

Using Desktop Graphics Workstations for Interactive Remote Exploration of Large Data Sets¹

Lori A. Freitag²Raymond M. Loy³

Mathematics and Computer Science Division
Argonne National Laboratory

Abstract

The interactive visualization and exploration of large scientific data sets is a challenging and difficult task; their size often far exceeds the performance and memory capacity of even the most powerful graphics workstations. To address this problem, we have created a technique that combines multiresolution data reduction methods with parallel computing to allow interactive exploration of large data sets while retaining full-resolution capability. We describe the creation of reduced data sets using several different criteria including user-specified error bounds or a preset performance criterion. We discuss the software architecture of the system with particular emphasis on the algorithms used to efficiently create a reduced data set and the software used to communicate between the remote reduction server and the local graphics client. We present performance results for the visualization of Rayleigh-Taylor instability and hairpin vortex data sets.

1. Introduction

Interactively exploring tera- and petabyte data sets is an extremely challenging task, particularly for scientists whose primary access to visualization resources is a desktop graphics workstation. To address this problem, researchers are exploring a number of approaches that provide remote, interactive navigation of very large data sets including the following.

- *Image-based rendering techniques* use two or more reference images from multiple viewpoints to reconstruct either the geometry in a scene or new images of the scene as the user's viewpoint changes (see, for example, [1,2,3,4,5]). The primary advantage of this technique for remote data exploration is that the amount of data transmitted and manipulated locally is independent of the complexity of the scene or original data set. Thus the costs are fixed as data set sizes increase.

- *Remote parallel visualization servers* utilize remote computational resources to visualize full-resolution data sets either as the computation proceeds (e.g., [6,7]) or as a post-processing step (e.g., [8,9,10,11]). The geometries of the derived visualization entities, rather than images, are extracted and communicated to the graphics workstation for display. No approximation errors are introduced in this process, but the size of the reduced geometries grows as a function of the overall problem size.
- *Subsampling and clustering techniques* create smaller, full-dimensional data sets by sampling the original data at specified locations or by averaging clusters of points from the original data set. The simplest approach to subsampling is to create a uniform grid representation of the original data set, and this is common in practice. Alternatively, a hierarchical, multiresolution representation of the data can be constructed using, for example, quadtrees or octrees [12,13,14], progressive meshes [15], wavelets [16], or other clustering approaches [17,18]. The cost of these methods does not significantly increase as the number of visualization tasks increases, but the amount of data that can be used is limited by the speed and memory of the local graphics workstation. Thus, as the overall problem size increases, the percentage of subsampled points that can be used decreases leading to increased approximation errors.

Our approach uses a parallel octree infrastructure to create a multiresolution subsampled data set that is communicated to the local graphics workstation for visualization. We use an adaptive approach which allows the user to interactively request higher resolution in regions of interest without sacrificing local graphics performance. Thus, the approximation errors associated with data reduction can be mitigated at the cost of increased

¹ Portions of this paper have been previously published by the authors as "Adaptive, Multiresolution Visualization of Large Data Sets using a Distributed Memory Octree" which appeared in the Proceedings of SC99 held in Portland, Oregon in 1999. The authors were supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

² 9700 S. Cass Ave, Argonne, IL, 60439. freitag@mcs.anl.gov.

³ 9700 S. Cass Ave, Argonne, IL, 60439. rloy@mcs.anl.gov.

communication. In Section 2, we describe the parallel algorithms and data structures used to create the reduced data sets. We also discuss the insertion criteria that we have found to be the most useful. Our system combines the parallel octree software with a custom-developed vtk visualization tool, and we describe the architecture in Section 3. In Section 4, we show the use of this system to visualize data from simulations of Rayleigh-Taylor instabilities and hairpin vortices developing around a hemisphere. We show results that compare uniform and multiresolution subsampling techniques to demonstrate accuracy and efficiency tradeoffs between the two methods. Finally, in Section 5 we offer concluding remarks and indicate directions of future work.

2. Multiresolution Subsampling Using a Parallel Octree

Our approach to interactive remote data exploration is to use a parallel octree infrastructure to create a general-purpose tool for adaptive, multiresolution subsampling of the original data set. Currently, our system allows file-based scalar field input and inserts each data point into the appropriate leaf octant. That leaf is then evaluated according to a specified criterion and refined if necessary with its associated data points reassigned to the new leaf octants. Our subsampling code is general purpose and requires only spatial coordinate information from the original data set, with no connectivity information necessary. We have successfully used our approach with unstructured tetrahedral and hexahedral, block adaptive, and uniform meshes.

