Assignment 1

The purpose of this assignment is to give you some insight into image interpolation, point operations and histogram methods, as well as to give you some practical experience in processing images using Python with OpenCV and NumPy.

Image Interpolation (Points = 2*5 = 10)

Using the "lena" image as input, first rescale the image to 1/4 times of its size, i.e., 1/2x in each dimension and then rescale this downscaled image to 4 times of its size, (i.e., back to its original dimension) using the:

1. Nearest neighbor interpolation implementation from scratch
2. Nearest neighbor interpolation using OpenCV built-in function
3. Bilinear interpolation implementation from scratch
4. Bilinear interpolation using OpenCV built-in function
5. Bicubic interpolation using OpenCV built-in function

Store the output images as "lena_nearest_scratch", "lena_nearest_cv", "lena_bilinear_scratch", "lena_bilinear_cv" and "lena_bicubic_cv", respectively.

Point Operations (Points = 2*3 = 6)

Using the image "cameraman" as input:

1. Find the negative of the image and store the output as "cameraman_negative".
2. Apply power-law transformation on the image (experiment with different gamma values) and store the (best) output as "cameraman_power".
3. Apply bit-plane slicing on the image and store the outputs as "cameraman_b1", "cameraman_b2", …, "cameraman_b8".

You must complete the above tasks by implementing these functions from scratch, i.e., do not use any built-in OpenCV functions.

Histogram Processing (Points = 4+5 = 9)

1. Apply histogram equalization on the "einstein" image and store the output as "einstein _equalized".
2. Apply histogram specification on "chest_x-ray1" image so that it matches the histogram of "chest_x-ray2". Store the output as "chest_x-ray3".

You are not allowed to use any built-in OpenCV functions for the above tasks but you may use NumPy functions for computing histogram and cdf etc.

1. Deliverables:

Your deliverable should include the following material:

1. Complete source code. This should be complete, so that it could be easily rebuilt and executed on a mirror system. The code may contain Python file(s) (.py) or Jupyter Notebook(s) (.ipynb).
2. Resulting output images. Make sure that you include all output images. Also, make sure to follow the given names for the output images. Non-compliance will result in marks deduction.
3. Documentation, as indicated in "Report Content" below, in pdf format.

Convert everything into a single zip file and submit to MLS.

2. Report Content:

Write a short report that:

- Provides a brief theoretical overview of each process you implemented.
- Discusses the overall quality of your results. This includes a visual comparison of the quality of the interpolation methods side-by-side, maybe by plotting difference images or just discussing the noticeable differences.
- Implementation issues that you encountered, if any.

The report size should not be more than 2 pages of text (excluding images).

3. Marking Scheme

Total marks are 25. The individual marks for each task are given with the task description above.

4. Instructions

- Submit the material to MLS in a single zip file (described below).
- Inside your root directory, make a separate directory for each deliverable.
- Use Python version > 3.6 and OpenCV version > 3.4
- **Do not use any high-level OpenCV functions except those that load and store an image, and those that have explicitly been mentioned in the task description.**
- All code that you submit should be original. Do not mine the internet for solutions.

5. Bonus Task

- For bonus points: In addition to visually comparing the quality of each image interpolation output to the original image, quantify your results using the Mean Squared Error (MSE) metric. A description and formula for MSE can be found here.