Práctica 0

Visión por Computador

Grupo de Prácticas 2

Juan José Herrera Aranda - 26516011-R
juanjoha@correo.ugr.es

29 de septiembre de 2021

Índice

La práctica se ha realizado y ejecutado en un ordenador portátil con distribución Ubuntu 20.4, procesador Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz , con una memoria RAM de 16GB de tipo DDR4 y una arquitectura x86_64.

1. Ejercicio 1

Escribir una función que lea el fichero de una imagen y permita mostrarla tanto en grises como en color (im=leeimagen(filename, flagColor)).flagColor es la variable que determina si la imagen se muestra en escala de grises o en color

Básicamente lo que hacemos en el ejercicio es leer un fichero imagen y un flag y mostramos la imagen a color o en escala de grises. Hemos llamado a la función **imread** 6.1 de opencv para leer la imagen, notamos que la lee en formato BGR en vez de RGB. A continuación, obtenemos el número de canales para saber si la imagen es tribanda o monobanda.

En caso de ser monobanda y estar activado el flagColor, lo ignoraremos y mostraremos la imagen a escalas de grises. En caso contrario, si la imagen es tribanda, no está en escala de grises y estar el flagColor desactivado, mostraremos la imagen en color y en escala de grises.

La transformación de una imagen tribanda a escala de grises lo realizamos con la función **cvtColor** con el parámetro **COLOR_BGR2GRAY** 6.1.

Si la imagen es monobanda la visualización con matplotlib no tiene enjundia, en cambio, si es tribanda y hemos leído con opencv tenemos que hacer una inversión de los canales, para ello hay dos posibilidades:

- Usando **cvtColor(imagen,COLOR_BGR2RGB)**
- Leyendo la matriz imagen de la forma imagen **[:,:,::-1]**

Además, tenemos que tener en cuenta que para que imprimir una imagen a escala de grises en matplotlib debemos añadir el parámetro **cmap='gray'**, si no lo hacemos la imagen se imprime con una tonalidad ultravioleta (no deseada)

Mostramos a continuación los resultados:

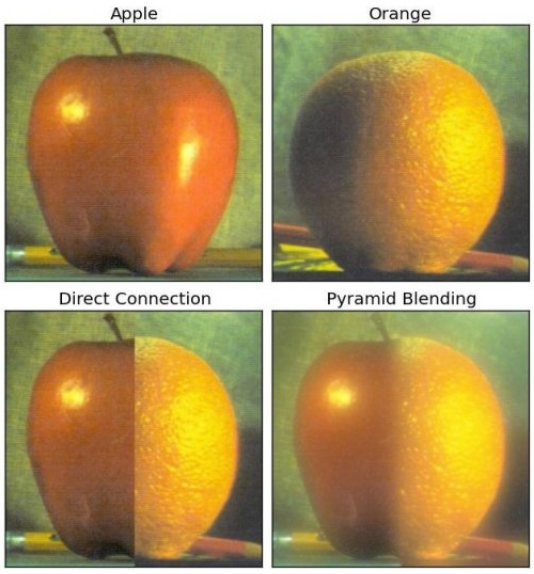

Figura 1: Imagen a color (original)

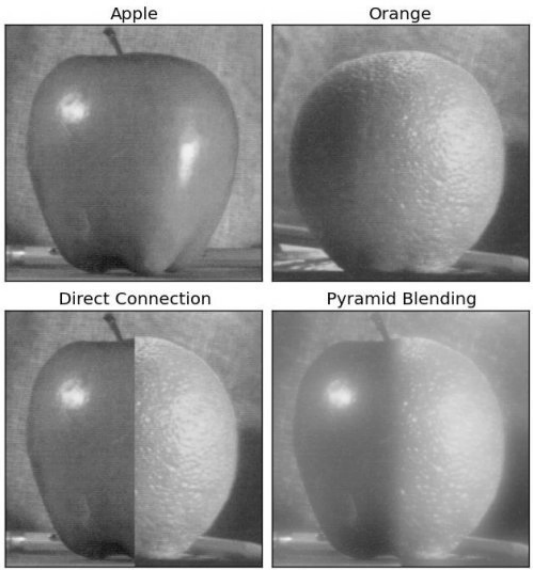

Figura 2: Imagen a escala de grises

2. Ejercicio 2

Escribir una función que permita visualizar una matriz de números reales cualquiera/arbitraria, tanto monobanda como tribanda (pintaI(im)). Para ello se deberá escalar el rango de cada banda al intervalo [0,1] sin pérdida de información

