

Progressive Transmission of Lossily Compressed Terrain

Zhongyi Xie¹, Marcus A. Andrade^{1,2}, W Randolph Franklin¹, Barbara Cutler¹,
Metin Inanc¹, Jonathan Muckell¹, and Daniel M. Tracy¹

¹ Rensselaer Polytechnic Institute,
Troy, New York, 12180-3590, USA;

² Universidade Federal de Viçosa, Viçosa, Brazil

Abstract. The distribution and management of spatial data requires strategies for handling large amount of terrain data that are now large available. Especially, data like LIDAR and Digital Elevation Model (DEM) data which are being used in a large group of diversified users. In this paper, we propose a progressive terrain data transmission scheme which take advantage of Over-determined Laplacian PDE (ODETLAP), which can achieve a compromise of high compression ratio and accuracy. The ODETLAP can be thought of as a compressor of original terrain data and we use Run Length encoding as well as linear prediction to reach a higher compression ratio. The latter two alone can do a better job than bzip2. Concretely, our technique is capable of reducing a hilly DEM dataset to 1% of its original binary size and a mountainous DEM dataset 3% of its original size. The accuracy loss in elevation and slope are discussed.

Key words: DEMs, Terrain Compression, Progressive Transmission, Levels of Details, Lossy Encoding.

1 Introduction

In the past a few years the geographical information science(GIS) community has seen an explosion in the data volume and great improvement in the accuracy of data acquisition. High-resolution terrain elevation data at 3m is now widely available through USGS' web site, which makes possible more realistic rendering of terrain features by all kinds of visualization softwares. Such growth of terrain data is reflected in the growth in number of internet sites offering terrain data as freeware, for example, United States Geological Survey's DEM data sets [1]. Also we see great growth in Mobile GIS services and GPS units are more and more affordable and widely used.

Digital Elevation Models (DEMs) represent the continuous surface of earth using a regular grid of samples which record the height value. It is a potential tool for terrain analysis at varied spatial and temporal scales.

One common goal for general compression implementations is to achieve maximum compression while minimizing processing time. However there are users who are more concerned about the quickness of the algorithm while also exists

some users that expect the maximum compression ratio irrespective of the processing time due to the availability of supercomputers. In this paper we propose a DEM-specific lossy compression algorithm and its application in progressive transmission. Our algorithm is more concerned about the compression ratio than the accuracy and CPU-time.

This paper is organized as follows: Section 2 gives a brief summary of existing algorithms that does terrain compression and progressive transmission, section 3 gives an overview of our algorithm, section 4 presents our algorithm for DEM compression. . Section 5 summarizes the experiments we did and comparison with other methods.

2 Related works

This section presents a short review of existing methods for progressive transmission of geospatial data over the world wide web. Basically, the methods can be divided in two categories: raster data and vector data transmission.

In general, most of methods developed for raster data transmission are based on image compression techniques since they can provide good compression with no (or low) loss of information. Some simple methods randomly select subsets of pixels from the image being transmitted and incrementally completes the image by adding pixels [2, 3]. Other more elaborate strategies [4–6] use some hierarchical structure, such as quadtree, to partition the image and select more pixels from those parts containing more details. Thus, the image quality can be incrementally improved by transmitting/including points in those image parts.

Other more sophisticated methods, for example [7–12] are based on advanced compression techniques such as wavelets decomposition, JPEG compression and JPEG2000. In general, these compression methods decompose the data in rectangular sub-blocks and each block is transformed independently. Furthermore, the image data is represented as a hierarchy of resolution features and its inverse at each level provides subsampled version of the original image. Thus, it is quite natural to apply this strategy for progressive transmission.

Generally, the raster data transmission over the internet is used for visualization purpose and so, the raster progressive transmission methods are very efficient since visual meaning can be extracted from images at very low resolution. However, in some applications, the objects need to be manipulated and/or processed to compute or to extract some additional information and, in this case, a vector representation is used.

Often, in terrain modeling and surface reconstruction and rendering, vector data are represented using triangular meshes [13–15] that can be compressed by several methods which are either based on optimal point decimation, such as [16–19] or exploit the combinatorial properties of the mesh [20–22]. Although these compression methods can be used for progressive transmission, an important bottleneck is the high storage space required for vector representation (even after compression).

