Programming Assignment #5
CS253, Fall 2016
Due: Thursday, Oct. 13th

"*Clustering Images*"

Motivation

In PA4, you converted a set of images into normalized histograms, and for every image you found the most similar other image. That was a small step in an important direction. This time you will take a bigger step to a more interesting destination: image clustering.

This is also the part of the course where the programming assignments get harder. We are into the middle third of the course, and the topic is now object oriented design. Therefore, the assignments get bigger in order to give you the opportunity to do some real design. But with opportunity comes danger: starting with this assignment, if you don't do object oriented design your code will quickly become unmanageable. Nothing in this assignment suggests polymorphism (or inheritance or hierarchies), but for your own sake please design well-encapsulated objects.

Task

Your PA5 program will take two command line arguments. The first input is the same as the input to PA4: a single file containing a list of filenames of depth images. As in PA4, your program will read every image listed in the input file and convert them into normalized histograms. The second input should be an integer whose value is greater than zero and less than or equal to the number of image files.

The task in PA5 is to cluster the images according to their normalized histogram similarities, using an algorithm called *agglomerative clustering*. For historical reasons we call the second argument *K*, and *K* is the target number of clusters. If the number of input images is *N*, the goal is to group the *N* images into *K* groups, such that the elements of each group are as similar as possible. For example, the images we have been giving you are hands in different poses. Let's see if normalized histograms are a good enough similarity measure to group hands according to their poses…

Specifically, your program will begin by assigning the *N* images to *N* different clusters. If K = N, you are done clustering. If K is less than N, reduce the number of clusters by 1 by finding the two most similar clusters and merging them. If K = N-1, you are done clustering. Otherwise recursively find the two most similar clusters and merge them. Keep this process up until the number of clusters is equal to K.

What exactly is a cluster? A cluster is a set of one or more images. The normalized histogram of a cluster is the average of the normalized histograms of the images in the cluster. The similarity between two clusters is the similarity between their normalized histograms. (Warning: if you merge a cluster with 5 images and a cluster with 2 images, the resulting cluster's histogram is not the average of the histograms of the two clusters. It is the average of the normalized histograms of the 7 component images.)

The output of your program consists of the filenames in each cluster. If the input asks for *K* clusters, your program should write K lines of output to std::cout. Each line of output should contain the image filenames in one cluster, with the name of every input file appearing on one of the output lines. Obviously, in case of an error you should write an error message to std::cerr and return a negative value.

Submitting Your Work

You must submit a tar file that includes not only your .cpp and .h files, but also a makefile. The makefile MUST CREATE AN EXECUTABLE CALLED PA5. Submit the tar file as PA5 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.

For the first time, we will also test your program using the memcheck tool in Valgrind, to see if it has any memory problems (memory leaks, use of uninitialized values, double deletes, etc.). If your program has any memory problems, points will be deducted.

Hints

- People are still failing to test their tar files. After you make your tar file, create an empty directory, copy your tar file to it, untar it, type make, and then test your code.
- Test your code (on a small case) using Valgrind.
- I recommend creating a Category class that is distinct from your Histogram class, while keeping Image and DataSet classes. Of course, you can adopt any design you want…

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 Thursday, Oct. 13th. 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.