INSTITUTO TECNOLOGICO DE CULIACAN

ING. SISTEMAS COMPUTACIONALES

INTELIGENCIA ARTIFICIAL (11:00 A 12:00)

UNIDAD 4 PROYECTO DETECTOR DE EMOCIONES

BRYAN JAVIER ANGULO SANDOVAL

ZETH ODIN ALFONSO JIMENEZ VELAZQUEZ

LIBRERIAS A UTILIZAR:

1. cv2 (OpenCV - Open Source Computer Vision Library)

OpenCV es una biblioteca de código abierto enfocada en **procesamiento de imágenes y visión por computadora**. Está escrita en C++ pero tiene una poderosa interfaz en Python (cv2).

Funcionalidades clave:

- **Captura de video y cámaras web**: cv2.VideoCapture() permite acceder a cámaras en tiempo real.
- **Lectura y escritura de imágenes y videos**: cv2.imread(), cv2.imwrite(), cv2.VideoWriter().
- **Detección de objetos**: Reconocimiento facial, detección de bordes, detección de movimiento, etc.
- **Transformaciones geométricas**: Rotar, escalar, recortar, redimensionar.
- **Procesamiento de color**: Conversión entre BGR, RGB, HSV, escala de grises, binarización, umbralización.
- **Aplicación de filtros**: Blur, sobel, Canny (detección de bordes), etc.

En el reconocimiento de emociones:

- Detecta rostros en imágenes o video en tiempo real.
- Preprocesa la imagen (escala de grises, recorte, resize).
- Visualiza resultados (dibujar cajas, mostrar texto/emociones en pantalla).

2. os (Operating System Interface)

¿Qué es?

os es un módulo estándar de Python que permite **interactuar con el sistema operativo**, como si estuviéramos ejecutando comandos en la terminal o el explorador de archivos.

Funcionalidades clave:

- **Navegar por carpetas y archivos**: os.listdir(), os.path.join().
- **Crear, mover o eliminar carpetas/archivos**: os.mkdir(), os.remove().

- **Verificar existencia de rutas**: os.path.exists().
- **Obtener rutas absolutas o relativas**: os.getcwd(), os.path.abspath().

En el reconocimiento de emociones:

- Crea una carpeta para cada clase de emoción (feliz, triste, etc.).
- Accede a las imágenes durante el entrenamiento del modelo.
- Guarda rostros extraídos por OpenCV en carpetas organizadas.

3. numpy (Numerical Python)

¿Qué es?

NumPy es una biblioteca fundamental para **cálculo numérico** y **computación científica** en Python. Permite trabajar con **arrays multidimensionales (matrices)** de forma eficiente.

Funcionalidades clave:

- **Arreglos de N dimensiones (ndarray)**: más eficientes que las listas de Python.
- **Operaciones matemáticas vectorizadas**: suma, multiplicación, media, desvío estándar, etc.
- **Manejo de datos de imágenes**: imágenes en blanco y negro o color pueden representarse como matrices NumPy.
- **Algebra lineal y estadística**: imprescindible en aprendizaje automático.

En el reconocimiento de emociones:

- Convierte las imágenes en matrices para alimentar al modelo.
- Normaliza los datos (ej. escalar píxeles entre 0 y 1).
- Transforma imágenes a vectores para clasificadores clásicos (SVM, etc.).

4. time

¿Qué es?

time es un módulo estándar de Python para **medir, manipular o pausar el tiempo**.

Funcionalidades clave:

- time.sleep(segundos): pausa la ejecución del programa.
- time.time(): devuelve el tiempo actual en segundos desde el "epoch" (útil para medir duración).
- time.strftime(): formato de tiempo legible para humanos.

En el reconocimiento de emociones:

- Controla la velocidad de captura de imágenes para el dataset.
- Evita guardar múltiples imágenes idénticas (con pausa entre cada frame).
- Mide el tiempo que tarda la detección o predicción para analizar rendimiento.

5. imutils

¿Qué es?

imutils es una biblioteca de Python desarrollada por Adrian Rosebrock (autor de PyImageSearch) para **simplificar tareas comunes de OpenCV**.

Funcionalidades clave:

- imutils.resize(): redimensiona imágenes sin calcular manualmente el tamaño.
- imutils.rotate_bound(): rota imágenes sin cortar partes.
- imutils.grab_contours(): extrae contornos sin preocuparse de las versiones de OpenCV.
- Funciones para recortar regiones, centrar imágenes, etc.

