Time-Varying Interval Volumes

Caixia Zhang*, Daqing Xue*, Roger Crawfis*, Rephael Wenger*

The Ohio State University

Abstract
In this paper, we study the interval segmentation and direct rendering of time-varying volumetric data to provide a more effective and interactive volume rendering of time-varying structured and unstructured grids. Our segmentation is based upon intervals within the scalar field between time steps, producing a set of geometrically defined time-varying interval volumes. To construct the time-varying interval volumes, we cast the problem one dimension higher, using a five-dimensional iso-contour construction for interactive computation or segmentation. The key point of this paper is how to render the time-varying interval volumes directly. We directly render the 4D interval volumes by projecting the 4D simplices onto 3D, decomposing the projected 4-simplices to 3-simplices and then rendering them using a modified hardware-implemented projected tetrahedron method. In this way, we can see how interval volumes change with the time in one view. The algorithm is independent of the topology of the polyhedral cells comprising the grid, and thus offers an excellent enhancement to the volume rendering of time-varying unstructured grids. Another advantage of this algorithm is that various volumetric and surface boundaries can be embedded into the time-varying interval volumes.

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

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. This paper mainly deals with the interval segmentation and rendering for time-varying structured and unstructured datasets.

A traditional method to render time-varying data is to take a snapshot of the data for each particular time step and generate an animation from the time series data. This method is useful, but it relies on human memory and cognitive abilities to tie together spatio-temporal relationships. An alternative method is to display the movement of the time series data in a single image using direct rendering of high dimensional data. Some people have worked on the hypervolume visualization [BPR*98] and high dimensional direct rendering of time-varying volumetric data [WWS03], but their algorithms do not apply for unstructured grids.

Interval volumes have been used to segment and render structured and unstructured volumetric data [BCZ*04]. It is a region-of-interest extraction algorithm, and fast volume visualization techniques are employed to render the interval volumes. Under the framework of the high-dimensional iso-contouring algorithm, interval volumes can easily be computed using two schemes for time varying data sets. The first scheme computes the interval volumes separately for each time step of the dataset using the algorithm discussed in [BCZ*04]. The user can then cycle through all the time steps to visualize the data. It is obvious that this scheme does not show the relationship of the interval volumes between different time steps well.

An alternative approach is to use the high-dimensional isosurfacing algorithm to compute a 4-dimensional volume representing a time-varying interval volume. This can be accomplished by applying the isosurfacing algorithm directly on a 5-dimensional grid to generate a surface comprised of 4-simplices. Then, we can render the 4-tetrahedra by slicing the interval volumes between the time steps. An alternative method, and the focus of this paper, is to render the 4-simplex from the interval volumes by integrating in the time directly, rather than projecting and slicing them.

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

2. Previous Work
Previous work that relates to our research primarily focuses on high-dimensional scientific visualization, unstructured volume rendering and interval volumes.

High-Dimensional Visualization: Hanson et al. [HH92][HH92*][HC93] introduced a general technique, as well as an interactive system, for visualizing surfaces and volumes embedded in four dimensions. In their method, 3D scalar fields were treated as elevation maps in four dimensions in the same way 2D scalar fields could be viewed as 3D terrains. Bajaj et al. [BPR*95] developed an
interface that provides “global views” of scalar fields independent of the dimension of their embedded space and generalized the object space projection technique into a hyper-volume projection method. Texture mapping hardware was utilized to directly render n-dimensional views of the global scalar field. Wooding et al. [WS03][WWS03] treated the time-varying data as four-dimensional data, and applied high dimensional slicing and projection techniques to generate an image hyperplane. The results of their technique generated a volume that is the projection of hyperplanes along a 4D projection vector, which can be rendered using traditional volume rendering techniques.

**Unstructured Volume Rendering:** Shirley and Tuchman [ST90] presented an 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 [AC97][Max02]. Williams extended Shirley-Tuchman’s approach to implement direct projection of other polyhedral cells in their HIAC rendering system [WMS98] and used high accuracy light integration functions to model the light transport through the medium [Wil92]. 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 shader pipelines on the GPUs [WKE02][WFM02][KQE04]. 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(nlogn)) and projection (O(n)) of the polyhedral cells. As an alternative to projection, polyhedral cells can also be rendered using ray casting [WKM03].

