

Tarea 3: Reconocimiento facial

Alan Andrés Mérida Morales, 202100023^{1,*}

¹Facultad de Ingeniería, Escuela de Ingeniería Mecánica Eléctrica,
Universidad de San Carlos, Ciudad Universitaria, Zona 12, Guatemala.

I. OBJETIVOS

- Instalar OpenCV según el sistema operativo que se vaya a utilizar junto con sus librerías.
- Realizar el registro de distintos rostros realizando el reconocimiento y entrenamiento.
- Comparar los distintos métodos de entrenamiento y determinar el de mejor funcionamiento.

II. INTRODUCCIÓN

Este trabajo presenta una implementación básica de un sistema de reconocimiento facial utilizando Python y OpenCV, que abarca la captura de rostros, el entrenamiento de un modelo y el reconocimiento facial. Para ello, se instalaron OpenCV y las bibliotecas necesarias. Las imágenes faciales se obtuvieron a partir de videos, que sirvieron como registros para el entrenamiento del modelo. Finalmente, se evaluó la efectividad del modelo en el reconocimiento de rostros en tiempo real, brindando una base práctica para futuras investigaciones en el campo del reconocimiento facial.

III. MARCO TEÓRICO

A. Reconocimiento facial

El reconocimiento facial es una subárea del reconocimiento de patrones que se ha convertido en una herramienta clave en diversas aplicaciones, desde sistemas de seguridad y autenticación hasta la interacción humano-computadora. Su popularidad se debe a su carácter no invasivo y a la precisión que puede alcanzar con tecnologías modernas. Los sistemas de reconocimiento facial se dividen en dos etapas principales: la detección de rostros y el reconocimiento propiamente dicho. La detección de rostros es el proceso de localizar y extraer regiones faciales en una imagen o video, mientras que el reconocimiento consiste en identificar o verificar la identidad de la persona mediante la comparación de estas regiones con un conjunto de datos previamente almacenados

B. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto diseñada para aplicaciones de visión por computadora y aprendizaje automático. Fue lanzada inicialmente en 2000 por Intel y ha evolucionado para convertirse en una herramienta fundamental en el campo del procesamiento de imágenes y visión por computadora. OpenCV proporciona una amplia gama de funcionalidades para la manipulación de imágenes, detección de objetos, reconocimiento de patrones y análisis de video. Es compatible con múltiples lenguajes de programación como Python, C++, y Java, y puede ejecutarse en diversas plataformas, incluidas Windows, Linux, y macOS.//

OpenCV incluye algoritmos optimizados para realizar tareas complejas como la detección de rostros, segmentación de imágenes, y reconocimiento de objetos. En particular, es ampliamente utilizado en proyectos de reconocimiento facial, gracias a su capacidad para manejar grandes volúmenes de datos y aplicar algoritmos de aprendizaje automático y visión por computadora de manera eficiente.

C. Métodos de Entrenamiento para el Reconocimiento Facial

El reconocimiento facial es una técnica que permite identificar o verificar a una persona mediante la comparación de sus características faciales con un conjunto de imágenes preexistentes. OpenCV ofrece varios métodos para el entrenamiento de modelos de reconocimiento facial, entre los cuales se destacan:

- **EigenFaceRecognizer:** El método EigenFaces es uno de los primeros enfoques exitosos para el reconocimiento facial. Se basa en la descomposición de las imágenes faciales en componentes principales mediante la técnica de Análisis de Componentes Principales (PCA). En este enfoque, cada imagen de rostro es representada como una combinación lineal de componentes base, llamados ".eigenfaces". El reconocimiento se realiza proyectando una nueva imagen sobre este espacio de eigenfaces y comparando su representación con las imágenes almacenadas en la base de datos.

- **FisherFaceRecognizer:** FisherFaces es una mejora del método EigenFaces y se basa en el Análisis Discriminante Lineal (LDA). Mientras que PCA

* 3690273450101@ingenieria.usac.edu.gt

maximiza la varianza total en los datos, LDA maximiza la separabilidad entre las clases (en este caso, diferentes personas). Esto significa que FisherFaces busca representar las imágenes faciales de manera que las diferencias entre las clases (personas) sean más pronunciadas, lo que a menudo resulta en un mejor rendimiento en condiciones de iluminación y expresiones faciales variables.

