Interactive Maximum Projection Volume Rendering

Wolfgang Heidrich
Computer Graphics Lab
University of Waterloo
wheidrich@cgl.uwaterloo.ca

Michael McCool
Computer Graphics Lab
University of Waterloo
mmccool@cgl.uwaterloo.ca

John Stevens
Playfair Neuroscience Unit
Toronto Western Hospital
john@camtwh.eric.on.ca

Abstract

Maximum projection is a volume rendering technique that, for each pixel, finds the maximum intensity along a projector. For certain important classes of data, this is an approximation to summation rendering which produces superior visualizations.

In this paper we will show how maximum projection rendering with additional depth cues can be implemented using simple affine transformations in object space. This technique can be used together with 3D graphics libraries and standard graphics hardware,thus allowing interactive manipulations of the volume data. The algorithm presented in this paper allows for a wide range of tradeoffs between interactivity and image quality.

1 Introduction

The existing approaches to volume visualization can be classified into two categories: direct volume rendering and model based techniques. While these two techniques have often been portrayed as competitors, we think they should actually be seen as being complementary.

The method described in this paper uses geometric models to implement a discrete version of a well known direct rendering algorithm. These models allow for the efficient use of widely available computer graphics hardware.

In the following we briefly review some aspects of direct volume rendering before presenting our discrete technique.

1.1 Direct Volume Rendering

Direct volume rendering [6, 8, 11] integrates the value of a *continuous* volume function along a given projector. This continuous volume function usually has to be reconstructed from discrete sampling points.

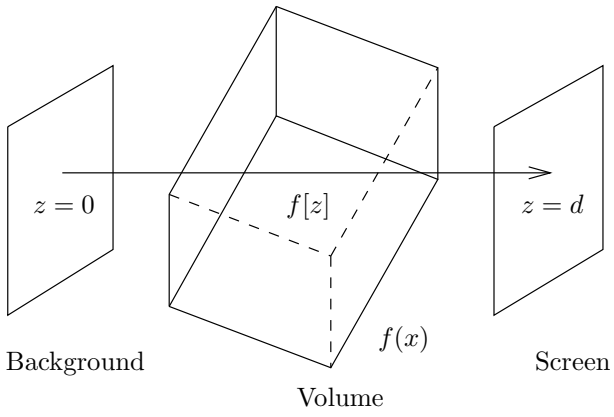


Figure 1: *The path along a projector is parameterized by the value z. Each projector starts at a background image and extends to an imaging plane.*

Typically, the function computed along a projector is an absorption/emission integral. This integral is based upon the physical metaphor of emissive particles suspended in an absorptive medium. Along a single projector, the integral can be written as [14, 9]

$$\Phi[d] = \int_0^d e^{-\int_z^d \mu[\zeta]d\zeta} \phi[z]dz + e^{-\int_0^d \mu[z]dz} \Phi[0]. \quad (1)$$

Here we parameterize every projector linearly and define the function $f[z] = f(\mathbf{x}_0 + z\vec{r})$ with $|\vec{r}| = 1$ along each projector, where $z = 0$ at some background plane beyond the volume of interest and $z = d$ at the imaging plane. This notation is adopted so that the direction of light along the projector towards the pixel corresponds to positive increments in the parameter z. The situation is shown in Figure 1.

The value $\Phi[d]$ is the flux density reaching the imaging plane, and $\Phi[0]$ defines a background image. $\mu(\vec{x})$ denotes the linear absorption coefficient of the medium, and the flux density towards the imaging

plane per unit length, emitted by suspended particles, is given by $\phi(\vec{x})$. These values need to be derived somehow from the original volume data, which is typically not in this form (otherwise we could just look at it directly). All these values may have a wavelength dependency as well, allowing color images to be produced.

By manipulating the way in which the values of the original volume data are interpreted as absorptive $\mu(\vec{x})$ and emissive $\phi(\vec{x})$ properties, various effects can be achieved, including isosurfaces [10, 11] and apparently opaque objects.

