Volume Interval Segmentation and Rendering

Praveen Bhaniramka[†], Caixia Zhang[*], Daqing Xue[*], Roger Crawfis[*], Rephael Wenger[*]
[†]Silicon Graphics, Inc. [*]The Ohio State University

Abstract

In this paper, we segment the volume into geometrically disjoint regions that can be rendered to provide a more effective and interactive volume rendering of structured and unstructured grids. Our segmentation is based upon intervals within the scalar field, producing a set of geometrically defined interval volumes. We present many advantageous properties in using interval volumes, and provide several new rendering operations or shaders to provide effective visualizations of the 3D scalar field. In particular, we demonstrate new technologies that allow interval volumes to be rendered interactively and/or used to reduce the amount of rasterization or rendering primitives in a volume renderer. We illustrate the use of interval volumes to highlight contour boundaries or material interfaces. Several surface shaders that can easily be integrated in the volume renderer are presented. To construct the interval volumes, we cast the problem one dimension higher, using a higher-dimensional isosurface construction for interactive computation or segmentation. The algorithm is independent of the dimension and topology of the polyhedral cells comprising the grid, and thus offers an excellent enhancement to the volume rendering of unstructured grids. We present examples using hexahedral and tetrahedral cells from time-varying and multi-attribute datasets.

CR Categories: I.3.3 [Picture/Image Generation]: Display algorithms

Keywords: Volume Rendering, Unstructured Grids, Interval Volumes, Time-Varying data, Projected Tetrahedra, Shirley-Tuchman

1 Introduction

With the widespread use of high performance computing systems, some application simulations are capable of producing large datasets. These simulations tend to be time varying, adding another dimension to the problem. Additionally, these simulations produce multiple attributes like density, momentum and energy at each of the sample points (nodes). It is valuable to visualize the interrelationships between these values in the current geometric context.

Interactive visualization of curvilinear and unstructured data sets is critical and has been an active area of research for quite some time now. Maximum interactivity has been achieved using massively parallel supercomputers [8][20][21] to render the data in parallel using image-order [6] and object-order [7] decomposition techniques. Alternatively, the unstructured grids can be resampled into regular rectilinear grids and then rendered taking advantage of hardware accelerated rendering using 3D textures [18]. Along with the need for interactivity, there is a need for better data visualization tools, which allow navigating the data set in a more intuitive manner, as well as allow for correlation between the multiple attributes.

In this paper, we try to address some of these issues by using interval volumes as a region-of-interest extraction algorithm and by using fast volume visualization techniques. We show how interval volumes can be computed interactively for arbitrary polyhedral cells using a fast isosurfacing algorithm. The isosurfacing algorithm is completely automatic [3][5] and does not require either manually triangulating the surface patches [19][25] or performing a simplicial decomposition of the polyhedral cells [1][24][30][31]. We show that this algorithm extends to general convex polyhedra and prove that our algorithm is correct. We further build upon this work to come up with new visualization techniques for effective visualization of interval volumes and present results from 3-dimensional and 4-dimensional (time-varying + 3-spatial dimensions) grids with hexahedral and tetrahedral cells. Compared with other direct volume rendering methods, this interval volume method can segment the volume data and highlight the boundary surfaces between the interval volumes. Specifically, our main contributions described in this paper are as follows:

1. Improved interval volume calculation and triangulation based on higher-dimensional iso-contouring.
2. Proof-of-correctness of the interval volume algorithm.
3. Using interval volumes to segment the volume, highlight contour surfaces, and render the data effectively.
4. Application of interval volumes for both structured and unstructured grids, as well as multi-attribute and time-varying data.
5. Application of interval volumes to flow visualization.

In the following sections we look at some of the previous work done in this domain. We then give an overview of our interval volume computation algorithm, followed with visualization techniques for our interval volumes. We end with some results of our work and present future directions of research in this domain.

2 Previous Work

Unstructured Volume Rendering – Shirley and Tuchman [28] presented a new algorithm for hardware accelerated rendering of unstructured tetrahedral grids by approximating the projection to screen space using a set of triangles. Grids consisting of different cells are first decomposed into a tetrahedral representation using simplicial decomposition techniques [1][24]. Williams extended Shirley-Tuchman's approach to implement direct projection of other polyhedral cells in their HIAC rendering system [35] and used high accuracy light integration functions to model the light transport through the medium [34]. Recently, with the advent of programmable graphics hardware, a tremendous amount of work has been done in implementing the Shirley-Tuchman algorithm on graphics hardware using the programmable vertex and fragment

[†]{praveenb}@sgi.com
[*]{zhangc, xue, crawfis, wenger}@cis.ohio-state.edu

IEEE Symposium on Volume Visualization and Graphics 2004
October 11-12, Austin, Texas, USA
0-7803-8781-3/04/$20.00 ©2004 IEEE

shader pipelines on the GPUs [32][37]. In all of the above cases, the rendering performance of the projected tetrahedra algorithm is typically proportional to the number of cells to be rendered. The rendering process involves visibility sorting (usually $O(n\log n)$) and projection ($O(n)$) of the polyhedral cells. As an alternative to projection, polyhedral cells can be sliced to compute a polygonal approximation for hardware-accelerated rendering [39][10].

Interval Volumes - An *interval volume* is the set of points in a scalar field enclosed between two isosurfaces defined by two different isovalues. In [22], Max et. al. split each linear tetrahedron into interval based upon the transfer function. Fujishiro [12] introduced interval volumes as a solid fitting algorithm. A few applications of interval volumes were presented in [14][13]. Fujishiro computed a tetrahedralization of the interval volume by computing the intersection of two convex polyhedra enclosed by the isosurfaces given by the Marching cubes algorithm [19], within each cell. Nielson [25] computes the tetrahedralization by first decomposing each cube in the grid to five tetrahedra. Nielson then uses an efficient lookup table to compute the interval volume within each simplex and decompose it into tetrahedra. The tetrahedralization is constructed manually by analyzing all the possible intersections of a tetrahedron with an interval enclosed by two isosurfaces. Banks [2] counts the cases for a family of visualization techniques, including iso-contours and interval volumes. Ji [17] tracks the interval volumes using higher dimensional isosurfacing.

