

Serie IV

1st Henry Isaac Pineda García,
202001466.

Escuela de Mecánica Eléctrica
Facultad de Ingeniería.
Universidad de San Carlos de Guatemala.
Ciudad de Guatemala, Guatemala.

I. CÓDIGO

I-A. A) Grabador, Reproductor y Graficador de Audio

```
1 % Comprueba si estamos ejecutando en MATLAB o en Octave
2 if (exist('OCTAVE_VERSION', 'builtin') ~= 0)
3     % Estamos en Octave
4     pkg load signal;
5 end
6
7 % Menú principal
8 opcion = 0;
9 while opcion ~= 5
10     disp('Seleccione una opción:');
11     disp('1. Grabar');
12     disp('2. Reproducir');
13     disp('3. Graficar');
14     disp('4. Graficar densidad');
15     disp('5. Salir');
16     opcion = input('Ingrese su elección: ');
17
18     switch opcion
19     case 1
20         % Grabación de audio
21         try
22             duracion = input('Ingrese la duración de la grabación en segundos: ');
23             disp('Comenzando la grabación...');
24             recObj = audiorecorder;
25             recordblocking(recObj, duracion);
26             disp('Grabación finalizada. ');
27             data = getaudiodata(recObj);
28             audiowrite('audio.wav', data, recObj.SampleRate);
29             disp('Archivo de audio grabado correctamente. ');
30         catch
31             disp('Error al grabar el audio. ');
32         end
33     case 2
34         % Reproducción de audio
35         try
36             [data, fs] = audioread('audio.wav');
37             sound(data, fs);
38         catch
39             disp('Error al reproducir el audio. ');
40         end
41     end
42 end
```

Figura 1: Grabador, Reproductor y Graficador de Audio, 1

```
40 end
41 case 3
42     % Gráfico de audio
43     try
44         [data, fs] = audioread('audio.wav');
45         tiempo = linspace(0, length(data)/fs, length(data));
46         plot(tiempo, data);
47         xlabel('Tiempo (s)');
48         ylabel('Amplitud');
49         title('Audio');
50     catch
51         disp('Error al graficar el audio. ');
52     end
53 case 4
54     % Graficando espectro de frecuencia
55     try
56         disp('Graficando espectro de frecuencia...');
57         [audio, Fs] = audioread('audio.wav');
58         N = length(audio);
59         f = linspace(0, Fs/2, N/2+1);
60         ventana = hann(N);
61         Sxx = pwelch(audio, ventana, 0, N, Fs);
62         plot(f, 10*log10(Sxx(1:N/2+1)));
63         xlabel('Frecuencia (Hz)');
64         ylabel('Densidad espectral de potencia (dB/Hz)');
65         title('Espectro de frecuencia de la señal grabada. ');
66     catch
67         disp('Error al graficar el espectro de frecuencia. ');
68     end
69 case 5
70     % Salir
71     disp('Saliendo del programa...');
72 otherwise
73     disp('Opción no válida. ');
74 end
75 end
```

Figura 2: Grabador, Reproductor y Graficador de Audio, 2

I-A1. Comentario y Observaciones: El programa despliega de 5 opciones, que con las cuales podemos grabar, reproducir, realizar gráficas o cerrar el programa.

Al seleccionar la opción de grabar, debemos indicar por cuantos segundos vamos a grabar, para poder tener un control del tiempo limite, el programa crea un archivo de audio en el cual se almacenara la grabación.

La calidad del sonido es baja y al tener un micrófono inalámbrico se puede entender que allá interferencias o ruido que monten al audio.

Su reproducción se da de el ultimo archivo de audio creado, en la primera gráfica se puede observar la modulación que tiene la voz, esta modulación podría ser PCM y la otra gráfica que nos muestra la densidad que llega a tener la grabación mas el ruido.

Un programa muy interesante tanto del lado de la programación como del lado de las comunicaciones digitales.

