The Multiresolution Toolkit: Progressive Access for Regular Gridded Data

John Clyne
Scientific Computing Division, National Center for Atmospheric Research
1850 Table Mesa Dr., Boulder, CO 80303 USA
clyne@ncar.ucar.edu

Abstract

We present the Multiresolution Toolkit (MTK), a wavelet based software system for enabling progressive access to large, regular gridded data sets. Transformations into and out of the wavelet domain using our methods are highly efficient, permitting application users to make effective speed/quality tradeoffs. The transformations operate on floating point data, are simple to implement, and lossless, making our approach a viable alternative data representation format. The method may be easily incorporated into new or existing visualization and analysis tools with only minor modification. We demonstrate the utility of our system by exploring a large turbulence simulation on a desktop workstation using a collection of multiresolution applications we have developed or extended with MTK.

Key Words
Data Treatment and Visualization, Visualization Software, Progressive Data Access, Wavelets, Large Data

1 Introduction

In numerous scientific disciplines the ability to generate data, either through instrumentation or numerical simulation, has out-stripped our ability to manipulate, visualize and analyze these data. The result is that many data sets, often produced at great expense, are not fully investigated. In this paper we describe a wavelet-based software system that provides progressive data access, enabling great reduction in computing resources required to operate on these data sets by providing reduced approximations to them. The underlying premise for this work is that many forms of scientific inquiry can be performed using coarsened, and therefor less resource intensive, approximations of the data. The information extracted from these approximations may be sufficient for the task at hand or in many cases may serve to simply narrow the domain (spatial and/or temporal) upon which further investigation may be conducted at greater detail. Though others have proposed hierarchical data representation schemes [17, 3, 10, 18], to our knowledge our method is the only one that offers all of the following important traits:

- **Out-of-core:** Our implementation operates out-of-core for both forward and inverse transformations, permitting extremely large data sets to be processed using only a modest memory footprint.
- **High efficiency:** Both inverse *and* forward transforms are highly cpu and I/O efficient.
- **Subsetting:** Subregions of the data may easily and efficiently be extracted at varying resolutions.
- **Minimal storage overhead:** Only a few bytes of header information are required, typically representing less than 0.01% of the data.
- **Resampling:** Approximations are produced by resampling the original data, not by simply subsampling.

2 Previous Work

The challenges posed by large data analysis are not limited to the numerical simulation community, or visualization in general. Much work has been done in the area of compression, possibly combining hierarchical representation, with an aim towards enabling interactive display by permitting speed/quality tradeoffs.

Triangle mesh simplification methods have demonstrated the utility of data coarsening combined with progressive refinement for the display of the largest triangle meshes [16]. In the area of data visualization, Schroeder et al. were probably the first to apply mesh simplification to address the notorious problem of over tesselation with the marching cubes and related isosurfacing algorithms [15]. Lindstrom and Silva investigate out-of-core mesh simplification methods for some of the largest published scientific data sets [9]. While these approaches can greatly reduce triangle counts while preserving surface fidelity, and are well-suited to progressive display, they suffer from long preprocessing times, the need for additional storage if the full resolution representations are to be preserved, and preclude interactive isosurface selection. Further, these methods provide little benefit to other stages in the pipeline beyond rendering.

Others have employed compression and hierarchical methods to quantized data in order to reduce processing, storage and/or communication requirements specifically for direct volume rendering. Levoy was the first to

employ hierarchical decompositions in an effort to accelerate ray casting by using octrees skip areas of low opacity [8]. Ihm and Park were the first to adapt 3D wavelet compression strategies that permit random access, an essential capability for the changing access patterns required by 3D data visualization [7]. Rodler treats 3D volumes as a collection of 2D slices, employing video coding methods to the slices and 2D wavelet transforms within each slice [12]. The wavelet hierarchy presented by Guthe et al. is most similar to our own, though their focus is direct volume rendering, not general data simplification [6]. All of these methods employ lossy compression strategies, quantizing floating point data into narrow integer representations, thus requiring multiple copies of the data to be maintained if the original data are to be preserved.

