|  |  |
| --- | --- |
| **CARRERA:** | **Ingeniería Biomédica** |

|  |  |  |  |
| --- | --- | --- | --- |
| **MATERIA:** | **Procesamiento Digital de Señales** | **GRUPO:** | **8DV** |

|  |  |
| --- | --- |
| **NOMBRE MAESTRO:** | **Erwin Sosa López** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea:** | **Proyecto Final** | **FECHA DE ENTREGA:** | **18 de junio de 2020** |

|  |  |
| --- | --- |
| **Alumno** | **MATRÍCULA** |
| **Luis Fernando Cardeña Ley** | **E15080882** |

**Introducción**

La detección de imágenes u objetos es una tecnología informática que procesa imágenes y detecta los objetos en ella. A menudo se confunde la detección de imágenes con la clasificación de imágenes. Cuando solo se desea localizar patrones de imágenes, se usa la detección de objetos, pero cuando se desea localizar, identificar y clasificar objetos en la imagen, se utiliza la clasificación de imágenes. Un ejemplo de clasificación de imágenes es cuando realizamos una búsqueda de imágenes, y el motor de búsqueda tiene que procesar diferentes imágenes y clasificar los objetos en ellas para entregarnos los resultados que contengan los objetos que estamos buscando.

El reconocimiento de imágenes es la habilidad de una inteligencia artificial de detectar objetos, clasificarlos y reconocerlos. El último paso es un proceso muy cercano al nivel de reconocimiento que tiene un humano. Un ejemplo de la aplicación de reconocimiento de imágenes es el reconocimiento facial, la capacidad de un algoritmo de localizar e identificar rostros en una imagen.

En el siguiente trabajo se aplicará una librería de reconocimiento facial en Python para crear un programa simple que pueda identificar el rostro de un usuario en diferentes fotos desconocidas por el programa. Se busca probar qué tan preciso resulta el programa al identificar a una persona en diferentes fotografías. Para ello, el programa deberá identificar y registrar los rostros de la persona en una serie de imágenes de muestra donde no se encuentre ningún rostro más que el del usuario que se desea registrar.

Python es un lenguaje de programación de código abierto que se caracteriza por su sintaxis simple. Actualmente es uno de los lenguajes más utilizados para el área de inteligencia artificial y machine learning, debido a las múltiples librerías especializadas para esas tareas en combinación con la simplicidad del lenguaje. Para el desarrollo del siguiente programa se utilizará principalmente un módulo de reconocimiento facial y una versión para Python de OpenCV, una biblioteca libre de visión computacional escrita en C++. La primera librería realizará la mayor parte del procesamiento y opencv-python se usará para mostrar las imágenes y graficar sobre ellas. El programa se probó en una computadora con un sistema operativo basado en Linux, y si se quisiera probar en Windows, se necesitaría una versión actualizada de Python, instalar las dependencias necesarias para usar la librería de detección de rostros, y finalmente, instalar la librería con el gestor de paquetes de Python, pip. También se tendría que modificar el formato de las rutas de archivos en algunas líneas.

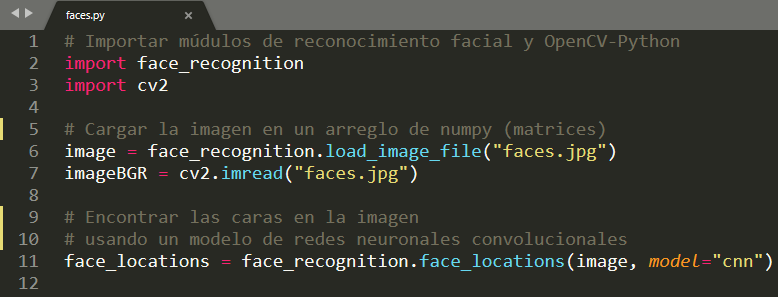
**Desarrollo**

Primero se mostrará un programa de ejemplo de la documentación de la librería *face\_recognition* que realiza la lectura de una fotografía y detectar las caras en ella. El programa se guardará en el archivo *faces.py,* y la primera parte se muestra en la figura 1.