3 Progressive Transmission based on ODETLAP

Conventional terrain/map related software that requires the availability of all data does not support any operation before the transmission process ends. Despite the development of networking technology and hardware, terrain data are still relatively large (a few Giga bytes) compared to the network speed (below 1 Mega bytes for most users). As a consequence, conventional transmission methods' response time would inevitably make them far more than being interactive. Progressive transmission, was proposed as a solution to the above problem. The idea is to successively send the terrain data starting with a coarse simplification, and on the user end, visualize or start some other operation in the first place. Although progressive transmission may extend the total transmission time due to the extra time needed for multiple operations, it has advantages over the traditional method in the following ways. Firstly, the client does not have to wait until all the data are received before any operation or viewing can be done, which increases the interactivity of the system. Secondly, the client can determine whether the current surface is accurate enough before all the data are sent and received. This saves the transmission time and resource need and increases the flexibility and efficiency of the system.

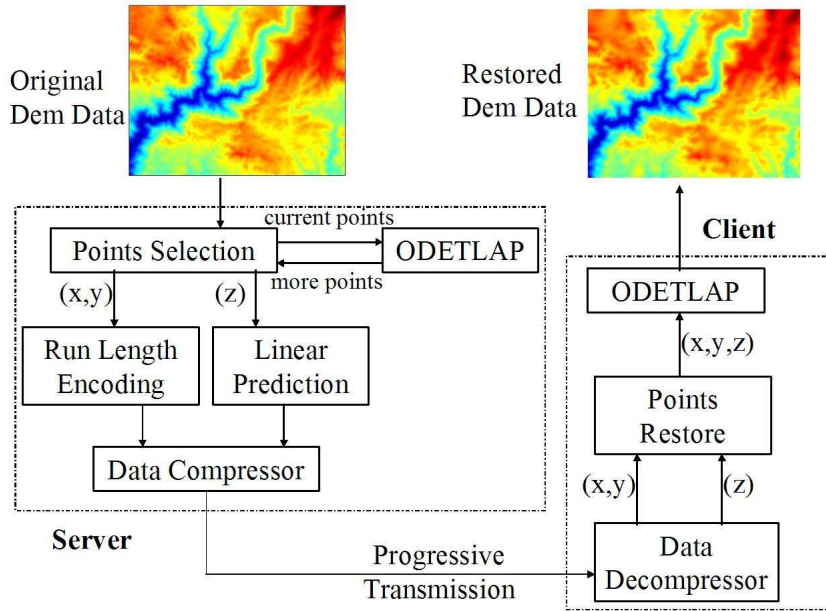


Fig. 1: Progressive transmission flowchart

Our ODETLAP based progressive transmission system consists of a server which first selects a subset of original points and sends those points to a client that reconstructs the surface using ODETLAP. After that, the reconstructed terrain is evaluated. If necessary, the client would request more points from the server, which could result in a more accurate approximation of the terrain. Figure 1 shows the whole process. The user end can then do more operations based on the reconstructed surface. While other existing progressive transmission methods transmits all the data from the server to the client, our progressive transmission system has the advantage that only a subset of points need to be sent, which saves a lot of networking resource as well as transmission time. As long as some data points reach to the client side, the whole terrain can be approximated by the ODETLAP, so the user can see the terrain as early as possible. When more points arrive at the client side, it can reconstruct the terrain more accurately from the extended points set, which gives user a more detailed view of the terrain.

4 Over-determined Laplacian Approximation

4.1 Definition

The Over-determined Laplacian Approximation (ODETLAP) is an extension to Laplacian equation

$$4z_{xy} = z_{x-1,y} + z_{x+1,y} + z_{x,y-1} + z_{x,y+1} \quad (1)$$

This equation states that for every non-border point identified by coordinate (x, y) in the elevation matrix, the elevation z_{xy} is equal to the average of its neighbors. Handling of border points form a special case and is omitted due to the lack of deep theoretical interest. This equation has the limitation of being unable to represent local maxima in terrain modeling, which is the reason we import equation 2.

$$z_{xy} = h_{xy} \quad (2)$$

Equations 1 and 2 form a basis for our over-determined linear system. The system's input is a series of points (x, y, z) which indicates the elevation of certain locations (x, y) . For those locations with known elevation, we have both equations, and for the rest locations, only equation 1 is valid. The relative importance of two sets of equations is determined by a parameter R during the approximation/interpolation process. Weighting equation 2 over equation 1 results in a more accurate surface which sacrifices smoothness, while weighting equation 1 over equation 2 gives us a smooth surface.

