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 to the manifold. In this paper we document the implementation of a image transform called wedgelets to accomplish this task. 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.

1. Background

The use of aircraft-based Light Detection and Ranging (LIDAR) systems have emerged as an important tool for creating digital terrain elevation models. An aircraft equipped with a LIDAR system transmits pulses of short wavelength laser light towards the ground. The time the beam takes to travel to the earth's surface and back coupled with the airplane's internal GPS navigation system provides accurate terrain elevation data. The data gathered from this procedure consists of a large number of points in $\mathbb{R}^3$. This set of points is referred to as a *p*oint cloud.

A problem exists when one attempts to consider these point clouds as images of terrain. The sampling rate within the point cloud is generally non uniform due to variations in airspeed, altitude, terrain curvature etc. Furthermore, the points are not aligned to a grid meaning that the natural interpretation of points as image pixels is not valid.

Methods for generating a representation of the terrain from point cloud data generally focus on using interpolative processes to approximate local terrain behavior using low degree polynomials on arbitrarily shaped meshes. Wedgelets is such a technique with the exception that there are regularity conditions imposed on the mesh. These conditions

Date: 20 December 2007.

allow the information encoded by the wedgelet transform to be encoded in a compact data structure.

2. Wedgelet Transform

The wedgelet transform was first described by Donoho for representing functions $f : [0,1]^2 \to \mathbb{R}$ such that f is piecewise constant with C^2 boundaries between constant regions. The transform works by partitioning I^2 into a set of overlapping dyadic squares Q_j of size 2^{-j}.

$$Q_j = \{[2^j k, 2^j(k+1)] \times [2^j m, 2^j(m+1)] : 0 \leq j \leq J, 0 \leq k, m < 2^{J-j}\}$$

[3] This partitioning induces a quadtree structure on the unit interval.

Divide each dyadic square into two 'wedges' by drawing a line through the square. On each side of the line approximate the function by a constant. Let $g_{i,j}$ be the approximation on the i'th dyadic square at scale j. If $f_{i,j}$ is the restriction of f to the given square, find the line that minimizes $||f_{i,j} - g_{i,j}||_2$.

The above transform described by Donoho naturally generalizes if 'piecewise constant' is replaced by 'piecewise linear' or 'piecewise n'th degree polynomial'. In the case where f can be reasonably approximated by a piecewise linear function minimizing $||f_{i,j} - g_{i,j}||_2$ is a relatively cheaply computed linear least squares optimization. For functions requiring higher degree polynomials computing the optimal approximating function is nonlinear least squares which may require substantial computational resources.

3. Implementation

3.1. **Wedgelet Transform.** Our implementation of the wedgelets transform was written in C using the GNU GSL science libraries to perform the matrix computations. It currently runs on a 2.2ghz intel dual core system with 4GB of RAM. Our algorithm for computing the wedgelet transform works as follows. The point cloud is inputed to the algorythm as an n by three matrix of floating point numbers.

$$PC = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_{n-1} & y_{n-1} & z_{n-1} \end{bmatrix}$$

From this the matrix $P = [\vec{x}, \vec{y}, \vec{1}]$ and the vector $\vec{b} = [\vec{z}]$ are initialized.

The user must choose several parameters.

- The number of angles to be used.

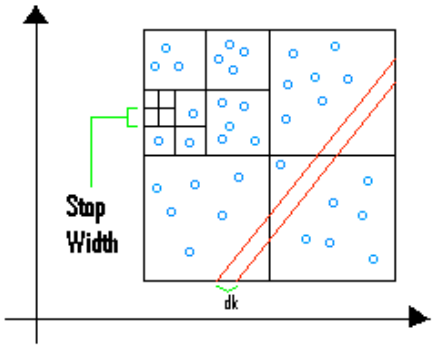

FIGURE 3.1. Illustration of User Set Parameters

- The minimum line translation Δk.
- The stop width: the minimum dyadic square size.

The first two parameters define the discrete set of lines to be used in dividing the wedges and the second stops the recursive process.

For each line in the chosen discrete set the matrix P and vector $\vec{b}$ are divided into the two partitions induced on the point cloud by the given line.

$$P = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_3 & y_3 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} P_1 = \begin{bmatrix} x_1 & y_1 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_7 & y_7 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} P_2 = \begin{bmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

For each of these matrices the least squares solution to the system $P_1[dx_1, dy_1, z_1]^t = \vec{b_1}$, $P_2[dx_2, dy_2, z_2]^t = \vec{b_2}$ is computed. The planes defined by $g_{i,j,1} = dx_1x+dy_1y+z_1$ and $g_{i,j,2} = dx_2x+dy_2y+z_2$ are the best linear approximations to the data on each side of that given wedge. If $||f_{i,j}-g_{i,j}||_2$ is minimal then the coefficients $(\theta, k, dx_1, dy_1, z_1, dx_2, dy_2, z_2)$ are stored. Each of P_1 and P_2 are dyadically divided into four submatrices and the above process is recursively applied.