In this paper, we use interval volumes to create disjoint volume segments, or intervals. We show how interval volumes provide a more compact representation of the regions-of-interest in both structured as well as unstructured datasets. In addition to reducing the cell count, interval volumes provide a segmentation of the data into easily discernable regions. We feel this material layer interface provides an advantage over smoothly varying transfer-functions, which are often too fuzzy to display distinct material boundaries. In this paper, we present many different schemes, employed for interactive visualization of large time-varying data sets, using fast interval volume construction coupled with projection based volume rendering.

3 Interval Volume Computation

In [3], we presented a new algorithm for computing isosurfaces in arbitrary dimensional data sets. The algorithm proceeds by generating isosurface patches within each d-dimensional polyhedral cell comprising the d-dimensional grid. The output of the algorithm is a set of (d-1)-dimensional simplices forming a piecewise linear approximation to the isosurface. The algorithm constructs the isosurface piecewise within each cell in the grid using the convex hull of an appropriate set of points. In [5] we present a proof of correctness for the n-dimensional isosurface construction and show that it correctly produces a triangulation of a (d-1)-manifold with boundary.

For a function $f(x,y,z)$ sampled on a three dimensional grid, the interval volume [12] is defined by $I_f(\alpha,\beta) = \{(x,y,z)\colon \alpha \le f(x,y,z) \le \beta\}$. More generally, for a function $f : R^d \rightarrow R$ in any dimension, the interval volume is defined by $I_f(\alpha,\beta) = \{(x_1,\ldots,x_d)\colon \alpha \le f(x_1,\ldots,x_d) \le \beta\}$. Intuitively, the interval volume is the set of points enclosed between the two isosurfaces corresponding to the isovalues, α and β. For a d-dimensional grid, the interval volume is a d-dimensional subset of the grid and can be represented by a collection of d-simplices.

The interval volume can be represented as the projection of an isosurface in one higher dimension. Let π be the projection function mapping R^{d+1} to R^d given by $\pi(x_1,\ldots,x_d,x_{d+1}) = (x_1,\ldots,x_d)$.

Theorem 1: Given a continuous function $f : R^d \rightarrow R$ and two scalar values $\alpha < \beta$, if $F(x_1,\ldots,x_d,t) = f(x_1,\ldots,x_d) - (\alpha\,(1-t) + \beta\, t)$ and S is the isosurface given by $F(x_1,\ldots,x_d,t) = 0$ for $0 \le t \le 1$, then $\pi(S)$ is a 1-1 mapping of S onto the interval volume $I_f(\alpha,\beta)$.

Proof: The isosurface S is given by the equation $f(x_1,\ldots,x_d) - (\alpha\,(1-t) + \beta\, t) = 0$ or, equivalently, $f(x_1,\ldots,x_d) = (\alpha\,(1-t) + \beta\, t)$. Since t is between 0 and 1, inclusive, $f(x_1,\ldots,x_d)$ is between α and β, inclusive, for every point on S. Thus $\pi(S)$ is a subset of $I_f(\alpha,\beta)$. For every point $(x_1,\ldots,x_d) \in I_f(\alpha,\beta)$, we have $f(x_1,\ldots,x_d)$ equals some γ where $\alpha \le \gamma \le \beta$. Choosing some t_γ such that $\alpha\,(1- t_\gamma) + \beta\, t_\gamma = \gamma$ gives the point $(x_1,\ldots,x_d, t_\gamma)$ which lies on S. Thus $\pi(S)$ equals $I_f(\alpha,\beta)$. For each point $(x_1,\ldots,x_d) \in I_f(\alpha,\beta)$, the scalar t_γ is unique and thus $\pi(S)$ is 1-1.
Q.E.D.

Theorem 1 leads directly to an interval volume construction algorithm using isosurface construction. The scalar field is lifted into one higher dimension, an isosurface is constructed in that higher dimension, and the isosurface is projected back down into the original dimension.

The interval volume algorithm proceeds as follows:

1. Let $f(x_1,\ldots,x_d)$ define a d-dimensional function.

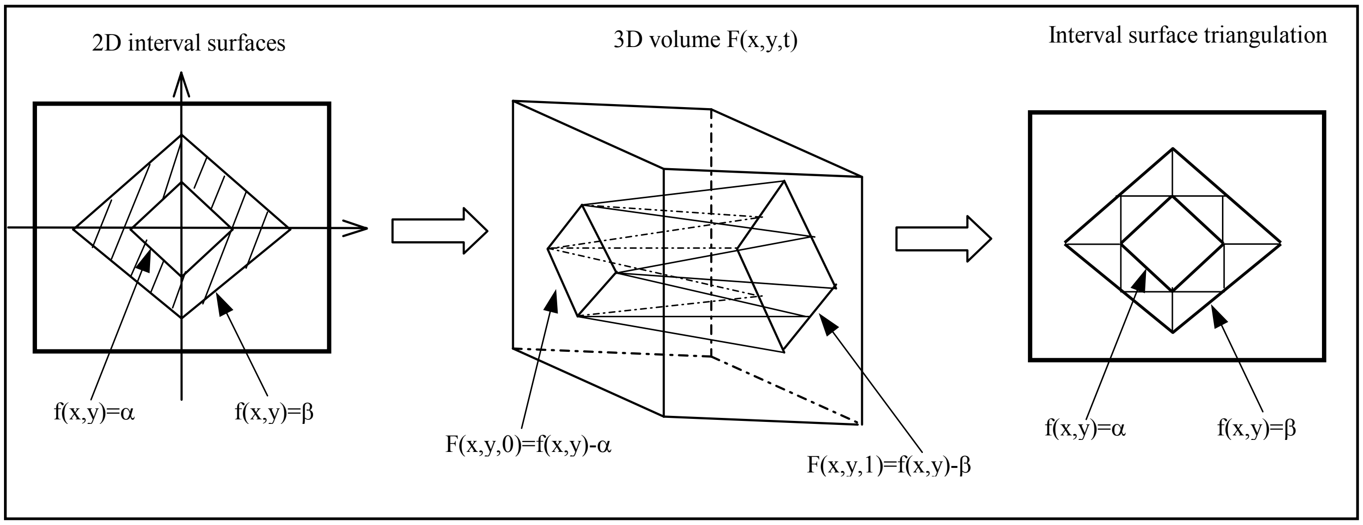

Figure 1. Two-dimensional illustration of the interval volume algorithm

2. Let scalar values, α, β ($\alpha < \beta$), be the desired isovalues bounding the interval.
3. Let $F(x_1,\ldots,x_d, t)$ be the $(d+1)$-dimensional function, given by, $F(x_1,\ldots,x_d,t) = f(x_1,\ldots,x_d) - (\alpha(1-t) + \beta t)$, such that

$$F(x_1,\cdots,x_d,t) = \begin{cases} f(x_1,\cdots,x_d) - \alpha, & for\ t = 0 \\ f(x_1,\cdots,x_d) - \beta, & for\ t = 1 \end{cases}$$

4. Compute the zero-valued isosurface, S, given by $F(x_1,\ldots,x_d, t) = 0$ for $0 \le t \le 1$.
5. Let π be the projection function mapping R^{d+1} to R^d given by $\pi(x_1,\ldots,x_d,x_{d+1}) = (x_1,\ldots,x_d)$. The desired interval volume, $I_f(\alpha,\beta)$, is then given by $\pi(S)$.

For a regular three dimensional scalar grid with hexahedral cells, we construct the interval volume by lifting the hexahedral cells to four dimensional hypercubes and building the isosurface piecewise within each hypercube[3]. The interval volume is then given by projecting the isosurfaces to R^3. In our implementation, we pre-compute a lookup table for all possible intersections of the 4-dimensional isosurface with a 4-cell. See [29] for the lookup table generation code. If a 3-cell has n vertices, we compute the interval volume by lifting the 3-cell to a 4-cell with $2n$ vertices. The isosurface lookup table for a cell with $2n$ vertices has $2^{2n} = 4^n$ cases. However, some of these cases are never used in interval volume generation. Actually, only 3^n of the cases can contribute to the interval volume computation. See [17] for more details.

Once the table has been pre-computed, we use this lookup table to compute the isosurface triangulations, piecewise, within each 4-cell. This approach provides considerable speedup to the isosurface construction process.

Figure 1 provides a two-dimensional analogy for the interval volume algorithm. Here, the 2D function is $f(x,y)=|x|+|y|$, and we are interested in the region $\alpha \le f(x,y) \le \beta$. We construct a 3D volume according to the step 3 above. The zero-valued isosurface in the volume is the triangulation using an isosurfacing algorithm, like ours [3]. The final interval surface enclosed by the α– and β–valued isocontours is given by the 2D projection of the isosurface triangulation. Since two isocontours for different isovalues can never intersect, we avoid flipped triangles in the resulting mesh.

In order to visualize a larger range of isovalues ($\alpha_1, \alpha_2, \ldots, \alpha_n$), our approach can be extended to tetrahedralize the volume using n steps along the t axis. Thus, the function, $F(x_1,\ldots,x_d,t)$, is given by

$$F(x_1,\ldots,x_d,t) = f(x_1,\ldots,x_d) - (\alpha_i + (\Delta\alpha_i / \Delta\tau_i)(t - t_i)) \quad \text{for } t_i \le t < t_{i+1}$$

where,

$$\Delta\alpha_i = \alpha_{i+1} - \alpha_i \text{, and } \Delta\tau_i = t_{i+1} - t_i$$

This gives us our necessary interpolating property:

$$F(x_1,\ldots,x_d,t_i) = f(x_1,\ldots,x_d) - \alpha_i$$

This allows multiple intervals within the volume to be visualized consecutively, showing the different contour surfaces at the intersection of the adjacent intervals.

Since the isosurface triangulation is consistent, the interval volume triangulation will also be consistent. The problem of face mismatches across cell boundaries is often referred to as the *cracking problem* [25]. Albertelli [1] and Max [24] also address this issue and present algorithms for consistent tetrahedral decomposition of polyhedral cells. Our algorithm handles the cracking problem in the table generation stage by using a lexicographical ordering of the isosurface vertices and then building the convex hull incrementally, adding one vertex at a time in the specified order. This is similar to the scheme used by [25] and [24], which ensures canonical triangulations across cell boundaries and generates consistent meshes. The use of an isosurface lookup table makes the interval volume algorithm efficient.

Entries in the four dimensional isosurface lookup table can have as many as 26 simplices. However, as previously noted, not every four dimensional case can be realized by interval volumes. Our algorithm produces at most 22 simplices in the interval volume for any three dimensional cube. We contrast this with [25] which divides the cube into five tetrahedra and constructs the interval volume in each tetrahedron. Within each tetrahedron the interval volume can require up to six simplices giving a total of thirty tetrahedra for the cube. This total can be realized by the cube in Figure 2 whose black vertices have a value above the isovalue β and whose white vertices have avalue below the isovalue α.