**Interval Volumes:** An interval volume is the set of points in a scalar field enclosed between two isosurfaces defined by two different isovalue. Fujishiro [FMS95] introduced interval volumes as a solid fitting algorithm. A few applications of interval volumes were presented in [Guo95][FMS96]. 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 [LC87], within each cell. Nielson [NS97] computed the tetrahedralization by first decomposing each cube in the grid to five tetrahedra. Nielson then used an efficient lookup table to compute the interval volume within each simplex and decompose it into tetrahedra. The tetrahedralization was constructed manually by analyzing all the possible intersections of a tetrahedron with an interval enclosed by two isosurfaces. Banks [BL03][BLS04] counted the cases for a family of visualization techniques, including iso-contours and interval volumes.

The above work is on the interval volumes of a single scalar field. For interval volumes with respect to a time-varying dataset, Ji et al. [JSW03] tracked the interval volumes using higher dimensional isosurfacings, and rendered an iso-contour surface of the interval volumes. They did not directly render the 4D interval volumes. In this paper, we work on the computation and direct rendering of the time-varying interval volumes. We use interval volumes to create disjoint volume segments, or intervals. Interval volumes provide a segmentation of the data into easily discernable regions. The direct rendering of the time-varying interval volumes makes it possible to get the distribution and relationship of the interval volumes across time steps, and help us to understand the time-varying structured and unstructured volumetric fields.

**3. Time-varying Interval Volume Computation**

In [BW00], 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 [FMS95] we present a proof of correctness for the d-dimensional isosurface construction and show that it correctly produces a triangulation of a (d-1)-manifold with boundary. Here, we give a short review of the algorithm. See [BZX04] for more details.

For a function \( f(x,y,z) \) sampled on a three dimensional grid, the interval volume [FMS95] is defined by \( I(\alpha,\beta) = \{ (x,y,z) : \alpha \leq f(x,y,z) \leq \beta \} \). More generally, for a function \( f : R^d \rightarrow R \) in any dimension, the interval volume is defined as:

\[
I(\alpha,\beta) = \{ (x_1, \ldots, x_d) : \alpha \leq f(x_1, \ldots, x_d) \leq \beta \}.
\]

Intuitively, the interval volume is the set of points enclosed between the two isosurfaces corresponding to the isovalue, \( \alpha \) and \( \beta \). 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 algorithm proceeds as follows:

1. Let \( f(x_1, \ldots, x_d) \) define a d-dimensional function.
2. Let scalar values, \( \alpha, \beta \) (\( \alpha < \beta \)), be the desired isovalue bounding the interval.
3. Let \( F(x_1, \ldots, x_d, w) \) be the (d+1)-dimensional function, given by:

\[
F(x_1, \ldots, x_d, w) = f(x_1, \ldots, x_d) - (\alpha (1-w) + \beta w),
\]

such that:

\[
F(x_1, \ldots, x_d, w) = \begin{cases} 
    f(x_1, \ldots, x_d) - \alpha, & \text{for } w = 0 \\
    f(x_1, \ldots, x_d) - \beta, & \text{for } w = 1
\end{cases}
\]

4. Compute the zero-valued isosurface, \( S \), given by \( F(x_1, \ldots, x_d, w) = 0 \) for \( 0 \leq w \leq 1 \).
5. Let \( \pi \) 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, \( I(\alpha,\beta) \), is then given by \( \pi(S) \).

For a time-varying scalar grid with hexahedral cells, the construction of the time-varying interval volumes is a five-dimensional isosurfacings problem. Given a four dimensional scalar field \( f(x,y,z,t) \), the interval volume consists of all the points which satisfy \( \alpha \leq f(x,y,z,t) \leq \beta \).
Following the above interval volume algorithm, in order to compute the time-varying interval volume, we first create a five dimensional scalar field $F(x,y,z,t,w)$, such that $F(x,y,z,0) = f(x,y,z) \cdot \alpha$ and $F(x,y,z,1) = f(x,y,z) \cdot \beta$. Then, the interval volume $\alpha \leq f(x,y,z,t) \leq \beta$ can be extracted by first computing the zero isosurface of the five dimensional function $F(x,y,z,t,w)$, and then projecting the resulting isosurface along the w axis to four dimensional space.

