Visión por Computador
grado en ingeniería informática

PRÁCTICA 0 (BONUS)

Realizado por:
Alba Casillas Rodríguez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2021-2022

Índice

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.

Para el desarrollo de este primer ejercicio, se han definido dos funciones, la primera de ellas: **leeimagen(filename, flagColor)** hará uso de una función definida por la librería **OpenCV**:

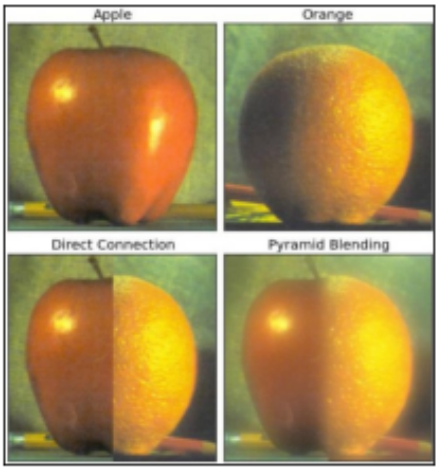
cv2.imread(filename, flagColor)

la cual carga la imagen especificada por el parámetro *filename*.

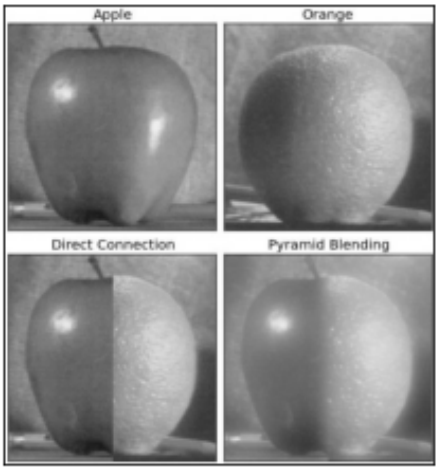
Además, con el parámetro *flagColor* se indicará si la imagen será mostrada a color, mediante el valor **cv2.IMREAD_COLOR** (valor por defecto); o en escala de grises, especificándo en el parámetro el valor **cv2.IMREAD_GRAYSCALE**.

Y otra función **mostrarimagen(im)** que hace uso de las funciones proporcionadas por la librería **matplotlib** para mostrar imágenes. Para ello, tenemos en cuenta que **OpenCV** almacena las imágenes en BGR en lugar de RGB, por lo cual debemos hacer una conversión mediante la función que encontramos en **OpenCV**:

cv2.cvtColor(im, cv2.COLOR_BGR2RGB).



(a) Muestra de la imagen a color.



(b) Muestra de la imagen en escala de grises

Figura 1: Visualización de orapple.jpg

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 idea de este ejercicio se reduce en normalizar la matriz de la imágen al intervalo [0,1].

Si simplemente se dividiese entre 25 (valor máximo del píxel), se perdería información, ya que no estaríamos teniendo en cuenta que posiblemente los valores mínimo y máximo de la matriz no sean 0 y 255.

Por tanto, para tener en cuenta los valores máximo y mínimo de la matriz, se realiza la normalización con la siguiente fórmula:

$$m\_normalizada = \frac{m - m_{min}}{m_{min} - m_{max}} \tag{1}$$

3. Ejercicio 3

Escribir una función que visualice varias imágenes distintas a la vez (concatenando las imágenes en una última imagen final): 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 la concatenación de imágenes, primero se ha realizado la lectura de las imágenes a concatenar, haciendo uso de la función del *ejercicio 1*. Una vez leídas todas las imagenes, formamos una lista, la cuál será el parámetro que le proporcionaremos a la función: **pintaMI(vim)**

Para la concatenación de las imágenes, se usará la función **hconcat** disponible en **OpenCV**. Como parámetro de esta función, se le pasará la lista de imágenes en forma de matrices, las cuales necesariamente deben tener igual *dimensión* y *profundidad*.

Al principio del desarrollo de este ejercicio, al no cumplir con estos requisitos obtuve el siguiente error:
error:(-215:Assertion failed) src[i].dims $\leq$ 2 && src[i].rows == src[0].rows && src[i].type() == src[0].type() in function 'cv::hconcat'

Para igualar la profundidad, a las matrices monobanda (dimension 2) le he aplicado la función ya utilizada en ejercicios anteriores **cv2.cvtColor(imagen, cv2.COLOR_GRAY2BGR)**, de manera que la matriz pasará a ser tribanda para el programa. Mientras que, para que todas las imágenes tengan la misma dimensión, he obtenido la altura máxima de todas las imágenes de la lista, y a aquellas de menos tamaño, se le ha añadido un borde. Para ello, se ha hecho uso de la función **copyMakeBorder**, cuya especificaciones se puede encontrar en la referencia bibliográfica [4].

