Direct Volume Rendering of Unstructured Grids in a PC based VR Environment

Paul Benölken
Fraunhofer IWU
Reichenhainer Straße 88
09126 Chemnitz

Holger Graf
Fraunhofer IGD
Fraunhofer Straße 5
64283 Darmstadt

paul.benoelken@t-online.de holger.graf@igd.fhg.de

ABSTRACT

In this paper we present our solution for fast, direct volume rendering of unstructured grids on standard PC workstations. We describe our modification of the incremental slicing approach for achieving high performance as well as the application of geometry-compression methods for minimizing the memory requirements. Furthermore, we show our implementation for the interactive modification of transfer functions (classification) in a virtual reality environment by using 3D interaction widgets. Finally we present and discuss the results, we achieved with our application in a VR environment.

Keywords

Direct Volume Rendering, Unstructured Grids, Geometry Compression, Virtual Reality

1. INTRODUCTION

Direct volume rendering of unstructured 3D scalar fields has been the subject of a number of research activities and publications over the past decade. Many efforts are directed towards improving the rendering performance by parallelization or efficiently using graphics hardware. Although great advances have been reported, many techniques are either restricted to handling fixed cell types and optical models or require massive parallel computing facilities for achieving sufficient frame rates. However, due to the increasing power and wide availability of consumer graphic boards, different volume rendering algorithms have been implemented on standard PC hardware. Nevertheless, the existing solutions still do not fully match the requirements of interactive applications like those required in virtual reality environments.

Beside interactive visualization, the classification of the scalar field by generating transfer functions plays an important role for the volume rendering process.

The Journal of WSCG, Vol.13, ISSN 1213-6964
WSCG'2005, January 31-February 4, 2005
Plzen, Czech Republic.

A favourable transfer function exposes the important data structures and hides the non relevant parts of the dataset. Frequently used are opacity functions assigning colors and opacities based on scalar values. Many datasets contain complex structures with overlapping scalar values. Here the application of opacity functions frequently fails. Different techniques have been developed within the last years for visualizing more details by applying multidimensional transfer functions as well as for providing more convenient user interfaces. However the classification of unstructured volume data is not addressed.

2. Related Work

Besides different optimizations and parallelization for improving software solutions like ray castings [Lev90] or splatting techniques [Wes90], several research activities addressed the exploitation of the graphics hardware as in [WE97]. Cell projections like the projected tetrahedra (PT) method [ST90] have become one of the most popular methods for hardware accelerated volume rendering.

Different approaches have been considered for improving the image quality [SBM94], [GRS+02], [RE02] as well as for achieving higher frame rates by accelerating the required cell sorting [Wil92], [SMW98], [CKM+99]. Other hardware accelerated methods like [WKE02] avoid the sorting process by applying an emissive optical model and exploiting the vertex programming facilities of current graphic boards for implementing a view-independent cell projection. Nevertheless, besides tetrahedral elements, unstructured grids resulting from finite

element computations frequently contain different polyhedral elements (e.g. prisms, hexahedra etc.). Hence applying the PT algorithm for such grid types requires a tetrahedral decomposition of each cell, which in turn increases the amount of cells to be processed and thus reduces the overall performance. Hence, the recently presented approaches extend the original PT algorithm for processing different kind of polyhedra. Roettger and Ertl [RE03] analyse the maximum performance of PC graphic accelerators like the NVIDIA GeForce series of cards and apply an emissive optical model for achieving fast renderings.

Although several accelerations and improvements were successfully applied as in [Mor04], the performance of these algorithms is still below the interactive frame rates achievable for regular grids. Hence alternative solutions presented by [Wes01] and [WE01] implement different strategies for re-sampling unstructured volume data onto cartesian grids in a preprocessing step. Interactive performance is gained by employing texture based volume rendering on the re-sampled dataset. Nevertheless, these solutions are constrained by the available texture memory on the graphic boards. Larger datasets or higher resolutions are processed by splitting the cartesian grids into multiple blocks (bricks). However, due to the limited bandwidth, data transfer from the CPU main memory to the GPU texture memory results in performance losses.

Another solution, proposed by Max et al. [MWSC03] implements cell projections (including cell sorting) and slicing methods. Both methods are parallelized for improving the performance and supplemented with anti-aliasing techniques for handling small cells.

