CMP SCI 4420/5420 **Image Processing and Computer Vision**
Assignment # 4 Due Date: October 25, 2022

Intensity Transformation through Histogram Equalization and Matching

You are given a set of images which require enhancement. Some images are dark, while others are light, low contrast, or high contrast. How do you enhance these images?

Purpose

The goal of this assignment is to learn some techniques to improve the visual quality of a given image. You are given an image of size $M \times N$ pixels, with k bits used for the intensity resolution. The image may be very dark or very bright and can be improved if we can redistribute the intensities to span a wider range of intensities than are being used.

Histogram Equalization and Matching

There are several techniques for image enhancement. Histogram equalization is a technique to enhance the visual quality of the above types of images. Histogram equalization automatically determines a transformation function which is applied on the input image to produce an image that has an approximately uniform histogram. An image with a uniform histogram shows a great deal of detail and high dynamic range.

In some application domains, uniform histogram-based image enhancement may not be appropriate. In such applications, a user specifies the shape of the histogram desired for the processed image. The method used to generate an output image with the specified histogram from an input image is called histogram specification/histogram matching.

Task

Apply both histogram equalization and histogram specification techniques to enhance a given image. Please do it from first principles, that is, implement the algorithm discussed in class; *do not use the built-in function in the OpenCV library*.

You will select the function to be applied from command line.

Your program should provide the following functions:

- Perform histogram equalization on a grayscale image.
- Perform histogram matching on a grayscale image using a reference grayscale image.
- Perform histogram matching on a grayscale image using a normalized histogram. The specified histogram will be in a file, with one floating point number on each line, giving the probability of the pixel with that intensity. Needless to say, this file will have 256 lines.

Invoking the solution

Your solution will be invoked using the following command:

```
histo [-h] [-m=n] imagefile [histogram_file]
```

`histo`	Name of your executable
`-m`	n=1: Histogram equalization (default) n=2: Histogram matching with image n=3: Histogram matching with file

Read in the images as grayscale images. In the function, verify that the image is a grayscale image and throw an exception if it is not.

Suggested implementation steps

1. Parse the command line. You can create your own parser or use the class `CommandLineParser` provided by OpenCV. Each of the optional arguments may have a default value. If the user specifies the option `-h`, print a help message and exit.
2. Compute histogram of the image by scanning the pixel intensities.
3. Create histogram equalization as a function.
4. Apply histogram matching using the normalized histogram.
5. Make sure that you catch any exceptions thrown by OpenCV functions. They will help to save you a lot of trouble.

Criteria for Success

Your code should successfully achieve the histogram equalization and matching functions, both with specified histogram or reference image.

Grading

I'll use the following rubric to assess your submission.

1. *Overall submission; 25pts* Program compiles and upon reading, seems to be able to solve the assigned problem.
2. *Command line parsing; 5pts* Program is able to parse the command line appropriately, assigning defaults as needed; issues help if needed.
3. *Histogram equalization; 20 pts* Program is able to perform histogram equalization on grayscale images correctly.
4. *Histogram matching by reference image; 20 pts* Program is able to match the images correctly.
5. *Histogram matching by specified histogram; 10 pts* Program is able to apply histogram specification from normalized histogram.
6. *Code readability; 5pts* The code must be readable, with appropriate comments. Author and date should be identified.
7. *Code reusability; 5pts* The code is structured in the form of functions, in separate files with their own headers, to facilitate reusability.
8. *Exception handling; 5pts* The code should handle exceptions, thrown from within the code or OpenCV.
9. *Meta-files, 5pts* Meta-files such as `README`, `Makefile`, and revision history are provided and are appropriately written.

Submission

Submit an electronic copy of all the sources, `README`, and results. Create your programs in a directory called *username*.4 where *username* is your login name on delmar. This directory should be located in your `$HOME`. Once you are done with everything, *remove the executables and object files*, and issue the following commands:

```
% cd
% chmod 755 ~
% ~bhatias/bin/handin cs5420 4
% chmod 700 ~
```

Do not copy-paste these commands from the PDF; type in those commands.