# Interactive Visualization of Hybrid Medical Data Sets

Jörg Meyer, Steffen Gelder, Kay Kretschmer, Karsten Silkenbäumer, Hans Hagen
University of Kaiserslautern
Department of Computer Science
P. O. Box 3049
D-67653 Kaiserslautern
Germany

E-Mail: jmeyer@informatik.uni-kl.de
URL: http://davinci.informatik.uni-kl.de/~jmeyer

January 3, 1997

**Abstract**

*Visualization of large medical data sets requires advanced techniques in image processing and data reduction. Compound data sets, which consist of several types of data, necessitate special treatment for each component. A data set which consists of both volumetric data and geometric data is called a "hybrid data set".*

*The goal of our new approach is the development of an* interactive *rendering pipeline with special timing predicates, which accepts medical data sets derived from CT or MRI scanners as well as geometric data from CAD systems, and, after pre-processing and data reduction, conjoins them in a common representation. Interactive behavior is an important feature of the system, because it enables the user to manipulate and adjust the visualization straight on demand.*

**Keywords:** Scientific Visualization, Medical Imaging, Interactive Visualization, Volumetric Data, Geometric Data, Hybrid Data, Data Reduction, Volume Rendering, Surface Rendering, Hybrid Rendering, Hierarchical Rendering, Multiresolution Techniques, CSG Objects

## 1 Introduction

Advanced technologies in scientific visualization and medical imaging helped to improve image quality in diagnostics, therapy planning, and other medical applications. Recent progress in advanced hardware and software technology provides new approaches and allows for new

algorithms to be implemented. The design of a system is no longer determined by restrictions of small memory configurations and slow pixel rendering, but high-speed graphic engines can be used to visualize a data set.

With basic hardware acceleration features most graphic engines can handle small data sets only and provide just simple and limited feature extraction capabilities. Therefore there is a need for special pre-processing algorithms, which reduce the complexity of a data set without noticable loss of information before passing it to the graphic engine.

Interactive manipulation of a data set can be incorporated at each step of the rendering pipeline. During initial data acquisition, selected features of the data set can be read or omitted from the original data set, thus dividing it into a visible and a non-visible portion. Adaptive data reduction is another means for choosing a desired level of detail in an image. Transformation into a visible object, which involves coloring, application of transparency etc. - in contrast to the viewing transformation, which determines perspective and is located at the final step of the rendering pipeline - also requires interactive control. Figure 1 shows the rendering pipeline:

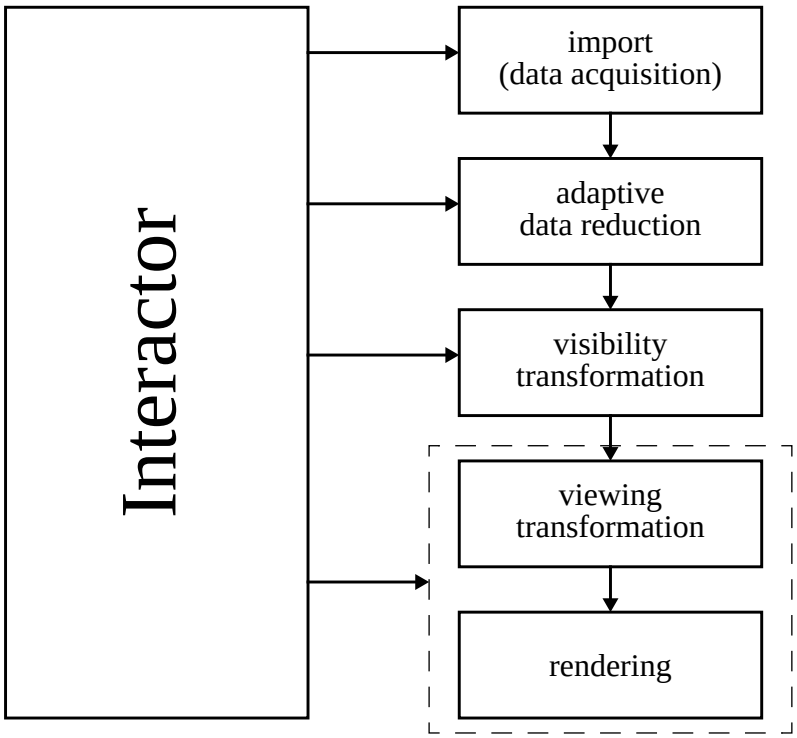


Fig. 1: rendering pipeline

The data flow indicates that there are several access points for manipulating a data set interactively. The fastest and most effective way to interact with the data set is the viewing transformation, which is a built-in feature of the graphic engine. Therefore all time-consuming processes have to be executed in the pre-processing stage in order to achieve interactive response times during visualization.

The following chapters will give an outline of the design and features of each step of the rendering pipeline.

## 2 Input data

In medical applications, it is often necessary to visualize real data, i. e. scanned data from CT or MRI imaging systems, together with artificial objects, either structuring elements, such as rulers, planes or rays, or hand-modelled items, such as prosthetic implants, surgery tools etc.

Therefore our rendering pipeline must be capable of visualizing volumetric data as well as geometric data in different representations (polygon data, surface data, CSG objects, etc.) In order to maintain precision and accuracy of data when combining different kinds of data sets, we try to avoid well-known standard methods, which usually try to convert one set into the representation of another set in order to achieve a common data structure [12].

### 2.1 Data representation

If we would use a common representation, the different stages of the pipeline could be used in the same manner for all different types of input data. This would be an advantage, but application of conversion methods always implies the introduction of inaccuracies and artifacts. For example, while positioning geometric data into a volumetric grid (figure 2), the algorithm applies a discrete scan conversion to the original object. This results in a loss of detail due to discrete sampling, and requires approximations where data does not fit exactly into the regular volumetric grid.

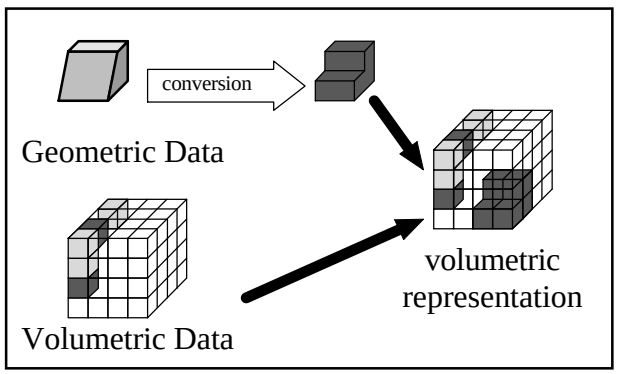


Figure 2: volumetric representation

On the other hand, if volumetric data is to be converted into a geometric representation (figure 3), a surface reconstruction algorithm, such as “Marching Cubes” [7], will be applied. This causes problems when there is no latent or real surface information inherent in the data set, or if weak contrasts disable thresholding methods from finding a valid surface [4].

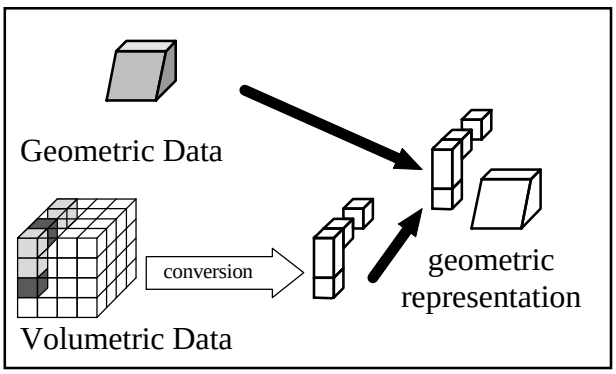


Figure 3: geometric representation

Therefore we introduce a new pipelining approach for rendering volumetric and geometric data simultaneously. The method is called *pipelined hybrid rendering*. Levoy [11] introduced a raytracing approach for hybrid rendering. For each pixel, a ray is cast into the scene, scanning both data sets at the same time. Volumetric data is sampled at discrete positions along the ray, while geometric data is interleaved whenever the ray hits a surface.

