Visualization of Dynamic Adaptive Resolution Scientific Data

Andrew Foulks, R. Daniel Bergeron, and Samuel H. Vohr

Department of Computer Science *
University of New Hampshire, Durham, NH

ABSTRACT

Interactive visualization of very large data sets remains a challenging problem to the visualization community. One promising solution involves using adaptive resolution representations of the data. In this model, important regions of data are identified using reconstructive error analysis and are shown in higher detail. During the visualization, regions with higher error are rendered with high resolution data, while areas of low error are rendered at a lower resolution. We have developed a new dynamic adaptive resolution rendering algorithm along with software support libraries. These libraries are designed to extend the VisIt visualization environment by adding support for adaptive resolution data. VisIt supports domain decomposition of data, which we use to define our AR representation. We show that with this model, we achieve performance gains while maintaining error tolerances specified by the scientist.

Keywords: Adaptive Resolution Data, Visualization of Error, I/O Bottleneck

1. INTRODUCTION

The size and scope of modern multidimensional datasets produced by the scientific community leaves us with a challenge: how to visualize and analyze multiple terabytes of data interactively on a typical, off the shelf workstation. Typically, today’s simulations generate far more data than the scientist can effectively understand. Understanding the science behind the data of these complex datasets remains a prominent scientific goal. The large size of the data represents an obstacle to timely scientific research, even on powerful modern workstations.

Large data sets offer more precise details and allow the scientist to gain greater insights from the data. Their size however, lead to two major obstacles to effective interactive browsing by the scientist. First, the I/O bandwidth limitation: the costs of transferring the data from storage to memory can be prohibitive given the scope of typical scientific data sets today. Second, the resolution limitation: Both human and display resolution limits restrict the amount of information that can be displayed and processed at any moment. Ideally, we would like to perform visualizations on a workstation or a laptop and still impart the information contained in large data sets.

Interactive visualization of simulation data is a very effective technique for gaining insight into the phenomena behind the data. It is the huge size of these science datasets that limit the interactivity in the visualization environment. Often space science disciplines must deal with data that is much too large to be stored in main memory or to be represented on a display screen of only a few million pixels. A multiresolution data representation¹ can help the scientist deal with this problem: A coarse resolution can be used to see an overview of the data and give the scientist the opportunity to adjust the rendering parameters to create the visualizations. Once the visualization parameters have been set, the rendering can be run off-line at a finer resolution, producing a more detailed animation at the cost of losing interactivity.

As an extension of this idea, we propose using an adaptive resolution representation to serve essentially the same purpose – to provide the scientist with an overview of the data. In this model we continue to take advantage of I/O performance gains while keeping the most interesting regions of the data at finer resolutions.

Our adaptive resolution representation data model is built upon 4 basic concepts. (1) We first build upon our existing multiresolution data model, (2) The data undergoes an error analysis phase, (3) An AR representation

*This work was supported in part by NASA grant AISR05-0071, and the National Science Foundation grant IIS-0082577. 
{rafoulks,rdb,svohr}@cs.unh.edu
is constructed based on an acceptable error tolerance specified by the scientist, and (4) Any geometrical gaps introduced between regions of differing resolutions are reconstructed before rendering. The adaptive resolution algorithm brings together different resolution chunks or subsets of uniform data, and the sequence of uniform data chunks is sent to the rendering pipeline.

This paper has three main goals. First, we describe our multiresolution data model with error and demonstrate software support for multidimensional scientific data. Second, we introduce an algorithm that dynamically builds an adaptive resolution representation of chunked multiresolution data with error, and an implementation that is integrated into the VisIt visualization toolkit. Finally, we show that rendering an adaptive resolution representation of the data results in a reduction in the I/O requirements while maintaining image quality.

2. PRIOR WORK

Multiresolution techniques for visualization of time varying data are an active research area. Guthe et al.\(^3\) use multiresolution representations to achieve high performance volume rendering. Our research is similar in spirit, but we focus on distinguishing between spatial and temporal hierarchies and how they can be effectively integrated into a single software framework. Finkelstein et al.\(^4\) use some interesting multiresolution techniques, but they focus on video images, where we are more interested in scientific data.

