Green, S. and Paddon, D. J. Exploiting Coherence for Multiprocessor Ray Tracing. IEEE Computer Graphics and Applications, 9(6), November 1989, pp 12-26.

Hart, J. C., Lescinsky, G. W., DeFanti, T. A. and Kauffman, L. H. Scientific and Artistic investigation of multi-dimensional fractals on the AT & T Pixel Machine. The Visual Computer 9, 1993, pp 346-355.

Levoy, M. Display of Surface from Volume Data. IEEE Computer Graphics and Applications, 8(3), 1988, pp 29-37.

Neumenn, U. Parallel Volume Rendering Algorithms Performance on Mesh-Connected Multiprocessors. Proceedings of Siggraph 1993 Parallel Rendering Symposium, San Jose, 1993. pp 97-104.

Paddon, D. J. and Chalmers, A. G. The Effect of Configurations and Algorithms on Performance. In Parallel Computing on Distributed Memory Multiprocessors, NATO ASI Series, Springer-Verlag, 1993, pp 77-97.

Priol, T. and Bouatouch, K. Static Load Balancing for a Parallel Ray Tracing on a MIMD Hypercube. The Visual Computer, 5(1/2), 1989, pp 109-119.

Sabella, P. A Rendering Algorithm for Visualizing 3D Scalar Fields. Computer Graphics 22(4), 1988, pp 51-88.

VISUALIZATION OF THE VOLUME DATASETS IN SCIENCE

Juraj Jankovič
Faculty of Mathematics and Physics
Comenius University
842 15, Bratislava, Slovakia

Volume visualization is one of the fast-growing areas in scientific visualization. It helps to look at scalar or vector datasets and understand them more easily. An overview of basic algorithms used for this purpose, some of enhancements and optimizations are described here.

Introduction

This paper contains a summary of the most important contributions to visualization of volume data. The earliest techniques are mentioned together with more detailed descriptions of the well-known algorithms, which are already used in science, and the newest enhancements.

The first part explains the terms, procedures and heuristics used in the field. This is important, because the standard language is still not established for this field. The main part covers volume rendering techniques and algorithms, their advantages and disadvantages. More space is devoted to the most important methods, but it is impossible to cover everything in one paper. So it is necessary to consult the papers listed in bibliography for detailed information.

Data

Because the large number of fields of science, which produce the volume datasets, is increasing all the time, together with the new applications and techniques, the volume data can be obtained from different sources. The basic categories of these sources were identified in [Watt92] :

1. Empirical data sets constructed from a mathematical model such as Computational Fluid Dynamics (CFD).

2. Data sets derived from an object, such as by tomographic scanning in the medical area.

3. Data set is some kind of computer graphics model.

There are many techniques available today. For the second category for example - Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), Single Photon Emission Computed Tomography (SPECT) - are all used in medicine. These are the methods used mostly in science. However, there exist some another ways of acquiring volume data [Elvi92] : voxelizing geometric description of objects, sculpting digital blocks of marble, hand-painting in three-dimensions with a wand, or writing programs that generate interesting volumes using stochastic methods.

The algorithms described here deal with visualizing single scalar data volumes. But there are vector, tensor, bi-modal, and higher-dimensional data as alternatives to scalar data, which are used in different fields of science. Some of the techniques can be extrapolated to visualize these types of data, although visualization of most of them is still an active area of research.

Volume

Volume datasets are usually treated as an array of volume elements. Two approaches are used depending on volume element type. Resampling the volume between gridpoints during rendering process occurs in almost every algorithm. This requires interpolation and it is impossible to check its reliability, since the underlying function is not usually known, and it is not known whether the function was sampled above the Nyquist frequency [Brac86], [Watt92]. It must be assumed that common interpolation techniques are valid for an image to be considered valid.

The first volume element type is a voxel. It is an area of non-varying value surrounding a central gridpoint. This approach has the advantage that no assumptions are made about the behavior of data between gridpoints - only known data values are used for generating an image.

The second type is a cell. It is a volume element (usually a hexahedral area) which corners are gridpoints and which value varies between the gridpoints. Trilinear and tricubic ones are the most commonly used functions for interpolating values inside the cell. Images generated using this approach appear smoother, but the validity cannot be verified.

