Reconocimiento facial con OpenCV

Curso 2k19/2k20

Apellidos, nombre	Laura Medina Chaveli (laumecha@inf.upv.es)
Titulación	Grado de Ingeniería informática
Fecha	Abril, 2020

UNIVERSIDAD POLITECNICA DE VALENCIA

Índice

UNIVERSIDAD POLITECNICA DE VALENCIA

Índice de figuras

1 Resumen de las ideas clave

En este artículo vamos a presentar una aplicación capaz de realizar una detección facial para, posteriormente, llevar a cabo un reconocimiento de caras en tiempo real utilizando la librería OpenCV [1].

Utilizando visión por computador e Inteligencia Artificial, la herramienta tendrá como fuente de entrada una cámara web conectada al dispositivo y será capaz de detectar un rostro para, posteriormente, compararlo con una base de datos predefinida e informar al usuario a quien pertenece dicho rostro.

2 Introducción

Este artículo analizará, primeramente, las diferencias entre el funcionamiento de la detección facial y el reconocimiento de caras con el uso de visión por computador. Pondremos en marcha algunos ejemplos de estas dos prácticas y analizaremos los resultados.

En segundo lugar, explicará cómo realizar los cambios necesarios a uno de los ejemplos para que sea posible realizar detección facial y añadir nuevos usuarios en tiempo real. Se comentará el procedimiento realizado así como los resultados obtenidos. Finalmente, se analizarán las limitaciones observadas y se propondrán un conjunto de soluciones para ellas.

3 Objetivos

Los objetivos de este trabajo son los siguientes:

- Conseguir ejecutar los ejemplos aportados [2, 3] para desarrollar una herramienta capaz de realizar detección facial utilizando la librería de OpenCV.
- Utilizar una cámara web conectada al dispositivo para poder realizar dicho reconocimiento en tiempo real.
- Comunicar en todo momento al usuario la persona reconocida en pantalla.
- Añadir la funcionalidad de gestionar nuevos usuarios no existentes a la base de datos con el fin de poder reconocerlas en tiempo real.

4 Trabajo previo

4.1 Aspectos motivacionales y herramientas existentes

Cada vez es más frecuente escuchar el descubrimiento de nuevas tecnologías que hace unos años serían solo películas de ficción. ¿Quién sería capaz de pensar que podría ser posible pagar en algunos restaurantes con mirar unos segundos a una cámara o desbloquear un teléfono móvil con la

cámara frontal del móvil? Estos solo son unos pocos ejemplos de la cantidad de posibilidades que puede ofrecer el reconocimiento facial hoy en día.

Todo esto es posible gracias al progreso del Deep learning y la Visión Por Computador. Hoy en día, existen diversas herramientas y librerías que nos ofrecen toda la funcionalidad de la Inteligencia Artificial de la manera más simple posible.

En este trabajo hemos utilizado la librería OpenCV dado que es la más optimizada para el procesamiento de imágenes en tiempo real. Además, OpenCV incorporó un módulo de Redes Neuronales [5] con la actualización de la versión 3.3.

4.2 Detección facial

La detección facial y el reconocimiento de caras pueden ser concebidos como dos conceptos iguales. Sin embargo, en la realidad, estas dos prácticas están muy diferenciadas.

La detección de rostros es el procedimiento de identificar la presencia de caras dentro de imágenes. Podríamos decir que nosotros, los humanos, detectamos un rostro si observamos que la imagen contiene ojos, nariz y boca. De manera similar, los algoritmos de detección facial intentan simular este comportamiento además de realizar validaciones para comprobar si, efectivamente, la región detectada se trata realmente de un rostro.

Cabe decir, que la detección facial es el primer paso para realizar un reconocimiento, por lo cual, entender su funcionamiento es crucial.

Existen muchas herramientas desarrolladas que permiten la detección facial de manera muy efectiva. En este trabajo nos hemos centrado sobre este ejemplo [4] ya que los resultados son muy buenos.

Aunque en la web se encuentre la explicación más detallada, es necesario resaltar y explicar algunos conceptos para entender cuál es el procedimiento para realizar la detección de rostros y, por lo tanto, los conceptos por venir del reconocimiento facial.