En el reconocimiento de emociones:

- Redimensiona los frames de la cámara para mejorar velocidad y eficiencia.
- Asegura que las imágenes de rostros tengan un tamaño estándar antes de almacenarlas o clasificarlas.
- Facilita transformaciones geométricas en imágenes de rostros.

Explicacion de cada parte del código:

CapturandoRostros.py

```
import cv2

import os

import imutils
```

cv2: Importa OpenCV para capturar video, procesar imágenes y detectar rostros.

os: Permite crear carpetas y manejar rutas del sistema operativo.

imutils: Facilita algunas operaciones comunes con imágenes (como redimensionar).

```
#emotionName = 'Enojo'

#emotionName = 'Felicidad'

#emotionName = 'Sorpresa'

#emotionName = 'Tristeza'
```

Estas líneas están comentadas porque debes activar una sola línea a la vez dependiendo de qué emoción deseas capturar.

Por ejemplo, si estás capturando rostros con expresión de enojo, descomentas:

```
emotionName = 'Enojo'

dataPath = 'C:/Users/Usuario/PycharmProjects/PythonProject/train' #Cambia a la ruta donde hayas almacenado Data

emotionsPath = dataPath + '/' + emotionName
```

dataPath: Es la **carpeta base** donde se almacenará el dataset.

emotionsPath: Es la **subcarpeta para la emoción específica**, por ejemplo: .../train/Enojo.

```
cap = cv2.VideoCapture(3)
```

Inicia la **captura de video desde una cámara**.

El número 3 representa el índice de la cámara (en muchos casos, puede ser 0 o 1 si usas webcam integrada o una externa).

Puedes probar con otros valores si no se detecta video.

```
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

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(emotionsPath + '/rostro_{}.jpg'.format(count),rostro)

                count = count + 1
```

Dibuja un **rectángulo verde** alrededor del rostro en el frame.

Recorta el rostro desde la imagen original (auxFrame) usando coordenadas.

Redimensiona el rostro recortado a **150x150 píxeles** (tamaño típico para redes neuronales o clasificadores).

Guarda la imagen con un nombre incremental rostro_0.jpg, rostro_1.jpg, etc., en la carpeta de la emoción.

Aumenta el contador para el siguiente rostro.

```
k =  cv2.waitKey(1)

        if k == 27 or count >= 200:

                break
```

Espera 1 milisegundo a que el usuario presione una tecla.

Si se presiona Esc (tecla 27) o si ya se guardaron 200 rostros, el bucle se detiene.

Entrenando.py:

```
import cv2

import os

import numpy as np
```

import time

- cv2: OpenCV, para procesamiento de imágenes y modelos de reconocimiento facial.
- os: Para navegar entre carpetas y archivos del sistema.
- numpy: Para manejar arrays numéricos (como etiquetas).
- time: Para medir cuánto tarda el entrenamiento.

def obtenerModelo(method, facesData, labels):

if method == 'EigenFaces': emotion_recognizer = cv2.face.EigenFaceRecognizer_create()

if method == 'FisherFaces': emotion_recognizer = cv2.face.FisherFaceRecognizer_create()

if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

- **Recibe**:
  - method: nombre del algoritmo (EigenFaces, FisherFaces, LBPH)
  - facesData: lista con las imágenes de rostros
  - labels: lista con las etiquetas correspondientes a cada rostro
- **Crea** un modelo del tipo especificado.

Sobre los métodos:

- EigenFaces: método basado en componentes principales (PCA). Sensible a iluminación y posición.
- FisherFaces: basado en análisis discriminante lineal (LDA). Mejora la discriminación entre clases.
- LBPH (Local Binary Patterns Histogram): más robusto a cambios de iluminación y escala, ideal para reconocimiento más práctico y en tiempo real.

print("Entrenando ("+method+")...")

inicio = time.time()

emotion_recognizer.train(facesData, np.array(labels))

```
    tiempoEntrenamiento = time.time()-inicio
    print("Tiempo de entrenamiento ( "+method+" ): ", tiempoEntrenamiento)
```

Inicia el entrenamiento del modelo con los datos de entrada (facesData) y sus respectivas etiquetas (labels).

