

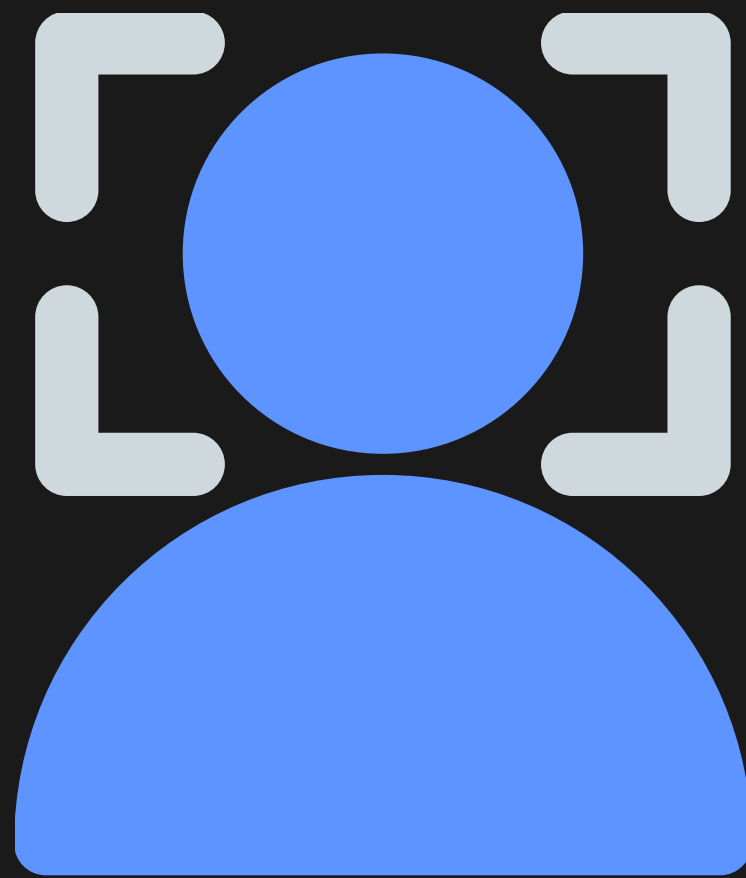
RECONOCIMIENTO FACIAL

Ontiveros Ojeda Eliel Alfonso

Velazquez Mercado Cesar Alejandro

INTRODUCCIÓN

El reconocimiento facial ha revolucionado la forma en que interactuamos con la tecnología y ha encontrado aplicaciones en una amplia variedad de campos, desde la seguridad hasta la autenticación de usuarios, la automatización de tareas y la personalización de servicios. Este programa está diseñado para identificar y reconocer rostros en imágenes de validación, comparándolos con una imagen de referencia tomada de una carpeta de entrenamiento.

