Extraction of Crack-free Isosurfaces from Adaptive Mesh Refinement Data

Gunther H. Weber[1,2,3], Oliver Kreylos[1,3], Terry J. Ligocki[3], John M. Shalf [3,4], Hans Hagen[2], Bernd Hamann[1,3], and Kenneth I. Joy[1]

[1] Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California, Davis
[2] Department of Computer Science, University of Kaiserslautern, Germany
[3] National Energy Research Scientific Computing Center (NERSC), Lawrence Berkeley National Laboratory, Berkeley
[4] National Center for Supercomputing Applications (NCSA), University of Illinois, Urbana-Champaign

Abstract. Adaptive mesh refinement (AMR) is a numerical simulation technique used in computational fluid dynamics (CFD). It permits the efficient simulation of phenomena characterized by substantially varying scales in complexity of local behavior of certain variables. By using a set of nested grids at different resolutions, AMR combines the simplicity of structured rectilinear grids with the possibility to adapt to local changes in complexity and spatial resolution. Hierarchical representations of scientific data pose challenges when isosurfaces are extracted. Cracks can arise at the boundaries between regions represented at different resolutions. We present a method for the extraction of isosurfaces from AMR data that avoids cracks at the boundaries between levels of different resolution.

1 Introduction

AMR was introduced to computational physics by Berger and Oliger [3] in 1984. A modified version of their algorithm was published by Berger and Colella [2]. AMR has become increasingly popular in the computational physics community, and it is used in a variety of applications. For example, Bryan et al. [4] use a hybrid approach of AMR and particle simulations for simulation of astrophysical phenomena.

Fig. 1 shows a simple two-dimensional (2D) AMR hierarchy produced by the Berger–Colella method. The basic building block of a d–dimensional Berger-Colella AMR hierarchy is an axis-aligned structured rectilinear grid. Each grid g consists of hexahedral cells. Each grid can be positioned by specifying its origin $\boldsymbol{o}_g$. The underlying simulation method is a finite-difference method. Typically, a *cell-centered* data format is used, i.e., dependent function values are associated with the centers of the cells. We denote the region covered by the grid by Γ_g. Each grid contains a pointer to an array containing the dependent data values. These are stored in a simple array

An AMR hierarchy consists of several levels Λ_l comprising one or multiple grids. All grids in the same level have the same resolution, i.e., all grids in a level share the same cell size $\boldsymbol{\delta}_{\Gamma_l}$. The region covered by a level Γ_{Λ_l} is the union of regions covered by the grids of that level.

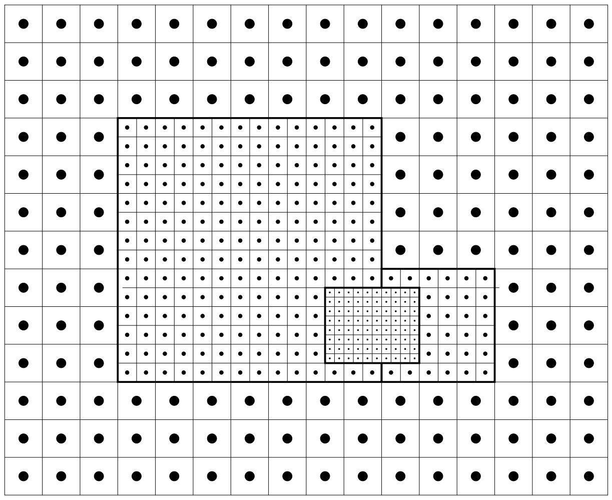


Fig. 1. AMR hierarchy consisting of four grids in three levels. The root level consists of one grid. This grid is refined by a second level consisting of two grids. A fourth grid refines the second level. It overlaps both grids of the second level. Boundaries of the grids are drawn as bold lines. Locations at which dependent variables are given are indicated by solid discs

