CS688/WST665

Programming Assignment #3
Due Oct.-21 (Fri.) (before 11:59pm)

Objective: Understand inverted index and binary code embedding for speeding up image retrieval process.
Developing environment: Since PA3 is related to your PA2, Linux OS is recommended. Any other environments and programming languages are also okay for the homework, but TAs cannot provide supports on other settings.

Requirements:

1) In PA2, you have implemented feature extraction using CNN. In this homework, you need to use PA2's result for binary encoding (spherical hashing) and inverted index.
2) After retrieving CNN feature (descriptor), you need to encode your descriptor (both data and query) using spherical hashing. In that way you can reduce necessary memory size and speed up the searching process as well. Show the difference regarding data size, search time and accuracy. (10 pts)
   a. Spherical hashing's code is on the course homepage. You need to use cmake for generating makefile for spherical hashing (see the hint for cmake and how to build in the Linux environment).
   b. For using spherical hashing, you need to adjust input file(descriptor) for its pre-defined format (see the hint for more detail)

```
// format is below
 (num. points) (dimensionality) : they are both int
 v0 (floats, number of elements is equal to dimensionality)
 v1
…
```

   c. After you get the encoded data, compare its size, search time and accuracy to original descriptor in the report.
3) After applying spherical hashing, you will have binary codes of image descriptors. You need to implement inverted index on that data. In that way, you can reduce search space. Check whether it shortens the search time or not (5 pts)
   a. You can either use simple K-means clustering or PQ (product quantization) for inverted index.
   b. For implementing inverted index, K-means clustering on your encoded descriptor is recommended (it is easier) but you can try for PQ if you want.
   c. FYI. PQ source code (http://people.rennes.inria.fr/Herve.Jegou/projects/ann.html)
   d. Since descriptor data is now binary code, you can use simple hamming distance for similarity. You can find example hamming distance function (Compute_HD) in the code (in Spherical_Hashing_Cmake_Projects/common.h)
   e. Compare the search time and accuracy between unprocessed result and 3)'s result in the report.
4) You do not need to combine 2) and 3) for one program. You can do it separately if you want.

Deliveries:

1) Binary and a source file [ex) Tester_L2_Main.cpp, points.cpp, your inverted index code]
2) A report should contain at least three parts
   a. How you implement 2) and 3).
   b. Comparison of 2)'s result and input descriptor data for size, search time, and accuracy.
   c. Search time and accuracy comparison between unprocessed descriptor data and 3)'s data
3) Send your work (binary, source files, and report document) to TA, (soo.kim813@gmail.com, Soomin Kim)

Policies:

1) Everyone must turn in their own assignment. You can collaborate with others, but any work that you turn in should be your own.
2) **Late submission will not be accepted**.