Further research activities are focused on the definition and manipulations of transfer functions. Typical graphical user interfaces are restricted to opacity functions. Modification of the opacity functions are done by moving control points of the 2D graph. He et al. [HHKP96] apply genetic algorithms for finding good transfer functions. In an iterative process the user selects the desired mapping from an automatically generated population of small thumbnail renderings. Marks et al. [MAB+97] propose a Design Gallery as a visual interface to the space of possible transfer functions. An image difference metric is used for arranging different renderings, from which the user selects the most appealing one. A more data-centric approach was presented by Kindlmann et al. [KD98]. Their semi-automatic method exploits the relationship between data values and their first and second derivatives in direction of the gradient, for detecting material boundaries. Further approaches define the transfer functions based on the principal curvatures reconstructed from the input data [HKG00] or apply dynamic programming for a template based adjustment of transfer functions [RSHSG00]. Kniss et al. [KKC01] use a direct manipulation widget in a desktop environment for specifying multidimensional transfer functions based on data value, gradient magnitude, and the second directional derivative. Botha and Post [BP02] presented a fast method based on slice-based preview for finding an appropriate transfer function.

3. Interactive Direct Volume Rendering

As shown by Chopra and Meyer [CM02] fast visualizations are feasible by using incremental slicing. Performance improvements have been achieved by avoiding redundant computations and replacing the active cell and active edge list in the original algorithm of Yagel et al. [YRLN96] with an active region list. Inspired by the mentioned slicing method of Max et al., our solution finally skips the computation of the active region list which originally was required for each change of the view direction.

Two different modes have been implemented for computing the set of volume slices. In interactive mode for each of the three X, Y, Z axes a set of slices is computed in a preprocessing step with a predefined slice distance. Since artifacts might become visible from intermediate view points, we additionally employ an immediate mode which computes a set of slices from the current view direction when the rotary motion stops.

3.1 Computation of Volume Slices

Given an unstructured grid with N_n node coordinates $n_i \in \{n_0 \ldots n_{Nn-1}\}$ and N_c grid elements $c_i \in \{c_0 \ldots c_{Nc-1}\}$ the geometry and topology of each cell c_i is defined by a list of cell nodes $\{n_{ci}\}$, where each cell node refers to a specific node coordinate n_i.
Starting with a given distance Δ_z of slicing planes and a view direction $\mathbf{x}_{view}$ the number of required slices N_{slices} as well as the start and endpoints, z_{start} and z_{end} in object space are calculated. Afterwards, the slice computation proceeds in the following way:

1. for each node n_i

   compute and store the node distance d_{ni} to the viewplane $\mathbf{p} = z_{end}$.
2. for each cell c_i

   for each cell node n_{ci}

   a.) compute the slice index $s_i = d_{ni} / (z_{end} - z_{start}) N_{slices}$

   b.) determine the min/max slice index s_{min} and s_{max}

   c.) for each slice index $s_i \in \{s_{min} \ldots s_{max}\}$ compute and store the slice polygons f_{si}

In step 2c we applied a variant of the marching cube method for incrementally slicing the volumetric cells.

Each cell node is classified as "0" or "1" depending on the sign of the node distance d_{nci}. Since in step 1 the viewplane distance d_{ni} was calculated for each grid node, the required computations of the specific slice-point distances d_{nci} for each cell node n_{ci} simplify to

$$d_{nci} = d_{ni} + \Delta_z (s_i + 1).$$

The generated bit sequence is used as an index into a lookup table, which identifies the cell edge for the vertex interpolation of the slice polygon. Different lookup tables are maintained for identifying the specific (e.g. tetra or hexa) polyhedron intersections. Hence a decomposition into tetrahedral elements is not anymore required. Finally each polygon set of a specific slice is stored into an indexed list PL_N as illustrated in figure 1. Therefore the computation of active region lists is skipped.

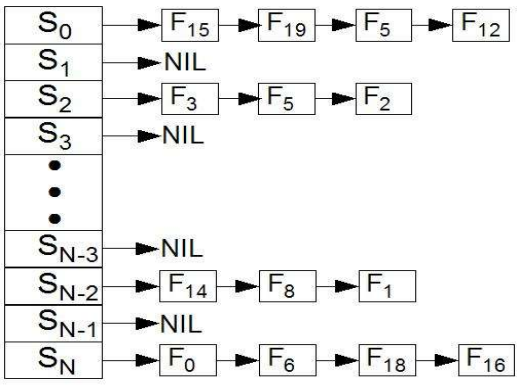

Figure 1: Structure of the indexed polygon list for storing the slice polygons.

