**FULL TITLE**
ASP Conference Series, Vol. **VOLUME**, **YEAR OF PUBLICATION**
**NAMES OF EDITORS**

# Visualization of Scalar Adaptive Mesh Refinement Data

G. H. Weber[1], V. E. Beckner[1], H. Childs[2], T. J. Ligocki[1], M. C. Miller[2], B. Van Straalen[1] and E. W. Bethel[1]

[1] *Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA*
[2] *Computing Applications and Research Department, Lawrence Livermore National Laboratory, Box 808, L-557, Livermore, CA 94551, USA*

**Abstract.** Adaptive Mesh Refinement (AMR) is a highly effective computation method for simulations that span a large range of spatiotemporal scales, such as astrophysical simulations, which must accommodate ranges from interstellar to sub-planetary. Most mainstream visualization tools still lack support for AMR grids as a first class data type and AMR code teams use custom built applications for AMR visualization. The Department of Energy's (DOE's) Science Discovery through Advanced Computing (SciDAC) Visualization and Analytics Center for Enabling Technologies (VACET) is currently working on extending VisIt, which is an open source visualization tool that accommodates AMR as a first-class data type. These efforts will bridge the gap between general-purpose visualization applications and highly specialized AMR visual analysis applications. Here, we give an overview of the state of the art in AMR scalar data visualization research.

## 1. Introduction

Adaptive Mesh Refinement (AMR) techniques combine the compact, implicitly specified structure of regular, rectilinear grids with the adaptivity to changes in scale of unstructured grids. In this paper, we focus on block-structured, *h-adaptive* (i.e., methods that produce dense grids in area with a large computational error) AMR techniques that represent the computational domain with a set of nested rectilinear grids or *patches* at increasing resolutions (Berger & Colella 1989) as opposed to tree-based refinement schemes such as the PARAMESH approach MacNeice et al. (2000). Figure 1. shows a simple example. Four regular patches are organized in three hierarchy levels. Grids belonging to a finer level are always completely enclosed by grids of the coarser levels.

Handling AMR data during visualization is challenging, since coarser information in regions covered by finer patches is superseded and replaced with information from these finer patches. During visualization it becomes necessary to manage selection of which resolutions are being used to avoid using conflicting data representations in overlapping regions. Furthermore, it is difficult to avoid discontinuities at level boundaries, which, if not properly handled, lead to visible artifacts in visualizations.

While AMR data can be node-centered, the majority of data sets we are currently visualizing use cell-centered values. This fact poses an additional chal-

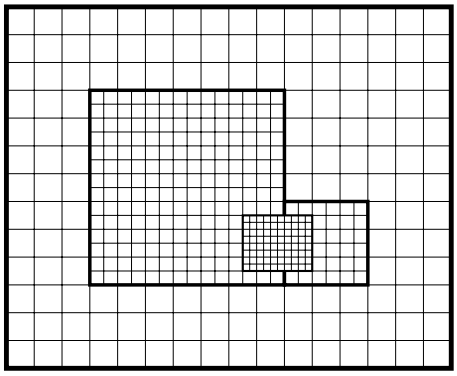

Figure 1. A simple Berger-Collela AMR hierarchy consisting of four patches organized in three hierarchy levels.

lenge since many visualization algorithms expect data in a node centered format. Despite the growing popularity of AMR simulations, little research has been done in effective visualization of AMR data. Furthermore, there is a lack of tools that treat AMR as first-class data type. In this paper we give an overview of the current state of visualization techniques for scalar AMR data with an emphasis on isosurface extraction.

## 2. Visualization Techniques for Scalar Data

### 2.1. Overview

Scalar quantities describe a variety of important physical characteristics such as temperature or pressure. Most simulations, including AMR simulations, include several scalar variables. Commonly used scalar data visualization techniques include false color/pseudocolor plots in conjunction with clipping or slicing, isosurface extraction, and direct volume rendering.