En Python, lo primero que se acostumbra a hacer es importar las librerías con el comando *import*, seguido del nombre de la librería. Cuando usamos una función que le pertenece a una librería, debemos poner primero el nombre de la librería seguido de un punto y la función de dicha librería que queremos llamar.

El programa *faces.py* cargará ‘*faces.jpg*’ en la variable *image*. Esta variable consiste en un arreglo de listas que contendrá los valores de cada pixel. Es importante mencionar que la librería *face\_recognition* maneja el estándar RGB para tratar las imágenes, mientras que la librería de OpenCV (cv2, en el programa) las maneja en formato BGR. Por esa razón, en las líneas 6 y 7 leemos la misma imagen dos veces, *image* estará en formato RGB e *imageBGR* en formato BGR.

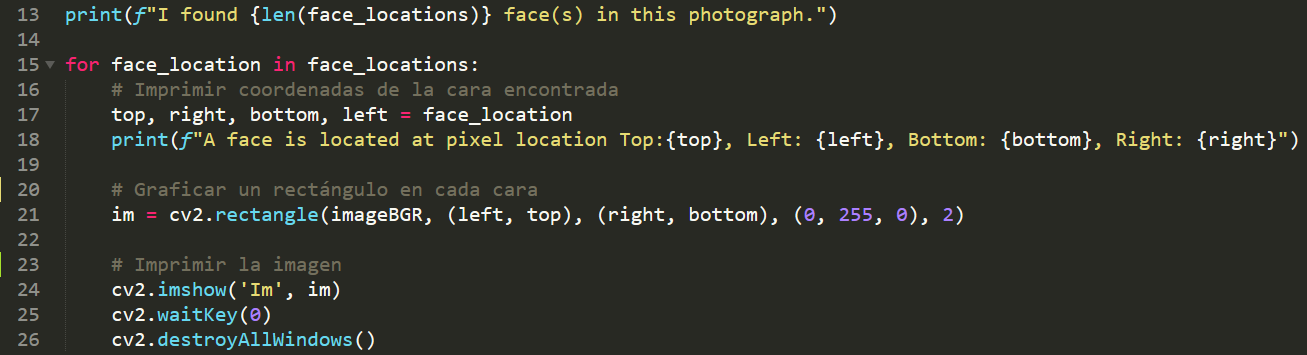
En la línea 11 usaremos la función *face\_locations*, perteneciente a la librería *face\_recognition*,para obtener el conjunto de localizaciones de cada cara dentro de las coordenadas de la imagen. Estos valores se guardarán en la variable *face\_locations*, que se usará para iterar sus valores en un ciclo.

**Figura 1.** Importación de librerías, cargar imagen y detección de caras.

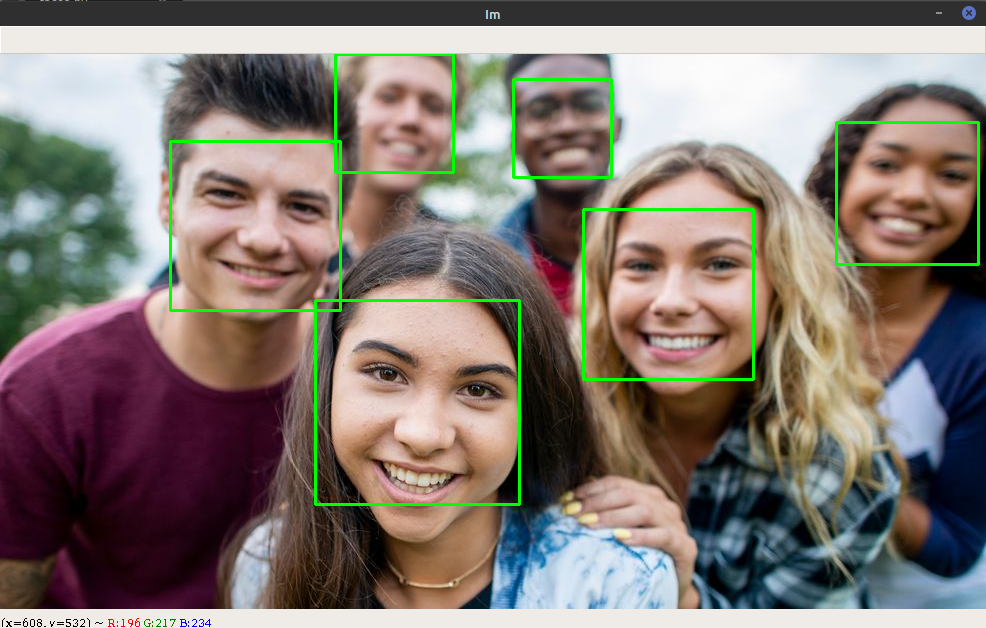
La última parte del programa *faces.py* imprimirá una vez la leyenda indicando el número de caras encontradas en la imagen. El número de imágenes dentro del string se indicará con el número de elementos en *face\_locations.*