When the dyadic squares are sufficiently small the matrices can become underdetermined. These cases are handled as follows.

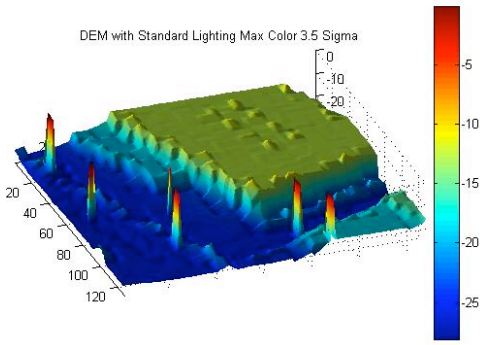

FIGURE 3.2. Instability using the QR factorization

- Either matrix contains zero points: In this case the wedge is discarded.
- A matrix contains one point: In this case the underdetermined system is set to a constant.
- A matrix contains two points: The wedge is set to zero in the undetermined dimension and the system is solved. This produces a unique solution

If either matrix contains few points then there is a stronger possibility that it will be nearly rank deficient. This produces a stability issue when solving the least squares problem using the QR factorization. If a matrix contains fewer than twenty points the SVD is used instead of the QR factorization for computing the least squares solution. If small singular values exist they are zeroed out and the system is solved only in terms of the larger singular values.

3.2. **Pruning.** We now have a representation of the function sampled by the point cloud at multiple dyadic levels. A criterion is needed to control which wedges are used to approximate the function. For fixed $\lambda \geq 0$ Donoho defined the optimal wedgelet partition of an image to

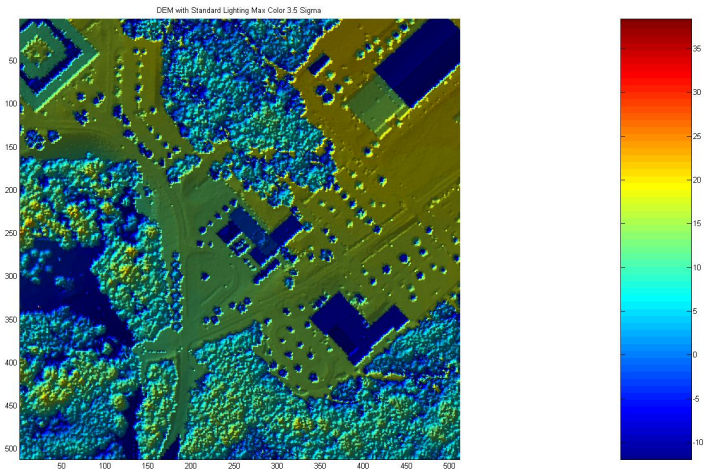

FIGURE 4.1. Image Reconstructed from Point Cloud using Matlab's Gridding

be the minimizer of the functional, $F(|W|, \lambda) = ||f - g|| + \lambda|W|$, where $|W|$ is the number of dyadic squares used in the approximation, and g is the approximating function. λ acts as a penalty for approximations using large numbers of wedges. Larger values of λ result in a coarser approximation while smaller values of λ produce a more detailed image at the cost of additional storage. In practice the minimizer can be found using any norm. In practice it has been experimentally verified that the $L1$ norm produces optimal results.

Once a suitable set of wedges has been chosen the function is reconstructed using the approximating functions on each dyadic square.

4. Sample Images

The sample point cloud came from the Army Corps of Engineers TEC division. The point cloud was taken over Ft. Belvior VA. The average sampling density was $2\frac{sample}{m^2}$. The following images are displayed at a scale $1pixel = 1m^2$.

The following plots show wedgelet reconstructions for several values of lambda. The last plot compares the structural similarity between the wedgelet approximation and the gridded image as a function of λ chosen during pruning, Information on the structural similarity index (TSSIM) can be found in [1]