Doleisch et al.\(^5\) use a feature based approach and other techniques from information visualization to keep the amount of data that is presented to the user manageable. Our goals are similar since we also wish to reduce the amount of data that is presented to the user. Our approach is different in that we have concentrated on multiresolution techniques (in both space and time), and have built an application that demonstrates how our framework supports these techniques effectively.

Silver and Wang\(^6\) describe visualization algorithms to handle time series computational fluid dynamics data sets. In their approach, they use feature tracking to improve user interaction and enhance the visualization of time series data sets. They do not, however, incorporate the notion of multiresolution into the organization of their data model.

Recent work in adaptive resolution techniques comes from Ramachandran et al.,\(^7\) in which an intelligent data thinning algorithm is used to deal with large volumes of streaming satellite data. Their approach is similar to ours in that they retain data in regions of high spatial frequency while decimating regions with low variances. They focus on two dimensional atmospheric and climate satellite data, whereas our focus is primarily on three dimensional simulation data.

Other researchers have addressed the notion of combining temporal and spatial multiresolution data representations for specific applications such as dynamic polygon meshes\(^8\) as well as isosurface and direct volume rendering generation for time-varying fields.\(^9\)–\(^11\) Some have tried representing vector fields hierarchically,\(^12\) storing error information during decomposition. Such data would fit well into our framework, because our approach is neither application-specific nor does it depend on a particular technique for creating the multiresolution hierarchy.

3. MULTIRESOLUTION REPRESENTATION

We have developed a multiresolution scientific data model\(^1\) that incorporates spatial and temporal semantics with localized error and we have implemented a database system based on that model.\(^13,\,14\) The model’s semantics are common to many scientific applications and therefore are valuable to a variety of disciplines.

The model has both a Java\(^15\) as well as a C++ implementation.\(^16,\,17\) We term our data STAR data (Space Time multi/ Adaptive Resolution). In this data model, spatial and temporal multiresolution hierarchies are integrated in a global tree structure in which the base (finest) resolution sits at the root of the tree. Each component of a data hierarchy consists of uniform data at various resolutions. Adaptive resolution data hierarchies are metadata structures built on the multiresolution data. A data reduction tool and a software library were developed to form the foundation necessary to allow users to generate and access by arbitrary combinations of spatial and temporal decomposed data.

Our current Java implementation focuses primarily on static analysis of the data, producing a single AR representation for a given error tolerance. The C++ implementation uses a hybrid approach. A preprocessing
step is used to generate multiresolution data, error analysis, and store the data in an efficient chunking scheme. The plugin code dynamically generates AR representations as the visualization is being rendered, adjusting the visualization as the tolerance is changed interactively by the scientist.

As shown in Figure 1, the model includes a standard multi-level data hierarchy for large scientific datasets. Original data is stored permanently at a repository site or sites. A preprocessing operation generates a multiresolution representation with increasingly coarse representations and the associated localized error information at each level. Users can download arbitrary subsets from any representation level, trading off accuracy and dataset size based on the needs of particular tasks.

3.1 Wavelet Decomposition

One particularly effective way to generate multiresolution data is to apply wavelet transformations to such data. As shown in Figure 1, a wavelet transformation is applied to the entire data set uniformly and results in a new summary data set and detail coefficients, each $1/2^d$ the size of the original where $d$ is the dimensionality of the data. Decomposition techniques using wavelets lead to a natural multiresolution hierarchy in which the original (highest resolution) data is the root of the hierarchy and each successive child is the result of applying a wavelet transformation (or other uniform decomposition operation) to its child.

Our model supports both the spatial and temporal domains. These resolutions are integrated into a single comprehensive data representation. These implementations are in our STARgen and STARview tools. STARview supports multiple spatial wavelet decompositions of each step of a time series dataset, which can itself be a member of a multiresolution time series decomposition. A temporal data reduction applies a one-dimensional time series data reduction for each spatial position. Given a time step we can generate $n$ spatial resolutions for that time step. We complete the hierarchy by generating a coarser temporal resolution for each spatial resolution created.

4. VISUALIZATION ENVIRONMENT

Scientists studying space weather at the University of New Hampshire use the VisIt visualization environment to investigate their simulation data. VisIt is a framework developed to render multidimensional scientific data. This platform aims to give researchers scientific visualization tools for visualizing scientific datasets. It is built largely upon the Visualization Toolkit (VTK) libraries, extending the interface and providing a comprehensive environment to the scientist. VisIt supports many different types of data and has a modular architecture that allows users to build data plugins to access other types of data.