1.2 Summation Rendering

The integral in (1) is nonlinear and therefore the number of simplifications that can be applied to it are limited. An important subclass of volumes for which this integral becomes linear is the class of purely emissive volumes.

A purely emissive volume is defined by setting the absorption coefficient $\mu(\vec{x}) = 0$ for all $\vec{x}$, resulting in a totally transparent volume. The rendering integral (1) reduces to

$$\Phi[d] = \int_0^d \phi[z]\, dz + \Phi[0],$$

which is linear with respect to $\phi[z]$. This integral mimics the image formation process of the gamma camera [1]. In this paper this method will be referred to as *summation* rendering.

1.3 Maximum Rendering

Maximum rendering finds the maximum emission along a projector and assigns this value to the projected point on the image plane. Unlike summation rendering, this is a nonlinear operation. However, we will show how widely available computer graphics hardware can be used to implement this maximum operator in an efficient way.

Maximum projection rendering is particularly useful for any application in which a strong signal is produced over a small volume, and might be obscured by low-level background noise produced over a much larger volume. In this case, maximum rendering is an improvement over summation rendering because the nonlinearity removes the "fog" that might obscure the small bright signal.

Maximum rendering can be characterized by the following equation:

$$\Phi[d] = \max(\max_{z\in(0,d]}(\zeta\phi[z]), \Phi[0]).$$

The scale factor ζ converts the flux density per unit length $\phi[z]$ to a flux density Φ. In the discrete case $\phi[z]$ might be sampled at some interval Δz, which would lead to $\zeta = \Delta z$.

This maximum operation, when given two equally-bright points on a projector, should choose the one with the largest z value, that is, the one closest to the image plane. This will become important once we consider intensity depth cueing in Section 2.2.

1.4 Discrete Maximum Rendering

A discrete version of maximum rendering can easily be derived by making a transition from a continuous emission function $\phi(\vec{x})$ to a set of discrete emission levels $\{\phi_1, \phi_2, \ldots, \phi_n\}$. These levels can be seen as thresholds which define a number of nested volumes or "shells" [17].

The maximum rendering algorithm can then be implemented by intersecting each projector with the volumes defined by the emission thresholds. The image intensity at the projector is the emission of the brightest volume that the projector intersects.

Similarly a discrete version of summation rendering can be implemented by summing up the differential intensities of the intersected volumes, weighted by length of the projector segment that lies within the volume.

The intersection of the projector with the volumes can be computed with the help of isosurfaces. The more volumes (and therefore isosurfaces) used, the closer the rendering can approach direct volume rendering. By using multiple surfaces and suitable blending techniques, the perception of solid objects with definite boundaries is softened. Analysis capabilities offered by the discrete approach remain, although now statistics are presented for multiple volumes. This is an advantage, however, since it allows the user to analyze the sensitivity of the statistics to the segmentation threshold.

2 Use of Graphics Acceleration Hardware

While maximum rendering can be directly translated into a ray casting scheme, it is somewhat harder to come up with a rendering method which makes use of graphics acceleration hardware.

Today's 3D graphics boards provide hardware support for drawing several hundred thousand Gouraud shaded polygons per second. The hidden surface removal algorithm used on these boards is almost always depth-buffer (z-buffer) based.

On first thought, one might think that maximum rendering and summation rendering in such an environment can simply be implemented by drawing the isosurfaces with a transparency which accounts for the emission of the corresponding volume. Unfortunately the z-buffer algorithm only produces correct results for transparency if the polygons are sorted by z [13]. Resorting all polygons for every rotation of the volume, however, is computationally expensive even for relatively small volumes.

2.1 Geometric Transformations for Maximum Rendering

Maximum rendering as discussed in Section 1.3 first finds the maximum emission along a projector, and then projects this value onto the image plane. The order of operations can be reversed, so that first the whole volume is projected onto the image plane, and then for each pixel the brightest value is computed.

