An Isosurface Continuity Algorithm for Super Adaptive Resolution Data [*]

Robert S Laramee [†]
VRVis, Center for Virtual Reality and Visualization
Donau-City-Strasse 1
A-1220 Vienna, Austria
`laramee@vrvis.at`

R. Daniel Bergeron [‡]
Department of Computer Science
Kingsbury Hall
University of New Hampshire
Durham, New Hampshire 03824, USA
`rdb@cs.unh.edu`

February 8, 2002

Abstract

We present the chain-gang algorithm for isosurface rendering of super adaptive resolution (SAR) volume data in order to minimize (1) the space needed for storage of both the data and the isosurface and (2) the time taken for computation. The chain-gang algorithm is able to resolve discontinuities in SAR data sets. Unnecessary computation is avoided by skipping over large sets of volume data deemed uninteresting. Memory space is saved by leaving the uninteresting voxels out of our octree data structure used to traverse the volume data. Our isosurface generation algorithm extends the Marching Cubes Algorithm in order to handle inconsistencies that can arise between abutting cells that are separated by both one and two levels of resolution.

Keywords: isosurface rendering, adaptive resolution visualization, marching cubes, uncertainty visualization, chain-gang

1 Introduction

Data analysis in the scientific community often deals with volume data. The size of today's volume data sets continues to grow much faster than our ability to render that data in an effective interactive environment. One approach to handling these very large

[*] The research presented here is supported in part by the National Science Foundation, grants IIS-9871859 and IIS-0082577 and the KPlus research program in Austria
[†] phone: +43 (316) 787-1740, fax: +43 (316) 787-777
[‡] phone: +1 (603) 862-3778, fax: +1 (603) 862-3493

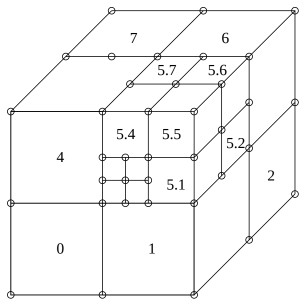

Figure 1: An example of a super adaptive resolution (SAR) data set with numbered octants. Cubes numbered 1 and 4 abut data that is two levels of resolution finer than themselves.

data sets is to represent them at multiple levels of resolution. A scientist can initially view and manipulate a relatively coarse version of the data in order to identify regions of interest which can then be viewed at a finer resolution. Alternatively, it is sometimes possible to create a single data set that retains fine resolution data in "interesting" regions (e.g. where the data is changing rapidly) while using coarse resolution data in other areas. This AR data representation can be more time and space efficient than multiresolution data, but it is more difficult to process. This paper describes the chain-gang algorithm: an extension of the Marching Cubes (MC) [9] isosurface volume rendering to support volume data defined with a super adaptive resolution representation. Unlike previous research, the chain-gang algorithm is able to resolve discontinuities in isosurfaces constructed from SAR volume data in which neighboring cubes can differ by two or more resolution levels.

2 Data Generation

In this section we describe our MR and SAR data sets which also include error data that can be included for uncertainty visualization.

2.1 Multiresolution Data

There are numerous techniques for generating a multiresolution representation (MR) of a volume data set. For the data sets shown in this paper we generate a hierarchy by subsampling attribute data. Subsampling chooses every other data point in each dimension to generate the next coarser resolution. This methodology for generating MR hierarchies has also been used in other applications [6]. We also tried another averaging technique, a wavelet approach [19], but we found subsampling to be more suited for this particular algorithm.

2.2 Super Adaptive Resolution Data

We use the term *super adaptive resolution* (SAR) data to describe a dataset that contains abutting neighbors that differ by *two* or more levels of resolution. An example of SAR data is shown in Figure 1. Cubes numbered 1 and 4 abut data that is two levels of resolution finer than themselves. Conceptually, this involves combining cubes whose vertex values are within a specified delta range. The SAR representations are decided by the change in values in a local area. We store a finer resolution representation in areas of rapidly changing sample data points, whereas areas with little change are stored with a coarser resolution representation. In other words, we can define a scalar value that represents an amount of change, δ, in the volume data set. When examining a volume cube, if the cube's vertices do not encompass a change greater than or equal to δ, the volume is represented by a larger cube (of coarser resolution). Each node of the octree includes a cube. If the difference between the maximum and minimum vertex scalar values is less than the threshold value, all of the octree node's children are pruned from the octree. The result is that the volume data is downsampled in areas of that are mostly flat and conversely maintains fine resolution in areas of high detail as in [13]. With this adaptive representation of the volume data, we do not save data which we may find to be uninteresting. In this fashion, we save both memory space resulting from a reduction in the amount of data stored and computation time during the rendering phase because many cubes have been eliminated from the data set.

SAR data gets its name from the amount of memory space that is saved relative to uniform resolution data. Our SAR data sets are only 1/5–1/10 the size of the full, uniform resolution data sets.

2.3 Error Data

In addition to the SAR data representation, we also generate a measure of the error introduced into the data by the SAR data generation. We assume that each point in the coarse data representation represents 8 points in the next finer resolution (the 8 child nodes in the octree). We compute the error of this point to be the average error representing the 8 original values. This error information can be included in the isosurface rendering [11]. One advantage to including the error data is that we can see which portions of the isosurface are generated from data with high error. These portions show up as yellow and red hue in the resulting images (see Results Section 5). One disadvantage to including the error data is that the rendering is no longer optimized with respect to time, since we are now processing multivariate data sets.

3 Related Work

Multiresolution data has been used to speed up isosurface rendering since early research showed that 30% to 70% of time spent rendering was spent processing empty cubes [17, 18]. Most of the approaches in this area rely on the creation of a single complete static MR representation of the data that is *adaptively traversed* to generate the isosurface. Some researchers have developed algorithms that generate isosurfaces from AR data including Engel et al. [3], Shekhar et al. [13], Shu et al. [14], and Westermann et al. [16].

3.1 Related Work in Isosurface Extraction

First we review some related work in the area of isosurface extraction. Livnat et al [8] presented an isosurface extraction algorithm for unstructured grids. The focus of their work was to minimize the search time for finding an isosurface over the value space and to extract the isosurface from an unstructured grid. A table of performance times is given only for the search phase of their algorithm. The focus of this paper is resolving the discontinuities that arise from SAR representations. In [8] the problem of discontinuities in an SAR representation is not addressed. Furthermore the preprocessing step of quick-sorting the data by both maximum and minimum coordinates is not a requirement of the chain-gang algorithm (as it is in [8]) since our representation is inherently structured. However, we can learn from their work that the search of the octree data structure has a maximum complexity of $O(k + k\log(n/k))$.