There is a number of data volume geometries. Nonuniform data geometries are common in computational fluid dynamics, meteorology, geology and so on. Following taxonomy of grids is presented by Speray and Kennon [Sper90] :

1. *Cartesian (i,j,k)* Typically a 3-D matrix with no intended world coordinates, so subscripts map identically to space. The data elements are cubic and axis aligned.

2. *Regular (i.dx,j.dy,k.dz)* Cells are identicall rectangular prisms aligned with the axis.

3. *Rectlinear (x[i],y[j],z[k])* Distances between points along an axis are arbitrary. Cells are still rectangular prisms and axis aligned.

4. *Structured (x[i,j,k],y[i,j,k],z[i,j,k])* This type, also known as *curvilinear*, allows nonboxy volumes to be gridded. Logically, it is a cartesian grid subjected to non-linear transformations so as to fill a volume or wrap around an object. Cells are hexahedra. See figure 1.

5. *Block structured (x [i,j,k],y [i,j,k],z [i,j,k])* Several structured grids in the same data volume.

6. *Unstructured (x[i],y[i],z[i])* There is no geometric information implied by this list of points and edge/face/cell connectivity must be supplied in some form. Cells may be tetrahedra, hexahedra, prisms, pyramids, etc., and they may be linear (straight edges, planar faces) or higher-order (e.g. cubic edges, with two interior points on each edge). See figure 2.

7. *Hybrid* It may occasionally be desirable to use structured and unstructured grids together, putting each where their fitting and computational strengths are most beneficial. See figure 3.

Speray and Kennon present also a technique for cutting these geometries to display cross-sections interactively.

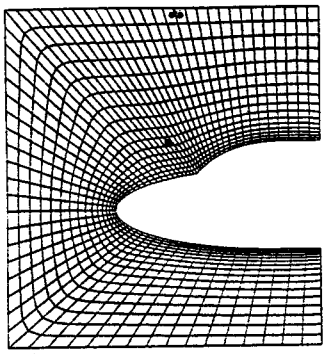
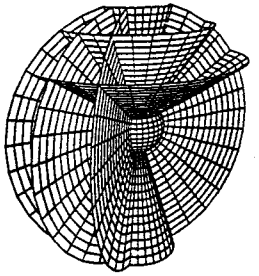


Figure 1 : a) Structured grid



b) Simple 3D structured grid

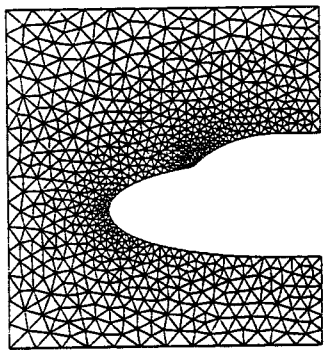


Figure 2 : Unstructured grid

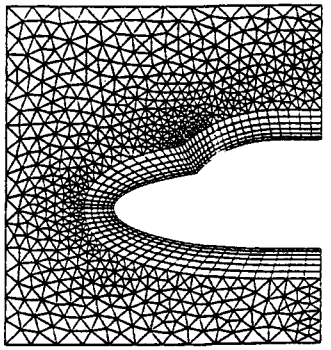


Figure 3 : Hybrid grid

Methods

The classification of volume visualization methods introduced by Elvins [Elvi92] follows. He divides the fundamental algorithms into two categories (see table 1):

Direct Volume Rendering (DVR) - These methods are characterized by mapping elements directly into screenspace. They are especially appropriate for creating images from datasets containing amorphous features like clouds, fluids, and gases. One disadvantage of these methods is that the entire dataset must be traversed each time an image is rendered.

Surface-Fitting (SF) - They fit surface primitives such as polygons or patches to constant-value contour surfaces in volumetric datasets. The treshold chosen by the user is used as this constant-value. Rendering hardware and well-known rendering methods are used to quickly render the extracted surface. These volume visualization methods suffer from problems such as false positive and negative surface pieces, and incorrect handling of small features in the data.

