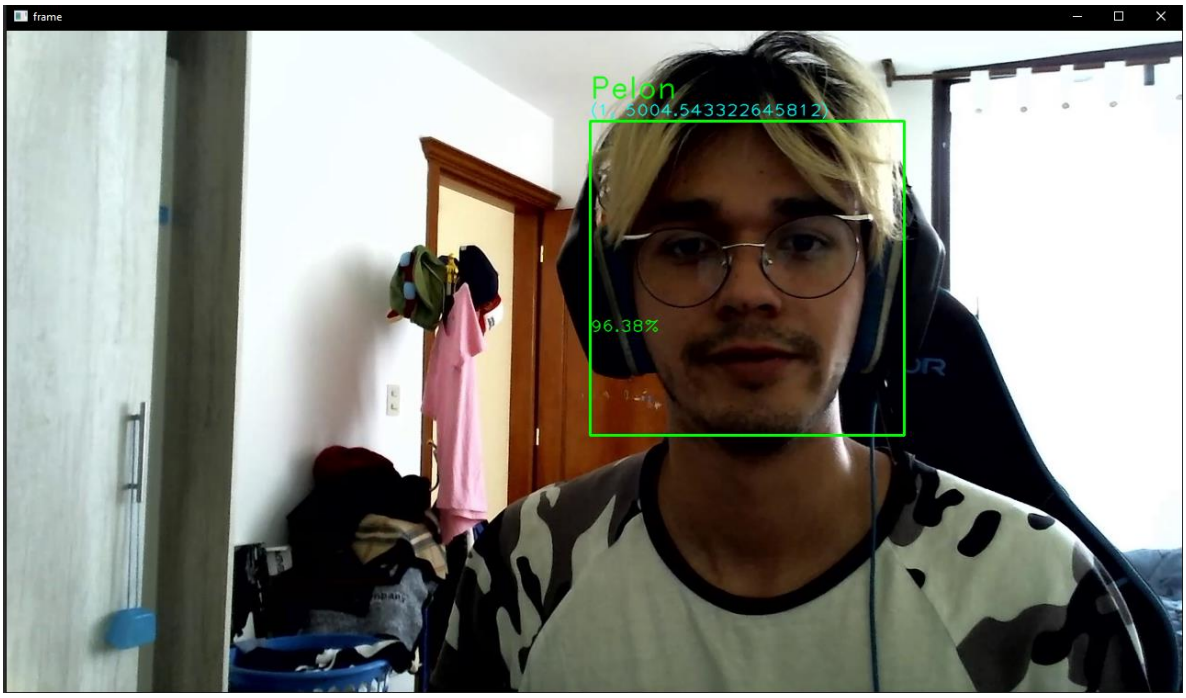


Como podemos observar Fisherfaces presenta un mejor desempeño al principio, pero conforme vamos aumentando el número de imágenes en el dataset por persona Eigenfaces es mejor.

