PhD Showcase: 3D Oceanographic Data Compression Using 3D-ODETLAP

PhD Student:You Li
Rensselaer Polytechnic Institute (RPI)
110, Eighth Street
Troy, NY, USA
liy13@cs.rpi.edu

PhD Student:Tsz-Yam Lau
RPI
110, Eighth Street
Troy, NY, USA
laut@cs.rpi.edu

PhD Student:Christopher S. Stuetzle
RPI
110, Eighth Street
Troy, NY, USA
stuetc@cs.rpi.edu

PhD Supervisor:Peter Fox
RPI
110, Eighth Street
Troy, NY, USA
pfox@cs.rpi.edu

PhD Supervisor:W. Randolph Franklin
RPI
110, Eighth Street
Troy, NY, USA
wrf@ecse.rpi.edu

ABSTRACT

This paper describes a 3D environmental data compression technique for oceanographic datasets. With proper point selection, our method approximates uncompressed marine data using an over-determined system of linear equations based on (but essentially different from) the Laplacian partial differential equation. Then this approximation is refined via an error metric. These two steps work alternatively until a predefined satisfying approximation is found.

Using several different datasets and metrics, we demonstrate that our method has an excellent compression ratio. To further evaluate our method, we compare it with 3D-SPIHT. 3D-ODETLAP averages 20% better compression than 3D-SPIHT on our eight test datasets, from World Ocean Atlas 2005. Our method provides up to approximately six times better compression on datasets with relatively small variance. Meanwhile, with the same approximate mean error, we demonstrate a significantly smaller maximum error compared to 3D-SPIHT and provide a feature to keep the maximum error under a user-defined limit.

Categories and Subject Descriptors

I.3.5 [**Computing Methodologies**]: Computer Graphics Computational Geometry and Object Modeling

General Terms

Algorithm, Experimentation, Performance

Keywords

PDE solver, 3D, Compression, Oceanographic

The SIGSPATIAL Special, Volume 2, Number 3, November 2010.

1. INTRODUCTION

As technology progresses, the availability of massive oceanographic data with global spatial coverage has become quite common. Various types of data, including salinity, temperature and oxygen of sea water, require measurement and storage in 3D, or even 4D over time for historical record. Researchers expect to be able to transmit these data over the internet, exposing them to the public, since the data provide a raw source of information about the environment we are living in.

Nevertheless, the research of processing and manipulating 3D oceanographic data has not advanced with the data inflation. Marine datasets are still stored as 3D value matrices where traditional compression algorithms don't perform well, and it's rare to find specially designed algorithms to compress 3D marine data. For example, Salinity[2] and Nutrients[5] in World Ocean Atlas 2005 are compressed with gzip, which was originally designed for plain text compression. Images have been produced for observation and analysis. Figure 1 visualizes the one original marine dataset [4] derived from WOA 2005.

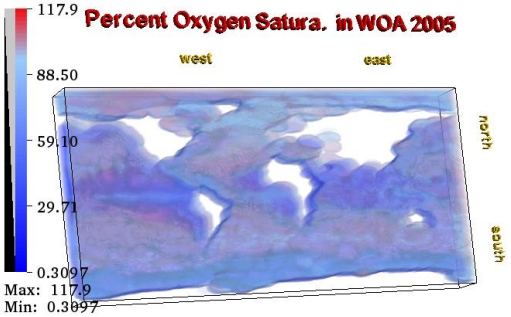

Figure 1: $180 \times 360 \times 33$ **Percentage oxygen saturation data**

In this paper, we use a 3D Over-determined Laplacian Partial Differential Equation (3D-ODETLAP) to approximate and lossily compress 3D marine data. First we construct an over-determined system using regular grid point selection; we then use an over-determined PDE to solve for a smooth approximation. The initial approximation might be

very coarse due to the limited number of selected points. Furthermore, we refine this approximation with respect to the original marine data by adding points with the largest error, and running 3D-ODETLAP again on the augmented representation to gain a better approximation. These two steps work alternately until a stopping criteria is reached, which is usually the maximum error.