The main idea is to maintain the original representation for both data sets while passing them through all preprocessing stages of our pipeline. The same transformation and projection algorithms will be applied to the different data sets. As a side effect, but turning out to be one of the major advantages of this technique, we state the fact that all stages of the pipeline can run in parallel, thus increasing overall performance and decreasing response time for the user interface.

## 2.2 Data structures

We use different data structures for each kind of data set. As already mentioned in the previous chapter, we try to maintain the original representation throughout the pipeline. Although the data set might be reduced or approximated, it always keeps its original data structure.

### 2.2.1 Volumetric data

Volume data, as provided by CT or MRI scanners, comes as a three-dimensional array of scalar values, usually arranged in a set of sequential slices [5]. It is fed into the pipeline through a flexible import module, which allows interactive removal and selection of ROIs (*regions of interest*). This is the initial access point for the interactor.

### 2.2.2 Geometric Data

Geometric data is separated into several groups. Polygon data (triangles) and CSG objects (*constructive solid geometry,* [1]) are imported through special description files. Polygon data is represented as a list of points, triangle nodes and attributes. Similar to volumetric data, the data set can be clipped in order to select ROIs.

For CSG objects, simple primitives are combined by means of regularized Boolean set operators that are included directly in the representation. In general, a regularized operation returns the closure of the interior of its set theoretic counterpart. In some implementations, simple solids (volume primitives), such as cubes or spheres, are used as primitives, ensuring that all regularized combinations are valid solids as well [2]. In other systems, primitives include half-spaces, which themselves are not bounded solids. Thus a cube can be defined as the intersection of six half-spaces, or a finite cylinder as an infinite cylinder that is capped off at he top and bottom by planar half-spaces [3].

We use a binary tree structure to store both primitives and connectivity information of CSG objects. Each object is stored as a tree with operators at the internal nodes and simple primitives at the leaves. Each internal node has exactly two children. The root node defines the solid object as a whole.

In addition to primitives and operators, supplementary information referring to position and size of the objects must be stored. Originally, the objects are represented in world coordinates, but each node contains additional information about transformations, thus allowing each part of the object to be liberally positioned in space.

There are five non-trivial combinations of two solids $A$ and $B$: $A \cup B$, $A \cap B$, $A$ - $B$, $B$ - $A$ and $A \otimes B$, which is equivalent to ($A \cup B$) - ($A \cap B$) (*symmetric difference*). However, the most commonly used operators are union, intersection and difference. It is not necessary to support

all three operators, e. g. if the universe ($\mathbb{R}^3$) is considered as a primitive, it is possible to express the intersection in terms of union and difference [6].

The following picture (figure 4) shows an MRI data scan of a human head (volumetric data set), merged with a geometric data set consisting of both polygon data (bottom plane) and a CSG object (surgical knife). The image has been rendered on a “Distributed Volume Priority Z-Buffer” (*DVPZB*), which has been developed in our group at the University of Kaiserslautern [5]. It is a script-file driven test platform for rendering algorithms and visualization techniques. The system helps us to determine which algorithms can be optimized for speed in order to achieve interactive frame rates.