Pseudocolor plots use a color map to assign colors to scalar values and visualize a scalar field by coloring each position in the domain accordingly. A prominent example for this plot type are temperature maps in weather forecasts where colors depict temperature ranges or amount of precipitation. In three dimensions (3D), pseudocolor plots are commonly utilized in conjunction with slicing, where the plot is restricted to a planar slice through the data set, or clipping, i.e., cutting away part of the geometry, to reveal the interior of the domain. Spreadsheets, which are somewhat related to slicing planes, provide direct access to data value and are valuable for debugging and extracting data for further analysis with a wider range of tools such as Matlab or paper and pencil.

Isosurface extraction generalizes the concept of contour lines in topographic maps to three dimensions. A user specifies an isovalue, and a resulting isosurface is computed, which connects all locations in the domain where the scalar field assumes this isovalue.

Direct volume rendering extends the concept of pseudocolor plots by allowing a user to specify transparency as well as color information for scalar values. This makes it possible to hide entire "uninteresting" value ranges when generating an image. Rendered images depend on the choice of a light model. The commonly used emission and absorption model (Max 1995) assumes that the volume is filled with particles that emit light with a color as specified by the color map and absorb light according to the additional transparency information. Another light model approximates uses the gradient of a scalar variable as normal to approximate the appearance of surfaces in the volume (Levoy 1988). Volume rendering is very flexible in the range of achievable visualizations and can simulate both pseudocolor plots and isosurfaces. However, it is computationally more expensive than these techniques. Furthermore, unlike isosurface extraction, it does not yield an explicit representation of a surface to be used for further processing and analysis.

## 3. AMR Visualization by Conversion to Other Types

Initial work on AMR visualization focused on converting AMR data to suitable conventional representations, which are subsequently used for visualization. Norman et al. (1999) described a method for visualizing AMR data using using standard unstructured grid techniques. Their method converts an AMR hierarchy into an unstructured grid composed of hexahedral cells. This resulting grid is then visualized utilizing standard AVS, IDL, and VTK algorithms. By converting AMR data to an unstructured mesh, the implicit definition of grid connectivity is lost. Overhead resulting from the required separate storage of grid structure results in poor performance and does not scale well to large AMR data sets. Furthermore, this approach prohibits using of the hierarchical nature of AMR data for efficient visualization algorithms. Recognizing these fundamental problems, Norman et al. continue by extending VTK to handle AMR grids as first-class data structure. They have yet to publish detailed descriptions of their techniques.

## 4. Crack-free Isosurface Extraction from AMR Data

### 4.1. Isosurface Extraction with Marching Cubes

Lorensen & Cline (1987) introduced the marching cubes (MC) algorithm for isosurface extraction, which has become the de-facto method for isosurface extraction in scientific visualization. MC operates on node centered data given on regular rectilinear grid. MC "marches" all cells one-by-one and constructs a triangulation approximating the isosurface in each individual "cube" cell. This construction is performed locally and only depends on the values at the vertices of the current cell.

An important property of the MC methods, which leads to problems for multiresolution data, is that it only considers function behavior at grid vertices and along cell edges when constructing an isosurface. The algorithm classifies each vertex of a given cell as either lying inside or outside the isosurface. Assuming that the same inside/outside classification of vertices will always use the

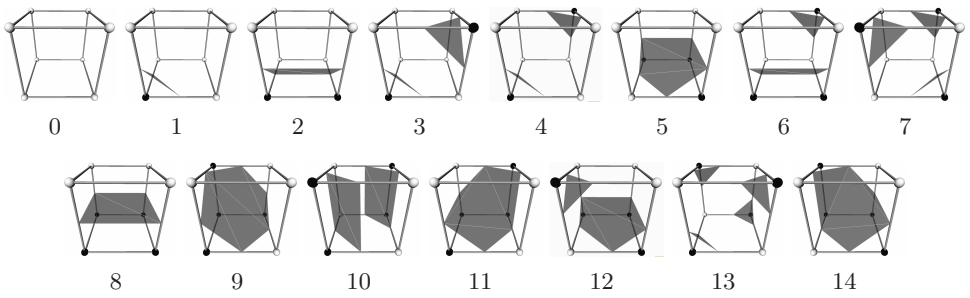


Figure 2. All 256 entries for the LUT used by MC can be constructed from the shown 15 base cases by using rotational symmetry and inversion. The vertex configuration, i.e, which vertices are inside (shown as black spheres in the figure) and outside (shown as white spheres in the figure) the isosurface, determines alone what triangulation is used by the algorithm.