Besides the polygon data, the indexed list structure contains empty slices, which are indicated by a NIL-pointer, so that intermediate slices can be inserted or skipped.

3.2 Semi Adaptive Slicing

The computation of volume slices allows fast direct volume renderings of unstructured grids. However the optimal choice of the slice distance Δ_z is still a problem of such slicing techniques. Small details are missed, if the slice distance was chosen too large, whereas small values of Δ_z result into over sampling. According to Chopra [CM02], no details are missed, if the distance Δ_z is chosen to be neither greater nor equal to the smallest edge e_{min} of the dataset. However, this approach results into extreme large amounts of polygons, which are hardly processed in real-time. Hence Max et al. propose the usage of splatting methods or cell projections for all those cells, which have been missed by the slicing operations.

An alternative approach is the semi adaptive computation of volume slices. For this purpose, a set of equidistant slices is created using our previously described method. In this first step, all elements which have been successfully sliced, are marked as processed. Afterwards the number of slice regions is doubled and the described slicing procedure is applied to the remaining unprocessed cells with the adapted slice distance Δ_{zi}. This step is repeated until either all elements or a predefined percentage has been sliced. Within each iteration step, a separate indexed polygon list PL_{Ni} is built. Afterwards the polygon list of the previous step PL_{Ni-1} is completed with the new generated one. This operation is accomplished by simple pointer operations by exploiting the previously mentioned indexed list structure.

This approach enables the generation of slice stacks with varying resolutions in z-direction. Additional slices may be inserted or skipped in a background process. Hence different levels of detail could be dynamically generated according to the user defined scaling and zooming operations.

3.3 Interactive Classification

An efficient visual analysis of the volume dataset requires an interactive classification and hence direct manipulation of the transfer function. Beside an immediate update of the scalar field visualization, the representation of the transfer functions is another important factor for gaining insight into the dataset and its value distribution.

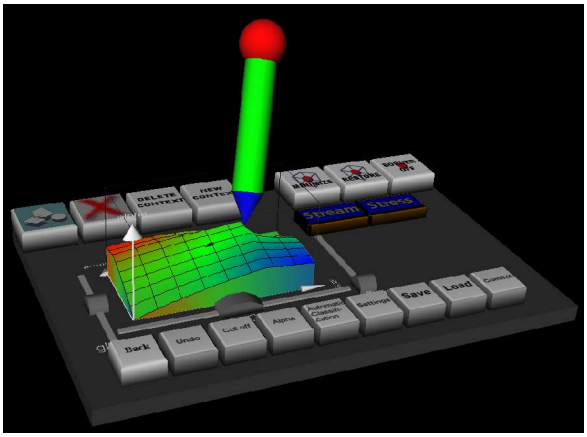

Figure 2: The interaction panel with the 3D classification widget and sliders for interactive manipulation of the transfer function.

Usually multi-dimensional transfer functions offer a higher flexibility for detecting and exposing characteristic structures. Nevertheless the creation of a meaningful representation for more than three independent parameters is not trivial. Therefore our implementation was restricted on 2D transfer-functions which appeared to be a good compromise

between the flexibility of multi-dimensional transfer-functions and one-dimensional opacity assignments.

Since one of our aims was the integration into a VR environment, a 3D representation of the transfer function appeared to be a natural extension for 3D interactions and visualizations.

Figure 2 shows the interaction panel with the 3D classification widget and the sliders we implemented for visualizing and manipulating 2D transfer functions. The transfer function is drawn as a height field by mapping scalar values and gradient magnitudes onto the tablet plane, whereas the opacity (alpha) values define the height of the palette.