In the work done by Weber et al [15] a different problem is addressed. They are concerned with resolving the discontinuities between different levels in an Adaptive mesh refinement (AMR) hierarchy. The resolution takes place at the data set level, not the isosurface level. Sample AMR grids are stitched together with a refinement ratio of 2. Special stitch cells are generated using a case-table approach that bridges the gap between adjacent levels in the AMR hierarchy. The resulting computation is then used for isosurface extraction (for example). However, the results can also be used for other purposes such as direct volume rendering.

3.2 Static MR Representation

Cox and Ellsworth [2, 1] observe that the amount of data generated by a visualization algorithm is relatively small compared to the total amount of data. This implies that sparse traversal methods can be created that reduce the amount of data needed to be accessed [7]. In these approaches the entire octree is constructed and then traversed adaptively. In other words, the data is *not* adaptive but the traversal is. The isosurface value is examined inside blocks and the different resolutions are accessed adaptively. Isosurface traversal begins at the coarsest level of resolution. Whenever the isosurface value falls within the minimum-maximum boundary of a block of volume data, the next finer level of resolution in that block is traversed. If the isosurface value falls within the minimum-maximum boundary of one of those finer resolution blocks, again, the next finer level of resolution in the corresponding block is traversed. This procedure is applied recursively as long as the isosurface value is found within the bounds of the volume data or until the finest resolution block is reached. Given an isosurface value, if it's not within the minimum-maximum boundary of a block, then entire branches of the tree data structure are skipped.

One characteristic of this approach is that neighboring cubes used to generate the isosurface are always at the same resolution. When the isosurface passes from one cube to its neighbor, the neighbor is at the same level in the octree. This is a consequence of having a full MR representation to start with. However, the full MR representation also has disadvantages. In particular, the full MR representation takes up more storage space and requires more computation than AR or SAR representations. It is often the case that a full MR representation is not needed due to volume data redundancy and areas in the volume that are simply not the focus for a scientist. Furthermore, an isosurface *cannot* pass through a cube with 8 equal vertex values, therefore these cubes should be summarized with SAR representations.

The focus of the research presented by Shekhar et al [13] is to minimize the number of triangles needed for an isosurface. Discontinuities in the isosurface are resolved only as a result of their octree-based decimation of marching cubes surfaces, not of the SAR data representation itself. Again, the data is not adaptive but the traversal is. Access to the finest resolution data is always available. The approach is a bottom-up traversal of the octree that uses adaptive downsampling as a means to reduce the number of surface primitives. The volume data is downsampled in areas of the isosurface that are mostly flat and conversely maintains fine resolution in areas of high detail. There are some drawbacks to their approach:

1. The reported savings of up to 90% in the number of surface triangles is not much of an improvement over the 80-90% savings reported by Montani et al [10] in their discretized MC algorithm.

2. Several passes are required through their octree data structure in order to achieve a compact surface representation. In the chain-gang algorithm only one pass is made through the entire isosurface (phase I) and one pass is made through only the adaptive portions of the isosurface (phase II).

3. The user is required to choose a seed cell, that is, a cell having a piece of the desired isosurface. This can be a drawback in terms of the time required for user interaction. Also, the user may not know where an appropriate seed cell is.

4. Their approach generates a maximum of one isosurface even though there may be more than one surface with the same isovalue.

5. They use a crack-patching strategy to resolve discontinuities. Crack-patching does not take full advantage of the finer resolution data that is available at the location of the discontinuity. However, the chain-gang algorithm does take advantage of the finer resolution data.

6. Not only does their patching scheme not take full advantage of all the data that is available, but it also introduces unnecessary error. Patching may move triangle vertices out of the cell from which it was generated. Since the patching sometimes moves points from a fine resolution edge more error is introduced, and the introduced error may exceed the user-specified error.

7. Although the authors claim to handle the case where a $4 \times 4 \times 4$ cell meets a $1 \times 1 \times 1$ cell, they do not present a clear explanation as to how this case is handled.

3.3 Adaptive Marching Cubes

Shu, Zhou, and Kankanhalli [14] were successful in speeding up the marching cubes algorithm with their version of an adaptive marching cubes (AMC) algorithm. Their goal was to bring the MC algorithm to interactive time. They reduced the number of triangles by up to 55% by adapting the size of triangles to fit the shape of the isosurface. Their research differs from ours in the following ways:

1. Their algorithm does not handle resolution transitions that are two or more levels apart as in Figure 2 (right). Our algorithm is able to resolve this type of discontinuity. Their algorithm only resolves discontinuities such as those found in Figure 2 (left).

2. They resolve discontinuities in the isosurface differently then we do. Cracks may appear between two different neighboring cubes at different levels of resolution. In what they called the “crack problem” [14], discontinuities in the surface are patched with polygons the same shape as those of the cracks. While crack patching is an efficient solution it does not take full advantage of the finer resolution data. In other words, the coarser resolution is left as is, with less accuracy. Our

algorithm updates the coarser resolution data with the finest resolution data that is available from the neighbor, thus improving the accuracy of the isosurface.

3. Their cracks are abstracted into 22 basic configurations of different sizes, a solution that requires $O(n^2)$ of working memory space for an $n \times n \times n$ volume data set. Our algorithm only requires the working memory space necessary to hold the SAR data set. The amount of working memory space taken up by the SAR data set is flexible and is often 1/5–1/10 the size of the equivalent MR data set (as in Results Section 5).

4. Their algorithm uses a static uniform resolution representation with a dynamic MR traversal. Our algorithm uses an SAR representation of the data and does not require access to the original MR representation.

5. Their volume data is stored using a single level resolution representation with an MR traversal. Our data is stored using an SAR representation with an AR traversal.

Westermann, Kobbelt, and Ertl [16] achieved real-time exploration of regular volume data using adaptive reconstruction of isosurfaces. Their work differs from ours in the following manner(s):

1. Their algorithm does not handle resolution transitions that are two levels apart (as in Figure 2, right). Our algorithm is able to resolve this type of discontinuity. Their algorithm only resolves discontinuities such as those found in Figure 2 (left).

2. Their algorithm requires knowledge about the direction in which the resolution transition occurs. The direction in which the resolution transition occurs is defined as radially outward from a *focus point oracle* (at the center of interest). In other words, they can only resolve discontinuities when they know the transition direction is from the focus point oracle, where the fine resolution data resides, to the coarse resolution data (outside the *radius of interest*). We can handle discontinuities occurring from any direction and in *multiple* directions.

3. They use the *focus point oracle* to decide which portions of the data are rendered at the finest resolution [16]. In our case, the subset of data at the finest resolution is chosen automatically, prior to run time.

4. As a consequence of the focus point oracle, their algorithm also relies on having access to the entire, uniform resolution data set whereas we start with a static SAR data set. that is often 1/5–1/10 the size of the equivalent MR data set (as in Results Section 5).