Here, we should note that the entries of the isosurface lookup table for 5D hypercubce are too large to be stored in the main memory. Since a 5D hypercube contains 32 vertices, the size of the table will contain $2^{32} = 4G$ entries. As pointed out in [JSW03], not all the four billion cases are possible. Only $3^{16} = 43M$ entries are possible for interval volumes. However, this size may be still too large to be processed in core. One solution is to compute the entries of the lookup table at runtime and cache them into a hash table which is small enough to fit into the main memory. In this paper, we use this caching method to store the 5D isosurface lookup table. See [OSU] for the lookup table generation code.

Since the isosurface triangulation is consistent, the interval volume triangulation will also be consistent. Our algorithm guarantees the consistency 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 [NS97] and [Max02], which ensures canonical triangulations across cell boundaries and generates consistent meshes. However, we still have to worry about the decomposition from 4-simplices to 3-simplices for the purpose of rendering. And we will address this problem in the next section.

4. Time-varying Interval Volume Rendering

After we extract the 4-simplices comprising the 4D interval volume, one rendering possibility is to slice the 4-simplices parallel to the time axis to generate 3-simplices (i.e. 3D tetrahedra) for a corresponding time step. The resulting 3D tetrahedra can then be rendered. This scheme is analogous to rendering time-varying isosurfaces [BWC00], but allows slicing at non-integral time steps to compute interpolated interval volumes between consecutive time steps. Figure 1 shows one example of the time slicing. The left image and the right image are the interval volumes with respect to the two time steps $t_1$ and $t_2$. The middle image is the corresponding interval volume with the time value $t = (t_1 + t_2)/2$.

In this paper, we are more interested in the direct rendering of the 4-simplices extracted from the 5D isosurface lookup table, in order to understand the distribution and relationship of the time-varying interval volumes across time steps. Now the challenge is how to render the 4-simplices to the 2D image space. The following subsections will explain the projection of 4-simplices to 3-simplices and the projection of 3-simplices to image space.
This classification will guide us in decomposing the projected 4-simplices into tetrahedra for the purpose of rendering.

4.1.2 Tetrahedralization of projected 4-simplices

Our first attempt at this problem was to project 4-simplices to 3D along the time axis by simply ignoring the time value \((t)\) and keeping only the position information \((x, y, z)\) for each vertex. Then, the projected 4-simplices are decomposed into tetrahedra in 3D space based on the above classification in Figure 2. Table 1 was constructed by hand and shows the possible decomposition of each class.

After rendering the resulting 3D tetrahedra using the Projected Tetrahedron method, we found that the result was not correct. An image of a constant plate is shown in Figure 4. There are some obvious patterns on the plate. As we know, many 4-simplices are extracted from the lookup table for each hypercube cell, and then are projected to 3D and decomposed to tetrahedra. By keeping track of the tetrahedral components inside each cell, we find that each tetrahedron is in right place and the tetrahedra as a whole fill the cell. However, we also find that the projections of a set of 4-simplices overlap. For example, for a cube cell from our constant plate example as shown in Figure 4, some space is shared five times by the projection of 4-simplices, while other space is shared only four times. This uneven overlapping distribution of the projected 4-simplices causes a non-constant opacity throughout the cell.

**Table 1: Original decomposition of the projected 4-simplices**

<table>
<thead>
<tr>
<th>Class</th>
<th>Possible Tetrahedra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>P1P2P3P4 and P1P2P3P5, P1P2P4P6, P1P3P4P6, P2P3P4P6, P1P2P5P6, P1P3P5P6 and P2P3P5P6</td>
</tr>
<tr>
<td>Class 2</td>
<td>P1P2P3P4, or: P1P2P3P5, P1P2P4P5, P1P3P4P5 and P2P3P4P5</td>
</tr>
<tr>
<td>Class 3(a)</td>
<td>P1P2P3P5 and P1P3P4P5, or P1P2P4P5 and P2P3P4P5</td>
</tr>
<tr>
<td>Class 3(b)</td>
<td>P1P2P3P4, or: P1P2P4P5, P1P3P4P5 and P2P3P4P5</td>
</tr>
<tr>
<td>Class 4</td>
<td>P1P2P3P5, or: P1P2P4P5 and P2P3P4P5</td>
</tr>
<tr>
<td>Class 5</td>
<td>P1P2P3P4</td>
</tr>
</tbody>
</table>