Direct manipulations are supported by using a 6DOF interaction pen for changing individual alpha values. Since this interaction is strongly influenced by the assignment of alpha and height values, we examined different mappings. Tests with linear mappings showed significant correlations for lower alpha values, whereas changes of higher alpha values are reflected by minor changes in the scalar field visualization. A better reflection of small alpha values was achieved by employing a logarithmic mapping with basis b using the following formula:

$$h = \ln((b-1)\,\alpha + 1) / \ln(b)$$

Where h is the computed height value and α indicates the corresponding transparency. The inverse function is used for obtaining the transparency from the user defined height value, which is:

$$\alpha = (b^h-1) / (b-1).$$

Indirect manipulations of the transfer function are accomplished by moving one of the 3D sliders on the interaction panel. Different operations have been implemented for changing the global transparency as well as for highlighting edges and material boundaries by adjusting gradient weights. Another operation supports the selection of specific regions by specifying mean and range of the gradient-value domain. Further interaction buttons have been integrated, for storing the current transfer function, for loading the settings from a previous session, as well as for performing undo and redo operations on the current transfer function. A list of transfer functions is used for tracking the changes within a session.
The Classification is accomplished by storing the transfer functions as 2D texture maps. The scalar values and gradient values are transferred to the implemented fragment program which applies texture mapping by taking the scalar values and gradient magnitudes as texture coordinates for determining the fragment color and finally performs the lighting calculations.

4. Our Solution

This chapter describes our realization of an interactive direct volume rendering system. It gives an overview of the architecture and some optimization strategies which are used in order to optimize the data handling on common PCs.

4.1 Architectural View

Figure 3 shows the client-server architecture of our application. Memory and CPU intensive tasks like the slicing of the grid and the computation of gradient values are performed by the server.

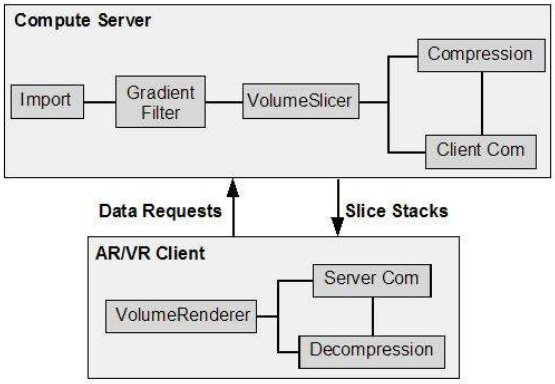

Figure 3. Client-Server architecture for direct volume rendering

The rendering client is in charge of requesting the processed data from the server and rendering the generated stack of polygon slices (stack processing) as well as for performing per fragment operations as lightning or texture reads using its Graphics Processing Unit or GPU. Furthermore, the rendering client handles the user interactions like changes in the view direction and manipulation of the transfer function.

An optional compression and decompression of the slice stacks can be applied for limiting the memory requirements and obtaining an efficient usage of the available bandwidth.

4.2 Geometry Compression

One of the requirements for the above mentioned method to interactively render unstructured grids is the storage of the computed polygon sets. Depending on the underlying grid resolution and the selected slice distance, the required storage space might easily exceed the capacities of common workstations.
Hence for limiting the memory requirements of our method, we integrated vector quantization and scalar normalization operations. In our implementation we used 16 Bit shorts for storing vertex coordinates and 8 Bit unsigned char values for each normal component and scalar values.

Thus the required memory was reduced by a factor of 2 compared to the original floating point size. Since the coding of the vertex positions consists of a translation and a scaling operation, the subsequent decoding is easily accomplished by the graphics hardware.

As already pointed out by Deering [Dee95], the resulting accuracy after decoding the compressed data, is usually sufficient for display purposes.

An additional reduction of the data size is achieved by transferring the individual slice polygons into indexed face sets, which are efficiently displayed using OpenGL vertex arrays. With this elimination of redundant vertex information and the mentioned quantization operation we achieved a reduction of the polygon slices by 25% of the original data size.

Further reductions have been achieved by integrating a compression scheme, which we successfully applied for arbitrary polygon sets from CAD models [BS02]. The algorithm encodes a polygonal mesh into a byte sequence which is finally compressed using Huffman encodings. We applied this method to the computed polygon slices and achieved a reduction of below 10 percent of the original size, on different slice sets.

5. Results

Our implementation was tested in our VR environment with different datasets and PC workstations. Figure 4 shows the interactive direct volume rendering of the space shuttle flow field in our VR environment.