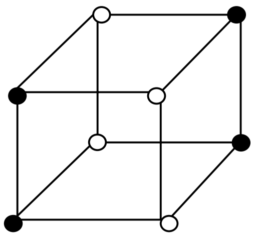

Figure 2. Cube (black vertices > β, white vertices < α)

3.1 Non-Hexahedral Grids

The higher dimension isosurface computation algorithm proposed in [3] is independent of the topology of the polyhedral cells in the grid. Hence, the same lookup table generation and interval volume computation algorithm can be applied to unstructured grids composed of other cell types including tetrahedra, pyramids and prisms [1]. When a 3-dimensional grid consisting of topological cubes (hexahedra) is used to compute the corresponding 4-dimensional grid, the resulting 4-cells are still topological cubes. This does not hold true for other types of grids. For a grid computed from tetrahedral cells, the 4-cells generated by our *dimension elevation* technique are not simplicial (4-tetrahedra). The new 4-dimensional grid consists of 4-prisms with tetrahedral faces. Each cell has 8 vertices and 16 edges as shown in Figure 3. The green and blue edges represent the tetrahedral faces corresponding to w=0 and w=1 and the maroon edges connect these two faces.

We use our lookup table generation algorithm to compute the lookup table for the 4-prism. A similar approach can be used to generate lookup tables for 4-prisms whose faces are 3-pyramids or 3-prisms. The isosurface lookup table shown in figure 3 has 256 (= 2^8) entries, but only 81 (= 3^4) entries are used

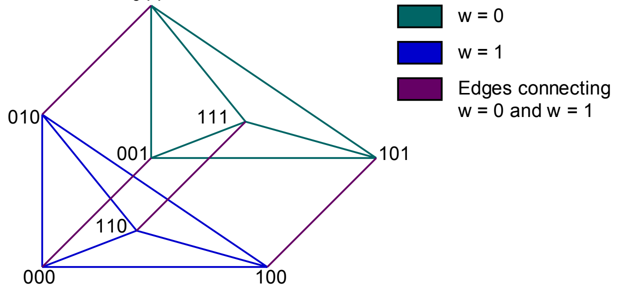

Figure 3. 4-prism with a tetrahedral base

for interval volumes. The maximum number of tetrahedra in the interval volume is 6 for any tetrahedron, which matches the maximum number of tetrahedra produced in [25]. The average number over the interval volume cases is approximately 3.9 tetrahedra.

3.2 Time-Varying Interval Volumes

Multiple time-steps add another dimension to the data set. For time varying data sets, interval volumes can be computed using two schemes. The first scheme computes the interval volumes separately for each time step of the data set using the algorithm discussed above. The user can then cycle through all the time steps to visualize and interact with the data in real time.

An alternative approach uses the following two-step algorithm. In the first step, we use our isosurfacing algorithm to compute a 4-dimensional volume representing a time-varying interval volume. This is accomplished by applying the isosurfacing algorithm directly on a 5-dimensional grid to generate a surface comprised of 4-simplices. In the second step, the 4-simplices are sliced along the time axis to interactively generate the interval volume for the corresponding time step. The slicing is achieved by applying an isosurfacing algorithm using the time value as an isovalue. This scheme is analogous to rendering time varying isosurfaces [3] and allows slicing at non-integral time-steps to compute interpolated interval volumes between consecutive time steps. As an alternative to slicing, we can volume render the 4-tetrahedra directly by using an opacity mapping or pseudo-coloring for the time variable.

4 Interval Volume Visualization

We use an implementation of the Projected Tetrahedron algorithm from Shirley and Tuchman [28] to render the interval volumes. The algorithm approximates a tetrahedron using one to four triangles depending on the screen projection of the tetrahedron’s vertices. We implement the Projected Tetrahedron algorithm using the vertex program of programmable graphics hardware [37]. For the visibility sorting, we use the MPVONC algorithm by Williams [33] which provides an $O(n\log n)$ algorithm for approximate visibility ordering of non-convex meshes. Even though the algorithm is not guaranteed to give correct results in the presence of certain *boundary anomalies*, it works well for our purposes.

We use a 2D texture to implement the volume rendering integral through a tetrahedron as suggested in [22]. The 2D texture stores a pre-computed exponential integration, which is then indexed according to the ray length through the projected tetrahedron’s thick vertex and the mean of scalar values at the ray entry and exit points. The table lookup and interpolation is done by the graphics hardware using per vertex texture coordinates. This scheme provides a simple approximation to the volume rendering integral described in [33]. The approximation to the optical model can be improved by using the pre-integrated volume rendering technique proposed by [26] which uses 3D textures to lookup a pre-computed light integral function.

4.1 Rendering Techniques for Interval Volumes

In this section, we build upon our work of interval volume construction to come up with new rendering techniques using interval volumes for effective visualization of volumetric data sets. We will first introduce the basic ones and then show our new shaders.

4.1.1 Constant Colored Intervals

We compute small intervals and use a constant color and opacity for rendering the complete interval by assigning the same isovalue (chosen to be the mean of the isovalues) to all the vertices in the interval volume. This has the advantage of accurate integration and the applicability of simplified and more efficient rendering algorithms. Wylie [37] and Weiler [32] both report rendering times for constant colored tetrahedra that are twice the speed of linearly varying tetraheda. The images in Figure 4 are interval volumes from the Tapered Cylinder data set rendered using this technique.

For multiple intervals, we assign each interval a distinct color value. Since these intervals are used to generate a single tetrahedral mesh for the cumulative interval, the grid needs to be sorted in a visibility order before rendering. For best results, we keep the number of intervals small and the adjacent colors visibly distinct. Figure 4a shows the data set with four intervals colored red, green, blue and yellow with constant opacities. A linear opacity ramp for the transfer function is used in Figure 4b, reducing the occlusion from the lower-valued intervals.