2. PRIOR ART

Generally speaking, most of the methods used for 3D environmental scalar data compression are based on image and video compression techniques since they provide good compression with no (or low) loss of information. 2D and 3D wavelet-based methods are among the best ones.

In general, 2D compression techniques such as 2D wavelets decomposition, JPEG compression, and JPEG2000 compression 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. 3D environmental scalar data can be split into 2D image slices. Therefore the above advanced 2D compression techniques can be applied on the slices.

While the above methods are basically applications of the 2D method on 3D data, there are also truly 3D volumetric data compression techniques. Muraki[7] introduced the idea of using a 3D wavelet transformation for approximation and compression of 3D scalar data. Luo et al[6] used a modified embedded zero tree wavelet (EZW) algorithm[11] for compression of volumetric medical images. The EZW-approach has been improved upon by Said and Pearlman[9] with the SPIHT algorithm for images. This approach has been extended to a 3D-SPIHT algorithm which was used for video coding and with excellent results also for medical volumetric data[14] using an integer wavelet packet transform. A compression performance comparison has been conducted between 3D-SPIHT and our proposed 3D-ODETLAP method later in this paper.

3. 3D-ODETLAP

3.1 Definition

As implied by the name, 3D-ODETLAP, or Three Dimensional Over-Determined Laplacian Partial Differential Equation, is an extension of a Laplacian PDE $\frac{\delta^2 z}{\delta x^2} + \frac{\delta^2 z}{\delta y^2} = 0$ to an overdetermined system of equations[12, 13]. Each unknown point induces an equation setting it to the average of its 3, 4, 5 or 6 neighbors in three dimensional space. We have the equation:

$$u_{i,j,k} = (u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i,j+1,k} + u_{i,j,k-1} + u_{i,j,k+1})/6 \quad (1)$$

for every unknown non-border point, which is equivalent to saying the volume satisfies 3D Laplacian PDE,

$$\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \frac{\delta^2 u}{\delta z^2} = 0 \quad (2)$$

In marine modeling this equation has the following limitations:

- The solution of Laplace equation never has a relative maximum or minimum in the interior of the solution domain, this is called the maximum principle[10] so local maxima are never generated.
- For different marine data distribution, the sole solution may not be the optimal one.

To avoid these limitations, an over-determined version of the Laplacian equation is defined as follows: apply the equation 1 to every non-border point, both known and unknown, and a new equation is added for a set S of known points:

$$u_{i,j,k} = h_{i,j,k} \quad (3)$$

where $h_{i,j,k}$ stands for the known value of points in S and $u_{i,j,k}$ is the "computed" value as in equation 1.

Because the number of equations exceeds the number of unknown variables, this means the system is over-determined. Since the system is very likely to be inconsistent, instead of solving it for an exact solution (which is now impossible), an approximated solution is obtained by trying to keep the error as small as possible. Equation 1 is approximately satisfied for each point, making it the average of its neighbors, which makes the generated surface smooth and resembles the real world situation. However, since we have known points where equation 3 is valid, they are not necessarily equal to the average of their neighbors. This is especially true when we have adjacent known points. Therefore, for points with multiple equations we can choose the relative importance of accuracy versus smoothness by adding a smoothness parameter when solving the over-determined system[3].

