Progressive Transmission of Lossily Compressed Terrain

Zhongyi Xie[1], Marcus A. Andrade[1,2], W Randolph Franklin[1], Barbara Cutler[1], Metin Inanc[1], Jonathan Muckell[1], and Daniel M. Tracy[1]

[1] Rensselaer Polytechnic Institute, Troy, NY, USA
[2] Universidade Federal de Viçosa, Viçosa, Brazil
{xiez,frankwr,inancm,muckej,tracyd}@rpi.edu
cutler@cs.rpi.edu,marcus@dpi.ufv.br

Abstract. The distribution and management of spatial data require strategies for handling large amount of terrain data that are now available. Especially, data like LIDAR and Digital Elevation Model (DEM) which have been used in a large group of diversified users. In this paper, we propose a progressive terrain data transmission scheme based on the 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 using run length encoding as well as linear prediction we can reach a higher compression ratio. In general, this technique is capable of reducing a hilly DEM dataset to 1% of its original binary size and a mountainous, to 3%. The accuracy loss in elevation and slope are also discussed.

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

1 Introduction

In the past few years the geographical information science (GIS) community has seen an explosion in the data volume and a great improvement in the accuracy of data acquisition. In particular, a huge volume of data about terrains is available usually represented as a Digital Elevation Models (DEM) that consists of a regular grid of samples storing the height value of the terrain surface. High-resolution DEMs (sampled at 3m or less) is now widely available [1] which make possible more realistic rendering of terrain features. Also, we see a great growth in mobile GIS services as GPS units.

This huge amount of data requires some special techniques to manipulate and/or transmit it and a strategy frequently used is to transmit a large terrain (or image) progressively. That is, the data is transmitted in successive blocks starting with a coarse simplification which is progressively refined sending more information (for example, more points or pixels). Thus, the receiver can visualize or execute some operations in advance, before receiving the whole data.

In this paper we describe a new method to transmit a terrain progressively based on ODETLAP (Over-determined Laplacian Approximation) [2] which is a method to reconstruct the terrain from a subset of points. As shown in section 5, the main advantage of this method is that it allows the terrain reconstruction using a very small number of points which is very interesting for progressive transmission since the user can start the terrain visualization or processing too early. In section 3, we present the progressive transmission method and in section 4, we give a short description of ODETLAP.

2 Related works

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 complete the image by adding pixels [3, 4]. Other more sophisticated strategies [5–7] 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 methods, for example [8–13] 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 sub-sampled 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 [14, 15] is used.

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). Consequently, conventional transmission methods' response time would inevitably make them far more than being interactive. So, this problem can be circumvented using progressive transmission where 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 advance. 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 can save the transmission time and resource need and increase the flexibility and efficiency of the system.

The method proposed in this paper is based on ODETLAP, a method to reconstruct the terrain from a subset of points (see section 4), and the 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 early. When more points arrive at the client side, it can reconstruct the terrain more accurately from the extended points set, which gives to the user a more detailed view of the terrain.

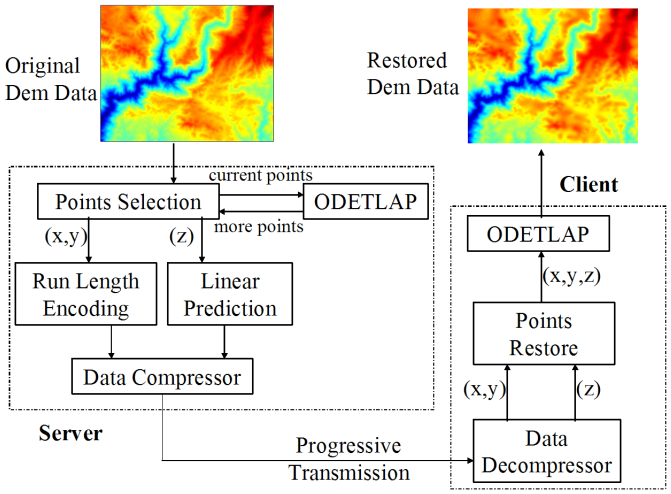

Fig. 1. Progressive transmission flowchart

4 Over-determined Laplacian Approximation

This section presents a short description of the Over-Determined Laplacian Approximation (ODETLAP); for more details see [2].