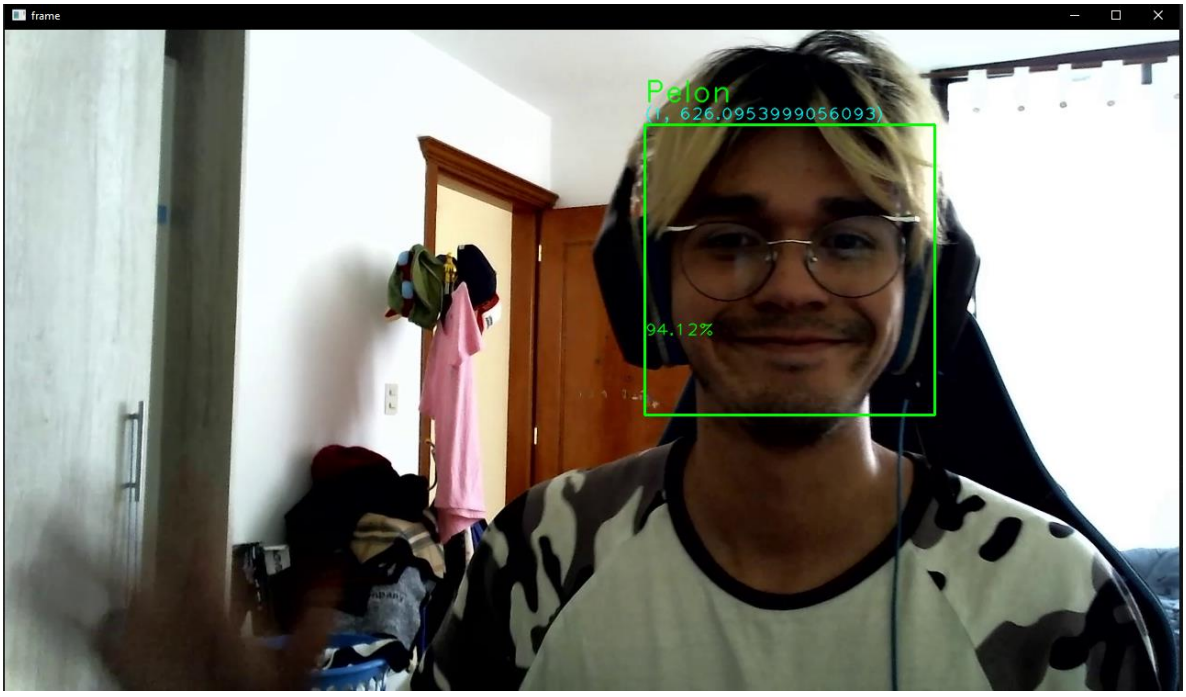
Para LBPHFaces mejoramos las debilidades de estos 2 ultimos y obtuvimos un porcentaje del 98%.