The hierarchy starts with the *root level* Λ_0, the coarsest level. Each level Λ_l may be refined by a finer level Λ_{l+1}. A grid of the refined level is commonly referred to as a *coarse grid* and a grid of the refining level as a *fine grid*. The *refinement ratio* r specifies how many fine grid cells fit into a coarse grid cell, considering all axial directions. The value of r is always a positive integer. A refining grid refines an entire level Λ_l, i.e., it is completely contained in Γ_{Λ_l} but not necessarily in the region covered by a single grid of that level. Each refining grid can only refine complete grid cells of the parent level, i.e., it must start and end at the boundaries of grid cells of the parent level. Furthermore, there is always a layer with a width of at least one grid-cell between a refining grid and the boundary of the refined level. Due to the hierarchical nature of AMR simulations, the resulting data lend themselves to hierarchical visualization. We discuss a new method for the direct extraction of isosurfaces from AMR data sets.

2 Related Work

Little research has been published regarding the visualization of AMR data. Norman et al. [12] convert an AMR hierarchy into finite-element hexahedral cells with cell-centered data that can take advantage of standard visualization tools (like AVS [1], IDL [7], or VTK [13]), while preserving the hierarchical nature of the data. Ma [9] describes a parallel rendering approach for AMR data. Even though he re-samples the data to vertex-centered data, he still uses the hierarchical nature of AMR data and contrasts it to re-sampling it to the highest resolution-level available. Max [10] describes a sorting scheme for cells for volume rendering, and uses AMR data as one application of his method.

Isosurface extraction is a commonly used technique for the visual exploration of scalar fields. Our work is based on the marching-cubes (MC) method, introduced by Lorensen and Cline [8]. A volume is traversed cell-by-cell, and the part of the isosurface within each cell is constructed using a look-up table (LUT). The LUT of the original article by Lorensen and Cline contained a minor error that could lead to cracks in the extracted isosurface. This is due to ambiguous cases where different isosurface triangulations in a cell are possible. Nielson and Hamann [11], among others, addressed this problem and proposed a solution to it. Van Gelder and Wilhelms [5] have provided a survey of solutions to this problem. They show that, in order to extract a topologically correct isosurface, more than one cell must be considered at a time. If topological correctness of the isosurface is not required, it is possible to avoid cracks without looking at surrounding cells. In our implementation, we use the LUT from VTK [13] that avoids cracks by taking special care during LUT generation.

Octree-based methods are among the methods used to speed up the extraction of isosurfaces. Shekhar et al. [14] use an octree as a hierarchical representation of the data. By adaptively traversing the octree and merging cells that satisfy certain criteria, they reduce the amount of triangles generated for an isosurface. Their scheme removes the cracks in the resulting isosurface. Westermann et al. [15] modified this approach by adjusting the traversal criteria and improving the crack-removal strategy. Gross et al. [6] used a combination of wavelets and quadtrees to approximate surfaces, e.g., from terrain data. Using an estimate based on a wavelet transform their approach chooses a level in the quadtree structure to represent a given region. Handling transitions between quadtree levels is similar to handling those between levels in an AMR hierarchy.

3 Dual Grids

The MC method assumes that data values are associated with the cell vertices, but the AMR method produces values at cell centers. To deal with this incompatibility problem one can, for example, re-sample the data set to a vertex-centered format. However, re-sampling causes "dangling nodes" in the fine level. Even if the re-sampling scheme assigns the same values to the dangling nodes as the interpolation scheme assigns to them in the coarse level, dangling nodes can cause cracks when using the MC method (see [15]). We solve these problems by using a *dual grid* for isosurface extraction. This dual grid is defined by the function values at the cell centers. The cell centers become the vertices of the vertex-centered dual grid.

The dual grids for the first two levels of the AMR hierarchy shown in Fig. 1 are shown in Fig. 2. We note that the dual grids have "shrunk" by one cell in each axial direction with respect to the original grid. The result is a gap between the coarse grid and the embedded fine grids. Due to the existence of this gap, there are no dangling nodes that could cause discontinuities in an extracted isosurface.

