# Tarea No.3: Reconocimiento facial con OpenCV y Python

Marianne Nicté, Rodríguez Canek, 202000656[1, *]

[1] *Facultad de Ingeniería, Departamento de Electronica, Universidad de San Carlos, Ciudad Universitaria, Zona 12, Guatemala.*

La tarea consistio en crear un 3 scrips de python los cuales se encargarian de tomar muestras de los rostros, de hacer el entrenamiento para detectar los rostros y por ultimo crear un programa el cual se encargue de detectar el rostro ya sea de un video o de el uso de la camara de la computadora.

## MARCO TEÓRICO

### OpenCV

Desarrollado inicialmente por Intel, OpenCV es una biblioteca gratuita multiplataforma de visión artificial para el procesamiento de imágenes en tiempo real. El software OpenCV se ha convertido en una herramienta estándar para todo lo relacionado con la visión artificial. Hoy en día, OpenCV sigue siendo muy popular, con más de 29 000 descargas semanales.

Los autores escribieron OpenCV en C y C++. Funciona en los sistemas operativos más populares, como GNU/Linux, OS X, Windows, Android, iOS, etc. Está disponible gratuitamente bajo la licencia Apache 2 .

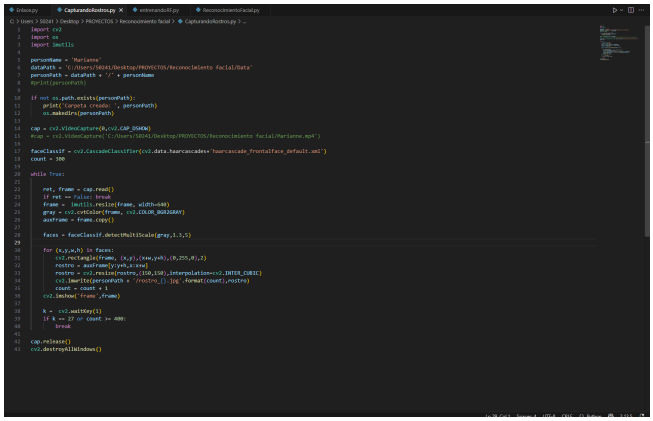
Se está desarrollando activamente la interfaz para Python, Ruby, MATLAB y otros lenguajes. Puedes acceder a ella mediante el comando "pip install opencv"para usuarios de Python y el comando "git opencv"para el control de versiones.

La biblioteca OpenCV contiene más de 2500 algoritmos, documentación extensa, código fuente y código de ejemplo para visión artificial en tiempo real. Los desarrolladores que utilizan el paquete y las bibliotecas de Python pueden integrar OpenCV en sus proyectos con comandos como "python opencv". Los gestores de paquetes facilitan esta integración, lo que simplifica la instalación y el control de versiones.


* e-mail: 3243383091703@ingenieria.usac.edu.gt

## I. RESULTADOS

Se define la ruta de almacenamiento, la cual contiene la carpeta Data, en esta se almacenaran las imagenes que nuestro programa capturara, tomaremos un video de nuestro rostro, realizando distintos gestos y con buena iluminación, esto ayudara a que el reconocimiento sea mucho más sencillo.



```
import cv2
import os
import imutils

personName = 'Marianne'
dataPath = 'C:/Users/50241/Desktop/PROYECTOS/Reconocimiento facial/Data'
personPath = dataPath + '/' + personName
#print(personPath)

if not os.path.exists(personPath):
    print('Carpeta creada: ', personPath)
    os.makedirs(personPath)

cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
#cap = cv2.VideoCapture('C:/Users/50241/Desktop/PROYECTOS/Reconocimiento facial/Marianne.mp4')

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()
```

Figura 1: Código de captura de rostros.

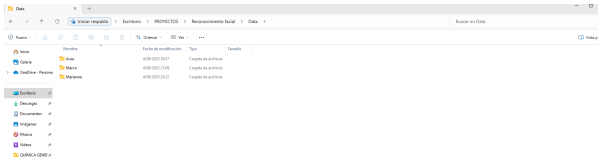


Figura 2: Carpetas de los datos obtenidos.

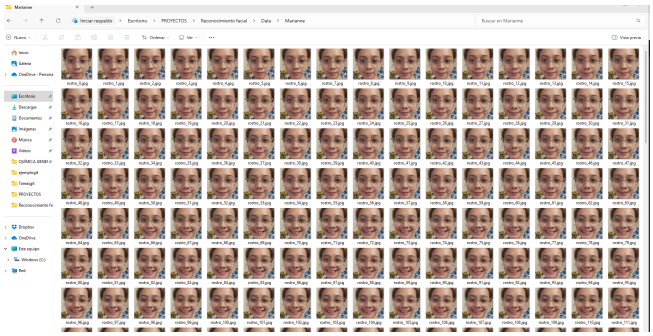


Figura 3: Captura de 300 rostros.