This corresponds to an orthographic or perspective transformation followed by a maximum operation. Such a maximum operation is already provided by the z-buffer algorithm during hidden surface removal. We would like to use this z-buffer for our maximum operation on intensities as well. Therefore we have to come up with a geometric transformation which reorganizes the geometry so that brighter points are closer to the image plane than darker points.

A simple way to accomplish this, if the z-axis is perpendicular to the image plane, is to set the z component of each point to its emission after the projective transformation. The transformation matrix which does this is

$$\begin{bmatrix} x \\ y \\ z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \phi_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \qquad (2)$$

where ϕ_i is the emission of the isosurface to which the point belongs. This transformation can be implemented as a scaling by factor 0 in z direction followed by a translation by ϕ_i. The result of this transformation is shown in Figure 2.

Note that if a perspective transformation is used, the above transformation has to be executed *after* the perspective is done. Otherwise perspective foreshortening will change the proportions of the geometry: since brighter surfaces are moved closer to the eye, the perspective transformation will make them appear bigger than dark ones.

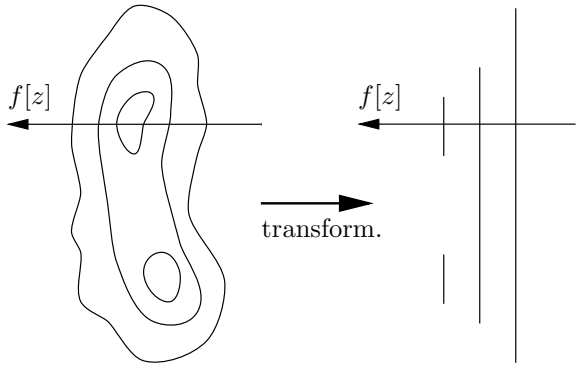


Figure 2: *The result of the geometric transformation for maximum rendering: the z component of each point is set to the emission of the isosurface it belongs to*

Unfortunately in some graphics environments it is not possible to specify additional affine transformations after the perspective. In this case it is still possible to use an orthographic projection instead of perspective, because the orthographic projection preserves proportions (it is affine), and therefore it can take place after the z adjustment.

2.2 Depth Cueing

It is often useful to add additional depth cues to the image in order to improve the 3D impression of the image. One of the simplest depth cues to implement is an intensity ramp that dims parts of the volume farther from the eye. A linear ramp along a projector can be described by

$$\beta[z] = \beta_0 + \beta_m z,$$

with $\beta_0 \geq 0$ and $\beta_m > 0$. The value β_0 is the attenuation at the background image, and β_m is the rate of change of this attenuation per unit length along the projector.

In the case of maximum rendering this ramp should be pre-multiplied onto the flux density before the maximum operation is performed.

$$\Phi[d] = \max(\max_{z \in (0,d]} (\zeta \beta[z] \phi[z]), \beta_0 \Phi[0]) \qquad (3)$$

This way the maximum operator will take depth cueing into account, and will always select the brightest point. Otherwise the point with the highest emission would be selected, even if it was very far away from the image plane, and therefore relatively dark. This anomalous reversal situation is depicted in Figure 3.