Algoritmo	Precisión	Error
EigenFaces	96%	4%
FisherFaces	94%	6%
LBPHFaces	98%	2%

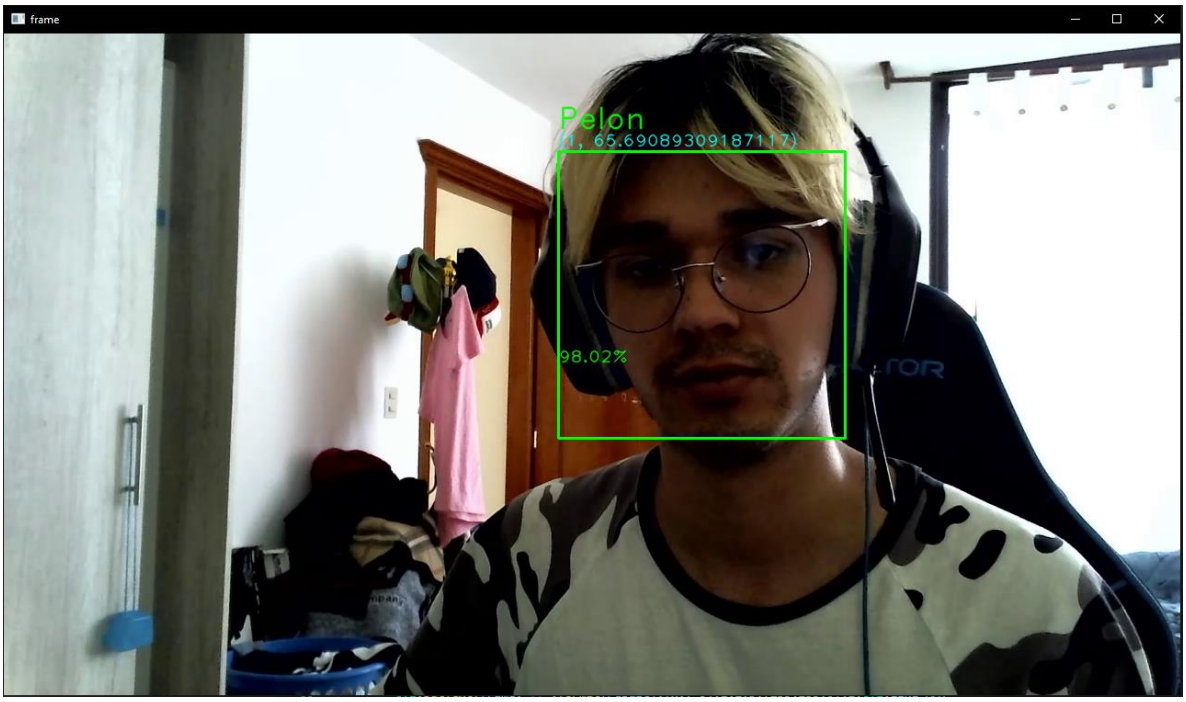
EigenFaces



FisherFaces



LBPHFaces



Evidencias

```python
import cv2
import os
import imutils

personName = 'Pelon'
dataPath = 'C:/Users/Lenny/Desktop/yoyoni/6to/Lunita/Proyecto final AI/data'
personPath = dataPath + '/' + personName
print(personPath)
if not os.path.exists(personPath):
    print('Carpeta creada: ', personPath)
    os.makedirs(personPath)

#cap = cv2.VideoCapture('Paublo.mp4')
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
```

```python
#cap = cv2.VideoCapture('Paublo.mp4')
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
count = 300

while True:
    ret, frame = cap.read()
    if ret == False: break
    frame = imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(gray, 1.3,5)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
        cv2.imwrite(personPath + '/rostro_{}.jpg'.format(count),rostro)
        count = count + 1
    cv2.imshow('frame', frame)

    k = cv2.waitKey(1)
    if k == 27 or count >= 400:
        break

cap.release()
cv2.destroyAllWindows()
```

entrenamiento.py > ...

```python
import cv2
import os
import numpy as np

dataPath = 'C:/Users/Lenny/Desktop/yoyoni/6to/Lunita/Proyecto final AI/data'
peopleList = os.listdir(dataPath)
print('Lista de personas: ', peopleList)


labels = []
facesData = []
label = 0

for nameDir in peopleList:
    personPath = dataPath + '/' + nameDir
    print('Leyendo imagenes')

    for fileName in os.listdir(personPath):
        print('Rostros: ', nameDir + '/' + fileName)
        labels.append(label)
        facesData.append(cv2.imread(personPath + '/' + fileName,0))
        image = cv2.imread(personPath + '/' + fileName,0)
        #cv2.imshow('image', image)
        #cv2.waitKey(10)
    label = label + 1
```

```
            #cv2.waitKey(10)
        label = label + 1

    print('labels= ',labels)
    print('Número de etiquetas 0: ',np.count_nonzero(np.array(labels)==0))
    print('Número de etiquetas 1: ',np.count_nonzero(np.array(labels)==1))

    #Entrenamos EigenFaces
    #face_recognizer = cv2.face.EigenFaceRecognizer_create()
    #face_recognizer = cv2.face.FisherFaceRecognizer_create()
    face_recognizer = cv2.face.LBPHFaceRecognizer_create()

    print("Entrenamos")
    face_recognizer.train(facesData, np.array(labels))

    #Almacenamos el modelo
    #face_recognizer.write('modeloEigenface.xml')
    #face_recognizer.write('modeloFisherface.xml')
    face_recognizer.write('modeloLBPHface.xml')

    print("Modelo almacenado")

    #cv2.destroyAllWindows()
```