2.1 Insertion Criteria

To create the reduced data set, field data values are inserted into the appropriate leaf octants using their spatial location. The field data values are averaged or otherwise agglomerated after insertion into the octree. To provide an indication of the error associated with the reduced data set, we compute and store statistical values such as the standard deviation, σ , and maximum deviation from the mean, e , for each leaf octant. These values are normalized by the mean to yield σ_n and e_n , respectively, which are included as additional scalar fields to be visualized so that the user has an indication of the fidelity of the reduced data set to the original data set. These measures of error also serve to highlight potential regions of interest; the cells with a large deviation from the average value are likely to have fine-scale structure that was not adequately captured by the reduction process.

We provide default routines to support reductions that meet user-specified bounds in standard or maximum deviation. Additionally, we provide stubs to allow custom, user-defined insertion criteria to ensure wide applicability of our software. However, it is hard to predict the size of the reduce data set that results from a given error threshold. To assist with meeting a given performance constraint

(such as the maximum number of leaf octants), the system can automatically determine the error threshold. Error values from octant nodes that are candidates for refinement or coarsening are analyzed using a recursive histogram. A single leaf octant that is refined results in at most 8 nonempty leaves (a net increase of 7); leaf siblings that are coarsened to a single leaf have a net decrease of 7. Using these estimates and the error distribution from the histogram, an error threshold can be chosen which results in the desired number of leaf octants. The entire process may be repeated until the result is within a specified tolerance of the goal.

2.2 Parallel Octree Infrastructure

To effectively manage a large distributed data set, the octree must also be distributed across the processors of the parallel computer, and we now describe the data structures and techniques used for parallel creation, coarsening, and traversal of the tree. Efficient traversal of the parallel octree data structure is enabled by inter-octant links, which may be either local or off-processor. Each processor maintains a local root list of octants whose parents are off-processor. Spatial information associated with each of these local root octants allows the coordinate information of its descendants to be easily inferred without communication. An octree partitioning algorithm using a space-filling curve [19], together with arbitrary octant migration, is used for load balancing the field data and the associated octree. If the data reduction were closely coupled to the parallel application, octant migration could be performed to track the location of the application data.

When creating an initial parallel octree from file input, each processor reads portions of the file in parallel and distributes the data to the processor that owns the corresponding piece of the domain. To maintain good parallel performance, the octree is periodically rebalanced during this process. After rebalancing, the spatial information and processor association of all local roots are exchanged which allows processor ownership of any point to be determined easily. If a point is contained in more than one local root, the subtree associated with the smallest of these local roots contains the leaf octant into which this point should be inserted.

Once the parallel octree has been constructed, refinement and coarsening to comply with new error criteria are performed in parallel for each processor's local roots and no communication is necessary. Leaves are recursively subdivided when they exceed the target error criteria, E_{max} . A set of leaf siblings are pruned when their values are less than the prescribed minimum error criteria, E_{min} , and the total value among the siblings is less than E_{max} . We note that it is not always possible to meet both a maximum and minimum error criteria; in our implementation E_{max} takes precedence. The current algorithm declines to coarsen when off-processor links are encountered. Complete coarsening could be accomplished by migrating terminal

octants which are local roots to their parent's processor and repeating the tree adjustment. When no local roots are terminal, the process is complete.

After the parallel octree has been adapted to meet the insertion criteria, copies of the octants with their average data (but without the much larger original data) are migrated to processor 0. Processor 0 then publishes a unique vertex list, the octree connectivity, and the reduced data set which can be accessed by the visualization program. Alternatively, these quantities may be stored to a file for later processing.

3. Software Architecture

Our interactive data exploration system is comprised of four major components:

1. field data input to the data reduction code either through file input or potentially through interactive requests to a running application,
2. the parallel octree code to create the reduced data set,
3. the local visualization environment, which can consist of externally developed desktop tools such as vtk [20], the General Mesh Viewer (GMV) [21], and IRIS Explorer [22], or custom tools built for state-of-the-art display devices such as the CAVE virtual reality theater [23], and
4. a portable communication infrastructure that allows the user to make interactive requests for new reduced data sets from the visualization environment and will allow the octree code to obtain new field data from a running application.