It is important to note that the strategies used by Shu et al [14] and Westermann et al [16] to resolve discontinuities in AR data are based on case tables not unlike those used be the original MC algorithm [9]. A simple case table approach is not feasible nor extensible for resolving discontinuities that are separated by two levels of resolution. Also worthy of note is that we are implementing our algorithm in Java (not C++) taking advantage of it's platform independence.

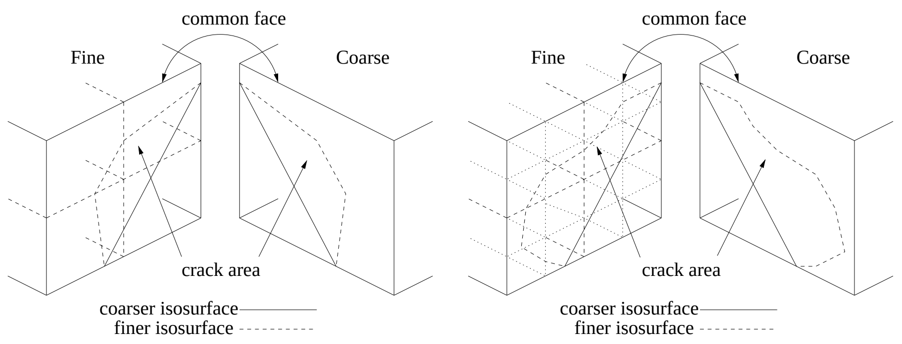

Figure 2: (left) An isosurface discontinuity that is resolved by Livnat [8], Shekhar [13], Shu [14], Westermann [16], and the chain-gang algorithm. (right) Another isosurface discontinuity that the chain-gang algorithm can resolve. The resolution of the neighboring data is separated by two levels.

3.4 Isosurface Discontinuities

Because we are dealing with adaptive resolution data, we may have a larger, unsubdivided volume with a triangle next to a smaller, neighboring cube with another triangle sharing a vertex with the larger triangle. This can introduce discontinuities in the isosurface. Looking at Figure 3a, we may generate a lower resolution triangle entirely in the larger voxel. Then we generate the triangles in the neighboring finer resolution voxel (entirely in the smaller cube). Another possible configuration is shown in Figure 3b. In both cases the two triangles introduce a surface discontinuity. The key to identifying this problem is recognizing that one or more triangle vertices are on the face of the larger voxel. The difference between the two cases is that the first one has 2 cube vertices above the isovalue on the *edge* of its coarser neighbor, while the second case has an additional cube vertex (above the isovalue) on the *face* of its coarser neighbor.

The algorithms presented by Livnat [8], Shekhar [13], Shu [14], Westermann [16], and the chain-gang algorithm all resolve those discontinuities in which the abutting cubes are one level apart in resolution, as in Figure 2 (left). However, only the chain-gang algorithm provides a clear means by which to resolve discontinuities two levels apart in resolution as in Figure 2 (right).

4 The Chain-Gang Algorithm

The chain-gang algorithm consists of two phases. The first phase consists of identifying the AR and SAR portions of the isosurface. The second phase then resolves the discontinuities between resolution transitions. Our approach uses a static SAR representation and an AR traversal algorithm. Any value from the data set may be chosen to generate an isosurface. The basic goal of our algorithm is to identify isosurface intersections that can be chained together. Groups of chains form a gang. As a result we've named our algorithm the chain-gang algorithm.

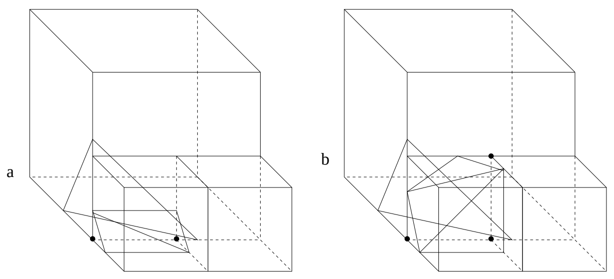

Figure 3: Two triangles generated at different resolutions that introduce isosurface discontinuity (a) the finer resolution cube has 2 cube vertices above the isovalue on the edge of its coarser neighbor (b) the finer resolution cube has 3 cube vertices above the isovalue on the edge and face of its coarser neighbor

4.1 Phase I, Identifying the Gang

Since we are rendering an adaptive representation of the volume data, complications arise from the adjacency of blocks at different levels of resolution. The elegance of the conventional MC algorithm is that each cube is treated independently. This elegance is lost when we render adjacent blocks at different levels of resolution. We must introduce additional methodology into the MC algorithm to overcome these complications.

Just as the MC algorithm acts conceptually as a filter applied to an input 3D data set and outputting an isosurface, the chain-gang algorithm acts as another filter applied on top of the MC algorithm. The input to the chain-gang filter is the isosurface output by the MC algorithm, with discontinuities, and the output is the same isosurface with the discontinuities resolved. Figure 2 shows two examples of these discontinuities.

In phase I of the chain-gang algorithm, the isosurface is separated into two portions: (1) the uniform resolution portion(s) of the isosurface and (2) the SAR portion(s) of the isosurface. The AR and SAR portions of the isosurface are defined as those where a resolution transition occurs from one cube to the next i.e. where a resolution transition occurs between two or more abutting cubes.

In phase I each cube generates triangles using the standard MC algorithm. In order to identify the SAR portions of the isosurface, each cube examines the resolution of its eight abutting face neighbors using an AR octree data structure. If a *coarser* neighbor is found then the *finer* resolution cube performs the following computation:

1. Each triangle computed by the MC algorithm is examined to see if one of its edges abuts the coarser resolution neighbor.

2. If a triangle edge is found that abuts the coarser resolution neighbor, that triangle segment is attached to the face of the coarser resolution face of the neighboring cube.

3. The coarser resolution neighbor is identified being part of the SAR portion of the isosurface and the triangle edge is identified as being a chain-link.

The computation of the chain-links is shown in Figure 4. We can see from the figure that whenever a finer resolution cube identifies one of its own triangle edges on the face

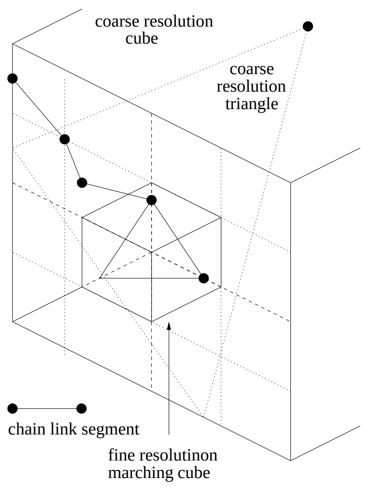

Figure 4: The first phase of the chain-gang algorithm: computing the chain links.

of its coarser neighbor, the triangle edge forms a chain-link on the coarser resolution face. We can also note that the order in which the cubes are traversed is independent from the chain-links themselves. The same chains will be formed independent of the cube traversal order.