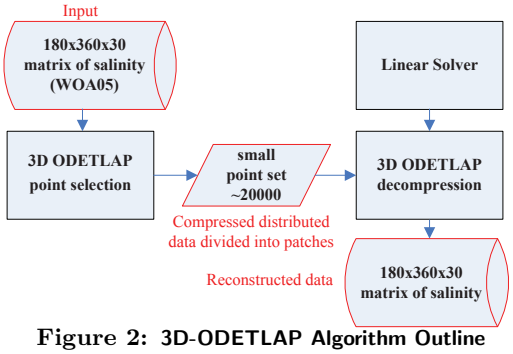

Figure 2: 3D-ODETLAP Algorithm Outline

In our implementation, equation 1 is weighted by R relative to equation 3, which defines the known locations. So a small R will approximate a determined solution and the surface will be more accurate while a very large R will produce a surface with no divergence, effectively ignoring the known points. Instead of interpolation, approximation is a more suitable term for this method because the reconstructed volume is not guaranteed to go through the input data points. 3D-ODETLAP can be used as a lossy compression technique since the original terrain can be approximated with some error using the set of points S for equations 1 and 3.

3.2 Algorithm Outline

Input: $3D - MarineData : V$
Output: $PointSet : S$
$S = RegGridSelection(V)$
$Reconstructed = 3D - ODETLAP(S)$
while $MeanError > Max\_MeanError$ **do**
  $S = S \cup Refine(V, Reconstructed)$
  $Reconstructed = 3D - ODETLAP(V)$
end
return S

Algorithm 1: 3D-ODETLAP algorithm pseudo code

The 3D-ODETLAP algorithm's outline is shown in Figure 2 and the pseudo code is given below. Starting with the

original marine volume data matrix, there are two point selection phases: firstly, the initial point set S is built by a simple regular grid selection scheme and a first approximation is computed using the equations 1 and 3. Given the reconstructed surface, a stopping condition based on an error measure is tested. In practice, we have used the mean percent error as the stopping condition. If this condition is not satisfied, the second step is executed. In this step, k≥1 points with the biggest error are selected based on our "forbidden zone" method described below and then added into the existing point set S; this extended set is used by 3D-ODETLAP to compute a more refined approximation. As the algorithm proceeds, the total size of point set S increases and the total error converges.

3.3 Forbidden Zone

The refined point selection strategy has flaws in that those selected points are often clustered due to high value fluctuation within a small region. In oceanographic datasets, if one point with large error is far away from others, it is most likely that its adjacent points are also erroneous and will be selected as well. Therefore, refined points selected may be redundant in some regions, which is a waste in compression.

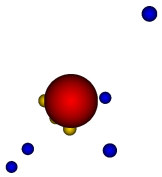

Figure 3: Forbidden Zone check

Forbidden zone is the check process we use to add new refined points: the spatial local neighbors of the new point will be checked to see if there is any existing refined points added in the same iteration. If yes, this new point is abandoned and the point with next biggest error is tested until we have a predefined number of refined points. In Figure 3, the big red sphere is the forbidden zone of one point. The yellow points are inside this sphere and thus not included. All the blue points are outside the zone and included.

3.4 Implementation Speed-up

3.4.1 Normal Equation

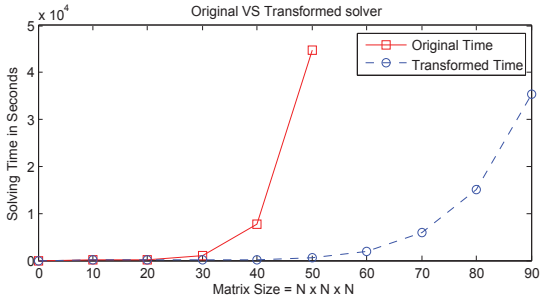

Figure 4: The red line represents the solving time of original solver and the blue line represents that of transformed solver by normal equations

In order to solve the linear system $Ax = b$ when A is non-symmetric, we can solve the equivalent system

$$A^T Ax = A^T b \tag{4}$$

which is Symmetric Positive Definite in our case because in our *over-determined* system, A is a rectangular matrix of size $n \times m$, $m < n$. This system is known as the system of the *normal equations* associated with the least-squares problem,

$$minimize||b - Ax||^2 \tag{5}$$

Before applying the *normal equations* method on our *over-determined sparse linear system*, our underlying solver to solve the over-determined system uses sparse QR decomposition in Matlab. It runs much slower than the Cholesky factorization, which solves Symmetric Positive Definite linear system as shown in Figure 4. Even with the overhead introduced by matrix multiplication, the *normal equations* method still solves our linear system significantly faster. The *normal equations* method enables us to solve small linear system very quickly, which makes our next scheme *Dividing into Boxes* feasible.