In previous work, we have integrated a multiresolution data representation into the VisIt environment. This code is scheduled to be added to the main trunk of VisIt and will be distributed with the system starting with release 2.1.

In this paper, we extend our previous implementation to support dynamically generated adaptive resolution data sets. We utilize VisIt’s domain decomposition API to render adaptive resolution data as a set of uniform data subsets. Using this approach, any existing rendering algorithm or extension to the system can use our multi and adaptive resolution data without modification or without direct knowledge of the nature of the data.
Our software consists of several modules, including a Database plugin, responsible for reading our multi and adaptive resolution format; and an Operator Attribute that defines a graphical user interface widget to control the multiresolution level of refinement and error tolerance specification.

5. CHUNKED DATA

Scientists have for many years been storing their multidimensional data in chunks. In a linear representation of the data, volume elements located near one another in the topology may in fact be far away in the file, requiring multiple reads to gain access to the data. By contrast, a chunked representation preserves some of the topological relationships based on a given access pattern (see Figure 2). A chunked representation preserves the natural spatial relationships inherent in volume data and thus improves performance greatly for many common access patterns.

We use chunked data primarily for performance. Since we already perform a multiresolution analysis with error before the visualization stage, we use this step to synchronize the sizes of the error regions with the chunking scheme used to store the data. As long as the error region sizes are not too small, we achieve I/O performance gains using this technique. Chunked data storage is not necessary, however. For flat files, our implementation dynamically retrieves sub regions of data as requested by the code that constructs the AR representation, regardless of the data storage format.

![Figure 2. Linear Storage vs. Chunked Storage](image)

6. ERROR MODEL

The reduction in size of the original data naturally leads the scientist to question the accuracy of the lower resolution representations. We have developed an error representation to help quantify the degree of lossiness in the lower resolution representations.

6.1 Reconstruction Algorithm

Our error model relies on the notion of reconstructed data. Low resolution data is resampled to match both the geometry and the topology of the original resolution data. Our model only requires that an approximation algorithm exists; there are many possible approaches that one can take to do this resampling. Our implementation mimics the inverse of the discrete Haar wavelet transform. The algorithm is based on using small groups of points (a voxel, or 8 values in 3 dimensions, or 4 values in two dimensions). We create a reconstructed dataset $R_r$ by iterating over the voxels in the data such that given a voxel in the low resolution data, the corresponding group of points at the high resolution is twice the size of each dimension. In three dimensions, each group of 8 low resolution values (2x2x2) is reconstructed into 64 values (4x4x4).

The reconstructed values are calculated from the low resolution data at level $r$ and a reconstruction function $R(\mathcal{R}_r)$. This function can be simple linear interpolation and extrapolation, if we assume that the original function describing the data is continuous, but need not be restricted to interpolation. Reconstructed datasets have the same topology and geometry as the original high resolution data. There is one reconstructed dataset per low resolution data. We use the notation $R_r$ to denote the dataset reconstructed from resolution $r$ dataset.
6.2 Point Error Data

A scientific dataset is a set of points with a geometry, a topology, and a geometry map function that maps points to locations, and a topology map function that yields neighboring information for a given point. Given a multiresolution dataset $R_0$ with a topology and a geometry, we define a point error dataset as the set of points resulting from the absolute value of the difference between the original value at the highest resolution, and its topologically equivalent reconstructed value. For each timestep we compare the reconstructed values to the original high resolution data (the ground truth), yielding a set of point error datasets, all at the same resolution as the original data. The error values, in general, for a point $p = (i, j, k)$ in three dimensions, are

$$e_p = |s_p - r_p|$$

We use the notation $E_{r,0}$ to denote these point error datasets. The first subscript ($r$) describes which low resolution dataset was used in the reconstruction process. The second subscript (0) indicates that the error dataset is at the same resolution as the original high resolution data set, $R_0$. There is, therefore, no $E_{0,0}$ dataset because all error values in this dataset would be 0. $E_{1,0}$ is the point error dataset whose error values represent the errors from resolution 1 data.

The point error datasets, being the same resolution as the original data, are prohibitively large and are therefore we desire techniques to reduce the size. Often we wish our point error datasets to match the size of the resolution that it describes. Our system offers one of two approaches to reduce the size of point error datasets.