The more general approach of using hierarchical approximations to represent the data sets themselves with varying resolutions was introduced by Cignoni et al. [3] and Wilhelms and Van Gelder [17]. Cignoni et al. generate multiresolution representations of scattered data through successive tetrahedral refinement of a coarsened initial approximation [3]. While Cignoni's method is capable of supporting irregular as well as regular data, at the same time it fails to preserve regularity in coarsened representations even when it originally existed. Though not based on wavelet theory, Wilhelms and Van Gelder's approximation strategy is more similar to our own, exploring a small number of basis functions to reconstruct the field [17]. Zhou introduces yet another tetrahedra based framework, differing from that of Cignoni in that interpolation is avoided by relying on subsampling, making their method more efficient, though limited to regular grids [18]. For the most part all of these methods are cpu and memory intensive, do not support region subsetting, and impose significant storage overhead.

Most closely related to our own efforts are those of Pascucci and Frank [10]. Unlike our own wavelet-based approach, Pascucci and Frank's method is based upon Space Filling Curves (SPF) that enable subsetting and progressive access for regular gridded data by simply reorganizing the layout of the data storage. These methods do not alter the data in any way and are therefore inherently invertible. However, coarsened resolutions are then necessarily produced by nearest neighbor sampling, leading to inferior approximations.

3 Algorithm

Our algorithm, described in more detail elsewhere [4], assumes a Cartesian gridded, regularly spaced data volume of dimension $N_x \times N_y \times N_z$ where each dimension is of size 2^j for some positive integer j. The power-of-two restriction may be relaxed with either padding or careful attention to boundary conditions. We reorder our $N_x \times N_y \times N_z$ volume into blocks of size b, where b is also a power of two, and associate subvolumes consisting of eight neighboring blocks with a $2 \times 2 \times 2$ *superblock* . For each block in a superblock

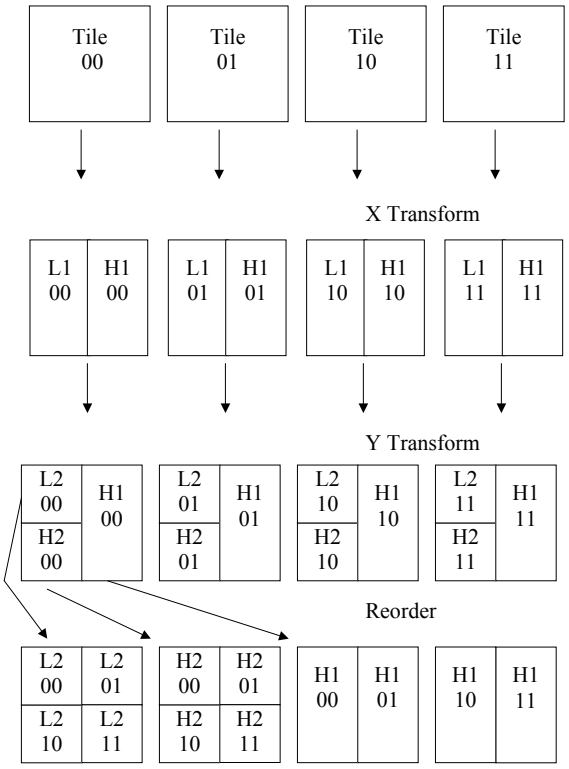

Figure 1. Forward transformation and reordering of four neighboring 2D tiles composing a supertile

we apply a single pass of a separable wavelet transform using the Quincunx decomposition ordering [13]. That is, we first transform along the X axis, decomposing the block into two subbands. We then apply the Y direction transform only to the low-frequency λ coefficients resulting from the X transform. Similarly, we then apply the Z axis transform only to the new λ coefficients resulting from the Y axis pass. After a single complete pass of the wavelet transform has been completed for all the blocks in a superblock, we gather the λ_{0-1} and high-frequency γ_{0-1} coefficients of the superblock together in the manner depicted by the 2D example of a supertile shown in figure 1.

After each superblock has been transformed in this manner, we recursively apply the procedure to each set of λ coefficients produced in the previous pass, forming new superblocks with each pass. With each pass the number of available λ blocks is reduced by a factor of eight. Processing stops when only a single λ block remains or after a user-determined limit is reached.