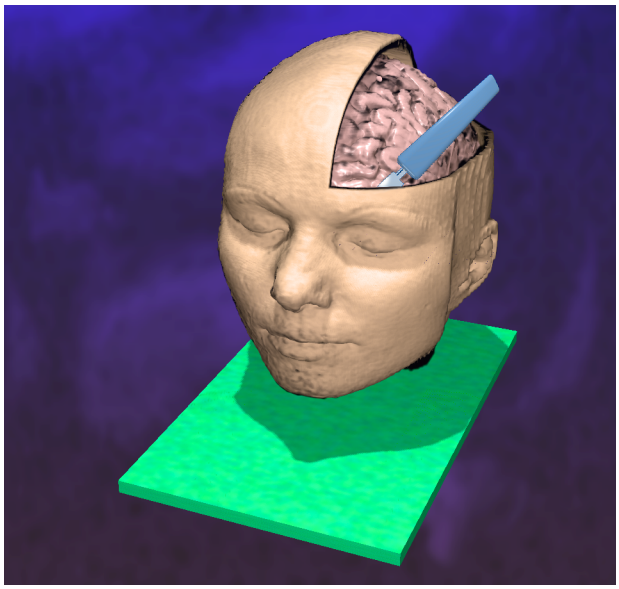

Figure 4: hybrid rendering of a human head

## 3 Data reduction

Different types of data sets require different techniques of data reduction. For our system, we try to apply algorithms which maintain data in the original representation. The topology of a data set should be unchanged, and the approximation should represent the data set with the best accuracy which is possible at a specified level of detail. The algorithm should be scalable, i. e. the level of detail should be controlled by a single parameter, which is set automatically, depending on the computing time which is available in order to achieve interactive frame rates.

### 3.1 Volumetric Data

One of the best algorithms to achieve the goals, which have been pointed out previously, is a multiresolution wavelet approach. This technique allows us to reduce complexity of a data set at different levels of detail. The algorithm is scalable, and thus provides a means for visualization of a data set at different resolutions. The transformation preserves most of the features of a data set, neglecting only high-frequency components, i. e. fine details, in the image, without grouping or clustering the data set into blocks. In contrast to simple node removal algorithms, it preserves gradient information and relevant properties of a data set, without the risk that some important features are hazardously omitted from the grid.

A 3-D version of the algorithm, which works on a grid (instead of a 2-D slice) and thus achieves even better compression rates, is currently under development.

### 3.2 Geometric Data

The geometric data set is subdivided into two groups: polygon data (triangles) and CSG objects. For polygon data, standard algorithms can be applied.

Schroeder et al. [8] describes a data reduction algorithm for decimation of triangle meshes. He tries to reduce the number of triangles in a mesh, which represents the surface of an object, by removing vertices and triangles with only minor contribution to the topology and shape of the object. The resulting hole is retriangulated in order to reduce the number of triangles, which are involved.

During the decimation step all vertices in the mesh are visited and tested if they can be deleted. If a vertex is removed, all surrounding triangles that use this vertex are deleted as well. To perform the tests on a vertex, an ordered list of triangles that use the vertex is created. Two distance criteria are proposed to determine if a vertex becomes subject to deletion.

The original algorithm terminates if the user is satisfied with the number of triangles left. We use both the number of iterations and the complexity of the remaining data set as criteria for termination. This helps us to keep up with time limits.

Turk [9] tries to improve the effectiveness of his algorithm by spreading a set of new vertices onto the surface of the model. A *mutual tessellation* between the original data set and the new vertices is used as an intermediate polygonal model, which is then optimized by locally removing most of the original points and creating a local retriangulation of the neighborhood. If each edge of the data set belongs to a maximum of two triangles only, the algorithm produces a new model of the object without changing its topology. One of the weaknesses of the algorithm is that it changes the shape and probably smoothes sharp edges and well defined corners of an object. For medical applications this might seem to be negligible, but we always have to maintain accuracy of data in the best possible way, if the system is used in clinical surgery. This drawback is abolished by the fact that the algorithm can easily be extended to adapt automatically to variations of surface curvature.

