E3-Segmentación

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 la manipulación de algoritmos básicos de segmentación.

Ejercicio 1. Umbralización [0,4 ptos.]

En OpenCV están implementados varios métodos de umbralización:

https://docs.opencv.org/4.1.0/d7/d4d/tutorial_py_thresholding.html

Toma una imagen fotográfica con iluminación desigual (una parte de la imagen está más iluminada que la otra) de un objeto sobre un fondo relativamente homogéneo. Pásala a escala de grises.

a) Aplica el método de Otsu, ¿cuál es el umbral calculado? Si se pudiese sumar una misma cantidad a todos los niveles de gris de la imagen, ¿cómo variaría el umbral de Otsu?
b) Explica en qué consisten las dos técnicas de umbral adaptativo, mean y Gaussian, que implementa OpenCV y aplícalas, explicando y ajustando los parámetros de forma razonada. ¿Es mejor el resultado? ¿Por qué?

Ejercicio 2. Canny [0,4 ptos.]

El método de Canny es el más comúnmente usado para detectar bordes en una imagen cualquiera. Información sobre la implementación del método de Canny en OpenCV la puedes encontrar aquí:

https://docs.opencv.org/4.1.0/da/d22/tutorial_py_canny.html

a) Explica los argumentos de la función implementada en OpenCV y relaciónalos con los pasos explicados en clase.
b) Toma tú mismo/a una imagen fotográfica (.jpg) de un sudoku o un pasatiempo que contenga una cuadrícula. Pásala a escala de grises y aplica el método de Canny para obtener los bordes de la imagen. Ajusta los parámetros (razonadamente) para detectar lo mejor posible las líneas de la cuadrícula y los números o letras del pasatiempo.

Ejercicio 3. Hough [0,6 ptos]

La transformada de Hough está implementada en OpenCV:

https://docs.opencv.org/4.1.0/d6/d10/tutorial_py_houghlines.html

a) Explica los parámetros de la función y la salida del método y relaciónalos con los pasos explicados en clase.
b) Ajusta esos parámetros para detectar lo mejor posible todas las líneas de la cuadrícula de la imagen procesada del ejercicio anterior. Visualízalas sobre la imagen original.
c) Considera ahora una imagen en la que el sudoku aparece bien encuadrado, es decir, la cuadrícula no está inclinada, pero que aparezcan además otros objetos con bordes rectos con otra inclinación, por ejemplo, por el borde de una mesa o un lápiz inclinado al lado del sudoku. Modifica, razonadamente, la implementación anterior para que se detecten sólo las líneas de la cuadrícula en la imagen (y no otras con otra inclinación).

Ejercicio 4. K-medias [0,6 ptos]

El método de K-medias puede usarse como método de cuantificación del color mediante el agrupamiento de píxeles de color semejante. Cada píxel puede pensarse como un punto en el espacio tridimensional dado por el color.

El método de K-medias se encuentra implementado en OpenCV:

Understanding K-Means Clustering:
https://docs.opencv.org/4.1.0/de/d4d/tutorial_py_kmeans_understanding.html

K-Means Clustering in OpenCV:
https://docs.opencv.org/4.1.0/d1/d5c/tutorial_py_kmeans_opencv.html

a) Explica los parámetros que usa el método, aclarando en qué consisten los criterios de parada, y su salida.
b) Toma una imagen fotográfica en la que aparezcan una carretera, paisaje con vegetación y cielo. Realiza una cuantificación del color mediante el algoritmo de k-medias que permita segmentar las tres zonas principales de la imagen, usando el menor número de grupos posibles. Razona los parámetros seleccionados y explica el resultado obtenido.

c) Prueba a realizar el apartado anterior en otro modelo de color más apropiado para segmentar el color y compara razonadamente los resultados.

Ejercicio 5. Morfología [0,5 ptos.]

A partir de la imagen (E3-5.jpg) de un documento de texto, se pide:

a) realizar un preprocesado de la imagen mediante operaciones de suavizado y umbralización para convertir la imagen en binaria con la menor cantidad de ruido posible;
b) realizar las operaciones morfológicas binarias más convenientes para obtener un bloque de píxeles conectados por cada línea de texto (como si se tachara cada línea con un rotulador muy grueso que cubre todas las letras), sin que se toquen los bloques de distintas líneas y sin que sean más anchas que antes. No se debe girar la imagen. Se debe **razonar primero** qué tipo de elemento estructural y qué operaciones pueden ser convenientes para esta tarea y después experimentar con distintos tamaños de elementos estructurales y con esas operaciones morfológicas para escoger la mejor combinación.

Información sobre operaciones morfológicas en OpenCV:

https://docs.opencv.org/4.1.0/d9/d61/tutorial_py_morphological_ops.html