Volume Visualization Algorithms		
Surface-Fitting	Direct Volume Rendering	
	Projection methods	Image-order methods
Opaque cubes (Cuberille) Contour connecting Marching cubes Dividing cubes	V-buffer Splatting Pixar slice shearing (Volume rendering)	Ray casting Cell integration Sabella method
Marching tetrahedra Splitting box	Fourier volume rendering	

Table 1. Algorithms in the field of volume visualization

The problem when externally defined geometric objects are needed in the final image (for example - when radiation beam paths, defined as cylinders or cones are added to scene with volume rendered tumors from MRI) can be solved generally by two ways :

1. use a SF algorithm to find the iso-surface of volume object, then render this isosurface together with geometric objects,
2. voxelize the geometric objects into either rendered or separate volume, render the volume(s) using a DVR technique.

One more basic characteristic of volume visualization method is important. It is the way of creating the image. Either an image-order traversal of the pixels in the image plane or an object-order traversal of the elements in the volume can be used. And the later one can be performed in two ways : front-to-back or back-to-front. Each type of traversal has its own advantages and even their combinations are used in some algorithms.

Data

Another function of volume visualization is data classification. It is worth the attention, because it is important that the user sees what he needs to, and any incorrect colors or other errors can be at least confusing. In case of SF method data classification means choosing a treshold - a value of material which surface has to be visualized. Usually choosing only one value at given time is possible. A color table is created in case of DVR method. It contains color and opacity values corresponding to the range of data values. This table is usually specified by the user and the quality of the final image depends most on his knowledge of material and experiencies. Programmers are responsible for ensuring that interpolated values cannot map to non-existing materials or incorrect colors.

Viewing and shading

Using perspective views in volume visualization is not without problems, because it can cause misrepresentation of the data. On the other hand, it can help to see "the depth" of the image. Some techniques for using perspective projection are described in [Novi90]. Most algorithms use orthographic viewing together with other depth cues (animation, depth-brightness attenuation ...).

Gradient shading is the most frequently used shading method in visualiation of volume datasets. The gradient is found at each gridpoint by applying a central difference formula. Gradients within a cell are interpolated. The gradient at a point in a volume approximates the normal to an imaginary surface passing through the point. Most standard shading models can be applied afterwards. Some algorithms precalculate dot products of the light source vector and the normal vector and store the result, some others do heuristic shading with a table lookup. Gradients are rather used than normals of the surface primitives in SF methods, because of accuracy.

Surface-fitting algorithms

One of the first techniques, called the *opaque cube* or *cuberille* algorithm, was reported by Herman and Liu [Herm79]. They apply a treshold to 3-D data set to detect desired structure. The resulting boundary voxels are treated as opaque cubes and rendered. A more elaborate approach extracts a polygonal surface from the voxels. The blocky appearance of a given solid can be improved by lowpass filtering. Shading is provided by assigning a shade to each face using a variant of Warnock's reflection model [Warn69]. A Z-buffer algorithm is used for hidden surface removal. This technique can be improved by using the value of the local gradient to determine the normal used in shading calculations. Opaque cube algorithms are especially bad at showing small features of data, but are simple to implement and fast.

Another early volume visualization technique was reported by Farrell [Farr83]. He simply overlays successive 2-D frames, displacing each image in x and y. An HSL color model is used. Intensity within a particular frame is mapped into hue. The color selected in 3-D space is a function of the z-depth of the image. Decreasing lightness is used in planes of decreasing z (figure 4). Farrell also considers some facilities that have become standard in volume rendering, including an interactive view point change and a cut plane for displaying details.

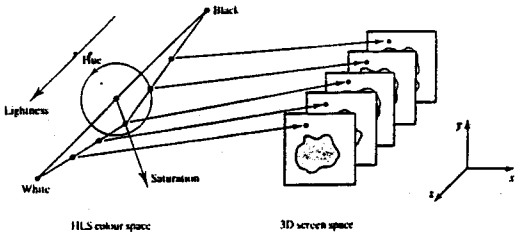


Figure 4 : Pseudocolor mapping for volume data

A well known method is called *contour-connecting*. This idea was originally suggested in [Kepp75] and later refined in [Fuch77], [Ekou91], and many others. It is an object-order surface-fitting method. After specifying a treshold value, a closed contour is found for each data slice at this value. This is provided by image processing techniques now, but human intervention is still useful for low-contrast data. Next problem is the one of finding an optimal tesselation (triangles are used usually) connecting the curves in each two adjacent slices. This can be reduced to finding a path in a directed graph using heuristics [Kepp75], or to finding a minimum cost path in a directed toroidal graph [Fuch77]. After selecting viewing, lighting and rendering parameters, the strips of triangles are passed to a surface renderer. Main advantages of this method are simplicity and a number of known surface-rendering algorithms.