The idea of ODETLAP is when we have incomplete information about the actual elevation matrix, we can use the known value and the constraint(average of its neighbors) to approximate/interpolate the elevation value for every unknown and known point. So ODETLAP can be considered as a solver whose input is a set of known points (x, y, z) and an interpolation parameter R and

outputs the DEM matrix of the complete terrain. The benefits of ODETLAP include the ability of handling continuous as well as broken contour lines of elevations, processing kidney-bean-shaped contours without giving fictitious at regions inside and infer local maxima from a series of contours.

4.2 Algorithm

Since ODETLAP is capable of reconstructing the whole DEM matrix from a few sparse input points (x, y, z) (typical input size range from 1% to 10%), we use it as a decompressor in our algorithm. Figure 2 presents the flow chart of our algorithm. The DEM firstly undergoes a points selection (See section 4.3) which pick a subset of posts S as input to ODETLAP solver. Together with the contour lines/border points and some other user supplied points, ODETLAP solver would reconstruct from S the whole DEM matrix of elevations. So this gives us a initial approximation of the elevation matrix. However, due to our pursuit of higher compression ratio, we normally pick a very small subset of points ($|S| = 1000$), consequently the initial approximation normally contains error that is above our needs for accuracy. So after obtaining the initial approximation, unless it accurate enough, (which happens when the original elevation matrix that is mostly flat) some refinement steps are executed. In each step, approximated surface is compared with the original DEM and points that are farthest from the actual ones are picked with care to form a new set S . We assume one point is sufficient to “correct” the points in its neighborhood, thus multiple points in the neighborhood are redundant, points added in the same step are checked against each other to avoid pairs that are too close to each other (for example, less than 5 pixels apart). The refinement steps end when overall RMS error falls below the required accuracy limit.

This process can be easily used for progressive transmission such that the points selection is done in the server end, based on ODETLAP. These points are sent to the client and the terrain is reconstructed in the client using ODETLAP as well. Notice that the client needs to know the value of smoothness parameter R used in the server end.

4.3 Points Selection

Incremental TIN is used as points selection method. It is based on the Franklin’s algorithm [23] which builds a triangulated irregular network (TIN) using a greedy insertion method to approximate a surface. Working in a breadth first search way, incremental TIN starts triangulating by selecting 3 random points and iteratively splits the existing triangles by finding points that are farthest away. This strategy is used to define the initial set for ODETLAP algorithm. That is , an initial set with K points consists of the first K points selected by incremental TIN process.

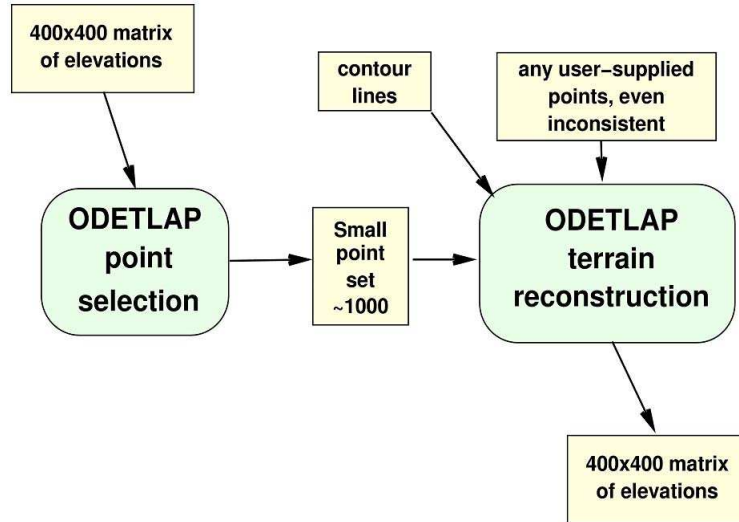


Fig. 2: ODETLAP algorithm: Square box in Bisque are data and curved box in Skyblue are operations.

4.4 Compressing Points

To improve the transmission ratio, the points to be transmitted are compressed using the following strategies: the (x, y, z) coordinates are split into (x, y) pairs and z alone. The former are compressed using an adapted run length encoding method, described below in section 4.5 and the z sequence is compressed using linear prediction and bzip2. This splitting process gives good compression ratio, as shown in 1, because the (x, y) values are restricted to the terrain size, while the z values are more "arbitrary".

