Ejercicios con Python Volumen 1:
Edición y Captura de Imágenes con Python y la librería OpenCV

La librería OpenCV para Python da soporte al desarrollo de aplicaciones de visión artificial, nosotros la utilizaremos para practicar con Python e ir creando nuestra propia librería de funciones para la manipulación de ficheros, algunas de estas funciones nos serán muy útiles cuando trabajemos con las APIs de reconocimiento de imágenes.

Se proponen los siguientes ejercicios, que formarán parte de la librería **imagenes.py**:

1) Crea una función que pasándole la ruta de una imagen, la rote 180 grados y genere una nueva imagen.

f1

f1_r180

2) Crea una función que pasándole la ruta de una imagen, genere una nueva imagen a partir de ella con los colores invertidos.

f1

f1_invcol

3) Crea una función que pasándole la ruta de una imagen, genere una nueva imagen a partir ella pero en escala de grises.

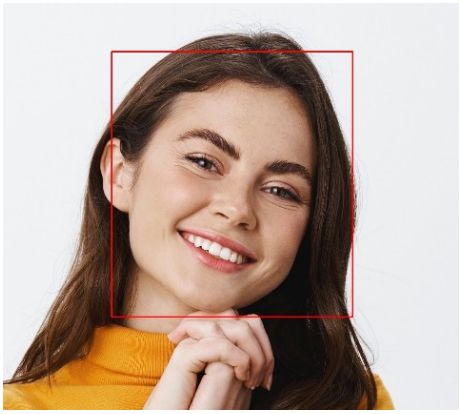
f2

f2_gris

4) Crea una función que pasándole la ruta de una imagen, marque un cuadrado a partir de dos coordenadas.

Esta función no reconoce el rostro, se le han pasado las coordenadas del marco como parámetros.

dibujar_cuadrado("imagenes\\f2.jpg",(1500,250),(2000,800))



5) Crea una función que pasándole la ruta de una imagen, invierta los colores de un cuadrado a partir de dos coordenadas, pasadas por parámetro.

invertir_color_cuadrado("imagenes\\m2.jpg",(400,500),(500,600))

m2

m2_invcua

Se genera una imagen igual pero con los colores invertidos en un determinado cuadrado de la imagen.

6) Crea una función que pasándole la ruta de una imagen, la recorte para evitar dimensiones con valores impares

m1

m1_par

Aparentemente las imágenes son iguales, pero m1_par.jpg es un pixel mas estrecha.

7) Crea una función que pasándole la ruta de una imagen, retorne la imagen espejada

m2

m2_esp

8) Crea una función que pasándole la ruta de una imagen, invierta la mitad izquierda y la copie en la derecha.