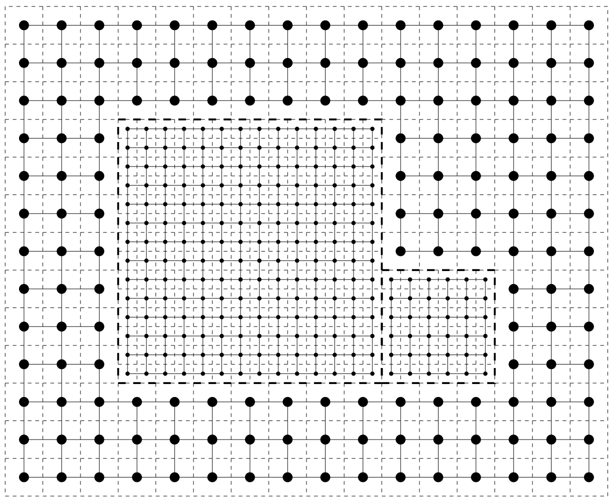


Fig. 2. Dual grids for the three AMR grids comprising the first two hierarchy levels shown in Figure 1. The original AMR grids are drawn in dashed lines and the dual grids in solid lines

4 Stitching 2D Grids

To avoid cracks in extracted isosurfaces as a result of gaps between grids, a tessellation scheme is needed that "stitches" grids of two different hierarchy levels. The resulting *stitch mesh* is constrained by the boundaries of the coarse and the fine grids and can be used to merge levels seamlessly. The stitch mesh must not subdivide any boundary elements of the existing grids. In the 2D case, this is achieved by requiring that only existing vertices are used and no new vertices generated. Since one of the reasons for using the dual grids is to avoid the insertion of new vertices, whenever possible, this poses no problems.

In the 2D case, a constrained Delaunay triangulation can be used to fill the gaps between grids. For two reasons, we chose not to do this. While in the 2D case only edges must be shared between the stitching grid and the dual grids, entire faces must be shared in the 3D case. The boundary faces of rectilinear grids are quadrilaterals and cannot be shared by tetrahedra without being subdivided, thus causing cracks when used in an MC-based isosurface extraction scheme. Furthermore, an index-based approach is more efficient, since it takes advantage of the regular structure of the boundaries while avoiding problems that might be caused by this regular structure when using a Delaunay-based approach.

The stitching process for a refinement ratio of two is shown in Fig. 3. Stitch cells must be generated for edges along the boundary and for the vertices of the fine grid. The stitch cells generated for the edges are shown in dark grey, while the stitch cells generated for the vertices are drawn in light grey. For the transition between one fine and one coarse grid, each edge of the fine grid is connected alternatingly to either a vertex or an edge of the coarse grid. This yields triangles and deformed quadrilaterals

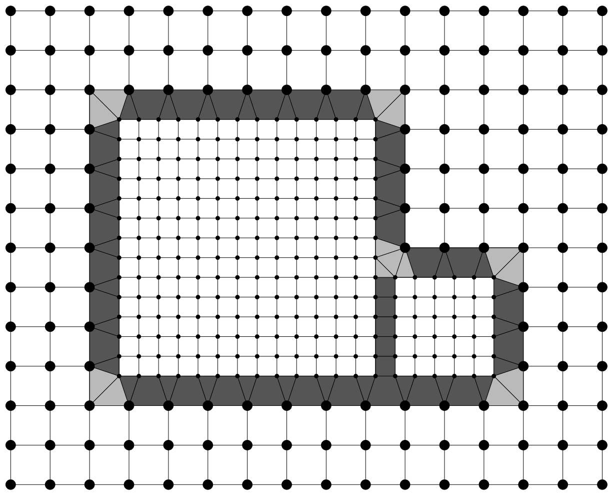


Fig. 3. "Stitch cells" for first two levels of AMR data set shown in Fig. 2

as additional cells. The quadrilaterals are not subdivided, since such a subdivision is not unique. (This in turn would cause problems in the 3D case when these quadrilaterals become boundary faces shared between cells.) The vertices are connected to the coarse grid via two triangles. Here, a consistent partition of the deformed quadrilateral is possible. The obvious choice is to connect each edge to the two coarse edges that are "visible" from it.