© The Eurographics Association 2005
The key observation in the incorrect opacity is that the length of the projection through time cannot be ignored during the projection of the 4-simplices along the time axis. During the projection, each vertex obtains a ∆t value. The value ∆t is calculated in a similar way as the calculation of ∆z in the projected tetrahedron algorithm, but along the time dimension. The basic idea is that a ray is cast along the time projection and ∆t is calculated as the length of the ray that passes through the 4-simplices. Here the projection is from 4D to 3D. So, a vertex in 3D has a non-zero ∆t value if it has two different t values along the ray in the t dimension and the ∆t is calculated as the difference of the two t values. That means, one vertex has a non-zero ∆t if it is overlapped with another vertex in 3D (here, the vertex can be an original projected vertex or a point which is interpolated by other projected vertices after the projection to 3D). Vertices with a non-zero ∆t value are illustrated by the black points in Figure 5 for each case of the projected 4-simplices.

In class 5, P4 and P5 are coincident after the projection and P4 has a non-zero ∆t. In class 4, P4 has a non-zero ∆t value which is the difference of P4,t and the interpolated t value between P1 and P3. In class 3(a), the new vertex P6 which is the intersection point of the lines P1P3 and P2P4 has a non-zero ∆t value which is the difference of two interpolated t values between P1P3 and P2P4. While for class 3(b), the ∆t at P5 is non-zero which is equal to the difference of P5,t and the t value interpolated inside the triangle P1P2P3. Similarly, for class 2, the ∆t value at P5 is the difference of P5,t and the interpolated t value inside the tetrahedron P1P2P3P4. In class 1, the new vertex P6 is the intersection point of the triangle P1P2P3 and the line P4P5. The ∆t value at P6 is the difference of the interpolated t value inside P1P2P3 and the interpolated t value along the line P4P5.

After determining the vertex with a non-zero ∆t for each class, the decomposition of the projected 4-simplices into tetrahedra should make sure that the vertex with a non-zero ∆t is one vertex of the decomposed tetrahedra. Now the decomposition becomes a unique process. The unique decomposition is listed in Table 2 for each class of 4-simplices. For each decomposed tetrahedron, one vertex has a non-zero ∆t value and each point inside the tetrahedron has an interpolated ∆t value. The ∆t distribution inside the tetrahedron also contributes to the final opacity of the rendered tetrahedron.

### Table 2: Final decomposition of the projected 4-simplices

<table>
<thead>
<tr>
<th>Class</th>
<th>Decomposed Tetrahedra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>P1P2P4P6, P1P3P4P6, P2P3P4P6, P1P2P5P6, P1P3P5P6 and P2P3P5P6</td>
</tr>
<tr>
<td>Class 2</td>
<td>P1P2P3P5, P1P2P4P5, P1P3P4P5 and P1P2P5P6</td>
</tr>
<tr>
<td>Class 3(a)</td>
<td>P1P2P3P5, P2P3P5P6, P3P4P5P6 and P1P4P5P6</td>
</tr>
<tr>
<td>Class 3(b)</td>
<td>P1P2P4P5, P1P3P4P5 and P2P3P4P5</td>
</tr>
<tr>
<td>Class 4</td>
<td>P1P2P3P4 and P2P3P4P5</td>
</tr>
<tr>
<td>Class 5</td>
<td>P1P2P3P4</td>
</tr>
</tbody>
</table>

#### 4.2 Projection of 3-simplices to image space

We use an implementation of the Projected Tetrahedron algorithm from Shirley and Tuchman [CST90] to render the projected tetrahedra from the 4-simplices. The algorithm approximates a tetrahedron using one to four triangles depending on the screen projection of the tetrahedron’s vertices. We implement the PT algorithm using a vertex program in programmable graphics hardware [WMF*02].