m2

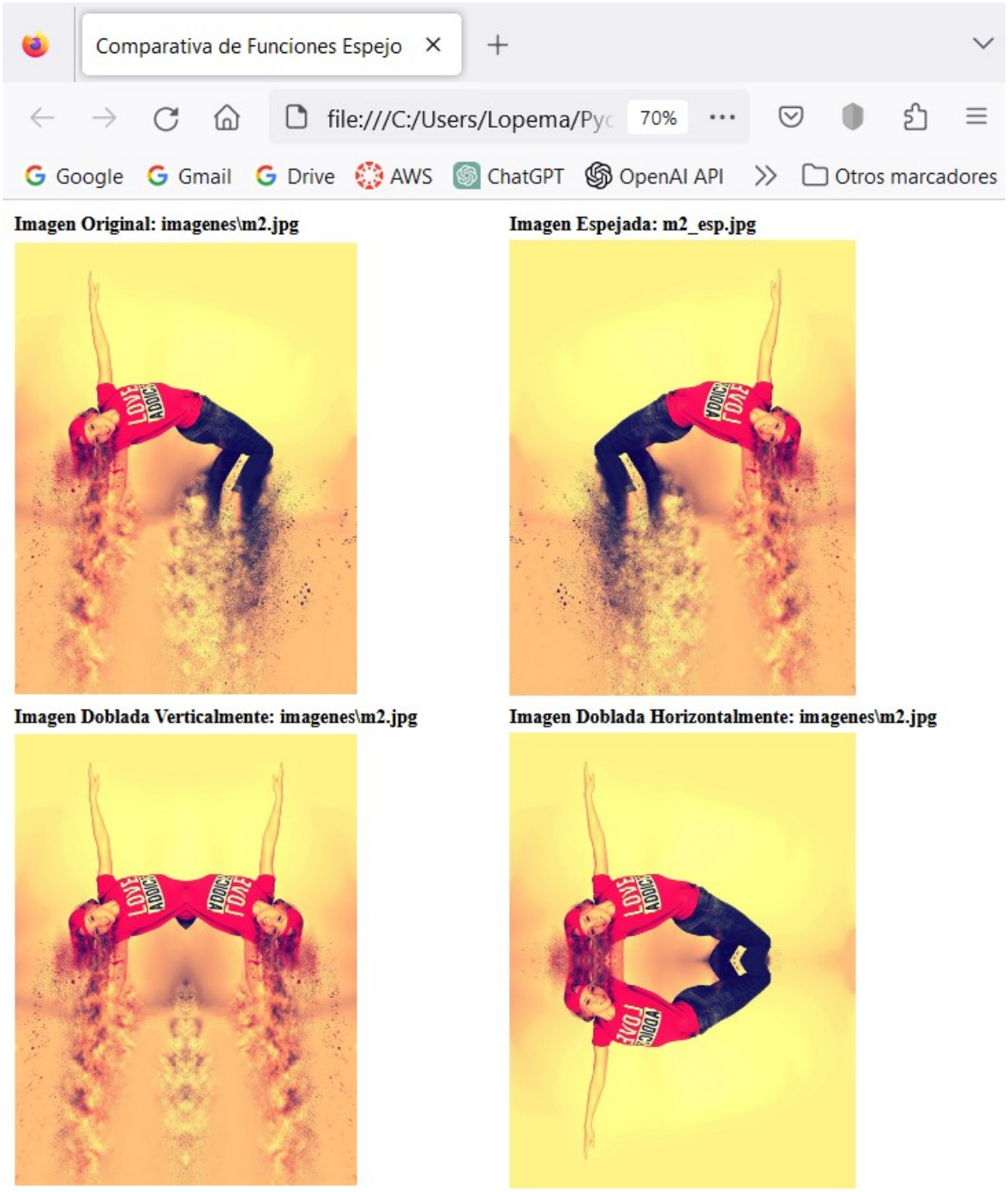
m2_dobver

9) Crea una función que pasándole la ruta de una imagen, invierta la mitad superior y la copie en la inferior, efecto espejo por la horizontal.

m2

m2_dobhor

10) Crea una función que pasándole la ruta de una imagen, genere un documento html donde muestre la imagen original y las generadas en las tres funciones anteriores en una tabla, de la forma:

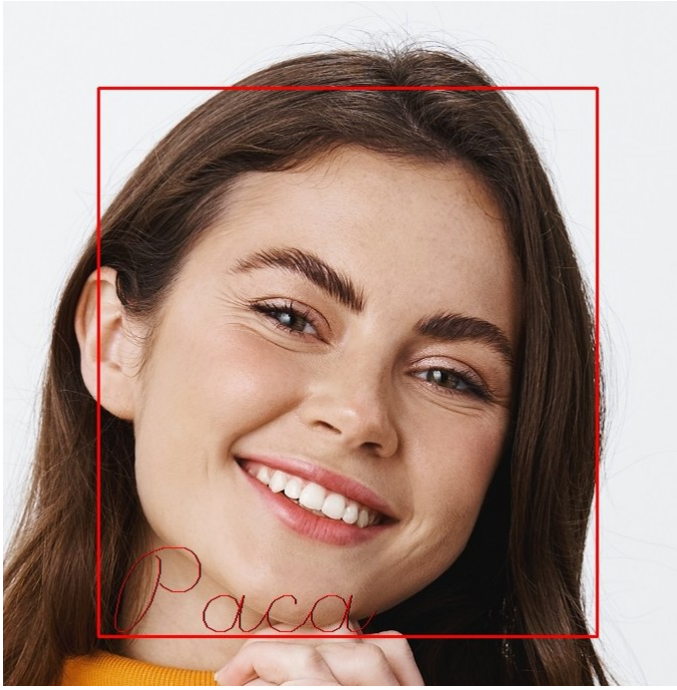


11) Crea una función que pasándole la ruta de una imagen, además de marcar un cuadrado a partir de dos coordenadas, como se hizo en el ejercicio 4, añada un texto en la parte inferior del cuadrado.

Al igual que en el ejercicio 4, realmente no se está detectando caras, se están pasando las coordenadas del marco a la función, además de la imagen y el texto.

dibujar_cuadrado_texto("imagenes\\f2.jpg", (1500,250),(2000,800),"Paca")

La detección de caras lo dejamos para posteriores ejercicios.



12) Crea una función que pasándole la ruta de una imagen, emborrone una zona determinada

f2

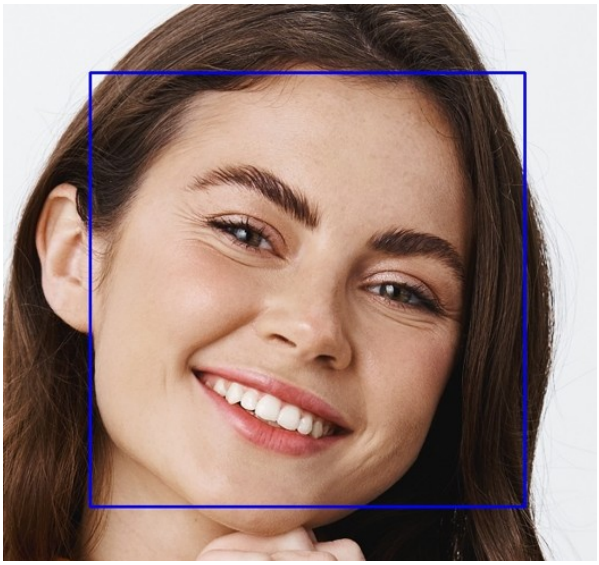
f2_emb

emborronar_cuadrado("imagenes\\f2.jpg",(1500,250),(2000,800))

Para emborronar se ha utilizado la función medianBlur con un tamaño de kernel muy alto.

13) Ahora si. **Crea una función que pasándole la ruta de una imagen, detecte y marque las caras de dicha imagen utilizando la funcionaliad de CV2.** Esta librería posibilita la detección de objetos mediante aprendizaje automático en cascada. Podemos entrenar nuestros propios clasificadores, pero para este ejercicio utilizaremos un clasificador preentrenado que puedes encontrar en el gitHub de OpenCV (opencv/data/haarcascades/)

En este ejercicio utilizaremos el clasificador preentrenado “haarcascade_frontalface_default.xml” para detectar rostros de frente.



Investiga sobre las siguientes funciones:

- cv2.CascadeClassifier('modelos/haarcascade_frontalface_default.xml')
- face_cascade.detectMultiScale(imagen_grises, 1.3, 5)

Obviamente, además de detectarla, la marcaremos, tal como hicimos en el ejercicio 4, pero esta vez no le pasaremos las coordenadas a la función.

Añade un parámetro a la función, que permita además emborronar las caras. Dicho parámetro tendrá un valor por defecto para que solo emborrone si se indica implicitamente.

```
marcar_cara( file: "imagenes\\f2.jpg", emb: True)
```

Básicamente, es lo añadir la funcionalidad que implementamos en el ejercicio 12.