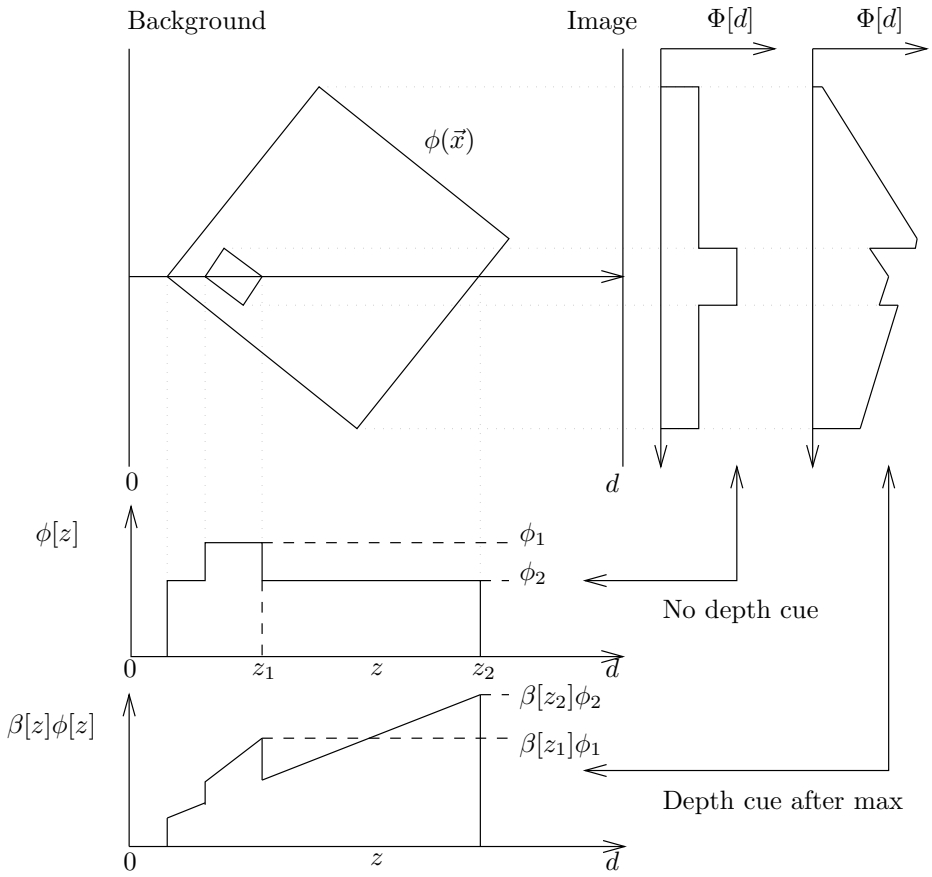


Figure 3: *What can happen when the maximum operation is performed* before *depth cueing: an inversion. Below is shown the function along the diagrammed projector, the second graph showing the effect of depth cueing. To the right are corresponding graphs of the value of* $\Phi[d]$ *for all projectors. This problem can be avoided by applying the maximum operation* after *depth cueing, as described in the text.*

For the discrete case (3) can be simplified to account only for the intersection points with the isosurfaces. Let z_i be the intersection of the projector with isosurface i. If the projector has multiple intersections with the same isosurface, we select the one which is closest to the image plane. Since we defined the maximum operator to choose the closest point in cases where two or more points are equally bright, we can write the above equation as

$$\Phi[d] = \max(\max_i(\zeta\beta[z_i]\phi_i), \beta_0\Phi[0]) \qquad (4)$$

2.3 Transformations for Depth Cueing

In an implementation with depth cueing we would like to be able to control the two degrees of freedom which a linear ramp provides. We can do that by directly adjusting the parameters β_0 and β_m of the linear ramp, or by fixing these parameters and applying a linear function to the z component of each point instead.

For reasons that will become clear later, we choose the latter approach, and apply the depth cue to a shifted and scaled $z_i' = az_i + b$. If we assume that the background is darker than the brightest point along a projector, Equation (4) reduces to

$$\Phi[d] = \zeta(\beta_0 + \beta_m z_i')\phi_i = \zeta(\beta_0 + \beta_m(az_i + b))\phi_i.$$

This can be rearranged into

$$\Phi[d] = \zeta\beta_m\left(\phi_i a z_i + \phi_i(b + \frac{\beta_0}{\beta_m})\right).$$

In other words, the transformation to control the linear ramp can be pushed into the transformation we do before the maximum operation, and only a single maximum comparison is needed on the combined