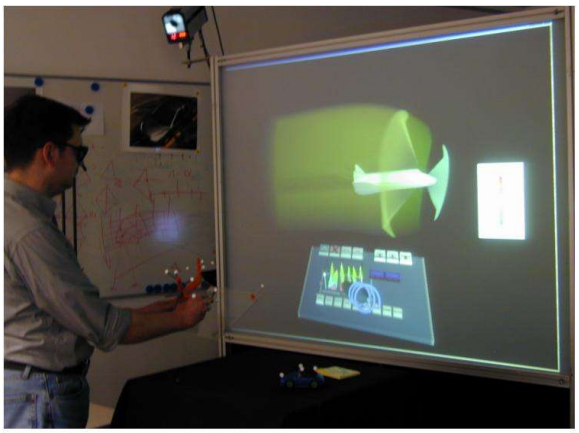

Figure 4: Interactive direct volume rendering the space shuttle flow field in a VR environment.

The classification of the volume data and hence the manipulation of the underlying transfer function is done by directly changing the 3D classification widget with the 6DOF interaction pen or by using the 3D menu elements on the interaction panel. The position and orientation of the dataset is controlled by the user by placing the tracked artifact object with its tracking sensors in the VR environment.

The evaluation of our method was done using different datasets with different sizes and cell elements. The shuttle dataset shown in figure 4 consists of 226800 nodes and 215512 hexaedron-elements. Figure 5 shows the result from a simulation of a polymere injection of a single nozzle, which was displayed using our direct volume rendering method. .

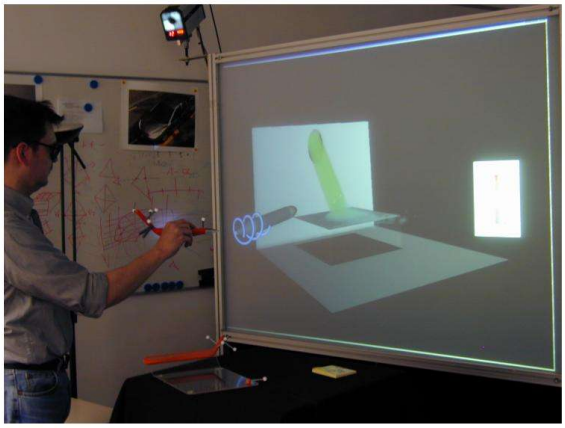

Figure 5: Direct volume rendering of a polymer injection from a single nozzle

The datasets consists of 87754 nodes, 223936 tetra elements and 26900 hexahedrons. Figure 6 shows the result of our direct volume rendering method with the bluntfin dataset. The dataset consists of 40960 nodes and 37479 hexaedron elements.

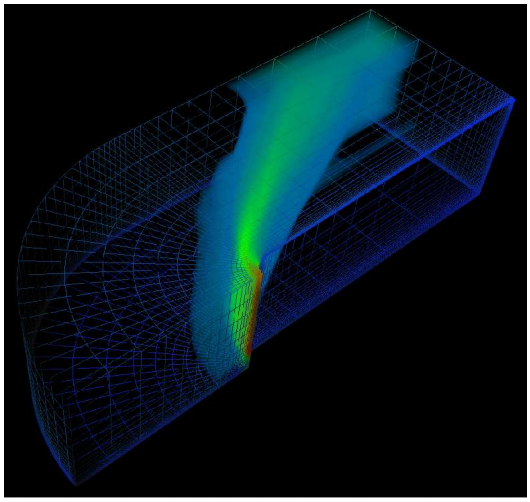

Figure 6: Direct volume rendering of the bluntfin dataset.

Some benchmarks have been driven in order to validate our approach. Figure 7 shows the timings of the slicing and quantization operation, which we achieved for the used datasets with different slice numbers.

The measurements were performed on a mobile PC equipped with a 3,06 GHz Intel CPU, 512MB main memory and a Windows XP operating system.

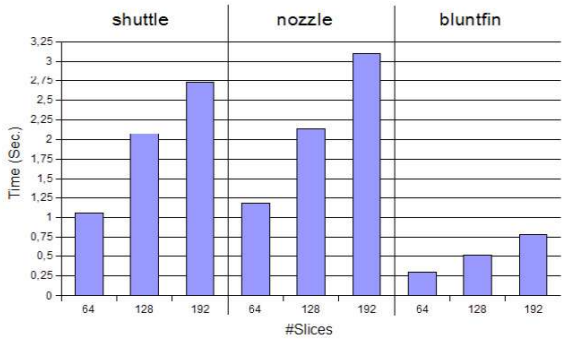

Figure 7: Timings for slicing the datasets with different slice numbers.