Crea tus propias versiones de esta función, haciéndola mas versátil, por ejemplo, permitiendo también añadir texto, tal como se hizo en el ejercicio 11.

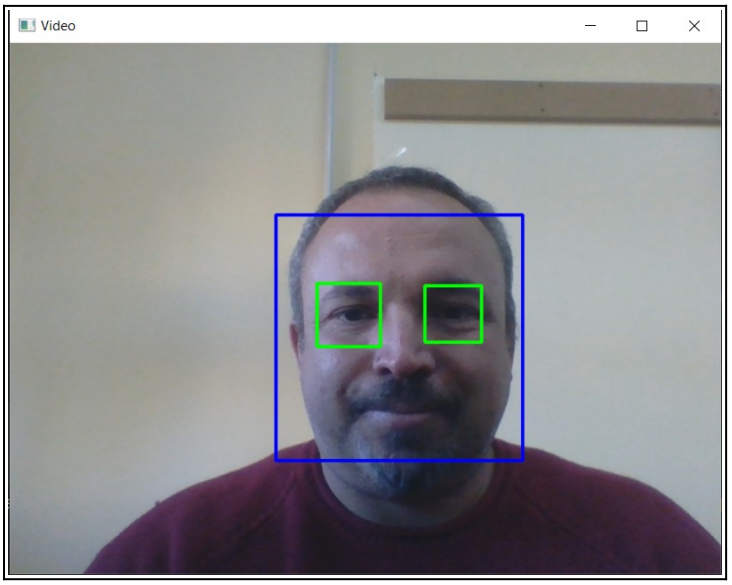
Estos enlaces pueden serte de interés:

Detección de rostros, caras y ojos con Haar Cascad
https://unipython.com/deteccion-rostros-caras-ojos-haar-cascad/

14) Crea una función que realice capturas con la webcam y marque cara y ojos del rostro.

En este ejercicio utilizaremos el clasificador preentrenado “haarcascade_eye.xml” para detectar ojos además del clasificador preentrenado del ejercicio anterior para detectar rostros.

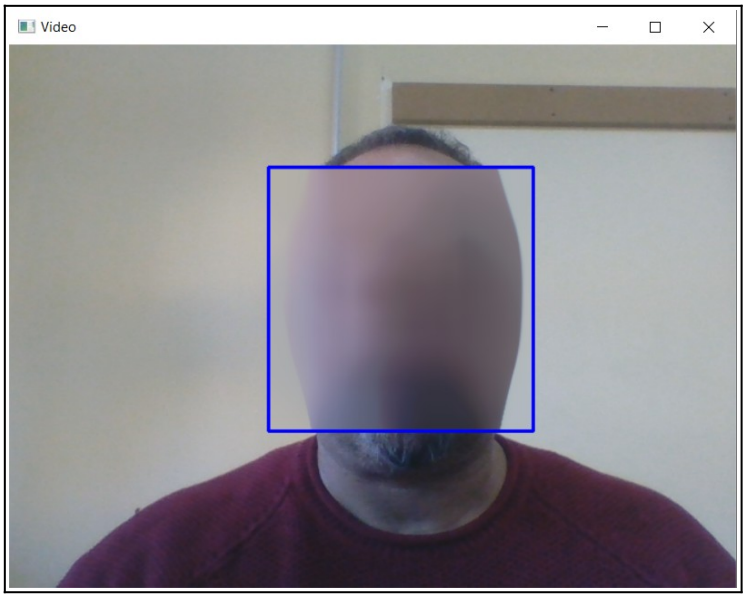
Para capturar imágenes de la webcam utilizaremos la función cv2.VideoCapture(0)



Este ejercicio lo podeis encontrar resuelto en:
https://decodigo.com/python-deteccion-de-rostros-con-opencv
aunque se aconseja intentarlo desde cero.