La resolución de este ejercicio presenta una función que recibe dos matrices, una monobanda y otra tribanda. Normaliza las matrices para que sus valores queden comprendidos entre [0,1] aplicando la siguiente fórmula

$$mat\_normalizada = \frac{mat - \text{mín}(mat)}{\text{máx}(mat - \text{mín}(mat))}$$

Posteriormente se han creado de forma aleatorio dos matrices de números aleatorios y de tamaño 3x3. Comentar también que los valores de las matrices originales son enteros comprendidos entre [0,255] y al normalizarse, éstos pasaran a ser flotantes pertenecientes al intervalo [0,1]. Por tanto, vemos que hay dos posibles formas de representar una matriz, con enteros y con flotantes.

```
[[0.36165947 0.14594594 0.37429503]
 [0.         0.03532438 1.        ]
 [0.60812794 0.43066434 0.8158411 ]]
==========================================================

[[[5.81355264e-01 5.66254169e-01 3.30313179e-01]
  [9.55596628e-01 1.13629205e-01 9.82882073e-01]
  [1.50452871e-01 7.45601625e-01 7.47042414e-01]]

 [[1.24362588e-01 1.18908045e-01 9.77312851e-01]
  [4.53206040e-01 0.00000000e+00 8.65417386e-01]
  [1.00000000e+00 1.50590809e-01 6.64011853e-01]]

 [[8.75668041e-01 2.53906194e-01 2.51761801e-01]
  [5.94741661e-01 8.89795290e-02 7.76633908e-01]
  [8.57954777e-04 8.37335073e-01 8.30832779e-01]]]
```

Figura 3: Matrices normalizadas monobanda y tribanda respectivamente

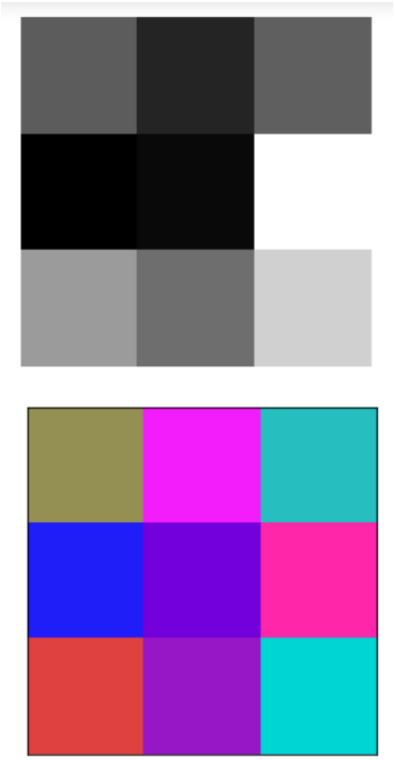

Figura 4: Representación visual de las matrices monobanda y tribanda respectivamente

3. Ejercicio 3

Escribir una función que visualice varias imágenes distintas a la vez (concatenando las imágenes en una última imagen final 1): pintaMI(vim). (vim será una secuencia de imágenes) ¿Qué pasa si las imágenes no son todas del mismo tipo (nivel de gris, color, blanco-negro)?

Para realizar este ejercicio tenemos que tener una serie de factores en cuenta. Por una parte, las imágenes deben de tener la misma dimensión (filas y columnas de la matriz) por lo que debemos de hacer un redimensionamiento común. En este caso, hemos tomado el número máximo de píxeles de las filas (**heigth**) de todas las matrices y también de las columnas (**width**). Tras ello, hemos redimensionado todas las imágenes a esos dos valores. Hemos optado por esta elección de los valores porque de esta forma la imagen no pierde calidad, ya que si una imagen la redimensionamos con menos píxeles de los que tiene, empeora.

Por otra parte debemos tener en cuenta si hay imágenes que sean monobanda y tribanda a la vez. En ese caso, debemos añadir dos canales más a las imagenes con solo uno para que de esta forma tengan tres. Para ello usamos la función **cvtColor** uasando el parámetro **COLOR_GRAY2BGR** 6.1.

