

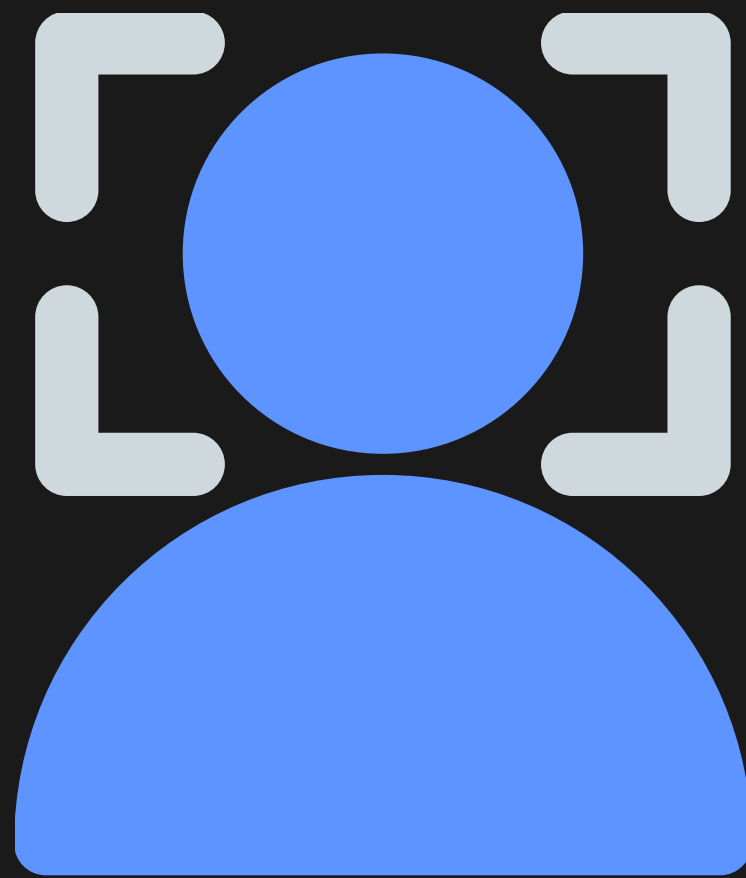
RECONOCIMIENTO FACIAL (CONTROL DE ACCESO)

Ontiveros Ojeda Eliel Alfonso

Velazquez Mercado Cesar Alejandro

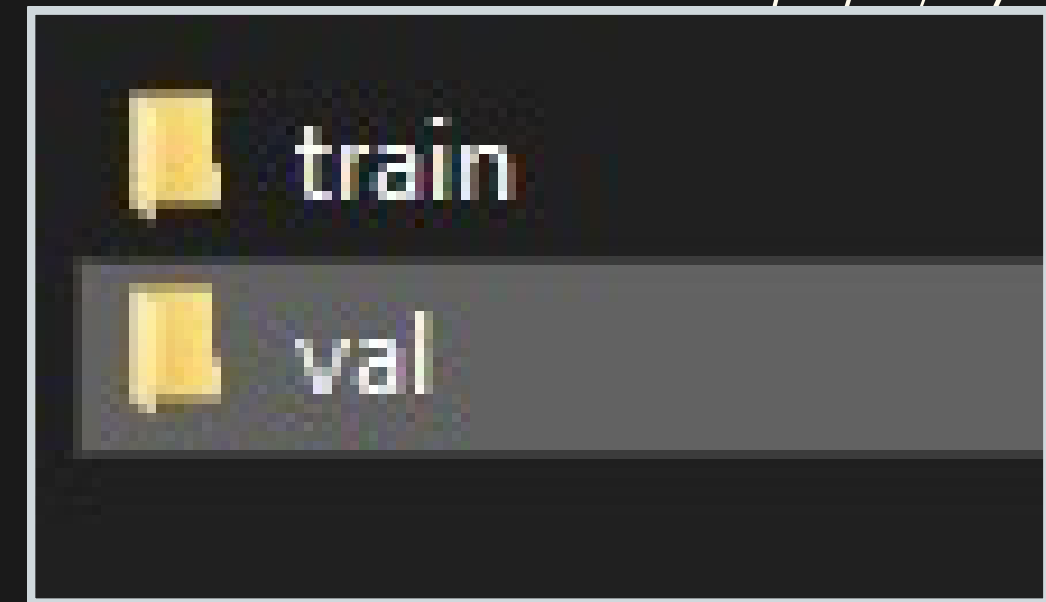
INTRODUCCIÓN

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 los cuales son detectados utilizando la cámara y comparándolos con imágenes de referencia tomada de una carpeta de entrenamiento.



ANTERIORMENTE

- 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.