- **LBPHFaceRecognizer:** Local Binary Patterns Histograms (LBPH) es un enfoque que se centra en capturar patrones locales en las imágenes faciales. A diferencia de EigenFaces y FisherFaces, LBPH es robusto ante variaciones de iluminación y detalles faciales menores. El método funciona dividiendo la imagen facial en pequeñas regiones y describiendo cada una de ellas mediante histogramas de patrones binarios locales. Estos histogramas se combinan para formar un vector de características que se utiliza para la clasificación. LBPH es particularmente efectivo en aplicaciones donde se requiere un alto nivel de robustez en entornos no controlados.

D. Captura de Datos con Videos

En los sistemas de reconocimiento facial, la captura de datos es un paso crítico que afecta directamente la calidad del modelo entrenado. La captura de rostros a partir de videos es una técnica comúnmente utilizada para recolectar múltiples imágenes de una persona en diferentes condiciones, como iluminación, ángulos y expresiones faciales. Esto ayuda a crear un conjunto de datos más diverso y representativo para entrenar el modelo de reconocimiento facial.

El proceso generalmente involucra los siguientes pasos:

- **Detección de Rostros:** Utilizando técnicas como la detección de rostros basada en cascadas de Haar o detectores de HOG (Histogram of Oriented Gradients), se localizan las caras en cada cuadro del video.
- **Detección de Rostros:** Las imágenes de los rostros detectados se extraen y, en muchos casos, se normalizan para tener un tamaño y formato consistentes. El preprocessamiento también puede incluir la conversión a escala de grises, la mejora de contraste, y la eliminación de ruido.
- **Almacenamiento de Imágenes:** Los rostros capturados se almacenan como imágenes individuales en una base de datos o un conjunto de entrenamiento. Esto permite su posterior uso en el entrenamiento del modelo de reconocimiento facial.
- **Entrenamiento del Modelo:** Con el conjunto de datos recopilado, se entrena el modelo utilizando uno de los métodos mencionados anteriormente (EigenFace, FisherFace, LBPH). Este modelo será capaz de identificar o verificar rostros en tiempo real.

La captura de datos con videos ofrece la ventaja de generar un gran volumen de datos en un corto período de tiempo, lo que es esencial para entrenar modelos precisos y robustos. Además, permite capturar las variaciones naturales que ocurren en la apariencia de un rostro, lo que mejora la capacidad del modelo para reconocer a una persona en diferentes condiciones.

IV. RESULTADOS

A. Códigos realizados

1. Capturar rostros

```
11/24, 9:02 p.m.                                     capturandoRostros.py
capturandoRostros.py
1 import cv2
2 import os
3 import imutils
4
5 personName = 'Alan'
6 dataPath = 'C:/Users/Usuario/Desktop/Reconocimiento facial/data' #Cambia a la ruta donde hayas almacenado Data
7 personPath = dataPath + '/' + personName
8
9 if not os.path.exists(personPath):
10     print('Carpetas creadas: ', personPath)
11     os.makedirs(personPath)
12
13 #cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
14 cap = cv2.VideoCapture('C:/Users/Usuario/Desktop/Reconocimiento facial/alan.mp4')
15
16
17 faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
18 count = 300
19
20 while True:
21
22     ret, frame = cap.read()
23     if ret == False: break
24     frame = imutils.resize(frame, width=640)
25     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
26     auxFrame = frame.copy()
27
28     faces = faceClassif.detectMultiScale(gray, 1.3, 5)
29
30     for (x,y,w,h) in faces:
31         cv2.rectangle(frame, (x,y),(x+w,y+h), (0,255,0), 2)
32         rostro = auxFrame[y:y+h,x:x+w]
33         rostro = cv2.resize(rostro, (150,150), interpolation=cv2.INTER_CUBIC)
34         cv2.imwrite(personPath + '/rostro_{}.jpg'.format(count),rostro)
35         count = count + 1
36
37     cv2.imshow('frame',frame)
38
39     k = cv2.waitKey(1)
40     if k == 27 or count >= 400:
41         break
42
43 cap.release()
44 cv2.destroyAllWindows()
```

Figura 1: Código creado para la captura de datos de los rostros.

```
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()
```

localhost:55073/f7366458-b0be-4725-8534-70d88d38c24f/

1/1

Figura 2: Código creado para la captura de datos de los rostros.

