Procesado de imagen y visión por computador

Segmentación de imágenes

0. Introducción

La segmentación en las imágenes pretende subdividir las mismas en las regiones u objetos constituyentes de acuerdo a algún criterio de similitud. Como se ha visto en la presentación de esta sesión, la mayor parte de las técnicas tradicionales de segmentación se pueden incluir en una de las tres siguientes aproximaciones:

- **Detección de bordes**. Mediante esta técnica se definen los bordes de los objetos o formas dentro de la imagen para el posterior análisis. Entre las técnicas más destacadas podemos nombrar la de Canny (prácticamente un estándar en la detección de bordes) u otras como las de Sobel, que utiliza máscaras de convolución.
- **Umbralizado**. Sobre imágenes en escala de grises o en la componente de iluminación de una imagen en color se puede fijar un umbral que permita extraer los objetos buscados. En imágenes en escala de gris la umbralización se emplea cuando hay una clara diferencia entre los objetos y el fondo de la escena. El umbral que distingue los objetos del fondo de la escena puede tener un valor fijo o adaptarse a los cambios de iluminación mediante técnicas como la de Otsu, que determina el umbral óptimo en cada imagen maximizando la varianza entre clases de manera automática.
- **Segmentación en color**. El color es una fuente de información importante dentro de las imágenes y muchas técnicas de segmentación lo utilizan para extraer objetos de un determinado color de interés. Las técnicas para la segmentación en color pueden ir desde el más sencillo umbralizado en un determinado espacio de color a técnicas más avanzadas basadas en clustering, redes neuronales, máquinas de vectores soporte y las muy vigentes redes neuronales convolucionales.

Esta sesión pretende manejar algunos de los algoritmos más comunes en segmentación de imágenes.

1. Detección de bordes

En este apartado vamos a probar el método de detección de bordes de Canny[1], el cual presenta tres parámetros variables de ajuste. Para ello dispone del programa *apartado1.py* que realiza un ejemplo de aplicación del mismo. Pruebe el programa y analice cómo funciona sobre las imágenes originales Politecnica1.png y Politecnica2.png.

Añada dos barras de desplazamiento que permitan ajustar los dos umbrales T_1 y T_2 del algoritmo de Canny para obtener los mejores resultados en la detección de puertas. Determine unos valores razonables para los dos umbrales en cada caso. Opcionalmente, puede intentar representar en la ventana de visualización simultáneamente la imagen del mapa de bordes y la imagen original con los bordes detectados. En este último caso, deberá utilizar la función *cv2.hconcat* considerando que el mapa de bordes es una imagen en escala de gris (una dimensión) y la imagen original es RGB (tres dimensiones).

1 https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

Procesado de imagen y visión por computador

Segmentación de imágenes

2. Umbralización

En este caso queremos probar la umbralización directa de imágenes en escala de grises y comprobar los problemas que suponen los cambios de iluminación si se trabaja con un umbral de valor fijo. En el fichero *apartado2.py* hay un ejemplo de umbralización sobre una imagen que puede tomarse como base (use la imagen Monedas.*jpg*).

A continuación se trabajará con un vídeo en el que se tiene un escenario de iluminación variable. En este caso, se pide:

1. **Realizar** un programa, a partir del script *apartado2.py, que* permita umbralizar el vídeo *movil.avi* mediante umbralización con umbral fijo.
2. **Modificar** el programa anterior para que el umbral se adapte a los cambios de iluminación. Para ello se utilizará la técnica de Otsu[2], que ya viene implementada en la función *threshold* y se imprimirá por consola de comandos el umbral fijado por la función *threshold* para cada fotograma. Ejecutar el programa obtenido sobre el vídeo *movil.avi*. Opcionalmente, puede mostrar en la ventana de resultados simultáneamente cada frame del vídeo con su correspondiente resultado de umbralización mediante la función *cv2.hconcat*.

3. Segmentación en color

Finalmente utilizaremos la información de color para segmentar. En el fichero *apartado3.py* tiene un ejemplo usando el espacio de color RGB. Se pide ejecutarlo sobre las imágenes *pokemon.png* y *balls.png*. Probar su funcionamiento hasta entender cómo funciona.

3-a. Segmentación en espacio HSV (Opcional)

Opcionalmente, puede añadir el código necesario para cambiar el espacio de color de RGB a HSV[3] y realizar la segmentación en ese espacio a partir de la función *cv2.inRange*. Tenga en cuenta que OpenCV usa los siguientes rangos:

H: 0 – 180

S: 0 – 255

V: 0 – 255

Considere que debido al rango de variación diferente para cada componente de color del espacio HSI, el umbral de ajuste será diferente para cada una de ellas. Para ello, deberá introducir tres valores umbrales como argumentos de entrada del script (*umbralH*, *umbralS* y *umbralV*). Pruebe la modificación introducida y observe si los resultados mejoran sobre el espacio RGB o no.

2 http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0
3 http://docs.opencv.org/master/df/d9d/tutorial_py_colorspaces.html#gsc.tab=0