These four components and their interactions are shown in Figure 1. The arrows between the components indicate the communication necessary for an interactive environment. The width of the arrows indicates the relative size of the messages; small messages are needed to request new reduced data sets or field data values; large messages are required to transfer information from the application to the octree code and from the octree code to the visualization environment. Solid boxes and solid arrows indicate components and interactions that currently exist; dashed box outlines and arrows indicate components and interaction models that are planned for future instantiations of the toolkit. The visualization toolkits listed in Roman font are currently supported; near-term support is planned for those in italics. Asterisks indicate extensible toolkits that will support interactivity; the others must be used with

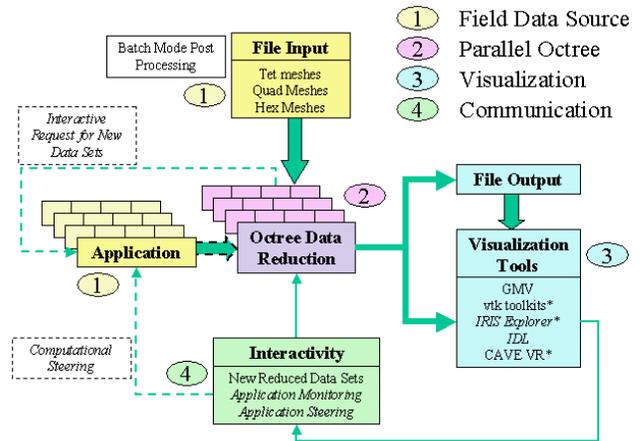


Figure 1: The four primary components of the data reduction toolkit and their interactions.

file input and output. We note that because our reduced data set is derived from an adaptive octree data structure, only visualization tools that can support multiresolution or unstructured data sets are considered.

We currently support two desktop visualization environments: GMV and a custom desktop toolkit based on vtk classes. GMV is a freely available software package from Los Alamos that is distributed in binary form. It is lightweight and easy to use but not extensible to support interactive requests to the parallel octree code. To support dynamic, adaptive level-of-detail requests, we developed a visualization tool using JAVA Swing components [24] to provide a GUI to the vtk classes that provide the basic functionalities of contouring, cutting planes, and vector glyphs. The user interface supports adaptive level-of-detail requests to the parallel octree code so that the user may interactively change the leaf criterion and thereby the resolution of the reduced data. The new criterion may be applied either globally or in a specified region of interest. In this way, the user can "zoom in" with high-resolution views in local subregions without sacrificing graphics performance. All results presented in this paper were obtained using the vtk environment.

Communication between the octree code and the desktop graphics application is performed using the ALICE Memory Snooper (AMS) [25] from Argonne National Laboratory. This communication infrastructure meets all of our design requirements; that is,

- it is both *portable* and *flexible* so that the parallel octree code and the visualization environment can run simultaneously on different, possibly remote, computer architectures,
- it allows *dynamic linking* and decoupling of multiple processes so that (1) the user can

periodically monitor a long-running application and
 (2) multiple scientists can collaborate while visualizing their data, and

- it helps manage *synchronization* of distributed data sets to ensure that the reduced data set is self-consistent.

The AMS library was designed to provide a lightweight API and infrastructure that enables computational steering and remote monitoring of application programs. The design is based on a client/server model using TCP/IP and Unix sockets that allows users to connect to a running application and access or modify the application's published variables at user-defined synchronization points. Thus it can be used both for fulfilling interactive requests for new data sets from a long-running simulation and for modifying or "steering" the simulation. In the current instantiation, the octree code publishes the reduced data set and variables relating to the criteria for refinement and coarsening of the tree. The vtk client accesses that information and is able to change the refinement and coarsening criteria, define a bounding box to specify a region of interest, and access the resulting reduced data set.

4. Results

In this section we compare our multiresolution subsampling technique with a subsampling approach that uses a uniform grid. Subsampling using uniform grids has several performance advantages including faster creation of the reduced data set, a smaller amount of data communicated to the graphics workstation, and more efficient visualization algorithms. In fact, for a given reduced data set size, it can be shown that remote data exploration tools using uniform grids will outperform their multiresolution counterparts by a factors of four or more depending on the visualization algorithms and graphics hardware used [26]. Thus, the multiresolution technique will outperform the uniform grid method only if the subsampling approximation errors are reduced by the same amount or more using significantly fewer grid points.

