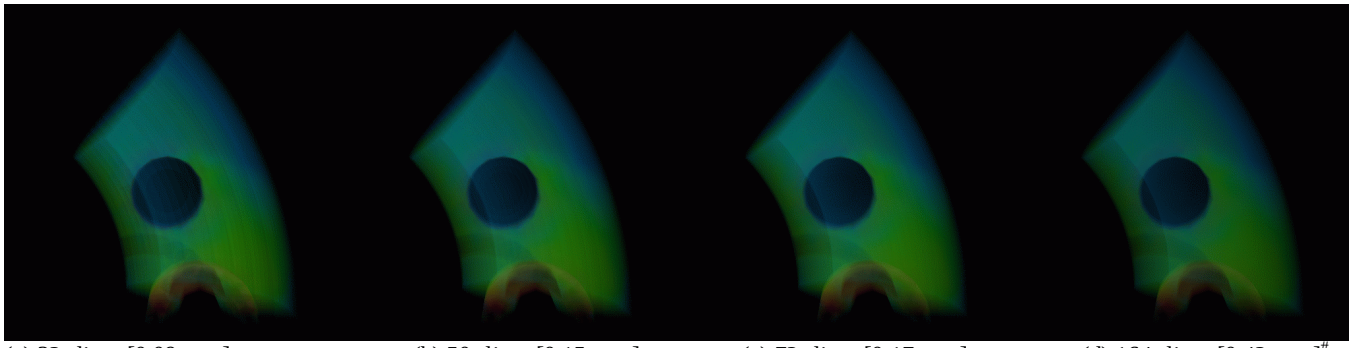
# Incremental Slicing Revisited: Accelerated Volume Rendering Of Unstructured Meshes

Prashant Chopra and Joerg Meyer*
Mississippi State University, Engineering Research Center



(a) 32 slices [0.09 secs] (b) 50 slices [0.15 secs] (c) 72 slices [0.17 secs] (d) 164 slices [0.42 secs][#]

**Figure 1:** Four *levels of detail* of the 12,936 tetrahedral cells Spx dataset, rendered on an SGI R10000 194MHz, 2048 MB RAM, using our accelerated incremental slicing method. The number in brackets behind the number of slices shows the actual slicing times for the respective images. [#]The number of slices to render image (d) was computed using Nyquist's sampling theorem.

## Abstract

We accelerate the incremental slicing method for near interactive volume rendering of unstructured grids with conventional polygon rendering hardware support. Our method uses less memory compared to the method of Yagel et al. [25], with significant improvements in rendering time. Using our method, a rendering performance gain of at least 400% was achieved for some datasets, e.g., the Blunt fin dataset. We completely eliminate sorting of edges as a preprocessing step. Further, we introduce a hierarchical *z*-region paradigm, which can be incorporated with the dataset-input module to get the depth information of the cells with no extra computational overhead. We argue that keeping an active edge list and an active cell list as data structures is redundant, because this information is inherent in the *z*-region paradigm in our method. In pursuit of accuracy, we explore the application of Nyquist's classical sampling theorem when slicing the datasets, and conclude that images of comparable visual quality can be obtained by relaxing this requirement, while maintaining the rendering performance. Finally, we conclude with the results of our method, and a discussion on how progressive refinement of 2-manifold triangular meshes (for 2D slices of cutting planes parallel to the view plane) can help to achieve additional interactivity in a volume visualization system.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – surfaces and object representations.

**Keywords:** volume rendering, multi resolution, level-of-detail, unstructured meshes.

* {prash | jmeyer}@erc.msstate.edu
http://www.cs.msstate.edu/~graphics

## 1 INTRODUCTION

The last decade has witnessed some successful attempts towards interactivity in volume visualization systems. However, interactive volume rendering still remains an active research arena; because of continuous advances in computational systems technology and methodology that have helped scientists run numerical simulations of ever increasing complexity, outputting high-dimensional datasets (e.g., time-varying volumetric datasets) in the form of high order simplicial complexes (polyhedral meshes), both homogeneous and heterogeneous. A nice example is an earthquake simulation dataset (11,800,639 vertices, 69,448,288 tetrahedra), with associated vector attributes for 800 time-steps per node [1, 9, 10]. To interactively explore such datasets still remains a challenge for the computer graphics research community.