4.1 Definition

The ODETLAP is an extension to Laplacian equation

$$4z_{xy} = z_{x-1,y} + z_{x+1,y} + z_{x,y-1} + z_{x,y+1} \tag{1}$$

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. This equation has the limitation of being unable to represent local maxima in terrain modeling, which is the reason for the inclusion of the equation:

$$z_{xy} = h_{xy} \tag{2}$$

where h_{xy} is the actual elevation value.

Equations (1) and (2) form a basis for the over-determined linear system. The system's input is a series of points (x, y, z) indicating 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 used. 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 a interpolation parameter R and outputs the DEM matrix of the complete terrain. The benefits of ODETLAP includes 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% of the original points), we can use it as a decompressor. Figure 2 presents the flow chart of our algorithm. The DEM firstly undergoes a points selection 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 ($|S| = 1000$) and consequently the error in the initial approximation normally is above the predefined accuracy. Thus, after obtaining the initial approximation, unless it is accurate enough (which happens when the original elevation matrix 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. That is, multiple points in the neighborhood are redundant and thus, points added in the same step are checked against each other to avoid pairs that are too close (for example, less than 5 pixels apart). The refinement steps end when overall RMS error falls below the required accuracy limit.

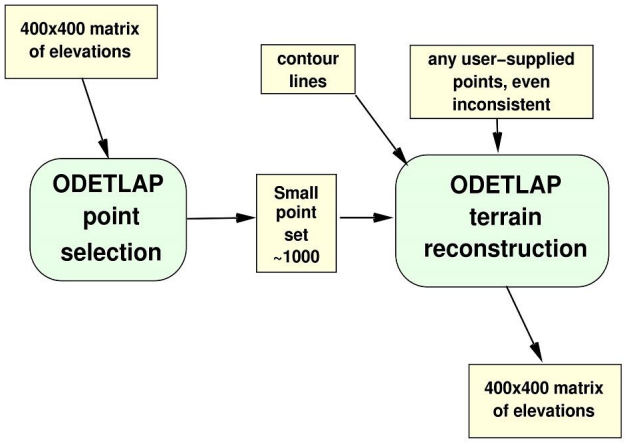

Fig. 2. ODETLAP algorithm: square box indicates data and curved box, operations.

This process can be easily adapted for progressive transmission doing the points selection in the server end, based on ODETLAP, and sending these points to the client where the terrain is reconstructed (also using ODETLAP). Notice that the client needs to know the value of smoothness parameter R used in the server end.

4.3 Points Selection

The initial set of points are selected based on an incremental triangulation using the Franklin's algorithm [16] which builds a triangulated irregular network (TIN) using a greedy insertion method to approximate a surface. The incremental TIN starts triangulating 3 points randomly selected and iteratively splits the existing triangles by finding points that are farthest away. This strategy is used to define an initial set with k points.

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 in section 4.5, and the z sequence is compressed using a linear prediction method and *bzip2* compressor program. This splitting process gives a good compression ratio because the (x, y) values are restricted to the matrix elevation 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 range $[1, N]$, where N is the DEM size, 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 only a binary bitmap showing whether one point is used in S or not. 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
2. $254 <= L < 510$, use FFFE as a marker and use a second byte for $L - 254$
3. $510 <= L < 766$, use FFFF as a marker and use a second byte for $L - 510$
4. $L >= 766$, then use FFFFFFFF as a two byte marker and use next two bytes for L.

In general, 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 size 400×400.

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 predication has been very successful in image processing [17]. However, due to the processing overhead, only recently have such predictor been widely used [18].

The sequence z that we are going to compress consists of the elevation values of the selected points by previous mentioned algorithm. Because points appear in the order they were selected, we need to order them by their corresponding x, y coordinates so that during reconstruction process, the correlation can be reestablished.

There are several different ways to do linear prediction, and within JPEG standard, there are seven modes of prediction [19]. However, most of them use

neighborhood information from two dimensions which in our case does not exist. As a result, we are only using the simplest mode which predicts the next entry in the sequence by the previous one. That is, given the original data sequence, the 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*.

Table 1 presents the results to compress 6 different datasets with 1000 points each one selected from 6 DEMs representing 3 hilly and 3 mountainous terrains (figure 3 shows an image of these terrains). The table includes the total size, in bytes, required by the method (RLE + LP) described above and the size required by *bzip2* to compress the triple (x, y, z). As you can see, the RLE+LP method requires about 60% of the space required by *bzip2*.

