CMP SCI 4420/5420 **Image Processing and Computer Vision**
Assignment # 6 Due Date: November 22, 2022

Image Compression and Highlighting

Purpose

This project has two programs. One program requires you to highlight a specific area of a grayscale image. The second program requires you to compute the Huffman code for a grayscale image and to compute the compression ratio achieved.

The main purpose of the first program is to get familiar with mouse-based operations in OpenCV.

Image Highlighting

Image highlighting is generally used to bring attention of a viewer to a specific area of an image while dimming the rest of the image to provide context.

Task

Write a function to display an image. On the image, select a rectangular area by using your mouse. You can start the selection by using the left click of the mouse and keeping it pressed. Move the mouse to a different point and release the mouse button. It will be nice if you can display the rectangle that is being used to select the area but do not show the rectangle once you release the mouse. Your program should capture the top left and bottom right coordinates of the rectangle. These points will be used to select the region of interest (ROI) that needs to be highlighted.

After the rectangle is selected, make a copy of the ROI and dim the entire image by multiplying it by 0.75. Perform histogram equalization on the ROI using the OpenCV function and insert the equalized image back into its position. Display the final image.

Image Compression

Compute the Huffman code for a given grayscale image. Output different code blocks (in binary) and determine the compression ratio achieved.

Task

Your program will take a grayscale image as input. It will compute the histogram of the grayscale image and use the probabilities to find the Huffman code in binary. You may use bitwise operations to compute the code or compute them as binary strings (using `std::string` class). If you use bitwise operators, please create it as an ordered pair or a `struct` with one part containing the length of the code and the second part containing the actual code. In either case, it will be useful to write a function to display the code, especially taking care of preceding zeroes.

Compute the average length of code blocks in number of bits and the compression ratio achieved. Also display the average code length achieved by Shannon's theorem.

Note that you are not required to create/save the compressed image; your code will just create the lookup table for code blocks. You should be able to count the number of bits in each code block.

Invoking the solution

Your solutions will be invoked using the following commands:

```
highlight filename
huffman filename
```

Suggested implementation steps

I suggest using an image that is at least 640×480 pixels for this project.

1. Parse the command line. You can create your own parser or use the class `CommandLineParser` provided by OpenCV. This project does not require any options but you are welcome to create a help message.
2. Create the GUI to determine the ROI. Add the functionality to display the rectangle after you are able to capture the coordinates. Account for the case where a user may indicate bottom-right coordinate before the top-left coordinate.
3. Apply the OpenCV function to equalize histogram for the extracted rectangle.
4. Display the final image.
5. Compute the histogram of the grayscale image using OpenCV function.
6. Compute the probabilities.
7. Compute the Huffman code. You may want to write down the helper functions as needed prior to computing the code.
8. Write the code into a file.
9. Compute the entropy and compression ratio.

Criteria for Success

Your code should be able to perform both the functions, in separate executables. In case you want to use the color image for the first program, make sure that you perform the functions on the value channel by converting the color image into HSV space. Convert the image back to BGR prior to display.

Please use OpenCV functions wherever you can, such as to compute histogram and to perform histogram equalization. Do not use the functions that you developed from scratch in the previous projects.

Grading

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

1. *Overall submission; 30pts* 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. *Proper use of mouse operations; 15pts* Program is able to select the ROI appropriately using mouse as described.
4. *Highlight filter; 10pts* Program appropriately computes and applies the filter as specified.
5. *Huffman code; 15 pts* Program appropriately computes the Huffman code and average bit length.
6. *Code readability; 10pts* The code must be readable, with appropriate comments. Author and date should be identified.
7. *Code reusability; 10pts* 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.

Submission

Submit an electronic copy of all the sources, `README`, Makefile(s), and results. Create your programs in a directory called *username*.6 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 6
% chmod 700 ~
```

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