Point Cloud Normal Estimation

Programming Problem III

CS 3513

Due: March 30, 4:30 p.m.

Estimating normals of a point cloud using eigenvectors.

Description: The programming problem is to estimate normals at each point of a point cloud without trying to reconstruct the surface. The basic process is that for each point $\boldsymbol{p}_i$:

- Find the k points in the cloud that are nearest to $\boldsymbol{p}_i$ using Euclidean distance. That is, find k points $\boldsymbol{p}_j$ that have smallest values of $\|\boldsymbol{p}_j - \boldsymbol{p}_i\|_2$ over all points in the cloud.
- Convert the k+1 points ($\boldsymbol{p}_i$ and the k neighbor points) to have zero mean.
- Form the estimated covariance matrix of the k+1 points.
- Calculate eigenvectors and use the eigenvector associated with the smallest eigenvalue as an estimate for the normal $\boldsymbol{n}_i$ at point $\boldsymbol{p}_i$.

See the PCA example from the class slides for an example of estimating the covariance matrix from a set of points.

Details: The solution to the programming problem requires a driver script. Name the driver program `pp3`. The command line parameters for pp3 are *inFileName*, *outFileName*, and k in that order. The input file will have one point per line. Each line has three value separated by whitespaces, that is, x y z. The output file should have one line per point. Each line should have the point location and normal separated by whitespaces, that is, x y z n_x n_y n_z. The normal vectors should be a unit vectors ($\|\boldsymbol{n}_i\|_2$=1). The parameter k is the number of nearest neighbors to use for the normal estimation. Example input files will be placed in the Data directory of the class subversion repository. The example output files in the class repository set the sign of the normal based on its orientation with respect to the centroid of all the points. For example with point $\boldsymbol{p}_i$, normal $\boldsymbol{n}_i$, and centroid $\boldsymbol{c}$ then if dot($\boldsymbol{p}_i$ - $\boldsymbol{c}$, $\boldsymbol{n}$i) < 0 then change the direction of the normal by $\boldsymbol{n}$i = -1*$\boldsymbol{n}_i$. Recall that a centroid of a point distribution is the center of mass or in this case, the mean location of the points. You may use an eigenvalue function from your matrix from (such as `numpy.linalg.eig`) to calculate the eigenvectors of the covariance matrix. Implementing inverse power iteration is extra credit.

General notes:

- Include your name in every file that you submit. If your source code is a modified version of code from a book, class example, or from the web, the comments must explicitly state *modified version of ...* or *based on example ....*
- If submitting using subversion, create a directory named `pp3` in your class directory and add your assignment files there.

- If submitting using D2L dropbox, place all of your assignment files in a directory named `pp3` and archive that directory using zip (or tar or etc.). Submit the archive. Note: be sure to take care of being consistent on using upper/lowercase in filenames. Some file systems ignore case and others don't.
- Submit a readme.txt file. The readme file should describe all known bugs that were not removed from the program and discuss problems encountered when developing and testing the program. The readme file should describe the programming language, complier version, OS environment, and all necessary steps to compile and execute your code. Non-standard libraries should be submitted with the code. You may assume that numpy (for python), eigen (for C++), and jama (for java) are all ready installed. Any external sources of information should be explicitly mentioned in the readme.txt and in the comments of the source code.

Extra Credit Components for programming problem 3

- (15%) Find the smallest eigenvalue and associated eigenvector by implementing inverse power iteration with LU factorization (see Figure 6.4 b). Note that the covariance matrices are positive semi-definite so that you can use Cholesky decomposition in place of LU. In either case, implement your own forward and back substitution methods for the factorization.