ANTERIORMENTE

```
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> █
```


ANTERIORMENTE

```
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\Eliel\OneDrive\Documentos\Universidad\Lenguaje_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)

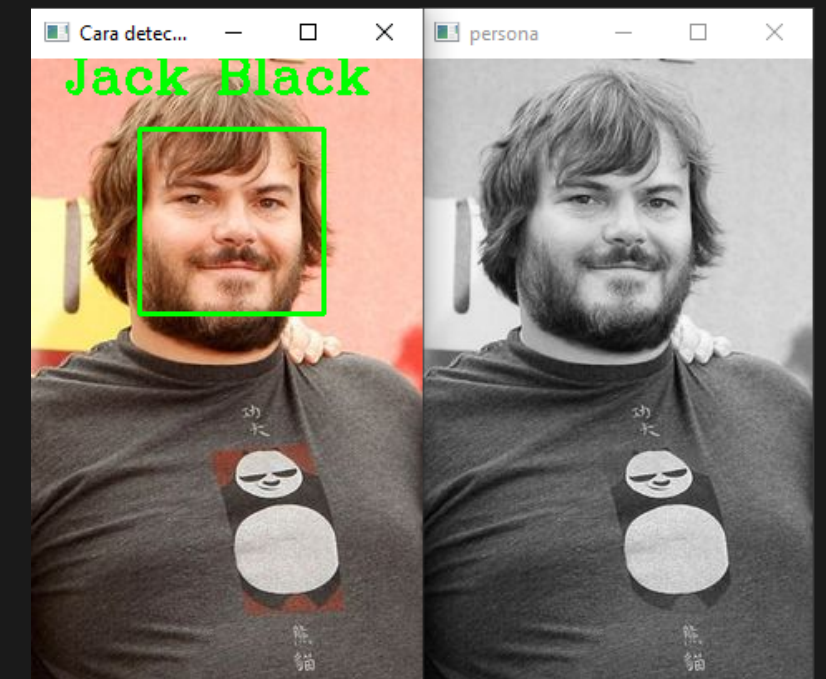
cv.imshow('Detected Face', img)

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 75.810940130200921



MEJORAS

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

people = []
DIR = r
'C:/Users/Eliei/OneDrive/Documentos/Universidad/Lenguaje_C/Lenguaje_C/Actividades/Caras/
train'

haar_cascad = cv.CascadeClassifier('haar_face.xml')
features = []
labels = []

def create_train():
    global people

    for person in os.listdir(DIR):
        if os.path.isdir(os.path.join(DIR, person)) and person not in people:
            print(f'Entrenando para: {person}')
            people.append(person)

        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_rec = haar_cascad.detectMultiScale(gray, scaleFactor=1.1,
minNeighbor_t4)
            e
            e

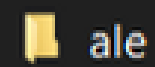
            for (x, y, w, h) in faces_rec:
                faces_roi = gray[y:y+h, x:x+w]
                features.append(faces_roi)
                labels.append(label)

create_train()
print('Entrenamiento Finalizado -----')

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

face_recognize = cv.face.LBPHFaceRecognizer_create()
r
()
face_recognize.train(features, labels)
r
face_recognize.save('face_trained.yml')
np.save('features.npy', features)
np.save('labels.npy', labels)
```

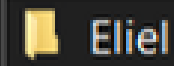
Lo primero que tenemos que hacer es compilar nuestro programa llamado `face_recognition` que a su vez manda a llamar a `face_trained`. Este programa es el encargado de escanear las imágenes dentro de la carpeta train las cuales nos servirán más adelante para compararlas después.



ale

29/11/2023 02:26 p. m.

Carpeta de archivos



Eliei

28/11/2023 11:50 p. m.

Carpeta de archivos

MEJORAS

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

people = []
DIR = r
'C:/Users/Eliei/OneDrive/Documentos/Universidad/Lenguaje_C/Lenguaje_C/Actividades/Caras/
train'

haar_cascad = cv.CascadeClassifier('haar_face.xml')
e
l'
features = []
labels = []

def create_trai():
    global people

    for person in os.listdir(DIR):
        if os.path.isdir(os.path.join(DIR, person)) and person not in people:
            print(f'Entrenando para: {person}')
            people.append(person)

        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)
            y
            faces_rec = haar_cascad.detectMultiScal(gray, scaleFactor=1.1,
minNeighbor t4)
e
e
s
            for (x, y, w, h) in faces_rec:
                faces_roi = gray[y:y+h, x:x+w]
                features.append(faces_roi)
                labels.append(label)

create_trai()
print('Entrenamiento Finalizado -----')

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

face_recognize = cv.face.LBPHFaceRecognizer_create
r
()
face_recognize.train(features, labels)
r
face_recognize.save('face_trained.ym')
np.save('features.npy', features)
np.save('labels.npy', labels)
```

`face_trained` se encarga de abrir la cara y escanear las caras que encontro en face trained a la vez nos dice la confianza que hay y nos permite agregar nuevas caras

Vombre = ale con una desconfianza de: 119.68400549956772

Vombre = ale con una desconfianza de: 97.38203435495268

Vombre = ale con una desconfianza de: 95.02514815748889

MEJORAS

De igual manera nuestro programa crea un archivo de texto en el cual mantiene un registro de las personas conocidas que detecte.

```
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:23: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:24: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:25: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
2023-11-29 10:04:26: Cara conocida - Eliel Alfonso
```



```
1 output_folder_registro = 'C:/Users/Eliel/OneDrive/Documentos/Universidad/Lenguaje_C/Lenguaje_C/Actividades/Caras/Registros'
```



```
1 with open(registro_path, 'a') as registro_file:
2     registro_file.write(f'{time.strftime("%Y-%m-%d %H:%M:%S")}: Cara conocida - {people[label]}\n')
```



registro

29/11/2023 02:42 p. m.

Documento de tex...

82 KB

CONCLUSIÓN

Nuestro programa de reconocimiento facial desarrollado con OpenCV constituye una iniciación a la implementación del procesamiento de imágenes y la comprensión facial. Utiliza algoritmos y un clasificador de rostros para activar nuestra cámara y detectar similitudes con las caras almacenadas en la base de datos. Esta tecnología posibilita la ejecución de proyectos de seguridad, como cerraduras inteligentes que se desbloquean al reconocer la cara de usuarios autorizados.



END