The *marching cubes* algorithm was introduced by Lorensen and Cline [Lore87] but the similar method was independently reported by Wyvill and McPheeters [Wyvi86]. In the first phase of the algorithm the user specifies a treshold value to define desired surface. Next all cubes0 (cells) that are intersected by this surface are found. Cells that straddle the treshold are closely examined in the second phase. Value of each of eight corners of the cube can be above or below the treshold value. That gives 256 different ways in which a surface can intersect a cube. This number can be reduced to 15 by reflection and rotation (figure 5). Triangles creating the final surface are formed from groups of three cell-edge intersection points. Gradients at these points are interpolated between gradients at the corners of the cell and later used for shading.

This process sometimes results in false positive and negative triangles in the iso-surface because of connecting the wrong set of points. The *marching tetrahedra* algorithm breaks up each cell into five [Shir90], six, or 24 tetrahedra and do the edge intersection tests with the tetrahedra. Two triangles are sufficient to create the iso-surface in case of tetrahedral cell, so problem of connecting the points is reduced. The disadvantage of this

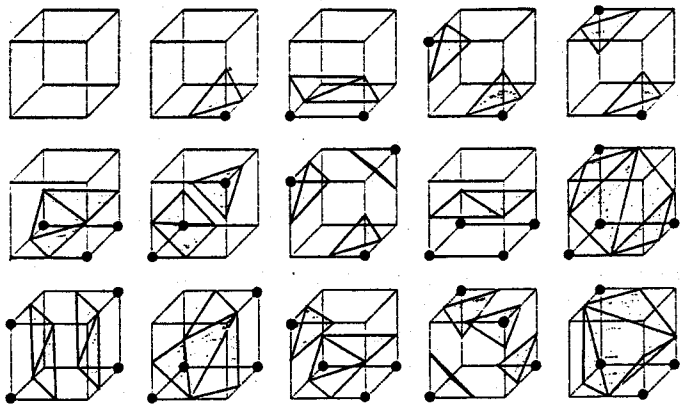


Figure 5 : Triangulated cubes

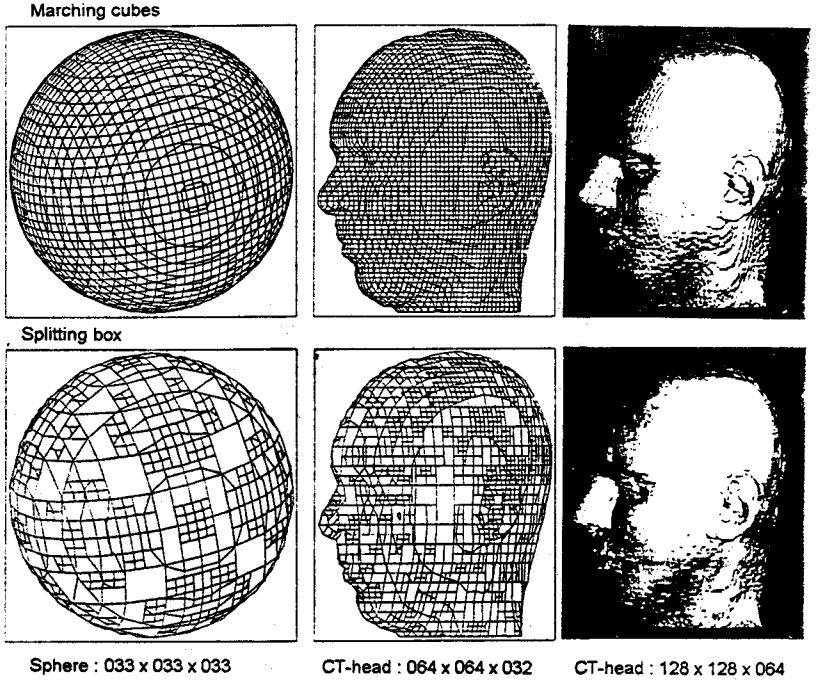


Figure 6 : Comparison of marching cubes and splitting box algorithms

method is that it generates more triangles. Another way of reducing the ambiguous point-connecting situation is described in [Niel91].