Hoppe, DeRose et al. [10] use the same technique of spreading new points onto the surface. They use an energy function

$$E\,(K, V) = E_{dist}\,(K, V) + E_{rep}\,(K) + E_{spring}\,(K, V)$$

with

$$E_{dist}\,(K, V) \;=\; \sum_{i\,=\,1}^{n} d^2\,(x, \phi\,(|K|)\,)$$

$$E_{rep}\,(K) \;=\; c_{rep} \cdot m$$

$$E_{spring}\,(K, V) \;=\; \sum_{\{j,\,k\}\,\in\,K} \kappa \cdot \left\| v_j - v_k \right\|^2$$

to minimize the original data set. A mesh $M$ is described as a pair $M = (K, V)$, where $K$ reflects the topology of the mesh, and $V$ its geometric realization in $\mathbb{R}^3$. $K$ is called a *simplicial com-*

*plex*, *V* is a set of vertex positions. The geometric realization can be obtained from the topological realization $|K|$ by the linear mapping $\Phi : \mathbb{R}^m \rightarrow \mathbb{R}^3$, which sends the *i*-th standard basis vector of $\mathbb{R}^m$, identified with the vertices of *K*, to $\mathbb{R}^3$.

$E_{dist}$ is the term that measures the deviation of the mesh geometry from the original mesh. This is done by calculating the squared distances from every initial data point to the mesh. To put it in other words, a mesh that deviates more from another mesh, is penalized with a higher energy value.

$E_{rep}$ is the term that monitors the number of vertices in a mesh. The constant value $c_{rep}$ is a parameter that allows the user to choose between a dense mesh with a better fit and a sparse mesh with a somewhat less optimized geometrical fit. If $c_{rep}$ has a small value, meshes with more vertices are preferred, because $E_{dist}$ gains more influence on the total energy. If $c_{rep}$ is large, sparse representations are preferred, because a large number of vertices is penalized.

$E_{spring}$ has been introduced as a regulating term that helps the optimization process to find local minima. This term describes the energy of a spring, which is placed on every edge of the mesh. The spring constant is $\kappa$. The spring energy function eliminates spikes on the surface at regions with no data.

The algorithm minimizes over both *K* and *V*. The inner loop optimizes over *V* with a fixed simplicial complex *K*, while an outer loop minimizes over *K*, with $K \in \mathbf{K}$. $\mathbf{K}$ is the set of simplicial complexes homeomorphic to the initial simplicial complex.

The algorithm reduces the number of vertices in a mesh in a significant way. Vertices are concentrated in regions with high curvature, and long edges are aligned in directions of low curvature. This method guarantees accuracy of data, since sharp features of the original mesh are retained.

CSG objects usually are less complex than polygon objects. Hence often it is not necessary to apply data reduction algorithms in order to raise performance. A simple approach would be the application of mesh reduction algorithms, as used for triangles, to a triangulated representation of the CSG object. But this would require a resampling of the data set and unwanted conversions at early stages of the pipeline. Therefore scalable data reduction algorithms for CSG objects are currently under development.

## 4 Interactive Rendering

Interactive visualization of large medical data sets requires fast rendering, which can be achieved by data reduction, pre-processing of the data set, and a powerful graphic engine. There is always a trade-off between quality and speed, but for medical applications data reduction algorithms should preserve accuracy of data. The speed and overall performance of the pipeline can be ensured using scalable algorithms for data reduction and pre-processing.

Figure 1 in the introductory chapter indicates that interaction can be applied to all stages of the pipeline, starting from the import module, to data reduction and visibility transformation, and finally to the rendering stage.

The rendering stage makes use of an Open GL interface to the graphic engine, which makes the algorithm independent from hardware and enables us to run the software on a high speed Silicon Graphics Indigo 2 Extreme workstation as well as on a regular Pentium personal computer. The algorithm automatically adapts to the system performance, which is provided by a specific hardware. Interactive behavior is maintained throughout the visualization process, only image quality is reduced to a certain degree. But this is compensated by post-processing, i. e. refining, the final image. If no further events have to be processed any more, that

means if there is no input from the user and the mouse is not touched any more, the image is incrementally refined and finally displayed with maximum detail.