4.5 Run Length Encoding

The coordinates (x, y) are different from z because they distribute evenly within the matrix dimensions values, for example from 1 through 400, while z values distribute more closely. The run-length encoding is a simple lossless compression technique which, instead of storing the actual values in the sequence, stores the value and the count of sequence containing the same data value. *Run* is just a consecutive sequence that contains the same data value in each element. Since the (x, y) values correspond to positions in a matrix, then we need to store here is only a binary bitmap showing whether one point is existing in S or not, there is no need to store the actual value. Thus, given a binary matrix of size $N \times N$, the method is the following:

For each run length L , test if

1. $L < 254$, then use one byte for it

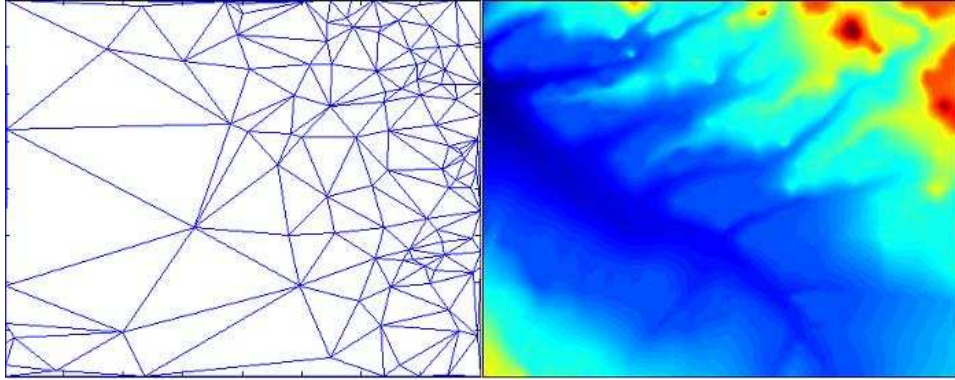


Fig. 3: Triangulated Irregular Network

2. $254 \leq L < 510$, then use FFFE as a marker byte and use a second byte for $L - 254$
3. $510 \leq L < 766$, then use FFFF as a marker byte and use a second byte for $L - 510$
4. $L \geq 766$, then use FFFFFFFF as a two byte marker and use next two bytes for L.

We can see from the histogram of the run length (Figure 4) that most runs are below 512, that means for most runs we need only 1 byte to store it. Here we assume all runs are shorter than 65535, which is a reasonable value for terrains of 400×400 resolution.

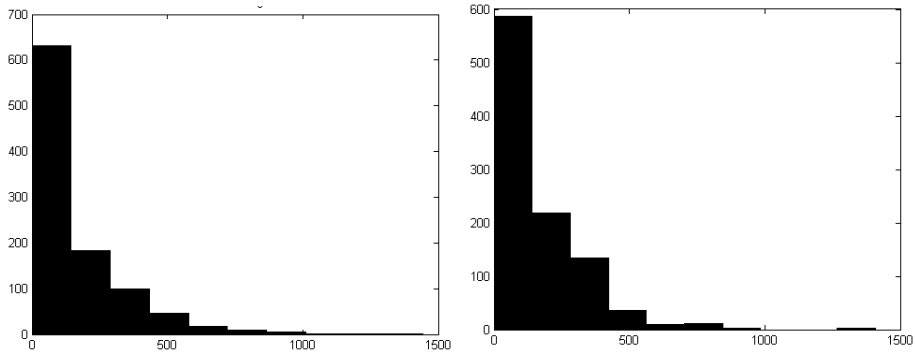


Fig. 4: Histogram of two DEMs, we can see most runs are below 512.

4.6 Linear Prediction

Unlink (x, y) coordinates, z contains more redundancy due to the inherent redundancy in the original terrain. Normally terrain data contains a high degree of correlation and that means we may predict the elevation value from its neighbors. The method of linear prediction has been very successful in image processing [24].

The sequence z that we are going to compress contains elevation information in the selected points by previous mentioned algorithm. Because points appear in the order they were selected by the algorithm mentioned in section 4.2, we need to order them by their corresponding x, y coordinates so that during reconstruction process, the correlation can be reestablished. Linear predictors attempt to identify the redundancy in adjacent points.

There are several different ways to do linear prediction, and within JPEG standard, there are seven modes of prediction [25]. However, most of them use neighborhood information from two dimensions which in our case do not exist. As a result, we are only using the simplest mode which predicts the next entry in the sequence by the previous one.

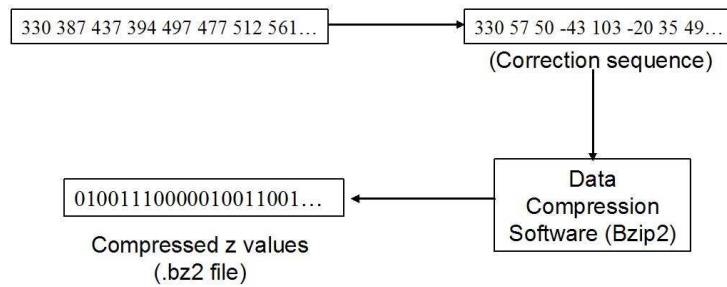


Fig. 5: Compressing z values using linear prediction.