Una vez realizado todo lo anterior procedemos a la concatenación. Hay dos formas de concatenar, horizontal y verticalmente. En este caso la concatenación se ha hecho horizontal. Las funciones empleadas son **hconcat** o **vconcat**. 6.1

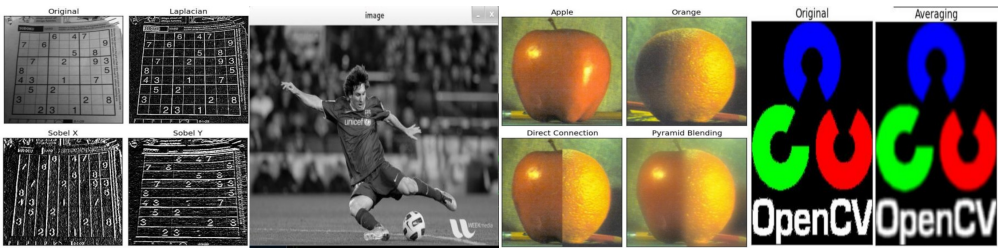

Figura 5: Imagen concatenada

4. Ejercicio 4

Escribir una función que modifique el color en la imagen de cada uno de los elementos de una lista de coordenadas de píxeles. En concreto, los alumnos deben insertar un cuadrado azul de 100x100 píxeles en el centro de la imagen a modificar

Para la realización del ejercicio, se ha calculado el número de filas y de columnas que tiene la imagen y haciendo una división entera por dos, obtenemos el punto medio de la imagen. El cuadrado azul será aquel con vértices (punto_medio_abcisas - 50, punto_medio_abcisas + 50, punto_medio_ordenadas - 50, punto_medio_ordenadas + 50). Finalmente, tenemos en cuenta que al ser la imagen leída con opencv, está en formato BGR, por lo que el color azul corresponderá con la terna[255,0,0].

Como comentario, decir que en el ejercicio se ha hecho para imágenes que son tricanal. En caso de que se pasara como argumento una imagen monocanal, no funcionaría pues el cuadrado azul se imprimiría también en escala de grises.

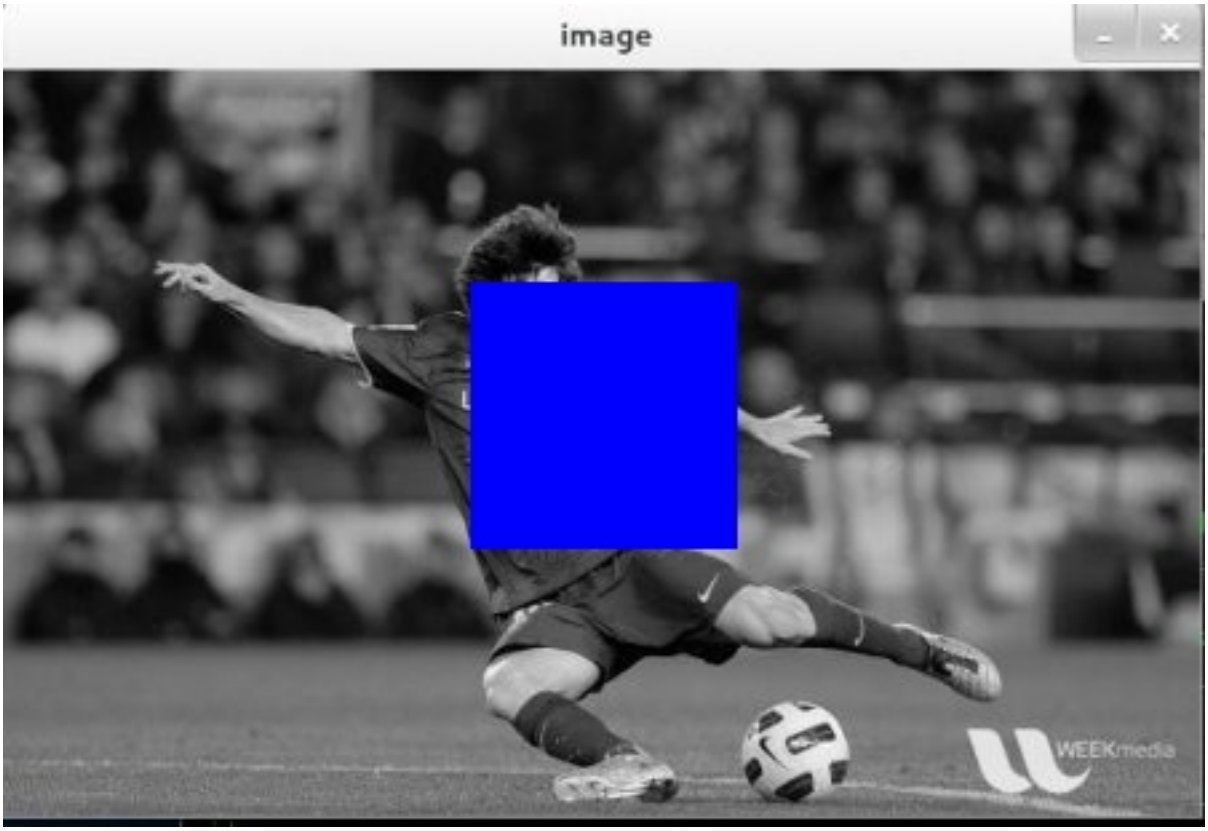

Figura 6: Imagen con un cuadrado azul de 100x100 en el centro

5. Ejercicio 5

Una función que sea capaz de representar varias imágenes con sus títulos en una misma ventana

