# Percepción - Práctica 1

Percepción y Control para Sistemas Empotrados

Curso 2020-21

---

Izar Castorina – Grupo 1

Para esta práctica, una toma de contacto con los algoritmos y los métodos de tratamiento de imágenes y detección de características, se nos ha pedido realizar cuatro tareas: una primera presa de contacto con las herramientas, la mejora de algunas imágenes proporcionadas y la eliminación del ruido contenido en ellas, la binarización de otras imágenes y la extracción de contornos.

El programa realizado en Python dispone de un pequeño menú a través de consola que permite seleccionar cuál de las cuatro partes ejecutar, además de dar la opción de ejecutarlas todas de forma secuencial. Todos los plots resultantes de las varias partes han sido almacenados en la carpeta *Resultados/Plots* adjunta.

# 1 Presa de contacto

Para tratar las imágenes, hemos empleado la librería `scikit-image`. Las primeras operaciones que hemos realizado han sido la abertura, el redimensionamiento y el guardado de imágenes a través de las funciones `imread`, `transform.resize`, `imsave`, y `color.rgb2gray`. Las imágenes guardadas se encuentran en la carpeta *Resultados* adjunta a la entrega de este documento.

También, hemos empleado la librería `pyplot`, incluida en `scikit-image`, para visualizar en una misma gráfica, la imagen original y sus versiones reescaladas. Esto ha sido posible creando cuatro *sub-plots* al interior de una figura, y empleando la función `imshow` para asignar una imagen a cada uno de ellos. El resultado final viene mostrado a pantalla.

# 2 Mejora de imágenes y eliminación del ruido

En este apartado, hemos afrontado el problema del ruido presente en algunas de las imágenes de ejemplo, además de ver cómo visualizar de forma gráfica y modificar el histograma de una imagen.

Para visualizar el histograma, una vez cargada la imagen, hemos empleado la función `exposure.histogram` para guardar los datos relativos a las intensidades de gris presentes en la imagen, visualizándolo gracias a la función `pyplot.bar`. Sucesivamente, hemos efectuado la operación de *stretching* del histograma, efectivamente aumentando el contraste de la imagen y haciendo sí que no hubiese intensidades "no usadas" en los extremos del histograma. Otra vez, hemos vuelto a visualizarlo empleando la misma función.

Una vez convertida la imagen a bytes sin signo con `img_as_ubyte`, hemos podido emplear la función `exposure.equalize_hist` para ecualizar el histograma y

mejorar de una forma diferente el contraste de la imagen. La operación ha sido repetida empleando el método CLAHE a través de la función `exposure.equalize_adapthist`. Empleando la función `exposure.cumulative_distribution` hemos visualizado la distribución de intensidades de las dos imágenes ecualizadas, pero esta vez de forma cumulativa. Sucesivamente, los cuatro gráficos elaborados han sido condensados en una sola figura de manera análoga a lo que se hizo al final del primer apartado de la práctica. Se han añadido títulos a cada sub-plot para que se entienda mejor cuál es el resultado de cada operación. Lo mismo se ha repetido para los histogramas y las distribuciones cumulativas generadas.

Hecho esto, hemos pasado al tratamiento del ruido en dos imágenes de prueba, una con ruido Gaussiano y otra con ruido de tipo Sal/Pimienta.

Para las dos imágenes se han aplicado los mismos filtros: de media, Gaussiano, y de mediana. Los resultados para cada imagen han sido representados en sub-plots, de forma idéntica a la descrita hasta ahora.

# 3 Binarización

El proceso de binarización permite elaborar la información contenida en una imagen y sacar otra imagen en la cual solo hay dos valores para cada pixel: blanco o negro. Empleando técnicas y coeficientes diferentes, es posible separar e identificar objetos y formas dentro de la imagen.

Se nos ha pedido binarizar cuatro imágenes, de `practica1_4` a `practica1_7`, empleando diferentes factores de binarización (0.2, 0.5 y 0.7) y finalmente a través del método de Otsu.

Dado que las operaciones a realizar eran las mismas para cada imagen, se ha hecho uso extenso de bucles, almacenando las imágenes fuente en una lista, los *thresholds* en otra, y los resultados en una lista anidada. Para la aplicación del método de Otsu, se han creado dos listas vacías en las cuales almacenar los valores y las imágenes resultantes de la elaboración. A cada imagen ha sido aplicado cada valor de *threshold* y guardado, y en un bucle sucesivo se ha aplicado el método de Otsu también a todas las imágenes.

Finalmente, para cada imagen ha sido creada una gráfica, con 4 sub-plots, en el cual se enseña la imagen original y el resultado de las binarizaciones según los valores dados. Finalmente, se visualiza una gráfica con las cuatro imágenes binarizadas según el método de Otsu.

# 4 Extracción de contornos

Para extraer un mapa de bordes de las mismas cuatro imágenes del apartado anterior, se han aplicado los filtros de Sobel, Roberts, Prewitt y Canny. Cada imagen ha sido procesada con los cuatro y almacenada. Finalmente, se ha creado una gráfica con cuatro sub-plots, en la cual se enseña el resultado de palicar cada filtro a cada una de las imágenes. Para el filtro de Canny ha sido necesario especificar en la función `imshow` que se representara la imagen como compuesta por enteros sin signo (`astype(numpy.uint8)`), para ser consistentes con los resultados de los otros filtros aplicados.