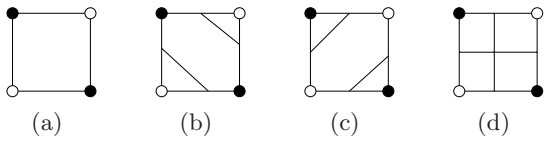


Figure 3. An ambiguous face configuration consists of vertices alternating between being inside and outside of the isosurface (a). Three different contour topologies on that face are possible: separating vertices outside the contour (b), separating vertices inside the contour (c), or separating all vertices (d) resulting in a non-manifold contour.

same triangulation of the isosurface within a cell, MC stores these triangulations in a lookup table (LUT). Using rotational symmetry and the assumption that inverting inside and outside classification yields the same triangulation, it is possible to reduce the 256 entries of this LUT to the fifteen base cases shown in Figure 2. All triangulations stored in the LUT reference intersection points along the cell's edges as vertices of the produced triangulation. These intersection points are computed assuming the functions varies linearly along an edge.

A problem in the original MC algorithm, which was originally pointed out by Dürst (1988), can lead to cracks in the isosurface. For certain configurations, contour topology on a cell face (or inside a cell) is not determined uniquely by the vertex vertex configuration. Faces with vertices whose classifications alternate between inside and outside the isosurface cause ambiguity problems, see Figure 3. Three cases are possible: Contours can separate vertices outside the isosurface, see Figure 3(b), vertices inside the isosurface, see Figure 3(c), or all vertices resulting in a non-manifold contour, see Figure 3(d).

All basis configurations of Lorensen and Cline's paper separate the vertices of an ambiguous face that are inside the isosurface. An inverse case (i.e., a case with inverted inside and outside vertex classifications) uses the same triangulation. Consequently, triangulations produced by inverting a base case will

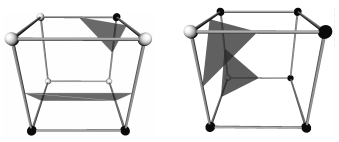


Figure 4. Adjacent cell configuration resulting in a crack in an isosurface extracted by an unmodified MC approach. Since the two triangles in the case $\overline{3}$ cube separate vertices outside the isosurface, the "hole" in the isosurface on the right side of the case 6 cell is not closed up properly.

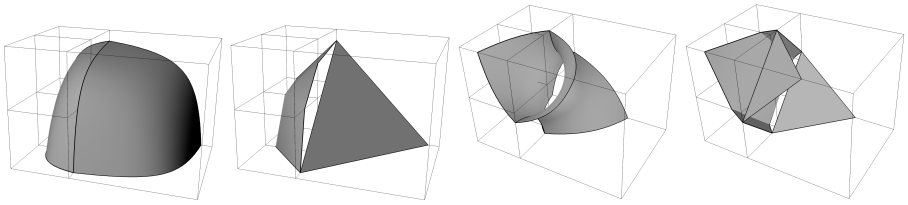


Figure 5. (a, b): The linear approximation of contours on boundary faces used by the MC method (b) can lead to cracks in an isosurface at coarse-fine level boundaries, even if values at dangling nodes are chosen to be consistent with the coarse level. (c, d): Ambiguous faces at coarse-fine level boundaries can lead also lead to cracks in an extracted isosurface if the finer representation resolves the ambiguity.

separate vertices outside the isosurface. If such a cell and a cell whose triangulation is derived from a basis configuration without using inversion share an ambiguous face, a crack in the extracted isosurface triangulation arises, see Figure 4.

Montani, Scateni & Scopigno (1994) proposed an extended set of basis configurations. For each MC basis configuration containing an ambiguous face, i.e., basis configurations 3, 6, 7, 10, 12, and 13, they add its inverse to the set of basis configurations. The LUT created by their method, which is used, for example, in the Visualization Toolkit (VTK) by Schroeder, Martin & Lorensen (1998), consistently separates vertices outside the isosurface, preventing holes arising due to inconsistent topology on faces between cells. Other approaches use additional information to determine the contour topology uniquely on ambiguous faces or inside a cell. Nielson (2003) provides an in-depth analysis for possible configurations for trilinear interpolation.