1. We can consider using an approximation function. The approximation function is used to determine the approximate values at locations not specified in the geometry of the desired resolution. For low resolution data produced from wavelet decomposition, we know the geometrical locations of the data. For each location, we apply our approximation function to the appropriate point error dataset to yield estimates of the absolute error values at the new locations.

2. A second approach is to perform the wavelet decomposition algorithm on the point error dataset, yielding a hierarchy of point error datasets, each matching the size of its low resolution data counterpart.

We note that if the approximation function in the first approach is simple averaging, the result is the same as applying Haar wavelets in the second approach. In general, however, we need not be restricted to the discrete wavelet transform to obtain these hierarchies.

Once we have reduced the size of our point error datasets, we will then have, for each lower resolution, a corresponding point based error data set. We call this a point error multiresolution hierarchy, which mirrors the original multiresolution hierarchy. Each low resolution point based error data set has the same number of sample points as its corresponding low resolution data set.

6.3 Region Error

While point error data sets can be useful, particularly when visualizing the errors, it is often the case that the scientist wishes to have other types of error information about the low resolution data. Point error datasets, in our model, are absolute differences, because further statistical analysis is limited when dealing with a single point.

Our error model includes a more flexible region based approach. Using the point error datasets as our starting point, analysis is performed on groups of points, yielding more informative statistical analysis. Given an arbitrary region of point error values, we can record maximum and average absolute error, maximum relative error, standard deviation, peak signal to noise ratio, as well as other higher order statistical quantities.

As region size is reduced, the statistical analysis carries less meaning because of the small sample sizes. In the limit, as the region size approaches a single value, the model reduces to the point based representation described in Section 6.2. Because we can have different region sizes, the region based error model can be represented hierarchically based on region size.
Ideally, the sizes of the error regions match the size of the data chunks during the error analysis phase. The ideal size of the error regions is still ongoing research. We use the term granularity to describe the size of the error regions. As the granularity decreases, the number of chunks increases. If the size is too small relative to the size of the size of the chunks to be rendered, there could be a performance hit as the system gathers many small chunks of data at the same resolution to form a much larger region at a single uniform resolution.

Conversely, if the error regions are too large, we start to see regions that do not meet the tolerance specified because of a small subset of that region. The entire region will have to be rendered at a higher resolution even though only a small number of data points in the region may have contributed to the error tolerance failure.

Our solution is to use a hierarchical error region model. Essentially, we store error statistics for a variety of region sizes. Large region sizes mean that only a very few error values must be stored and is not costly spacewise. We subdivide each region using an octree data structure until we get to a minimum block size, which is a parameter in the system. If the minimum block size is a single value, we end up with the point error data model in which case only the absolute differences can be recorded. In our preliminary experiments, a single voxel (8 values) was, in some cases, too small to be an effective granular size. In this paper, we use a chunking scheme of 8x8x8, which results in a granularity of 64x16x32 (32768 values) for high resolution data and 8x2x4 (64 values) for the low resolution.

7. DYNAMIC ADAPTIVE RESOLUTION REPRESENTATION

Adaptive resolution representations of the data vary the resolution of different subsets by analyzing the local and global behavior of the data.28, 29 A coarse representation may be used for a region of the dataset if the amount of change in that region falls below a user specified error tolerance $\delta$. Typically, the scientist with knowledge of the dataset will specify $\delta$ based on the level of coarseness that is acceptable. The effect of this model is that regions of high interest are shown at a finer resolution, while less important regions are shown at coarse resolution.

Using low resolution data to represent regions of uninteresting data can vastly reduce the size of the data that needs to be read in and rendered. This representation can therefore help to reduce I/O and rendering pipeline computation costs.

Scientists have developed several different kinds of spatial subdivision representations that are used to implement adaptive resolution data, including hierarchical quadrees and octree structures for adaptive representations.30 More sophisticated structures such as the k-D tree have been used, but performance has been problematic.30 One drawback is that often a custom rendering algorithm must be developed to render the k-D tree properly.

We currently do not use k-D trees. We instead take one of two approaches:

1. If the data is flat, we have the error analysis phase produce a region error hierarchy using several different granularities. At visualization time, we use efficient subsetting operations31 to perform the error analysis.

2. If the data is chunked, we direct our error analysis to exploit the particular storage format. When using chunked data, the chunks represent natural partitions of the data, which are then used to build an adaptive resolution representation based on its chunking scheme.