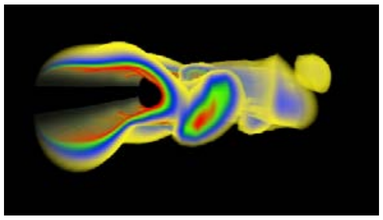
Figure 4a. Multiple constant colored intervals (236K tetrahedra)

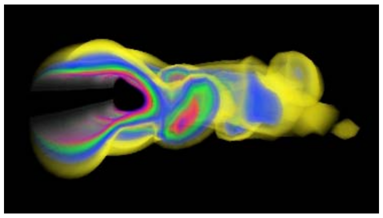
Figure 4b. Multiple constant colored, linear opacity intervals

4.1.2 Time-Cycled Intervals

With new hardware capable of rendering over one million cells per second, and the fact that interval volumes consume relatively few tetrahedra, real-time rates can be guaranteed for even the most refined of meshes by limiting the interval size. Thus, we can leverage time in our portrayal of the interval volumes. We build a separate display list or vertex array for each interval and loop through each interval, displaying it using a constant color. Since the cells have a constant color, the volume integration can be calculated in any order and no sorting is required. This is similar to the Data Slicer technique [11], where constant color and splatting are utilized.

Figure 5 shows three adjacent intervals rendered using this technique and a continuous transfer function applied which uses distinct colors for adjacent intervals. The interval volumes are computed directly from the hexahedral grid, without any decomposition. The intervals contain 46K, 57K and 61K tetrahedra respectively which can be rendered at approximately 20 frames per second. The minimal tetrahedral decomposition for the whole grid, using 5 tetrahedra per hexahedral cell [1] has approximately 615K tetrahedra. An MPEG movie

(TimeCycle.mpg) showing this technique is presented as supplementary material.

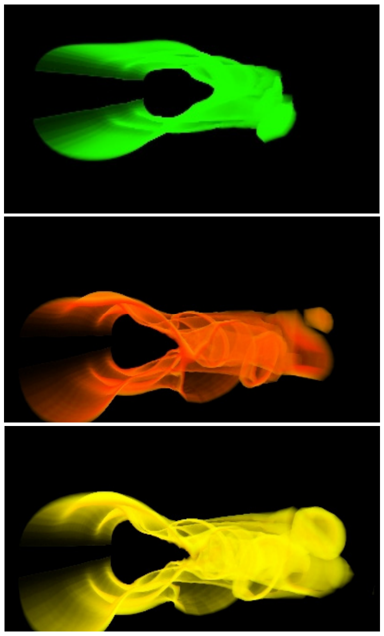
Figure 5. Interval volumes extracted by progressively increasing the mean interval value

4.1.3 Prioritized Intervals

A new technique is derived here from the medical imaging community. Maximum Intensity Projection or MIP prioritizes features of interest by modifying the ray integration function to select the scalar value with the greatest intensity [15]. The MIP algorithm prevents an important feature from being occluded by a less important feature by bringing the important feature to the forefront. We use a painters-like algorithm and let the user prioritize the intervals to ensure that the highest priority interval is the most visible. This is easily accomplished by sorting the intervals, not according to the viewing rays, but according to their priorities. Thus, we paint the higher priority intervals on top of the lower priority intervals. Much like MIP, when this is combined with interactive rendering, the motion parallax provides the necessary depth information, while the intervals of interest are always visible. Figure 6 shows snapshots of this technique applied to the Tapered cylinder data set. The priorities are reversed in the adjacent figures to show different features of the flow. An MPEG movie (MIP.mpg) showing this technique can be found in supplementary material. Unlike 4.1.1, in this case we treat the distinct intervals as separate tetrahedral meshes and render them independent of the other intervals. Hence, the tetrahedral meshes representing the individual constant colored intervals themselves do not need to be sorted.

4.1.4 Intervals with Textured Boundary Surfaces

Direct volume rendering of interval volumes might generate fuzzy-looking images for certain datasets. Highlighting the boundary surfaces between the interval volumes would allow us to provide a better mental segmentation of the volume and prevent important internal features from being occluded. Our interval volume computation algorithm offers the ability to compute the boundary surfaces between interval volumes without any computational overhead. The surfaces are extracted during the interval volume construction, simply by checking if the vertices

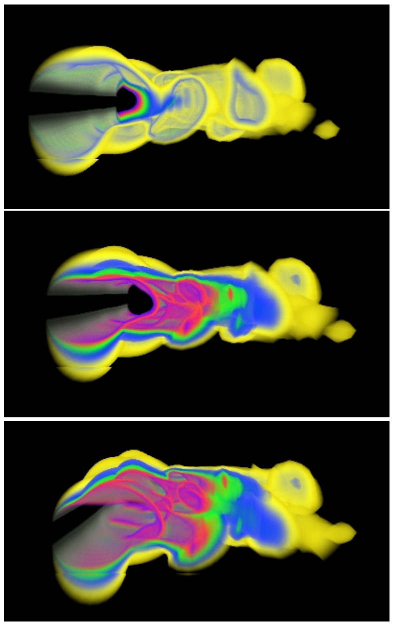
Figure 6. Prioritized intervals - first figure (Y, B, G, R) shows external surface of the flow, the next two (R, G, B, Y) show the internals of the flow using different viewpoints.

are on the boundary or not. This occurs when all vertices of a face have the value which is equal to one of the iso-values. This information is easily encoded into the isotables. The surfaces are rendered as semi-transparent polygons to prevent occlusion of internal features. Coupled with texture mapping and surface shading, these surfaces give better depth cues and make the visualization more informative for certain applications.

Figure 7 shows an interval volume with the inner boundary surface rendered in yellow with specular shading. Notice how the internal surface of the interval, which would otherwise be indistinguishable, is easily highlighted using our technique.