Addressing these issues, a number of direct (near interactive) volume rendering techniques for unstructured polyhedral meshes have been explored and established in the recent past. A majority of these techniques addresses the rendering of homogeneous tetrahedral meshes. Two reasons explain this fact: *one*, the convex shape of a tetrahedral element, and its faces, makes it suitable for volume rendering on polygon-based rendering systems; and *two*, that all the other higher order simplicial complexes can be broken down to tetrahedra very conveniently. These near interactive direct volume rendering techniques can primarily be classified into two categories: object-space-based methods, and image-space-based methods.

The image-space-based methods fundamentally work by letting individual rays (one per image-space element) intersect the polyhedral elements, and then interpolating the attribute value at the points of intersection based on the local neighborhood vertex attributes. Such methods are computationally expensive, but yield high-quality images by guaranteeing that the dataset is sampled exactly.

The object-space-based methods, however, consider each object-space primitive (polyhedron) one by one, and shade a corresponding image space element by composition, in either back-to-front, or front-to-back order, from the view plane. Such methods are an approximation of the computationally more expensive image space methods, but are usually faster; and hence have been subjects of interest in the area of interactive volume rendering for quite some time. Despite being simpler than their image-space counterparts, even these approximation methods have not been able to achieve real-time volume rendering of large volumetric datasets unless specific hardware support is present. This is because a correct visibility ordering of the polyhedra is still required to compose the shading elements in the correct order to generate correct images.

The object-space-based methods can further be categorized into projection-based, and slicing-based methods. In this paper, we focus on these two categories. We discuss the advantages of one over the other, and why the domain of slicing-based methods is still an attractive alternative when approximate volume rendering is required. We show how a high level of interactivity is possible in volume rendering by simply exploiting conventional polygon-based rendering hardware, by proposing alternate data structures to eliminate computational and storage redundancy, and by minimizing the overheads to the rendering pipeline in the original slicing tetrahedra work by Yagel et al. [25].

Essentially, we present an improved slicing tetrahedron algorithm for interactive 3D image processing of unstructured volumetric meshes.

## 2 RELATED WORK

We classify recent publications on object-space-based methods in the area of near-interactive volume rendering (i.e., frame rates > 0.1 fps) broadly into two categories: projection-based methods [2, 3, 16, 17, 19, 23, 24], and slicing based methods [6, 21, 25]. (These categories do not take into account the methods that exploit special purpose hardware for volume rendering [13, 14, 20, 24].)

## 2.1 Projection-based Methods

Correct and fast visibility sorting techniques have for long been the subjects of research in this category. A number of attempts have been made in recent years to develop a general sorting algorithm, which is fast, accurate, and can be employed to a different number of datasets, i.e., is not restricted to a particular type of datasets.

Shirley and Tuchman [19] were amongst the first to attempt the use of polygon-rendering hardware support for approximate volume rendering of tetrahedral meshes. They employ visibility sorting based upon Williams' method [22], and project tetrahedra one by one in back to front order. However, the visibility-sorting algorithm by Williams [22] does not function correctly for all the kinds of datasets. It has limitations, for instance, it cannot guarantee correct visibility ordering for datasets with cycles. Further, it is not an accurate projection method, and was improved by subsequent publications that focused on correct and faster sorting of the elements [2, 18, 23].

Stein et al. [20] attempted to improve the visual quality of these approximations by employing hardware texture support, and a more accurate sorting algorithm that was bound by a time complexity of $O(n^2)$. Williams et al. [23] came up with an accurate sorting and projection algorithm, with still improved time performance. Visibility sorting performance was further improved by exploiting the spatial coherence between tetrahedral cells with respect to a sweep-plane [16, 17].