At the end of phase I, every cube has been identified as belonging to the uniform resolution portion of the isosurface or the SAR resolution portion. All of those cubes belonging to the adaptive portions will have a set of chain links on one or more of their faces.

4.2 Phase II: Assembling the Gang

Phase II of the chain-gang algorithm consists of assembling the chain-links into continuous chains and then using the chains as a basis for the triangle fans output by the algorithm.

At the beginning of phase II, each SAR cube has a set of chain links on one or more of its faces. For each face of the cube these chain-links are assembled one-by-one into coherent chains by joining the chain-links at their corresponding endpoints. At the end of this process, the SAR cube has a group of chains i.e. a chain-gang. When the chain-gang has been assembled, the cubes own triangles (computed from the standard MC algorithm) are then searched. The following computation is performed:

1. A coarse triangle (i.e., a triangle inside a coarse resolution cube) edge is matched with a chain. In order for there to be a match, a coarser triangle edge must be on the same cube face as the chain, as in Figure 4.

2. If a matching coarse triangle edge has been paired with a chain, the coarse triangle edge is removed.

3. The coarse triangle edge is then replaced with the chain.

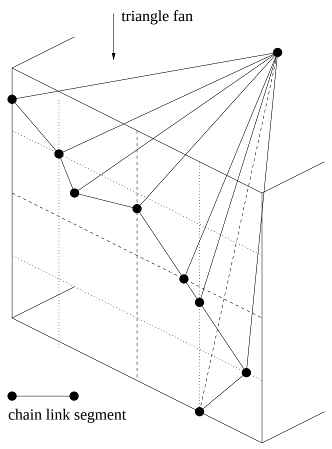

Figure 5: The second phase of the chain-gang algorithm: assembling the chain segments into chains and outputting the triangle fan(s).

4. A triangle fan is formed by using each chain-link as the new base of coarser resolution triangle.

Figure 5 shows the chain-gang assembly. A coarser resolution triangle inside a coarser resolution cube is replaced by a triangle fan formed by the chain and one vertex from the original coarser resolution triangle. The coarser resolution triangle vertex remains if it is not on the same face as the finer-resolution chain.

There are cases when more than one chain is associated with only one coarse resolution triangle. Figure 6 shows how the chain-gang algorithm identifies chains on multiple faces of a coarse resolution cube. In Figure 6, there are two chains on different sides of a coarser resolution cube that create a discontinuity that must be resolved with only one coarser resolution triangle. These cases are handled by applying the chain-gang algorithm in an iterative fashion. In short, after phase I has computed all of the chain segments, we apply phase II, repeatedly, until the discontinuity is resolved. In this way, we show that this is not a special case.

First we apply phase II of assembling the chain-gang and outputting a triangle fan, ignoring any other possible chains. The result of this step is shown in Figure 7. Here we see one triangle fan generated thus resolving the discontinuity between one chain and one coarse resolution triangle (on the left hand side). However, this is only an intermediate step. There is still a discontinuity between the second chain and the resulting isosurface on the inside of the coarser resolution cube (on the right). The chain-gang algorithm then uses the adjacent triangle in the triangle fan to resolve the discontinuity. This is the shaded triangle in Figure 7.

Figure 8 shows the result of the second application of phase II. A second triangle fan is generated using the vertex labeled *B* (as shown). If a coarse resolution triangle edge abuts the side of a cube before phase II is applied, a triangle edge is also guaranteed to abut the same side after phase II is applied. And this triangle edge can be used in another discontinuity resolution.

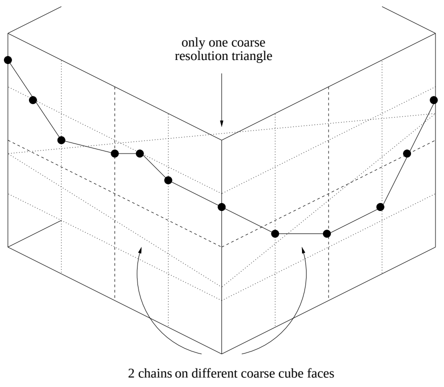

Figure 6: This figure shows the result of the chain-gang algorithm attach chain segments onto multiple sides of a coarse resolution cube.

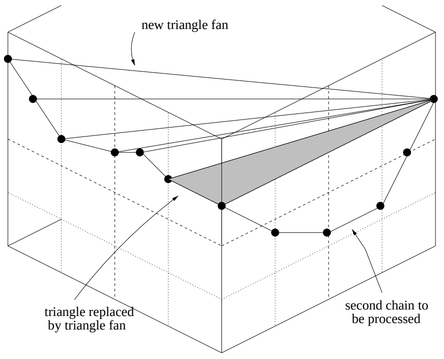

Figure 7: This is the intermediate result of resolving the discontinuity between one chain (on the left) and one side of a coarse resolution triangle. The shaded triangle shows which triangle is used to resolve the remaining discontinuity between the other chain (on the right) and the intermediate surface in the coarser cube.

4.3 Special Isosurface Discontinuity Cases

There are special discontinuity cases that arise when three different resolution levels are allowed to meet in the same data set. One such case is shown in Figure 9. Figure 9 shows cubes from three different resolutions meeting along a common edge. The key to resolving this discontinuity is for the finest resolution cube to treat all of its coarser resolution neighbors the same. In other words, no distinction is made between a coarser neighbor that is one resolution level coarser or two resolution levels coarser. In both cases, the finer resolution cube attaches its own triangle edge to the face of its coarser neighbor.

Figure 10 shows how this type of discontinuity is resolved. Two chains and two triangle fans (at different resolutions) are computed. Note that there is one triangle vertex that is shared amongst triangles at 3 different resolutions. Normally this would result in a case of inconsistent interpolation between cube vertices. However, the chain gang algorithm

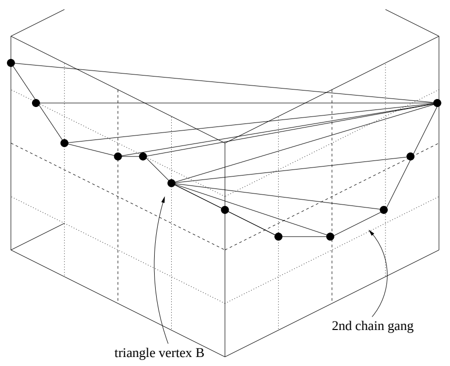

Figure 8: This is the result of phase II of the chain-gang algorithm being applied in an iterative fashion. A second triangle fan is generated using the shaded triangle shown in Figure 7. The vertex labeled *B* is used to form a triangle fan with the second chain (on the right-hand-side).

can overcome this by assembling the chains in finest-to-coarsest order. Figure 10 shows the finer chain that is assembled first (the base of the first triangle fan). Then the chain gang algorithm assembles the chains on the coarsest cubes. Since all of the triangles point to a common shared triangle vertex list and each triangle vertex stores its resolution, the finer resolution triangle vertex position is used automatically. It is through this triangle vertex list that inconsistent interpolation is corrected.