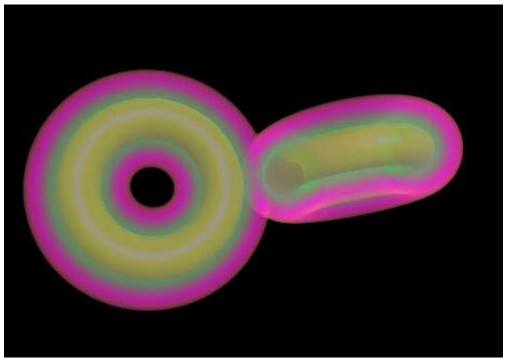
Figure 7. Interval volume with boundary surface highlighted

We also apply this technique to flow visualization. For a flow field, we first generate an implicit stream volume by pre-advecting the field and storing the advection information in the implicit volume. We then use interval volumes to render the regions-of-interest along with boundary surfaces between these intervals. The surfaces are shaded and textured using techniques to provide better segmentation of the volume and provide better flow information. Figure 8 shows multiple boundary surfaces with silhouette enhancement to help distinguish the surfaces from the volume. Figure 9 shows a textured boundary surface using 1D texture mapping. In this figure, the surface is a stream surface, and the texture shows the streamlines. A more detailed description on using interval volumes to extract flow volumes is presented in [38].

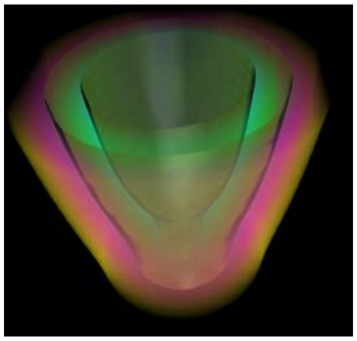

Figure 8. Boundary surface with silhouette enhancement

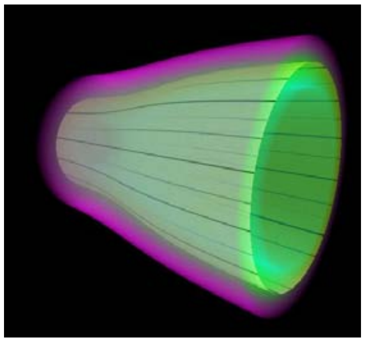

Figure 9. Boundary surface with 1D texture mapping

4.2 Time-Varying Data sets

As an alternative to traditional animation, multiple time-steps can be rendered within the same view [36]. This can be done by assigning different color values to each time step and then rendering them as multi-valued volume intervals. These intervals are smeared on top of each other. Fast moving regions have distinct colors as shown in Figure 10, which uses four consecutive time steps rendered using this technique.

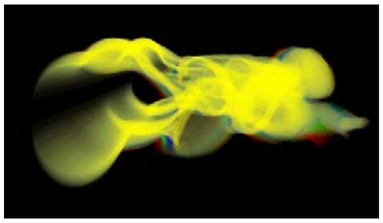

Figure 10. Four time-steps of the flow smeared on top of each other and rendered with different colors (R, G, B, Y)

4.3 Multi-Attribute Data sets

Computing the interval volume using one attribute and then rendering this volume using another set of attributes allows better spatial correlation between these attributes. Figure 11 shows an interval volume computed using density values and then rendered using the corresponding energy values. Animating this over time shows how the energy distribution changes in high-density regions of the grid.

The interval volume algorithm can also be extended to implement constructive solid geometry operations on multi-attribute data sets using multi-pass algorithms. For example, to implement an intersection between ranges of two scalars defined over the field,

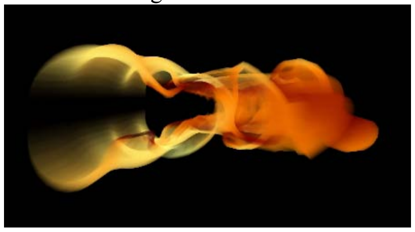

Figure 11. Multi-attribute visualization. Interval volume computed using density but rendered using energy.

we first extract an interval volume from the original grid using the range for the first scalar value. During the interval volume construction, we interpolate and store the second scalar values in the resulting grid as well. The output of the first pass is then used in a second interval volume construction pass using the range provided for the second scalar attribute. The resulting mesh would correspond to the geometric intersection of the two scalar ranges. The same algorithm can be applied in any dimension.

Figure 12 shows an example from a flow visualization application. Here, we consider two attributes: implicit value and advection time. Figure 12a and 12b are the interval volumes, computed using the implicit value and advection time in the flow, respectively. The stream volume is first generated using the range of implicit values as isovalues. The stream volume is then truncated to a range of the advection time steps by applying the interval volume algorithm to the volume in 12a. Figure 12c is the intersection of Figure 12a and 12b.

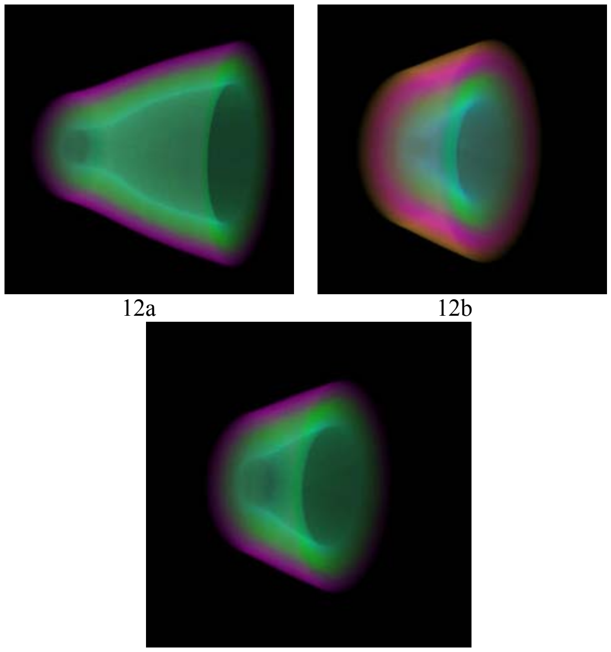

Figure 12. Intersection of interval volumes for two attributes

5 Results

All the results presented in this paper have been generated using a PC with a QuadroFX 3000 graphics card and a Pentium IV 3.4 GHz processor.

Table 1 presents interval volume lookup table statistics for various polyhedral cells. The table gives the maximum and average number of simplices over all possible combinations of vertex values.