Some methods tried to achieve interactivity by employing accurate and fast sorting techniques for Delaunay meshes [24]. However, not all numerical simulations output perfect Delaunay-triangulated meshes, especially when a complex mesh has been simplified using a decimation method that does not guarantee Delaunay simplifications, which becomes even more difficult for volumetric meshes.

Most of the research in this category has been centered around balancing the trade-off between speed and memory usage, because many of the computationally efficient and accurate projection algorithms require large amounts of memory which makes them unsuitable for interactive rendering of datasets of the order of a million elements. Although recent attempts like zSweep by Farias et al. [3] have been shown to be memory efficient, and still remain exact, projection-based methods are still far from being real-time volume rendering systems without dedicated hardware support [14, 20].

## 2.2. Slicing-based Methods

There have been only a few publications in this category, probably because it is just another approximation of the projection-based methods: the projections of the faces of a polyhedral element are approximated by a polygon representing its intersection with a sweep plane. However, Yagel et al. [25] have shown that this method can render visually comparable images faster by exploiting the conventional polygon rendering hardware without having to explicitly store any kind of vertex or face adjacency information unlike a majority of the projection-based methods.

Westermann [21] recently proposed an interactive volume-rendering paradigm based upon slicing tetrahedra. However, an implementation of this paradigm requires 2D texture support in rendering hardware, which is the main source of speed up.

Referring back to Yagel et al's algorithm [25], there are some redundancies in the algorithm, which can be eliminated or optimized to achieve even higher interactivity. For example, the data structures to hold an active cell list or an active edge list can be completely eliminated as shown in our method (please see section 3.1). A cross-reference between these two data structures has to be performed for every sweep, which can be avoided. A computation of intersection polygons after all the edges for current *z*-value (active edges) have been intersected is an overhead, which can be avoided. Further, our method does not require depth sorting of edges at all. In addition to these

improvements, we introduce two new features to the incremental slicing tetrahedra algorithm:

- A hierarchical *z*-region data structure which replaces the active cell list and active edge list data structures; and
- A progressive refinement scheme for 2D slices holding the intersection triangles. This scheme is based upon decimation schemes for 2-manifold triangle surface meshes, and provides another method for level-of-detail (LoD) prototyping of datasets during interactive exploration.

# 3 THE ACCELERATED ALGORITHM

## 3.1 Hierarchical *Z*-Region Paradigm

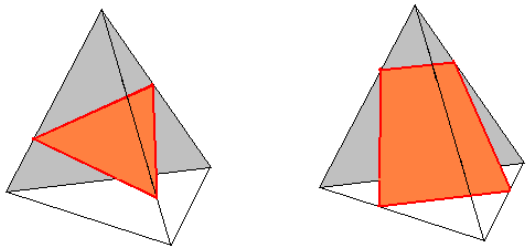
We introduce a method to improve the depth information data structures and progressive refinement capabilities of the algorithm by Yagel et al. [25]. This paradigm is similar to the bucket-sorting method employed by Yagel. However, it allows us to eliminate the following data structures and computational overheads:

- No edge sorting is required in our method. Instead, each tetrahedral cell is assigned to one or two *z*-regions depending upon the maximum and minimum *z*-value of its vertices. This can be done while reading the dataset, with no additional computational cycles required.
- The *z*-region data structure inherently stores the information of all the cells that *are being sliced* by the current sweep plane; hence no additional data structure such as an active cell list or an active edge list is required.
- Since active cell and edge lists are no more required, the cross-reference overheads (when computing the intersection polygons once all the active edges have been intersected by the current sweep plane as required in [25]) are eliminated.

The hierarchical *z*-region method is simply a segregation of the finite volume of interest (enclosed between two sweep planes, parallel to the viewing plane, with minimum and maximum depth values) into regions of either equal or unequal widths.

## 3.2 Intersection Polygons

We assume that any input polyhedral mesh has first been broken down to a tetrahedral mesh before rendering. In this case, an infinite plane can slice a tetrahedral element in only two kinds of geometric primitives with a finite area of intersection: a triangle, or a quadrilateral. Both cases are shown in Figure 2. We discard the cases of intersection when a plane passes through exactly one edge or exactly one vertex because the resulting intersection polygons have zero area.