Se utiliza una estructura de iteración *for* para:

1. Determinar los puntos *top, right, bottom* y *left* que servirán como coordenadas para graficar los recuadros en los rostros. Estos cuatro valores se obtienen de la variable del ciclo que itera en una lista de localizaciones, *face\_locations*.
2. Usar la función *cv2.rectangle* para dibujar los rectángulos sobre cada cara en la imagen original. Como esta es una función de la librería OpenCV, usamos la imagen BGR.
3. Imprimir la imagen con los recuadros. En cada iteración agregará un cuadro, cada iteración continuará cuando se presione una tecla y al final cerrará la ventana.

**Figura 2.** Imprimir imagen original con las caras marcadas con un cuadro.

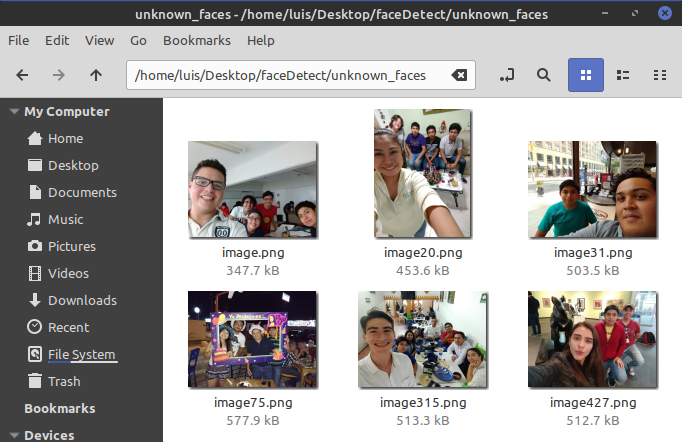
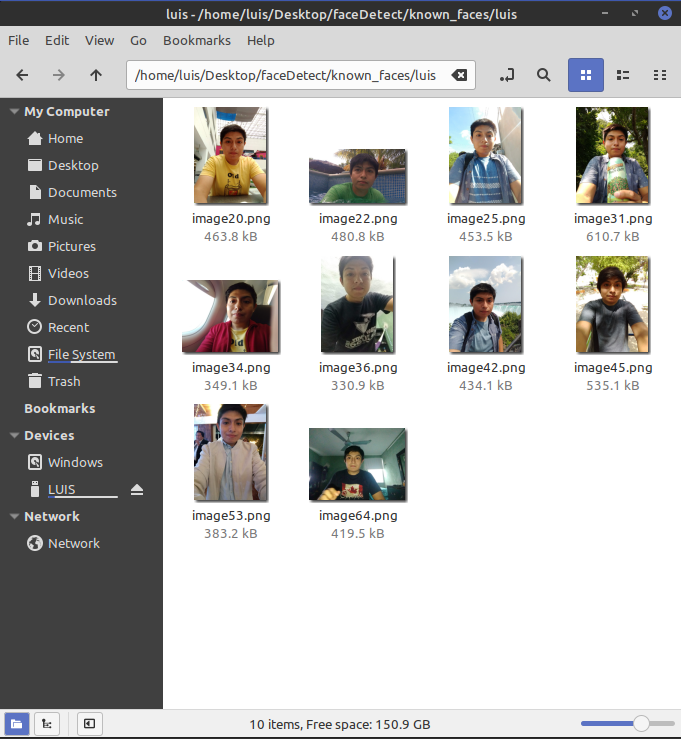
El resultado de ejecutar este programa se muestra en la figura 3. Un cuadro de color verde fue agregado sobre la fotografía original en cada cara y se muestra en una ventana emergente.