The whole procedure is shown in Figure 5. Given the original data sequence, predicted sequence is computed by predicting each element by the previous one. Their difference is stored in a new correction sequence, which will be compressed by some data compression software like bzip2.

The compressed size is given in table 1. We compress output (x, y, z) triplets from ODETLAP using runlength for (x, y) linear prediction for z . We also compress the same points using bzip2 for comparison. The number of points is also listed. The six test data are rendered in 6 for reference. They are all 400 by 400 DEM datasets. As we described in sections 4.6 and 4.5, the points are separately processing and compressed using run-length encoding and linear prediction. The results shows that our compression method is effective and, in all circumstances it has a smaller size than bzip2.

Data	Initial			75%RMS			50%RMS			Final		
	RLE			RLE			RLE			RLE		
	+LP	Bzip2	# Pts	+LP	Bzip2	# Pts	+LP	Bzip2	# Pts	+LP	Bzip2	# Pts
hill1	625	789	160	658	839	170	799	1003	210	1227	1733	390
hill2	718	817	160	811	948	190	1092	1367	290	2389	3837	900
hill3	594	746	160	685	875	190	762	987	220	762	987	220
mtn1	719	820	160	887	1042	210	1374	1882	400	5172	8828	2150
mtn2	702	810	160	904	1091	220	1348	1836	390	5071	8858	2130
mtn3	766	840	160	844	974	190	993	1198	240	5001	8502	2050

Table 1: Progressive Transmission: Compressed size of points sent are shown, with the corresponding error given as the column title. Algorithm ends when RMS error of the reconstructed surface < 10 . Compressed results of our method and Bzip2 are compared, RLE+LP means Runlength encoding for (x, y) and linear prediction for z .

4.7 ODETLAP Performance Analysis

Table 4.7 gives a summary of our ODETLAP algorithm’s compression performance. Thumbnails of these datasets are shown in Figure. 6. The table shows some information about each terrain, including elevation range, size and standard deviation of the original terrain data as well as the test results. For each dataset, we run a series of tests with different number of starting points, ranging from 400 to 4000. When refinement is needed, 10 percent of points are added in 10 iterations. After that we find the minimum number of points needed to get elevation Root-Mean-Square error below 10. We see in order to control RMS error, we need more points when standard deviation is relatively large (> 150).

	Hill1	Hill2	Hill3	Mtn1	Mtn2	Mtn3
Elev range	505m	745m	500m	1040m	953m	788m
Orig size	320KB	320KB	320KB	320KB	320KB	320KB
Stand. Dev.	78.9	134.4	59.3	146.0	152.4	160.7
Compr. size	2984B	5358B	1739B	9744B	9670B	9895B
Compr. ratio	107:1	60:1	184:1	33:1	33:1	32:1
# pts selected	1040	2080	520	4160	4160	4160
Elev. RMS error	8.49	9.93	8.31	9.48	9.55	9.68
Elev. RMS perc.	1.68%	1.33%	1.66%	0.91%	1%	1.23%
Slope RMS error	2.81°	5°	1.65°	8.34°	8.36°	7.87°

Table 2: ODETLAP Compression results: Compress 3 hilly and 3 mountainous data sets using ODETLAP algorithm and further compression mentioned in section 4.4. Root-Mean-Square elevation and slope errors are recorded.