2(a) 2(b)

**Figure 2:** Two cases of intersection of a tetrahedron with a sweep plane resulting in polygons of finite area. Note that the intersection quadrilateral in 2(b) can be easily broken into two triangles for rendering.

## 3.3 Nyquist Slicing

In pursuit of accuracy, we exploit the classical *Nyquist Sampling Theorem* when slicing an unstructured mesh with sweep planes parallel to the viewing plane. We make the following observation: a volumetric mesh $M$ is a discrete approximation of continuous 3D Euclidean space bounded by the mesh's outer surfaces $S_i$. In such a discrete volumetric model, the values of the (scalar/vector) attributes change only at discrete points: the vertices $v_i$ of the mutually exclusive polyhedral cells $c_i$ that span this bounded volume. Hence, we propose this hypothesis:

*Let $e_{min}$ be the length of the shortest edge in an unstructured volumetric mesh M. If M is to be sliced by a series of parallel sweep planes $P_i$ , no attribute changes would be left unsampled in the bounded volume along the direction of sweep if the minimum spacing $\Delta s$ between any two subsequent planes $P_i$ and $P_{i+1}$ is not more than and not equal to $e_{min}$. We call this optimal spacing $\Delta s_o$.*

We conducted experiments with at least three datasets in accordance of this hypothesis. A spacing value $\Delta s_o$ is expected to give images of best visual quality, which is supported by the results. We discuss the implications of this hypothesis further in the conclusions.

## 3.4 The Algorithm

Our basic improved slicing tetrahedra algorithm is given below:

1. Determine the minimum and maximum *z*-values, $z_{min}$ and $z_{max}$, for the given dataset.
2. Divide the space between $z = z_{min}$ and $z = z_{max}$ into *numRegions* sub-regions of uniform spacing

   $$\Delta s = (z_{max} - z_{min})/ numRegions.$$

   This creates a virtual *z*-region between $z = z_{min}$ and $z = z_{max}$ of *numRegions* sub-regions *subZRegion[i]* , $i \in \{0, 1, 2, \ldots, numRegions - 1\}$.

3. While reading the dataset, for each tetrahedron *T*:
   Find the maximum and minimum *z*-stretch values for *T*. Assign *T* into two sub-regions *subZRegion[i] and subZRegion[j], i can be equal to j,* depending upon its minimum and maximum *z*-stretch values. Flag it as UNRENDERED.

4. $zP = z_{min}$.

5. **for** $i = 0$ to *numSubRegions-1* **do**
   a. Intersect current sweep plane with all the elements *T* which were encountered for the first time for a *subZRegion[j], j* $\leq i$.
   b. Determine the case of intersection (Figure 2). Send the triangle(s) of intersection to the rendering pipeline. If *T* is encountered twice for the current *subZRegion[i],* flag it as RENDERED.
   *c.* $zP = zP + \Delta s/2$

It is to be noted that for this algorithm, the run-time storage complexity for a polyhedral mesh of n elements is:

Polyhedral Mesh: *O (n); z*-region data structure: *O (2*n).*

Hence, we get rid of the extra data structures for keeping active cell/edge list, and of the updating overheads thereof at the run-time.

# 4 RESULTS

Tables 1 through 5 summarize the results obtained from sample runs of the algorithm on three datasets. All execution times are for an SGI R10000 194MHz with 2048 MB RAM, running Irix 6.5. Datasets: Spx (courtesy Peter Williams, Lawrence Livermoore National Laboratory), Delta wing: (courtesy NASA Ames Research Center), and Blunt fin: (courtesy C. M. Hung and P. G. Buning, 1990, NASA).

| Dataset | Tetrahedra | Time(s) | Avg slices/ tet |
|---|---|---|---|
| Spx | 12,936 | 0.09 | 2.34 |
| Blunt fin | 187,395 | 1.00 | 2.00 |
| Delta wing | 1,005,675 | 1.43 | 1.11 |