### 4.2. Applying the Marching Cubes Method to AMR Data

AMR data is particularly difficult to handle when visualizing scalar fields via isosurface extraction. This difficulty is due to the fact that AMR often uses a cell centered data format while the marching cubes algorithm (Lorensen & Cline 1987), which is de-facto standard isosurface extraction algorithm in scien-

tific visualization, expects values at the vertices of a grid. Furthermore, when extracting an isosurface using the marching cubes method, t-junctions can lead to visible cracks in an isosurface, even if dangling nodes (i.e., nodes that are present in the fine level but not in the coarse level) have values that are consistent with the coarse level representation, see Figure 5 (a, b). Figure 5(a) shows an isosurface at a coarse-fine boundary where values at dangling nodes are identical to the values that interpolation in the coarse level assigns to their location. In theory, this configuration should lead to a consistent, crack-free isosurface. However, as described in Section 4.1., the MC algorithm approximates curved contours on boundary faces with line segments that are the edges of a triangulation, see Figure 5(b). Since a single cell in the coarse level shares a boundary face with four cells of the finer level, this approximation leads to a mismatch between isosurface approximations in the coarse level, where the curvilinear contour is approximated by a single line segment, and in the fine level, where the same contour is approximated by a poly-line consisting of three line segments. The result is a visible crack in the extracted isosurface.

Shared ambiguous faces are another source of more subtle cracks, as illustrated in Figure 5 (c, d). Here it is possible that the "subdivision" of the boundary in the finer level resolves an ambiguous face in the coarse level. Since most MC implementations use implicit disambiguation (Montani et al. 1994), such disambiguation can result in inconsistent isosurface topology in coarse and fine hierarchy level. For example, the isosurface component in the coarse-level cell in Figure 5(d) separates shared-face vertices that are outside the isosurface, while the isosurface component in the fine-level cells separates shared-face vertices that are inside the isosurface.

A simple scheme to apply the MC algorithm to cell centered AMR data, which is commonly used by visualization tools, re-centers the grid by performing a resampling step that computes values at node positions. Subsequently, isosurface extraction via MC uses the resulting grid. This approach leads to cracks, which can be handled in a variety of ways. Most currently available visualization tools, including ParaView and VisIt simply ignore cracks in the isosurface. Alternatively, it is possible to use techniques that have been developed for other (most commonly Octree-based) multiresolution data representations. Shu, Zhou & Kankanhalli (1995) introduced an adaptive marching cubes implementation and proposed to patch cracks in an isosurface by filling them with an appropriate triangulation. However, while this strategy works for cracks resulting from representing the same curvilinear at different levels of resolution (Figure 5 (a, b)), it can lead to non-manifold (not locally flat) surfaces when applied to cracks that result when a finer level resolves an ambiguity in the coarser level (Figure 5 (c, d)). Furthermore, the small triangles that result from this approach reduce the quality of an isosurface triangulation. Shekhar et al. (1996) developed a multi-resolution MC approach that removes cracks by adjusting adjusting the additional finer-level triangulation vertices to coincide with the line-segment approximation in the coarse level. Westermann, Kobbelt & Ertl (1999) modified this approach by adjusting replacing triangles in the coarse cell by a triangle fan that matches up with the polyline in the finer level. All these approaches fail, if a crack results from the resolution of an ambiguous face in the finer level (Figure 5 (c, d)). While it is possible to avoid this situation by utilizing a MC implementation that utilizes additional information to resolve ambiguities in the coarse

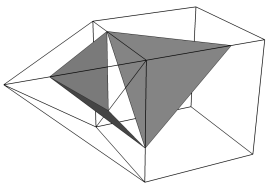

Figure 6. If the tessellation filling the gaps between hierarchy levels, which result from the use of dual grids, does not share complete boundary faces with rectilinear cells of the dual grid, cracks in an isosurface can result. This is a result of the fact, that the same curved contour on the boundary is represented at different detail, i.e., a single line on the rectilinear face and a polyline consisting of two line segments on the two triangular faces.

level (Nielson 2003), no widely available visualization tools uses this approach, since it imposes additional computational overhead.

