

# PostgreSQL - Octave - Python\*

Emerson Aldair Pérez Rivera, 201902852<sup>1</sup>, \*\*

<sup>1</sup>Facultad de Ingeniería, Universidad de San Carlos,  
Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto diseñada para aplicaciones de visión por computadora y aprendizaje automático. Desarrollada inicialmente por Intel, esta herramienta se ha convertido en un estándar en la industria debido a su robustez, versatilidad y amplia adopción en proyectos de investigación y comerciales. OpenCV ofrece una amplia gama de funciones que permiten a los desarrolladores procesar y analizar imágenes y videos, facilitando tareas como la detección de objetos, reconocimiento de patrones y seguimiento de movimientos.

## I. OBJETIVOS

- Introducirse al uso de OpenCV
- Poder detectar rostros ya sea en imágenes o videos
- Poder Reconocer rostros guardados

## II. CODIGOS UTILIZADOS

### B. Detección Facial - Video

```
1 import cv2 as cv
2 import numpy as np
3
4 cap = cv.VideoCapture(0)
5 face_reco = cv.CascadeClassifier('T3/
6     haarcascade_frontalface_default.xml')
7
8 while True:
9     ret,frame = cap.read()
10    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
11
12    faces = face_reco.detectMultiScale(gray,
13        1.3, 5)
14    cv.imshow('Video', frame)
15    for (x,y,w,h) in faces:
16        cv.rectangle(frame, (x,y), (x+w, y+h),
17        (0,255,0), 2)
18
19    cv.imshow('Reconocimiento', frame)
20    if cv.waitKey(1) & 0xFF == ord('s'):
21        break
22    cap.release()
23    cv.destroyAllWindows()
```

### C. Captura de Imagenes

#### A. Detección Facial - Imagen

```
1 import cv2 as cv
2 import numpy as np
3
4 faceClassif = cv.CascadeClassifier('T3/
5     haarcascade_frontalface_default.xml')
6
7 image = cv.imread('T3/personas.png')
8 gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
9
10 faces = faceClassif.detectMultiScale(gray,
11    scaleFactor=1.1,
12    minNeighbors=6,
13    minSize=(5,5),
14    maxSize=(400,400))
15
16 for (x,y,w,h) in faces:
17    cv.rectangle(image,(x,y),(x+w,y+h),(0,255,0),
18    2)
19
20 cv.imshow('image',image)
21 cv.waitKey(0)
22 cv.destroyAllWindows()
```

```
1 import cv2 as cv
2 import os
3 import imutils
4
5 personName = 'Emerson'
6 dataPath = '/home/emerson/Documentos/
7     data_imagenes'
8 personPath = dataPath + '/' + personName
9
10 if not os.path.exists(personPath):
11     print('Carpetas creadas: ',personPath)
12     os.makedirs(personPath)
13
14 cap = cv.VideoCapture(0)
15 #cap = cv.VideoCapture('/home/emerson/Videos/
16     my_video-1.mkv')
17
18 faceClassif = cv.CascadeClassifier(cv.data.
19     haarcascades+
20     haarcascade_frontalface_default.xml')
21 count = 0
22
23 while True:
24
25     ret, frame = cap.read()
26     if ret == False: break
27     frame = imutils.resize(frame, width=640)
28     gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
29     auxFrame = frame.copy()
```

\* Proyectos de Computación

\*\* g-mail: 3741958620101@ingenieria.usac.edu.gt

```

26     faces = faceClassif.detectMultiScale(gray
27     ,1.3,5)
28
29     for (x,y,w,h) in faces:
30         cv.rectangle(frame, (x,y),(x+w,y+h)
31         ,(0,255,0),2)
32         rostro = auxFrame[y:y+h,x:x+w]
33         rostro = cv.resize(rostro,(150,150),
34         interpolation=cv.INTER_CUBIC)
35         cv.imwrite(personPath + '/rostro_{}.jpg'.
36         format(count),rostro)
37         count = count + 1
38         cv.imshow('frame',frame)
39
40         k = cv.waitKey(1)
41         if k == 27 or count >= 600:
42             break
43
44 cap.release()
45 cv.destroyAllWindows()