**Table 1:** Summary of results from an implementation of our algorithm on three datasets for *numRegions* = 32.

| Dataset | Tetrahedra | Time(s) | Avg slices/ tet |
|---|---|---|---|
| Spx | 12,936 | 0.15 | 3.12 |
| Blunt fin | 187,395 | 1.42 | 2.58 |
| Delta wing | 1,005,675 | 1.74 | 1.18 |

**Table 2:** Summary of results from an implementation of our algorithm on three datasets for *numRegions* = 50.

| Dataset | Tetrahedra | Time(s) | Avg slices/ tet |
|---|---|---|---|
| Spx | 12,936 | 0.17 | 4.04 |
| Blunt fin | 187,395 | 1.85 | 3.29 |
| Delta wing | 1,005,675 | 2.04 | 1.26 |

**Table 3:** Summary of results from an implementation of our algorithm on three datasets for *numRegions* = 72.

| Dataset | Tetrahedra | Time(s) | Avg slices/ tet |
|---|---|---|---|
| Spx | 12,936 | 0.24 | 5.25 |
| Blunt fin | 187,395 | 2.69 | 4.19 |
| Delta wing | 1,005,675 | 2.53 | 1.36 |

**Table 4:** Summary of results from an implementation of our algorithm on three datasets for *numRegions* = 100.

# 5 OPTIMIZATION

*Progressive Refinement of Slices:* Our algorithm can be optimized for two levels of progressive refinement: progressive increase in the number of sub-regions, and a progressive refinement of the 2D slices of intersection triangles employing one of the surface simplification algorithms [4, 5, 7, 8, 11, 12, 15].

| *numRegions* | Time(s) | Avg slices/ tet |
|---|---|---|
| 32 | 0.09 | 2.34 |
| 50 | 0.15 | 3.12 |
| 72 | 0.17 | 4.04 |
| 100 | 0.24 | 5.25 |
| 164* | 0.42 | 8.02 |

**Table 5:** Summary of results from an implementation of our algorithm on the Spx dataset (12,936 tetrahedra).
*Based upon Nyquist's Sampling Theorem (please see section 3.3).

*Adaptive Slicing:* This algorithm has been modified into one for which $\Delta s$ is computed in accordance with Nyquist's Sampling Theorem (section 3.3). It can further be modified into an adaptive slicing algorithm for which the spacing between any two subsequent planes $P_i$ and $P_{i+1}$ is not bound to be constant for all $i \in \{0, 1, 2, \ldots, numRegions-1\}$, i.e., $\Delta s$ can vary between $\Delta s_o$ and a value computed upon the user input *numRegions* (hierarchical *z*-region). Further, the *z*-region paradigm provides inherent clipping capabilities.

# 6 CONCLUSION

We presented an accelerated slicing-based algorithm for interactive volume rendering of unstructured polyhedral meshes. Our algorithm eliminates redundant data structures for maintaining depth information, and treats tetrahedral cells as atomic volume-units when slicing, contrasting the original slicing tetrahedra algorithm by Yagel et al. [25]. We showed how maximum accuracy could be achieved by employing Nyquist's Sampling Theorem when slicing a dataset. Results show that low accuracy images can be progressively refined to maximum accuracy images obtained following Nyquist Slicing, allowing flexible interactivity in a real-time volume visualization system. We introduced an additional notion for levels of detail in slicing based methods: progressive coding of the 2D polygon slices of intersection.

# ACKNOWLEDGEMENTS

Thanks to Peter Williams, Lawrence Livermore National Laboratory for the *Spx* dataset; Ricardo Farias, Mississippi State University, for tetrahedral mesh subdivision code. This work has been sponsored in part by the National Science Foundation under award no. 6066047–0121989 for the SPUR (Seismic Performance for Urban Regions) project [1, 9, 10]. SPUR Contributors: Gregory L. Fenves, Bozidar Stojadinovic (UC Berkeley), Jacobo Bielak et al. (Carnegie Mellon University), Tomasz Haupt, Purushotham Bangalore, Michael L. Stokes (Mississippi State University).