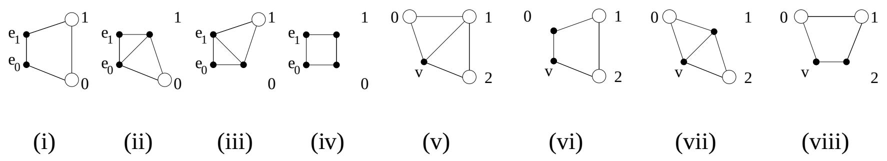


Fig. 4. Possible cases for connecting a boundary edge $\overline{e_0 e_1}$ ((i)–(iv)) or a boundary vertex v ((v)–(viii)) to a coarse grid. If cells of the coarse grid are refined, the coarse grid points (circles) are replaced by the corresponding refining point (solid black discs)

In the case of multiple grids, a check must be performed: Are the grid points in the coarse grid refined or not? If a fine edge is connected to a coarse point, this check is simple. If the coarse point is refined, the fine edge must be connected to another fine edge; this yields a rectilinear instead of a triangular cell. The case of connecting to a coarse edge is more complicated and illustrated in Figs. 4 (i)–(iv). If both points are refined (Fig. 4(iv)), the fine edge is connected to another fine edge. As a result, adjacent fine grids yield the same cells as a continuous fine grid. The problem cases occur where only one of the points is refined (Figs. 4(ii) and 4(iii)). Even though it is possible to skip these cases and handle them as vertex cases of the other grid, a more consistent approach is to include them in the possible edge cases. However, the same tessellations should be generated for both cases, as shown in Fig. 4.

The cases arising from connecting a vertex are illustrated in Figs. 4 (v)-(vii). In addition to replacing refined coarse grid points by the nearest fine-grid point, adjoining grids must be merged. If either of the coarse grid points 0 (Fig. 4(v)) or 2 (Fig. 4(v)) is refined, it is possible to "promote" the border vertex to a border-edge segment by connecting it to the other refined grid point and treating it as an edge, and using the connection configurations from the previous paragraph, i.e., those shown in Fig. 4 (i)–(iv). (This case occurs along the bottom edge of the fine grids shown in Fig. 3.)

Even though general integer-refinement ratios r are possible for AMR grids, in 2D simulations, refinement ratios of two and four are the most common ones used. The stitching process can be generalized to more general refinement ratios. Instead of connecting edge segments of the refining grid alternatingly to a coarse-grid edge segment and point, $(r-1)$ consecutive edge segments must be connected to one common coarse-grid point. Every r–th fine edge must be connected to a coarse edge. Even though the valence of the grid points of the coarse grid is increased, this is not a problem with the commonly used refinement ratios. Furthermore, general refinement ratios do not add more refinement configurations, since the fundamental connection strategies remain the same.

5 Stitching 3D Grids

Our index-based approach can be generalized to 3D AMR grids. In the simple case of one fine grid embedded in a coarse grid, quadrilaterals, edges and vertices of the fine grid must be connected to the coarse grid. In each of the two directions implied by a quadrilateral, a decision must be made whether to connect to a vertex or an edge. The various combinations result in quadrilaterals being connected to either a vertex, a line segment (in the two possible directions) or another quadrilateral. The cell types resulting from these connections are pyramids (Fig. 5(i)), deformed triangle prisms (Fig. 5(ii), and deformed hexahedral cells (Fig. 5(iii)).

The edge case can be viewed as a combination of the vertex and edge cases of the 2D case. If the viewing direction is parallel to the edge (such that it appears to the viewer as a point), it must always be connected to two perpendicular edges of the coarse grid. In the direction along the edge, one connects it to a point or a parallel edge. The combination results in the edge to be connected to either two perpendicular edges or two quadrilaterals of the coarse grid. This results in two tetrahedra, shown in Fig. 5(iv), or two deformed triangle prisms, shown in Fig. 5(v), as connecting cells. The vertex case is the combination of two 2D vertex cases. This results in each vertex being connected to three quadrilaterals of the coarse grid via pyramid cells, as shown in Figure 5(vi).