Hemos definido una función con tres parámetros, un vector con las imágenes, otro vector con los títulos y el número de filas que queremos que tenga la representación. Con el número de filas calculamos el número de columnas haciendo la siguiente división $\frac{num\_images}{num\_filas}$. A continuación dividimos la ventana en subventanas con la función de matplotplib.pyplot **subplot** y finalmente indicamos que se va a imprimir en esa ventana junto con su correspondiente título.

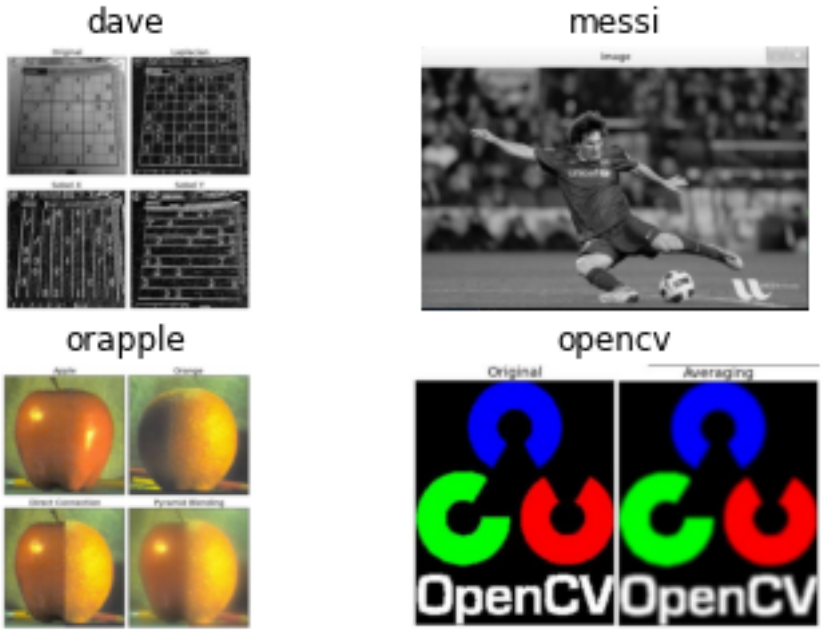

Figura 7: Varias imágenes en una misma ventana con sus respectivos titulos

6. Anexos

En esta parte se comentarán las funciones más importante que se han usado a la hora de realizar los ejercicios y que han sido clave en la realización de los mismos.

6.1. OPENCV

- **cv.imread(filename[, flags])** : Carga una imagen de un archivo pasado en el primer argumento y la devuelve. Si el archivo no puede ser leido se devolverá una matriz vacía. En caso de que la imagen sea a color, los canales serán de la forma BGR. El segundo argumento es un flag del tipo **ImreadModes**:

- **cv.cvtColor(src, code[, dst[, dstCn]]) -> dst** : Convierte una imagen de un espacio de color a otra. Devuelve una imagen transformada

  - El primer parámetro es la imagen de entrada
  - **dst** es la imagen de salida del mismo tamaño y profundidad que src (opcional).
  - **code** es el espacio de color al cual queremos ver la conversión, toma valores:
    - **COLOR_BGR2RGB**
    - **COLOR_RGB2GRAY**
    - **COLOR_BGR2GRAY**
    - **COLOR_GRAY2RGB**
  - **dstCn**: número de canales en la imagen de distino, si el parámetro vale 0, el número de canales se toma automáticamente desde el primer parámetro y el segundo (opcional)

- **cv.hconcat(src[, dst]) -¿dst**: Aplica una concatenación horizontal de un array de imágenes. Su análogo vertical se llama **vconcat**

6.2. MATPLOTLIB.PYPLOT

- **imshow (X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None, vmin=None, vmax=None, origin=None, extent=None, *, filternorm=True, filterrad=4.0, resample=None, url=None, data=None, **kwargs)** : Muestra una imagen por pantalla, en 2D. La entrada puede ser RGB(A). Para mostrar escala de grises tenemos que usar los parámetros cmap='gray', vim=0, vmax=255.