Table 1. Interval volume lookup table statistics

Polyhedron	Dimension	Table Entries	Average Simplices	Maximum Simplices
Tetrahedron	3	3^4	3.96	6
4-simplex	4	3^5	8.35	14
5-simplex	5	3^6	17.27	30
Hexahedron	3	3^8	12.09	22

For the Tapered Cylinder data set, the current implementation of our algorithm takes approximately 172 milliseconds per time-step to compute the interval volume. This number was computed using the average over 20 time-steps (13000 to 13190) for a constant interval size (0.9934 - 0.9944) using the density attribute. The average number of tetrahedra generated in this case was approximately 55.9K per time-step. Our isosurfacing algorithm does a naive linear search through the grid cells for isosurface intersection. Preprocessing schemes [27] can be used to speed up the interval volume computation considerably by skipping empty cells. In the above case, the interval volumes intersect approximately 9150 cells on an average (the total number of tetrahedra is 55.9K and the average is 6 tetrahedra per cell), which is ~7.4 % of the number of cells in the grid. A histogram of the data set indicates that a large portion of the values have rather low and insignificant density values. In addition to saving valuable rendering cycles, interval volumes allow skipping these irrelevant regions, which could otherwise occlude other interesting features in the data set.

Using the hardware implementation of the Shirley-Tuchman algorithm [37], we are able to achieve a rendering rate of approximately 1200K tetrahedra per second for constant-color tetrahedra and 700K tetrahedra per second for linear-color tetrahedra.

The interval volume extraction time and the volume rendering time for the datasets are listed in Table 2. In this table, we select the isovalues which correspond to the images shown in this paper.

Table 2. Interval volume computation and rendering performance

Data set	Interval volume Construction time	Number of tetrahedra	Rendering time (frames per second)	
			Constant color	Linear color
Tapered Cylinder (curvilinear, 64x64x32)	172 ms	55.9K	21.5 fps	12.5 fps
Implicit flow Dataset (64x64x64)	377 ms	601.8K	2.0 fps	1.2 fps
Stream volume (601.8K tetrahedra)	343 ms	346.5K	3.5 fps	2.0 fps
Torus distance field Dataset (256x256x256)	6,127 ms	1,868.7K	0.64 fps	0.38 fps

Even with the implementation of hardware Projected Tetrahedra, the rendering performance is not fast enough for large dataset and/or thick interval volumes. One of our future goals is to use a tetrahedral simplification technique (like the TetFusion algorithm [9]) to improve the performance in these cases.

6 Conclusions and Future Work

In this paper, we have presented an algorithm for computing interval volumes in structured and unstructured grids using a fast isosurface extraction algorithm. We have shown how the algorithm can be used with convex polyhedra of arbitrary dimensions and also presented results on different 4 and 5-dimensional polyhedral cells. We have also shown how interval volumes can be used for interactive and more informative volume visualizations by providing distinct and discernable layers of volumetric material, either viewed together or as an animated sequence. Different rendering techniques have been demonstrated for interactive visualization of the data set.

We believe that our algorithm has the potential of being an essential component of volume visualization tools. The current algorithm can be augmented with feature detection techniques to aid the user in identifying useful/interesting intervals in the field. We also want to extend the concept of constructive solid geometry for multi-attribute data sets to do arbitrary operations like unions and subtractions.

7 Acknowledgements

The Tapered Cylinder, Blunt Fin and Oxygen Post data sets are part of NASA's online data set repository and can be obtained from http://www.nas.nasa.gov/Research/Datasets/datasets.html. The VolVis reviewers pointed out many grammatical errors and possible improvements to the text. Part of this work was supported by a US National Science Foundation Career Award (#9876022) and by NSF award #ACI-0222903.

References

[1] ALBERTELLI, G., AND R. A. CRAWFIS, *Efficient subdivision of finite-element datasets into consistent tetrahedra*, in Proceedings of IEEE Visualization '97, p.213-219, October 18-24, 1997, Phoenix, Arizona.

[2] BANKS, D., AND S. LINTON, Counting Cases in Marching Cubes: Toward a Generic Algorithm for Producing Substitopes, In Proceedings of IEEE Visualization 2003, pp. 51-58.

[3] BHANIRAMKA, P., R. WENGER, AND R. CRAWFIS, *Isosurfacing In Higher Dimensions*, in Proceedings of IEEE Visualization 2000, Ertl, Hamann, Varshney, Ed., IEEE Visualization Proceedings, 2000, 15-22.

[4] BHANIRAMKA, P., AND Y. DEMANGE, *OpenGL Volumizer: A Toolkit for High Quality Volume Rendering of Large Data sets*, in IEEE Volume Visualization, 2002, Boston, MA.

[5] BHANIRAMKA, P., R. WENGER, AND R. CRAWFIS, *Isosurface Construction in any dimension using convex hulls*, IEEE Transactions on Visualization and Computer Graphics, March/April, 2004, Vol 10, No, 2, pp 130-141.

[6] BENNETT, J., R. COOK, N. MAX, D. MAY, P. WILLIAMS, *Parallelizing a high accuracy hardware-assisted volume renderer for meshes with arbitrary polyhedra*, in Symposium on Parallel and Large-data Visualization and Graphics '01, San Diego, California.

[7] BURTON, L C., R. MACHIRAJU, AND D. S. REESE, *Dynamic View-Dependent Partitioning of Structured Grids with Complex*

Boundaries for Object-Order Rendering Techniques, in Parallel Visualization and Graphics, 1999, San Francisco, CA.

[8] CHEN, L., I. FUJISHIRO, AND K. NAKAJIMA, *Parallel Performance Optimization of Large-Scale Unstructured Data Visualization for the Earth Simulator*, in Eurographics Workshop on Parallel Graphics and Visualiztion, 2002.

[9] CHOPRA, P., AND J. MEYER, *TetFusion: An Algorithm for Rapid Tetrahedral Mesh Simplification,* In Proceedings of IEEE Visualization 2002, pp. 133-140.