15) Crea una función que realice una captura con la webcam, como en el ejercicio anterior, pero que esta vez, en lugar de marcarla, la emborrone.

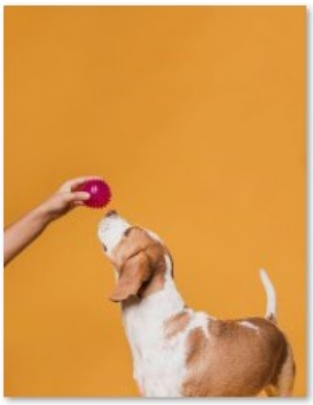
Combinamos lo que hemos hecho en el ejercicio 14 con lo que aprendimos a hacer en el ejercicio 12.



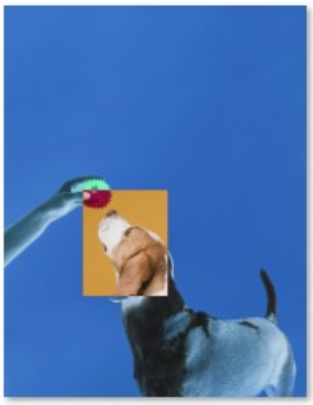
Ejercicios de Ampliación

Se proponen los siguientes ejercicios de ampliación, que formarán parte de la librería imagen2.py:

5b) Como variante del ejercicio 5 se propone invertir los colores de toda la imagen menos del marco indicado. Añadirlo como funcionalidad extra, es decir, por defecto la función operará como hasta ahora pero podrá pedírsele que realice la operación aquí descrita.



p1



p1_invcua

El ejercicio 5 invertía los colores de un marco indicado por unas coordenadas pasadas por parámetro, vamos a añadir un parámetro para determinar si se invertirá el marco o el resto de la imagen. La llamada a la función, ahora tendrá la forma:

```
invertir_color_cuadrado("imagenes\\p1.jpg",(1500,3500),(3100,5500),True)
```

11b) Como variante del ejercicio 11, se propone transformar toda la imagen fuera del marco a tonos de gris. Añadirlo como funcionalidad extra.

11c) Como variante del ejercicio 11, se propone que la imagen generada contenga solo el marco seleccionado, incluyendo un perímetro en tonos de grises, además la etiqueta podrá estar vacía.

11d) Como variante del ejercicio 11 se propone que el texto sea opcional y se ajuste al tamaño del cuadro.

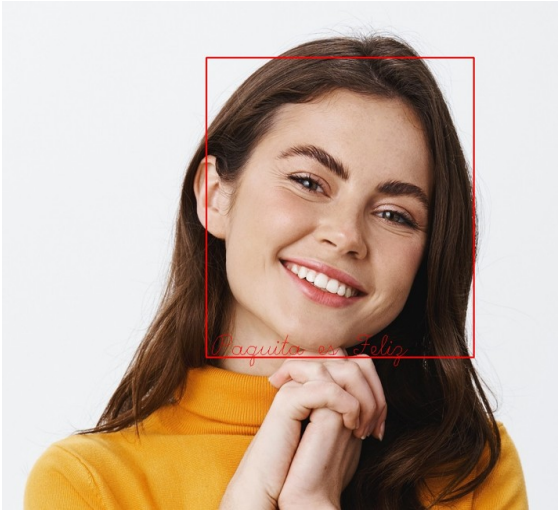
La nueva función tendrá varios modos de ejecución:

Modo normal, sin pasar a escala de grises y sin recorte.

```
dibujar_cuadrado_texto( file: "imagenes\\f2.jpg", inicio: (1500,250), fin: (2000,800), texto: "Paca", gris: False, recortar: False)
```