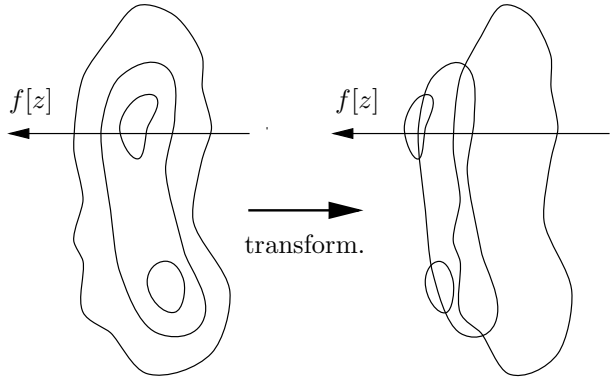


Figure 4: *The result of the geometric transformation including depth cueing: the geometry is scaled and translated so that brighter points are closer to the image plane.*

depth/intensity. Setting

$$\begin{aligned} a_i^* &= \phi_i a \qquad \text{and} \\ b_i^* &= \phi_i(b + \beta_0/\beta_m), \end{aligned}$$

transformation (2) can therefore be redefined as

$$\begin{bmatrix} x \\ y \\ z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a_i^* & b_i^* \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Figure 4 demonstrates the effect of this transformation.

Instead of drawing each isosurface with a color corresponding to its emission, like in Section 2.1, now we simply transform each isosurface with the appropriate matrix, and then draw it in full brightness. The hardware depth cueing of the graphics system we use will take care that surfaces with higher emission will appear brighter on the screen than ones with lower emission. On systems without hardware depth cueing the linear ramp can be applied as a post processing step by reading the z value from the depth buffer and modifying the brightness appropriately.

We now have parameters β_0, a and b to customize the rendering algorithm. If we set $a = 0$, $b = 1$, and the brightness of the background image $\beta_0 = 0$, we have exactly the algorithm described in Section 2.1, without any depth cueing. By varying b, we can adjust the contrast between isosurfaces of different brightness.

The parameter a controls the amount of depth cueing added to the image. It will actually be the ratio of a and b that will give the full tradeoff of depth cueing to intensity. Increasing β_0 will increase the overall brightness of the image.

2.4 Approximate Summation Rendering

As noted in Section 1.4, the discrete version of the summation rendering algorithm has to compute the sum of intensities of the volumes that a projector intersects, weighted by the length of the projector segment that lies within the volume.

Unfortunately there is no simple geometric transformation (affine or projective) which would allow us to correctly compute the total length of projector segments, and therefore the weights. However, if all isosurfaces have roughly the same topological properties, summation rendering can be approximated by setting all weights to the same value, for example $1/n$, where n is the number of isosurfaces. Unfortunately, the approximation might be very coarse if only a small number of isosurfaces are used.

On graphics systems which support *accumulation buffers*, this approximation of summation rendering can be implemented easily. Accumulation buffers provide hardware support for creating a weighted sum of different images. For summation rendering this means that each isosurface has to be drawn into a separate image, using transformations and depth cueing as described above. Then the accumulation buffers are used to add the images together.

3 Implementation

A volume visualization program for regularly gridded data has been implemented using the methods described in this paper. The program uses the 3D graphics library OpenGL [2] to exploit the graphics hardware of different platforms in a portable way. It has been tested on workstations from Silicon Graphics (SGI) and Digital Equipment (DEC).

OpenGL supports both z-buffers and accumulation buffers, and the linear depth cue described in Section 2.2 can be implemented using the linear fog facility which OpenGL provides. OpenGL will apply the linear ramp in software if the graphics hardware does not directly support fog. In addition to the transformations given in this paper, another scaling was introduced in order to normalize the resulting z components to values between 0 and 1. This is the interval on which the OpenGL z-buffer operates.

The user interface allows for three different rendering modes: maximum rendering, approximate summation rendering, and the usual lighted rendering of iso-

surfaces. In the latter mode only the outermost shell will be visible, of course. Three scrollbars provide the user with full control over the rendering parameters β_0, a and b.

The program also allows the user to interactively add or delete isosurfaces, or to temporarily disable individual surfaces. For the creation of the isosurfaces a simple marching cubes algorithm [12] is used.

The resulting volume model can be interactively rotated on the screen, where the performance depends on the complexity of the volume data, the number of isosurfaces, and the graphics hardware.