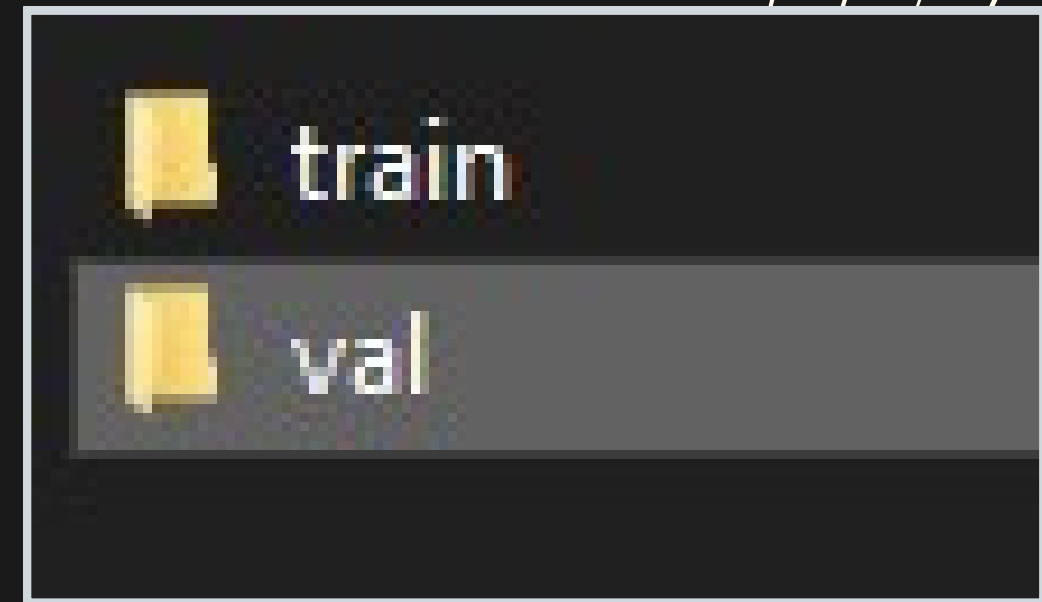


¿COMO FUNCIONA?

1.- Cargamos una imagen de referencia desde la carpeta de entrenamiento, que servirá como punto de comparación.

2.- Luego, exploramos la carpeta de validación, que contiene las imágenes que queremos evaluar.

3.- Para cada imagen en la carpeta de validación, utilizamos OpenCV para detectar rostros y luego aplicamos algoritmos de comparación facial para determinar si la persona en la imagen de validación es la misma que en la imagen de referencia.



¿COMO FUNCIONA?

```
import os
import cv2 as cv
import numpy as np

people = ['Jack Black', 'Johnny Depp', 'Keanu Reeves']
DIR = r
'C:\Users\Eliel\OneDrive\Documentos\Universidad\Lenguaje_C\Lenguaje_
C\Actividades\Caras\train'

haar_cascade = cv.CascadeClassifier('haar_face.xml')

features = []
labels = []

def create_train():
    for person in people:
        path = os.path.join(DIR, person)
        label = people.index(person)

        for img in os.listdir(path):
            img_path = os.path.join(path, img)

            img_array = cv.imread(img_path)
            if img_array is None:
                continue

            gray = cv.cvtColor(img_array, cv.COLOR_BGR2GRAY)

            faces_rect = haar_cascade.detectMultiScale(gray,
scaleFactor=1.1, minNeighbors=4)

            for (x,y,w,h) in faces_rect:
                faces_roi = gray[y:y+h, x:x+w]
                features.append(faces_roi)
                labels.append(label)

create_train()
print('Training done -----')


features = np.array(features, dtype='object')
labels = np.array(labels)

face_recognizer = cv.face.LBPHFaceRecognizer_create()

# Train the Recognizer on the features list and the labels list
face_recognizer.train(features, labels)

face_recognizer.save('face_trained.yml')
np.save('features.npy', features)
```

Lo primero que tenemos que hacer es compilar nuestro programa llamado `face_trained`. Este programa es el encargado de escanear las imágenes dentro de la carpeta `train` las cuales nos servirán más adelante para compararlas después.



Jack Black

Johnny Depp

Keanu Reeves

```
PS C:\Users\costco\Documents\codigo c\opencv presentacion> & C:/Users/costco/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/costco/Documents/codigo c/opencv presentacion/face_trained.py"
● Training done -----
○ PS C:\Users\costco\Documents\codigo c\opencv presentacion> █
```