By combining the ideas we have presented from our multiresolution data model, and the region based error analysis, we have developed an algorithm that will generate an adaptive resolution representation dynamically. In our implementation we use a general spatial subdivision model to divide the data evenly into $p \times q \times r$ subspaces. Given data in a chunked format, it makes sense for performance reasons to use the same spatial subdivision scheme for both the region error representation as well as the adaptive resolution representation.

At visualization time, we use our dynamic chunked adaptive resolution algorithm to build an AR representation from the multiresolution chunked data and region based error information. The AR data structure generated must conform to the error tolerance specified by the user. The advantage of using a dynamic algorithm is that the scientist can experiment with different error tolerances to get the desired performance characteristics during visualization.
By using more general spatial subdivision and combining this with chunked data, we can leverage existing algorithms that expect the data to be uniform. The dynamic nature of our adaptive resolution model allows the user to choose any region of interest and display it at an arbitrarily high resolution (up to the original data size).

### 7.1 Adaptive Mesh Refinement

It is important to distinguish between what we term **adaptive resolution** data and what the scientific community generally means when **Adaptive Mesh Refinement** techniques are used. Both approaches produce an adaptive resolution representation. The main difference is that AMR grids are typically chosen before the simulation is run. The scientist chooses regions of high resolution where the most detail is expected to be required. However, there is no guarantee that these high resolution regions will always have high frequency changes, especially over the course of many simulation timesteps. Often the scientist running the simulation defines high resolution submeshes encompassing a larger volume than necessary in order to assure that the interesting details are captured in the event of an imperfectly defined mesh.

Adaptive Resolution data is created after the simulation has been run. Error analysis identifies regions of high frequency changes. These regions are stored at higher resolutions than regions of the data where little or no changes take place. Adaptive resolution data representations can be generated from any type of data: unstructured, rectilinear, or AMR. An AR representation can further reduce the size of an AMR dataset if the scientist did not carefully predict which areas had the highest error.

In sum, the main purpose of using AMR data is to speed up the *simulation* time. The main purpose of using AR data is to speed up the *visualization* time by achieving performance gains in the I/O subsystem.

### 7.2 MHD Simulation Data

![Figure 3. Multiresolution GGCM Density Data](image)

Space science researchers have built and simulated numerical models of the solar wind and its interaction with the earth’s magnetosphere for only the past 20 years or so.\(^{32}\) The data we are currently working with comes from simulations run at the Space Science Research center at the University of New Hampshire. Their simulation of solar wind activity records 11 primary physical attributes, including plasma density, bulk velocity, magnetic field, and pressure. The data is a 3D time series in which points are sampled on a stretched Cartesian grid.\(^{33}\) The particular dataset used for the examples in this paper contains 1490 recorded time steps spanning a numerical simulation of 18660 time steps. The total size of this dataset is 1.1 Terabytes.
7.3 Visualizations

7.3.1 Multiresolution Visualizations

The images shown here represent renderings of the scalar density data at a single time step from Raeder et al.’s Geospace General Circulation Model (GGCM) simulation data. Figure 3(a) shows the data at high resolution (512x128x256). Figure 3(b) shows the data at resolution 3 (64x16x32).

7.3.2 Visualization of Error

Generally speaking, our point based error model lends itself quite nicely to visualizations because each error point has a natural location in the geometry. In this case it is quite easy to simply treat the error as data and use off the shelf rendering algorithms (pseudocolor maps, isosurfaces, volume renderings) to generate images.

With the region based error model, statistical analysis yields important information about regions of data. However, one of the drawbacks is that error values are associated with regions, not points. For the purposes of visualization, we can assign the region error values a point location that is at the center of the region they represent.

![Figure 4. Point Vs. Region Based Rendering](image)

In some cases it may not make sense to represent region error values in the visualization as point based data. A better approach may be to represent the values as cell based data. The cell based approach more naturally maps to the concept of regions. VisIt Pseudocolor renderings provide two ways to represent the data, either as nodes, or as zones, corresponding to point based and cell based techniques, respectively. Figure 4 shows the difference between the rendering approaches. In a point based rendering, shown in Figure 4(a), interpolation is performed between the sample points to obtain smoother visualization. In the region based approach, shown in Figure 4(b) and (c), each sample point is mapped to a color for the region in which it occupies. 4(a) and (b) both show the error associated with data at $R_1$, and 4(c) shows the error associated with the data at $R_3$. The images highlight the nature of the error model. The error associated with $R_3$ is in every chunk higher than the error associated with $R_1$.