## REFERENCES


[1] Chopra, P., J. Meyer, and Michael L. Stokes. Immersive Visualization Of A Very Large Scale Seismic Model. In *Sketches and Applications of SIGGRAPH'01 (Los Angeles, California, August 2001)*, page 107. ACM SIGGRAPH, ACM Press. August 2001.

[2] Comba, J., Klosowski, J., Max, N., Mitchell, J., Silva, C., and Peter Williams. Fast Polyhedral Cell Sorting For Interactive Rendering Of Unstructured Grids. In *Proceedings of EUROGRAPHICS'99, Computer Graphics Forum (1999)*, pages 369-376. Eurographics Association. 1999.

[3] Farias, R., Mitchell, J., and C. Silva. Zsweep: An Efficient And Exact Projection Algorithm For Unstructured Volume Rendering. In *ACM/IEEE Volume Visualization Symposium'00 (2000)*, pages 91-99. 2000.

[4] Garland, M. Multi-resolution modeling: Survey & Future Opportunities. In *EUROGRAPHICS'99, State of the Art Report (STAR) (Aire-la-Ville, CH, 1999)*, pages 111-131. Eurographics Association. 1999.

[5] Garland, M., and P. Heckbert. Surface Simplification Using Quadric Error Metrics. In *Proceedings of SIGGRAPH'97*, pages 115-122. ACM SIGGRAPH, ACM Press. 1997.

[6] Giertsen, C. Volume Visualization Of Sparse Irregular Meshes. In *IEEE Computer Graphics and Applications, 12(2) (March 1992)*, pages 40-48. 1992.

[7] Hoppe, Hugues. Progressive Meshes. In *Proceedings of SIGGRAPH'96 (New Orleans, Louisiana, August 1996)*, pages 99-108. ACM SIGGRAPH, ACM Press. August 1996.

[8] Isenburg, Martin, and Jack Snoeyink. Face Fixer: Compressing Polygon Meshes With Properties. In *Proceedings of SIGGRAPH'00 (New Orleans, Louisiana, July 23-28, 2000)*, pages 263-270. ACM SIGGRAPH, ACM Press. 2000.

[9] Meyer, J., and Prashant Chopra. Building Shaker: Earthquake Simulation In A CAVE™. *Work in progress, IEEE Visualization'01 (San Diego, California, October 2001)*, Abstract, page 3. 2001.

[10] Meyer, J., and Prashant Chopra. Strategies For Rendering Large-Scale Tetrahedral Meshes For Earthquake Simulation. *SIAM/GD'01 (Sacramento, CA, November 2001)*, Abstract, page 30. 2001.

[11] Popovic, J., and Hoppe, H. Progressive Simplicial Complexes. In *Proceedings of SIGGRAPH'97*, pages 217-224. ACM SIGGRAPH, ACM Press. 1997.

[12] Renze, K. J., and J. H. Oliver. Generalized Unstructured Decimation. *IEEE Computer Graphics and Applications 16 (6)*, pages 24-32. 1996.

[13] Rezk-Salama, C., Engel, K., Bauer, M., Greiner, G., and T. Ertl. Interactive Volume Rendering On Standard PC Graphics Hardware Using Multi-textures And Multi-stage Rasterization. In *SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware (2000)*, pages 109-119. 2000.

[14] Roettger, S., Kraus, M., and T. Ertl. Hardware Assisted Volume And Isosurface Rendering Based On Cell Projection. In *Proceedings of IEEE VISUALIZATION'00*, pages 109-116. 2000.

[15] Schroeder, William J., Jonathan A. Zarge, and William E Lorensen. Decimation Of Triangle Meshes. In *Computer Graphics* 26(2), pages 65-70. 1992.

[16] Silva, C., and J. Mitchell. The Lazy Sweep Ray Casting Algorithm For Rendering Of Irregular Grids. In *Transactions on Visualization And Computer Graphics (April – June 1997), 4(2)*, pages 142-157. 1997.

[17] Silva, C., Mitchell, J., and A. Kaufman. Fast Rendering Of Irregular Grids. In *Proceedings ACM Symposium on Volume Visualization'96*, pages 15-23. 1996.