Este ejemplo [4] hace uso del módulo de Redes neuronales de OpenCV [5] juntamente con el modelo de Caffe. Para hacer esto posible, se necesitan dos ficheros: el primer fichero donde se encuentra el modelo de la arquitectura de la red neuronal y en el cual se definen las capas y los parámetros de la red neuronal, y un segundo fichero donde podemos encontrar el modelo Caffe. Con estos dos ficheros cargados mediante las funciones de OpenCV, esta librería es capaz de crear la red neuronal por la que pasarán las imágenes para detectar los rostros.

Sin embargo, antes de pasar las imágenes por la red neuronal, estas han de ser transformadas en blob mediante OpenCV. Pero ¿qué es un blob y por qué son importantes? Como el objetivo de este documento no es profundizar en el funcionamiento de las Redes Neuronales, solo destacaremos que se usan para preparar las imágenes de entrada para la clasificación de una red neuronal.

De esta manera, cuando tenemos la imagen convertida en blob, la pasamos por la red neuronal y la aplicación se encarga de enmarcar las caras detectadas.

Gracias a todo esto, podemos observar resultados muy buenos. Cabe destacar que es remarcable la velocidad en la que aplicación realiza los cálculos pertinentes en tiempo real para detectar caras con un índice de FPS bastante bueno.

4.3 Reconocimiento facial

El reconocimiento facial es mucho más complejo. El objetivo final del reconocimiento es encontrar un rostro en una imagen y asociarla con una persona.

Su proceso consta de diferentes fases. La primera consiste en generar un modelo embeddings previo. Para generar este modelo se necesita una base de datos con imágenes de los rostros de los posibles usuarios. Para cada una de las imágenes de entrada de la base de datos se realizará un proceso de detección facial explicado en el punto anterior.

Cuando obtenemos el rostro detectado de cada imagen, extraemos la región de la imagen en la que se encuentra la cara y creamos un blob para insertarlo en un segundo modelo.

La segunda fase consta de entrenar este segundo modelo generado para construir el sistema final por el que pasarán las imágenes sobre las cuales se desee realizar reconocimiento facial.

La última fase corresponde a cargar las imágenes a reconocer y, después de realizar detección facial sobre estas, pasarlas por el último sistema creado.

Como podemos apreciar, los pasos a realizar son más extensos. De la misma manera, también nos encontramos con que el reconocimiento facial es más costoso computacionalmente hablando. Por esta razón, ¿conseguiremos realizar este proceso en tiempo real de manera que la experiencia sea satisfactoria para el usuario? Probemos algunos ejemplos.

4.3.1 Ejemplo de la web oficial OpenCV

Este apartado está basado en la experiencia como usuario personal de este ejemplo [3]. En la figura 1 podemos observar que el reconocimiento facial funciona. Sin embargo, la velocidad de ejecución deja mucho que desear. El tiempo en empezar a funcionar es excesivo y, además, durante la ejecución vemos que la aplicación no funciona de manera fluida. ¿Significa esto que no hay manera de desarrollar una herramienta capaz de realizar reconocimiento facial de manera fluida? Veamos otro ejemplo.

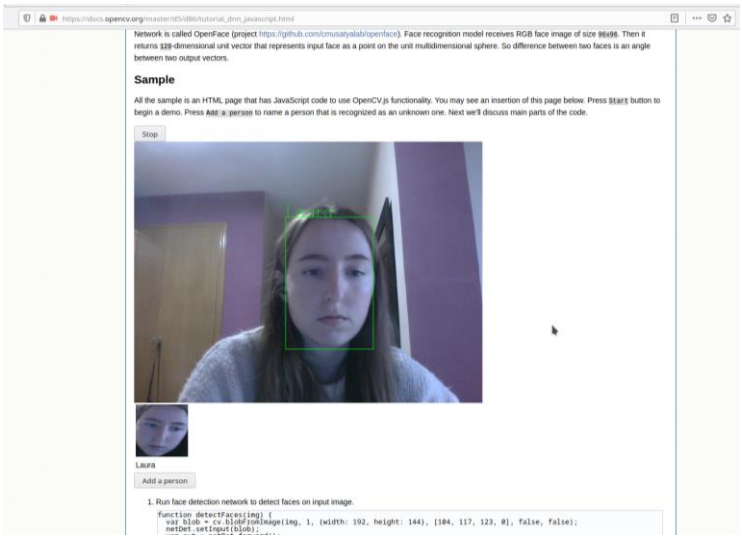


Figura 1: ejecución del ejemplo de reconocimiento facial de la web oficial de OpenCV

4.3.2 Ejemplo de partida

Este apartado está basado en la experiencia como usuario personal del este ejemplo [2]. Como podemos observar en la figura 2, después de añadir diversas fotos en la base de datos de esta aplicación, la aplicación consigue reconocerme.