Compared to the projection of the normal tetrahedra, there is one difference in the rendering of the time-varying interval volumes: the tetrahedra here have a non-constant ∆t distribution from the projection along the time axis. Therefore, when we calculate the opacity of the projected triangles, we should consider both the contribution of the ∆t for the projection along the time axis and the contribution of the ∆z for the projection along the z-axis.

The transparency along a ray passing through a tetrahedron is represented as following for the rendering of the 4-simplices:

\[
T = \exp\left(-\int_{\tau}^{\tau+\Delta t} \int_{\Delta z} \right) = \exp\left(-\int_{\tau}^{\tau+\Delta t} \int_{\Delta z} \right)
\]

Here, ∆t is the extinction coefficient. The opacity along a ray is represented as

© The Eurographics Association 2005
\[ \alpha = 1 - e^{-\tau \Delta t \Delta z} \]  

Figure 6(a) shows a constant tetrahedron P1P2P3P4, which is composed of four projected class-5 4-simplices, each with a non-zero \( \Delta t \) value at one vertex (represented as black points). By adding the interpolated \( \Delta t \) values from four tetrahedra, every point \( P \) inside the tetrahedron has a constant \( \Delta t \) (as shown in equation 3). Figure 6(b) shows the distribution of the \( \Delta t \) inside the tetrahedron. This is what we expect for a constant tetrahedron which is composed of four projected 4-simplices extracted from a 5D isosurface lookup table.

\[
\begin{align*}
\Delta t_p &= \Delta t_1 + \Delta t_2 + \Delta t_3 + \Delta t_4 \\
&= \frac{\text{vol}_{P3P4P} \times \Delta t_1 + \text{vol}_{P1P4P} \times \Delta t_2 + \text{vol}_{P1P3P} \times \Delta t_3}{\text{vol}_{P1P2P3P4}} \\
&= \Delta t \quad (\text{if } \Delta t_1 = \Delta t_2 = \Delta t_3 = \Delta t_4 = \Delta t)
\end{align*}
\]  

The transparency along a ray passing through any point \( P \) inside the constant tetrahedron is:

\[
T = T_1 \cdot T_2 \cdot T_1 \cdot T_4 = e^{-\tau \Delta t \Delta z} \cdot e^{-\tau \Delta z \Delta t} \cdot e^{-\tau \Delta t \Delta z} \cdot e^{-\tau \Delta z \Delta t} = e^{-\tau \Delta t \Delta z}
\]  

The opacity along the ray is \( \alpha = 1 - e^{-\tau \Delta t \Delta z} \). It shows the opacity along a ray passing through any point \( P \) inside the constant tetrahedron depends on the constant \( \Delta t \) inside the tetrahedron and the \( \Delta z \) from the projection along the \( z \)-axis.

Since the zero-thickness vertices in PT algorithm do not necessarily have zero \( \Delta t \) thickness, and the vertex with non-zero thickness in PT algorithm may have zero thickness of \( \Delta t \), so we cannot directly multiply the \( \Delta t \) and the \( \Delta z \) at each vertex and then interpolate it inside the projected triangles. Actually, the bi-variant function should be evaluated at each pixel. That means, we should multiply the interpolated \( \Delta t \) and the interpolated \( \Delta z \) for each pixel inside the projected triangles. We develop a modified implementation of the Shirley and Tuchman algorithm using the vertex and fragment programs to consider both the contributions of the \( \Delta t \) and the \( \Delta z \). In the vertex program, we calculate the \( \Delta t \) and \( \Delta z \) for each vertex of the projected triangles, then their contributions to the opacity are multiplied in the fragment program for each pixel.

Considering the contribution of the \( \Delta t \) on the opacity, the data in Figure 4 is rendered correctly as in Figure 7.

### 4.3. Visualization techniques of time-varying interval volumes

In this section, we build upon our work of the computation and projection of the time-varying interval volumes to come up with some visualization techniques for effective visualization of the time-varying volumetric data sets. As discussed in previous sections, the tetrahedra for time-varying interval volumes have \( \Delta t \) distribution from the projection along the time axis and they overlap themselves in 3D space. This causes some occlusion and compositing problems. In this section, we will figure out the suitable visualization techniques for the time-varying interval...
volumes. For Figures 8 to 16, please also see the color plate.