```

#### D. Entrenamiento de Modelos EigenFaceRecognizer - LBPHFaceRecognizer

```

1   import cv2 as cv
2 import os
3 import numpy as np
4
5 dataPath = '/home/emerson/Documentos/
       data_imagenes'
6 peopleList = os.listdir(dataPath)
7 dataModelEntrenamiento = '/home/emerson/
       Documentos/data_modelo_entrenamiento'
8 if not os.path.exists(dataModelEntrenamiento):
9     print('Carpeta creada: ',
       dataModelEntrenamiento)
10    os.makedirs(dataModelEntrenamiento)
11    print('Lista de personas: ', peopleList)
12
13 labes = []
14 facesData = []
15 label = 0
16
17 for nameDir in peopleList:
18     personPath = dataPath + '/' + nameDir
19     print('Leyendo las imágenes')
20
21     for fileName in os.listdir(personPath):
22         print('Rostros: ', nameDir + '/' +
23         fileName)
24         labes.append(label)
25         facesData.append(cv.imread(personPath + /
26         +fileName,0))
27         image = cv.imread(personPath + '/' +
28         fileName,0)
29         cv.imshow('image',image)
30         cv.waitKey(10)
31         label = label + 1
32
33 print('labels= ',labes)
34 print('Número de etiquetas 0: ',np.
       count_nonzero(np.array(labes)==0))
35
36 #cv.destroyAllWindows()
37
38 face_recognizerEigenFace = cv.face.
       EigenFaceRecognizer_create()

```

```

35 print('Entrenando Modelo EigenFaceRecognizer...
       ')
36 face_recognizerEigenFace.train(facesData, np.
       array(labes))
37 face_recognizerEigenFace.write(
       dataModelEntrenamiento + '/modeloEigenFace.xml
       ')
38 print('Modelo Almacenado en: ',
       dataModelEntrenamiento + '/modeloEigenFace.xml
       ')
39 print("Modelo de Entrenamiento Terciando...")
40
41
42 face_recognizerLBPHFace= cv.face.
       LBPHFaceRecognizer_create()
43 print('Entrenando Modelo LBPHFaceRecognizer...')
44 face_recognizerLBPHFace.train(facesData, np.
       array(labes))
45 face_recognizerLBPHFace.write(
       dataModelEntrenamiento + '/modeloLBPHFace.xml'
       )
46 print('Modelo Almacenado en: ',
       dataModelEntrenamiento + '/modeloLBPHFace.xml
       ')
47 print("Modelo de Entrenamiento Terciando...")
48
49 # face_recognizerFisher= cv.face.
       FisherFaceRecognizer_create()
50 # print('Entrenando Modelo FisherRecognizer...')
51 # face_recognizerFisher.train(facesData, np.
       array(labes))
52 # face_recognizerFisher.write(
       dataModelEntrenamiento + '/modeloFisher.xml')
53 # print('Modelo Almacenado en: ',
       dataModelEntrenamiento + '/modeloFisher.xml')
54 # print("Modelo de Entrenamiento Terciando...")

```

#### E. Reconocimiento Facial EigenFaceRecognizer

```

1  import cv2 as cv
2 import os
3
4
5
6 dataPath = '/home/emerson/Documentos/
       data_imagenes/' #Cambia a la ruta donde
       hayas almacenado Data
7 imagePaths = os.listdir(dataPath)
8 print('imagePaths=' ,imagePaths)
9
10 face_recognizer = cv.face.
       EigenFaceRecognizer_create()
11
12 # Leyendo el modelo
13 face_recognizer.read('/home/emerson/Documentos/
       data_modelo_entrenamiento/modeloEigenFace.
       xml')
14
15
16 cap = cv.VideoCapture(0)
17 #cap = cv.VideoCapture('/home/emerson/Videos/
       my_video-1.mkv')
18
19 faceClassif = cv.CascadeClassifier(cv.data.
       haarcascades +
       haarcascade_frontalface_default.xml')
20

```

```

21 while True:
22     ret,frame = cap.read()
23     if ret == False: break
24     gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
25     auxFrame = gray.copy()
26
27     faces = faceClassif.detectMultiScale(gray
28                                         ,1.3,5)
29
30     for (x,y,w,h) in faces:
31         rostro = auxFrame[y:y+h,x:x+w]
32         rostro = cv.resize(rostro,(150,150),
33                            interpolation= cv.INTER_CUBIC)
34         result = face_recognizer.predict(rostro)
35
36         cv.putText(frame , '{}'.format(result),(x,
37 y-5),1,1.3,(255,255,0),1,cv.LINE_AA)
38
39         # EigenFaces
40         if result[1] < 5700:
41             cv.putText(frame , '{}'.format(
42 imagePaths[result[0]]),(x,y-25)
43             ,2,1.1,(0,255,0),1,cv.LINE_AA)
44             cv.rectangle(frame, (x,y),(x+w,y+h)
45             ,(0,255,0),2)
46         else:
47             cv.putText(frame , 'Desconocido',(x,y-
48 -20),2,0.8,(0,0,255),1,cv.LINE_AA)
49             cv.rectangle(frame, (x,y),(x+w,y+h)
50             ,(0,0,255),2)
51
52         cv.imshow('frame',frame)
53         k = cv.waitKey(1)
54         if k == 27:
55             break
56
57     cap.release()
58     cv.destroyAllWindows()
59