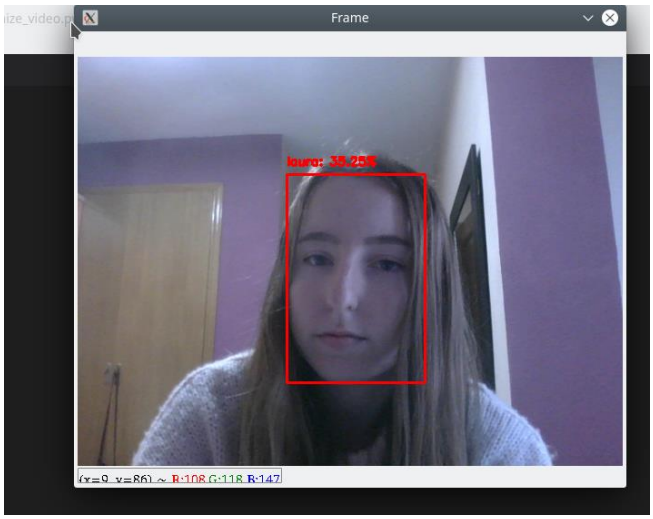


Figura 2: ejecución del ejemplo de reconocimiento facial del ejemplo de partida

Este resultado no es nada nuevo, dado que el reconocimiento ya era conseguido con el ejemplo anterior. Sin embargo, lo impresionante es la fluidez con la que se consigue hacer funcionar la aplicación.

Por esta razón, se ha decidido tomar como ejemplo de partida esta aplicación y realizar las modificaciones necesarias para añadir la nueva funcionalidad.

5 Cambios realizados

En este apartado vamos a comentar los cambios realizados al ejemplo de partida. También estudiaremos las limitaciones que hemos observado y las posibles soluciones. Cabe recordar que el objetivo final era añadir la funcionalidad de gestionar nuevas personas en la base de datos de usuarios a reconocer en tiempo real.

Dado que el ejemplo de partida estaba dividido en diferentes partes, la primera tarea fue integrar todos los subcódigos en uno mismo con el fin de poder volver a generar los modelos necesarios para el reconocimiento facial con solo pulsar un botón.

La segunda tarea consistió en desarrollar una interfaz que permitiera la gestión de ventanas, botones y texto. Dado que el objetivo de este trabajo no es el de explicar las interfaces en python, no voy a profundizar en este concepto.

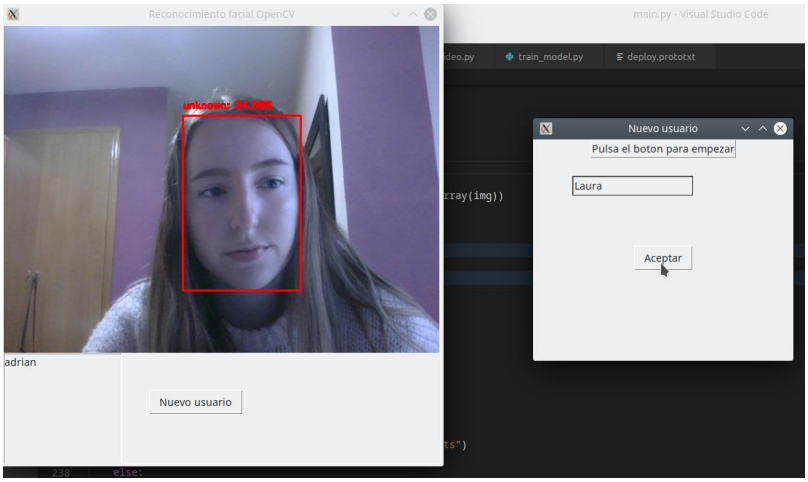
Para añadir una nueva persona correctamente en la base de datos hacen falta diversas imágenes del usuario en cuestión. Por esto, decidimos que, al pulsar el botón, la aplicación fuera capaz de guardar cinco fotos del usuario hechas con un segundo de diferencia para que las imágenes no fueran las mismas. La razón por la que debía de ser más de una es porque, de esta manera, se consigue un mayor índice de aciertos.