4.3.1. Direct rendering of the time-varying interval volumes

We can render the time-varying interval volumes directly from the extracted 4-simplices, using the projection methods as discussed in sections 4.1 and 4.2. Since the time-varying interval volumes actually project to the same three-dimensional space (i.e., it is a self-intersecting volume), no accurate sorting is possible. In this section, an additive compositing operator is used to blend the 4-simplices into the image. The color of the vertex is encoded using the time value. For the overlapped vertices (as shown with the black points in Figure 5), the time value is calculated as the average \( t \) value of the two overlapping vertices.

Figure 8 is an example of the direct volume rendering result of a simple test function comprised of a linear ramp in time. Here, the color at \( t=t1 \) is green, and the color at \( t=t2 \) is red. The color between \( t1 \) and \( t2 \) is encoded between green and red. From this figure, we can see the transition from the green to the yellow, and to the red as the field moves over time.

4.3.2. Highlighted surface boundaries

Similar to the interval volumes with embedded boundary surfaces in [BCZ*04], we can embed the boundary isosurfaces into time-varying interval volumes to highlight interior features. The boundary surfaces are extracted during the construction of the time-varying interval volumes without extra computation cost, simply by checking if the vertices are on a boundary or not.

For time-varying interval volumes, there are two types of boundaries: volumetric boundaries and surface boundaries. In this subsection, we first consider the surface boundaries. Given a time-varying interval volume defined by two isovalue \( \alpha \) and \( \beta \), and two time steps \( t1 \) and \( t2 \), there are four boundary isosurfaces at: (a) \( t=t1 \) and \( f(x,y,z,t)=\alpha \), (b) \( t=t1 \) and \( f(x,y,z,t)=\beta \), (c) \( t=t2 \) and \( f(x,y,z,t)=\alpha \), (d) \( t=t2 \) and \( f(x,y,z,t)=\beta \). Since these surfaces are the boundary of the tetrahedra which compose the time-varying interval volume, these boundary surfaces are rendered together with the 4D interval volume. The four isosurface boundaries are illustrated in Figure 9 in the above order (a) to (d) from left to right. From the figure, we can see how the isosurfaces change with time and with value.

4.3.3. Volumetric boundaries

There are also four kinds of volumetric boundaries for a time-varying interval volume defined by two isovalue \( \alpha \) and \( \beta \), and two time steps \( t1 \) and \( t2 \): (a) time-varying isosurfaces at \( f(x,y,z,t)=\alpha \) and \( t1 \leq t \leq t2 \), (b) time-varying isosurfaces at \( f(x,y,z,t)=\beta \) and \( t1 \leq t \leq t2 \), (c) interval volumes at \( \alpha \leq f(x,y,z,t) \leq \beta \) and \( t=t1 \), and (d) interval volumes at \( \alpha \leq f(x,y,z,t) \leq \beta \) and \( t=t2 \). The four volumetric boundaries are shown in Figures 10(left), 10(right), 11(left) and 11(right), respectively.

The volumetric boundaries are rendered using the normal projected tetrahedron algorithm, without the contribution of \( \Delta t \) on the opacity. Here, the boundary interval volumes are sorted, not according to the viewing rays, but according to their priorities (the time here) to bring an important interval volume at a specific time step to the forefront. This priority-based sorting technique comes from the idea of the Maximum Intensity Projection (MIP) in medical community. Also, a constant color is assigned to each volumetric boundary. Figures 15 and 16 are rendered using this technique.

