Visión por Computador: Trabajo 0

Álvaro Fernández García

Septiembre 2018

Nota: Si se desea ejecutar el código para comprobar su funcionamiento, al principio del script he colocado una variable llamada "IMG", que deberá contener la ruta a la imagen con la que se desea probar.

1. Escribir una función que lea el fichero de una imagen y la muestre tanto en grises como en color (im=leeimagen(filename, flagColor))

   Basta con utilizar la función cv2.imread(), expecificando la ruta a la imagen como primer parámetro, y flagColor como segundo. Si flagColor es igual a 1, la imagen se leerá a color (concretamente en formato BGR). Si por el contrario flagColor es 0, la imagen se leerá solo en escala de grises.

   A continuación, llamamos a la función que se construye en el Ejercicio 2, para mostar la imagen.

2. Escribir una función que visualice una matriz de números reales cualquiera ya sea monobanda o tribanda (pintaI(im))

   En primer lugar mostramos la imagen con cv2.imshow(), recibiendo un título para la ventana y la matriz que continene la información de la imagen. A continuación llamamos a cv2.waitKey(0), que esperará los milisegundos indicados como parámetro a que se pulse una tecla (si ponemos 0, como en nuestro caso, esperará un tiempo indefinido). Por último, llamamos a cv2.destroyAllWindows() para cerrar las ventanas.

   El efecto es que se muestra la imagen, y tras pulsar cualquier tecla se cierra.

3. Escribir una función que visualice varias imágenes a la vez: 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)?

   El proceso es sencillo, basta con concatenar horizontalmente (con cv2.hconcat()) todas las matrices de imágenes contenidas en "vim"para crear una sola matriz, y posteriormente mostrarla.

Previamente realizo dos comprobaciones: la primera que la lista "vim"no esté vacía, y la segunda que si sólamente contiene una imagen, simplemente la mostramos.

Con respecto a la pregunta, los píxeles de las imágenes en escala de grises están formados por un sólo número entero que determina la intensidad, mientras que por el contrario, en las imágenes a color, cada píxel está formado por tres números: azul, verde y rojo (BGR). Si intentamos concatenar dos imágenes de distinto tipo, nos encontramos con que la tercera dimensión de la matriz imagen no coincide (1 para Gris, 3 para BGR). Además tampoco podrían concatenarse dos imágenes con distinta altura.

4. Escribir una función que modifique el color en la imagen de cada uno de los elementos de una lista de coordenadas de píxeles. (Recordar que (fila, columna) es lo contrario a (x,y). Es decir fila=y, columna=x)

   Como ya se ha adelantado en el ejercicio anterior, la posición (x,y) de la matriz imagen, representa al píxel de la fila y-ésima y la columna x-ésima , cuyo contenido será el valor del color en BGR si la imagen es en color, o el valor de la intensidad si es en escala de grises. Por tanto, para resolver el ejercicio, basta con recorrer todos los píxeles de la lista de entrada, indexar la matriz con ellos, y asignar el nuevo color.

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

   He realizado este apartado de dos maneras:

   - Empleando MatplotLib: Utilizamos subplot para colocar las distintas imágenes con plt.imshow(), especificamos el título de cada imagen con plt.title() y eliminamos los ticks de los ejes x e y para que no parezcan gráficos. Solo tiene una pequeña dificultad, y es que para mostrar las imágenes a color, es necesario pasar del BGR con el que trabaja OpenCV a RGB, que es lo que emplea MatplotLib, aunque por contrapartida, tiene como ventaja que podemos colocar imágenes a color y en escala de grises, así como imágenes con distintas alturas.
   - Empleando OpenCV: El proceso es un poco más costoso, y además tiene la gran desventaja de que como se utiliza el método del Ejercicio 3, ni se pueden mostrar imágenes de distinto tipo, ni con distintas alturas. No obstante el procedimiento es el siguiente: Procesamos cada imagen individualmente, añadiéndole en primer lugar un marco con cv2.copyMakeBorder(); a continuación le añadimos un pequeño "layout"superior a la imagen (que será donde se colocará el texto). Para ello creamos un array de numpy vacío de 50xAnchoImagenx3, le asignamos el color blanco, y se lo colocamos a la imagen en la parte superior (con cv2.vconcat()). Por último, añadimos el título en el layout blanco con cv2.putText().

     Una vez procesadas todas las imágenes, las concatenamos con la función del Ejercicio 3.