The size of the triangles generated in the 'marching' methods, which is often smaller than the size of the pixel, lead to new algorithm, called *dividing cubes* [Clin88]. Each cell in the volume, with corner values that straddle the treshold, is projected into screen space. If it projects into an area larger than a pixel, it is divided into subcells, each of which is rendered as a surface point. Otherwise the entire cell is rendered as a surface point.Surface points are composed of a value, a location in object space, and a gradient for shading. They are rendered into the image buffer using a standard algorithm such as painter's or z-buffer. So dividing cubes algorithm does not use any intermediate surface primitives. Rendering surface points instead of surface primitives saves a great deal of time. There exists also a hardware implementation of the dividing cubes algorithm [Clin90].

The newest contribution to development of surface-fitting algorithms is *splitting-box* algorithm reported by Müller and Stark [Müll93]. It is based on marching cubes and improves it by reducing the large number of contour chains (figure 6). Algorithm starts with the box given by the input grid. This one is bisected perpendicular to its longest edge into two sub-boxes. The resulting boxes are recursively bisected untill a 2x2x2 box is reached. Each box arising during this process is checked whether each of its 12 edges possesses at most one transition of the contour surface (Müller and Stark call such a box *MC box*). Contour chains are generated for recognized MC box with the same approach as used with the marching cubes algorithm. Then bisection of the box is continued with the goal of checking the quality of approximation of the contour chains with respect to the true contour surface in the MC box. If the approximation is acceptable, it is used for output. Otherwise, the bisecting process is used to find a better approximation.

Direct Volume Rendering

Together with its optimizations and enhancements, *ray-casting* is the most often used DVR algorithm [Tuy84], [Levo88], [Upso88], [Levo90a]. It is based on firing a ray from each pixel of the screen space through the volume. The opacities and colors are summed along the ray until the opacity reaches the unity or ray exits the volume.

When the data-classification tables are set up and viewing and lighting information is specified, the rays are fired in an image-order traversal of the pixels. An interpolation is used to find the value for the intersection point inside the cell, or the value of the nearest gridpoint is used in case of voxel-based approach. Then the gradient shading is used for color and the opacity is attenuated. The resulting values of opacity and color are added to the pixel and the same process is repeated for next step along the ray. If the ray exits the volume before reaching the unity of the opacity value, the accumulated color tuple is multiplied by the opacity.

Difference between ray-casting and ray-tracing is that the casted rays are not bounced after hitting reflective objects, but they continue in a straight line. So rays can be fired from each pixel independently, and ray-casting can be parallelized at the pixel level.

Some optimizations were proposed by Levoy [Levo90b]. The first one is based on hierarchical spatial enumeration. He works with the pyramid of binary volumes, with goal to reduce the number of intersection points needed for summing the total opacity and color of the pixel. The second one is the adaptive termination of ray casting. Its goal is to quickly identify the last sample location along the ray that significantly changes the color of the pixel.

Sabella [Sabe88] describes an alternative approach. He models voxels as light-emitting particles. Four values : attenuated light intensity, maximum light value, distance of this maximum value, and the centroid of voxel-value along the ray, are accumulated for each ray. These values are set in relation to the hue, saturation, and lightness in HSV model. This process produces high-quality, but not realistic images, which are good for perceiving and understanding of the field.