5. Results

We apply the rendering and visualization techniques of the time-varying interval volumes explained in section 4 to several datasets. Figure 12 shows time-varying interval volumes of a vortex dataset, rendering the vorticity magnitude in the range of \((8.0, 12.0)\). The color is encoded using time \( t \): green at \( t=t1 \), red at \( t=t2 \), and yellow for overlapped regions between \( t=t1 \) and \( t=t2 \). From this figure, we can see how the interval volumes move over time. We can also notice that some new components are generated over time, such as the purely red one in Figure...
Figure 13 is the interval volumes of the vortex dataset for three time steps. The color mapping with time $t$ is in the following way: blue at $t=t_1$, green at $t=t_2$, red at $t=t_3$, cyan for overlapped regions between $t=t_1$ and $t=t_2$, yellow between $t=t_2$ and $t=t_3$. So, for the overlapped region among $t=t_1$, $t=t_2$ and $t=t_3$, the color is white using the additive compositing operator. In this figure, areas where contours are appearing over time are predominantly red, while areas that faded over time are predominantly blue. Areas which maintain a high isovalue over time appear white. Figure 14 shows time-varying interval volumes for the NASA Tapered Cylinder dataset, by rendering the density attribute in the range (0.982124, 0.9852195). This dataset is a curvilinear grid in PLOT3D format. Similarly, this figure shows the movement of the interval volumes with the time.

By rendering two interval volumes at $t=t_1$ and $t=t_2$ extracted from the volumetric boundary into one image using the MIP technique discussed in section 4.3.3, we can see how the interval volumes move with the time steps. In this way, an important interval volume at a specific time step is brought to the forefront, preventing being occluded by the interval volumes at other time steps. Figures 15 and 16 show two interval volumes at $t_1$ and $t_2$ for the vortex dataset and the Tapered Cylinder dataset in one view using the MIP technique. Here, yellow color represents the interval volume at $t_1$, and red color is for $t_2$. In the two figures, the interval volume at $t_2$ is given higher priority. By comparing the images using MIP and the corresponding time-varying interval volumes, we can interpret the images of time-varying interval volumes well.

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. The interval volume computation time and the volume rendering time for the datasets are listed in Table 3.

<table>
<thead>
<tr>
<th>Data set</th>
<th>4D interval volume construction and decomposition time</th>
<th>Number of 4-simplices</th>
<th>Number of tetrahedra (with volume)</th>
<th>Rendering time (linear color)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test function (2x20x10x10)</td>
<td>75ms</td>
<td>9,720</td>
<td>27,054</td>
<td>30ms</td>
</tr>
<tr>
<td>Vortex dataset (2x128x128x128)</td>
<td>12.2s</td>
<td>319,304</td>
<td>882,044</td>
<td>970ms</td>
</tr>
<tr>
<td>Tapered Cylinder (curvilinear 2x64x64x32)</td>
<td>18.5s</td>
<td>349,624</td>
<td>941,098</td>
<td>1,030ms</td>
</tr>
<tr>
<td>Vortex dataset (3x128x128x128)</td>
<td>22.5s</td>
<td>654,846</td>
<td>1,807,460</td>
<td>1,980ms</td>
</tr>
</tbody>
</table>

Table 3: 4D interval volume computation and rendering performance
In the Table 3, the 4D interval volume construction and decomposition time includes the time to calculate the entries of the isosurface lookup table, the time to construct 4-simplices and the time to decompose 4-simplices to tetrahedra. Due to the decomposition of 4-simplices to 3-simplices and the overlapping copies of the 3-simplices in 3D space, there are more tetrahedra for time-varying interval volumes.

6. Conclusions and Future Work
In this paper, we have presented an algorithm for computing time-varying interval volumes in structured and unstructured grids using a fast isosurface extraction algorithm. And we have explained the rendering methods of the 4-simplices by projecting and decomposing the 4-simplices to 3-simplices, and using a modified hardware-implemented projected tetrahedron method. In this way, we can render the time-varying interval volumes which integrate multiple time steps into a single view and we can see the movement of the interval volumes over time in one view. Different visualization techniques have been demonstrated for the visualization of the time-varying structured and unstructured data sets.

The current rendering technique uses the hardware implemented projected tetrahedron method [WMF*02]. We can use the new PT implementation presented in [KQE04] to improve the quality of the images and we can consider to speed up the projection from 4D to 3D by taking advantage of the modern graphics hardware. Also, 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 time-varying interval volumes.

Acknowledgements
The Tapered Cylinder data set is from NASA’s online data set repository. Part of this work was supported by NSF award #ACI-0222903.

References