3.1 Implementation

The paragraphs above describe our conceptual approach. In practice, we balance memory and efficient disk access, applying the forward transform using a depth-first, binary tree traversal with slabs of raw block data serving as leaf nodes. The data is streamed out-of-core with slabs read in sequen-

tially from disk, only as needed, in a single pass. In this way our memory footprint can be kept quite small, requiring only memory to contain less than $4 \times Nx \times Ny$ blocks, where Nx and Ny are the resolution X and Y volume coordinate dimensions in blocks, respectively. The depth-first tree traversal operates as follows: If a node is a leaf node, a slab of blocks is read into memory as λ_0 coefficients. If a node is not a leaf node and not the root, the forward wavelet transformation is applied along all three axis to the λ_j coefficients stored by the node's two children as indicated in figure 1. The resulting γ_{j-1} coefficients are written to disk and the new λ_{j-1} coefficients are stored in memory for subseqent processing when the node's parent is visited. If the node is the root, or the user-defined maximum transformation limit is reached, then the λ_j coefficients are written to disk as well.

The inverse transform operates in much the same way as the forward, reading λ and γ coefficients only as needed. Of course if we are reconstructing a subvolume, or the full volume at a coarsened resolution, our memory requirements are even lower. We also note that the reordering step can be performed during the transform, compressing these two operations into one.

The block based method just described has a number of attractions worth noting:

- The blocks, when sized appropriately, may be transformed quickly on cache-dependent microprocessors.
- By gathering our λ_j coefficients into new blocks at the end of each transformation pass, we can lessen the effects of block boundary artifacts typical of block-based encoding schemes.
- The number of coefficients transformed within a block does not decrease with each pass, facilitating the use of higher-order wavelets with larger supports.
- We can easily *and* efficiently reconstruct approximations of subvolumes or volumes at any desired power-of-two resolution. This is in contrast to previous block based approaches; by reordering our storage of coefficients such that coefficients are grouped by level instead of by spatial location, we can better accommodate page-based storage access, greatly reducing cache and page faults in the case of in-core access and I/O operations for out-of-core access.

4 The Multiresolution Toolkit

To demonstrate the applicability of our progressive data scheme we have implemented a suite of tools we collectively call the Multiresolution Toolkit (MTK). MTK presently consists of a base C++ object library, upon which multiresolution tools may easily be layered, a direct volume rendering application, MDB, extensions to the popular commercial analysis package, RSI's Interactive Data Languate (IDL), and finally, a data importer for the Visualization Toolkit (VTK) [14].

4.1 MTK library

The base library consists of a collection of C++ class objects for applying forward and inverse wavelet transformations on 3D Cartesian gridded data, as described previously. The class library encapsulates a 3D implementation of the Lifting method based on Swelden's [5] Liftpack software. Hence, a wide family of wavelet transforms are supported. Data operated on by the transformation objects are assumed to have been previously blocked. Padding is not necessary for non power-of-two data; the wavelet support is adapted as necessary.

Layered on top of the core transformation object library is a file API for reading and writing wavelet coefficients from disk. The file format itself is not exposed. The API provides a simple interface for streaming, out-of-core access with transformation files; files may be read by arbitrary axis-aligned, block-bounded subregions, and read or written by slabs. The width of a slab is one block. In addition to storing the actual wavelet coefficients the wavelet block files also contain a small amount of metadata, including min, max values for each block. Each approximation level is stored in a separate file to enable the off-line storage of the highest-frequency detail coefficients if desired.

4.2 MDB

The Multiresolution Data Browser (MDB) implements a direct volume rendering engine with support for progressive data access. In particular, region subsetting is supported (axis-aligned, rectangular parallelpipeds). MDB uses a block-based LRU cache to improve performance. Blocks of data are quantized to 8-bit quantities on the fly at negligible cost and stored in the data cache. The cache size may be set arbitrarily large by the user. The cache is particularly beneficial when moving back and forth between resolutions and for supporting temporal animations. A 200 time step 128^3 volume requires 400MBs of memory and is easily accommodated by even a modestly equipped PC. Once loaded into main memory the volumes may be streamed for interactive temporal animations. Two direct volume rendering engines are currently supported, one based on SGI's Volumizer [1], the other is implemented directly on top of OpenGL using 2D texture mapping, enabling use on PCs with commodity graphics cards.

Because our algorithm preserves the regularity of the data, adapting our volume rendering engines for progressive access is trivial; they need only be capable of handling data with varying resolution from frame to frame. We must also scale opacities for the changing sampling distances so that the translucency does not depend on the number of sampling points which vary for different approximations. We can accomplish this oblivious to the rendering engine by adjusting the opacity lookup table. The composting operation is non-linear and the correct equation for opacity adjustment is $o' = 1.0 - \sqrt[n]{1.0 - o}$, where n is 1^{j+1}.