4.4 Inconsistent Interpolation

The standard algorithm for computing the intersection of a surface with a cube edge is to compute the linear interpolation between the end points of the edge. In a conventional volume data set, all cells are the same resolution, hence all shared edges have the same vertex values and thus neighboring cells interpolate to the same position. However, in our data set, two neighboring voxels at different resolutions may compute different linear interpolations. This ambiguity is easy to resolve: we always use the interpolated point of the finer resolution voxel. In order to identify an instance of inconsistent interpolation, each octree node inspects the neighboring nodes that share the edge on which an edge intersection occurs.

5 Results

Our evaluation experiments were run on an HP PC with a 1 GHz Intel Pentium III Processor and 500 Mbytes of RAM running Red Hat Linux 6.2. We use Sun Microsystem's Java version 1.2 (ported to Linux by Blackdown.org) and utilizes Java 3D. The isosurface generation algorithm is an Adaptive Marching Cubes derived from *The Visualization Toolkit* by Schroeder, Martin, and Lorensen [12], The display algorithm uses the VisAD [5, 4]. graphics library.

One of our test data sets is a $113 \times 256 \times 256$ slice CAT scan of a cadaver head taken on a General Electric CAT Scanner and provided courtesy of North Carolina Memorial Hospital and Siemens Medical Systems, Inc., Iselin, NJ. Figure 11 shows a 128^3 uniform resolution rendering of the medical image data with an isovalue of 0.185. The

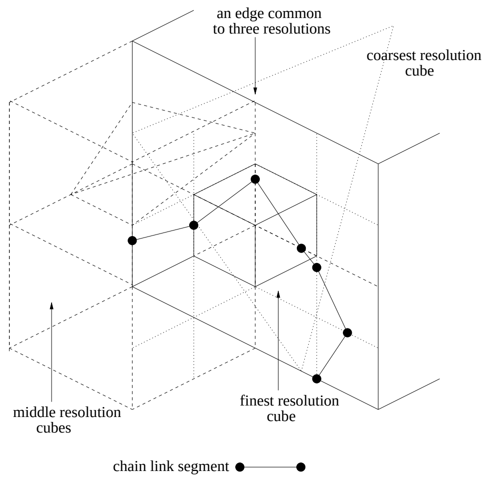

Figure 9: The chain gang algorithm also resolves discontinuities in which three different resolutions meet along the same edge. Phase I does not discriminate between neighbors that are one or two levels coarser in resolution.

other data set is a $128 \times 128 \times 64$ slice CAT scan of a lobster.

The AR threshold, δ, is the difference between the minimum and maximum scalar vertex values of a node's cube and all of its children. For example, a δ value of 10% results in all octree nodes whose scalar vertex values vary by less than 10% being trimmed from the original, uniform MR representation. Occurrences of discontinuities are clearly a function of both δ and the isosurface value chosen. With a higher δ we are more likely to encounter discontinuities for a given resolution. Also, for a constant δ the number of discontinuities varies with isovalue. With the proper AR threshold, we can compare the chain-gang algorithm with other AR algorithms. With our data sets we found the values of δ that create both AR data sets and SAR data sets. This is the fundamental difference between the previous work and the work presented here. The higher δ is, the more often neighboring data will differ by two levels of resolution. The chain-gang algorithm amounts to the same as previous AR algorithms when run on AR data while the added functionality is seen only with SAR data sets.

We gain considerable savings in space from an SAR data representation. Table 1 compares the storage and processing requirements of an AR data set with an SAR data set. The adaptive measure column identifies AR data sets and SAR data sets. In this case the AR data sets for the cadaver were generated using a δ value of 5% while the SAR data sets were generated using δ values of both 10% and 15% (labeled SAR and SAR+ respectively). The AR data sets for the lobster were generated using a δ value of 15% while the SAR data sets were generated using δ values of both 20% and 25% (labeled SAR and SAR+ respectively). The fourth column compares the storage requirements for each data set as a percentage of the original uniform resolution data set.

The AR cadaver head data set uses approximately 50% more storage space than the SAR data set and approximately 85% more storage space than the SAR+ data set. For the lobster data set the AR representation uses approximately 4% more storage space

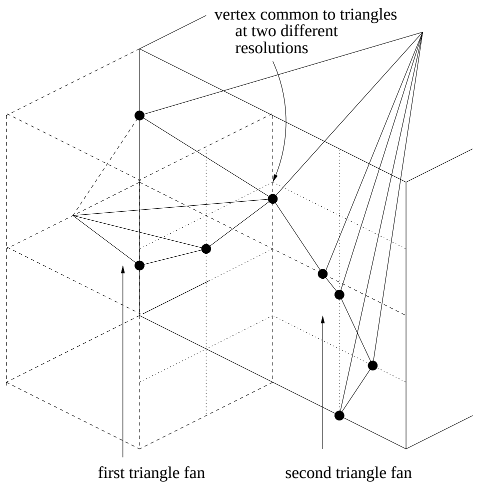

Figure 10: Here we illustrate how the chain gang algorithm resolves discontinuities in which three different resolutions meet along the same edge. Two triangle fans are created at two different resolutions. The inconsistent interpolation is also resolved.

data set	adaptive measure	isovalue	storage space (%)	no. of cubes in isosurface
head res 128^3	AR	0.185	28.3	338, 202
		0.408		158, 840
	SAR	0.185	18.8	316, 417
		0.408		156, 649
	SAR+	0.185	15.3	289, 751
		0.408		149, 068
lobster res 128^3	AR	0.185	11.5	77,918
	SAR		11.1	77,825
	SAR+		10.8	77,441

Table 1: This table compares the storage requirements of an SAR data set with an AR data sets.

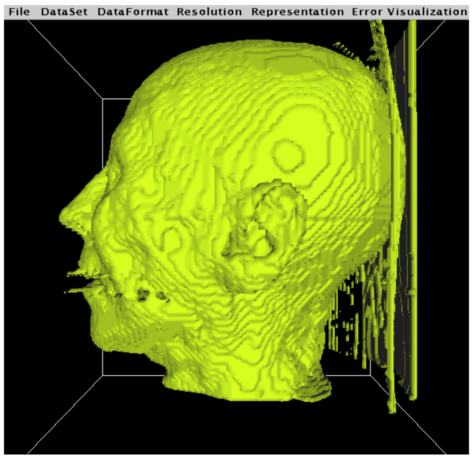
File DataSet DataFormat Resolution Representation Error Visualization

Figure 11: A 128^3 resolution rendering of the cadaver head data set with an isovalue of 0.185

data set	adaptive measure	isovalue	no. of triangles in isosurface	no. of 2RTs
head res 128^3	AR	0.185	215, 713	0
		0.408	208, 710	0
	SAR	0.185	214, 375	106
		0.408	207, 912	260
	SAR+	0.185	201, 880	1, 163
		0.408	203, 874	977
lobster res 128^3	AR	0.185	96, 112	0
	SAR		96, 111	47
	SAR+		95, 644	222

Table 2: This table compares sample AR isosurfaces with sample SAR isosurfaces.

than the SAR data set and approximately 6% more storage space than the SAR+ data set.

