CS 548: Assignment 1

Filtering

1 Overview

You will implement code to:

- load images from a directory
- load a filter from a text file
- apply the filter to each image
- save each output image into another directory

2 Filters

You will load a text file that contains the filter mask you will apply to your images. This file will have the following format:

```
<rows> <columns>
<H/L>
<mask values>*
```

For example, a (3×3) Gaussian filter would look like this:

```
3 3
L
1 2 1
2 4 2
1 2 1
```

The <H/L> part will either be **H = high-pass filter** or **L = low-pass filter**.

For the low-pass filters, when computing convolution, you will need to divide by the sum of the values in the filter. So, for the Gaussian filter above, you would divide by 16.

For the high-pass filters, do NOT divide by the sum of the mask values, because then you would be dividing by zero.

You may assume the filters have odd-numbered sides.

You may NOT assume the filters will have equal width and height!

3 Requirements

- You may use the sample code given in the OpenCV/Boost guides as a starting point.
- Your code will rely on the OpenCV and Boost libraries ONLY.
- Your code should NOT use OS-specific libraries/functionality!
- Your program should be able to take three command line parameters:
  1. The directory path containing your image files to process (e.g., *srcDir*)
  2. Path to the filter file
  3. The directory path your program will save the processed images (e.g., *dstDir*)

```
assign1 ../images ./filter.txt ../output
```

- The *.cpp file that contains your main() function should be named “assign1.cpp”.
- Your program should search for any images in *srcDir* with the extension “.bmp” or “.BMP”
- You do NOT have to do a recursive search in the directory.
- The image should be loaded and saved as a GRAYSCALE image, so use IMREAD_GRAYSCALE when calling *imread*.
- For your destination image, you should create a FLOATING-POINT grayscale image for output using the following:

```
Mat filterImage = Mat::zeros(cv::Size(colCnt, rowCnt), CV_32FC1);
```

- If you write functions that take OpenCV Mats, you may have to pass them in by reference.
- To get and set the value for a single pixel from a UNSIGNED CHAR, GRAYSCALE image:

```
uchar pixel = image.at<uchar>(y, x);
image.at<uchar>(y, x) = pixel;
```

- To get and set the value for a single pixel from a FLOATING-POINT, GRAYSCALE image:

```
float pixel = image.at<float>(y, x);
image.at<float>(y, x) = pixel;
```

- For output images from **low-pass filtering**, make sure your output pixel values are in the interval $[0, 255]$.

- For output images from **high-pass filtering**:
    1. Compute the sum of all POSITIVE values in the mask ahead of time ($maskSum$).
    2. Add on $maskSum * 255.0$ to your output pixel value.
    3. Divide output pixel value by $maskSum * 2.0$.
    4. Finally, make sure your output pixel values are in the interval $[0, 255]$.
- For each image:
    1. Load the image data into an OpenCV Mat
    2. Apply the mask using **convolution**
    3. Save the blurred image with the same name as the original file, but save it in *dstDir*
- Again, you should be using **CONVOLUTION**! You should NOT assume the mask is isotropic (rotation-invariant)!
- Do NOT use OpenCV's blurring and/or filtering functions! You MUST write these functions yourself.
- Make sure you deal with boundary conditions! Assume any pixel out of bounds has a value of 128 (half-intensity).
- Consider making a class or struct to hold filter information.

Tip: If you want to CHECK your output, load the "ground truth" images provided and do a pixel-by-pixel comparison with your results. **However, do NOT leave this code in your submission!**

4 Grading

Below is a list of SOME of the grading penalties.

The following will result in grading penalties:

- Incorrectly weighting each pixel in the mask area
- Failing to divide by the sum of the mask values for low-pass filters
- Dividing by zero with the high-pass filters
- Failing to deal with boundary conditions
- Doing correlation instead of convolution
- Saving output files with a different filename than the original filename

- Failing to save the output images
- Output images that are egregiously incorrect
- OS-specific code
- Use of hard-coded paths instead of command line parameters
- Using cin instead of command line parameters
- Code that does not compile
- Code that crashes, takes forever to run, or does not run at all
- Using OpenCV filtering functions instead of using your own functions
- Outputting the “ground truth” images instead of the images filtered with your algorithm
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class