3.1 Results

Several tests with different data sets were made in order to measure the performance on different graphics platforms. For the first measurement we chose the $64 \times 64 \times 64$ data set HIPIP from the Chapel Hill Volume Rendering Test Date Sets (CHVRTD), and created 11 isosurfaces with 44846 polygons. A second measurement was done with the same data, but with only 6 isosurfaces and 14844 polygons.

Another data set (SAT) contains a $21 \times 21 \times 51$ grid of saturation values from a groundwater simulation of oil contamination [15]. From this data 20 isosurfaces with a total of 16846 polygons were created.

Finally, an MR angiogram of a human brain was used. This data, which can be obtained from the UMDS Image Processing Group, consists of a $256 \times 256 \times 124$ grid, from which 12 isosurfaces with 177594 polygons have been created. Maximum rendering is especially useful for angiograms, because they contain a lot of localized detail which would be obscured by other rendering methods. Figure 5 shows the MR angiogram data set rendered with different parameter settings. Table 1 gives an overview of the performances measured on different platforms.

The figures shown for the Silicon Graphics Onyx with VTX graphics are somewhat inaccurate, because for these high frame rates the time spent synchronizing with the CRT refresh signal is significant. Since the refresh frequency of the CRT was set to 60Hz, and frame rates slightly above 20 have been measured, the rendering of one image takes between 2 and 3 refresh cycles, which corresponds to a frame rate of $20 - 30$ frames per second.

The results for all platforms show that reasonable frame rates can be achieved for maximum rendering, even on relatively low-end graphics acceleration boards. On the other hand the figures also show that approximate summation rendering should only be used on platforms which provide significant hardware support for accumulation buffers. Otherwise the computation of the weighted sum of images dominates the rendering time, and the number of isosurfaces becomes the limiting factor for performance.

	SGI Onyx	SGI Indigo	DEC Alpha
Maximum			
Hipip 1	8.51	1.59	1.63
Hipip 2	>20	4.50	4.26
Angiogram	2.66	0.74	0.39
Sat	>20	4.01	3.90
Summation			
Hipip 1	5.60	0.23	0.047
Hipip 2	12.0	0.41	0.080
Angiogram	1.38	0.071	0.040
Sat	5.48	0.056	0.028

Table 1: *Frame rates for different data sets on different platforms in frames per second. The SGI Onyx is equipped with VTX graphics, the SGI Indigo has a XS-24 board, and the DEC Alpha a PXG Turbo.*

4 Conclusions

In this paper we have presented two interactive volume rendering techniques, maximum rendering and approximate summation rendering. We have shown how discrete versions of these methods that exploit graphics hardware can be implemented using isosurfaces. The number of isosurfaces can be adjusted to provide a wide range of tradeoffs between performance and image quality.

Simple geometric transformations can be used to combine depth cueing with discrete versions of the maximum rendering and summation rendering algorithms. Affine transformations and polygonal models make the use of standard graphics hardware possible, thus allowing for interactive manipulation of viewport and depth cueing parameters.

5 Acknowledgments

We would like to thank Andre Unger and Nathan P. Konrad, University of Waterloo for providing the SAT data set.

The data set HIPIP originates from a simulation done by Louis Noodleman and David Case, Scripps Clinic, La Jolla, California, and can be obtained from the CHVRTD test suite at `omicron.cs.unc.edu`.

The MR angiogram data set is available from the United Medical and Dental Schools (UMDS) Image

Processing Group in London, and can be found at `http://www-ipg.umds.ac.uk/archive/heads.html`.

An earlier implementation of this algorithm was created under ICAR at the Playfair Neuroscience Institute at the Toronto Western Hospital. ICAR is a neurobiological version of CAMRA, a real-time volume visualization system produced by ISG Technologies.

OpenGL is a trademark of Silicon Graphics Inc.

References

[1] Harrison H. Barrett and William Swindell. *Radiological Imaging*, volume 1. Academic Press, 1981.

[2] OpenGL Architecture Review Board. *OpenGL Reference Manual: The Official Reference Document for OpenGL*. Addison-Wesley, 1992.

