MATLAB/PYTHON INTRODUCTORY EXERCISE ON IMAGE BASICS

TUTOR: MIRELA POPA

TEACHING ASSISTANT: ESAM GHALEB

DATE: 13/04/2021

In this exercise you will familiarize yourself with the essentials of Matlab/Python programming for image processing and then implement several basic routines. Download the file **exercise1.rar** from Canvas/Modules/week 2 and unzip it in a separate directory. This file includes two images (***lab1a.png*** and ***lab1b.png***) that you will be working on for this exercise.

a) Matlab:

Use Matlab documentation to learn about the following commands which we will use in this exercise: `imread`, `imshow`, `figure`, `subplot`, and `imresize`

To obtain information about a particular command you can type **help** followed by the command name in the Matlab command prompt.

Python:

To solve the exercises from this lesson we are suggesting the following libraries:

- OpenCV (python-opencv, cv2) – for basic image manipulations, such as reading an image into array, converting to a certain color map, etc. (documentation).
- NumPy – for working with arrays. An image can be represented as arrays where each element is linked with a pixel value of the image (documentation).
- Matplotlib – for plotting graphs and showing images we usually use pyplot module of the package (documentation).

If you do not have Python installed on your PC, it might be useful to install Anaconda environment. The package involves various libraries, for example, NumPy and Matplotlib libraries come with the package. The package also includes Jupyter Notebook environment which might be useful to write code and show results inline (more information here).

To install OpenCV, we suggest you use *pip* package manager (also included in Anaconda) in the command line as shown below:

```
>> pip install opencv-python
```

After installing Python environment and libraries, you can import libraries as follows:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

b) Create a vector y with 100 elements holding the values of $y = x^2$, where x is a real value between 0 and 1. Make a 100x100 matrix A with each row being equal to y. Display the result with:

   a. **Matlab:**
      `imshow(A)` and `imshow(uint8(A))`.
   b. **Python:**
      `plt.imshow(A)` and `plt.imshow(A.astype(int))`.

   Comment on the results.

c) Read the image "lab1a.png" and display it in a window with the command `imshow` (`plt.imshow` for Python).
   Note for Python: You can Use cv2 library to read image from file into numpy array. By default, cv2 function reads image as BGR (Blue-Green-Red), not RGB (Red-Green-Blue).

d) Convert the color image to gray values and display. What is the size of the resulting matrix?
   Note for Python: When using `plt.imshow()` function for a grayscale/binary image, specify `cmap='gray'` parameter to show it in the grayscale color map.

e) Convert the color image to hsv and display each channel in a different subplot, along with the original image.

f) Convert the color image to different binary images, using various thresholds (at least 5). Display each obtained binary image in a different subplot, along with the original image. Give an appropriate title to each of the results.

g) Choose a bounding box of the original image (start with a small part of the car – write a crop function) and resize it using the different methods we discussed. For this task, use:
   a. **Matlab:** `imresize`. Try different interpolation methods and notice the results, by using imresize(I,scale,method), for example for Nearest Neighbor use 'nearest', but try also 'bilinear', 'bicubic', for different types of kernels.
   b. **Python:** `cv2.resize`. Use different InterpolationFlags (e.g. cv2.resize (image, None, fx=.75, fy=.75, interpolation = cv2.INTER_LINEAR)

Create a subplot (appropriately titled) for each method and scale used to resize it. Display the above results also with lab1b.png.