Procesado de imagen y visión por computador

Curso 2018/2019

OPERACIONES Y MODIFICACIONES DE IMÁGENES

1. Suma de imágenes

En el código llamada *Apartado1.py* se muestran las diferencias entre la suma usando el módulo Numpy y la suma usando OpenCV. Compruebe que los resultados difieren y piense a qué es debido.

2. Corrección de la iluminación

Si el patrón de iluminación de una escena es conocido, o se puede estimar, se puede utilizar éste para uniformizar la iluminación de la imagen, mediante la multiplicación de la imagen con el patrón.

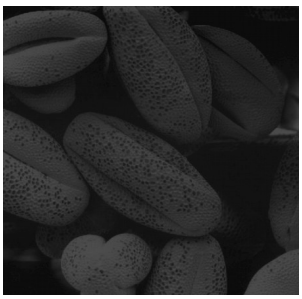
Vamos a partir de las imágenes *Radiografia.jpeg* y *EscalaGris.png* y multiplicándolas intentaremos conseguir una iluminación más uniforme en la imagen *Radiografia.jpeg*. Por ello, y como la parte inferior de la imagen es más clara que la superior, la parte inferior deberá quedar multiplicada por un valor más pequeño, para que el color del fondo quede con un valor de gris más o menos constante. Es decir, el degradado debe partir de color blanco en la parte superior, hasta un color gris en la parte inferior. Esto lo conseguimos con la imagen *EscalaGris.png*.

En el programa *Apartado2.py* tiene todo lo necesario para conseguir el efecto deseado y sólo falta rellenar los parámetros de entrada de la función *multiply()*[1].

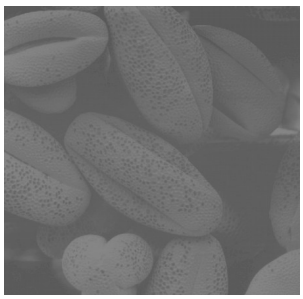
3. Manipulación del histograma

Como se ha visto en la presentación, el histograma y su manipulación pueden ser una herramienta para la mejora o modificación de la imagen. En este apartado vamos a observar y analizar el histograma de varias imágenes y el efecto que produce su normalización y ecualización.

Las imágenes que vamos a utilizar son las siguientes (Digital Image Processing. Gonzalez&Woods):


Oscura


Clara


Bajo contraste


Alto contraste

A cada una de ellas se le aplicara una normalización y una ecualización del histograma. Obtenga a continuación los nuevos histogramas. Compare los histogramas y las imágenes modificadas con los originales y comente qué observa.

En el programa *Apartado3.py* tiene un ejemplo de normalización del histograma. Puede tomarlo como base y probarlo en todas las imágenes. Para probar la ecualización use la función *equalizeHist()*[2].

1 http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#multiply
2 http://docs.opencv.org/modules/imgproc/doc/histograms.html?highlight=equalizehist#equalizehist

4. Reducción de ruido en imágenes

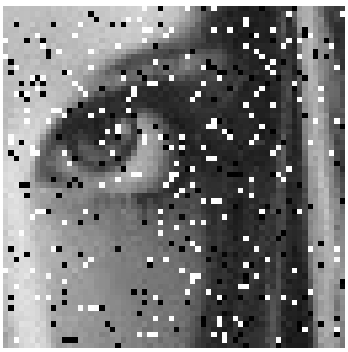
En este apartado vamos a probar distintas técnicas para reducir el ruido en imágenes, así como métodos para añadir ruido a éstas. Probaremos únicamente con imágenes en escala de grises, debido a su mayor sencillez, aunque lo visto aquí se puede aplicar a imágenes en color con ligeras modificaciones.

4.1. Ruido impulsivo

En el ruido impulsivo, cada píxel de la imagen se sustituye, con una probabilidad p, por un valor aleatorio, independiente del valor del píxel en la imagen original, mientras que el resto $(1-p)$ mantienen su valor original. El ruido de *sal y pimienta* es un tipo particular, donde los píxeles que cambian su valor, solo toman los valores 0 (negro) o 255 (blanco). En la siguiente figura se muestra un ejemplo, para una probabilidad p igual a 0,1:

Imagen original

Imagen con ruido impulsivo



Detalle

Al ser p igual a 0,1, significa que el 10 % de los píxeles están modificados, mientras que el 90 % restante conservan su valor original. En este ejemplo concreto, además, la mitad de los píxeles modificados quedan con valor 0, y la otra mitad con valor 255, aunque este parámetro también puede ser configurado.

En el programa *Apartado4.py* tiene un ejemplo de como se aplica el filtro de mediana a una imagen con ruido impulsivo añadido. Concretamente añadimos ruido impulsivo sobre la imagen *Lenna.png,* con p=0,1 y realizamos la reducción de ruido mediante un filtro de mediana de 3x3. Finalmente obtenemos una medida de la calidad obtenida mediante lo que se conoce como PSNR[3] y mostramos las dos imágenes juntas para comparar los resultados.

Pruebe el programa añadiendo más o menos ruido y modificando el filtro de mediana cambiando el tamaño 3x3 a 5x5 o mayor.

4.2. Ruido gaussiano

En el programa anterior sólo añadíamos ruido impulsivo pero también se podría añadir ruido gaussiano con la función *ruido()*. En este apartado se pide que modifique el programa anterior para añadir ruido gaussiano en distintas cantidades y pruebe a reducirlo con el filtro de mediana pero también con la función GaussianBlur()[4].

Estos son algunos ejemplos de filtrado de imágenes pero puede encontrar más en: http://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html#gsc.tab=0.

3 https://es.wikipedia.org/wiki/PSNR

4 http://docs.opencv.org/modules/imgproc/doc/filtering.html?highlight=gaussianblur#cv2.GaussianBlur

5. Redimensionado.

En este apartado vamos a utilizar la función que nos permite redimensionar una imagen para hacerla más o menos grande y adaptarla a nuestras necesidades.

En el programa *Apartado5.py* se pueden encontrar distintos ejemplos de redimensionado. Ejecute el programa y compruebe su efecto.

Posteriormente, y usando dicho programa como base, realice el siguiente ejercicio:

1. Cargue una imagen y redimensione su altura y anchura a la mitad.
2. Vuelva a redimensionarla (con el mismo tipo de interpolación) para recuperar el tamaño original.
3. Obtenga el PSNR entre la imagen original y la que ha hecho decrecer y crecer previamente. Comprobará que las dos imágenes no van a ser iguales porque en el redimensionado se pierde información.
4. Repita este procedimiento cambiando en cada caso el tipo de interpolación y compruebe que los resultados son ligeramente diferentes. Hágalo para los tipos CUBIC, AREA, LINEAR y LANCZOS4.