[18] Silva, C., Mitchell, J., and Peter Williams. An Exact Time Visibility Ordering Algorithm For Polyhedral Cell Complexes. In *Proceedings ACM Symposium on Volume Visualization'96*, pages 87-94. 1998.

[19] Shirley, P., and A. Tuchman. A Polygonal Approximation To Direct Scalar Volume Rendering. In *Proceedings of SIGGRAPH'90, ACM Computer Graphics, 24 (5)*, pages 63-70. ACM SIGGRAPH, ACM Press. 1990.

[20] Stein, C., Becker, B., and N. Max. Sorting And Hardware Assisted Rendering For Volume Visualization. *In ACM Symposium on Volume Visualization'94*, pages 83-90. 1994.

[21] Westermann, R. The Rendering Of Unstructured Grids Revisited. In *EUROGRAPHICS/IEEE TCVG Symposium on Visualization'01*. 2001.

[22] Williams, P. Visibility Ordering Meshed Polyhedra. In *ACM Transactions on Graphics, 11(2)*, pages 102-126. 1992.

[23] Williams, P., Max, N., and C. Stein. A High Accuracy Volume Renderer For Unstructured Data. In *IEEE Transactions On Visualization And Computer Graphics*, 4(1), pages 37-54. 1998.

[24] Wittenbrink, C. Cellfast: Interactive Unstructured Volume Rendering. In *IEEE Visualization'99, Late Breaking Hot Topics*, pages 21-24. 1999.

[25] Yagel, R., Reed, D., Law, A., Shih, P., and N. Shareef. Hardware Assisted Volume Rendering Of Unstructured Grids By Incremental Slicing. In *ACM Symposium on Volume Visualization'96*, pages 55-63. 1996.

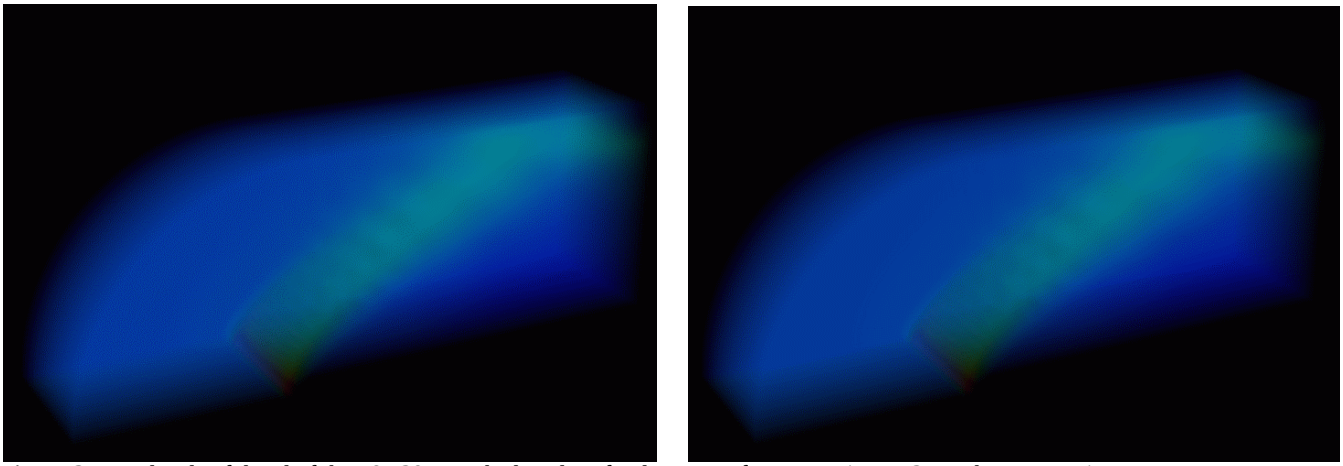


**Figure 3:** Two levels of detail of the 187,395 tetrahedra Blunt fin dataset. Left: *numRegions* = 32; right: *numRegions* = 72.