¿COMO FUNCIONA?

```
import numpy as np
import cv2 as cv

haar_cascade = cv.CascadeClassifier('haar_face.xml')

people = ['Jack Black', 'Johnny Depp', 'Keanu Reeves']

face_recognizer = cv.face.LBPHFaceRecognizer_create()
face_recognizer.read('face_trained.yml')

img = cv.imread(r'
'C:\Users\Elie1\OneDrive\Documentos\Universidad\Lengua
je_C\Lenguaje_C\Actividades\Caras\val\Keanu Reeves
\2.jpg')

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Person', gray)

# Detect the face in the image
faces_rect = haar_cascade.detectMultiScale(gray, 1.1,
4)

for (x,y,w,h) in faces_rect:
    faces_roi = gray[y:y+h,x:x+w]

    label, confidence = face_recognizer.predict(
faces_roi)
    print(f'Label = {people[label]}
with a confidence of {confidence}')

    cv.putText(img, str(people[label]), (20,20), cv.
FONT_HERSHEY_COMPLEX, 1.0, (0,255,0), thickness=2)
    cv.rectangle(img, (x,y), (x+w,y+h), (0,255,0),
thickness=2)

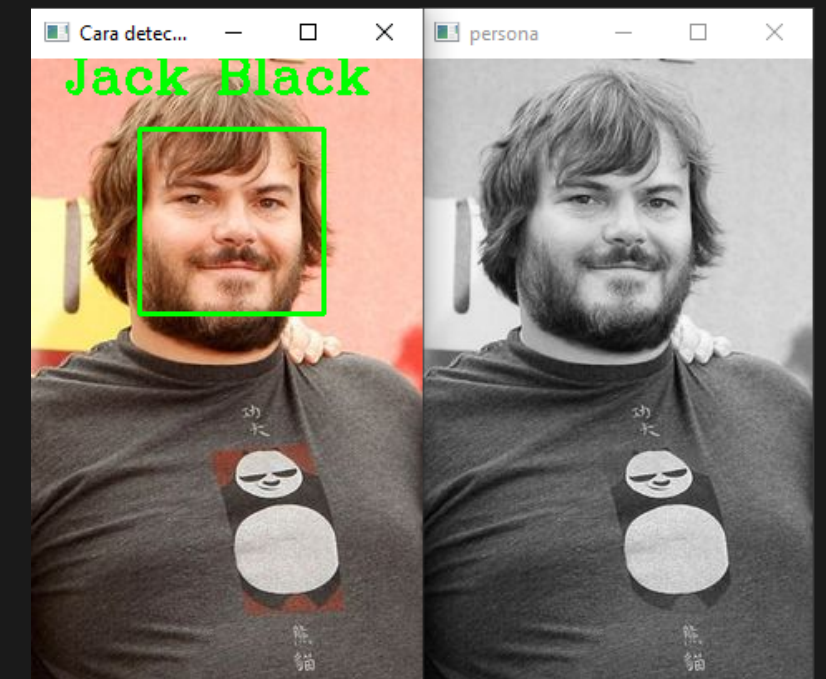
cv.imshow('Detected Face', img)

cv.waitKey(0)
```

Al copilar el programa `face_recognition` este se encarga de comparar la imagen que le mandamos con las encontradas en el programa `face_trained`

Al final despliega el porcentaje de que tan seguro esta de que es la persona que dice ser y también muestra la imagen que mandamos con el nombre de la persona

`Sello = Jack Black con una confianza 75.81050330200591`



APLICACIONES



Control de acceso a edificios, habitaciones o sistemas

Detección de caras no autorizadas en áreas de alta seguridad

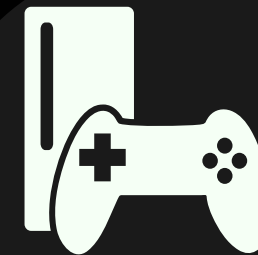
**SEGURIDAD Y
CONTROL DE ACCESO**



Reconocimiento facial para desbloquear dispositivos

Etiquetado automático de fotos en redes sociales

**AUTOMATIZACIÓN
DE TAREAS**



Personalización de Personajes: Permiten personalizar los personajes de los videojuegos de manera más realista.

Seguimiento de Movimientos Faciales: Esto se traduce en un control de juegos más inmersivo

VIDEOJUEGOS

CONCLUSIÓN

Nuestro programa de reconocimiento facial realizado con OpenCV sirve como una introducción a la aplicación del procesamiento de imágenes y comprensión de rostros. Mediante la utilización de algoritmos y un clasificador de rostros (El cual mandamos imágenes para entrenarlo), hemos conseguido comparar rostros y mostrar el porcentaje de similitudes con la imagen que mandamos a analizar.