Different type of direct volume rendering algorithms is represented by *V-buffer* [Upso88], and *splatting* [West90]. Upson and Keeler presented the V-buffer method in two ways. The first one is similar to ray-casting, with higher order interpolation, subcell quadrature, and enhanced efficiency (simple data structure, no shadowing ...). The second method is called cell-by-cell processing. It performs a front-to-back object-order traversal of the cells in the volume. Interpolation between cell corner-values is used and contribution of

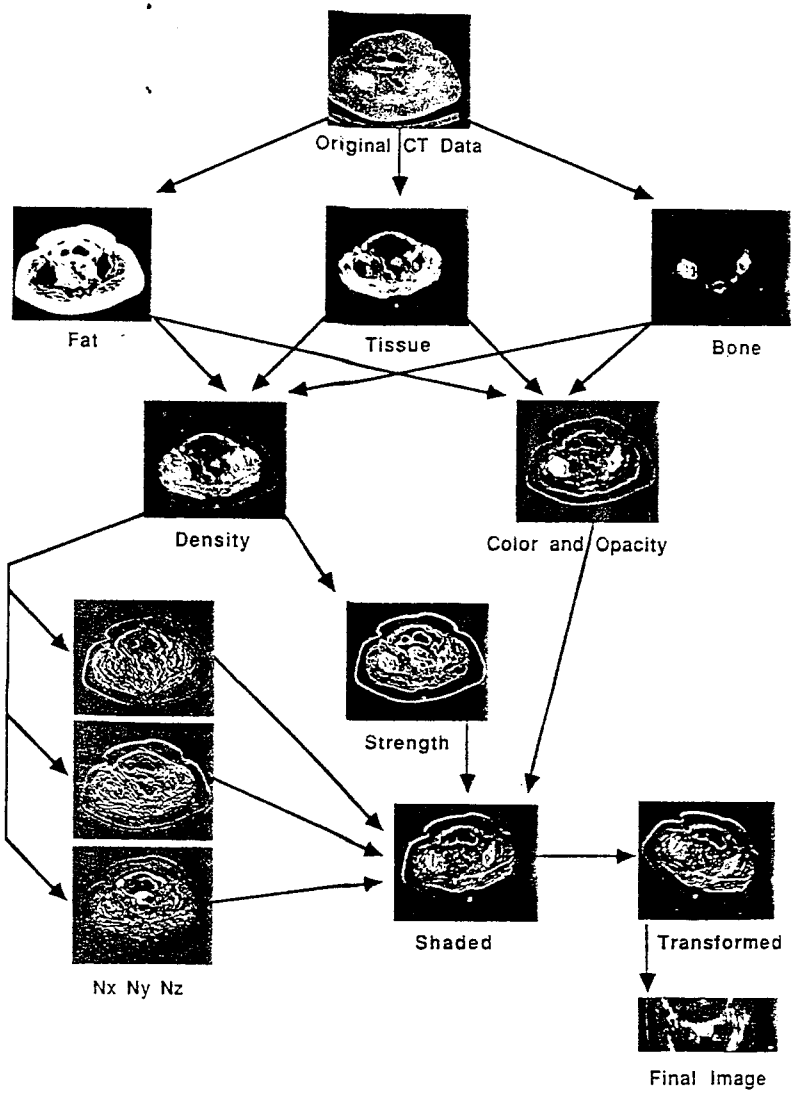


Figure 7 : Volume rendering process

each cell is stored in an image buffer. The splatting algorithm is similar to this second method, but it is voxel-based. Some optimizations for splatting are described in [Hanr91].

Montani and Scopigno reported the Sticks representation scheme for volumetric data [Mont90]. This scheme produces a degree of data compresion greater than the Octree model. And beside the reduction of the required memory space, it also improves the performance of the rendering algorithms based on ray-casting. This technique allows an efficient representation and rendering of volume datasets on low-capability workstations.

The algorithm reported by Drebin, Carpenter, and Hanrahan as *Volume rendering* [Dreb88], takes advantage of the hardware capabilities of the Pixar Image Computer and is rarely used on other type of hardware. But it is a typical direct method for rendering volumes containing mixtures of materials (figure 7). The simulation of the absorbtion of light along the ray path to the eye is used. Attention is devoted to avoiding any artifacts caused by aliasing and quantization.