3.4.2 Dividing into Boxes

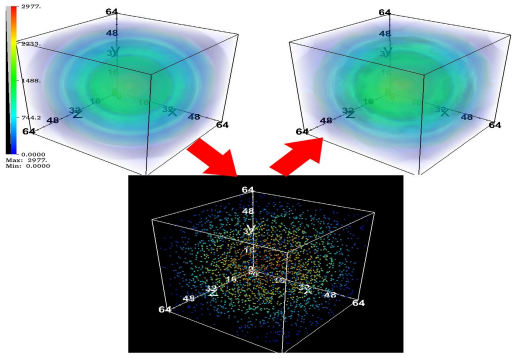

Figure 5: Reconstruction from 1% random selected points using weighted average method

As you can see from Figure 4, solving a reasonable large 3D matrix, i.e., $100 \times 100 \times 100$, still takes too much time. Linear regression test shows that 3D-ODETLAP is running approximately at $T = \Theta(n^6)$ for an $n \times n \times n$ matrix. And the memory cost is also high: solving a $90 \times 90 \times 90$ 3D dataset takes about 9.8 CPU-hours and 55 GB of writable memory on four 2.4GHZ processors workstation with 60 GB of main memory running Ubuntu 9.04 and 64-bit Matlab R2009a. This is unacceptable because the 3d datasets in the real world is often significantly larger than the test one and the running time may be prohibitively long.

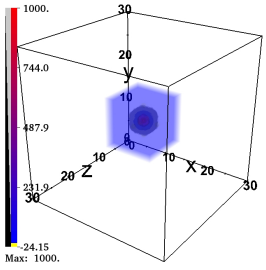

Figure 6: Altering a single known point in the data has a limited radius of impact during reconstruction.

Within datasets, some points are quite distant and thus have nearly no influence on each other. Figure 6 shows that when we assign a high value to one point in the input to 3D-ODETLAP, only those points within a small neighborhood of this point are affected. Beyond that small region, the effect becomes negligible. This supports our hypothesis that it should be possible to divide large data sets into separate boxes, run 3D-ODETLAP on them individually, and achieve similar results to the non-box 3D-ODETLAP solution. Specifically, we created a $64 \times 64 \times 64$

	Size(in bytes)							
	Salinity	Temperature	Dissolved oxygen	Apparent oxygen uti.	Percentage oxygen satura.	Phosphate	Nitrate	Silicate
Compr.xyz(RLE)	5983	3913	3994	4904	5026	5026	6118	4129
Compr.v	21394	17595	17046	19871	21717	19417	25787	18947
compr.(xyz+v)	27377	21508	21040	24775	26743	24493	31905	23076
Bzip2	36733	25687	24082	27829	35649	27658	39844	31018
Size Reduction(%)	34.17%	19.42%	14.45%	12.32%	33.30%	12.92%	24.88%	34.41%

Table 1: File size comparison after compression between Run length encoding method and simply storing quadruplets which contains three coordinates and values. Our RLE method further reduces file size ranging from 12% to 34% as indicated in the fourth row

3D matrix A based on function $A = (max\_distance - distance)^2$, where $max\_distance$ is the maximum distance from any points within the cube to its center and distance is the distance of this point to the center of the cube. Then we divide compressed A into a 4×4 matrix whose elements are $16 \times 16 \times 16$ boxes. Then we run 3D-ODETLAP on each box individually and merge them together as shown in Figure 5.

Nevertheless, if we simply merge all the boxes together, we tend to have large errors at the edge and plane of a box, where there is less information to work with than running 3D-ODETLAP on the matrix as a whole.