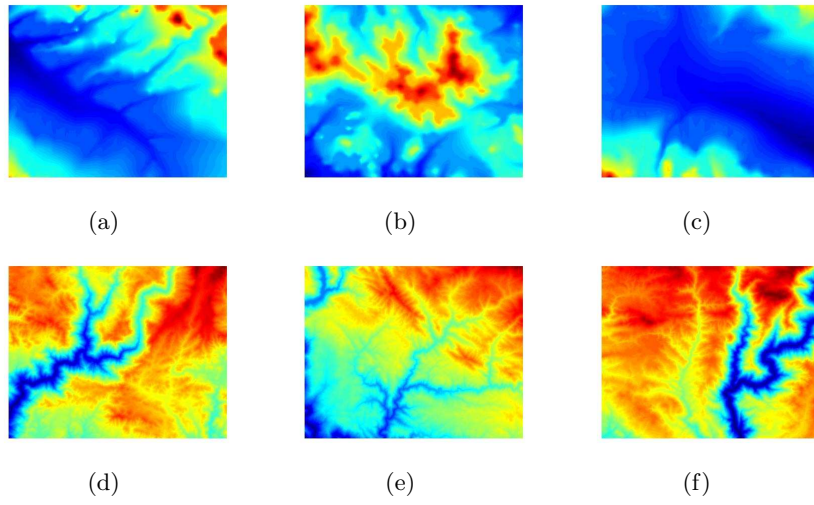


Fig. 6: All six 16-bits original terrain: 400 by 400 resolution

5 Tests and Results

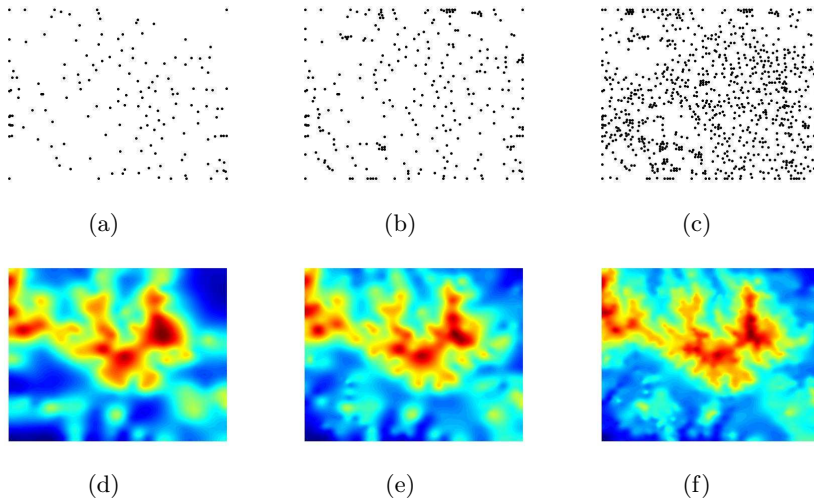


Fig. 7: Progressive example: Hilly datasets(hill2): Compressed sizes are 1.2KB, 3.5KB and 7.1KB

We have tested our algorithm on a benchmarks of six real DEMs shown in Figure. 6. Each DEM is on a 400×400 grid, with spacing of 30 meters. First three (Sub-figure (a),(b) and (c) in Figure. 6) are hilly data sets while the last three are mountainous. We pick one from each kind and plot the points transmitted and render the reconstructed terrain in Figure. 7 and Figure. 8. Information such as compressed sizes and Root-Mean-Square error are also given in the figures.

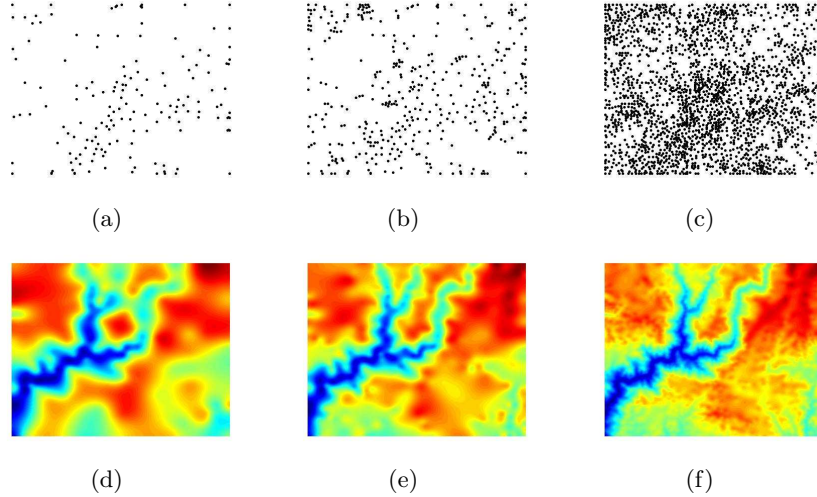


Fig. 8: Progressive example: Mountainous datasets(Mtn1): Compressed sizes are 1.3KB, 3.6KB and 7.3KB

In Table 1, we give the number of points, compressed size in different stages of the progressive transmission. The process begins with 160 initial points selected by TIN, points are selected afterwards according to the absolute elevation error in current reconstructed surface. In order to show the amount of packets sent in different stages of the whole process, we monitor the RMS error and record the points when it falls below 75%, 50% and 25% (Due to the page width limit and the similarity between results from 25% *RMS* and final result, we omit the column of 25%). The whole process ends when RMS error is smaller than 10. The sizes after further compression are recorded. The Table 1 values are also plotted in Figure 9.