To explore these tradeoffs, we subsample two different application data sets. The first data set is from a Rayleigh-Taylor (R-T) simulation in two dimensions and contains $N=5.3 \times 10^4$ data points. This data set is characterized by a contact discontinuity between two fluids of different density and represents a broad class of applications whose primary features are sharp discontinuities which are typically local, lower-dimensional phenomena. In the left image in Figure 2, we show uniform grid subsampling using 15616 data points; on the right, we show the leaf octants from a multiresolution subsampling using 4102 data points. The average errors are .0347 and .0339, respectively showing that you can achieve the same error using far fewer multiresolution grid points which offsets the performance differences.

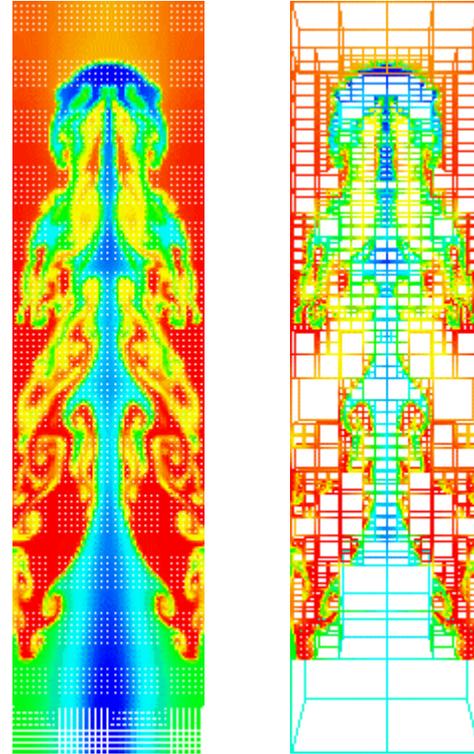


Figure 2: The left image shows the results of uniform subsampling using 15616 grid points for the two-dimensional Rayleigh-Taylor data set; the right image shows the results of multiresolution subsampling using 4102 grid points.

The second data set is from a three-dimensional simulation of hairpin vortices developing in flow around a hemisphere and contains $N=2.05 \times 10^6$ data points. This problem is representative of a class of applications in which the scalar field of interest describes fully three-dimensional features. In Figure 3, we show isosurfaces for a vorticity indicator of -0.28. The left image shows uniform grid subsampling using 34798 grid points which results in an average error of 3.18. The right image shows multiresolution subsampling with 34725 grid points which results in an average error of 1.46. Thus a much smaller error, and much more detail, can be obtained using the same number of multiresolution grid points than can be obtained with a uniform grid.

5. Directions of Future Work

Many computations are performed on nonuniform, adaptive grids which implies that a uniform grid subsampling will have more error in regions containing a large number of grid points which are typically the areas containing features of interest. Multiresolution approaches address this difficulty; subsampled points may be concentrated in the same regions as in the original



Figure 3: The upper image shows the results of uniform subsampling using 34798 grid points for the hairpin vortex data set; the lower image shows the results of multiresolution subsampling for 34725 grid points.

computation. In addition, our results indicate that they can be more cost effective than uniform grid subsampling when the performance metric is the cost to achieve a given level of error.

Ongoing work for this toolkit includes adding new functionalities to improve the flexibility of the octree code, performance tuning and testing, and eventually adding computational steering capabilities to monitor and change ongoing large-scale simulations. To improve the flexibility, we will provide more options for averaging or interpolating the data as it is inserted into the octree and for determining when to subdivide leaf octants including, for example, gradient tests on scalar fields. In addition, several performance improvements can be obtained in the parallel octree code. In particular, processing the initial data set from a file can be improved by leveraging ongoing work in collective communication within the ROMIO implementation of MPI-IO. We also plan to test the performance on wide area networks to tune the infrastructure for remote access to the reduced data set.

Acknowledgments

We acknowledge the FLASH project for providing the motivating applications described in this paper. In particular, we thank Mike Singer of Cornell and Henry Tufo of the University of Chicago for their assistance in

preparing the Rayleigh-Taylor and hairpin vortex data sets. We also thank Matt Ahrens and Ibrahima Ba for their help in incorporating the ALICE Memory Snooper into the data reduction software system.