I-B. B) Entrenador y Reconocimiento Facial

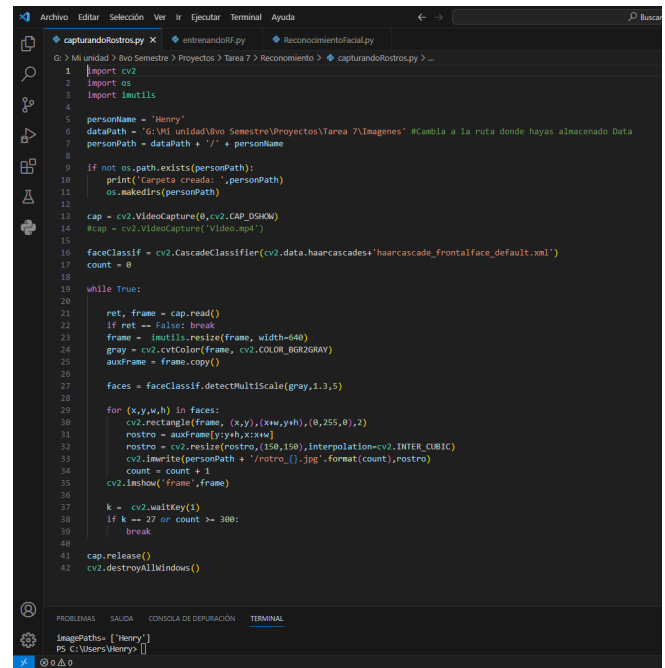


Figura 3: Captura de Rostros

```

Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda
+ capturandoRostros.py  + entrenandoRF.py  + ReconocimientoFacial.py
G: > Mi unidad > 8vo Semestre > Proyectos > Tarea 7 > VideoTutorialesAuxiliatura090-main > Reconocimiento > entrenandoRF.py
6  peopleList = cv2.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, 0))
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
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 # Almacenando el modelo obtenido
40 #face_recognizer.write('modeloEigenFace.xml')
41 #face_recognizer.write('modeloFisherFace.xml')
42 face_recognizer.write('modeloLBPHFace.xml')
43 print('Modelo almacenado...')
44
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
imagePaths= ['Henry']
PS C:\Users\Henry>

```

Figura 4: Entrenador

```

Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda
+ capturandoRostros.py  + entrenandoRF.py  + ReconocimientoFacial.py
G: > Mi unidad > 8vo Semestre > Proyectos > Tarea 7 > VideoTutorialesAuxiliatura090-main > Reconocimiento > ReconocimientoFacial.py
1  import cv2
2  import os
3
4  dataPath = 'G:\Mi unidad\8vo Semestre\Proyectos\Tarea 7\Imagenes' #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('video.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)
33         result = face_recognizer.predict(rostro)
34
35         cv2.putText(frame, '{}'.format(result), (x,y-5), 1, 1.3, (255, 255, 0), 1, cv2.LINE_AA)
36         ...
37
38     # EigenFaces
39     if result[1] < 5700:
40         cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x,y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
41         cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
42     else:
43         cv2.putText(frame, 'Desconocido', (x,y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
44         cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 0, 255), 2)
45
46     # FisherFace
47
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
imagePaths= ['Henry']
PS C:\Users\Henry>

```

Figura 5: Reconocimiento Facial 1

```

37
38 # EigenFaces
39 if result[1] < 5700:
40     cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x,y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
41     cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
42 else:
43     cv2.putText(frame, 'Desconocido', (x,y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
44     cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 0, 255), 2)
45
46 # FisherFace
47 if result[1] < 500:
48     cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x,y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
49     cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
50 else:
51     cv2.putText(frame, 'Desconocido', (x,y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
52     cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 0, 255), 2)
53
54 # LBPHFace
55 if result[1] < 70:
56     cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x,y-25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
57     cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
58 else:
59     cv2.putText(frame, 'Desconocido', (x,y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
60     cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 0, 255), 2)
61
62 cv2.imshow('frame', frame)
63 k = cv2.waitKey(1)
64 if k == 27:
65     break
66 cap.release()
67 cv2.destroyAllWindows()
68
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
imagePaths= ['Henry']
PS C:\Users\Henry>