2. Entrenamiento

```
11/8/24, 9:03 p.m. entrenandoRF.py
entrenandoRF.py

1 import cv2
2 import os
3 import numpy as np
4
5 dataPath = 'C:/Users/Usuario/Desktop/Reconocimiento facial/data' #Cambia a la ruta donde hayas
almacenado Data
6 peopleList = os.listdir(dataPath)
7 print('Lista de personas: ', peopleList)
8
9 labels = []
10 facesData = []
11 label = 0
12
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))
21         image = cv2.imread(personPath+'/'+fileName,0)
22         #cv2.imshow('image',image)
23         #cv2.waitKey(10)
24         label = label + 1
25
26 #print('labels:',labels)
27 #print('Número de etiquetas 0: ',np.count_nonzero(np.array(labels)==0))
28 #print('Número de etiquetas 1: ',np.count_nonzero(np.array(labels)==1))
29
30 ##Métodos para entrenar el reconocedor
```

Figura 3: Código creado para realizar el entrenamiento de los modelos reconocedores

```
30 ##Métodos para entrenar el reconocedor
31 #face_recognizer = cv2.face.EigenFaceRecognizer_create()
32 #face_recognizer = cv2.face.FisherFaceRecognizer_create()
33 face_recognizer = cv2.face.LBPHFaceRecognizer_create()
34
35 # Entrenando el reconocedor de rostros
36 print("Entrenando...")
37 face_recognizer.train(facesData, np.array(labels))
38
39
40
41 # Almacenando el modelo obtenido
42 #face_recognizer.write('modeloEigenFace.xml')
43 #face_recognizer.write('modeloFisherFace.xml')
44 face_recognizer.write('modeloLBPHFace.xml')
45 print("Modelo almacenado...")
```

Figura 4: Código creado para realizar el entrenamiento de los modelos reconocedores

3. Reconocimiento facial

```
11/8/24, 9:03 p.m. ReconocimientoFacial.py
ReconocimientoFacial.py

1 import cv2
2 import os
3
4 dataPath = 'C:/Users/Usuario/Desktop/Reconocimiento facial/data' #Cambia a la ruta donde hayas
almacenado Data
5 imagePaths = os.listdir(dataPath)
6 print('imagePaths:',imagePaths)
7
8 face_recognizer = cv2.face.EigenFaceRecognizer_create()
9 #face_recognizer = cv2.face.FisherFaceRecognizer_create()
10 #face_recognizer = cv2.face.LBPHFaceRecognizer_create()
11
12 # Leyendo el modelo
13 face_recognizer.read('modeloEigenFace.xml')
14 #face_recognizer.read('modeloFisherFace.xml')
15 #face_recognizer.read('modeloLBPHFace.xml')
16
17 #cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
18 cap = cv2.VideoCapture('C:/Users/Usuario/Desktop/Reconocimiento facial/melanie.mp4')
19
20 faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
21
22 while True:
23     ret,frame = cap.read()
24     if ret == False: break
25     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
26     auxFrame = gray.copy()
27
28     faces = faceClassif.detectMultiScale(gray,1.3,5)
29
30     for (x,y,w,h) in faces:
31         rostro = auxFrame[y:y+h,x:x+w]
32         rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
```

Figura 5: Código creado para probar los modelos y comprobar el reconocimiento

```
32 rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
33 result = face_recognizer.predict(rostro)
34
35 cv2.putText(frame,'{}' .format(result),(x,y-5),1,1,(255,255,0),1,cv2.LINE_AA)
36
37 # EigenFaces
38 # if result[1] < 5700:
39 #     cv2.putText(frame,'{}' .format(imagePaths[result[0]]),(x,y-25),2,1,1,
(0,255,0),1,cv2.LINE_AA)
40 #     cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
41 # else:
42 #     cv2.putText(frame,'Desconocido',(x,y-20),2,0.8,(0,0,255),1,cv2.LINE_AA)
43 #     cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
44
45 # FisherFace
46 #if result[1] < 6000:
```

localhost:55073/cd695e6-14a7-4dd0-9306-210ee26ddb5f 1/2

Figura 6: Código creado para probar los modelos y comprobar el reconocimiento

```
60
61 cv2.imshow('frame',frame)
62 k = cv2.waitKey(1)
63 if k == 27:
64     break
65
66 cap.release()
67 cv2.destroyAllWindows()
```