Weber et al. (2001b, 2003a) developed a method that extracts crack-free isosurfaces from cell centered AMR data by interpreting cell centers of each patch of the AMR hierarchy as the vertices of a new patch, which is the *dual grid* to the original patch. Within these dual grids, isosurfaces are extracted utilizing the standard marching cubes method. The use of dual grids leads to gaps between different levels of an AMR hierarchy. Weber et al. use a procedural scheme to fill these gaps with "stitch" cells (tetrahedra, pyramids, triangle prisms and deformed cubes) ensuring that this step produces no t-junctions, since these t-junctions would again lead to cracks in a resulting isosurface, see Figure 6. Subsequently, they extract isosurface portions within gaps between hierarchy levels utilizing the marching cubes methods by giving appropriate case tables for these new cell types.

Fang et al. (2004) presented an alternate isosurface extraction approach for node centered AMR data. Their main goal is preserving the original patch structure and "identity" of cells, enabling a user to determine to what particular patch cell a triangle of an isosurface belongs. Their method achieves this goal by extending refined patches until it is possible to assign values to dangling nodes that are consistent with interpolation results in the coarser level. Subsequently, they decompose coarse-level cells at the boundary to a finer level into a set of pyramids that connect the cell center with all boundary faces. For each "facet" of a subdivided face, i.e., a face at the boundary to a finer level, a separate pyramid is created, ensuring that marching cubes will not produce cracks in an extracted isosurface.

Meshless methods are an alternative way to extract crack-free isosurfaces from multiblock data. Co, Porumbescu & Joy (2004) presented an isosurface extraction scheme that uses radial basis functions to define a continuous interpolant for multiblock data. Subsequently they extract a set of sample points that lie on the isosurface and use splatting to render the isosurface.

### 4.3. Volume Rendering of AMR Data

Max (1993) described sorting schemes for cells during volume rendering including one method specifically geared toward AMR data. Ma (1999) described

and compared two approaches for rendering of structured AMR data using the PARAMESH framework. A PARAMESH MacNeice et al. (2000) hierarchy organizes grids as blocks in a quadtree (in 2-d space) or an octree (in 3-d space) structure. Inner nodes of this tree correspond to regions that need further refinement while leaf nodes specify a grid whose resolution is given by the current hierarchy level. Ma described two approaches for volume rendering of AMR data. One method resamples a hierarchy on an uniform grid at the finest resolution. The resulting grid is evenly subdivided and each part rendered in parallel on a separate processor. Partial images are combined using binary-swap composition.

A second method preserves the AMR structure. Individual blocks (leaves of the octree) are distributed among the processors in a round-robin fashion to achieve static load balancing. Since a block structure can lead to many small ray-segments, Ma buffers these segments into larger messages to decrease communication overhead. Individual blocks are rendered using ray-casting. Two sampling schemes are used: A simple approach using a fixed, constant sample distance and an adaptive approach that decreases sample distance in finer resolution blocks.

Weber et al. (2001a) described an interactive, hardware accelerated volume rendering approach to generate previews of AMR data and a higher-quality software approach based on cell projection. Both approaches use data duplicated in coarser hierarchy levels as a less accurate approximation for the data in finer levels. The hardware-based approach uses a k-d-tree-like structure to partition an AMR hierarchy into blocks of homogeneous resolution and renders these blocks in back-to-front order. Based on view-dependent criteria (e.g., the number of pixels covered by a voxel) and a measured rendering time for the current frame, the traversal depth into the individual patches of the AMR hierarchy is chosen to achieve interactive rendering rates.

Weber et al. (2001a) also described a software cell-projection-based approach to render AMR data sets in higher quality. While rendering a level of an AMR hierarchy, additional information is stored for each pixel that makes it possible to "replace" the contribution of those parts of the domain that are refined by another hierarchy level with a more accurate representation, supporting progressive rendering of AMR data sets. In later work, Weber et al. (2001c) used the dual mesh and stitch cells introduced for isosurface extraction Weber et al. (2003a) to define a $C^0$ continuous interpolation scheme and utilized this interpolation method in their progressive cell-projection rendering approach.