We also show in Figure 10 the relationship between compressed size and RMS elevation error of the surface reconstructed by ODETALP. The compressed size corresponds to Table 1, and the RMS error of the reconstructed surface is shown in y axis. We see that when 1KB of compressed packet are sent, the RMS error would drop by almost 50% and when 2KB more are sent, the RMS error will be smaller than 20.

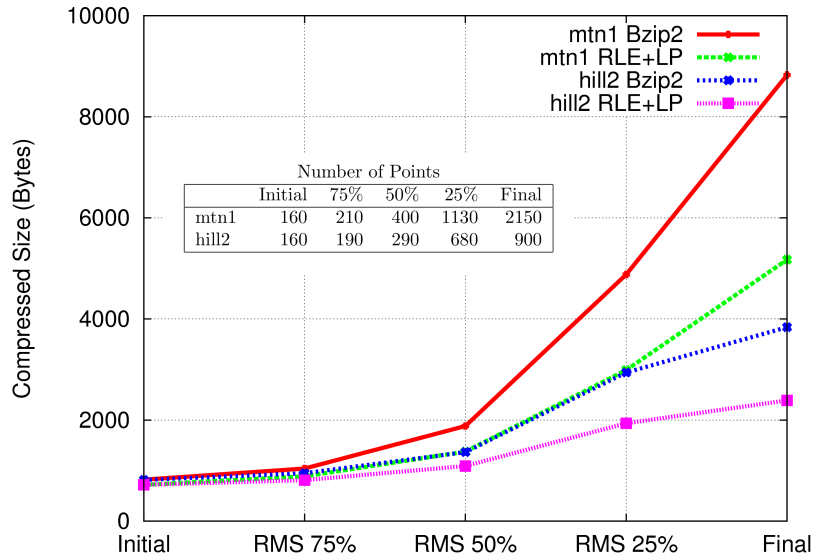


Fig. 9: Compressed size at different stages of compression for both our method and bzip2. We list 5 stages: when initial points are selected, when RMS error drops below 75%, 50%, 25% of the initial RMS error for the first time and when RMS error is smaller than 10.

6 Conclusion and Future Work

In this paper, we presented a progressive transmission method based on an over-determined Laplacian PDE (ODETLAP) terrain representation model. Using our ODETLAP method, it is possible to achieve a lossy compression with compressed size of 1% to 3% of the original binary file while keeping a reasonable error. This is highly useful when compression ratio is emphasized over accuracy. Based on the lossy compression technique, the progressive transmission scheme improves the performance of terrain image transmission. Since the data being transmitted is no longer the whole terrain but a small subset of points, it is possible to achieve greater transmission throughput. In addition, we implemented several methods to further compress the points set so that higher compression ratio is achieved.

Acknowledgments. This research was supported by NSF grants CCR-0306502 and DMS-0327634, by DARPA/DSO/GeoStar, and by CNPq - the Brazilian Council of Technological and Scientific Development. We thank Prof. Franklin T. Luk and Jared Stookey for valuable discussions on terrain representation.

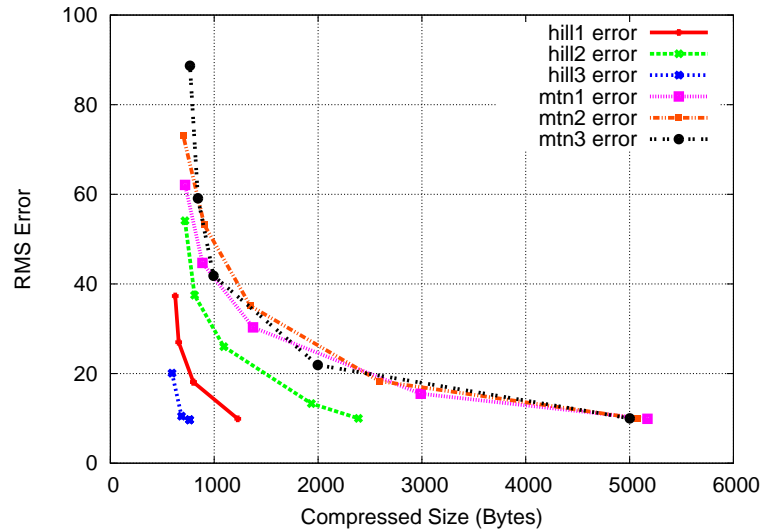


Fig. 10: Root-Mean-Square error of reconstructed surface is plotted against the size of compressed points. As before, we use run length encoding and linear prediction to do the compression.