[3] Lih-Shyang Chen, Gabor T. Herman, R. Anthony Reynolds, and Jayaram K. Udupa. Surface shading in the cuberille environment. *IEEE Computer Graphics and Applications*, 5(12):33–43, December 1985.

[4] H. N. Christiansen and T. W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *ACM Computer Graphics (ACM SIGGRAPH '78 Proceedings)*, 12(3):187–192, August 1978.

[5] H. E. Cline, C. L. Dumoulin, H. R. Hart Jr., W. E. Lorensen, and S. Ludke. 3D reconstruction of the brain from magnetic resonance images using a connectivity algorithm. *Magnetic Resonance Imaging*, 5:345–352, 1987.

[6] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, 22(4):65–74, August 1988.

[7] G. Frieder, D. Gordon, and R. A. Reynolds. Back-to-front display of voxel-based objects. *IEEE Computer Graphics and Applications*, 5(1):52–59, January 1985.

[8] J. T. Kajiya. Ray tracing volume densities. *ACM Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, 18(3):165–174, July 1984.

[9] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. *ACM Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, 23(3):271–280, July 1989.

[10] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[11] Marc Levoy. *Display of Surfaces from Volume Data*. PhD thesis, University of North Carolina at Chapel Hill, Department of Computer Science, May 1989.

[12] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Computer Graphics (ACM SIGGRAPH '87 Proceedings)*, 21(4):163–189, July 1987.

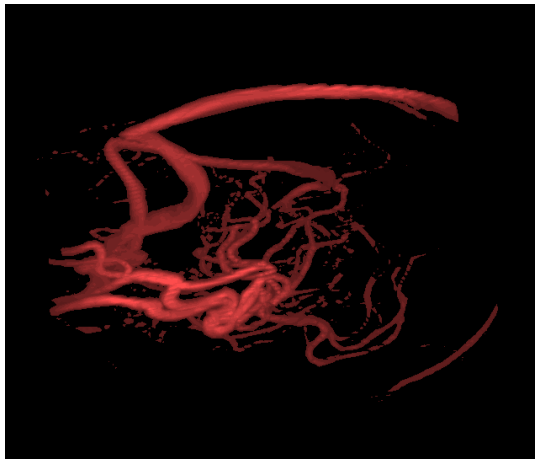
[13] Abraham Mammen. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Computer Graphics and Applications*, 9(4):43–55, July 1989.

[14] Michael McCool. Compact data structures for volume visualization. Master's thesis, University of Toronto, Department of Computer Science, 1991.

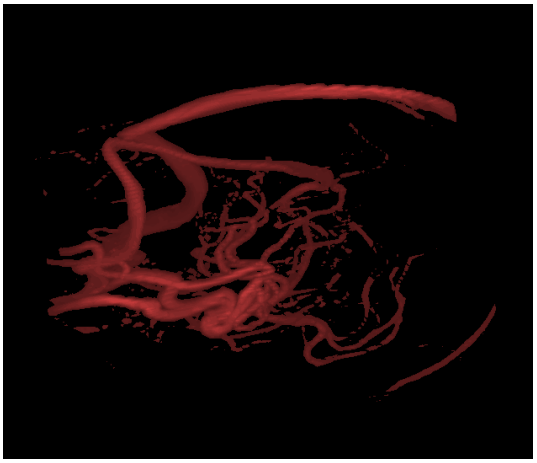
[15] A.J.A. Unger, E.A. Sudicky, and P.A. Forsyth. Mechanisms controlling air sparging for remediation of heteogeneous formations contaminated by dense non-aqueous phase liquids. In *Water Resources Research*, 1995. accepted for publication.

[16] J. Upda, H. Huang, and K. Chuang. Surface and volume rendering in 3D imaging: a comparison. *Journal of Digital Imaging*, 4:159–168, 1991.