Kreylos et al. (2002) described a framework for remote, interactive rendering of AMR data. The framework consists of a "lightweight" viewer and a renderer running on one or several remote machines. The method of Kreylos et al. "homogenizes" an AMR hierarchy, i.e. partitions it in blocks of constant resolution using a k-d tree. Resulting blocks of constant resolution are distributed among processors and rendered using either a texture-based hardware-accelerated approach or a software-based cell-projection renderer. Two distribution strategies are implemented: One strategy attempts to distribute cost evenly among processors, the other variant tries to minimize data duplication. Weber et al. (2003b) built on this work and compared various AMR partitioning strategies for parallel volume rendering of AMR data.

Kähler & Hege (2002) introduced a method that partitions Berger-Colella AMR data into homogeneous resolution regions and visualizes it using texture-based hardware-accelerated volume rendering. Their partitioning scheme uses a heuristic that is based on assumptions concerning the placement of refining grid to minimize the number of constant-resolution blocks. Generally, this approach generates fewer blocks than the approach described by Weber et al. (2001a) and the approach developed by Kreylos et al. (2002). Subdivision into a smaller number of blocks is beneficial when data is rendered on a single machine.

In later work, Kähler et al. (2002) used a set of existing tools to render results of a simulation of a forming star using the framework developed by Bryan (1999). They define camera paths within a CAVE environment using the *Virtual Director* virtual reality interface. Subsequently, they render animations of the AMR simulation utilizing their previously developed hardware-accelerated volume rendering approach Kähler & Hege (2002). To enhance depth perception, rendered images are augmented with a background that is obtained by rendering a particle simulation of the formation of the early universe. Recently, Kähler et al. (2006) implemented a GPU-based ray-casting approach for AMR data, which improves rendering quality considerably compared to slicing-based approaches and supports a more complex light model with wavelength dependent absorption.

By specifying a transfer function, and a range of isovalues, Park, Bajaj & Siddavanahalli (2002) produced volume-rendered images of AMR data based on hierarchical splatting, see Laur & Hanrahan (1991). Their method converts an AMR hierarchy to a k-d-tree structure consisting of blocks of constant resolution. Each node of this k-d tree is augmented with an octree. Octree and k-d-tree nodes contain a 32-bit field, where each bit represents a continuous range of isovalues. Using the k-d tree and the octree, regions containing values within the specified range are identified and rendered back-to-front using hierarchical splatting.

Kaehler et al. (2005) described a framework for visualization of time-varying AMR data. Their method addresses the problem that most AMR simulations update finer AMR patches more frequently than coarse patches. Considering two subsequent time steps, their interpolation scheme first ensures that both time-steps have the same refinement configuration, i.e., that each cell that is refined in one time step is also refined in the other time step. Values for cells that are not refined in the current time step but the other are obtained by interpolation. Subsequently, they define an interpolation scheme to compute intermediate values in regions that are covered by coarser level and thus, updated less frequently. In addition to this interpolation scheme, their framework automatically handles remote data access and computes interpolated values on the machine, which also runs the simulation.

## 5. Visualization of AMR Data with VisIt

VisIt Childs & Miller (2006) is a richly featured visualization and analysis tool for large data sets. It employs a client-server model where the server is parallelized. The nature of parallelization is data parallel; the input data set is partitioned among VisIt's processors. In the case of AMR data, each patch is treated as an

atomic unit and assigned to one of the processors on VisIt's parallelized server. For example, patches "level zero, patch zero" and "level one, patch two" may be assigned to the server's processor zero, while patches "level zero, patch one," "level one, patch zero," and "level one, patch one" may be assigned to processor one.

Visualization and analysis of massive scale data sets is an important use case for VisIt. As such, it employs many optimizations to enable the processing of this data. For example, VisIt is able to use spatial extents meta-data to reduce the amount of data that is processed. When a slice of a three-dimensional data set is being rendered, VisIt is able to to limit the patches processed to those that actually intersect the slice. Although this functionality may sound straight forward, it is difficult to implement in a richly featured, module based framework. More information about the contract methodology that enables these optimizations can be found in Childs et al. (2005).