### A. ENTRENAMIENTO DE LOS MODELOS DE CAPTURA DE ROSTROS

Luego de que capturamos las fotos, ahora vamos a proceder a entrenar a nuestro programa para que detecte e indentifique cada uno de los rostros, hay distintos metodos, pero en este caso vamos a utilizar EigenFaceRecognizer, ya que este proporciona un poco mas de precisión y exacto para detectar los rostros

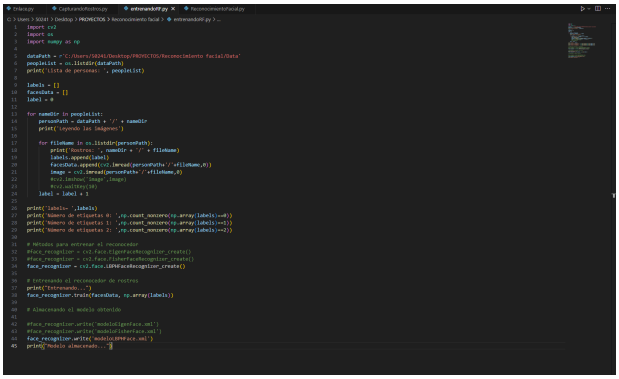


Figura 4: Código para entrenamiento.

### B. RECONOCIMINETO FACIAL

Finalmente luego de entrenarla, ahora podemos proceder a indentificar cada uno de los rosotros que hayamos registrado y con los cuales entrenamos a nuestro programa.

Figura 5: Código para detección de rostro

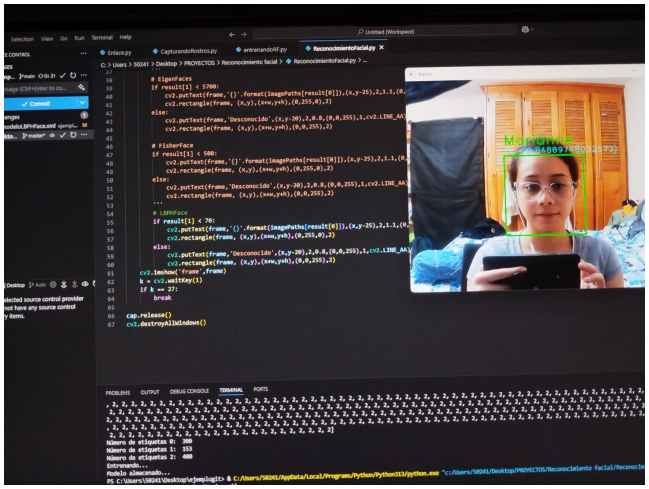


Figura 6:Correcta de detección de rostros.

## II. APLICACIONES

El programa desarrollado puede aplicarse en distintos contextos, entre ellos:

- Sistemas de seguridad basados en reconocimiento facial.
- Generación de bases de datos personalizadas para el entrenamiento de modelos de inteligencia artificial.
- Proyectos relacionados con reconocimiento y procesamiento de imágenes.

En el caso particular de esta tarea, se extrajeron 300 imágenes del video previamente mencionado. Estas imágenes sirven como base para la construcción del sistema de reconocimiento facial. Una mayor cantidad de muestras, especialmente con diferentes expresiones faciales, contribuye a mejorar la precisión y robustez del sistema.

## III. CONCLUSIÓN

- Python, junto con la biblioteca OpenCV, constituye una herramienta poderosa para el desarrollo de soluciones en el ámbito de la visión por computadora. A lo largo de la tarea, se trabajaron conceptos fundamentales como la detección de rostros, el procesamiento de imágenes y el almacenamiento de datos.
- El entorno de desarrollo *Visual Studio Code* resultó ser una opción eficiente, al facilitar la integración de múltiples bibliotecas y permitir una ejecución fluida del código.
- Se optó por emplear el reconocedor *EigenFace*, ya que proporciona mayor precisión al momento de analizar rostros. Aunque existen otras alternativas como *LBPH* o *FisherFace*, estas pueden generar

resultados muy similares entre distintos rostros, lo cual puede dar lugar a errores en la identificación.

Tareas-y-proyectos

## IV. REPOSITORIO EN GITHUB

https://github.com/Marianne8934/