  
**Figura 3.** Detección de rostros.

El programa anterior fue una muestra de una aplicación para detectar rostros, pero no llega a identificarlos, solo declara que el objeto marcado es muy posiblemente un rostro. Para identificar rostros debemos cargar el programa con fotografías de muestra donde solo se observe el rostro de una persona.

Ahora diseñaremos el programa *recognition.py*, que realizará esta tarea analizando fotos de una persona, y luego identificará a esa persona en fotos completamente diferentes. La siguiente figura, del lado izquierdo, muestra las fotos conocidas que se usarán para registrar el rostro de una persona, y la del lado derecho contiene las fotos donde se tratará de identificar cada rostro detectado con cada uno de los rostros conocidos y detectando si hay un *match*.

Los rostros conocidos estarán en la carpeta ‘*known\_faces/luis’*, y las fotos donde se tratará de identificar a la persona estarán en ‘*unknown\_faces’*. Observar que dentro de la primera carpeta, hay una subcarpeta con el nombre *luis*, y dentro de ella se encuentran las fotos con su rostro. Podemos crear diferentes carpetas con el nombre de cada persona, y el programa tratará de identificar a cada uno de ellos. Por el momento solo probaremos con una persona.

  
**Figura 4.** Imágenes de prueba

Para entender mejor la lógica del programa, vamos a enumerar en puntos muy simplificados el proceso con el cual el programa identificará los rostros en las fotos desconocidas.

1. Un ciclo iterará sobre cada carpeta dentro del directorio de imágenes conocidas, y otro ciclo dentro del primero iterará sobre cada foto dentro de todas las carpetas. Se crearán dos listas de datos, una con la información de cada rostro conocido, y otra lista del mismo tamaño con el nombre de la persona correspondiente a esa información.
2. Un ciclo iterará sobre cada imagen desconocida anotando la ubicación de cada rostro detectado y los *encodings*, que se puede interpretar como la identificación de cada rostro. Luego, en otro ciclo, se comparará los *encodings* de cada rostro desconocido con los rostros conocidos, y se registrará cualquier *match* cuando la comparación dé como valor True. Cada vez que se detecte un True, este se usará para buscar el valor correspondiente al índice en la lista *known\_names*, y así se obtendrá el nombre de la persona en la foto desconocida. La última parte de este ciclo imprimirá con la ayuda de OpenCV los cuadros de cada cara identificada, junto con una etiqueta incluyendo el nombre de la persona.

