GENERATING COMPRESSED POINT CLOUD IMAGE REPRESENTATIONS

Christopher Miller; cmillerATmath.umd.edu
Advisor: Dr. John Benedetto; jbbATmath.umd.edu

Abstract. Urban terrain data can be conceptualizes as a piecewise two dimensional manifold embedded in a compact subset of $\mathbb{R}^3$. A problem arising frequently in image analysis is; given a discrete set of points sampled from a manifold called a point cloud, how can one form an approximation of the manifold. We propose the development of software suited to this task when the point cloud is formed by an aircraft using a Light Detection and Ranging (LIDAR) system to create a point cloud from a section of urban terrain. The key advantage of this algorithm over others is that the image representation will be highly compressed. This will allow data acquisition, transmission, and use to occur in real time.

Date: 14 October 2007.

1. Background

The use of aircraft-based LIDAR systems have become an important tool for creating elevation models of terrain data. To summarize the method: an aircraft is equipped with a laser range finder and a GPS system. The laser range finder calculates the ground elevation while the GPS system calculates the location of the aircraft over the surface of the earth. The combination of these two data streams produces an elevation model of a given area consisting of a large number of discrete points in three-dimensional space. This set of points is called a point cloud.

The problem of producing an image of the terrain from this point cloud is subtle and difficult. The first problem is that the point-cloud is non uniformly sampled. This is due to altitude and airspeed changes, as well as changes in the local curvature of the terrain being sampled. This problem makes typical multiresolution analysis difficult due to the varying density of information throughout nested subspaces.

The second problem is that due to various issues arising from the physical implementation of the LIDAR system, the resulting point clouds tend to contain a relatively high level of noise. If the reflectance or refraction index of a particular region being struck by the laser is an extreme value, incorrect values may be obtained.

Most methods for creating a representation of the terrain from the point cloud data focus on using interpolative processes to approximate the terrain locally using low degree polynomials on arbitrary meshes. These techniques have the disadvantage that they do not admit a compact representation. Since the mesh is arbitrary the sender needs to transmit the mesh information as well as the local polynomial coefficients. Other methods such as LOESS require the original data-set to be present in order to reconstruct the visualization. All of these methods prevent the real time transmission of terrain representations obtained through LIDAR point cloud sampling.

The objective of this project is to create software that will create a visual representation of urban terrain using a LIDAR point cloud. The method should be able to cope with the difficulties of non-uniformly sampled data and noisy data. The most important aspect however, is that the representation must be compact. This will enable real time transmission of the representation.

2. Approach

The algorithm we plan to implement was first proposed by Donoho called wedgelets[1]. Consider a point cloud of the form (x, y, z) where $x \in [0, 1], y \in [0, 1]$. The algorithm induces a multiresolution dyadic partition of the image space. $D_1 = [0, 1] \times [0, 1], D_2 = [0, 1/2] \times [0, 1/2], D_3 = [0, 1/2] \times [1/2, 1], D_4 = [1/2, 1] \times [0, 1/2], D_5 = [1/2, 1] \times [1/2, 1]$.... A line is drawn through each dyadic square dividing it into two disjoint sets called a wedge. On each wedge the function is approximated with a plane fitted using least squares regression. Donoho proved that for horizon model images this technique produced convergence on the order of h^2 where h is the scale of the smallest used dyadic square [1]. It is due to the rectilinear nature of urban terrain that we believe wedgelets is a promising representation.

The approximation on a given dyadic square can be described by eight parameters. Two parameters define the line while three parameters define each plane. It is clear that if the approximation introduces tolerable error on a particular dyadic and the original subset of the data contained in that square was large that there is considerable potential for compression. The wedgelet approximation to the surface will be taken to be some spanning set on the image space where the approximation on each square satisfies a user specified error bound on the residuals between the fitted planes and the data points. The data structure that wedgelets produces is a quad tree. By virtue of its structure we will need to encode little data regarding the location of each dyadic square relative to the entire image space.

One can define several norms over which to minimize the error. We will test several to determine which produces the best representation.

Noisy point clouds will present a problem. Wedgelets have noted some success in de-noising images[2]. De-noising using wedgelets however requires the user to accept a coarser mesh. This introduces greater approximation error which may or may not be tolerable. Most LIDAR noise consists of abnormally high or low values relative to a local neighborhood. We plan to deal with this problem by testing and implementing the outlier detection techniques outlined by [4].

Another problem is presented by the computational complexity of the algorithm. If the data set is large then the wedgelets algorithm will be required to preform QR factorizations on very large matrices several

thousand times to find the optimal wedge setup on each dyadic square. To combat this problem we plan to implement an algorithm described by Moenning and Dodgson[3]. Their algorithm reduces a point cloud by eliminating redundant sets of data. This process is lossy but will be vital to keeping running time reasonable.

3. Implementation

The fully recursive nature of the algorithm makes it a prime candidate to be paralyzed. We plan to run the algorithm on a dual core Intel Xenon running at 2.20 GHz. It is our hope that the algorithm for creating these representations will be able to be implemented in real time in the field. The platform we are currently using should be indicative of the type of computer that is readily available.

The algorithm will be implemented in C++. We plan on using only open source libraries so that the code should compile on any architecture without need for adjustment.

4. Testing and Validation

The Army Corps of Engineers has provided us with several point clouds taken from New Orleans, LA. They have also given us gridded images produced from those point clouds. Validation will be accomplished by comparing the result of our algorithm on the point cloud data with the gridded image. We should expect that the resulting image will correspond to the gridded image in terms of basic features. Our image may not be completely as accurate due to the compression but, our representation should be much smaller in memory.

We have access to a tool called TSSIM which compares two images. It gives special weight to the features of urban terrain that we are interested in. This tool will be the basis of our comparison.

Additional LIDAR point clouds and corresponding gridded images are available from the USGS via "http://lidar.cr.usgs.gov/".

5. Project Schedule

October.

- Present Project Proposal.
- Implement a first draft of the wedgelet algorithm.

November.

- Implement TSSIM
- Compare results with gridded images.
- Assess image quality and running time.

December.

- If deemed to be an issue reduce running time.
- Implement error norms other than L_2.
- Assess image quality when the optimal wedge is computed using other norms.
- Deliver Midterm Report.

January.

- Implement de-noising algorithms.
- Parallelize wedgelets algorithm. (time permitting)

February.

- Assess effectiveness of de-noising algorithms.
- Implement Moenning and Dodgson's point cloud reduction algorithm.
- Assess effectiveness.

March.

- Develop GUI front end.
- Begin Final Report.

April.

- Complete Final Report First Draft.
- Edit Final Report.

May.

- Present Results.
- Deliver Final Report.

References

[1] David L. Donoho *Wedgelets: Nearly Minimax Estimation of Edges*, The Annals of Statistics, Vol. 27, No. 3. (Jun., 1999), pp. 859-897.

[2] Laurent Demaret, Felix Friedrich, Hartmut Fhr, Tomasz Szygowski, *Multiscale Wedgelet Denoising Algorithms*, Proceedings of SPIE, San Diego, August 2005, Wavelets XI, Vol. 5914, X1-12

[3] Moenning C., Dodgson N. A.: *A New Point Cloud Simplification Algorithm.* Proceedings 3rd IASTED Conference on Visualization, Imaging and Image Processing (2003).

[4] S. Sotoodeh , *Outlier Detection In Laser Scanner Point Clouds*, IAPRS Volume XXXVI, Part 5, Dresden 25-27 September 2006