VisIt's handling of AMR data is made possible by marking coarse cells that are refined at a lower level as "ghost." This marking is done by adding an array to each patch that designates the status of each cell (ghost or non-ghost). Most algorithms can ignore the ghost markings; they operate identically on ghost and non-ghost cells. One advantage of using the ghost cells is that it allows structured grids to retain their native form. That is, removing the ghost cells before applying visualization algorithms would create a grid that was no longer structured. The resulting grid would often be unstructured and that unstructured grid could have a memory footprint that is an order of magnitude larger. Another advantage of using ghost cells is that they allow for proper interpolations to take place, which would not be possible if refined cells were removed before applying visualization algorithms. After all algorithms have been applied, a module walks the data set and removes all cells or geometry resulting from a ghost cell.

VisIt employs the standard Marching Cubes algorithm to contour data. Most AMR data is cell-centered, requiring interpolation to the nodes. For hanging nodes at the boundary of patches at different refinement levels, this interpolation is done incorrectly in VisIt and cracked isosurfaces can result. However, VisIt does *not* produce cracked isosurfaces when abutting patches are at the same refinement level. In this case, VisIt can create a layer of ghost cells around each patch that contains the values of neighboring cells from the other patches. These ghost cells allow for correct interpolation to take place, meaning that a consistent contouring takes place from patch to patch and no cracks are created in this case.

VisIt's employs a data parallel volume rendering scheme that is able to resolve the types of complex sorting issues that arise in unstructured meshes Childs et al. (2006). AMR meshes present a special type of load balancing challenge for VisIt's volume rendering algorithm, however. The running time of the algorithm is dependent on the amount of data and the amount of samples. For AMR meshes, the patches at the coarser refinement levels occupy a larger spatial footprint, and, as such, often cover a much larger portion of the picture and contribute more samples. Hence, sampling the patches at coarser refinement levels typically takes much longer than the sampling for patches at finer refinement levels. VisIt attempts to counteract this problem by minimizing the amount

of patches at the coarse refinement levels on any given processor. In terms of additional AMR handling, the volume rendering algorithm ignores all samples from cells that are marked ghost. So if a sample point is contained by many patches at different refinement levels, only the value at the finest level will be accepted, since all other levels will have the corresponding cell marked as ghost.

A trend of increasing importance is where visualization and analysis capabilities are coupled in one production quality application. Here, analysis means computing statistical moments of subsets of AMR hierarchies, distribution functions, computed/derived quantities, temporal analysis, etc. VisIt provides a rich set of analysis capabilities (such as integrating density over volume to obtain mass, calculating volumes, surface areas, and moments of inertia), all of which execute in parallel.

## 6. Ongoing and Future Work

We are currently working on extending VisIt's visualization capabilities for AMR data. To this end, we had extensive meetings with members of the LBNL Applied Numerical Algorithms Group (ANAG) and the LBNL Center for Computational Sciences and Engineering (CCSE). We optimized handling of AMR grids in VisIt. These optimizations can save on memory by a factor of ten and also support more efficient rendering. Additional performance and memory optimizations improve efficiency for the important use case of rendering patch boundaries. VisIt previously used very general algorithm that was unnecessarily slow. Our new, specialized algorithm is an order of magnitude faster and more memory efficient. We also added support to link picking capabilities and spreadsheet support (duplicating functionality present in ChomboVis). The request for this functionality indicates that visualization is still used frequently for debugging. On the other hand there is a growing need for data analysis and visualization capabilities to interpret the results of production AMR simulations. For example, we are currently working on adding line integral convolution-based vector field visualization capabilities to VisIt.

**Acknowledgments.** This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET). We thank the members of the LBNL Visualization Group, the LBNL ANAG, the LBNL CCSE, and the VisIt development team.