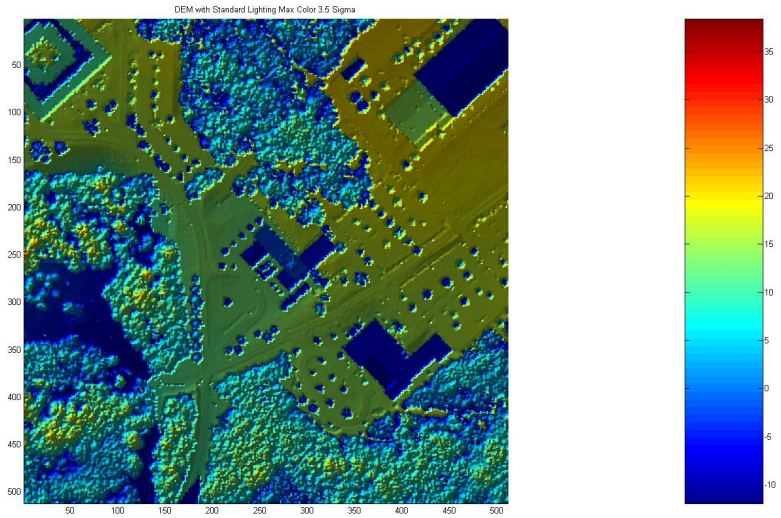

FIGURE 4.2. Using Wedgelets $\lambda = 1$

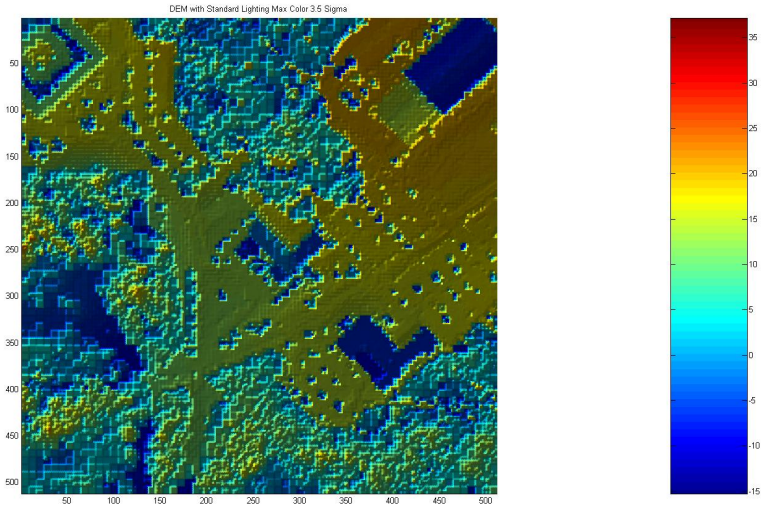

FIGURE 4.3. Using Wedgelets $\lambda = 75$

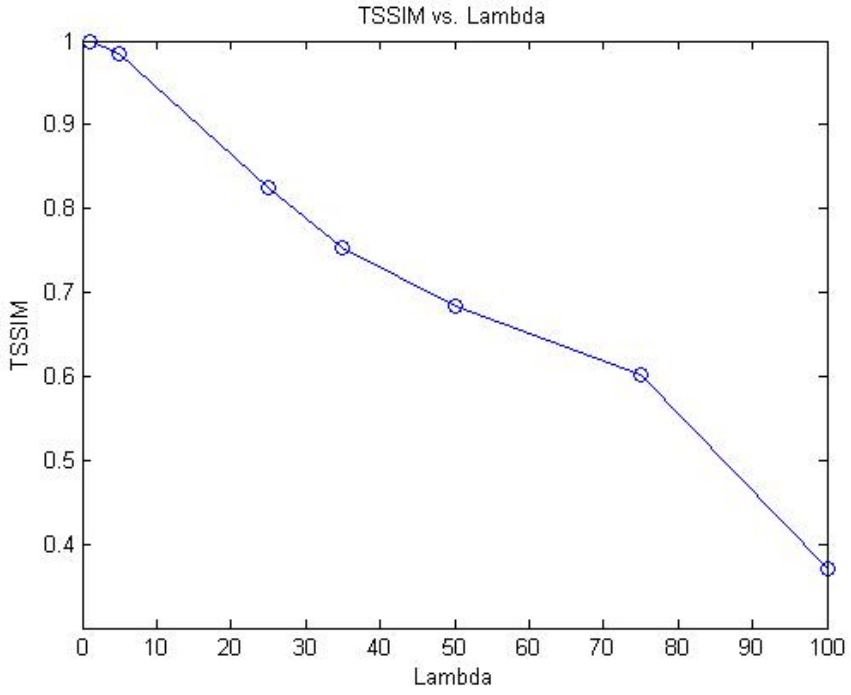

FIGURE 4.4. Structural Similarity Index (TSSIM) Between Original and Reconstructions

One would expect the quality index to be monotonically decreasing in λ and this is indeed the behavior we observe.

5. Further Work

January

- Implement de-noising algorithms. [5]
- Assess Compression Rates.
- Parallelize wedgelets algorithm.

February

- Assess effectiveness of de-noising algorithms.
- Implement Moenning and Dodgson's point cloud reduction algorithm. [4]
- Assess effectiveness.

March

- Develop GUI front end.
- Begin Final Report.

April

- Complete Final Report First Draft.

May

- Present Results.
- Deliver Final Report.

References

[1] Honghua Chang, Jianqui Zhang *Evaluation of Human Detection Performance Using Target Structure Similarity Clutter Metrics*, Optical Engineering, September 2006, Vol. 45, Issue 9, 096404.

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

[3] 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

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

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