7.3.3 Adaptive Resolution Visualization

In Figure 5, we show the same timestep rendered with an adaptive resolution with an error tolerance of 10%. In Figure 5(a), the AR rendering looks similar to the high resolution rendering shown in Figure 3(a). In Figure 5(b), the rendering shows the adaptive meshes created dynamically during the visualization.

The regions near the bow shock, which have the highest gradients, are the regions that have the highest data loss when using a multiresolution representation. The adaptive resolution representation here generally preserves areas near the bow shock at high resolution. The region away from the earth has very low frequency changes and can be represented with a lower resolution.
8. PERFORMANCE EVALUATIONS

To demonstrate that an adaptive resolution data model could be successfully integrated into the VisIt toolkit, we ran several benchmarks. We tested uniform chunked data and compared both the engine time and the rendering time for different adaptive resolution datasets. We create visualizations with VisIt’s Pseudocoloring visualization algorithm.

With our data model, our focus is primarily on speeding up the I/O portion of the rendering pipeline. We took several steps to ensure the OS did not cache or prefetch files. First, for each time step, we generate adaptive resolution datasets randomly and record the resulting error tolerance. This discourages any aggressive caching mechanism from anticipating block access patterns. Second, we randomly select 120 timesteps from the available data, so that the filesystem cannot aggressively cache future timesteps. Finally, we unmount and remount the filesystem containing the data before each invocation in order to reset the filesystem cache parameters.

8.1 Results

Figure 6 shows the difference in both engine and rendering time for adaptive and uniform datasets. The uniform data is rendered at the highest resolution. Both datasets used the same 8x8x8 spatial subdivision scheme.

As shown in Figure 6(a), the engine time for the adaptive resolution data is on average 50% faster than the engine time for the uniform resolution. In VisIt, the engine time is the total of the I/O time, the filter time, and the time to transfer the data to the viewer (renderer). No filter operations are used in our tests. The rendering time for the adaptive resolution data also saw significant improvements, with an average speedup of 100%. Figure 7 shows how the error varied over time during our testing phase. As expected, the data sets that had a higher absolute error had correspondingly more frequent use of low resolution grid chunks.

We expect to get even better results as the size of the data further increases. Once each timestep is too large to fit into memory, out-of-core rendering techniques must be used to render the high resolution images. The AR representation can reduce the size of the data to be rendered to keep the rendering in-core, and thus keep interactive rendering rates possible for such large data sets.

9. CONCLUSIONS AND FUTURE WORK

We have developed an adaptive resolution data model based on our multiresolution data model with error. We have implemented the software necessary to generate the multiresolution data and a dynamic adaptive resolution
data algorithm that builds an AR dataset at runtime from a tolerance specified by the user. The dynamic adaptive resolution algorithm performs an error analysis on each data chunk, and given a tolerance, assigns a resolution to each chunk that meets or exceeds that tolerance. The algorithm assembles the chunks and sends them to the rendering pipeline. We have shown that data represented by an adaptive resolution representation yields demonstrable performance gains in the I/O subsystem.

Our primary contributions include the dynamic adaptive resolution data model with error analysis, and the integration of our preliminary implementation with the VisIt visualization toolkit. By extending this toolkit, we can use existing uniform data rendering algorithms on our adaptive resolution data. The rendering algorithms in VisIt do not need to be aware of multiresolution data, error analysis, or dynamically generated adaptive resolution representations.

Future work calls for extending the data model to a support temporal based data, which is quite common for scientific datasets. Our multiresolution data model already supports temporal data, and we wish to extend our AR data model in an analogous manner to what we have done here with spatial data. In a temporal adaptive resolution data model, future timesteps may be partially or wholly eliminated from the time series based on a temporal error tolerance.
We also feel that to fully take advantage of an AR data model, data sizes must be too large to fit into main memory. Therefore we would like to test this approach in an out-of-core rendering environment. Although the current implementation of VisIt is not suitable for out-of-core rendering, we are actively researching environments for which we can render out-of-core images of both the multiresolution as well as the adaptive resolution data.

REFERENCES