Figura 7: Código creado para probar los modelos y comprobar el reconocimiento

B. Reconocimiento facial

1. Bases de datos creadas

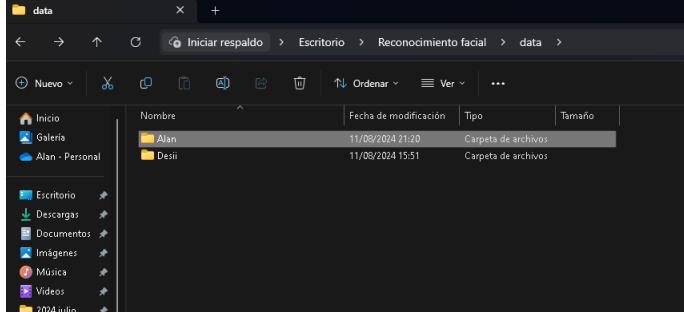


Figura 8: Base de datos creada a partir de los videos

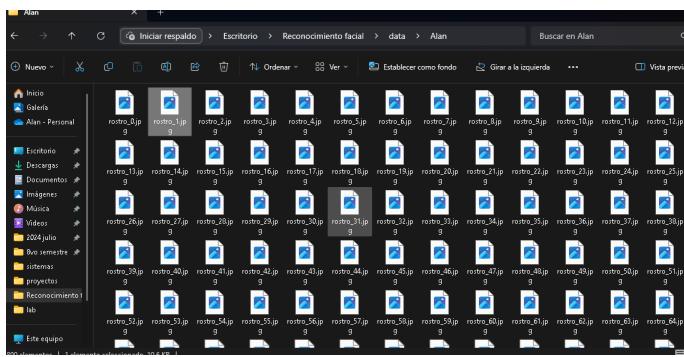


Figura 9: Base de datos creada a partir de los videos

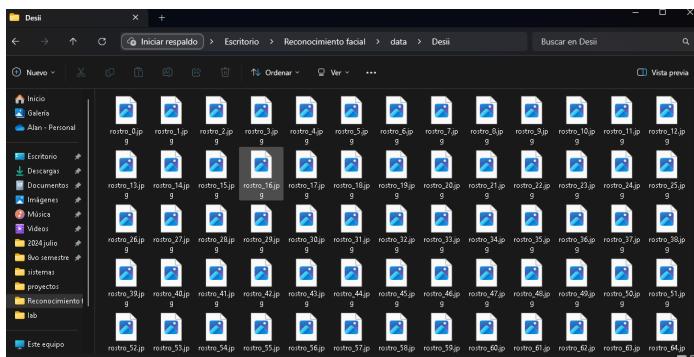


Figura 10: Base de datos creada a partir de los videos

2. Reconocimiento con el método EigenFaces



Figura 11: Identificación correcta de usuario 1 sin audífonos

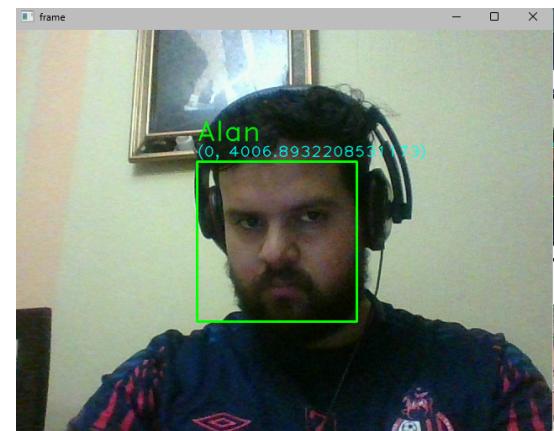


Figura 12: Identificación correcta de usuario 1 con audífonos

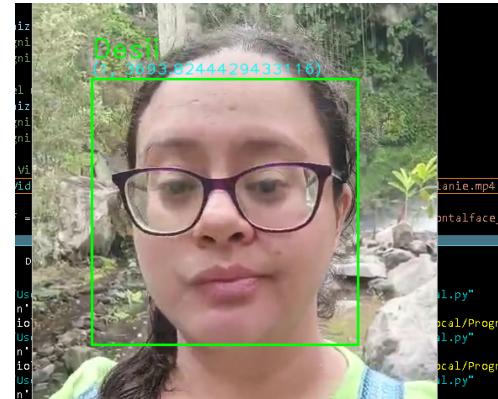


Figura 13: Identificación correcta de usuario 2

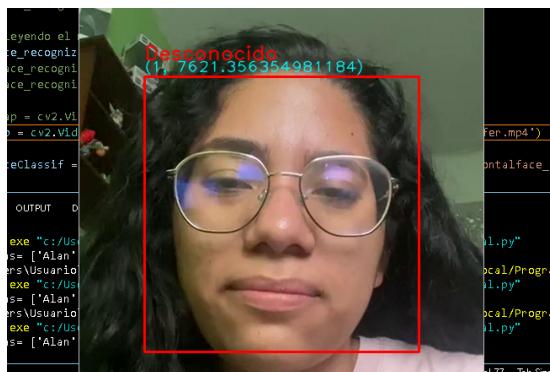


Figura 14: No se identifica al usuario por no tener registro.