When the coarse grid is refined by more than one fine grid, one must check each coarse-grid point for refinement. Edges might be "upgraded" to the quadrilateral case

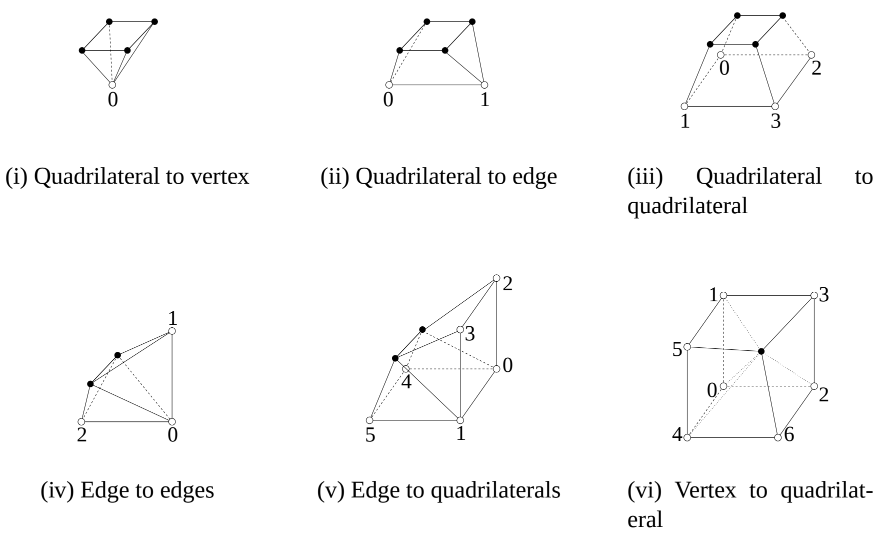

0
0
1
0
2
1
3
1
2
0
2
3
0
4
5
1
1
3
5
0
2
4
6
(i) Quadrilateral to vertex
(ii) Quadrilateral to edge
(iii) Quadrilateral to quadrilateral
(iv) Edge to edges
(v) Edge to quadrilaterals
(vi) Vertex to quadrilateral

Fig. 5. Possible connection types for quadrilateral, edge and vertex in 3D case

(for two adjacent edges). This occurs for the hexahedral cell (Fig. 5(iii)) when either grid points 2 and 3 or grid points 4 and 5 are refined. Vertices can be promoted to edges, or even quadrilaterals, when more than two grids meet at a given location. The fine vertex shown in Fig. 5(vi) can be promoted to an edge, if any of the coarse grid points 3, 5, or 6 is refined.