Una vez realizada la concatenación, se mostrará el resultado en una ventana externa debido al uso de las funciones para mostrar imágenes de **OpenCV**.

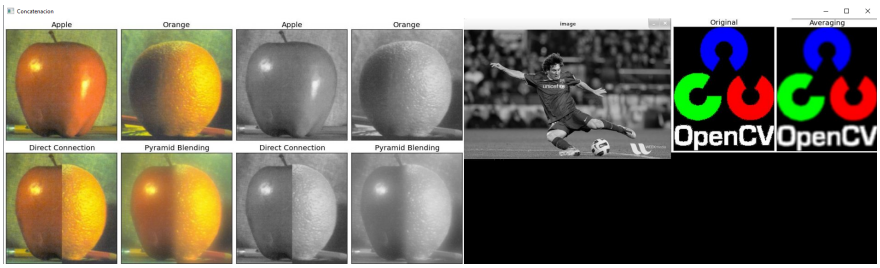


Figura 2: Concatenación de imágenes de distintas características.

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 pixeles en el centro de la imagen a modificar.

Siguiendo la misma dinámica descrita en la documentación de apoyo para la realización de esta práctica, primero se leerá y normalizará (dividiendo entre 255) la imágen deseada.

Debemos tener en cuenta, que para una imágen las coordenadas son *(fila=y, columna=x)*, de manera, que para modificar ciertos píxeles de una imagen tendremos en cuenta la siguiente estructura:

imagen[Y_inicio:Y_final , X_inicio:X_final, :] = [B, G ,R]

haciendo el inciso de que se considerará BGR en vez de RGB ([1,0,0] el color azul) ya que posteriormente el resultado será mostrado con las funciones proporcionadas por **OpenCV**.

Para que el cuadrado resulte en el centro de la imagen, primero obtenemos el ancho y el largo de la matriz, y tras esto, a cada una de esas dos dimensiones le restaremos el tamaño que debe ocupar el cuadrado (100 pixeles) y dividiremos entre dos.

El resultado de esa operación marcará el inicio (en anchura y altura) de la figura, y para calcular el final de esta, solo tendremos que sumar el número de píxeles a colorear; obteniendo el siguiente resultado:

Figura 3: Cuadrado azul de dimensión 100x100 sobre la imágen de Messi.

5. Ejercicio 5

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

Para la resolución de este ejercicio, se hace uso de la creación de una figura formada por varios *subplots*, una por cada imagen; ya que de esta forma se mostrarán todas las imágenes en una misma ventana con sus titulos correspondientes.

A parte de una lista de imágenes, se le pasará a la función otra lista con los titulos correspondientes de cada una.

Los títulos se añadirán a cada subplot mediante la función **set_title**.

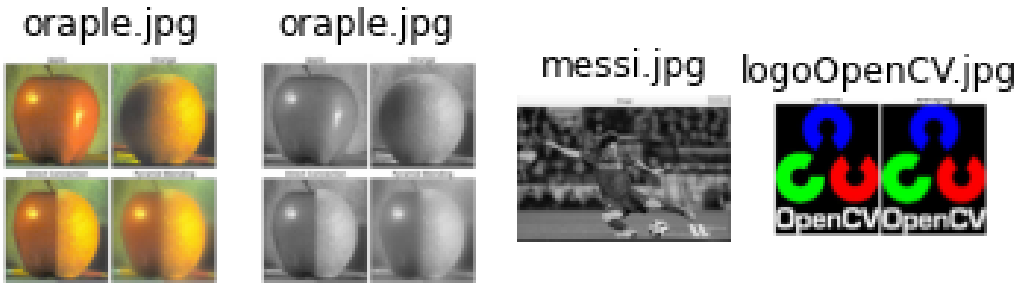


Figura 4: Visualización de una serie de imágenes con sus títulos correspondientes.

Referencias

[1] imread. *imread method specification*
https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/

[2] Normalization. *How to normalize pixel values*
https://machinelearningmastery.com/how-to-manually-scale-image#-pixel-data-for-deep-learning/

[3] Hconcat. *How to concat images*
https://docs.opencv.org/3.4/d2/de8/group__core__array.html#gaf9771c991763233866bf76b5b5d1776f

[4] copyMakeBorder. *How to add a border around an image*
https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga2ac1049c2c3dd25c2b41bffe17658a36

[5] subplots *Display multiple images in one figure*
https://www.delftstack.com/es/howto/matplotlib/how-to-display-multiple-images-in-one-figure-correctly-in-matplotlib/