The last column shows the number of cubes that were processed when generating the isosurface. With the AR cadaver head data, about 7% more cubes are processed in generating the isosurface than the SAR isosurface and about 17% more cubes that the SAR+ isosurface. For the lobster data, only about 1% more cubes are processed in generating the AR isosurface than the SAR or SAR+ isosurfaces (both of isovalue 0.185).

Table 2 compares some sample AR isosurfaces with sample SAR isosurfaces. The fourth column compares the number of triangles in each isosurface. The AR cadaver head isosurface generates approximately 1,330 more triangles than the SAR isosurface and approximately 14,000 more triangles than the SAR+ isosurface (for the isovalue 0.185). We note that for the lobster, the SAR lobster and the chain-gang algorithm retain approximately the same number of triangles as the AR lobster. We attribute this the algorithm's ability to maintain high levels of detail in isosurfaces with high curvature.

The last column shows the number of abutting cubes that differ by two levels of resolution in the given isosurface (2RT -two resolution transitions). A requirement of the AR data sets is that there are none of these transitions. Whereas the sample SAR+ isosurface contains many of these transitions. However, from looking at Figures 12 through 14 we can see that the image quality remains intact throughout. The uncertainty visualization shows exactly where some of the 2RTs occur. We assign no error (mapped to green) to the original 128^3 resolution portions of the isosurface while error is accumulated for coarser resolutions (mapped to non-green hue).

6 Conclusions and Future Work

Our research has shown that isosurface rendering of an SAR volume data set using the chain-gang algorithm can lead to more efficient rendering with minimal loss to image quality. We can extend this research in several directions including: (1) improvements to the rendering algorithm, (2) handling a wider range of AR data representations, (3) rendering larger data sets, and (4) faster rendering time. Improvements to the rendering time can be made using a Just-In-Time compiler (JIT) or a Java compiler.

We would also like to explore SAR isosurface rendering cases where a finer resolution isosurface abuts two or more sides of coarse resolution cube, but no isosurface is computed inside the coarse resolution cube. From our initial experiments using the data sets in this paper, this appears to be a rare occurrence, however, we would like to identify more precisely how often it occurs. Ideally we would handle these cases by connecting the the isosurface across the coarse resolution cube. However, whenever we encounter an unexpected case using the chain-gang algorithm, we simply subdivide the cube.

7 Acknowledgements

We would like to thank all those who have contributed to the finance of this research including: The NSF -National Science Foundation (grants IIS-9871859 and IIS-0082577), the VRVis Center for Virtual Reality and Visualization (`www.vrvis.at`) which is funded by both AVL (`www.avl.com`) and an Austrian Governmental research program called KPlus (`www.kplus.at`). We would also like to thank (1) Phil Rhodes and Ted Sparr of the University of New Hampshire for their valuable contributions, (2) Zoltan Konyha of VRVis for his valuable contributions and feedback, and (3) Bill Hibbard and the VisAD group for their technical support. And one final note: all of the programming code for this project is 100% open source. Anyone interested in obtaining the open source code can contact `laramee@vrvis.at`.

References

[1] Michael B. Cox and David Ellsworth. Application-controlled demand paging for out-of-core visualization. Proc. Of Visualization '97. IEEE Computer Society Press, October 1997.

[2] Michael B. Cox and David Ellsworth. Managing big data for scientific visualization. *ACM Siggraph '97*, 21, August 1997. Course #4 Exploring Gigabyte Datasets in Real-Time: Algorithms, Data Management, and Time-Critical Design.

[3] Klaus Engel, Rudiger Westermann, and Thomas Ertl. Isosurface extraction techniques for web-based volume visualization. In *Volume Visualization*, Visualization 99, California, October 1999.

[4] William Hibbard. Connecting people to computations and people to people. *Computer Graphics*, 32(3):10–12, 1998.

[5] William Hibbard. The visad java class library developers guide. The World Wide Web, November 1999. http://www.ssec.wisc.edu/˜billh/visad.html.

[6] Eric C. LaMar, Bernd Hamann, and Kenneth I. Joy. Multiresolution techniques for interactive texture-based volume visualization. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 355–362, San Francisco, 1999. IEEE.

[7] C. Charles Law, Kenneth M. Martin, William J. Schroeder, and Joshua Temkin. A multi-threaded streaming pipeline architecture for large structured data sets. In *Volume Visualization*, Volume Visualization 99, California, October 1999.

[8] Yarden Livnat, Han-Wei Shen, and Christopher R. Johnson. A near optimal isosurface extraction algorithm for structured and unstructured grids. *IEEE Transactions on Visual Computer Graphics*, 2(1):73–84, 1996.

[9] William E. Lorensen and Harvey E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Comput Graph*, 21:163–169, 1987.

[10] Claudio Montani, Riccardo Scateni, and Roberto Scopigno. Discretized marching cubes. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of the Conference on Visualization*, pages 281–287, Los Alamitos, CA, USA, October 1994. IEEE Computer Society Press.

[11] Alex T. Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997. ISSN 0178-2789.

[12] William J. Schroeder, Kenneth M. Martin, and William E. Lorensen. *The Visualization Toolkit*. Prentice-Hall, Inc, Upper Saddle River, New Jersey 07458, 1996.

[13] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings of the Conference on Visualization*, pages 335–344, Los Alamitos, October 27–November 1 1996. IEEE.

[14] Renben Shu, Chen Zhou, and Mohan S Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11:202–217, 1995.

[15] Gunther H. Weber, Oliver Kreylos, Terry J Ligocki, John M. Shalf, Hans Hagen, Bernd Hamann, and Kenneth I Joy. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In D.S. Ebert, J.M. Favre, and R. Peikert, editors, *Data Visualization 2001 (Proceedings of VisSym 2001)*, pages 25–34, Vienna, Austria, 2001. Springer-Verlag.

[16] Ruediger Westermann, Leif P. Kobbelt, and Thomas Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15:100–111, 1999.

[17] Jane Wilhelms and Allen Van Gelder. Topological ambiguities in isosurface generation. Technical report, University of California, Santa Cruz, California, December 1990. Extended abstract in ACM Computer Graphics. 2, 5 79–86.