### References

Berger, M., & Colella, P. 1989, Journal of Computational Physics, 82, 64

Bryan, Greg L. 1999, Computing in Science and Engineering, 1(2), 46

Childs, H., Brugger, E.S., Bonnell, K. S., Meredith, J.S., Miller, M., Whitlock, B. J., & Max, N. 2005, IEEE Visualization 2005, 190

Childs, H., Duchaineau, M. A., & Ma K.-L. 2006, Eurographics Symposium on Parallel Graphics and Visualization, 153

Childs, H., & Miller, M. 2006, SpringSim High Performance Computing Symposium (HPC 2006), 181

Co, C. S., Porumbescu, S. D., & Joy, K. I. 2004, Data Visualization (Proceedings of VisSym 2004), 273
Dürst, M. J. 1988, Computer Graphics, 22(2), 72
Fang, D. C., Weber, G. H., Childs, H. R., Brugger, E. S., Hamann, B., & Joy, K.I 2004, Proceedings of 2004 Hawaii International Conference on Computer Sciences (DVD-ROM conference proceedings), ISSN 1545-6722, 216
Kaehler, R., Prohaska, S., Hutanu, A., & Hege, H.-C. 2005, IEEE Visualization 2005, 175
Kähler, R., Cox, D., Patterson, R., Levy, S., Hege, H.-C., & Abel, T. 2002, IEEE Visualization 2002, 537
Kähler, R., & Hege, H.-C. 2002, The Visual Computer 18(8), 481, also: Zuse Institut Technical Report ZR-01-30
Ralf, K., Wise, J., Abel, T., & Hege, H.-C. 2006, Proceedings of Volume Graphics, 103
Kreylos, O., Weber, G. H., Bethel, E. W., Shalf, Hamann, B., & Joy, K.I. 2002. Lawrence Berkeley National Laboratory Technical Report LBNL 49954
Laur, D., & Hanrahan, P. 1991, Computer Grahpics (Proceedings of ACM SIGGRAPH 91), 25(4), 285
Levoy, M. 1998, IEEE Computer Graphics and Applications, 8(3), 29
Lorensen, W. E., & Cline, H. E. 1987, Computer Graphics (Proceedings of ACM SIGGRAPH 87), 21(4), 163
Ma, K.-L. 1999, Proceedings of Frontiers '99 the Seventh Symposium on the Frontiers of Massively Parallel Computation, 138
MacNeice, P., Olson, K. M., Mobarry, C., de Fainchtein, R. & Packer C. 2000, Computer Physics Communications, 126(3), 330
Max, N. L., 1993, Focus on Scientific Visualization (Springer-Verlag), 259
Max, N. L., 1995, IEEE Transactions on Computer Graphics, 1(2), 99
Montani, C., Scateni, R., & Scopigno, R. 1994 The Visual Computer, 10(6), 353
Nielson, G. M. 2003, IEEE Transactions on Visualization and Computer Graphics, 9(3), 341
Norman, M. L., Shalf, J. M., Levy, S., & Daues, G. 1999, Computing in Science and Engineering, 1(4), 36
Park, S., Bajaj, C., & Siddavanahalli, V. 2002, IEEE Visualization 2002, 521
Schroeder, W. J., Martin, K. M., & Lorensen, W. E. 1998, The Visualization Toolkit, 2nd edn (Prentice-Hall)
Shekhar, R., Fayyad, E., Yagel, R., & Cornhill, J. F. 1996, IEEE Visualization 1996, 335
Shu, R., Zhou, C., & Kankanhalli, M. S. 1995 The Visual Computer, 11(4), 202
Weber, G. H., Hagen, H., Hamann, B., Joy, K. I., Ligocki, T. J., Ma, K.-L., & Shalf, J. M.A 2001a, Proceedings of the SPIE (Visual Data Exploration and Analysis VIII), 4302, 121
Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., & Joy, K. I. 2001, Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization 2001, 25.
Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., & Joy, K. I. 2003a, Hierarchical and Geometrical Methods in Scientific Visualization (Springer Verlags), 19.
Weber, G. H., Kreylos, O., Liogcki, T. J., Shalf, J. M., Hagen, H., Hamann, B., Joy, K.-I., & Ma, K.-L. 2001c, Proceedings of Vision, Modeling, and Visualization 2001, 121.
Weber, G. H., Öhler, M., Kreylos, O., Shalf, J. M., Bethel, E. W., Hamann, B., Scheuermann, G., Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003, 51.
Westermann, R., Kobbelt, L., & Ertl, T., The Visual Computer 15(2), 100.