Survey of Parallel Volume Rendering Algorithms

Craig M. Wittenbrink
Computer Systems Laboratory

HPL-98-49 (R.1)
March, 1998

volume visualization,
splatting,
ray casting,
permutation warping,
taxonomy

Volume rendering is a compute and memory intensive application. Researchers have attempted to use parallel computers to speed up volume rendering for interactive frame rates. Many architectures and approaches have been developed. I developed a taxonomy of volume rendering by forming five categories: algorithm control flow, targeted hardware, application data characteristics, visualization method, and publication specifics. I discuss the current approaches in parallel volume rendering algorithms, and show how different taxonomies from the developed classification provide the ability to compare, contrast, and show potential for future research results.

To be published in the *Proceedings of International Conference on Parallel Distributed Processing Techniques and Applications (PDPTA 98),* Las Vegas, Nevada, July 1998.

Survey of Parallel Volume Rendering Algorithms

Craig M. Wittenbrink
Visual Computing Department
Hewlett-Packard Laboratories
Palo Alto, California

Abstract ***Volume rendering is a compute and memory intensive application. Researchers have attempted to use parallel computers to speed up volume rendering for interactive frame rates. Many architectures and approaches have been developed. I develop a taxonomy of volume rendering by forming five categories: algorithm control flow, targeted hardware, application data characteristics, visualization method, and publication specifics. I discuss the current approaches in parallel volume rendering algorithms, and show how different taxonomies from the developed classification provide the ability to compare, contrast, and show potential for future research results.***

Keywords: volume visualization, splatting, ray casting, taxonomy

1 Introduction

Volume rendering is a method for visualizing three dimensional sampled data, and has come to encompass an ever growing family of approaches. Figure 1 shows a 2D slice through a 256^3 volume of CT data. Figure 2 shows a volume rendering of the same data where only the bone has been segmented out by classification from the gray tissue densities of Figure 1. Because of the computational complexity, typically $O(n^3)$ for an $n \times n \times n$ volume of data, researchers have sought to use parallelism to achieve interactivity. There has been duplication of effort and a lack of comparisons of techniques.

There have been efforts to survey algorithms. Previous studies have used the following approaches to classify volume rendering: intermediate data representation [9, 2], view transform or processing order [9, 2, 23], compute platform [23, 6], top N algorithms [2, 1], top N hardware platforms [6], type of parallelism used [6], and visualization technique [19, 2].

Figure 3 shows eight subvolumes in the object space on the left, and eight subimages in the image space on the right. The view transform or processing order is the order in which the algorithm considers the primitives. Proposed processing orders have been object, hybrid, and image order [2, 23, 9].

These earlier surveys effectively distinguished the approaches, but did not compare or differentiate parallel algorithms. Recently, there have been hundreds of parallel algorithms published. Figure 4 shows a recent abstract search performed in the IEEE Inspec service. The number of papers in the field is growing rapidly.

Yagel [23] took an important step in generalizing the survey of approaches to include the platform. But, even with these classifications, it is difficult to determine what the best approaches might be for a new parallel machine, or what the ideal approaches would be for implementation in a hardware architecture. Because of this, I have developed the following taxonomy of parallel volume rendering approaches. I show specifically how the previous classifications did not disambiguate algorithms, and how my classification clearly does. I explain each characteristic of the taxonomy, discuss the classification of some exist-

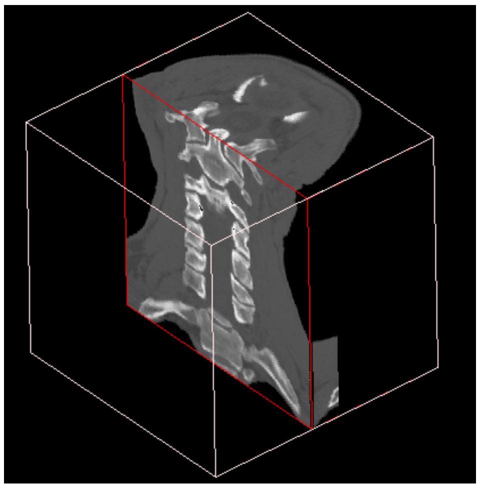
Figure 1: A view of a Spiral CT data set showing only a 2D cut plane through the dataset of a 3D volume of voxels.

