Programming Assignment #3
CS253, Fall 2016
Due: Tuesday, Sept. 20th

"*Comparing Images*"

Motivation

For the last two weeks, you have been histogramming sets of numbers. But they weren't just sets of numbers. They were depth images of human hands. You know what a standard image is – a set of color pixels arranged in an (x,y) grid. Depth images are the same, except instead of a color you get a single value which is the distance from the camera to the object. The reason our test files had so many 255s in them is that 255 is the value returned when the distance it too great for the sensor to judge.

This week, we add the header information that tells you it's an image (and what the size of the image is). This will allow you to view the inputs as images using many programs available on the department machines (e.g. eog). It also means you will have to change your reading code a little bit, and that more error cases are now possible. We are also going to make you change your histogram comparison code a little bit. And add one more method of comparing two inputs (depth images).

Task

Your program will take two file names as arguments. (In other words, the argc parameter in main should now be 3, and argv[1] and argv[2] will hold file names.) The input files should now be images in .pgm format (although you do not care what the file is named). Every pgm image file begins with two characters: P2. After that comes whitespace and then two integers, the width and height of the image. These are followed by a 3rd integer, which is the highest legal pixel value, in our case 255. Note that the width, height and max possible value are integers separated by whitespace. Any type or amount of whitespace.

After the maximum possible value, the rest of the image is the same as before: integers between 0 and 255, separated by whitespace (including possible trailing whitespace). Any non-whitespace character other than an integer is an error. Now, however, there must be exactly width*height integers in the file. Any fewer or any more, and the file contains an error. If either input contains an error, your program should print an error message to std::cerr and return -1.

In assignment #2, you compared two histograms by normalizing them, multiplying them together pairwise, and then returning the sum. This time you will compare the two images by histogramming them and normalizing their histograms. This time, however, when you compare the normalized histograms, instead of multiplying them together pairwise and taking the sum, you compute the pairwise minimum and then take the sum of the minima. Note that this will still produce a value between 0 and 1.

In addition, you can compare two images without creating histograms at all. Just compare the 1st pixel in the first image to the first pixel in the second image. Then compare the second pixel to

the second pixel, and so on. In this case, by compare I mean subtract the first image's pixel from the second, and then square the result to get a squared difference value. Then sum all the squared difference values to get the final comparison value (which is the sum of squared differences).

When there are no errors, your program should print two values to std::cerr separated by whitespace: the histogram comparison number (between 0 and 1), and then the sum of squared differences (which will be very large).

Submitting Your Work

Unlike PA1 and PA2, this time you determine how your code is compiled. You must submit a tar file that includes not only your .cpp and .h file, but also a makefile. The makefile MUST CREATE AN EXECUTABLE CALLED PA3. Submit the tar file as PA3 on Canvas.

Grading Your Homework

To grade your assignment, the GTAs will unpack your archive in an empty directory and compile it using "make". If your program does not compile using make, you may be awarded no points for the assignment. Assuming your program compiles, the remaining points will be determined by how your program performs on test cases, including test cases with errors in the input files. When the input files do not contain errors, you will receive full credit for a test if the value printed to std::cout is correct. If an input file does contain errors, you will receive full credit if your program produces a meaningful error message to std::cerr and returns -1. Note that unless there is an error, the ONLY thing you output to std::cout or std::cerr should be the computed value.

Hints

- Make sure the number of pixels declared in the header matches the number of pixels in the file.
- People are making tar mistakes. Before submitting you program, do with it what we will do: copy it to an empty directory, untar it, make it, and run it on test files (that you made).

Policies

All work you submit must be your own. You may not submit code written by a peer, a former student, or anyone else. You may not copy or buy code from the web. The department academic integrity policies apply.

You may not submit your program late. To receive credit, it must be submitted on Tuesday, Sept. 20th. There is no late period. The exception is an unforeseeable emergency, for example a medical crisis or a death in the immediate family. If an unforeseeable emergency arises, talk to the instructor.