[18] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[19] Pak Chung Wong and R. Daniel Bergeron. Multiresolution multidimensional wavelet brushing. In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization ’96*, pages 141–148. IEEE, 1996.

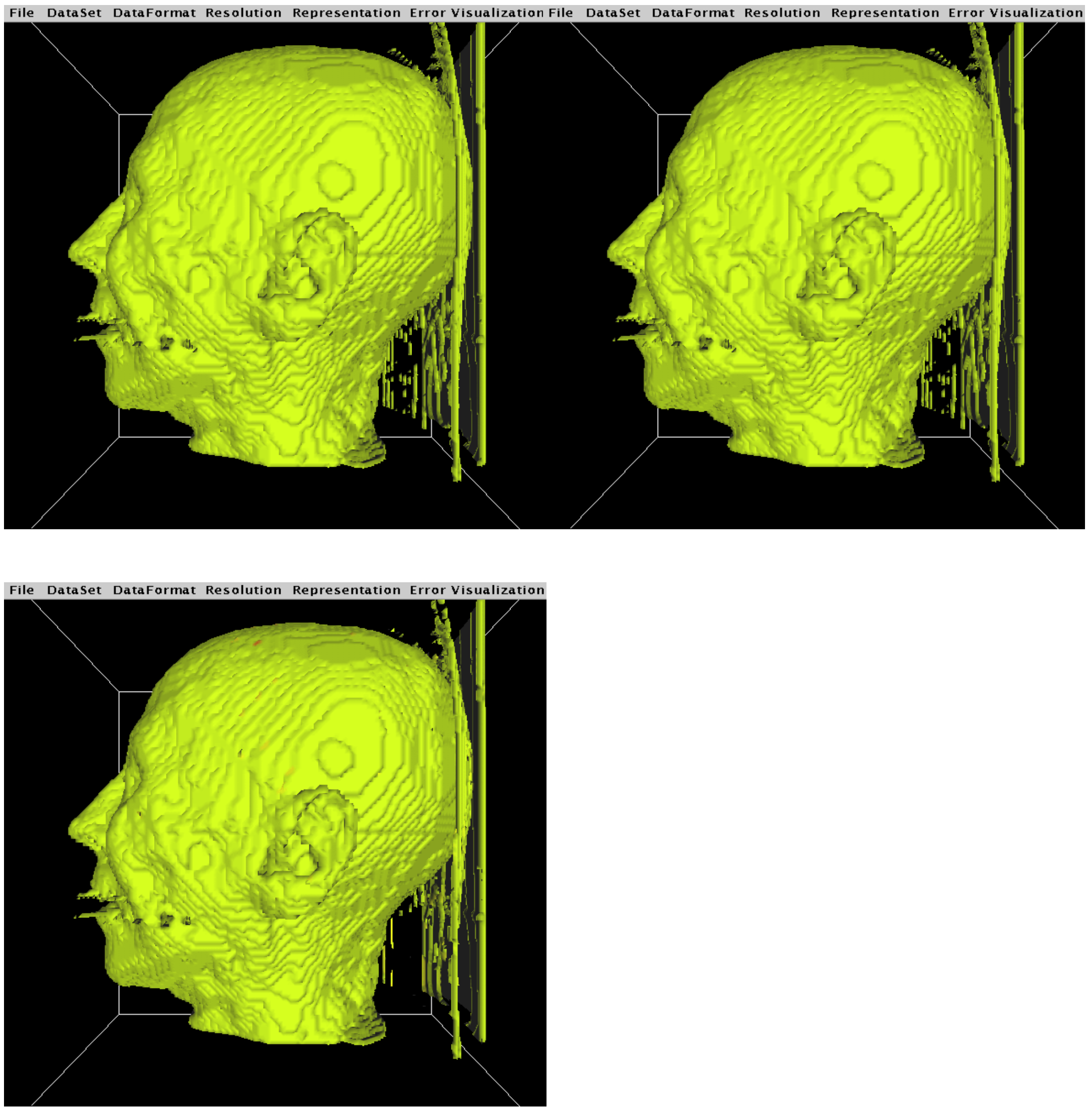
File DataSet DataFormat Resolution Representation Error Visualization
File DataSet DataFormat Resolution Representation Error Visualization
File DataSet DataFormat Resolution Representation Error Visualization

Figure 12: Three isosurfaces from the cadaver head data set, each with resolution 128^3 and an isovalue if 0.185: (top, left) a sample AR isosurface, (top, right) a sample SAR isosurface containing 106 2RTs, and (bottom, left) a sample SAR+ isosurface containing 1,163 2RTs. Yellow and red hue indicates SAR portions of the isosurface.

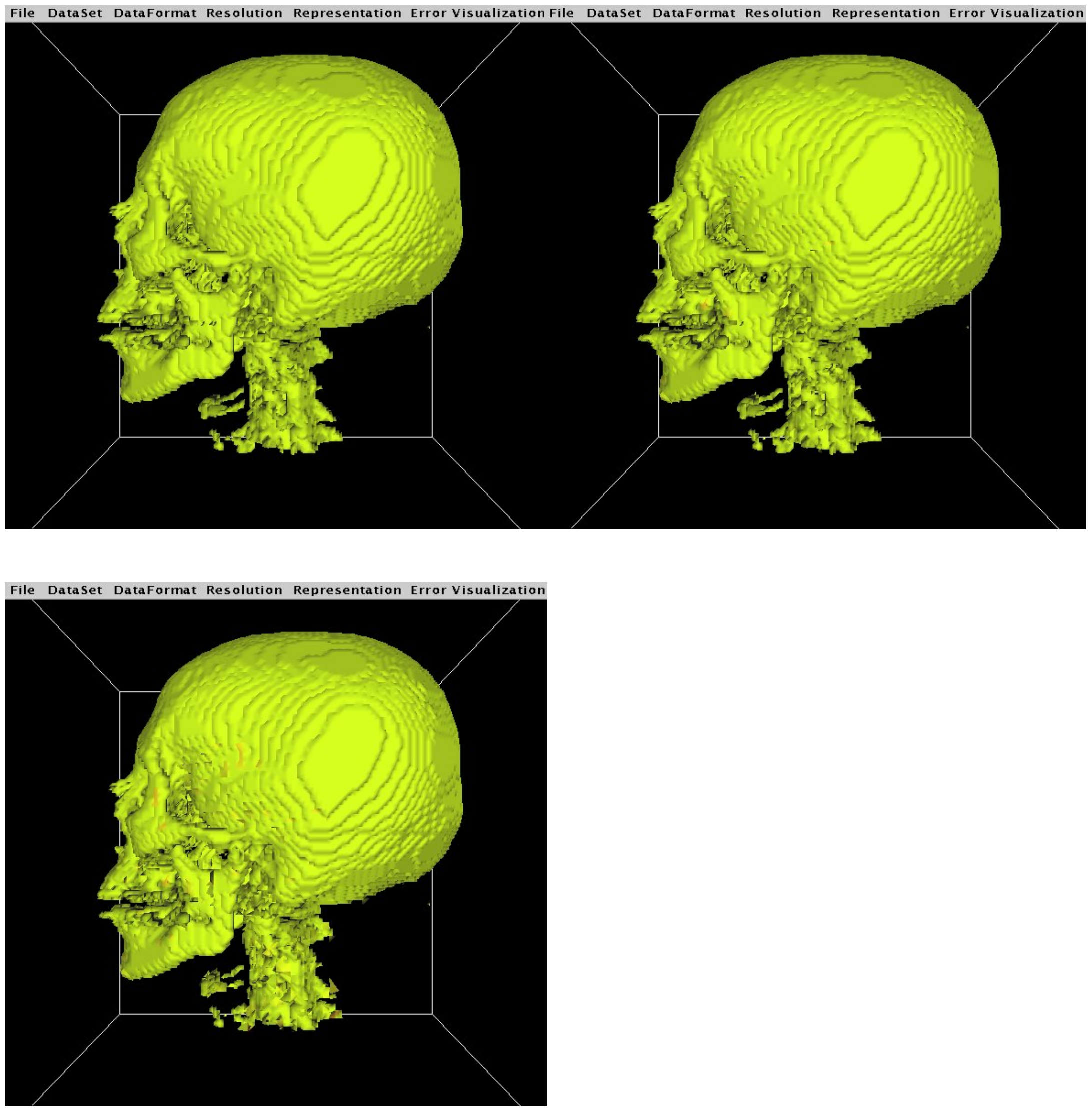
File DataSet DataFormat Resolution Representation Error Visualization
File DataSet DataFormat Resolution Representation Error Visualization
File DataSet DataFormat Resolution Representation Error Visualization