[10] CHOPRA, P., AND J. MEYER, *Incremental Slicing Revisited: Accelerated Volume Rendering of Unstructured Meshes*, Proceedings of IASTED Visualization, Imaging, and Image Processing 2002, Málaga, Spain, pp. 533-538, Sept. 9-12, 2002.

[11] CRAWFIS, R. *Real-time Slicing of Data Space*, In Proceedings of IEEE Visualization 1996.

[12] FUJISHIRO, I., Y. MAEDA, AND H. SATO**,** *Interval volume: a solid fitting technique for volumetric data display and analysis,* in IEEE Visualization '95, Atlanta, GA, 1995.

[13] FUJISHIRO, I., Y. MAEDA, H. SATO AND Y. TAKESHIMA, *Volumetric data exploration using interval volume*, in IEEE Transactions on Visualization and Computer Graphics, 2 (June 1996).

[14] GUO, B. *Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction*, in Proceedings of IEEE Visualization '95, Atlanta, GA.

[15] HEIDRICH, W., M. MCCOOL, AND J. STEVENS, *Interactive Maximum Projection Volume Rendering,* In Proceedings of IEEE Visualization 1995, pp. 11-18.

[16] JESPERSON, D., AND C. LEVIT, *Numerical Simulation of Flow Past a Tapered Cylinder*, in RNR Technical Report, RNR-90-021, October,1990

[17] JI, G., H. SHEN, AND R. WENGER, *Volume Tracking using Higher Dimensional Isosurfacing,* In Proceedings of IEEE Visualization 2003, pp. 209-216.

[18] LEVEN, J., J. CORSO, J. COHEN AND S. KUMAR, *Interactive visualization of unstructured grids using hierarchical 3D textures*, in Symposium on Volume Visualization '02, Boston, MA.

[19] LORENSEN, W. E., AND H. E. CLINE, *Marching cubes: A high resolution 3d surface construction algorithm*, in M. C. Stone, ed., *Computer graphics*, 1987, Anaheim, California, July 1987, pp. 163-169.

[20] MA, K. L., AND T. W. CROCKETT, *A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data*, in IEEE Symposium on Parallel Rendering, '97, Phoenix, Arizona.

[21] MA, K. L AND T. W. CROCKETT, *Parallel Visualization of Large Scale Aerodynamics Calculations: A Case study on Cray T3E*, in IEEE Parallel Visualization and Graphics, 1999, San Francisco, CA.

[22] MAX, N., P. HANRAHAN AND R. CRAWFIS. *Area and volume coherence for efficient visualization of 3d scalar functions*, in Computer graphics, November 1990, pp. 27-33.

[23] MAX, N., B. BECKER, AND R. CRAWFIS*, Flow Volumes for Interactive Vector Field Visualization*, in IEEE Visualization '93, Los Alamitos, CA.

[24] MAX, N. *Consistent Subdivision of Convex Polyhedra into Tetrahedra*, in Journal of Graphics Tools, 6 (3), 29-36, 2002.

[25] NIELSON, G. M., AND J. SUNG, *Interval volume tetrahedrization*, in R. Y. a. H. Hagen, ed., IEEE Visualization '97, IEEE, November 1997, pp. 221-228.

[26] RÖTTGER, S., AND T. ERTL, *A two-step approach for interactive pre-integrated volume rendering of unstructured grids*, in IEEE Volume Visualization, '02, Boston, MA, pp. 23-28

[27] SHEN, H. W., *Isosurface extraction in time-varying fields using a temporal hierarchical index tree*, in IEEE visualization '98, IEEE, October 1998, pp. 159-166.

[28] SHIRLEY, P. AND A. TUCHMAN, *A polygonal approximation to direct scalar volume rendering*, in Volume Visualization Workshop, 1990, pp. 63-70.

[29] The Ohio State University. Isotable generation software. http://www.cse.ohio-state.edu/graphics/isotable.

[30] WEIGLE, C., AND D. BANKS, *Complex-valued contour meshing*, IEEE Visualization '96, IEEE, October 1996, pp. 173-180.

[31] WEIGLE, C., AND D. BANKS, *Extracting iso-valued features in 4-dimensional scalar fields*, 1998 Volume Visualization Symposium, IEEE, October 1998, pp. 103-110.

[32] WEILER, M., M. KRAUS, AND T. ERTL, *Hardware Based View-independent Cell Projection*, in Symposium on Volume Visualization, 2002, Boston, MA.

[33] WILLIAMS, P. *Visibility Ordering of Meshed Polyhedra*, in ACM Transactions on Graphics, 11 (4), 103-126, April 1992.

[34] WILLIAMS, P., *A Volume Density Optical Model*, in IEEE Volume Visualization Symposium, '92, 61-68

[35] WILLIAMS, P., N. MAX, C. M. STEIN, *A High Accuracy Volume Renderer for Unstructured Data,* in IEEE Transactions on Visualization and Computer Graphics 4(1): 37-54 (1998).

[36] WOODRING, J., C. WANG, AND H. SHEN, *High Dimensional Direct-Rendering of Time-Varying Volumetric Data,* In Proceedings of IEEE Visualization 2003, pp. 417-424.

[37] WYLIE, B., K. MORELAND, L. A. FISK, AND P. CROSSNO, *Tetrahedral projection using Vertex Shaders*, in Symposium on Volume Visualization, 2002, Boston, MA.

[38] XUE, D., C. ZHANG AND R. CRAWFIS, *Rendering Implicit Flow Volumes*, to appear in Proceedings of IEEE Visualization 2004.

[39] YAGEL, R., D. M. REED, A. LAW, P. W. SHIH AND N. SHAREEF, *Hardware Assisted Volume Rendering of Unstructured Grids by Incremental Slicing*, in Proceedings of Symposium on Volume Visualization, 1996.