Segmented ODETLAP Compression

Wenli Li, W. Randolph Franklin, Salles V. G. Magalhães
Rensselaer Polytechnic Institute, Troy, NY, USA
liw9@rpi.edu, mail@wrfranklin.org, vianas2@rpi.edu

ABSTRACT

We have designed an algorithm for segmented ODETLAP compression. It can be more than 3 times as fast as unsegmented compression and use less than 10% more points for better compression errors. When hardware is available, it can process segments in parallel.

1. INTRODUCTION

ODETLAP is a discrete space approximation and compression method created by Franklin [1]. ODETLAP has two components: approximation and lossy compression. For approximation, it estimates grid point values from a subset of known points by building and solving an overdetermined system of linear equations. The equations include an averaging equation for each point and an additional known-value equation for each known point. The basic format of the averaging equation is the finite-difference approximation of the Laplace's equation, which makes the value of each point (x, y) equal to the average value of its neighbors:

$$u(x-1,y)+u(x+1,y)+u(x,y-1)+u(x,y+1)-4u(x,y)=0.$$

Multiplying both sides of the equation with a positive parameter R gives

$$R(u(x-1,y)+u(x+1,y)+u(x,y-1)+u(x,y+1)-4u(x,y))=0.$$

R does not change the equation but changes its weight relative to other equations. The known-value equation for each known point sets its value to its known value:

$$u(x,y)=f(x,y).$$

The last two equations constitute an overdetermined system of more equations than unknowns, whose approximate solution assigns a value to each point of the domain. The approximate value of a point is not exactly the average of its neighbors, and the approximate value of a known point is not exactly its known value. R controls the smoothness of the approximation by specifying the weight of averaging equations relative to known-value equations. Therefore, R is called the smoothing factor.

The other component of ODETLAP is lossy data compression. The method is to select an important subset of data points S from a dataset as its compressed representation, and decompress it by interpolating S on a grid using ODETLAP approximation. Xie [5] used a greedy method to select important points from a DEM. It first builds an initial point set S, for example, using the vertices of a TIN, and reconstructs a surface using ODETLAP approximation. Then it iteratively adds a number of points with the greatest absolute vertical errors to S, and reconstructs a new surface from S. The process stops when the root-mean-square error of the approximation is not more than a predefined threshold. To prevent the points added in the same iteration from clustering, a forbidden zone is applied to each new point so that additional new points are at least some distance apart from it.

To accelerate ODETLAP approximation, Stookey [4] parallelized it on an IBM Blue Gene/L by dividing a grid into overlapping patches. For example, to approximate a terrain, the method divides the grid and known points into overlapping patches of size 100×100, whose lower left corners are at $(50i, 50j)$, $i, j = 0, 1, \ldots$ Then it computes an approximation for each patch and merges the results using bilinear interpolation. The patches are grouped into blocks that are processed in parallel. To save time and memory for ODETLAP approximation, Li[2] used a method that divides a grid into two overlapping sets of boxes. Then it computes an approximation for each box and merges the results from the two sets using weighted average.

Table 1: Unsegmented results

Points	Time	AVGEE	RMSEE	MAXEE
6386	52m26s	12.0	15.6	50.0

Mitášová and Mitáš [3] developed a segmentation procedure for the interpolation of large datasets using completely regularized splines. The method is based on the local behavior of the interpolation function. It divides a grid into square segments so that the number of known points in each segment and its neighborhood is less than a threshold. Then it computes an interpolation for each segment from the points in its neighborhood of 3×3, 5×5 or more segments, so that the number of points is more than a threshold.

2. SEGMENTED ODETLAP COMPRESSION

Solving large linear systems is time-consuming but can be accelerated by parallel processing on GPU. We implemented ODETLAP approximation using the Cusp library, which is based on the Thrust library. On our server, the speedup of ODETLAP approximation is about 8 times using an NVIDIA Tesla K20Xm, over using a single thread of an Intel Xeon E5-2687W. With GPU-accelerated ODETLAP approximation, we can afford to process bigger datasets, and to add one point in each iteration of the greedy point selection method. However, it is still a time-consuming process. For example, Figure 1 shows a 600×600 DEM down-sampled from NED 1×1 degree block n43w074. Point values are in integer meters and have a range of $[-1, 1138]$.

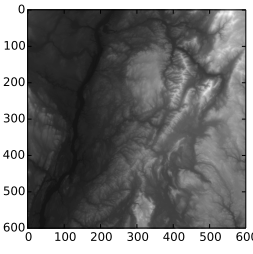

Figure 1: Sample dataset.

Table 1 shows the number of selected points, running time and compression errors of adding one point per iteration to an initial set of 40×40 regular points at positions $(15i + 7, 15j + 7)$, until the maximum absolute elevation error (MAXEE) is less than 50 meters. The other errors are average absolute elevation error (AVGEE) and root-mean-square elevation error (RMSEE) in meters. The smoothing factor of ODETLAP approximation is $R = 0.01$.

ODETLAP approximation shows local behavior in that the influence of a known point decreases with the increase of

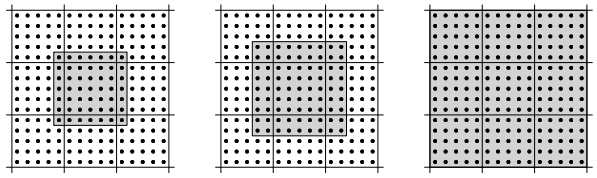

Figure 2: Neighborhood types.

distance. In stead of segmented approximation, we designed segmented compression such that adding a point only requires an approximation for a single segment.