4.3 Extensions to existing tools

We've extended two existing tools to take advantage of progressive data access: IDL, and the popular freeware visualization toolkit, VTK [14]. IDL is a general purpose data analysis and post-processing tool that is widely used by researchers in the earth sciences. Although recent versions of IDL provide threaded implementations of many of their computationally intensive intrinsic functions, many user-defined IDL functions are serial. Hence, performance is often far from interactive. The progressive data access capability we've added to IDL can greatly reduce the cost of many operations.

Similarly, we have added a MTK data importer to VTK. VTK, while possessing tremendous functionality, is notorious for its poor performance on large data. Again, progressive data access provides a means for accelerating any of VTK's robust suite of visualization algorithms.

5 Results

In the section that follows we present some results from our implementation. Our platform for experimentation is an SGI Octane2 with a 600Mhz, R14000 processor, 5 GB's of memory, V12 graphics, and a Fiber Channel attached RAID. Our data set is a 200 time-step, 512^3 solar turbulent convection simulation occupying nearly half a terabyte of storage. We've transformed the data using three passes of the Haar wavelet transform, as described, resulting in a base volume with a resolution of 64^3. The block size we have chosen for our tests is 32^3, which strikes a balance between processor cache efficiency and having an adequate number of coefficients for our transforms.

We are interested in exploring several aspects of the performance of our method: 1) the quality of the coarsened approximations, 2) the speed at which we can reconstruct a given volume or subvolume to a desired resolution, 3) the performance benefit to our data browser, and 4) forward transform performance. This last metric, often ignored elsewhere, is an important consideration, particularly when dealing with time varying data possessing multiple time steps with high spatial resolutions.

Figure 2 shows a volume visualizations of our turbulence data set that has been direct volume rendered using VTK's software ray caster. Each coarsening in resolution represents a factor of eight reduction in data and a corresponding reduction in storage, communication, and processing costs. Clearly the image degrades at lower resolutions, but even the 128^3 approximations, which represent 1/64th of the original data, preserve the overall structure of the simulation. Only the finest details are lost and images of this fidelity may still be used to identify the location, or observe dynamics, of the large scale vortices. At 256^3 hardly any degradation is observed and it is difficult to argue that the higher fidelity images resulting from the 512^3 data warrant the expense in their computation for the view point selected. Clearly if we were to zoom in on a feature, the higher resolution data would be desirable. But in this case view frustum clipping would eliminate much of the domain from the field of view, permitting us to subset the volume and avoid reconstruction of the entire domain.

Table 1. Timings in seconds for subregion extraction

Size	Transform	I/O	Total	SPF method
128^3	0.09	0.31	0.40	0.56
256^3	0.19	0.56	0.75	0.57
512^3	0.31	0.82	1.16	0.65
1024^3	0.39	1.64	2.02	0.70

To demonstrate scalability and test the efficiency of reconstruction we ran two different experiments using a 1024^3 upsampled version of our 512^3 data. The forward wavelet transformation was applied using four passes resulting in a base resolution of 64^3. In our first performance experiment we look at the efficiency of extracting a 128^3 subvolume from the center of the domain at varying transformation levels. Table 1 shows the average timings for the I/O and the computation of the reconstruction transformation itself. Also shown is the total reconstruction time using the SPF method of Pascucci and Frank. We observe that the costs of both I/O and the wavelet transformation increase in proportion to the number of passes required to reconstruct the subvolume. This result is expected as with each pass the number of additional coefficients required to reconstruct a subvolume of constant dimension remains constant. However, even in the case of the 1024^3 volume the extraction takes only a couple of seconds. We also note that I/O dominates the reconstruction costs, emphasizing how little overhead our data representation imposes. Lastly, we note that the SPF method, which does not rely on data transformations, has near-constant performance.

In our second experiment we look at the cost of reconstructing the entire volume domain at varying resolutions. Again our base resolution is 64^3. Table 2 shows the time distributions for our method and again compares the total time with the SPF approach. The coarsest volume in our test is 64^3 which is the resolution of our transformed base volume. Hence, there are only I/O costs for reconstructing this volume and no transformation costs. As we would expect, the costs increase in linearly with the total number of samples. However, unlike with our subvolume extraction experiments, the transformation and I/O costs are more balanced for full volume reconstructions. We speculate that this is due to efficiency gains in the I/O stage which can read in larger chunks of contiguous data when not extracting a subvolume. We note that the good balance between the I/O and transformation points to future possible efficiency gains by pipelining these steps. Finally, we note that unlike with subregion extraction the costs of the SPF method grow exponentially with the number of samples.