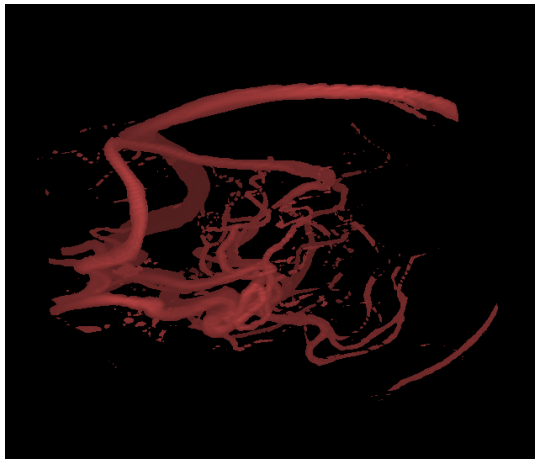
[17] J. Upda and D. Odhner. Shell rendering: Fast volume rendering and analysis of fuzzy surfaces. *Journal of Digital Imaging*, 4:159–168, 1991.



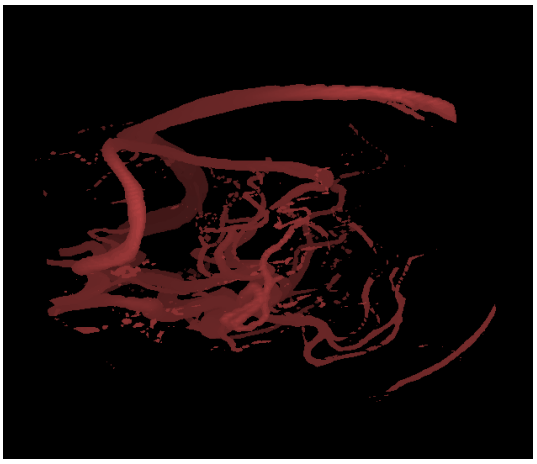
Max. rendering, Contrast: 100%, Depth Cue: 0%



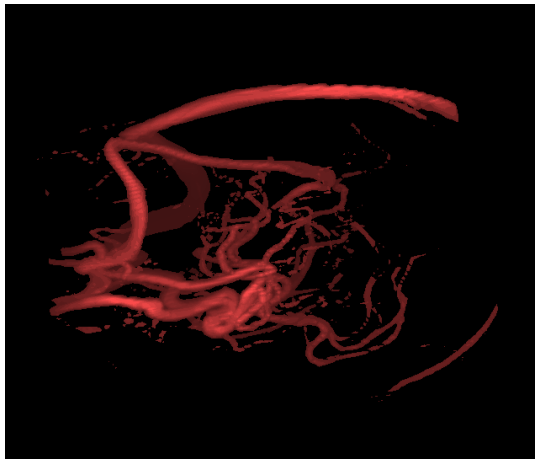
Max. rendering, Contrast: 66%, Depth Cue: 33%



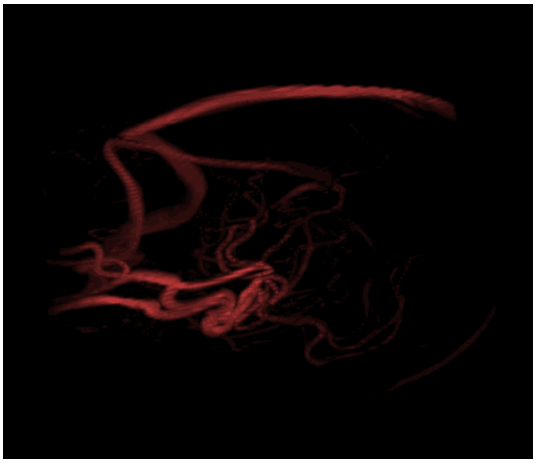
Max. rendering, Contrast: 33%, Depth Cue: 66%



Max. rendering, Contrast: 0%, Depth Cue: 100%



Max. rendering, Contrast: 100%, Depth Cue: 100%



Sum. rendering: Contrast: 100%, Depth Cue: 100%

Figure 5: *Different settings of the rendering parameters. Depth cueing improves the understanding of the 3-dimensional structure, while the contrast reveals fine detail in the data set.*