```

Figura 6: Reconocimiento Facial 2

II. RESULTADOS

```

--:-----
Ingrese su elección: 1
Ingrese la duración de la grabación en segundos: 10
Comenzando la grabación...
Grabación finalizada.
Archivo de audio grabado correctamente.
Seleccione una opción:
1. Grabar
2. Reproducir
3. Graficar
4. Graficar densidad
5. Salir
Ingrese su elección: 2
Seleccione una opción:

```

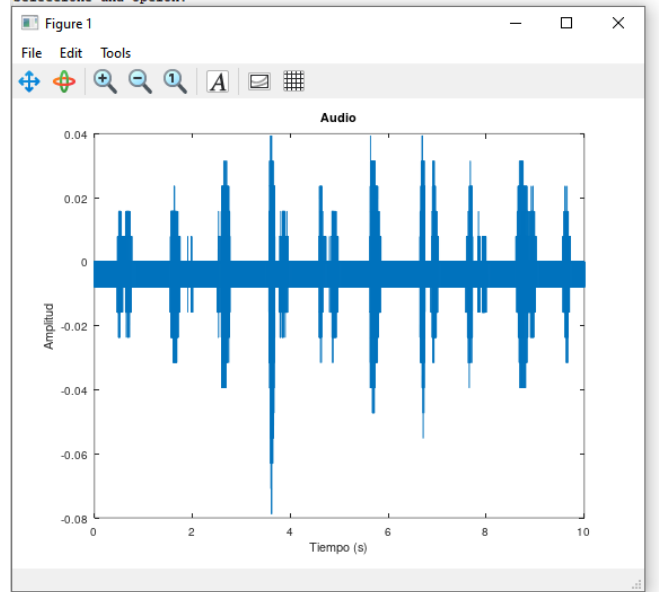


Figura 7: Gráfica de la modulación de la voz.

Ingrese su elección: 3
 Seleccione una opción:
 1. Grabar
 2. Reproducir
 3. Graficar
 4. Graficar densidad
 5. Salir
 Ingrese su elección: 4
 Graficando espectro de frecuencia...
 Seleccione una opción:

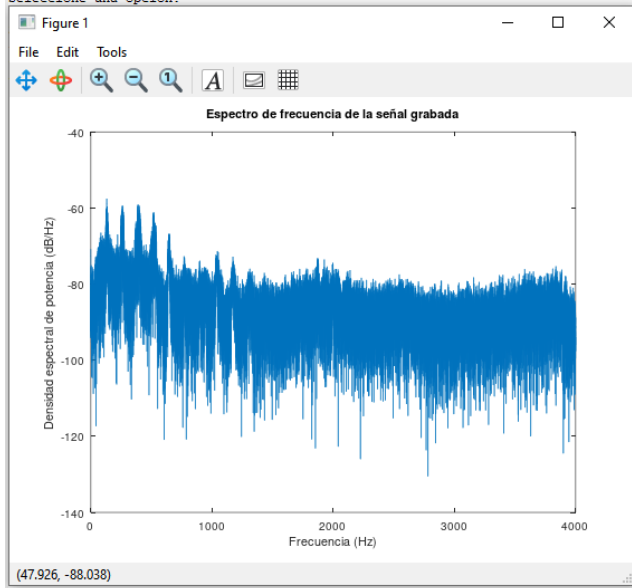


Figura 8: Gráfica de la densidad de la voz.

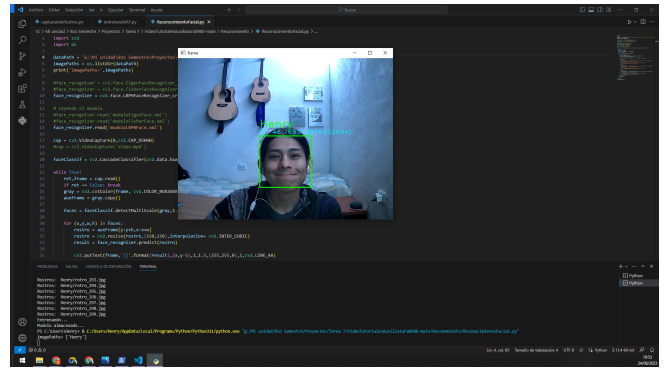


Figura 9: Reconocimiento Facial

III. REPOSITORIO

<https://github.com/HipG-3007/Proyectos.git>