Forward transformation costs are an important con-

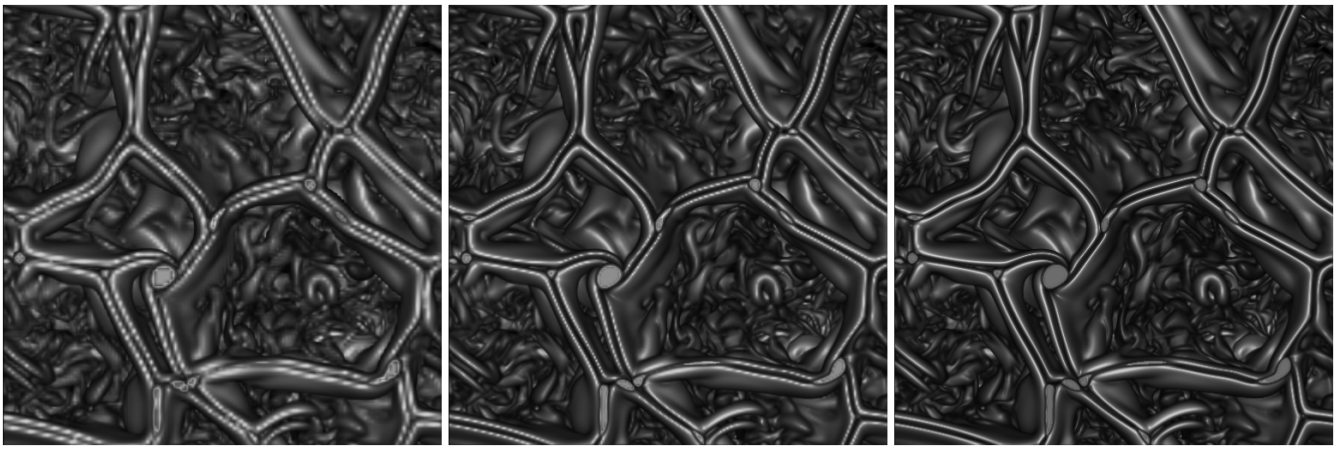

Figure 2. The Rast turbulence data set direct volume rendered at approximation resolutions of 128^3 and 256^3, and native 512^3 resolution, from left to right.

Table 2. Timings in seconds for full domain reconstuction

Size	Transform	I/O	Total	SPF method
64^3	0.0	0.13	0.13	0.10
128^3	0.11	0.22	0.33	0.60
256^3	1.21	1.14	2.36	17.43
512^3	11.01	9.85	20.86	278.17

Table 3. Timings for three passes of forward transform

Size	Read	Write	Trans.	Total	Memory
256^3	1.3	0.5	1.1	2.9	28 MBs
512^3	15.6	3.9	8.8	28.3	92 MBs
1024^3	207.7	42.7	67.1	307.5	351 MBs

sideration for extremely large data sets. Table 3 shows the time in seconds for reading raw data from disk, applying three passes of the forward transform, and writing the coefficients back to disk. The amount of processor memory allocated, as reported by the IRIX operating system, is also shown. We note that unlike other progressive access methods reported in the literature, forward transformation, like the inverse, is quite fast, and again the I/O times dominate. We also see that as expected our memory requirements grow only with N^2, making possible the transformation of a 1024^3 data set, which occupies 4GBs of disk space, using less than 400 MBs of main memory.

Finally, we look at application performance. Table 4 shows rendering rates of our multiresolution direct volume renderer, MDB, and Table 5 shows the performance of IDL computing a histogram at various resolutions. As we would expect, as the resolution decreases the performance increases. However, in both cases as the resolution becomes extremely coarse the inverse resolution/performance relation becomes non-linear; the speedup is not as we'd hope.

Table 4. Performance of volume rendering with MDB

	64^3	128^3	256^3	512^3
FPS	13.0	6.0	1.9	0.3
Speedup	47.1	22.0	6.9	1.0

Table 5. Performance of computing a histogram with IDL

	64^3	128^3	256^3	512^3
Time/sec	0.48	0.9	2.6	21.2
Speedup	44.0	23.6	8.1	1.0

We expect this is due to other factors (e.g. setup time) that begin to dominate the execution time. Nevertheless, highly interactive rates are achieved with the coarser data.