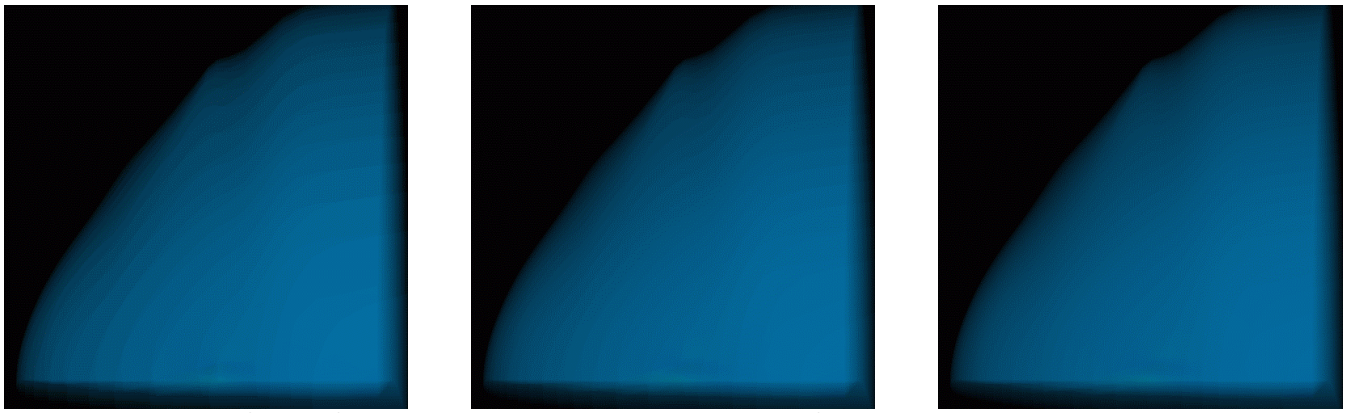


**Figure 4:** Three levels of detail of the 1,005,675 tetrahedra Delta wing dataset. Left: *numRegions* = 32; middle: *numRegions* = 50; right: *numRegions* = 72.

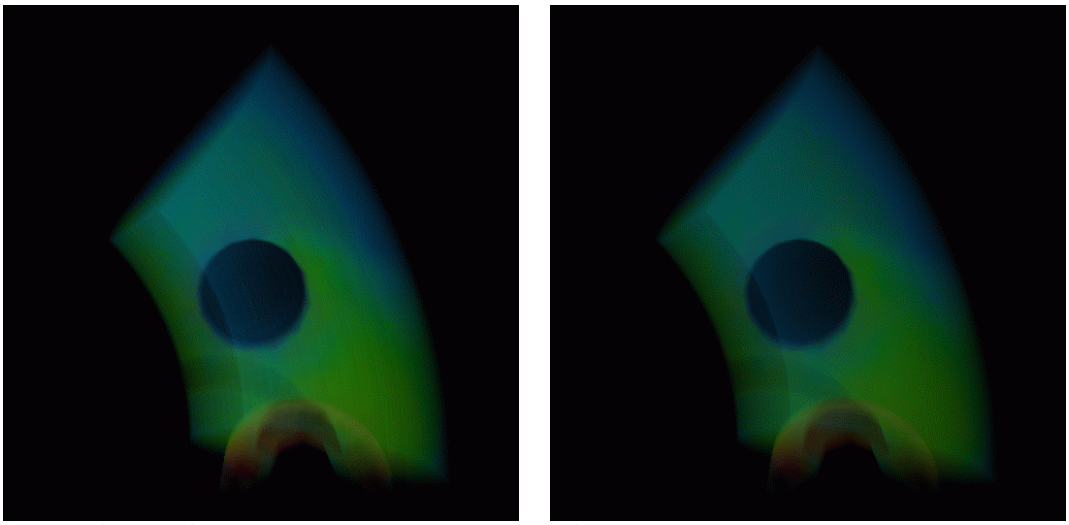


**Figure 5:** Two levels of detail of the 12,936 tetrahedra Spx dataset. Left: *numRegions* = 32; right: *numRegions* = 164, computed based upon Nyquist's Sampling Theorem.