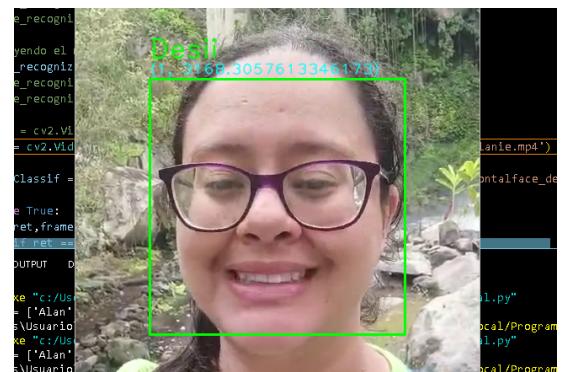


Figura 17: Identificación correcta de usuario 2

3. Reconocimiento con el método FisherFace



Figura 15: Identificación correcta de usuario 1 con audífonos.

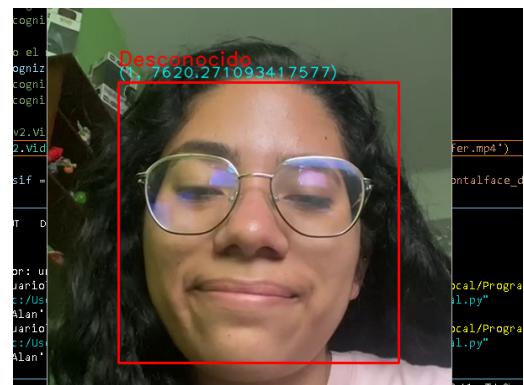


Figura 18: Usuario sin registro no se identifica

4. Reconocimiento con el método LBPHFace

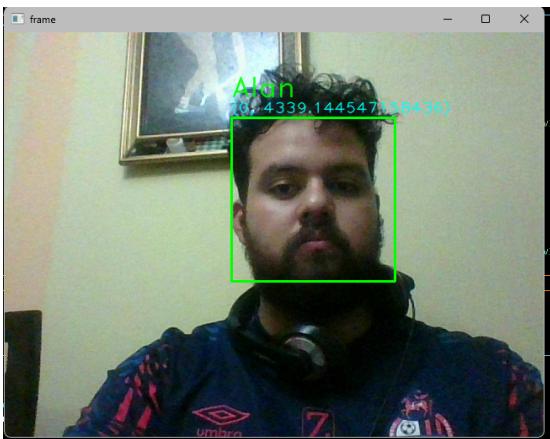


Figura 16: Identificación correcta de usuario 1 sin audífonos.