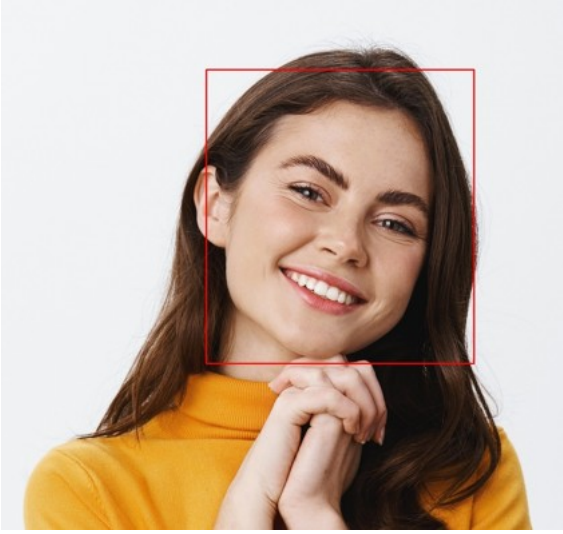
Ajustando la fuente al tamaño de la etiqueta.

```
dibujar_cuadrado_texto( file: "imagenes\\f2.jpg", inicio: (1500,250), fin: (2000,800), texto: "Paquita es Feliz", gris: False, recortar: False)
```



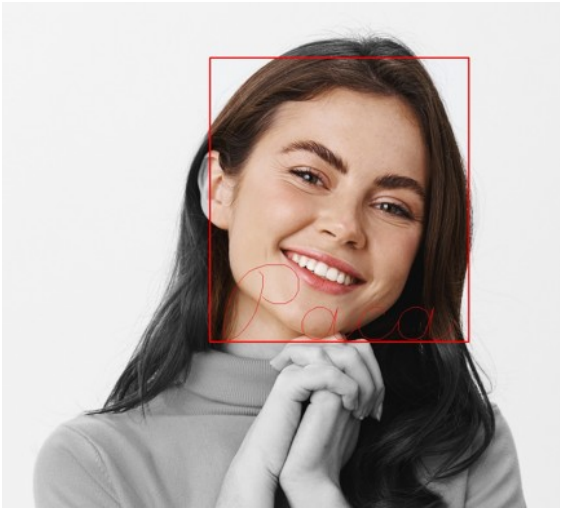
Permitiendo la omisión de la etiqueta.

```
dibujar_cuadrado_texto( file: "imagenes\\f2.jpg", inicio: (1500,250), fin: (2000,800), texto: "", gris: False, recortar: False)
```



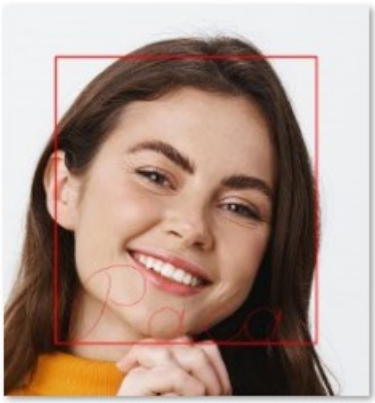
El modo a escala de grises, devuelve la imagen fuera del marco en tonos de grises.

```
dibujar_cuadrado_texto( file: "imagenes\\f2.jpg", inicio: (1500,250), fin: (2000,800), texto: "Paca", gris: True, recortar: False)
```



Por último, con el modo recorte activo, la imagen se recortará al marco indicado mas un marge (con o sin texto) y funcionará tanto con el resto de la imagen en su color original o a escala de grises.

```
dibujar_cuadrado_texto( file: "imagenes\\f2.jpg", inicio: (1500,250), fin: (2000,800), texto: "Paca", gris: False, recortar: True)
dibujar_cuadrado_texto( file: "imagenes\\f2.jpg", inicio: (1500,250), fin: (2000,800), texto: "Paca", gris: True, recortar: True)
```



f2_txt1

f2_txt2

En este caso se ha dado un margen de 100 pixeles sobre el marco.

14b) Revisa el resto de clasificadores preentrenados de Haar en la librería OpenCV para detectar otros elementos. Por ejemplo la boca. Investiga como podrías entrenar tu propio modelo.