E1-Procesamiento básico

Instrucciones

En respuesta a las preguntas que se plantean a continuación, se debe entregar en la plataforma de enseñanza virtual (por separado, **no** en un archivo comprimido)

- un archivo con extensión .ipynb creado con notebook Jupyter de Anaconda que contenga, tanto las respuestas teóricas, como el código programado;
- el archivo anterior en .pdf (File->Print preview->guardar como pdf);
- una declaración de autoría firmada por el alumno/a según la plantilla proporcionada;
- las imágenes/vídeos usados.

La resolución de todos los ejercicios (tanto teóricos como prácticos) debe ser **razonada** y el código desarrollado debe ser **explicado en detalle, justificando** todos los parámetros escogidos y referenciando las fuentes.

Objetivo

El objetivo de este entregable es el de instalar la librería que se pretende usar a lo largo de la asignatura para manipular imágenes y realizar algunos ejercicios básicos para así afianzar los conceptos teóricos explicados en los temas 1 y 2.

Instalación de OpenCV y manipulación de video [0,1 puntos]

El lenguaje de programación recomendado es Python. Asimismo, se recomienda instalar el entorno de programación Anaconda que trae incluida la librería OpenCV, que es la que vamos a usar para manipular imágenes. Para ello, sigue los siguientes pasos:

- Instalar Anaconda con la versión de Python 3.8: https://www.anaconda.com/products/individual
- Abrir Anaconda prompt e instalar la librería numpy (para cálculo científico) y opencv (para tratamiento de imágenes):

  `conda install -c anaconda numpy`
  https://anaconda.org/anaconda/numpy
  `conda install -c conda-forge opencv`
  https://anaconda.org/conda-forge/opencv
- Abrir Anaconda Navigator y crear un notebook nuevo en Jupyter (se crea con extensión .ipynb).
- Tutoriales básicos:
  - NumPy quickstart: https://numpy.org/devdocs/user/quickstart.html

- OpenCV-Python Tutorials: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

A continuación, haz un pequeño ejercicio como introducción a la manipulación de vídeo (https://docs.opencv.org/master/dd/d43/tutorial_py_video_display.html):

- Grábate en vídeo sosteniendo un objeto de un color homogéneo (ni blanco ni negro), que destaque sobre el fondo, y moviéndolo por la pantalla.
- Escribe un pequeño código que, tomando el vídeo a color, lo convierta a escala de grises y lo muestre en una ventana.

Ejercicio 1. Modelos de color [0,4 puntos]

El modelo HSV es uno de los más adecuados para seleccionar objetos según su color. Selecciona un frame determinado del vídeo en el que aparezca el objeto y conviértelo al modelo de color HSV.

a) Utiliza el nuevo modelo de color para seleccionar el objeto que sostienes en el vídeo, razonadamente. Es decir, explica qué tipo de información codifica cada canal y cómo se puede determinar un rango de valores apropiados en cada uno. Procesa la imagen para que sólo aparezcan en su color original los píxeles que pertenecen al objeto, y el resto de los píxeles sean negros.

b) Razona por qué el modelo de color HSV puede ser más útil para esta tarea que el RGB.

Ayuda: Changing color spaces
https://docs.opencv.org/4.5.2/df/d9d/tutorial_py_colorspaces.html

Ejercicio 2. Codificación de Huffman [0,7 puntos]

Genera una imagen img en forma de matriz de 8x8 con valores enteros aleatorios entre 0 y 9 usando np.random.randint(). La imagen diferencia Dimg la calcularemos de la siguiente forma:

- Se mantiene el primer píxel de la primera fila igual.
- El resto de píxeles se calculan como la diferencia al anterior, recorriendo las filas de la imagen en zig-zag. Por ejemplo:

Original image

9	8	7	7	7	5
7	7	7	7	4	4
6	6	6	9	9	9

Difference image

9	-1	-1	0	0	-2
0	0	0	3	0	-1
-1	0	0	3	0	0

a) ¿Cuál es el número de bits necesarios para almacenar cada píxel de img? ¿Y de Dimg?

b) Calcula la codificación de Huffman de la Imagen img y de la Imagen Diferencia Dimg y los promedios de bits en cada caso (se puede hacer "a mano" o con la ayuda de una tabla Excel o similar).
b) Calcula los radios de compresión de Huffman de img con respecto a img y de Huffman de Dimg con respecto a Dimg. ¿Ha sido más beneficioso usar Huffmann en la imagen diferencia? ¿Por qué? ¿Ocurre esto en general?

Ejercicio 4. Histograma [0,7 puntos]

Consulta la información sobre cálculo y visualización del histograma de una imagen:

https://docs.opencv.org/4.5.2/d1/db7/tutorial_py_histogram_begins.html

Considera una imagen fotográfica en escala de grises cuyos niveles de gris estén ampliamente repartidos en el intervalo [0,255] (y sea, por tanto, una imagen de mucho contraste).

a) Calcula y visualiza su histograma. Aplica una *transformación de intensidad lineal* que la convierta en una imagen con poco contraste **muy oscura,** sin que haya saturación, de manera que se pueda asegurar, a priori que el resultado se encuentra en [0,255] y manteniendo, en la medida de lo posible, "la forma" del histograma. Visualiza ahora su histograma para comprobar que el resultado es coherente.
b) ¿Qué transformación lineal se podría realizar ahora a la imagen resultado del apartado anterior para obtener una imagen de poco contraste **muy clara** que mantenga la misma cantidad de niveles de gris? Visualiza de nuevo el histograma resultante.

Nota: consideraremos niveles de gris "muy oscuros" aquellos por debajo de 85, por ejemplo, y niveles "muy claros" los que estén por encima de 170. Las soluciones que se piden deben aclarar u oscurecer, pero no "desnaturalizar" la imagen.

Sugerencias: Asegúrate que las operaciones que realizas no cambian el tipo de dato de las imágenes (que es uint8 por defecto). De ser así, busca la forma de devolverlos a ese tipo de dato. También hay que tener precaución con las operaciones que se realizan con arrays de datos uint8, puesto que el resultado puede estar dándose módulo 256.

Ejercicio 5. Transformaciones no lineales [0,6 puntos]

1. Razona qué transformación de intensidades **no lineal** (tipo log, exp, raíz, potencias) podría mejorar el contraste de la imagen obtenida en el apartado b) del ejercicio anterior.
2. Implementa dicha transformación de manera que el resultado se ajuste al intervalo [0,255], tratando de perder la menor información posible (de forma razonada).