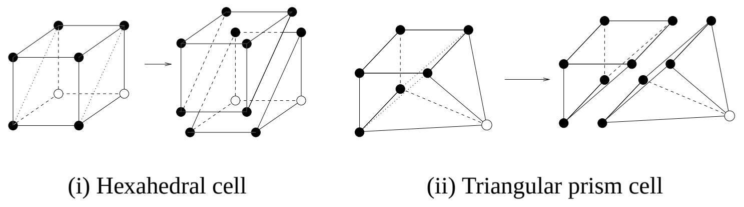


Fig. 6. Tessellations for 3D cells

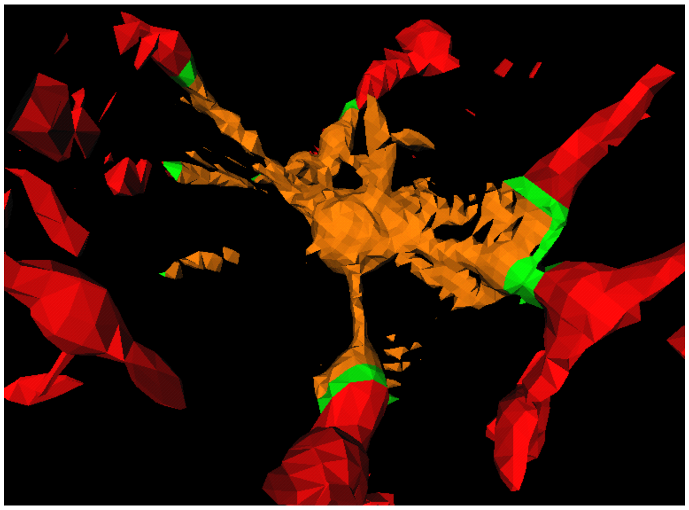
The possible refinement configurations result in a large number of cases to be considered. In situations where a fine quadrilateral, edge, or vertex is connected to coarse quadrilaterals, eight points are considered. These points form a deformed hexahedral cell. Each of the faces of the cell corresponds to a possible 2D refinement configuration shown in Fig. 4. It is important to note that the 2D refinement configurations that produce subdivided quadrilaterals are the same configurations that yield non-planar cell boundaries, i.e., boundaries that need to be subdivided. Fig. 6(i) illustrates that this subdivision information alone is sufficient to determine a tessellation. It is not necessary to consider the actual positions of the points. Each face of the cell in the figure is subdivided using the canonical tessellations depicted in Fig. 4, illustrated by the dotted lines. This subdivision of the faces implies tessellations of hexahedral cells into pyramids, triangular prisms, and tetrahedra. In the case of the pyramid (Fig. 5(i)), a refined coarse point is replaced by a fine quadrilateral; the result is a hexahedral cell. Refined coarse points in triangular prisms must be replaced by a fine edge. One of the the possible configurations is shown in Fig. 6(ii). If both coarse points of the triangular prism are refined, the resulting hexahedral cell does not have to be split further.

6 Isosurface Extraction

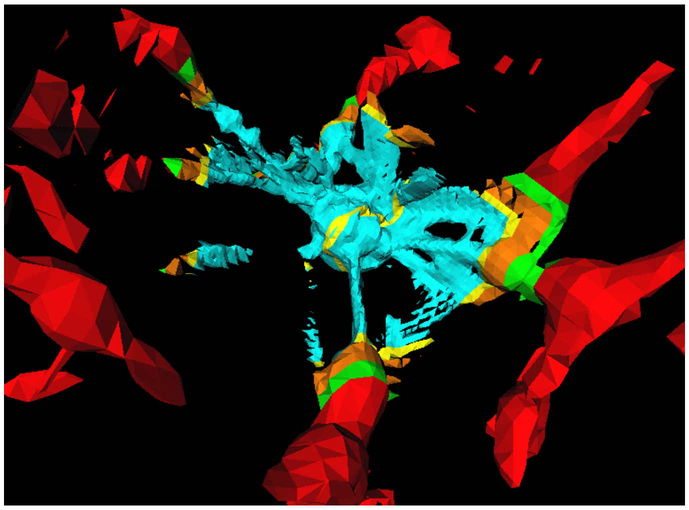
Within the individual grids, we apply a slightly modified MC approach. Instead of considering all cells of a grid for isosurface generation, we consider only those cells that are not refined by a finer grid. We do this by pre-computing a map with refinement information for each grid. For each grid cell, this map contains an index of a refining grid or an entry that the cell is unrefined. This enables us to quickly skip refined portions of the grid. For the generation of isosurface within the stitch cells, the MC method must be extended to handle the cell types generated during the stitching process. This is a straightforward extension achieved by generating case tables for each of the new cell types. These new case tables must be compatible with the one used in the MC approach, i.e., the ambiguous cases mentioned in Section 2 must be handled in exact the same way as for the hexahedral cells.

7 Results

Fig. 7 shows isosurfaces extracted from an AMR data set. The isosurface in Fig. 7(i) shows an isosurface extracted from two levels of the hierarchy, and Fig. 7(ii) one extracted from three levels. To highlight the transitions between levels, parts of the isosurface extracted from different levels of the hierarchy are colored differently. Isosurface parts extracted from the root, the first and the second level are colored red, orange and light blue, respectively. Portions extracted from the stitch meshes between the root and the first level are colored in green, and portions extracted from the stitch mesh between the first and second level are colored in yellow. The root level and the first level of the used AMR hierarchy each consist of one $32 \times 32 \times 32$ grid. The second level consists of 12 grids with dimensions $6 \times 12 \times 6$, 6×4, $8 \times 12 \times 10$, $6 \times 4 \times 4$, $14 \times 4 \times 10$, $6 \times 6 \times 12$, $12 \times 10 \times 12$, $10 \times 4 \times 8$, $6 \times 6 \times 2$, $16 \times 26 \times 52$, $14 \times 16 \times 12$, and $36 \times 52 \times 36$. All measurements were performed on an standard PC with a $700\mathrm{Mhz}$ Pentium III processor.