Mide y muestra el tiempo que tarda entrenando.

```
dataPath = 'C:/Users/Usuario/PycharmProjects/PythonProject/train'  # Cambia a la ruta donde hayas almacenado Data
emotionsList = os.listdir(dataPath)
print('Lista de personas: ', emotionsList)
```

Define el directorio raíz donde están las carpetas con los rostros (Enojo, Felicidad, etc.).

os.listdir() obtiene una lista con el nombre de cada carpeta (una por emoción).

Muestra esa lista.

```
for nameDir in emotionsList:
    emotionsPath = dataPath + '/' + nameDir

    for fileName in os.listdir(emotionsPath):
        labels.append(label)
        facesData.append(cv2.imread(emotionsPath+'/'+fileName, 0))
    label = label + 1
```

Recorre cada carpeta (una por emoción).

Para cada imagen dentro de esa carpeta:

- Lee la imagen en escala de grises (cv2.imread(..., 0)).
- Guarda la imagen en facesData.
- Asigna la etiqueta correspondiente (label) a esa imagen.

Cambia la etiqueta (label) para la siguiente emoción.

Reconocimiento de emociones.py:

```
import cv2

import os

import numpy as np
```

- cv2: Librería de OpenCV para visión por computadora (detección, reconocimiento, etc.).
- os: Para interactuar con el sistema de archivos (leer directorios).
- numpy: Para operaciones con matrices, usado aquí para crear imágenes vacías.

```
if method == 'EigenFaces': emotion_recognizer =
cv2.face.EigenFaceRecognizer_create()

if method == 'FisherFaces': emotion_recognizer =
cv2.face.FisherFaceRecognizer_create()

if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()


emotion_recognizer.read('modelo' + method + '.xml')
```

- Según el método seleccionado, se crea un reconocedor facial correspondiente.
- Se lee un archivo .xml que contiene el modelo entrenado para identificar emociones faciales.

```
dataPath = 'C:/.../Data/train'

imagePaths = os.listdir(dataPath)

print('imagePaths=', imagePaths)
```

dataPath contiene las carpetas con imágenes etiquetadas por emoción.

imagePaths obtiene la lista de etiquetas (nombres de carpetas) como: ['Felicidad', 'Tristeza', etc].

```
while True:
```

```
  ret, frame = cap.read()
  if ret == False: break
```

Bucle infinito que captura y procesa cada frame.

Si no se obtiene el frame (ret == False), se rompe el bucle.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
auxFrame = gray.copy()
nFrame = cv2.hconcat([frame, np.zeros((480, 300, 3), dtype=np.uint8)])
```

Convierte la imagen a escala de grises.

Hace una copia para no modificar el original.

nFrame combina el frame con un espacio vacío para mostrar el emoji al lado.

```
for (x, y, w, h) in faces:
  rostro = auxFrame[y:y + h, x:x + w]
  rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC)
  result = emotion_recognizer.predict(rostro)
```

- Extrae el rostro detectado del frame.
- Lo redimensiona a 150x150 píxeles.
- Realiza la predicción de la emoción con el modelo.

```
cv2.putText(frame, '{}'.format(result), (x, y - 5), 1, 1.3, (255, 255, 0), 1, cv2.LINE_AA)
```

En esta línea muestra el resultado numerico en pantalla

Para el modelo de EigenFaces:

```
if method == 'EigenFaces': if result[1] < 5700: cv2.putText(frame,
'{}'.format(imagePaths[result[0]]), (x, y - 25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2) image =
emotionImage(imagePaths[result[0]]) nFrame = cv2.hconcat([frame, image]) else:
```

cv2.putText(frame, 'No identificado', (x, y - 20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA) cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2) nFrame = cv2.hconcat([frame, np.zeros((480, 300, 3), dtype=np.uint8)])

Este bloque se encuentra dentro de un bucle for que itera sobre los rostros detectados. result contiene dos elementos:

result[0]: el índice de la etiqueta predicha (por ejemplo, 0 = “Felicidad”).

result[1]: el valor de distancia o error (entre menor, mejor encaje). Si es alto, significa que el modelo no está seguro.

Este bloque evalúa si el modelo reconoce al rostro basándose en un umbral de confianza (5700 en este caso).

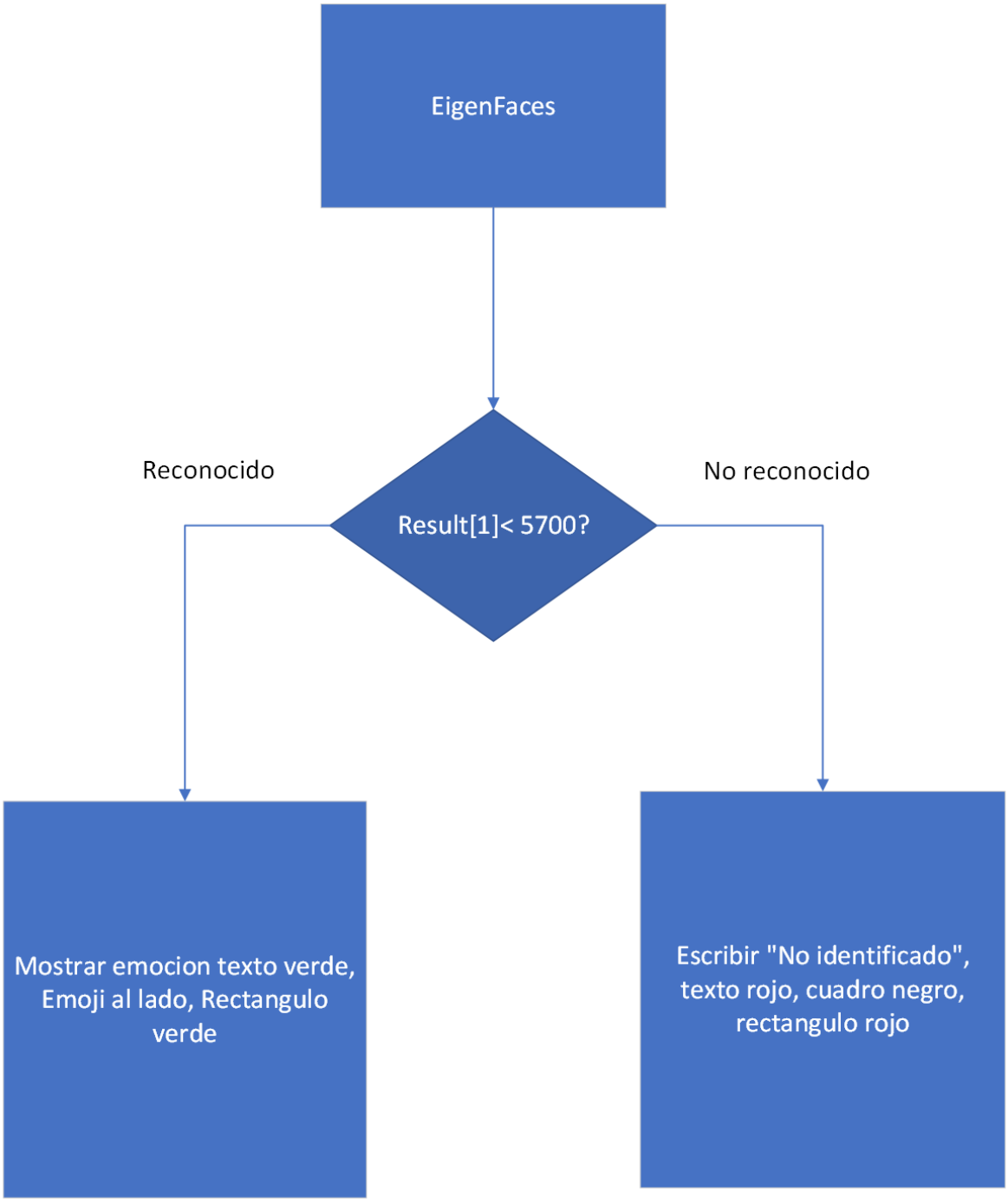
cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x, y - 25), 2, 1.1, (0, 255, 0), 1, cv2.LINE_AA)

Escribe el nombre de la emoción reconocida en la imagen del frame.

imagePaths[result[0]] obtiene la etiqueta textual desde la lista (imagePaths[0] = "Felicidad", por ejemplo).

(x, y - 25): coordenada para colocar el texto arriba del rostro.

(0, 255, 0): texto en verde (RGB).



LINK DE VIDEO DE PRUEBAS: https://www.youtube.com/watch?v=8LVvQPCWLdM