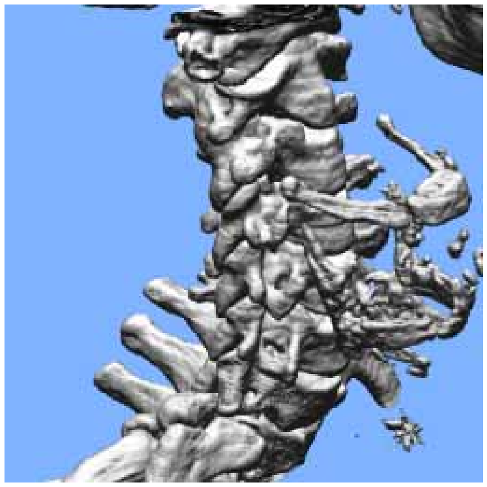
Figure 2: Volume rendering of Spiral CT data human vertebrae shown in Figure 1. Data Courtesy of Dr. Ramani Pichumani, Stanford University.

ing parallel volume rendering approaches, and draw conclusions about the most promising approaches for future research.

2 Classification

To create a classification for parallel volume rendering algorithms, I designate the following orthogonal feature groupings: *algorithm control flow, targeted hardware, application data characteristics, visualization method, and publication specifics.* I describe each of these groupings in more detail, and also develop new taxonomies from them. Examples from the literature of parallel volume visualization helps to evaluate the usefulness of my classification.

Algorithm control flow. I designate two main classifications for algorithm control flow, view reconstruction and outer loop iteration data space. The following four methods are used for calculating the view reconstruction in parallel volume visualization: i) backward (B), ii) forward (F), iii) multipass forward (MF), and iv) Fourier ($\mathcal{F}$).

The outer loop iteration data space is defined as whether the program's outer loop iterates through the i) object (O) or ii) image space (I). See Figure 3. For graphics and volume rendering, object space is the source, and image space is the destination. More details on this important classification are given in Section 2.1.

Targeted hardware. I designate the following five hardware platforms: i) graphics (G), ii) volume rendering (V), iii) parallel shared address space (PS), iv) parallel distributed address space (PD), and v) distributed (D). More explanation for targeted hardware is found in Section 2.2.

Application data characteristics. Input data topologies types include: rectilinear (R), curvilinear (C), and unstructured (U). Input data types visualized include: scalar, vector, tensor. Data units include: pressure, temperature, humidity, etc. And finally the voxel format specifies the number of bits for the voxel and alpha, and whether the voxel is color or gray scale.

Visualization method. The visualization method includes very many different parameters such as shading, the reconstruction filter, gradient, transmission model, and data classification.

Publication specifics. In published studies, there may be a collection of result data. I use: date, number of processors, volume size, prototype machine, asymptotic complexity analy-

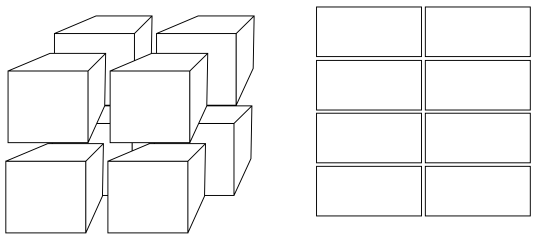

Figure 3: To achieve parallelism object space parallism can be done by splitting the 3D object space into separate regions and image space parallelism can be achieved by splitting the 2D screen space into regions.

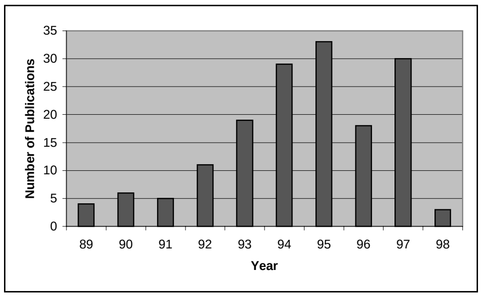

Figure 4: IEEE Inspec search on parallel and (volume rendering or volume visualization) 1989 to 1998.

sis, and performance (MVoxels/second, scalability).

