OpenCV exercises

1. Images –read, write and display; ROIs

a) Read the name of a file containing an image in 'jpg' format and show it in a window, whose name is the name of the file. Test whether the image was successfully read. Display the height and width of the image, on the console.

b) Read a color image in 'jpg' format and save it in 'bmp' format.

c) Read an image from a file, allow the user to select a region of interest (ROI) in the image, by clicking on two points that identify two opposite corners of the selected ROI, and save the ROI into another file.

2. Images – representation, grayscale and color, color spaces

a) Create a grayscale image, having 100(lines)x200(columns) pixels with constant intensity, 100; draw the two diagonals of the image with intensity 255. Display the image.

b) Create a color image, having 50(lines)x200(columns) pixels with constant intensity, 100; draw the two diagonals of the image, one in red color, the other in blue color. Display the image.

c) Read a color image, display it in one window, convert it to grayscale, display the grayscale image in another window and save the grayscale image to a different file.

d) Read an image (color or grayscale) and add "salt and pepper" noise to it. The number of noisy points must be 10% of the total number of image points. Suggestion: start by determining the number of image channels.

e) Read a color image (in RGB format), split the 3 channels and show each channel in a separate window. Add a constant value to one of the channels, merge the channels into a new color image and show the resulting image.

f) Read a color image (in RGB format), convert it to HSV, split the 3 HSV channels and show each channel in a separate window. Add a constant value to saturation channel, merge the channels into a new color image and show the resulting image.

3. Video – acquisition and simple processing

a) Display a video acquired from the webcam (in color) in one window and acquire and save a frame when the user presses the keyboard. Show the acquired frame in another window.

b) Display the video acquired from the webcam (in color) in one window and the result of the conversion of each frame to grayscale in another window.

c) Modify the program developed in b) so that the resulting frames are in binary format (intensity of each pixel is 0 or 255); use a threshold value of 128.

d) Implement a simple tracking algorithm for colored objects, using the following steps: 1) take each frame of the video; 2) convert from BGR to HSV color-space; 3) threshold the HSV image for a range of color values (creating a binary image); 4) extract the objects of the selected range (use a bitwise AND operation, using as operands the original and the binary image) .
(see https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html#converting-colorspaces)

4. Image enhancement – filtering

Take a noisy image (see problem **2.d**) and filter it (try different filter sizes), using:

(for **a-d** see https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html)

a) a mean filter;

b) a Gaussian filter;

c) a median filter;

d) a bilateral filter.

e) a filter defined by you, adapting the following code (note: remove the (1/14) factor in the assignment of kernel_3x3 and explain what happens):

```
import os
import numpy as np
from matplotlib import pyplot as plt
from cv2 import cv2
from scipy import ndimage
```

```
# Read image
datadir = 'C:/.../...' # to complete
img = cv2.imread(os.path.join(datadir,'image1.jpg')) # to be changed

# Convert to grayscale if needed
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Smooth using OpenCV GaussianBlur()
gaussianBlurred = cv2.GaussianBlur(img, (3,3), 0)

# Smooth using convolution operation coded below
kernel_3x3 = (1/14)*np.array([[1, 2, 1],[1, 4, 1],[1, 2, 1]])
print(kernel_3x3)
myConvolutionResult = ndimage.convolve(img, kernel_3x3)

# Show results
cv2.imshow("Original", img)
cv2.imshow("OpenCV Gaussian Blur", gaussianBlurred)
cv2.imshow("My 3x3 convolution w/Gaussian mask", myConvolutionResult)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. Image enhancement – histogram equalization

(see https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html)

a) Take a low contrast grayscale image and plot its histogram.

b) Enhance the image constrast using:
b1) simple histogram equalization, or
b2) CLAHE,
and show the resulting enhanced images and their histograms.

c) Repeat the previous operations on a color image.