Our solution is as follows: run two iterations on these data. First, 3D-ODETLAP is run individually on each box. Then, 3D-ODETLAP is run on the inner $48 \times 48 \times 48$ matrix which is the inner part of matrix A with each box size of $16 \times 16 \times 16$. These two iterations enable us to have two calculated values for each point in the inner $48 \times 48 \times 48$ matrix and only one value for the rest of matrix A. Simply taking the average of both calculated values will reduce the errors at the edges and planes to some extent. But since we run 3D-ODETLAP redundantly on each point, we have the option to bias those erroneous edge and plane points' values and select those values close to the center of another box. In

Variable	Unit	Data Range	Size(Mb)
Salinity	PPS	[5.00, 40.90]	8.16
Temperature	°	[-2.08, 29.78]	8.16
Dissolved oxygen	ml^{-1}	[0, 9.55]	8.16
Apparent oxygen utilization	ml^{-1}	[-1.43, 7.87]	8.16
Percent oxygen saturation	%	[0.31, 117.90]	8.16
Phosphate	μM	[0, 4.93]	8.16
Nitrate	μM	[0, 54.45]	8.16
Silicate	μM	[0, 256.24]	8.16

Table 2: Before compression, These are annual objectively analyzed climatology datasets on 33 standard depth levels from sea surface to 5500 metres on all 8 variables. Each point uses 4 byte to store its value in single precision and thus total size of each dataset is 8.16 Mb.

order to ignore those values of edges and plane points, we use a *Euclidean Distance* to assign weight to the two values of each point. The closer the point is to the center of its box, the more weight its value has and vice versa. Weighted average method produces smaller errors overall than others.

4. FURTHER COMPRESSION

The 3D spatial coordinates *(x,y,z)* are different from value v because they distribute evenly within the range $[1, 128]$, $[1, 128]$ and $[1, 32]$ respectively in our test data. But values v v are distributed more closely and require higher precision. The run-length encoding is a simple lossless compression technique and it stores the value and the count of sequence with the same value. Because the *(x,y,z)* values correspond to positions in a 3D matrix, we only need to store a binary value in each position to indicate whether this point is selected or not. So we define *Run* as a consecutive sequence of 0's or 1's to represent the selection of one point at each location. Thus, given a binary matrix of $N1 \times N2 \times N3$, we use simple run-length encoding to store value of *Run*.

5. RESULT AND ANALYSIS

5.1 Result on Oceanographic Data

We tested 3D-ODETLAP on actual marine data–WOA 2005, which is provided by NODC (National Oceanographic Data Center). WOA 2005 is a set of objectively analyzed climatological fields on a 1-degree latitude-longitude grid at 33 standard depth levels from sea surface to 5500 metres. Forperiods for the World Ocean. These variables include temperature, salinity, oxygen, oxygen saturation, Apparent Oxygen Utilization (AOU), phosphate, silicate and nitrate. Table 2 shows the datasets in detail. In the following experiments, all of our test data are derived from these datasets.

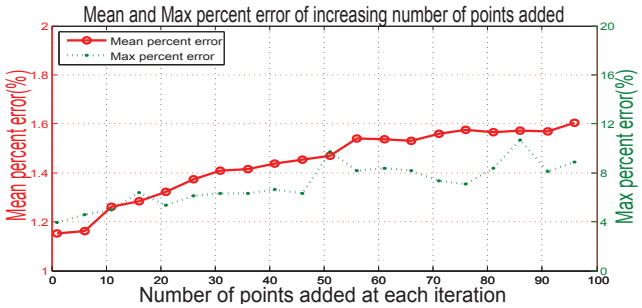

Figure 7: This figure shows the Mean(left) and Max(right) percentage error as the number of points added at each iteration increases from 1 to 96. The dataset is $64 \times 64 \times 32$ 3D matrix derived from Percent Oxygen Saturation in WOA 2005. The number of selected points is 2100, smooth variable R is set to 0.01, sub-box is of size $32 \times 32 \times 32$, initial point selection picks every 5 along the regular grid and forbidden zone is of size 3.