	Size (in bytes)					
	Hill1	**Hill2**	**Hill3**	**Mtn1**	**Mtn2**	**Mtn3**
compr. xy (RLE)	1250	1243	1279	1228	1244	1241
compr. z (LP)	1304	1354	1209	1456	1424	1503
RLE+LP	2554	2597	2488	2684	2668	2744
Bzip2	4136	4234	4025	4328	4416	4355

Table 1. Compression of 1000 points: split into xy bitmap and z sequence, then use run-length encoding (RLE) on xy and linear prediction (LP) on z.

To give an idea of the ODETLAP's performance, table 2 shows some information about the compressed representation of the 6 terrains shown in figure 3. For each dataset, a series of tests were run using different number of initial points, ranging from 400 to 4000. When refinement is needed, 10 percent of points are added until to get the elevation Root-Mean-Square error below 10.

The running time of ODETLAP is not real-time, mainly due to the amount of time needed to solve the partial differential equation. Generally speaking, on a laptop with 2GHz and 1GB of RAM, the reconstruction of those 400 by 400 terrains takes about 1 minute.

5 Tests and Results

We have tested our algorithm on a benchmarks of six real DEMs shown in figure 3. Each DEM is on a 400 × 400 grid, with spacing of 30 meters. First three (a),(b) and (c) are hilly datasets while last three are mountainous. Table 3 shows a summary of the results obtained to transmit progressively the 6 terrains in 5 steps: initially, 10% of points are sent to the client; next, for each data set, we selected as many points as necessary to reduce the RMS to 75%, 50% and 25% of its initial value; finally, in the last step, we transmitted points to achieve

	Hill1	**Hill2**	**Hill3**	**Mtn1**	**Mtn2**	**Mtn3**
Elev range (m)	505	745	500	1040	953	788
Stand. Dev. (m)	78.9	134.4	59.3	146.0	152.4	160.7
Orig size (KB)	320	320	320	320	320	320
Compr. size ($Bytes$)	2984	5358	1739	9744	9670	9895
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 (m)	8.49	9.93	8.31	9.48	9.55	9.68
Elev. RMS error (%)	1.68	1.33	1.66	0.91	1.00	1.23
Slope RMS error (°)	2.81	5.00	1.65	8.34	8.36	7.87

Table 2. ODETLAP's compression performance

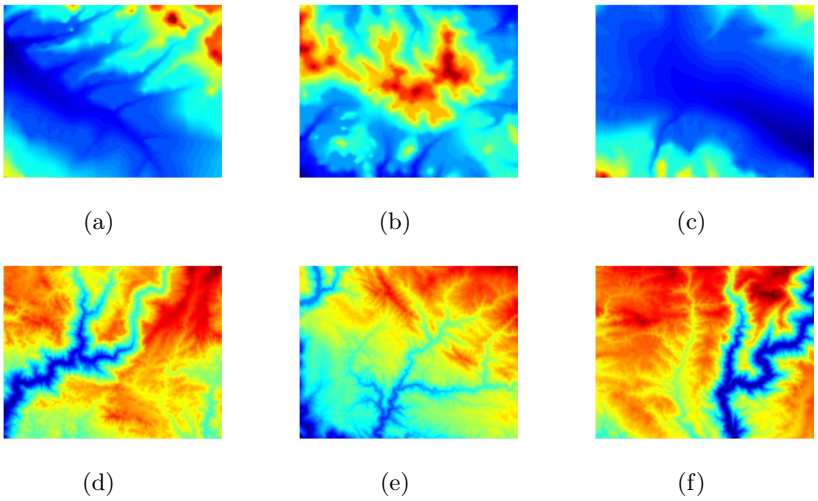

Fig. 3. All six 16-bits original terrain: 400 by 400 resolution

a RMS smaller than 10. The columns #Pts and Bytes show the number of points and the size in bytes in each transmission.

Figure 4 shows some steps of the transmission of the terrain MTN1. In (a), (b) and (c) is shown the selected points and in (d), (e) and (f) the terrain reconstructed using those points and also, the associated RMS error.