Geometric data, such as triangles, can be immediately passed to the graphic engine. CSG objects are currently transformed into surface objects with semi-transparency, which can be handled by the graphic engine at the final stage of the pipeline. But new techniques using VR methods and texturing are presently under development in order to improve overall performance and image quality. Volumetric data can be visualized both as isosurfaces with semi-transparency, or immediately using a texture buffer. The pre-processing stages designate the materials and select the portions of the data set to be visualized.

The parameters of the different stages of the rendering pipeline are initially scaled by heuristics and system performance checks, but during several iterations they automatically adapt to ensure evenly response times and interactive behavior.

Figure 5 shows the graphical user interface (*GUI*) for our InVIS (*Interactive Visualization*) system. It introduces the pipelining concept (on the left), and provides 2-D and 3-D preview tools.

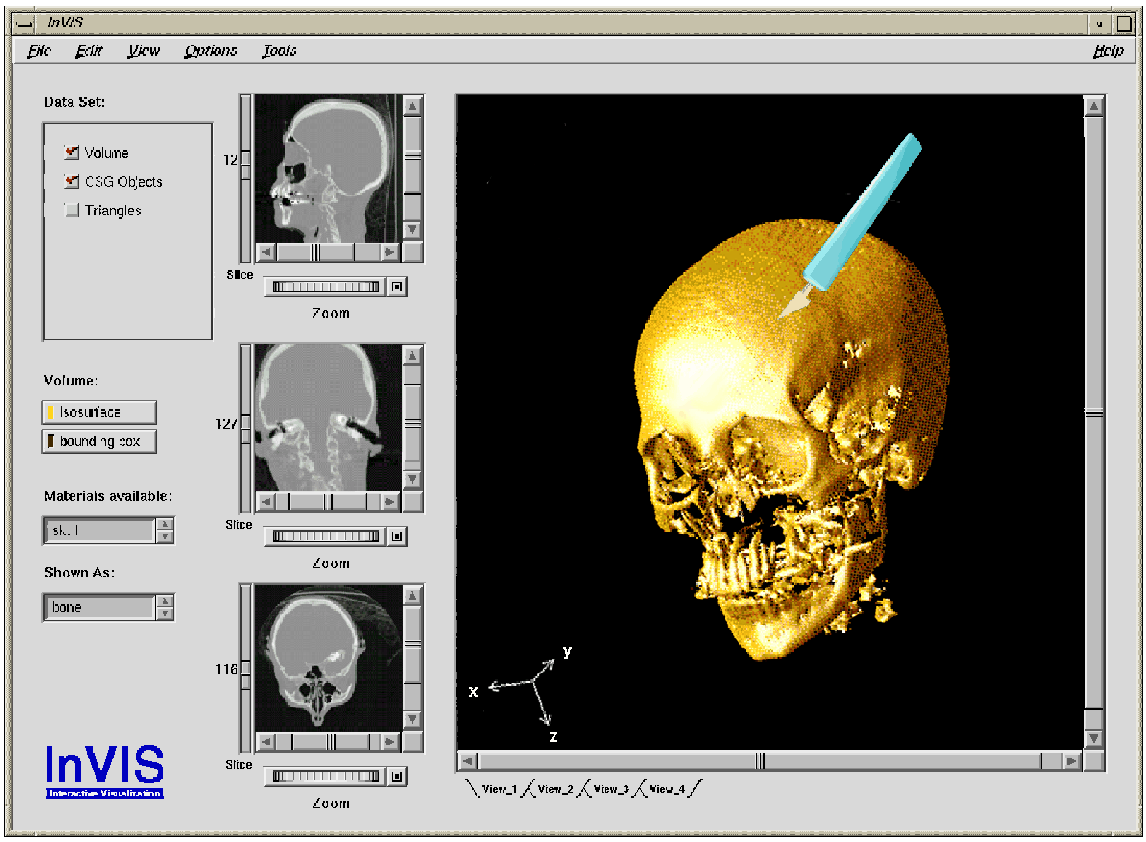


Fig. 5: InVIS user interface

The user interface has interactors to control 2-D slice browsers and a large 3-D working area. On the left, different components of the data set can be turned on and off during visualization. Materials and their appearance can be selected.

