CMP SCI 4420/5420 **Image Processing and Computer Vision**
Assignment # 5 Due Date: November 08, 2022

Lomographic Filters

Purpose

This project leads to the development of a lomography filter. Given an image, apply the lomography filter to it and save the output. The learning objectives in the project include: ability to use a trackbar to modify some parameter of a filter and the use of callback to apply the filter function. In addition, the project will demonstrate the use of a lookup table (LUT) to improve the performance of filters.

Lomography

Historically, lomography filter is used to modify a photograph to improve its aesthetics. It is used in modern applications such as Instagram to improve the quality of a picture to make it more pleasing. It is named after the Soviet-era cameras produced by Leningradskoye Optiko-Mekhanicheskoye Obyedinenie (LOMO). *Lomography* is claimed by Lomographische AG as a commercial trademark but has become a *genericized trademark*. A number of photo editor apps include a *lomo filter*.

Task

This project involves creating two separate filters for lomography. You will develop a simple GUI using two separate trackbars to change the parameters of those filters. The filters will be applied to an image supplied on the command line. Your program will provide the following two filters:

1. An effect to manipulate one color channel. You should apply the effect to the red channel using a LUT. The LUT can be computed using the following expression:
$$\text{LUT}_i = \frac{256}{1 + e^{-\frac{(i/256)-0.5}{s}}}$$
where $0.08 \leq s \leq 0.20$ is a parameter to be changed by trackbar. Since we use 8-bit images, the value of i varies from 0 to 255. This expression makes dark pixels darker and bright pixels brighter.
2. A vignette effect by applying a dark halo to the image; the radius of the halo circle is defined by the trackbar and is specified as a percentage value (between 1 and 100).

Invoking the solution

Your solution will be invoked using the following command:

```
lomo -h filename
```

`lomo`	Name of your executable
`filename`	File containing the image

The parameters prefixed with − are optional. You are free to use long parameter names.

Suggested implementation steps

You should use an image from your corpus for this assignment so that it fits completely into the screen.

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. Set up the two trackbars and display the image. Keep the default for color trackbar at 0.1 and that for the halo effect at 100. Since the trackbars can take only integers as parameters, set up the range for color trackbar from 0 to 20 and divide it by 100 to get the appropriate numbers within the callback function. Also, ensure that you ignore any value below 0.08 as that will cause undesirable effects.
3. *Color trackbar callback function.*
   (a) Create a 1D `LUT` of 256 elements. It is a `Mat` with 1 row and 256 colums, with type `CV_8UC1`. Populate it by evaluating the expression specified above. The callback will recompute this `lut` as the value of `s` is modified.
   (b) Decompose the input image into its three channels by using the function `split`. Apply `LUT` to the red channel (channel 2).
   (c) Merge the three channels back into a single output image and display it.
4. *Halo callback function.*
   (a) The input to halo filter is the output image from the color filter. If the color filter has not been activated, copy the input image into the output image of color filter. Make sure that you perform a deep copy (use `Mat::copyTo` function).
   (b) Create a 3-channel floating point matrix of the same size as the original image (type `CV_32FC3`. Assign each element of this matrix as 0.75.
   (c) Compute the maximum radius of the halo as the minimum of number of rows and colums in the image. Use the percentage from the trackbar to draw a circle of radius as r – percentage of maximum radius. Each pixel in this circle is assigned as 1 (white).
   (d) Use the `blur` function with a kernel of $r \times r$ on the halo matrix.
   (e) Convert the result of color image into a 32-bit floating point image (`CV_32FC3`). Multiply the output of color filter with the halo filter. Convert the result in 8-bit image (`CV_8UC3`) and display the resulting image.
5. If the user hits `q`, terminate the program. If the user hits `s`, save a copy of the image and terminate the program.
6. Make sure that you catch any exceptions thrown by OpenCV functions.

Criteria for Success

You are free to use your own methods to solve the problem. As a minimum, you should modify one channel. You are free to improve to modify all three color channels.

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; 10pts* Program is able to parse the command line appropriately, assigning defaults as needed; issues help if needed.

3. *Proper invocation of trackbars; 10pts* Program is able to invoke the trackbars and accept the user input in an appropriate manner. The modified images are displayed right away as the values in trackbars are changed.
4. *Color filter; 10pts* Program appropriately computes and applies the LUT as specified.
5. *Halo filter; 10 pts* Program appropriately computes and applies the halo filter as specified.
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.
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`, Makefile(s), and results. Create your programs in a directory called *username*.5 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 5
% chmod 700 ~
```

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