2.1 Algorithm

The main idea of the algorithm is to divide an $nrows \times ncols$ dataset into $\frac{nrows}{SS} \times \frac{ncols}{SS}$ segments of size $SS \times SS$, and then compress each segment in association with neighboring segments. Given a segmented dataset, it selects points from each segment until the maximum absolute approximation error of each segment is less than a threshold. The approximation of a segment is computed as part of the approximation of its neighborhood. A segment still needs processing if either its maximum approximation error is not less than the threshold, or new points are selected in its neighborhood from other segments. When a point is selected, all segments whose neighborhood contains it will need processing. To maintain uniform progress among all segments, they are processed in a round-robin manner. The details are shown in Algorithm 1.

Algorithm 1: Segmented compression

```
Data: a segmented dataset
Result: a set S of selected points
add the center of each segment to S;
mark all segments as needing processing;
while there are segments that need processing do
    for each segment s that needs processing in an order do
        while s needs processing for up to a number of
        iterations do
            compute the ODETLAP approximation of s's
            neighborhood;
            if the maximum absolute error in s is less than a
            threshold then
                mark s as not needing processing;
            else
                add the worst point p in s to S;
            for each other segment t in s's neighborhood do
                if p is in t's neighborhood then
                    mark t as needing processing;
```

The parameters are SS: segment size; NT: neighborhood type; $MAXITER$: the maximum number of iterations in processing a segment; and $ORDER$: the order of processing the segments that need processing.

2.2 Experiments

We considered three neighborhood types: (a) one extra point wide, (b) $\frac{SS}{2}$ extra points wide, and (c) one extra segment wide. Figure 2 shows each type of neighborhood as a gray box when $SS = 5$.

In experiments, we set $SS = 15$ and the maximum approximation error threshold of each segment to 50. The initial point set consists of the center of each segment, or 40×40 regular points. The smoothing factor of ODETLAP approximation $R = 0.01$. Table 2 shows the results of the algorithm using different NT's, with $MAXITER = 1$ and $ORDER$ being row-column order. The approximation of a dataset consists of the approximation of each segment. The table also shows the inflation of the number of selected points and the speedup of running time. As neighborhood size increases from (a) to (c), the inflation, speedup and errors all decrease.

Table 2: Segmented results (NT)

NT	Points	Inflation	Time	Speedup	AVGEE	RMSEE	MAXEE
(a)	8071	1.26×	6m45s	7.76×	12.0	15.6	50.0
(b)	7483	1.17×	9m26s	5.56×	11.8	15.3	50.0
(c)	7014	1.10×	20m14s	2.59×	11.8	15.3	50.0

Table 3: Segmented results ($MAXITER$)

$M.I.$	Points	Inflation	Time	Speedup	AVGEE	RMSEE	MAXEE
3	6915	1.08×	16m40s	3.15×	11.8	15.3	49.9
4	6898	1.08×	15m53s	3.30×	11.8	15.3	50.0
5	6888	1.08×	15m40s	3.35×	11.9	15.4	50.0

Table 3 shows the results using different $MAXITER$'s, with NT being (c) and $ORDER$ being row-column order. In general, as $MAXITER$ increases, the speedup decreases but converges quickly, while the other results are similar.

Table 4 shows the results with NT being (c), $MAXITER = 4$ and $ORDER$ being random order. Random order is slightly faster than row-column order.

3. CONCLUSIONS

We have designed segmented ODETLAP compression. For the sample dataset and a maximum error of 50, it is more than 3 times as fast as unsegmented compression. Average and RMS errors are sightly better, but the number of selected points is about 7% larger. Because a GPU is more efficient with a larger problem size, the speedup is greater if ODETLAP approximation is on CPU. The algorithm also works better for more unbalanced datasets. Besides, when hardware is available, segments can be processed in parallel.

Acknowledgement. This research was partially supported by NSF grant IIS-1117277 and CAPES (Ciência sem Fronteiras).

4. REFERENCES

[1] W. R. Franklin and M. Gousie. Terrain elevation data structure operations. In C. P. Keller, editor, *Proceedings of the 19th International Cartographic Conference*, pages 1011–1020, Aug. 1999.

[2] Y. Li. *CUDA-accelerated HD-ODETLAP: a high dimensional geospatial data compression framework*. PhD thesis, Rensselaer Polytechnic Institute, 2011.

[3] H. Mitášová and L. Mitáš. Interpolation by regularized spline with tension: I. theory and implementation. *Mathematical Geology*, 25(6):641–655, 1993.

[4] J. Stookey. Parallel terrain compression and reconstruction. Master's thesis, Rensselaer Polytechnic Institute, 2008.

[5] Z. Xie. Representation, compression and progressive transmission of digital terrain data using over-determined laplacian partial differential equations. Master's thesis, Rensselaer Polytechnic Institute, 2008.

Table 4: Segmented results ($ORDER$)

Points	Inflation	Time	Speedup	AVGEE	RMSEE	MAXEE
6821	1.07×	15m28s	3.39×	11.9	15.4	50.0