The performance of the slicing and quantization operation is strongly related to the size of the dataset and the number volume slices, as shown in the diagram below. The computation of detailed slice sets is done very quickly for small datasets whereas the calculation for large datasets requires a delay of up to three seconds. However, after the slice computation is completed, the display and blending of the slice stacks is performed by the graphics hardware.

Figure 8 summarizes the framerates we achieved for rendering and lighting the mentioned datasets using axis aligned slices with different slice distances and hence polygon numbers. The measurements have been carried out on a windows PC equipped with a 2,8 GHz Intel CPU, 1GB main memory and an NVIDIA GeForce FX 5900 graphics board with 128MB video memory.

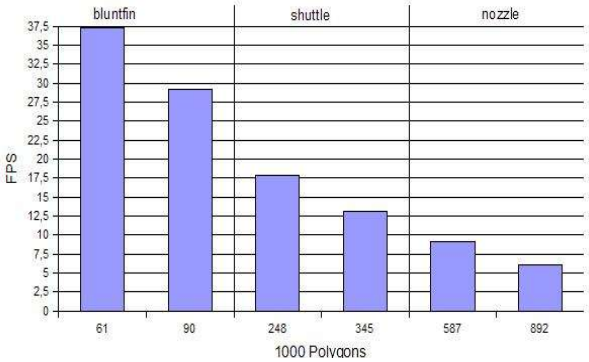

Figure 8: Framerates for rendering and lighting axis aligned slices with different polygon numbers using a view port of 1024 × 768.

The diagram shows the strong dependency of the achieved display performance from the number of polygons which have to be processed by the graphics hardware. As expected, small polygon sets are displayed with high framerates, wheras the display time slows down with the increasing number of polygons such that the limits of these graphicscards will be at approx. 1 million polygons.

Hence the performance of the presented approach is predominantly defined by the rastering and blending capabilities of the graphics hardware.

6. Conclusion

Our solution allows fast direct volume renderings of unstructured grids on a standard PC platform. The slicing algorithm quickly generates polygon slices for small datasets. Larger datasets can be processed by generating axis aligned slice sets in a preprocessing step. Alternatively slice sets with a reduced z-resolution might be used for previewing during the user interaction, while additional slices are computed in a background process.

As shown before the methods processes unstructured grids with mixed tetra- and hexaedral elements and is easily extended for handling further elements. A tetrahedral decomposition is not anymore required so that the amount of cell elements to be processed is strongly reduced which in turn leads to faster computations. The implemented compression methods additionally reduce the memory requirements so that the available bandwidth is more efficiently used.

Moreover, the method allows an improved balancing of the workload between GPU and CPU and hence an improved overall performance. In contrast to this the load of view-independent tetra-projections is predominantly on the GPU. Furthermore the semi-adaptive slicing approach offers an efficient alternative for displaying small grid cells without generating inadequate amounts of polygons.
Finally the created modules support the interactive visualization and classification of volume data in a VR environment. The developed 3D interaction facilities enable a direct and convenient manipulation of the transfer function as well as an intuitive navigation in virtual environments.

7. Outlook

The presented work currently allows the presentation of static volume data. Further work should concentrate on the inclusion of dynamic and time variant simulation data. With some modifications this method can be used for colourised and opacity animated renderings of scalar fields, which values change over time. Using a time invariant geometry the interpolation weights resulting from the slicing operations as well as the indices of affected grid nodes can be stored in addition to the vertex data. During the following time steps only the indexed nodes and the interpolation weights have to be transferred to the GPU which eventually performs the interpolation via a vertex program.

REFERENCES

[BS02] P. Benölken and A. Stork. Geometry compression for collaborative CAD applications. In Ralph H. u.a. Stelzer, editor, *CAD 2002*. Proceedings : Corporate Engineering Research, pages 121-129.

[BP02] C.P. Botha and F.H. Post. New technique for transfer function speci-fication in direct volume rendering using real-time visual feedback. In Proc. of the SPIE Int. Symposium on Medical Imaging, volume 4681, 2002.

[CKM+99] J. Comba, J. T. Klosowski, N.L. Max, J.S.B. Mitchell, C.T. Silva, and P.L. Williams. Fast polyhedral cell sorting interactive rendering of unstructured grids. In Computer Graphics Forum (Proc. Eurographics '99), volume 18, pages 369–376, 1999.