Para realizar esto, bastó con utilizar los métodos de OpenCV que permiten extraer un frame de la cámara y guardarlo en la ruta que se desee.

Como resultado se consiguió lo que se observa en la figura 3 y 4. Aquí podemos ver cómo, al pulsar en nuevo usuario, nos aparece otra pestaña donde, después de poner nuestro nombre, guarda las capturas y realiza otra vez el cálculo necesario para realizar el reconocimiento facial. Podemos observar como antes de añadir un nuevo usuario la aplicación no era capaz de asociar la cara con ningún usuario preexistente y, después de guardar el usuario Laura, lo reconocía con este nuevo.

También podemos observar la incorporación de una lista para saber en todo momento los usuarios existentes en la base de datos.

Pese a las nuevas funcionalidades, podemos notar que la aplicación sigue funcionando de manera fluida y con un índice de acierto bastante alto. No obstante, no todo es bueno. Veamos las limitaciones que podemos encontrar.



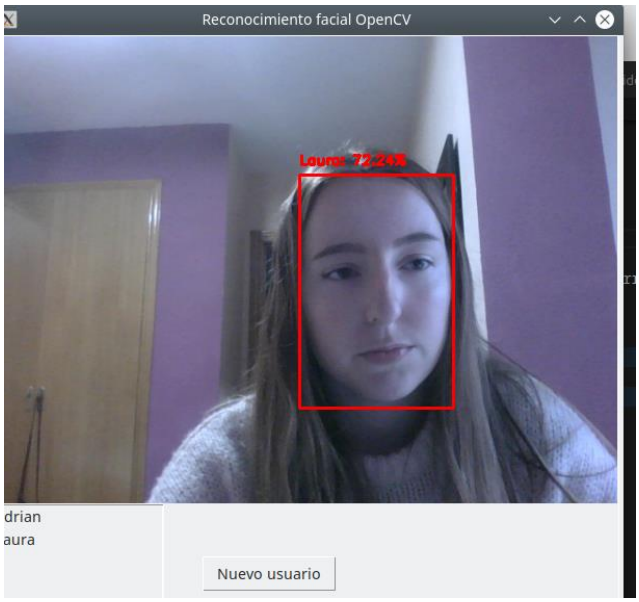


Figura 3 y 4: resultado de las modificaciones

5.1 Limitaciones

La primera limitación que encontramos son los errores de reconocimientos. Podemos observar en la figura 5 como con ciertos movimientos, la aplicación nos confunde con otra persona. Aunque sean muy pocas veces es algo bastante molesto.

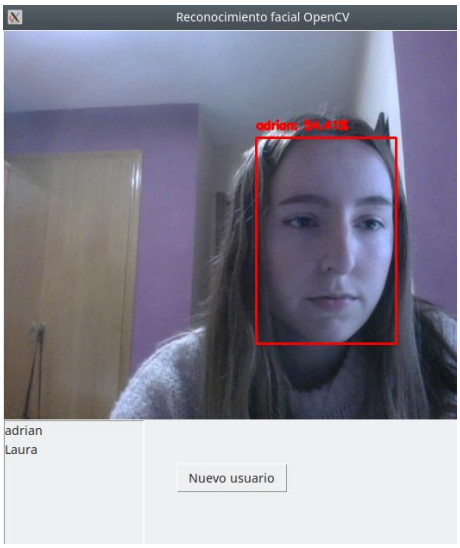


Figura 5: limitaciones de aciertos

¿Y qué pasa si se intenta reconocer una foto? Sabemos que hoy en día, por temas de seguridad, hay algunos teléfonos móviles con desbloqueo facial que no pueden ser burlados con fotos, ¿es este el caso? Si observamos las figuras 6 y 7 podemos saber que la respuesta es que no. Para poder implementar esta se necesitan dispositivos dedicados al reconocimiento facial, como la cámara e iluminador infrarrojo y un proyector de puntos.

