1. Suma de imágenes

Una de las operaciones más básicas con imágenes consiste en sumar dos imágenes de tamaño idéntico y generar una imagen resultante del mismo tamaño de ambas. El valor de cada píxel de la imagen resultante es la suma de los valores de los píxeles correspondientes a las dos imágenes de entrada. Si el formato de imagen que se está utilizando soporta valores de píxel enteros de 8 bits, es bastante probable que el resultado de la suma sea mayor que el máximo permitido por el valor del píxel. El efecto de los píxeles desbordados dependerá de la implementación.

El código del archivo *apartado1.py* muestra las diferencias entre dos formas de realizar la suma de imágenes: usando el módulo *Numpy* y usando *OpenCV*. Compruebe que los resultados son diferentes en ambos casos y justifique la manera de trabajar en cada uno de los dos casos.

Incluya la justificación del modo de trabajo en cada caso. Como indicio, se aporta el siguiente código que puede teclear, donde se realiza la suma de dos valores escalares mediante el módulo *Numpy* y usando *OpenCV*:

```
>>> import cv2
>>> import numpy as np
>>> x=np.uint8([250])
>>> y= np.uint8([10])
>>> print(cv2.add(x,y)) # 250+10 = 260 => 255
>>> print( x+y ) # 250+10 = 260 % 256 = 4
```

2. Suma ponderada de imágenes (Opcional)

La técnica de fusión de dos imágenes, muy utilizada en transición de diapositivas y en el cine, consiste en una suma de dos imágenes con la peculiaridad de que a cada una de ellas le asignamos un peso diferente, con lo que el resultado crea un efecto de transparencia. Matemáticamente, el valor de un píxel (x,y) en la imagen resultante $g(x,y)$ puede expresarse a partir de dos imágenes f_1 y f_2 como:

$$g(x,y)=\alpha f_1(x,y)+(1-\alpha)f_2(x,y)$$

Variando el valor del parámetro α de 0 a 1 podemos controlar el nivel de fusión. A partir del script *apartado2.py* pruebe a realizar la fusión de dos imágenes en color de igual tamaño varíe el valor de este parámetro. Para ello se utiliza la función *cv2.addWeighted*[1], que aplica la siguiente ecuación para obtener la imagen resultante:

$$g(x,y)=\alpha f_1(x,y)+\beta f_2(x,y)+\gamma$$

donde γ es un escalar que se añade en la suma y que en este caso es fijado a 0. Pruebe a obtener diferentes resultados variando el valor del parámetro α de 0 a 1.

Incluya las imágenes resultantes con varios valores de α y describa las conclusiones.

1 Enlace: https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html

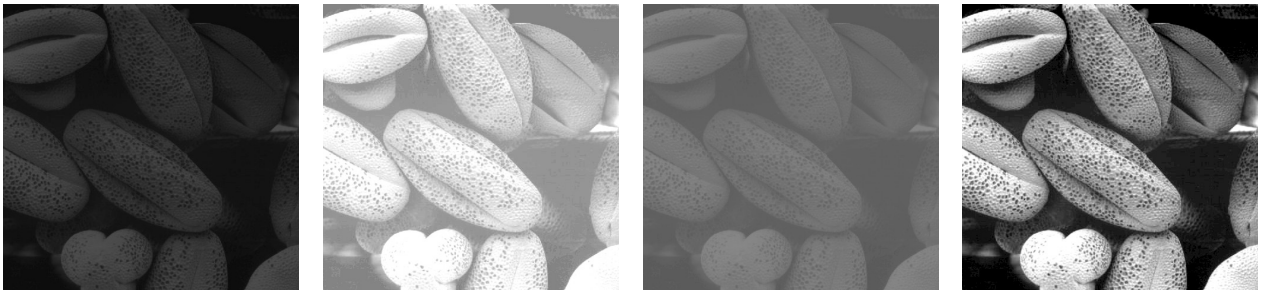
3. Corrección de la iluminación (Opcional)

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 *apartado3.py* tiene todo lo necesario para conseguir el efecto deseado y sólo falta rellenar los parámetros de entrada de la función *multiply()*[2].
Incluya la imagen resultante de este apartado.

4. Manipulación del histograma

Como se ha visto en la presentación teórica, el histograma y su manipulación pueden ser una excelente herramienta para la mejora y/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 aplicará 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 a*partado4.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()*[3].

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

Incluya las imágenes obtenidas con la normalización y la ecualización, así como las conclusiones oportunas.

5. Reducción de ruido en imágenes

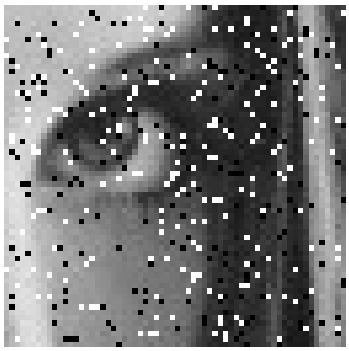
En este apartado vamos a probar distintas técnicas para reducir el ruido en imágenes, así como métodos para añadir ruido a éstas. Probaremos únicamente con imágenes en escala de grises, debido a su mayor sencillez, aunque lo visto aquí se puede aplicar a imágenes en color con ligeras modificaciones.

5.1. Ruido impulsivo

En el ruido impulsivo, cada píxel de la imagen se sustituye, con una probabilidad p, por un valor aleatorio, independientemente 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 uno de los dos siguientes 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 *apartado5.py* tiene un ejemplo de cómo 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[4] 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. ¿Cuáles son las conclusiones a nivel visual y de PSNR?
Prueba a introducir un tamaño de filtro par (por ejemplo, 4x4) ¿Cuál es la causa del error?
Incluya los PSNR obtenidos para cada tamaño del filtro usado y los comentarios.

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

5.2. Ruido gaussiano

En el apartado anterior hemos contaminado la imagen con ruido impulsivo, pero también se puede añadir ruido gaussiano. En este apartado se pide que modifique el script *apartado5.py* para generar ruido gaussiano en distintas cantidades. Pruebe a reducirlo con el filtro de mediana y también con la función GaussianBlur()[5]. ¿Cuáles son las conclusiones?
Estos son algunos ejemplos de filtrado espacial de imágenes, pero se pueden encontrar algunos más en la URL http://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html#gsc.tab=0.

6. Redimensionado (Opcional)

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

En el programa a*partado6.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 que en el punto 1) para recuperar el tamaño original.
3. Obtenga el PSNR entre la imagen original y la que ha hecho decrecer y crecer. 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.

Incluya los resultados de PSNR para cada tipo de interpolación y extraíga las conclusiones oportunas.

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