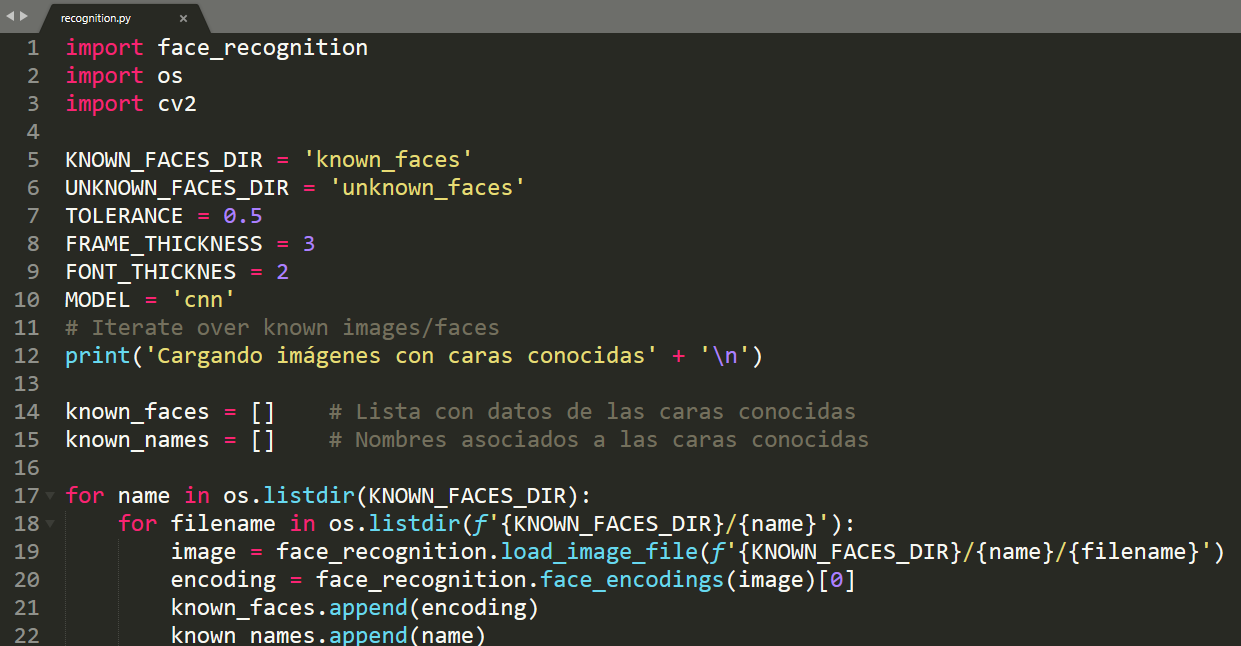
En la figura 5 podemos ver la primera parte del código. Iniciamos declarando librerías. La librería *os* sirve para leer los archivos del sistema operativo. Después tenemos una serie de variables que se usarán en el programa, como la ruta de los directorios, la tolerancia al identificar los rostros, características de los gráficos y la fuente y el tipo de modelo de detección de rostros. La tolerancia indica qué tan preciso será nuestro programa, si la tolerancia es muy alta, puede resultar en falsos positivos (identificar rostros que no son correctos), y si es muy baja, hará que el programa sea más estricto al identificar una cara. El valor recomendado es 0.6. El modelo utilizado es el de redes neuronales convolucionales, el cual es un tipo de red neuronal artificial que usa predicciones basadas en cálculos de probabilidad para determinar un resultado de salida de acuerdo con una entrada. Es un tema muy extenso correspondiente al área de aprendizaje de máquinas e inteligencia artificial, estos modelos ya han sido probados y aplicados para ser usados en este tipo de librerías.

Después se imprime una línea para indicar que se comenzarán a leer las imágenes conocidas y se declaran las listas de caras y nombres conocidos. Estas listas tendrán la misma extensión. Si en la carpeta de imágenes conocidas tenemos una carpeta con el nombre *luis* y 2 imágenes dentro de ella, y otra carpeta llamada *gabo,* con 3 imágenes, la lista de nombres conocidos se ordenará en orden alfabético como:

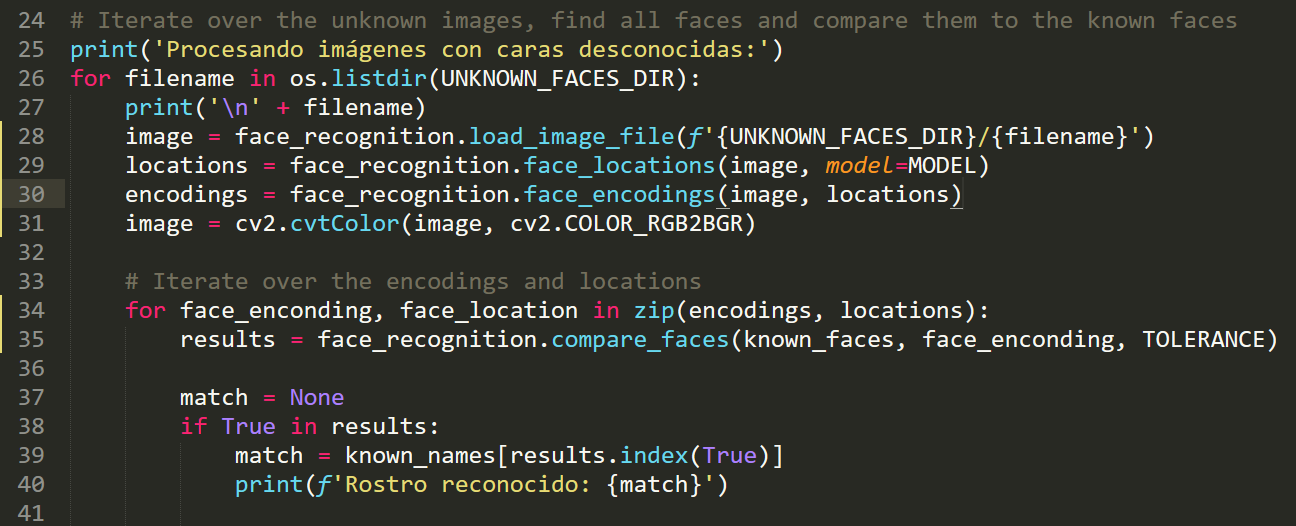
*known\_names = [‘gabo, ‘gabo, ‘gabo’, ‘luis, ‘luis’]*

Esta lista servirá para identificar cuando un rostro desconocido haga *match* con un valor contenido en la lista *known­\_faces* para así referenciar con ese mismo índice el nombre en la lista de nombres conocidos.

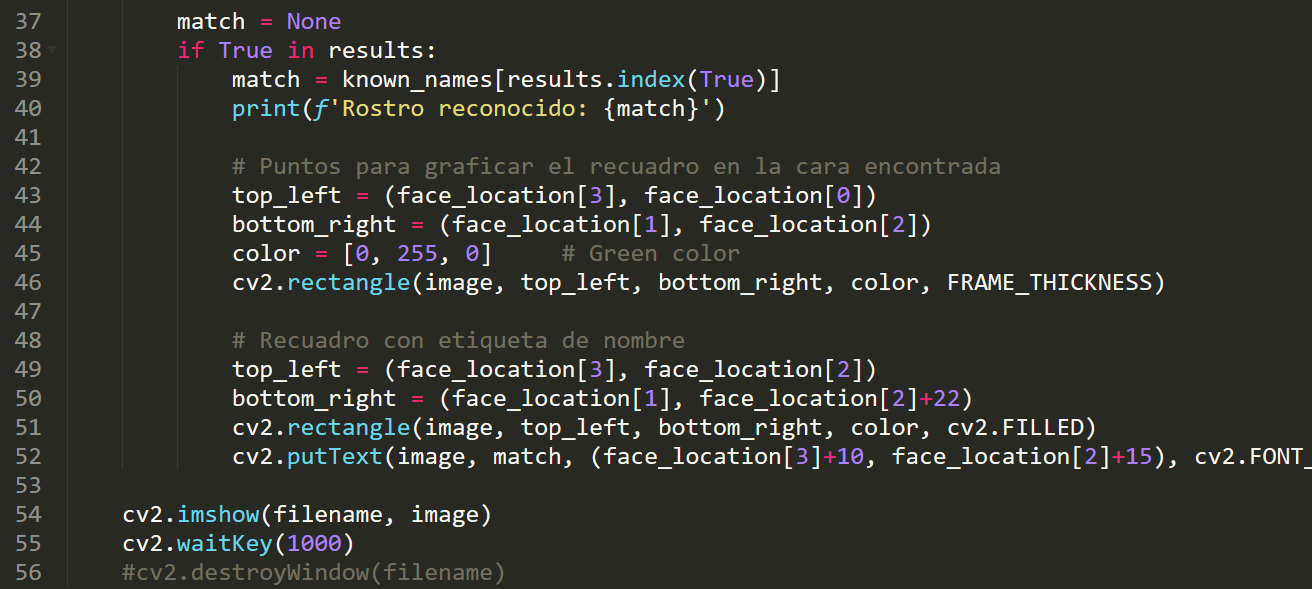
Lo que sigue es el ciclo que realiza la lectura de cada imagen desconocida y va llenando las listas anteriores con su información.