```

#### F. Reconocimiento Facial LBPHFaceRecognizer

1  
2

### III. MARCO TEORICO

#### OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto para la visión por computadora y el procesamiento de imágenes. Fue desarrollada inicialmente por Intel y ahora es mantenida por la comunidad y varias organizaciones. OpenCV proporciona una amplia gama de herramientas y funciones para tareas de procesamiento de imágenes y visión por computadora.



Figura 1: Logo OpenCV

Fuente: Elaboracion Propia, 2024

#### haarcascade\_frontalface\_default.xml

El archivo *haarcascade\_frontalface\_default.xml* es un clasificador en cascada entrenado para detectar caras en imágenes. Este archivo es parte del conjunto de clasificadores en cascada de Haar de OpenCV, que son modelos de detección de objetos basados en características de Haar. Mas modelos entrenados

### IV. RESULTADOS

#### A. Detección Facial a una Imagen



Figura 2: Imagen Ejemplo

Fuente: Elaboracion Propia, 2024

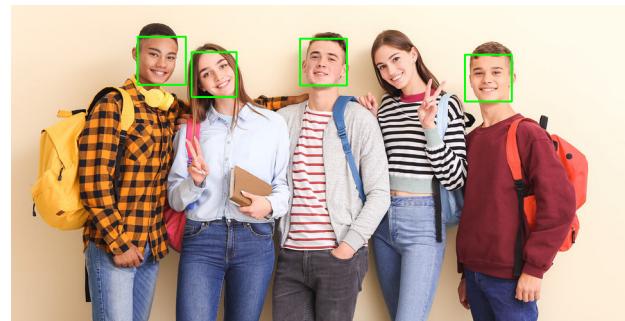


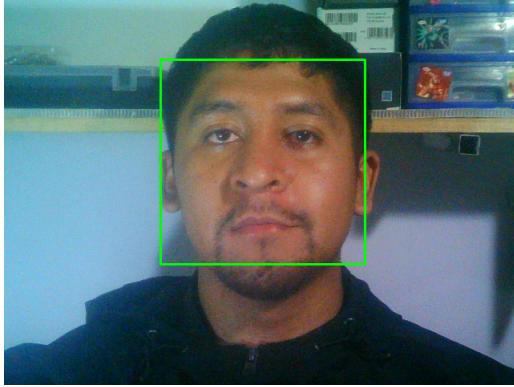
Figura 3: Imagen Ejemplo aplicando el reconocimiento