In our implementation, several parameters can affect the effectiveness of our method, including initial point selection, points added at each iteration, smooth variable R and size of forbidden zone. In order to achieve a high compression ratio while keeping the errors beneath a tolerant level, we approach as close as possible the optimal parameters for the all eight datasets by finding the best set of parameters for one dataset. This set of parameters works reasonably

well with other datasets and if given enough computing resources, even better parameters can be found.

Figure 7 shows that as we increase the number of points added at each iteration, both mean error and max error are getting larger. But this doesn't mean we should keep this number to minimum, because there is still a trade-off between computing time and compression ratio with a predefined error.

Besides the above parameters, the size of each sub-box also has an impact on the compression performance. As table 3 shows, the larger the size of the sub-box, the better performance we will gain. However, the complexity of our algorithm is extremely high as shown in Figure 4, so we chose the largest possible sub-box while keeping running time at tolerant level. For our test on WOA 2005 data, sub-box of size $30 \times 30 \times 30$ runs in about 2000 seconds at each iteration on four 2.4GHZ processors workstation with 60 GB of main memory running Ubuntu 9.04 and 64-bit Matlab R2009a. This is acceptable and we used a sub-box of size $30 \times 30 \times 30$ in all our tests, considering that there are only 33 levels of depth in WOA 2005.

Box Size	Mean Error(%)	Max Error(%)	Time (seconds)
$8 \times 8 \times 8$	1.8554	8.5103	150.40
$16 \times 16 \times 16$	1.3219	5.7181	293.90
$32 \times 32 \times 32$	1.1179	5.5175	1250.00

Table 3: This shows the result of running 3D-ODETLAP with different sub-box size on part of Percent Oxygen Saturation data from surface to 4000 metres, which is a 3D matrix of size $128 \times 128 \times 32$. Other parameters remain the same for all three tests.

Based on our implementation, we applied 3D-ODETLAP on the data in 2. This gave us eight distinct datasets in the size of $180 \times 360 \times 30$, which is reasonably large for testing purposes. In table5, we show that with a tolerant absolute percentage mean and max error, 3D-ODETLAP can achieve great compression ratio for all eight test datasets. Furthermore, the Silicate dataset from Table 5 and Table 4 has about the same mean percentage error: 0.9969% and 0.9996% respectively. But the compression ratio on the larger dataset is 165:1 while the one on the smaller dataset is only 81:1. This implies that 3D-ODETLAP may perform even better on larger datasets.

5.2 Compression Comparison with 3D-SPIHT

We have reported on experiments using all eight large oceanographic datasets extracted from WOA 2005. The size of each dataset is $128 \times 128 \times 32$ with each scalar caring a single-precision value, stored in 4 bytes each and resulting in a total size of 2 Mbytes. The same data set was used for compression in our experiments with the 3D-SPIHT method in order to provide an objective comparison of the compression performance of the two algorithms.

The quality of the compression is measured as above in terms of the mean and max percentage error over the range of the current dataset. The results are given in Table 4. Please note that we used bzip2 to further compress our results. Because 3D-SPIHT can't compress its results using bzip2 further, the comparison is fair. From Table 4, we can see that while intentionally keeping mean percentage error at approximately 0.15% for salinity dataset and 1% for the seven other datasets, 3D-ODETLAP generally produces a better compression ratio on nearly all eight different datasets and even better maximum percentage error. Except that 3D-SPIHT performs better on temperature data than 3D-ODETLAP. In fact, we can specify the maximum max error when running 3D-ODETLAP to meet the special needs of some applications, while the 3D-SPIHT compression scheme cannot.