**Figura 5.** Inicio del programa de reconocimiento facial.

La figura 6 corresponde a casi todo el segundo ciclo del programa. El primer ciclo *for* registra las ubicaciones y los *encodings* de cada rostro desconocido en cada foto de en la carpeta ‘un*known\_faces*’. El segundo ciclo compara el *encoding* de cada cara desconocida con los encodings de las caras conocidas, y cada que hay un *match* entre los *encodings*,se determina el nombre de la persona conocida con la lista de *know­n\_names*.

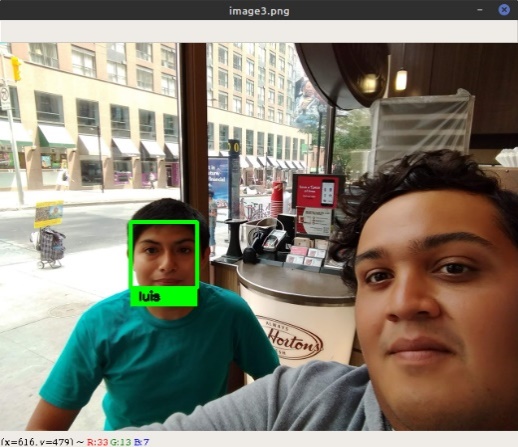
**Figura 6.** Segundo ciclo del programa *recognition.py*.

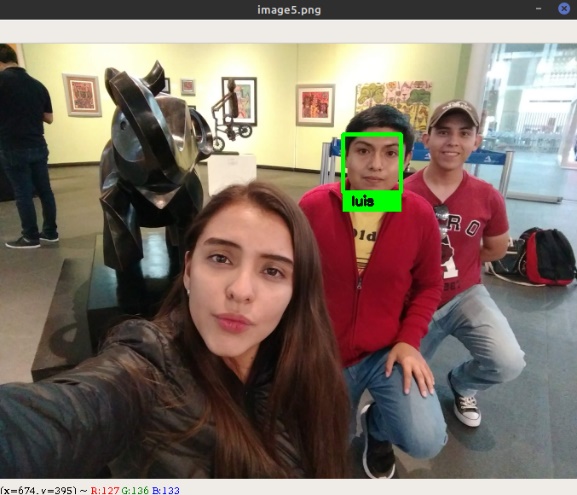
La última parte del segundo ciclo for principal del programa es para imprimir los recuadros de las personas identificadas en las fotos junto con una etiqueta con su nombre. Se utilizan las coordenadas del rostro obtenidas en la imagen desconocida y con ella se imprime el rectángulo. La etiqueta se escribe con los mismos datos pero cambiando de orden y moviendo un poco el recuadro para ajustarse en la parte inferior. Finalmente se muestra cada imagen.