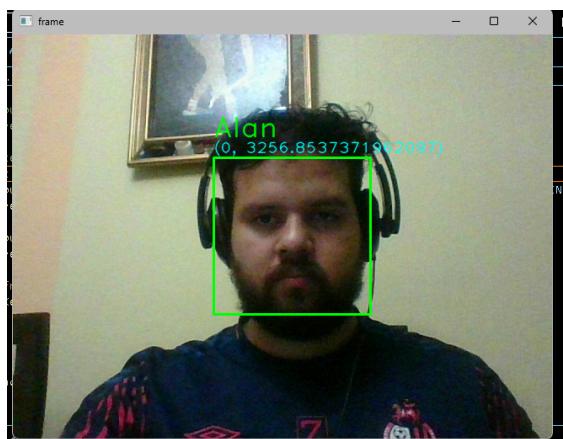


Figura 19: Identificación correcta de usuario 1 con audífonos

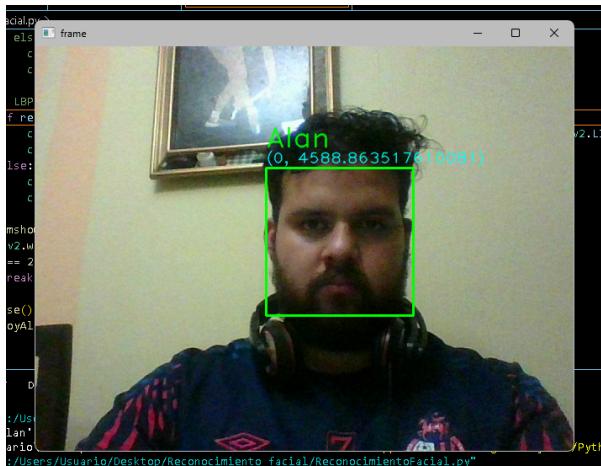


Figura 20: Identificación correcta de usuario 1 sin audífonos

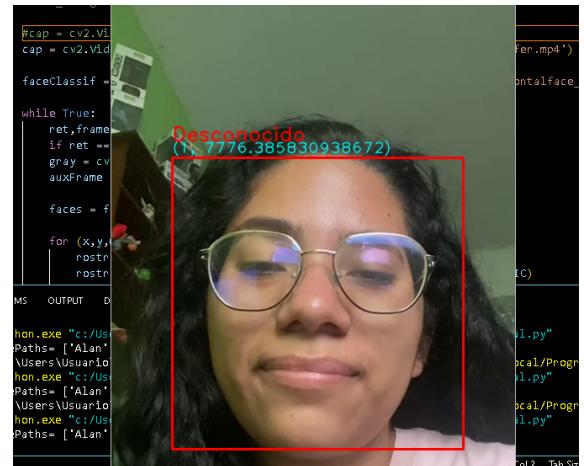


Figura 22: Identificación incorrecta de usuario no registrado.

C. Repositorio privado

Repositorio creado en Github

V. CONCLUSIONES

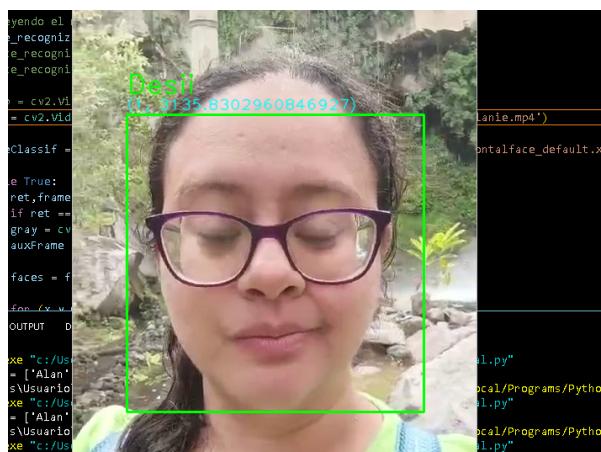


Figura 21: Identificación correcta de usuario 2

- * La instalación de OpenCV y las demás librerías necesarias para realizar el reconocimiento facial fue sencilla y rápida, no fue necesario descargar archivos desde un navegador web, todo se realizó desde la interfaz de líneas de comandos del sistema operativo windows.
- * Al realizar el registro de los rostros en la base de datos se debe realizar una base de datos más robusta de forma de cubrir casi todos los posibles casos de como se va a presentar el rostro y le sea posible reconocerlo.
- * Al momento de hacer el reconocimiento facial un cambio en la luz, en algún objeto alrededor del rostro e incluso una diferencia en el cabello puede confundir al sistema, por lo tanto, se debe de hacer el registro de los rostros en el ambiente donde se va a instalar el software para hacer el reconocimiento.
- * Comparando los 3 métodos de entrenamiento, en mi caso, el que resultó más efectivo sin necesidad de realizarle alguna modificación fue el método EigenFaces. En el caso de los otros dos métodos fue necesario modificar el dato de comparación, sin embargo, el único método que me dio una identificación falsa fue el método de LBPHFace.
- * El registro se debe de realizar con la misma cámara, de esta forma al tener la misma resolución en las imágenes almacenadas se va a reducir el riesgo de una identificación errónea.

```
[1] https://docs.opencv.org/master/
[2] https://es.wikipedia.org/wiki/Reconocimiento_
    facial
[3] https://docs.opencv.org/master/d7/d00/tutorial_
    meanshift.html
[4] https://towardsdatascience.com/
    face-recognition-with-python-using-opencv-d2b2bbd727c9
[5] https://www.geeksforgeeks.org/
    python-face-recognition/
[6] https://www.researchgate.net/publication/
    260743329_Face_Recognition_A_Survey
```