Variable	Mean Error(%)	Max Error(%)	Compress. File Size	Comp. Ratio
Salinity	0.1196	1.0870	60,037	130:1
Temper.	0.7283	4.1620	69,375	112:1
Dis. oxy.	1.3055	8.7205	65,501	119:1
A. O. U.	1.4942	10.3178	59,019	132:1
P. O. S.	1.4930	8.3470	73,832	105:1
Phospha.	1.0581	7.4413	62,693	124:1
Nitrate	1.3299	10.8315	71,060	109:1
Silicate	0.9969	9.8598	46,998	165:1

Table 5: This demonstrates the result of running 3D-ODETLATP on the 8 datasets derived from WOA 2005. Smooth variable R is set to 0.01, sub-box is of size $30 \times 30 \times 30$, initial point selection picks every 5 points along the regular grid, and refinement of point selection adds 20 points at each iteration and forbidden zone is of size 2. The size of each dataset is 7.42Mb with each point uses 4 bytes. For example, compressed Salinity dataset file size by 3D-ODETLAP is 60,037 bytes and thus has a compression ratio of 130:1.

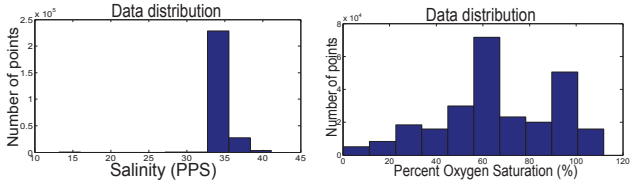

Figure 8: Histogram of salinity and Percentage Oxygen Saturation datasets used in Table 4

We can see from Table 4 that 3D-ODETLAP performs exceptionally well on Salinity data when compared with 3D-SPIHT. 3D-ODETLAP has a compression ratio 700% that of 3D-SPIHT's. This is mainly due to the intrinsic data distribution of different datasets. In Figure 8, almost 90% of salinity data scalar value fall into the range of 40 to 45 PPS. In contrast, Percentage Oxygen Saturation data scalar value distributes more evenly in its data range as shown in Figure 8. Because 3D-ODETLAP approximates the whole marine dataset by selecting certain points to "represent" their neighboring ones, a dataset of almost "flat", like salinity data, will require much fewer points to reconstruct from and thus will have better compression ratio while keeping errors the same with other methods. This feature of 3D-ODETLAP is especially valuable when we consider that lots of 3D marine, or even environmental datasets have a distribution with similar salinity in WOA 2005. Nevertheless, 3D-SPIHT is designed to be a general purpose compression method and can't take advantage of this special feature of marine datasets.

	3D-ODETLAP				3D-SPIHT		
Variable	Mean Error(%)	Max Error(%)	Compressed File(bytes)	Compression Ratio	Mean Error(%)	Max Error(%)	Compression Ratio
Salinity	0.0532	0.2174	27,377	77:1	0.0530	0.4946	11:1
Temper.	0.4993	2.0673	21,508	98:1	0.50	17.91	135:1
Dissolve.	0.9993	4.4145	21,040	100:1	1.002	24.9965	71:1
Apparen.	0.9999	4.0170	24,775	85:1	0.9991	20.3609	81:1
Percent.	0.9985	4.5672	26,743	78:1	0.9969	20.3610	65:1
Phospha.	0.9993	4.5241	24,493	86:1	0.99784	15.6922	65:1
Nitrate	1.0242	4.6946	31,905	66:1	1.0006	18.5360	59:1
Silicate	0.9996	5.1437	23,076	91:1	1.0018	21.6457	81:1

Table 4: This table demonstrates the comprehensive compression performance comparison between 3D-ODETLAP and 3D-SPIHT. Due to the data size limitations of 3D-SPIHT, we extract eight $128 \times 128 \times 32$ 3D data matrices and apply both methods on them. Mean Percentage Errors are kept approximately the same for comparison purposes. The size of each dataset is 2Mb with each point uses 4 bytes. For example, compressed Salinity dataset file size by 3D-ODETLAP is 27,377 bytes and thus has a compression ratio of 77:1.