Next I further characterize the specifics of these groups with concrete examples, especially the algorithm control flow in Section 2.1 and targeted hardware in Section 2.2.

2.1 Algorithm Control Flow

The outer loop iteration data space has been called the processing order, or data flow. In parallel volume rendering algorithms, the processing order designation as object space, image space, or hybrid is not specific enough. For example, four high performance parallel algorithms use the hybrid designation [2, 9, 23] of processing: Lacroute's Shear Warp [8] (Method A), Ma et al.'s Binary Swap technique [10] (Method B), my permutation warping [22] (Method C), and Yagel et al. and Schroder et al.'s rendering using templates or by line drawing [24, 16] (Method D). See Table 1. I separate out these approaches more clearly by using the view reconstruction and the outer loop iteration data space. See Table 2. For the majority of image and graphics processing a transform based paradigm is used to develop, validate, and analyze algorithms. In order to clarify the differences between hybrid algorithms, I have separated the view reconstruction filter from the processing order.

Figure 5 shows algorithms represented as a directed graph $G = (V, E)$, where each vertex shows a data representation $v \in V$ and each arc shows a transformation, $e \in E$. This transform graph is fundamental to image processing, volume rendering, and mathematical morphology. The edges represent alternative ways to calculate the same result. The view reconstruction may be defined as backwards, forward, multipass forwards, or Fourier. A backwards viewing reconstruction is where the inverse view transform, T^{-1}, is used to calculate voxel locations from pixel locations. Figure 6 shows forwards and backwards resampling. Using a backwards transform in volume rendering is most commonly called ray casting. A forward viewing reconstruction is where the view transform, T, is used to calculate pixel locations from voxel locations. Using a forward transform in volume rendering is most commonly called splatting [18]. A multipass forward viewing reconstruction is where the view transform is a decomposition into multiple transforms, $T = T_1 T_2 \ldots T_n$, that are used to calculate multiple intermediate voxel locations. There are multiple resampling steps to compute view transformed volume data, and efficiency results from regular memory accesses. A Fourier viewing reconstruction is where the view transform is achieved by using the Fourier space as an intermediate space. This technique relies upon the Fourier slice theorem, the relation of a line integral to the Fourier transform of a single projection. There are additional factors affecting the view reconstruction method, such as the type of transform supported, and any restrictions.

Processing Order	Object	Image	Hybrid
Algorithm			A [8],B [10], C [22],D [24, 16]

Table 1: Elvin, Yagel, Levoy taxonomy groups four algorithms into the same category.

View Reconstruction	Multipass Forward (MF)	Multipass Forward (MF)	Backward (B)	Backward (B)
Outer Loop	Object (O)	Image (I)	Object (O)	Image (I)
Algorithm	A [8]	D [24, 16]	C [22]	B [10]

Table 2: Proposed algorithm control flow grouping separates four algorithms into different categories.

Once the view reconstruction is specified, the processing or outer loop iteration data space can be determined. Instead of using processing order, the outer loop of iteration is used to differentiate algorithms. This is necessary for parallel algorithms, which ideally use both object space and image space parallelism for maximum performance. Figure 6 shows filtering. Filtering can be calculated by convolving a spatial filter with samples to create an output sample. This is shown by the backward edge in the graph. The same output can also be calculated by taking each input and convolving them with a filter that is summed in the output. This is the forward edge in the graph. The backward operation convolves sequence A with B to get sequence C, $C = A \text{Operator} B$. Psuedo code for outer loop destination processing (image space, I) is

```
Initialize $C$
For $c \in C$
    For $k \in B \cap A$
        $C[c] = \texttt{Operator}(C[c], A[c,k], B[c,k])$
```

The outer loop source (object space, O) processing is

```
Initialize $C$
For $k \in B \cap A$
    For $c \in C$
        $C[c] = \texttt{Operator}(C[c], A[c,k], B[c,k])$
```

