TAREA 3 : Reconocimiento Facial con Python y OpenCV*

Pablo Andres Montufar Perez, 201902235[1]

[1] *Facultad de Ingeniería, Escuela de Mecánica Eléctrica, Universidad de San Carlos, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.*

Python es un lenguaje de programación ampliamente utilizado por su simplicidad y versatilidad, especialmente en áreas como inteligencia artificial, visión por computadora y procesamiento de imágenes. En este proyecto, emplearemos Python junto con la biblioteca OpenCV para realizar reconocimiento facial.

I. INTRODUCCIÓN

Visual Studio Code es el entorno de desarrollo integrado (IDE) utilizado para escribir y ejecutar el código, proporcionando una plataforma eficiente para desarrollar proyectos basados en Python.

Figura 1: Logo de Python y OpenCV

II. CONFIGURACIÓN INICIAL

A. Creación de la Carpeta

Antes de escribir el código, se creó una estructura de carpetas para almacenar tanto los scripts como los datos generados. En este caso, se definió la ruta: `C:\Users\Pablo Andres\Desktop\TAREAS U\ProyectosDiciembre24\Tarea 3\Fotos y Videos`

B. Instalación de Librerías

Para habilitar el reconocimiento facial, se instalaron las siguientes bibliotecas desde la terminal de Visual Studio Code:

- `opencv-python`: Proporciona las herramientas principales de OpenCV.
- `opencv-python-headless`: Versión optimizada para servidores o sistemas sin GUI.
- `opencv-contrib-python`: Incluye módulos adicionales de OpenCV.
- `imutils`: Facilita la manipulación de imágenes.

Los comandos utilizados fueron:

```
pip install opencv-python
pip install opencv-python-headless
pip install opencv-contrib-python
pip install imutils
```

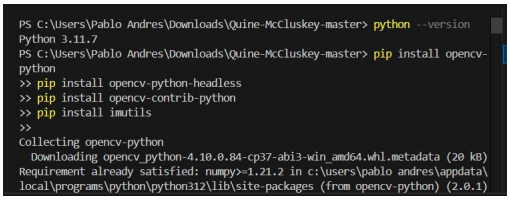

Figura 2: Instalación de librerías en Visual Studio Code

III. EXPLICACIÓN DEL CÓDIGO

El programa realiza las siguientes tareas principales:

A. Inicialización y Configuración

Se define una ruta para almacenar las imágenes capturadas, y se asegura de que la carpeta exista. Si no, se crea automáticamente:

```
if not os.path.exists(personPath):
    os.makedirs(personPath)
```

B. Detección de Rostros con OpenCV

Utilizando el clasificador Haar Cascade (`haarcascade_frontalface_default.xml`), el programa detecta rostros en tiempo real a través de la cámara o de un video cargado previamente. El método `detectMultiScale` es crucial para detectar rostros en diferentes tamaños y ubicaciones dentro de la imagen.

C. Procesamiento y Almacenamiento

Cada rostro detectado se recorta, redimensiona a 150x150 píxeles y se guarda en la carpeta creada:

* PROYECTOS DE COMPUTACION APLICADA A I.E. Sección A

```
rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC)
cv2.imwrite(personPath + '/rostro_{}.jpg'.format(count), rostro)
```

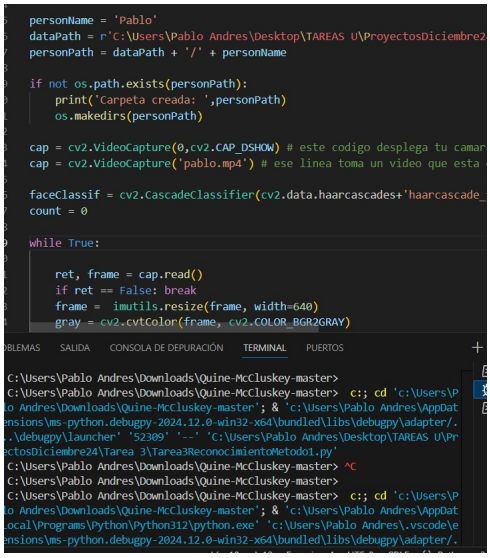

Figura 3: Detección de rostros en tiempo real

IV. APLICACIONES Y PROPÓSITO

Este programa tiene aplicaciones en:

- Sistemas de seguridad mediante reconocimiento facial.
- Preparación de datasets personalizados para entrenar modelos de inteligencia artificial.
- Proyectos educativos y de investigación relacionados con visión por computadora.

En este caso, el programa captura 300 imágenes de un rostro para construir un conjunto de datos que puede ser utilizado en sistemas de reconocimiento facial o aprendizaje automático.

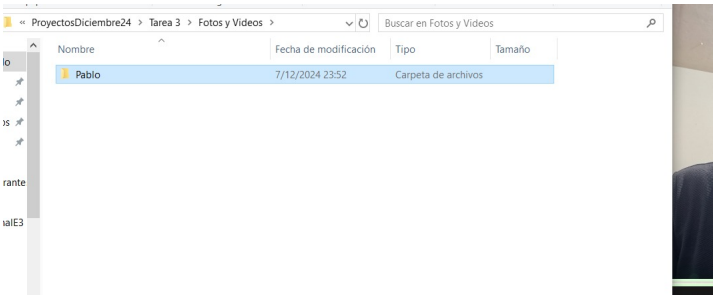

Figura 4: Imágenes almacenadas en la carpeta especificada

V. CONCLUSIÓN

Python y OpenCV son herramientas poderosas para desarrollar proyectos de visión por computadora. Con este programa, se exploraron conceptos clave como detección de rostros, procesamiento de imágenes y almacenamiento de datos. Visual Studio Code demostró ser un IDE eficiente para este tipo de aplicaciones, facilitando la integración de múltiples bibliotecas y la ejecución del código.

VI. REPOSITORIO DEL PROYECTO

El código fuente de este proyecto, junto con ejemplos y otros recursos, está disponible en el siguiente enlace:

Repositorio en GitHub

Este repositorio contiene el código original, las mejoras realizadas y las gráficas generadas por los programas descritos en este documento.