References

- [1] Manuel Oliveira and Gary Bishop. Image-Based Objects. In *Proceedings of 1999 ACM Symposium of 3D Graphics*, pages 191-198, Atlanta, Georgia, April 1999.
- [2] Nelson Max. Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In *Rendering Techniques '96: Proceedings of 7th Eurographics Workshop on Rendering*, pages 165-174, New York, 1996. Springer.
- [3] Steven Gortler, Li Wei He, and Michael Cohen. Rendering layered depth images. Technical Report 97-09, Microsoft Research, March 1997.
- [4] William Mark. Efficient reconstruction techniques for post-rendering 3D image warping. Technical Report TR98-011, University of North Carolina, March 1998.
- [5] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. LDI tree: A hierarchical representation for image-based rendering. In *Proceedings of SIGGRAPH 99*, pages 291-298, Los Angeles, California, August 1999.
- [6] Robert Haimes. pV3: A distributed system for large-scale unsteady CFD visualization. *AIAA paper*, 94-0321, January 1994.
- [7] Arsi Vaziri and Mark Kremenetsky. Visualization and tracking of parallel CFD simulations. In *Proceedings of HPC 95*. Society of Computer Simulation, 1995.
- [8] T.W. Crockett and T. Orloff. A MIMD rendering algorithm for distributed memory architectures. In *Proceedings of the Parallel Rendering Symposium*, pages 35-42, 1993.
- [9] Thomas Crockett. Beyond the renderer: Software architecture for parallel graphics and visualization. Technical Report ICASE Report No. 96-75, Institute for Computer Applications in Science and Engineering, 1996.
- [10] Kwan-Lui Ma. Parallel rendering of 3D AMR data on SGI/Cray T3E. In *Proceedings of the Frontiers 99 Conference*, pages 138-145, 1999.
- [11] C. Hanson and P. Hinker. Massively parallel isosurface extraction. In *Proceedings of Visualization 92*. IEEE Computer Society, 1992.

- [12] Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-time, continuous level of detail rendering of height fields. In *Computer Graphics Proceedings SIGGRAPH 96*, Annual Conference Series, pages 109-118. ACM, 1996.
- [13] Brian Von Herzen and Alan Barr. Accurate triangulations of deformed, intersecting surfaces. In *Computer Graphics Proceedings, SIGGRAPH 87*, volume 21, pages 103-110. ACM, 1987.
- [14] Eric LaMar, Bernd Hamann, and Kenneth Joy. Multiresolution techniques for interactive texture-based volume rendering. In *Proceedings of IEEE Visualization 99*, pages 355-362, October 1999.
- [15] Hugues Hoppe. Progressive meshes. In *Computer Graphics SIGGRAPH 96 Proceedings*, pages 99-108, 1996.
- [16] Jos Roerdink and Michel Westenberg. Wavelet-based volume visualization. Technical Report IWI 98-9-06, Institute for Mathematics and Computing Science, University of Groningen, 1998.
- [17] Bjoern Heckel, Gunther Weber, Bernd Hamann, and Kenneth Joy. Construction of vector field hierarchies. In *Proceedings of IEEE Visualization 99*, pages 19-26, October 1999.
- [18] Alexandru Telea and Jarke van Wijk. Simplified representation of vector fields. In *Proceedings of IEEE Visualization 99*, pages 35-42, October 1999.
- [19] Joseph E. Flaherty, Raymond M. Loy, Mark S. Shephard, Boleslaw K. Szymanski, James D. Teresco, and Louis H. Ziantz. Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws. In *J. Parallel and Dist. Comput.*, 47:139-152, 1997.
- [20] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.
- [21] Frank Ortega. General mesh viewer, user's manual, 1999.
- [22] IRIS explorer, release 3.5, user's guide, Unix version, 1993.
- [23] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *ACM SIGGRAPH 93 Proceedings*, pages 135-142. ACM, 1993.
- [24] Robert Eckstein, Marc Loy, and Dave Wood. *JAVA Swing*. O'Reilly and Associates, Sebastopol, California, 1998.
- [25] Ibrahima Ba, Christopher Malon, and Barry Smith. Design of the ALICE Memory Snooper, <http://www.mcs.anl.gov/ams>, 1999.
- [26] Lori Freitag and Raymond Loy. Comparison of remote visualization strategies for interactive exploration of large data sets. Technical Report ANL/MCS-P803-0300, Argonne National Laboratory, April 2000.