An interesting metnod was presented by Malzbender as *Fourier Volume Rendering* [Malz93]. This new approach is not a typical direct volume rendering technique. It operates on a frequency domain representation of the dataset and efficiently generates line integral projections of the spatial data it represents. The Fourier Projection-Slice Theorem is applicated for computing 2-D projections of 3-D datasets using only a 2-D slice of the data in the frequency domain. The advantage of this method is a significantly lower computational cost - the projection images can be computed with one to two orders of magnitude fewer operations than the classical methods. The resulting images look like X-rays of the dataset with no hidden surface effects, but sometimes this is not the disadvantage (figure 8).

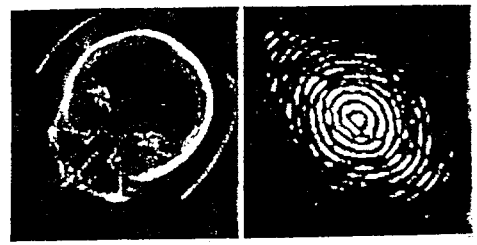


Figure 8 : a) Rendered image of cranial CAT data set

b) Its frequency domain representation

Summary

An overview of the most known algorithms and methods for visualization of the volume datasets was presented, including the earliest attempts together with the later approaches and the latest contributions. It will serve for selection of the techniques which will be implemented and may be developed in our future work. Our aim is also a development of some physically-based model and its rendering using some of the volume rendering techniques.

Most of the presented methods are used in medicine and so they are designed to display medical data, but they are succesfully applicable in other fields of science, such earth science, microscopy, and engineering.

The user has to decide which approach is the best for his purposes considering all the advantages and disadvantages of the methods. Surface-fitting algorithms, for example, produce a solid surface and it is impossible to see the interior of the volume and the rendering process has to be repeated each time the display of the surface of other treshold value is needed. On the other hand it enables the user to use the techniques developed for solid models. Another advantage appears when the interactive change of viewpoint is needed - the volume does not has to be rendered each time the viewpoint is changed. The direct volume rendering has to be performed with each change of the view but it gives more complex display of the dataset what is sometimes more important. And the techniques are still in development and in combination with improving hardware (higher display resolution, higher speed, larger memory) they could give much better results in few years.

Bibliography

[Clin88] Cline H.E., Lorensen W.E., Ludke S., Crawford C.R, Teeter B.C.
Two Algorithms for Three-dimensional Reconstruction of Tomograms
Medical Physics, Volume 15, Number 3, May/June 1988

[Clin90] Cline H.E., Ludke S., Lorensen W.E., Teeter B.C.
A 3D Medical Imaging Research Workstation
Volume Visualization Algorithms and Architectures, ACM SIGGRAPH '90 Course Notes, Course Number 11, ACM Press, August 1990

[Dreb88] Drebin R., Carpenter L., Hanrahan P.
Volume Rendering
Computer Graphics, Volume 22, Number 4, August 1988

[Ekou91] Ekoule A.B., Peyrin F.C., Odet C.L.
A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours
ACM Transactions on Graphics, Volume10, Number 2, April 1991

[Elvi92] Elvins T.
A Survey of Algorithms for Volume Visualization
Computer Graphics, Volume 26, Number 3, August 1992

[Farr83] Farrel J.
Colour Display and Interactive Interpretation of Three-dimensional Data
IBM J. Res. Develop., 27(4), 356-66

[FoDa91] Foley J., Van Dam A., Feiner S., Hughes J.
Computer Graphics - Principles and Practice
Addison-Wesley Publish. Comp., 1991

[Fuch77] Fuchs H., Kedem Z.M., Uselton S.P.
Optimal Surface Reconstruction from Planar Contours
Comunications of the ACM, Volume 20, Number 10, October 1977

[Good89] Goodsell D., Mian S., Olson J.
Rendering Volumetric Data in Molecular Systems
Journal of Molecular Graphics, March 1989

[Hanr91] Hanrahan P., Laur D.
Hierarchical Spatting: A Progressive Refinement Algorithm for Volume Rendering
Computer Graphics, Volume 25, Number 4, August 1991

[Herm79] Herman G.T.,Liu H.K.
Three-dimesional display of Human Organs from Computed Tomograms
Computer Graphics and Image Processing, Volume 9, Number 1, January 1979