(i) Isosurface extracted using two out of seven levels of the AMR hierarchy. Generating the stitch cells required approximately 55ms, generating the isosurface approximately 250ms



(ii) Isosurface extracted using three out of seven levels of the AMR hierarchy.Generating the stitch cells required approximately 340ms, generating the isosurface approximately 600ms

Fig. 7. Isosurface extracted from AMR hierarchy (data set courtesy of Greg Bryan, Massachusetts Institute of Technology, Theoretical Cosmology Group, Cambridge, Massachusetts)

8 Future Work

One possible extension of our method is to use a generic triangulation scheme ensuring crack-free isosurface extraction. This would allow the use of our method for other, more general AMR data, where grids might not necessarily be axis-aligned, e.g., data sets produced by the AMR method of Berger and Oliger [3]. Furthermore, it is possible to use the computed tessellation for other purposes, not just for extraction of isosurfaces. One other possible application is to use the tessellation for direct volume rendering to obtain high-quality volume-rendered images, see Max [10].

9 Acknowledgments

This work was supported by the Directory, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098; the Lawrence Berkeley National Laboratory; the National Science Foundation under contracts ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; the Los Alamos National Laboratory; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628.

We also acknowledge the support of ALSTOM Schilling Robotics and SGI. We thank the members of the NERSC/LBNL Visualization Group; the LBNL Applied Numerical Algorithms Group; the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, and the AG Graphische Datenverarbeitung und Computergeometrie at the University of Kaiserslautern, Germany.

References

[1] AVS5. Product of Advanced Visual Systems, further details can be found at `http://www.avs.com/products/AVS5/avs5.htm`.

[2] Marsha Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, May 1989. Lawrence Livermore National Laboratory, Technical Report No. UCRL-97196.

[3] Marsha Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, March 1984.

[4] Greg L. Bryan. Fluids in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, 1(2):46–53, March/April 1999.

[5] Allen Van Gelder and Jane Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, October 1994.

[6] Markus H. Gross, Oliver G. Staadt, and Roger Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):130–143, June 1996.

[7] Interactive Data Language (IDL). Product of Research Systems, Inc., see `http://www.rsinc.com/idl/index.cfm` for details.

[8] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):163–169, July 1987.

[9] Kwan-Liu Ma. Parallel rendering of 3D AMR data on the SGI/Cray T3E. In: *Proceedings of Frontiers '99 the Seventh Symposium on the Frontiers of Massively Parallel Computation*, pages 138–145, IEEE Computer Society Press, Los Alamitos, California, February 1999.

[10] Nelson L. Max. Sorting for polyhedron compositing. In: Hans Hagen, Heinrich Müller, and Gregory M. Nielson, editors, *Focus on Scientific Visualization*, pages 259–268. Springer-Verlag, New York, New York, 1993.

[11] Gregory M. Nielson and Bernd Hamann. The asymptotic decider: Removing the ambiguity in marching cubes. In: Gregory M. Nielson and Larry J. Rosenblum, editors, *IEEE Visualization '91*, pages 83–91, IEEE Computer Society Press, Los Alamitos, California, 1991.

[12] Michael L. Norman, John M. Shalf, Stuart Levy, and Greg Daues. Diving deep: Data management and visualization strategies for adaptive mesh refinement simulations. *Computing in Science and Engineering*, 1(4):36–47, July/August 1999.

[13] William J. Schroeder, Kenneth M. Martin, and William E. Lorensen. *The Visualization Toolkit*, second edition, 1998. Prentice-Hall, Upper Saddle River, New Jersey.

[14] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surface. In: Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 335–342, 499, IEEE Computer Society Press, Los Alamitos, California, October 1998.

[15] Rüdiger Westermann, Leif Kobbelt, and Thomas Ertl. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer*, 15(2):100–111, 1999.