reconocimientoFacial.py > ...

```
import cv2
import os

import math

def face_distance_to_conf(face_distance, face_match_threshold=0.67):
    if face_distance > face_match_threshold:
        range = (1.0 - face_match_threshold)
        linear_val = (1.0 - face_distance) / (range * 2.0)
        return linear_val
    else:
        range = face_match_threshold
        linear_val = 1.0 - (face_distance / (range * 2.0))
        return linear_val + ((1.0 - linear_val) * math.pow((linear_val - 0.5) * 2, 0.2))



dataPath = 'C:/Users/Lenny/Desktop/yoyoni/6to/Lunita/Proyecto final AI/data'
imagePaths = os.listdir(dataPath)
print('imagePaths=', imagePaths)

#face_recognizer = cv2.face.EigenFaceRecognizer_create()
#face_recognizer = cv2.face.FisherFaceRecognizer_create()
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
```

```python
#Leemos el modelo
#face_recognizer.read('modeloEigenface.xml')
#face_recognizer.read('modeloFisherface.xml')
face_recognizer.read('modeloLBPHface.xml')


#cap = cv2.VideoCapture('Videos/Pelon2.mp4')
#cap = cv2.VideoCapture('Pelon.mp4')
#cap = cv2.VideoCapture('Videos/paublo2.mp4')
cap = cv2.VideoCapture('Videos/Pelon4.mp4')
#cap = cv2.VideoCapture('Videos/Jordan.mp4')
#cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)



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

while True:
    ret,frame = cap.read()
    if ret == False: break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = gray.copy()

    faces = faceClassif.detectMultiScale(gray,1.3,5)
```

```python

    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.predict(rostro)

        cv2.putText(frame,'{}'.format(result),(x,y-5),1,1.3,(255,255,0),1,cv2.LINE_AA)
        '''
        #EigenFaces
        if result[1] < 5500:
            acc = round(face_distance_to_conf(result[1])*-1,2)
            cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
            cv2.putText(frame,'{}%'.format(acc),(x,y+230),1,1.3,(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-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
            cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)

        #FisherFace
        if result[1] < 800:
            acc = round(face_distance_to_conf(result[1])*-1,2)
            cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
            cv2.putText(frame,'{}%'.format(acc),(x,y+230),1,1.3,(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-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
            cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        '''
```

```
        #LBPHFace
        if result[1] < 70:
            acc = round(face_distance_to_conf(result[1])*-1,2)
            cv2.putText(frame,'{}'.format((imagePaths[result[0]])),(x,y-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
            cv2.putText(frame,'{}%'.format(acc),(x,y+230),1,1.3,(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-25),2,1.1,(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()
```

Conclusión

Este trabajo fue bastante divertido de desarrollar y un poco pesado para mi computadora, el modelo de Eigenfaces casi pesaba medio gigabyte entonces cuando lo cargaba mi computadora sufría bastante. El trabajo se logró con éxito y aprendí mucho sobre OpenCV que es una biblioteca libre de visión artificial.

Link al proyecto:
https://drive.google.com/file/d/1BKcX9E8S4atqA4bMY8iQmqYNYGGwdX3y/view?usp=sharing

Referencias Bibliográficas

Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face recognition with local binary patterns. In Computer vision-eccv 2004, pages 469–481. Springer, 2004.

Cesar Troya S.. (2016). LBP y ULBP – Local Binary Patterns Y Uniform Local Binary Patterns. 28/05/2021, de Universidad Autonoma de Barcelona Sitio web: https://cesartroyasherdek.wordpress.com/2016/02/26/deteccion-de-objetos-vi/

Peter N. Belhumeur, João P Hespanha, and David Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(7):711–720, 1997.

Joaquín Amat Rodrigo. (2020). Machine learning con Python y Scikit-learn. 28/05/2021, de cienciadedatos.net Sitio web: https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikit learn.html#Pipeline-y-ColumnTransformer

Sukhad Anand. (2019). Tutorials for face module. 28/05/2021, de OpenCV Sitio web: https://docs.opencv.org/4.2.0/de/d27/tutorial_table_of_content_face.html