[Hibb89] Hibbard W., Santek D.
Visualizing Large Data Sets in the Earth Sciences
IEEE Computer, Volume 22, Number 8, August 1989

[Kepp75] Keppel E.
Approximation Complex Surfaces by Triangulation of Contour Lines
IBM J. of Research and Develop., Volume19, Number 1, January 1975

[Krue90] Krueger W.
Volume Rendering and Data Feature Enhancement
Computer Graphics, Volume 24, Number 5, November 1990

[Levo88] Levoy M.
Display of Surfaces from Volume Data
IEEE Computer graphics and Applicatons, Volume 8, Number 3, March 1988

[Levo90a] Levoy M.
Volume Rendering, A Hybrid Ray Tracer for Rendering Polygon and Volume Data
IEEE Computer graphics and Applications, Volume 10, Number 2, March 1990

[Levo90b] Levoy M.
Efficient Ray Tracing of Volume Data
ACM Transactions on Graphics, Volume 9, Number 3, July 1990

[Lore87] Lorensen W.E., Cline H.E.
Marching Cubes : A High Resolution 3D Surface Construction Algorithm
Computer Graphics, Volume 21, Number 4, July 1987

[LoPi91] LoPiccolo P.,Mahoney D., Vasilopoulos A., Porter S.
The Visible Volume - internal Medicine, Small Worlds, A New World View, Insightful Analysis
Computer Graphics World, Volume 14, Number 4, April 1991

[Malz93] Malzbender T.
Fourier Volume Rendering
ACM Transactions on Graphics, Volume 12, Number 3, July 1993

[Mont90] Montani C., Scopigno R.
Rendering Volumetric Data using the STICKS Representation Scheme
Computer Graphics, Volume 24, Number 5, November 1990

[Müll93] Müller H., Stark M.
Adaptive Generation of Surfaces in Volume Data
The Visual Computer, 9, 1993

[Niel91] Nielson G.M., Hamann B.
The Asymptotic Decider : Resolving the Ambiguity in Marching Cubes
Proceedings of the IEEE Visualization '91 Conference, IEEE Computer Society Press, October 1991

[NeDr90] Ney D., Fishman E., Magid D., Drebin R.
Volumetric Rendering of Computed Tomography Data : Principles and Techniques
IEEE Computer Graphics and Applications, March 1990

[Novi90] Novins K., Sillion F., Greenberg D.
An Efficient Method for Volume Rendering Using Perspective Projection
Computer Graphics, Volume 24, Number 5, November 1990

[Sabe88] Sabella P.
A Rendering Algorithm for Visualizing 3D Scalar Fields
Computer Graphics, Volume 22, Number 4, August 1988, 51-58

[Shir90] Shirley P., Tuckman A.
A Polygonal Approximation to Direct Scalar Volume Rendering
Computer Graphics, Volume 24, Number 5, November 1990

[Sper90] Speray D., Kennon S.
Volume Probes : Interactive Data Exploration on Arbitrary Grids
Computer Graphics, Volume 24, Number 5, November 1990, 5-12

[Tuy84] Tuy H.K., Tuy L.t.
Direct 2-D Display of 3-D Objects
IEEE Computer Graphics and Applications, Volume 4, Number 10, October 1984

[Upso88] Upson C., Keeler M.
The V-Buffer : Visible Volume Rendering
Computer Graphics, Volume 22, Number 4, August 1988, 59-64

[Warn69] Warnock J.
A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures
Univ. Utah Computer Sci. Dept.,TR 4-15 (NTIS AD-753 671) 1969

[Watt92] Watt A., Watt M.
Advanced Animation and Rendering Techniques - Theory and Practice
Addison-Wesley Publish. Comp.,1992

[West90] Westover L.
Footprint Evaluation for Volume Rendering
Computer Graphics, Volume 24, Number 4, August 1990

[Wolf88] Wolfe R.H., Liu C.N.
Interactive Visualization of 3D Seismic Data : A Volumetric Method
IEEE Computer Graphics and Applications, Volume 8, Number 7, July 1988

[Wyvi86] Wyvill G., McPheeters C., Wyvill B.
Data Structure for Soft Objects
The Visual Computer, Volume 2, Number 4, August 1986