Fuente: Elaboracion Propia, 2024

## B. Detección Facial en un Video



Fuente: Elaboracion Propia, 2024



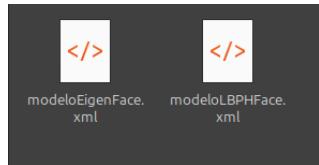
Fuente: Elaboracion Propria, 2024

## C. Modelos de Entrenamiento Generados

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Lista de personas: ['Emerson']
0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Número de etiquetas 0: 600
Entrenando Modelo EigenFaceRecognizer...
Modelo Almacenado en: /home/emerson/Documentos/data_modelo_entrenamiento/modeloEigenFace.xml
Modelo de Entrenamiento Terminado...
Entrenando Modelo LBPHFaceRecognizer...
Modelo Almacenado en: /home/emerson/Documentos/data_modelo_entrenamiento/modeloLBPHFace.xml
Modelo de Entrenamiento Terminado...
venvemerson@LN:~/Documentos/Tareas_Proyectos$ 
```

Figura 6

Fuente: Elaboracion Propria, 2024

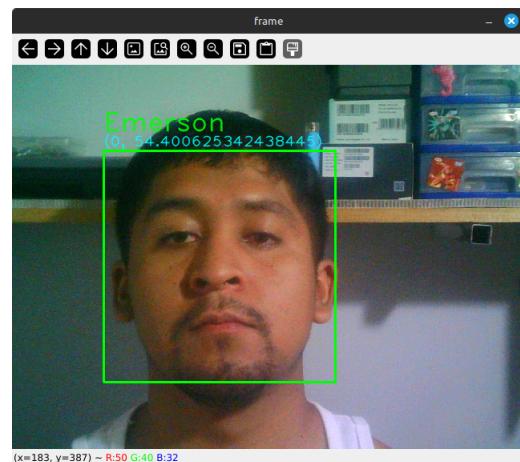


Fuente: Elaboracion Propria, 2024

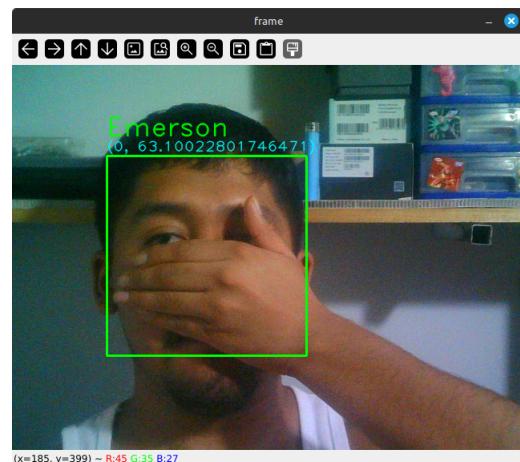


Fuente: Elaboracion Propria, 2024

## D. Reconocimiento Usando EigenFaceRecognizer



Fuente: Elaboracion Propria, 2024



Fuente: Elaboracion Propria, 2024

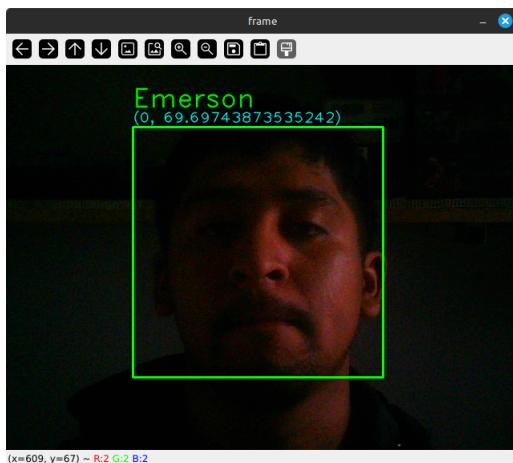


Figura 11: Reconocimiento EigenFaceRecognizer  
Fuente: Elaboracion Propia, 2024



Figura 12: Reconocimiento EigenFaceRecognizer  
Fuente: Elaboracion Propria, 2024



Figura 14: Reconocimiento LBPHFaceRecognizer  
Fuente: Elaboracion Propria, 2024



Figura 15: Reconocimiento LBPHFaceRecognizer  
Fuente: Elaboracion Propria, 2024

#### E. Reconocimiento Usando LBPHFaceRecognizer



Figura 13: Reconocimiento LBPHFaceRecognizer  
Fuente: Elaboracion Propria, 2024

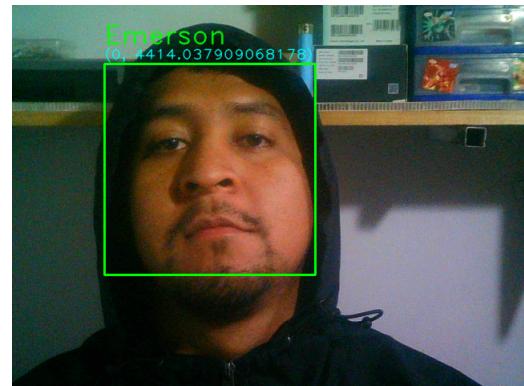


Figura 16: Reconocimiento LBPHFaceRecognizer  
Fuente: Elaboracion Propria, 2024

## V. APLICANDO RECONOCIMIENTO EN VARIAS PERSONAS

### A. Reconocimiento EigenFaceRecognizer



Figura 17: Reconocimiento EigenFaceRecognizer

Fuente: Elaboracion Propia, 2024

### B. Reconocimiento LBPHFaceRecognizer



Figura 18: Reconocimiento LBPHFaceRecognizer

Fuente: Elaboracion Propia, 2024

## VI. CONCLUSIONES

- El uso básico de OpenCV nos ha permitido comprender y aplicar los fundamentos del procesamiento de imágenes y videos de manera eficiente y accesible.
- OpenCV demostró ser una herramienta poderosa en el campo de la visión por computadora, ya que se pudieron detectar rostros con facilidad.
- Se observó que de los 2 modelos entrenados para el reconocimiento facial el método que mejor resultados entregó fue el modelo LBPHFaceRecognizer.

## VII. REPOSITORIO GITHUB

Click para ir al repositorio : [GitHub Emerson8513](#)