EC5422 – Procesamiento Digital de Imágenes Enero–Marzo/2018 (Martes1-3) Laboratorio 2.

Objetivo: Implementar en Python, usando las librerías OpenCv:

1. Las funciones globales de modificación de histograma, de ecualización del histograma,
2. Y las funciones locales de filtros espaciales .

Todo el código desarrollado debe subirse a Google Drive en la carpeta "PDI/Laboratorios/Laboratorio 2". El código se verificará y debe ser ejecutable desde la plataforma ANACONDA.

PARTE 1.

1. Entrar en la dirección http://www.imageprocessingplace.com/root_files_V3/image_databases.htm y seleccione una base de datos (puede elegir cualquier otra imagen de su gusto). Baje una imagen de prueba a su disco.
2. Haga un código en Python que:
    a. Lea la imagen y visualice el histograma de cada color (o el único histograma si es una imagen en niveles de gris).
    b. Aplique una función de "stretching" del histograma para expandirlo al máximo rango dinámico. Tenga cuidado con pixels aislados con valor 0 ó 255 que podrían no ser realmente el mínimo y máximo del histograma. (Para hacer el stretch corte un porcentaje de las "colas" del histograma; explore la función cvThreshHist).
    c. Aplique ahora una ecualización del histograma: explore la función cvEqualizeHist.

PARTE 2.

1. Para la misma imagen de la Parte 1, aplique, programando en Python y usando OpenCv:
    a. Una máscara de suavizado: revise (especialmente las 'X' y modifique un código similar a éste -tomado de la web-). Revise lo que hace la función np.ones y la función cv2.filter2D:
        i. import cv2
        ii. import numpy as np
        iii. from matplotlib import pyplot as plt
        iv. img = cv2.imread('XXX.XXX')

v. kernel = np.ones((X,X),np.float32)/X
vi. dst = cv2.filter2D(img,-1,kernel)
vii. plt.subplot(121),plt.imshow(img),plt.title('Original')
viii. plt.xticks([]), plt.yticks([])
ix. plt.subplot(122),plt.imshow(dst),plt.title('Promedio')
x. plt.xticks([]), plt.yticks([])
xi. plt.show()

b. Cambie el tamaño del kernel y ejecute de nuevo el código anterior.
c. Aplique ahora una máscara de perfilado.
d. Estudie y aplique cv2.GaussianBlur().
e. Revise e implemente este código sobre su imagen de trabajo, analice qué hace:
    i. import cv2
    ii. import numpy as np
    iii. from matplotlib import pyplot as plt
    iv. img = cv2.imread('XXX.XXX',0)
    v. laplacian = cv2.Laplacian(img,cv2.CV_64F)
    vi. sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=X)
    vii. sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=X)
    viii. plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
    ix. plt.title('Original'), plt.xticks([]), plt.yticks([])
    x. plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
    xi. plt.title('Laplaciano'), plt.xticks([]), plt.yticks([])
    xii. plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')
    xiii. plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
    xiv. plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')
    xv. plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
    xvi. plt.show()
f. Por último, aplique a su imagen el Laplaciano del Gaussiano con una máscara de tamaño 5x5. Analice qué hace openCv con los bordes. (Para aplicar una máscara gaussiana 5x5 recuerde que el kernel gaussiano es separable; y recuerde la asociatividad de la aplicación de los kernel).