[CM02] Prashant Chopra and Joerg Meyer. Incremental slicing revisited: Accelerated volume rendering of unstructured meshes. In Proceedings of IASTED Visualization, Imaging, and Image Processing 2002, pages 533–538, Sept. 9-12 2002.

[Dee95] Deering, M 1995. Geometry Compression Proceedings SIGGRAPH, 1995.

[GRS+02] Stefan Guthe, Stefan Roettger, Andreas Schieber, Wolfgang Straßer, and Thomas Ertl. High-quality unstructured volume rendering on the pc platform. In ACM Siggraph/Eurographics Hardware Workshop, 2002.

[HHKP96] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In Proceedings IEEE Visualiztaion, pages 227–234. IEEE, 1996.

[HKG00] J. Hladuvka, A. König, and E. Gröller. Curvature-based transfer functions for direct volume rendering. In Spring Conference on Computer Graphics (SCCG 2000), pages 58–65, 2000.

[KD98] G. Kindlmann and J.W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In IEEE Symposium On Volume Visualization, pages 79–86. IEEE, 1998.

[KKC01] J. Kniss, G. Kindlmann, and C.Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In Proceedings IEEE Visualization 2001, pages 255–262. IEEE, 2001.

[Lev90] M. Levoy. E±cient ray tracing of volume data. *ACM Transactions on Graphics*, 9 (3): 245-261, July 1990.

[MAB+97] J. Marks, B. Andalman, P.A. Beardsley, H. Pfister, et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In ACM Computer Graphics (SIGGRAPH '97 Proceedings), pages 389–400. ACM, August 1997.

[Mor04] Kenneth Dean Moreland. Fast High Accuracy Volume Rendering. Dissertation, The University of New Mexico, 2004.

[MWSC03] Nelson Max, Peter Williams, Claudio Silva, and Richard Cook. Volume rendering for curvilinear and unstructured grids. In Computer Graphics International, 2003.

[RE02] S. Roettger and T. Ertl. A two-step approach for interactive preintegrated volume rendering of unstructured grids. In Proceedings of IEEE Volume Visualization and Graphics Symposium 2002, pages 23–28, October 2002.

[RE03] S. Roettger and T. Ertl. Cell projection of convex polyhedra. In Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics, page 103108. ACM, 2003.

[RSHSG00] C. Rezk-Salama, P. Hastreiter, J. Scherer, and G. Greiner. Automatic adjustment of transfer functions for 3d volume visualization. In Proc. Vision, Modelling, and Visualization (VMV), pages 357–364, 2000.

[SBM94] C. M. Stein, B. G. Becker, and N. L. Max. Sorting and hardware assisted rendering rendering for volume visualization. In Symposium on Volume Visualization, pages 83–89, 1994.

[SMW98] C. Silva, J. Mitchell, and P. Williams. An interactive time visibility ordering algorithm for cell complexes. In ACM / IEEE Symposium on Volume Visualization, pages 15–22, 1998.

[ST90] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In Computer Graphics (San Diego Workshop on Volume Visualization), volume 24, pages 63–70, 1990.

[WE97] R. Westermann and T. Ertl. Rendering and re-sampling unstructured volume data by polygon drawing. Technical Report 17, Universität Erlangen-Nürnberg, 1997.

[WE01] Manfred Weiler and Thomas Ertl. Hardware-software-balanced resampling for the interactive visualization of unstructured grids. In Proceedings of the conference on Visualization '01, pages 199 – 206. IEEE Computer Society, 2001.

[Wes90] L. Westover. Footprint evaluation for volume rendering. In *Computer Graphics*, volume 24, page 367-376, 1990.

[Wes01] R. Westermann. The rendering of unstructured grids revisited. In Eurographics/ IEEE Symposium on Visualization 2001, pages 65–74, 2001.

[Wil92] P. Williams. Visibility ordering meshed polyhedra. In ACM Transaction on Graphics, volume 11, pages 103–126, 1992.

[WKE02] Manfred Weiler, Martin Kraus, and Thomas Ertl. Hardware-based view-independent cell projection. In *IEEE Symposium on Volume Visualization*, pages 13 - 22, 2002.

[YRLN96] R. Yagel, D. Reead, P. Law, A. Shihh, and Shareef N. Hardware assisted volume rendering of unsructured grids by incremental slicing. In *1996 Volume Visualization Symposium*, pages 55-62. IEEE Computer Society Press, 1996.