MATLAB/PYTHON EXERCISE ON EDGE DETECTION

TUTOR: MIRELA POPA

TEACHING ASSISTANTS: ESAM GHALEB

DATE: 15/04/2021

In this exercise, you will familiarize yourself with various edge detection techniques.

For the following, create *your own* functions (not using functions from the Image Proc. Toolbox in Matlab and functions from cv2 library in Python)

All output images of this exercise should be normalized from 0 to 1 before displaying. Download Lena.jpg (the famous 'Lena') and use it in this exercise, after converting it into grayscale.

a) Calculate Lena's gradient magnitude, binarize the result using different thresholds (0.05, 0.25, 0.5, 0.75, 1), and show the results in 5 sub-figures (subplot grid).

   Note: Write a function to implement 2D convolution. The function will take in a filter, and the image, and return the convolved image. In case if your filter falls outside the region of an image, use the zero-padding concept.

b) Implement a function that allows (**Optional**): to choose between Prewitt and Sobel operators, and allows the user to specify binarization thresholds, as well as orientation ('horizontal', 'vertical' and both).

c) Implement a function that returns edges at the zero-crossings of the image, using the 3x3 Laplacian operator we learnt in class, as well as the Laplacian of Gaussian method trying 5x5 and a 17x17 masks (Log5 and Log17.mat files uploaded on Canvas). For detecting the zero-crossings, create a 3x3 mask and apply it on every pixel on the convolved image.

   Note: Detect zero-crossings by comparing the sign of each element's neighbourhood. A positive element is considered to be zero-crossing if it contains at least one neighbour with negative value.

   Note for Python: Log5.mat and Log17.mat might be read into a NumPy array using scipy.io.loadmat function. Can you further improve the result?