Figure 13: Three isosurfaces from the cadaver head data set, each with resolution 128^3 and an isovalue if 0.408: (top, left) a sample AR isosurface, (top, right) a sample SAR isosurface containing 260 2RTs, and (bottom, left) a sample SAR+ isosurface containing 977 2RTs. Non-green hue indicates occurrences of 2RTs.

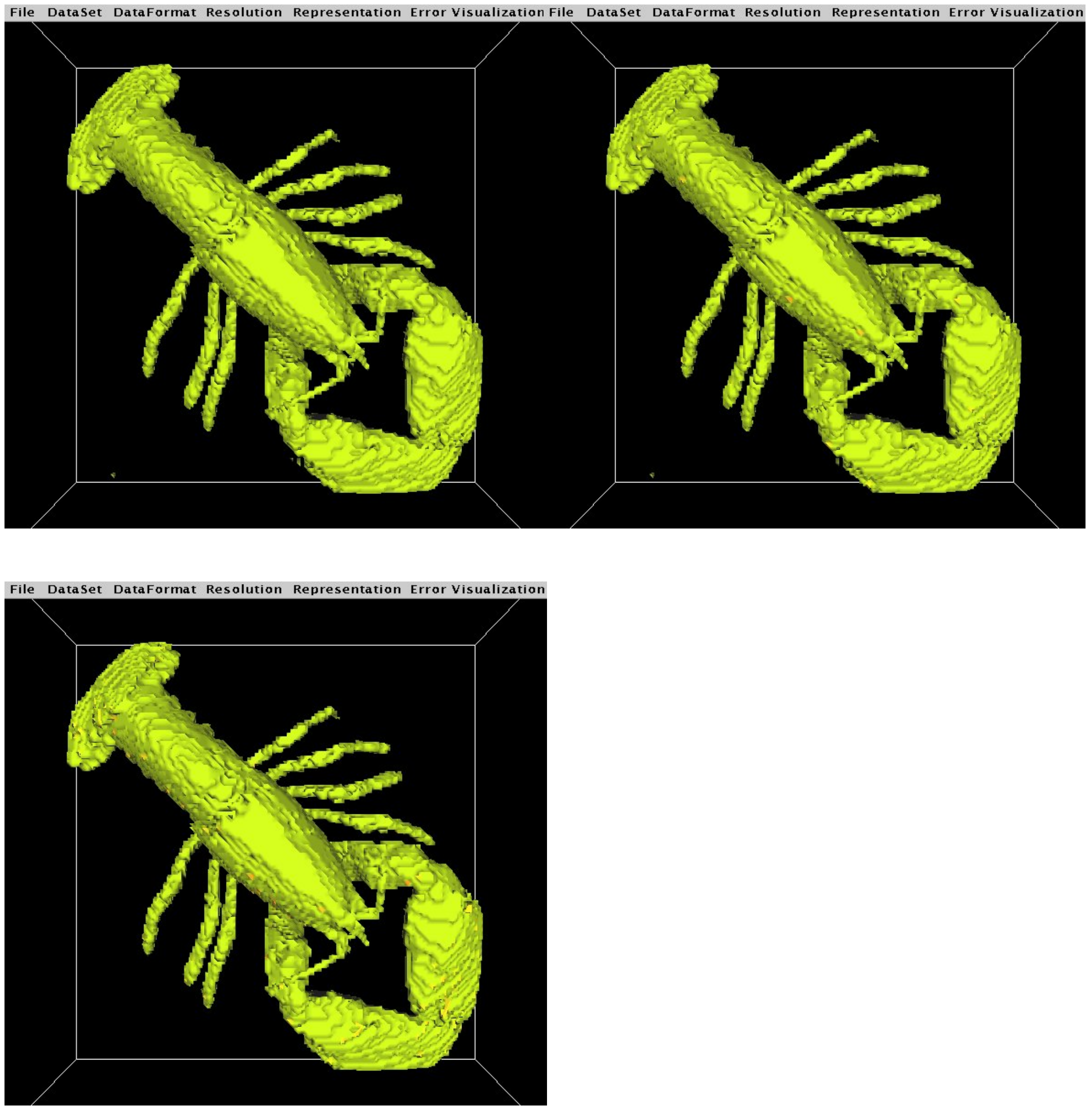
File DataSet DataFormat Resolution Representation Error Visualization
File DataSet DataFormat Resolution Representation Error Visualization
File DataSet DataFormat Resolution Representation Error Visualization

Figure 14: Three isosurfaces from a lobster data set, each with resolution 128^3 and an isovalue if 0.185: (top, left) a sample AR isosurface, (top, right) a sample SAR isosurface containing 47 2RTs, and (bottom, left) a sample SAR+ isosurface containing 22 2RTs. Yellow and red hue indicates occurrences of 2RTs.