Other interactive tools, such as cutting planes or rulers, are hidden in the “Tools” menu. The user interface can be easily adapted to user-specific needs.

## 5 Conclusion

The InVIS system, which is currently under development, is a platform for 2-D and 3-D visualization of hybrid medical data sets. Interactive rendering was the main goal for the development of this new system, and a design parameter for each stage of the pipeline. Overall performance is increased by putting as much effort as possible in the pre-processing, i. e. the initial stages of the pipeline, thus leaving most of the final rendering to a fast graphic engine, which allows interactive manipulation on perspective, size and level of detail, and gives the user the impression of immediate response to his actions.

Some operations, which would require more time than is available for interactive rates, are subdivided into hierarchical steps of execution. First, a coarse visualization is created, which represents the data set in a practical or approximated way, then, if no user action has to be reprocessed, a refined image is calculated in hierarchical time steps.

We introduced several methods of data reduction for different kinds of input data sets, and adapted them to produce approximations with hierarchical levels of refinement and detail. Each stage is designed to be controlled by a single parameter, which describes the level of detail, and is directly related to execution time. The relation is not necessarily a linear one, but this can be achieved approximately by remapping the control parameter. Scaling is based on heuristic approaches and measurement of single stage and overall performance. Using this value, each step of the pipeline is automatically scaled and controlled in order to maintain interactive response rates at the user interface.

## References

[1] Foley, J. D.; van Dam, A.; Feiner, S. K., Hughes, J. F.: *Computer Graphics*; pp. 557-560, 672, 901; Addison Wesley, July 1995

[2] Wyvill, Geoff; Trotman, Andrew: “Exact Ray Tracing of CSG Models by Preserving Boundary Information”; *Department of Computer Science*; University of Otago; Dunedin; New Zealand; pp. 411-428

[3] Rossignac, Jarek: “Representations, Design and Visualization of Solids and of Geometric Structures”, Tutorial; *Eurographics ‘93*, Barcelona; pp. 16-26; September 6-7, 1993

[4] Meyer, Jörg: *Hybrider Raytracer mit Anti-Aliasing*, University of Kaiserslautern, Germany; 1993

[5] Lengen, Rolf H. van: “The Volume Priority Z-Buffer”; *Focus on Scientific Visualization*; Hagen, H.; Müller, H.; Nielson, G. M. (publ.), Springer Verlag, New York; October 1992

[6] Silkenbäumer, Karsten; “Implementierung von Constructive Solid Geometry (CSG)-Objekten in ein hybrides Visualisierungssystem”; University of Kaiserslautern, Germany; December 1996

[7] Lorensen, William E.; Cline, Harvey E.: “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”; *ACM Proceedings, Computer Graphics, Vol. 21*; SIGGRAPH ‘87, Anaheim; July 1987

[8] Schroeder, William J.; Zarge, Jonathan A.; Lorensen, William E.: “Decimation of Triangle Meshes”; *ACM Proceedings, Computer Graphics, Vol. 26, No. 2*; pp. 65-70; July 1992

[9] Turk, Greg; “Re-Tiling Polygonal Surfaces”; *ACM Proceedings, Computer Graphics, Vol. 26, No. 2*; pp. 55-64; July 1992

[10] Hoppe, Hugues; DeRose, Tony; Duchamp, Tom; McDonald. John; Stuetzle, Werner: “Mesh Optimization”; *Computer Graphics Proceedings, Annual Conference Series*, pp. 19-26; 1993

[11] Levoy, Marc: “A Hybrid Ray Tracer for Rendering Polygon and Volume Data”; *IEEE Computer Graphics and Applications 10 (3)*; 1990

[12] Mosher Jr., C. E.; Johnson, E. R.: “Integration of Volume Rendering and Geometric Graphics”; *Proceedings of the Chapel Hill Workshop on Volume Visualization*; Chapel Hill, HC; May 1989