6. CONCLUSION AND FUTURE WORK

This paper demonstrates our recent progress in three dimensional oceanographic data compression and reconstruction that processes eight distinct datasets in WOA 2005 data which stores environmental information such as temperature, salinity, etc. Current popular 3D compression methods like JPEG 20000 and SPIHT extend their 2D compression to 3D. These methods may serve well for general purpose data compression, but for data that preserves features in a specific field, like WOA 2005 in marine study, they lack the flexibility to adjust themselves to the field. By contrast, the 3D-ODETLAP method has great adaptability to enable itself to find a good, though possibly not optimal, set of parameters to meet the needs for compressing data within a specific field.

Moreover, by applying wavelet-based 3D-SPIHT on the same oceanographic datasets we used testing 3D-ODETLAP, we find that 3D-ODETLAP can achieve much better compression ratio while intentionally keeping the mean percentage error on all respective datasets approximately the same. We can also obtain a much smaller maximum percentage error in our comparison.

One possible future extension is to speed up the linear solver for our PDE through parallelization. The emerging GPGPU[1] technology may be a candidate for potential research given the fact that NVIDIA provides CUDA[8](Compute Unified Device Architecture) for data parallel computing and it's been popular, relatively easy and much cheaper than computing on huge clusters.

7. ACKNOWLEDGMENTS

This research was partially supported by NSF grant CMMI-0835762. We also thank Prof. Miriam Katz for her valuable advice on data preparation.

8. REFERENCES

[1] GPGPU(General-Purpose computation on GPUs). `http://www.gpgpu.org`, (retrieved 3/22/2010).

[2] Antonov, J. I., R. A. Locarnini, T. P. Boyer, A. V. Mishonov, and H. E. Garcia. World ocean atlas 2005, volume 2: Salinity. page 182, 2006.

[3] W. R. Franklin. Applications of geometry. In K. H. Rosen, editor, *Handbook of Discrete and Combinatorial Mathematics*, chapter 13.8, pages 867–888. CRC Press, 2000.

[4] Garcia, H. E., R. A. Locarnini, T. P. Boyer, and J. I. Antonov. World ocean atlas 2005, volume 3: Dissolved oxygen, apparent oxygen utilization, and oxygen saturation. page 342, 2006.

[5] Garcia, H. E., R. A. Locarnini, T. P. Boyer, and J. I. Antonov. World ocean atlas 2005, volume 4: Nutrients (phosphate, nitrate, silicate). page 396, 2006.

[6] J. Luo, X. Wang, C. Chen, and K. Parker. Volumetric medical image compression with three-dimensional wavelet transform and octave zerotree coding. In *Proceedings of SPIE*, volume 2727, page 579, 1996.

[7] S. Muraki. Volume data and wavelet transforms. *IEEE Comput. Graph. Appl.*, 13(4):50–56, 1993.

[8] Nvidia Corporation. CUDA (Compute Unified Device Architecture). `http://developer.nvidia.com/object/cuda.html`, (retrieved 3/22/2010).

[9] A. Said and W. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on circuits and systems for video technology*, 6(3):243–250, 1996.

[10] G. Sewell. *The numerical solution of ordinary and partial differential equations.* Wiley-Interscience, 2005.

[11] J. Shapiro et al. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on signal processing*, 41(12):3445–3462, 1993.

[12] J. Stookey, Z. Xie, B. Cutler, W. Franklin, D. Tracy, and M. Andrade. Parallel ODETLAP for terrain compression and reconstruction. In *16th ACM GIS symp.*, 2008.

[13] Z. Xie, W. R. Franklin, B. Cutler, M. Andrade, M. Inanc, and D. Tracy. Surface compression using over-determined Laplacian approximation. In *Proceedings of SPIE Vol. 6697 Advanced Signal Processing Algorithms, Architectures, and Implementations XVII*, 27 August 2007.

[14] Z. Xiong, X. Wu, S. Cheng, and J. Hua. Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms. *IEEE Trans. Med. Imaging*, 22(3):459–470, 2003.