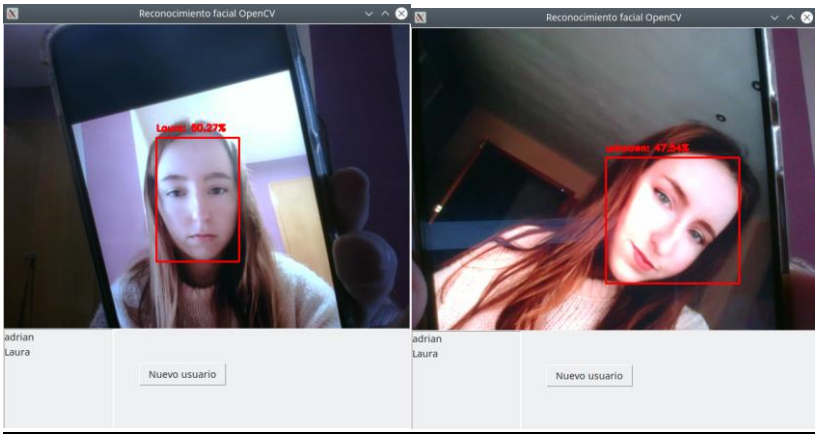


Figura 6 y 7: limitaciones en fotos

Otra limitación importante observada se encuentra en la figura 7. Aunque la persona de las figuras 6 y 7 sean la misma, podemos ver como para el sistema no lo es. Esto ocurre porque la foto de la figura 6 está realizada bajo las mismas condiciones (luz, presencia física) en las que se guardaron las cinco fotos almacenadas en la base de datos. En cambio, la foto de la figura 7 está realizada con condiciones distintas.

Una solución para este problema sería guardar diferentes fotos con diferentes condiciones de la persona en la base de datos. La figura 8 es el resultado de ejecutar dicha solución. En esta figura podemos observar como esta solución mejora los resultados.

No obstante, esto no es posible con la funcionalidad implementada. Aunque la aplicación nos deje un segundo para cambiar de ángulo la cara y así intentar aumentar el índice de acierto, no puede llegar a dar los mismos resultados que si cargamos fotos en entornos diferentes.

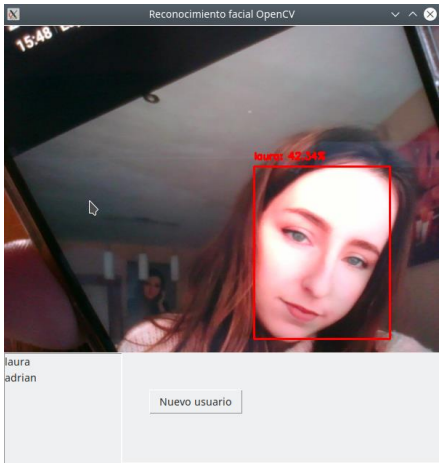


Figura 8: solución a la limitación de condiciones

6 Conclusión

Poner en marcha los ejemplos han servido para entender de manera práctica la detección facial y el reconocimiento facial. Además, hacerlo con el uso de OpenCV ha sumado simplicidad a una tarea que, de primeras, parece más complicada de lo que es.

En cuanto a los resultados, es sorprendente como la aplicación es capaz de realizar todos los cálculos en tiempo real y ofrecer tanta fluidez. Aunque haya aspectos por pulir, en general el resultado me ha parecido bueno.

7 Bibliografía

[1] "OpenCV API Reference" Disponible en:
https://docs.opencv.org/2.4/modules/refman.html

[2] "OpenCV Face Recognition" Disponible en:
https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

[3] "OpenCV Face Recognition sample" Disponible en:
https://docs.opencv.org/master/d5/d86/tutorial_dnn_javascript.html

[4] "Face detection with OpenCV and deep learning" Disponible en: https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/

[5]"OpenCV Deep Neural Networks (dnn module)" Disponible en:

https://docs.opencv.org/master/d2/d58/tutorial_table_of_content_dnn.html