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 ODETLAP method, it is possible to achieve a lossy compression whose ratio

Data	Initial		75%RMS		50%RMS		25%RMS		10%RMS	
	#Pts	Bytes	#Pts	Bytes	#Pts	Bytes	#Pts	Bytes	#Pts	Bytes
hill1	160	625	170	658	210	799	210	655	390	1227
hill2	160	718	190	811	290	1092	680	1936	900	2389
hill3	160	594	190	685	220	762	220	424	220	762
mtn1	160	719	210	887	400	1374	113	2989	2150	5172
mtn2	160	702	220	904	390	1348	950	2590	2130	5071
mtn3	160	766	190	844	240	993	660	1997	2050	5001

Table 3. The progressive transmission of 6 terrains in 5 steps: 10% of points, as many points as necessary to reduce the RMS error to 75%, 50% and 25% of the initial value and finally, to achive a RMS error smaller than 10.

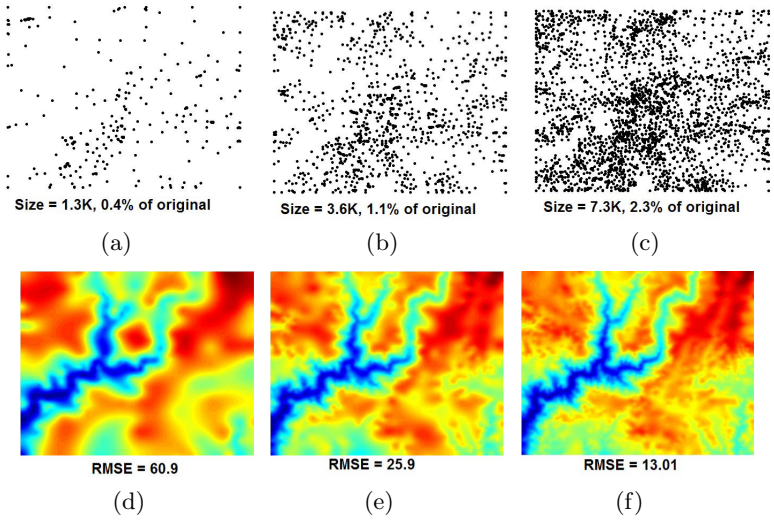

Fig. 4. Progressive example: Mountainous datasets(Mtn1): Compressed sizes are 1.3KB, 3.6KB and 7.3KB

is more than 30:1 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 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.

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.

References

1. USGS, "U. S. Geological Survey." http://www.usgs.gov.
2. W. R. Franklin, M. Inanc, Z. Xie, D. M. Tracy, B. Cutler, M. V. A. Andrade, and F. Luk, "Smugglers and border guards - the geostar project at rpi," in *15th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS 2007*, (Seattle, WA, USA), Nov. 2007.
3. T. Frajka, P. Sherwood, and K. Zeger, "Progressive image coding with spatially variable resolution," in *ICIP97*, pp. I: 53–56, 1997.
4. M. Rabbani and R. Porcellio, "Image transmission system with line averaging preview mode using two-pass block-edge interpolation," in *US_Patent 4764805*, 1988.
5. 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.
6. 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.
7. 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.
8. D. Azuma, D. Wood, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle, "View-dependent refinement of multiresolution meshes with subdivision connectivity," in *Proc. of AFRIGRAPH 2003*, pp. 69–78, ACM Press, 2003.
9. G. Davis and A. Nosratinia, "Wavelet-based image coding: An overview," *Applied and Computational Control, Signals, and Circuits* **1**(1), pp. 91–98, 1998.
10. 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.
11. 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.
12. 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.
13. L. Cao, "On the unequal error protection for progressive image transmission," *Image Processing* **16**, pp. 2384–2388, September 2007.
14. V. Natarajan and H. Edelsbrunner, "Simplification of threedimensional density maps," *IEEE Transactions on Visualization and Computer Graphics* **10**(5), pp. 587–597, 2004.
15. G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Trans. Graphics* **17**(2), pp. 84–115, 1998.
16. W. R. Franklin, "Triangulated irregular network, ftp://ftp.cs.rpi.edu/pub/franklin/tin73.tar.gz.," 1973.
17. P. Maragos, R. Schafer, and R. Mersereau, "Two-dimensional linear prediction and its application to adaptive predictive coding of images," *IEEE Transactions on Signal Processing* **32**(6), pp. 1213–1229, Dec 1984.
18. 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.
19. G. Wallace, "The jpeg still picture compression standard," in *Communications of the ACM*, **34**(4), pp. 30–44, 1991.