6 Discussion

As researchers employing numerical models strive to resolve details in their simulations at finer and finer scales, computational grids and the resulting data sets they generate have reached extraordinary sizes. Interactive analysis and post processing of these data at their native resolutions requires computational resources at least on par with those that produced the data. Unfortunately our experience at the National Center for Atmospheric Research has been that analysis platforms of this scale are seldom available. As a result, many numerical modelers are now faced with a deluge of data that they are ill-equipped to handle. Their challenge is often not to compute but to analyze.

While it is hoped that these data sets were not generated or acquired at such great scales needlessly, it is not necessarily the case that the full resolution data are essential for all aspects of analysis. In particular, highly effective

data browsing or *discovery* may be possible using lower resolution approximations if greater interactivity is enabled and the full resolution data is preserved for subsequent further analysis of detected features of interest [11, 2]. It is this paradigm – interactive browsing of coarsened data, followed by possibly non-interactive deeper analysis of features in a reduced domain – that our data representation scheme targets.

7 Conclusions

We have presented a collection of tools that enable exploratory data analysis of vast data sets by taking advantage of our wavelet-based, progressive access data representation scheme. Our toolkit permits many existing applications to be easily extended to support progressive access. The data representation enables highly efficient simplification and lossless reconstruction of floating point data at progressively finer resolutions, making the method an attractive alternative to the conventional Z-ordered storage of arrays. By simplifying the data set, as opposed to simplifying visualization output primitives, we stand to accelerate the entire visualization process.

We anticipate extending our research in several directions. Higher-order wavelets, though more computationally complex, could provide more accurate representations of the coarsened data than those possible with the highly computationally efficient Haar transform, allowing further speed/quality tradeoffs. Lossless compression might also be explored to reduce storage requirements. Irregularly spaced data might also be accommodated though the use of second generation wavelets.

Acknowledgements

The author would like to thank Randy Frank for providing the space filling curve source code and his insightful comments, and thank Mark Rast and NCAR's High Altitude Observatory for provision of the Rast solar turbulent convection data set.

References

[1] P. Bhaniramka and Y. Demange. Opengl volumizer: A toolkit for high quality volume rendering of large data sets. In *Proceedings of 2002 Workshop on Volume Visualization*, pages 45–54, 2002.

[2] N. Brummell. Scientist at jila, university of colorado. Personal Communication, 2001.

[3] P. Cignoni, L. De Floriani, C. Montoni, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *1994 Symposium on Volume Visualization*, pages 19–26, 1994.

[4] J. Clyne. An efficient hierarchical data representation scheme for gridded data, 2003. White Paper, http://www.vets.ucar.edu/Reports/wavelet.pdf.

[5] G. Fernández, S. Periaswamy, and W. Sweldens. LIFTPACK: A software package for wavelet transforms using lifting. In *Wavelet Applications in Signal and Image Processing IV*, pages 396–408. Proc. SPIE 2825, 1996.

[6] S. Guthe, M. Wand, J. Gonser, and W. Straßer. Interactive rendering of large volume data sets. In *IEEE Visualization '2002*, pages 53–60, 2002.

[7] I. Ihm and S. Park. Wavelet-based 3D compression scheme for interactive visualization of very large volume data. In *Computer Graphics Forum*, volume 18(1), pages 3–15, 1999.

[8] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9(3):245–261, 1990.

[9] P. Lindstrom and C. Silva. A memory insensitive technique for large model simplification, 2001.

[10] V. Pascucci and R. Frank. Global static indexing for real-time exploration of very large regular grids. In *Proceedings of Supercomputing 2001 Conference*, 2001.

[11] M. Rast. Scientist at the national center for atmospheric. Personal Communication, 2001.

[12] F. Rodler. Wavelet based 3d compression for very large volume data supporting fast random access. In *Proceedings of Pacific Graphics '99 Conference*, pages 108–117, 1999.

[13] D. Salomon. *Data Compression*. Springer-Verlag, 2000.

[14] W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*. Prentice Hall, 1996.

[15] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[16] E. Shaffer and M. Garland. Efficient adaptive simplification of massive meshes. In *Proceedings of IEEE Visualization '01*, pages 127–133, 2001.

[17] J. Wilhelms and A. V. Gelder. Multi-dimensional trees for controlled volume rendering and compression. In *1994 Symposium on Volume Visualization*, pages 27–34, 1994.

[18] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *IEEE Visualization '97*, pages 135–142, 1997.