**Figura 7.** Imprimir cuadros y etiquetas en las caras identificadas.

**Resultados**

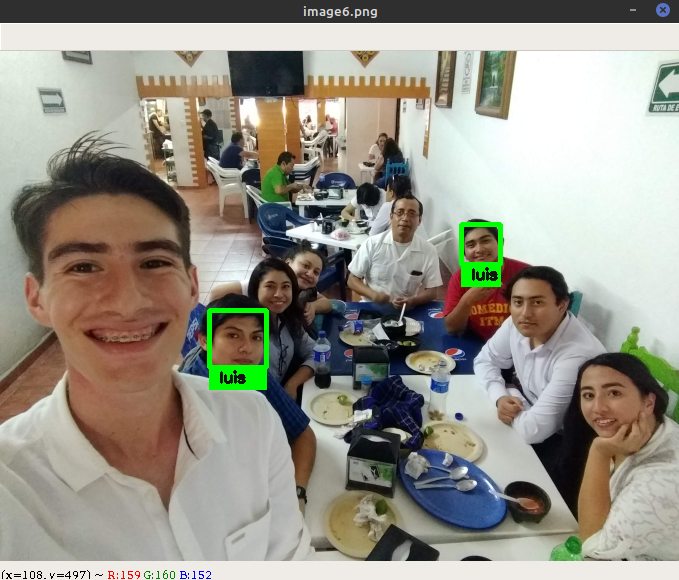
El programa fue capaz de reconocer con mucha precisión el rostro de la persona conocida, con un solo error en una de las fotos, donde identificó a la persona correcta y a otra que no lo era en una misma foto. Se muestran a continuación algunos de los resultados.

  
**Figura 8.** Prueba 1.

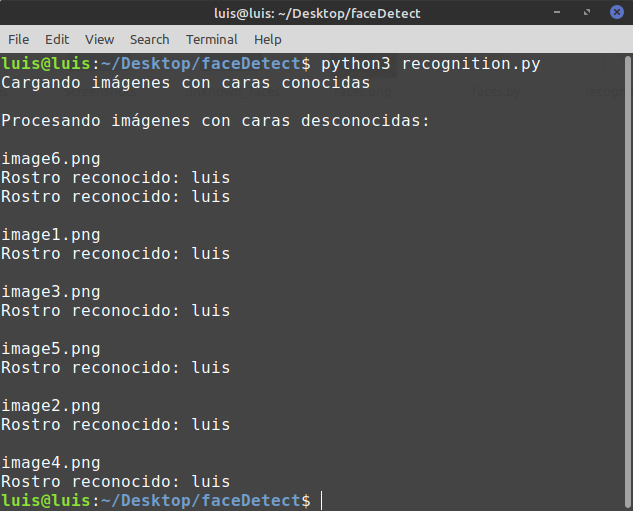
  
**Figura 9.** Prueba 2.

  
**Figura 10.** Prueba 3.

  
**Figura 11.** Prueba 4

  
**Figura 12.** Prueba 5 con un falso positivo.

Como se puede ver, la figura 12 muestra un falso positivo en los resultados, una persona fue identificada como *luis* en la foto, pero en realidad no lo era. La tolerancia se estableció en 0.5 para el programa, ya que fue el valor que dio mejores resultados, bajar la tolerancia a 0.4 hacía que se no se detecte a la persona en una de las fotos, y subirla a 0.6 ocasionaba un falso positivo más. En la figura 13 se puede ver lo que se imprimió en la terminal al correr el programa. Se puede observar que en una de las imágenes reconoció dos veces el rostro de *luis*, que es donde se dio el falso positivo.

  
**Figura 13.** Resultados del programa impresos en la terminal.

**Conclusión**

El reconocimiento de imágenes es un área del procesamiento de imágenes que utiliza los principios de del aprendizaje profundo para mejorar sus algoritmos y es una de las herramientas computacionales con mayor crecimiento en los últimos años, sobre todo con la llegada de la industria de la automatización. Con este programa pudimos hacer un detector de rostros que ofrece resultados bastante precisos usando librerías prediseñadas para el lenguaje de programación Python. Con la implementación de otras librerías es posible extender el programa para que funcione con video en tiempo real y pueda identificar a múltiples personas registradas al mismo tiempo. También puede servir para autenticar el acceso a algún programa de computadora a un cierto grupo de personas con una capa extra de seguridad.

**Documentación y fuentes de consulta**

<https://pythonprogramming.net/facial-recognition-python/>

<https://pypi.org/project/face-recognition/>

<https://medium.com/ai-techsystems/image-detection-recognition-and-image-classification-with-machine-learning-92226ea5f595>