References

1. D. B. Kidner and D. H. Smith, "Advances in the data compression of digital elevation models," in *Computers & Geosciences*, **29**, pp. 985–1002, October 2003.
2. T. Frajka, P. Sherwood, and K. Zeger, "Progressive image coding with spatially variable resolution," in *ICIP97*, pp. I: 53–56, 1997.
3. M. Rabbani and R. Porcellio, "Image transmission system with line averaging preview mode using two-pass block-edge interpolation," in *US Patent 4764805*, 1988.
4. K. L. Chung and S. Y. Tseng, "New progressive image transmission based on quadtree and shading approach with resolution control," *Pattern Recognition Letters* **22**, pp. 1545–1555, December 2001.
5. Y. C. Hu and J. Jiang, "Low-complexity progressive image transmission scheme based on quadtree segmentation," *Real-Time Imaging* **11**, pp. 59–70, February 2005.
6. B. Zeng, M. S. Fu, and C. C. Chuang, "New interleaved hierarchical interpolation with median-based interpolators for progressive image transmission," *Signal Processing* **81**, pp. 431–438, February 2001.
7. D. I. Azuma, D. N. Wood, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle, "View-dependent refinement of multiresolution meshes with subdivision connectivity," in *AFRIGRAPH '03: Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*, pp. 69–78, ACM, (New York, NY, USA), 2003.
8. G. Davis and A. Nosratinia, "Wavelet-based image coding: An overview," *Applied and Computational Control, Signals, and Circuits* **1**(1), pp. 91–98, 1998.

9. Y. Cho, W. A. Pearlman, and A. Said, "Low complexity resolution progressive image coding algorithm: Progres (progressive resolution decompression)," in *IEEE International Conference on Image Processing*, pp. 49–52, 2005.
10. K. Munadi, M. Kurosaki, K. Nishikawa, and H. Kiya, "Efficient packet loss protection for jpeg2000 images enabling backward compatibility with a standard decoder," *IEEE International Conference on Image Processing*, pp. 833–836, 2004.
11. O. Watanabe and H. Kiya, "Roi-based scalability for progressive transmission in jpeg2000 coding," *IEEE International Symposium on Circuits and Systems*, pp. 416–419, 2003.
12. L. Cao, "On the unequal error protection for progressive image transmission," *Image Processing* **16**, pp. 2384–2388, September 2007.
13. W. Evans, D. Kirkpatrick, and G. Townsend, "Right triangular irregular networks," *Algorithmica* (30), pp. 264–286, 2001.
14. L. D. Floriani, P. Marzano, and E. Puppo, "Multiresolution models for topographic surface description," *The Visual Computer* **12**(7), pp. 317–345, 1996.
15. B. Speckmann and J. Snoeyink, "Easy triangle strips for tin terrain models," *International Journal of Geographical Information Science* **15**(4), pp. 379–386, 2001.
16. L. D. Floriani, P. Magillo, and E. Puppo, "Compressing triangulated irregular networks," *GeoInformatica* **4**(1), pp. 67–88, 2000.
17. B. Jünger and J. Snoeyink, "Selecting independent vertices for terrain simplification," *Proceedings WSCG'98*, (Plzen, Czech Republic), 1998.
18. G. Al-Regib, Y. Altunbasak, and J. Rossignac, "An unequal error protection method for progressively transmitted 3d models," *IEEE Transactions on Multimedia* **7**(4), pp. 766–776, 2005.
19. M. J. E. M. Bertolotto, "Progressive transmission of vector map data over the world wide web," *GeoInformatica* **5**(4), pp. 345–373, 2001.
20. H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," in *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, p. 454, IEEE Computer Society, (Washington, DC, USA), 2000.
21. V. Natarajan and H. Edelsbrunner, "Simplification of threedimensional density maps," *IEEE Transactions on Visualization and Computer Graphics* **10**(5), pp. 587–597, 2004.
22. G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Trans. Graphics* **17**(2), pp. 84–115, 1998.
23. W. R. Franklin, "Triangulated irregular network, ." <ftp://ftp.cs.rpi.edu/pub/franklin/tin73.tar.gz>, (accessed 13 March 2008), 1973.
24. R. M. R. Maragos, P.; Schafer, "Two-dimensional linear prediction and its application to adaptive predictive coding of images," *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing]*, *IEEE Transactions on* **32**(6), pp. 1213–1229, Dec 1984.
25. G. Wallace, "The jpeg still picture compression standard," in *Communications of the ACM*, **34**(4), pp. 30–44, 1991.