The outer loop defines forwards or backwards calculation and each direction may calculate the same result. Also one can calculate the filtered image by first transforming into the frequency domain, multiplying by a filter, and then transforming into the spatial domain. This is the Fourier edge transitions in Figure 5. Examples of operations that can be calculated by changing the ordering of the processing loops include matrix product, gray scale dilation, convolution, and grey scale erosion. Below I define the example operators for matrix multiply (two dimensional $C = (i, j)$), Convolution, Grey scale dilation, and Grey scale erosion.

$$\begin{aligned}
&\text{Op}(C[c], A[c,k], B[c,k]) \\
&\qquad = C[i,j] + A[i,k] \times B[k,j] \\
&\qquad = C[i] + A[i-k] \times B[k], \\
&\qquad = \max(C[i], A[i-k] + B[k]), \\
&\qquad = \min(C[i], A[i+k] - B[k]).
\end{aligned}$$

Other examples include spatial warping, volume rendering, ray tracing, and radiosity. Examing an algorithm at this level of detail allows quick classification of approaches, identification of untried approaches, and analysis of the most effective approach for the application at hand.

2.2 Targeted Hardware

The targeted hardware designations: include the five platforms: G, V, PS, PD, and D. The graphics hardware (G) designation means special purpose graphics hardware, or hardware that is optimized for computer graphics rendering. The volume rendering hardware (V) designation means special purpose volume rendering hardware, or hardware that is optimized

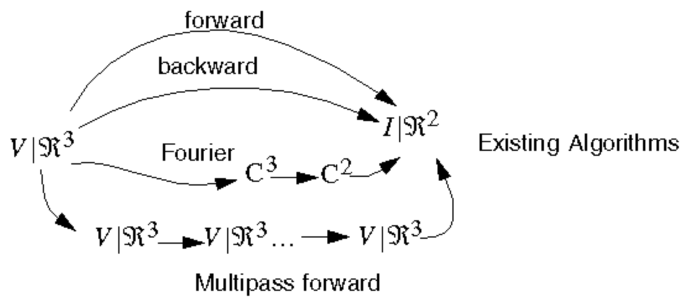

Figure 5: Transition graph from data spaces to final rendering in image space.

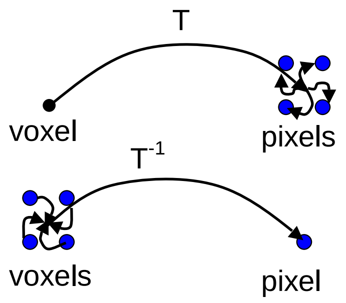

Figure 6: Forward resampling or backwards resampling.

for volumetric rendering. The parallel hardware designation means general purpose parallel hardware with a shared address space (PS) or distributed address space (PD), or hardware that is designed for tightly coupled general purpose parallel processing. The distributed hardware (D) designation means general purpose distributed hardware, or hardware that is designed for loosely coupled processing such as networks of workstations.

There are many other designations for characterizing the specific hardware platform, which are used as subclasses of these four categories. Methods include parallelism type from Flynn's classification (SIMD, MIMD, etc.), theoretical model of parallel computing used for algorithm analysis (such as the parallel random access machine, PRAM, type CRCW, EREW, CREW), and the type of memory addressing supported (distributed address space or shared address space).

To develop taxonomies from this multidimensional classification requires selecting the most appropriate axes to investigate. The algorithm control flow and targeted hardware are focussed on in the next section to investigate the variety of parallel volume rendering algorithms now available.

3 Parallel Taxonomy

To create a parallel taxonomy, one chooses features from the classification. Table 3 is one possible classification using the primary two classifications: control flow and targeted hardware. I have chosen to place the view reconstruction (4 choices) and outerloop (2 choices) as rows, and the targeted hardware (5 choices) as columns. This creates 40 unique combinations, not all of which have a published result. Other interesting combinations can be created, by choosing the parameters that are most relevant for the variants that are being compared. In order to see the separation of techniques, I have also listed alphabetically a selection of parallel volume rendering results with targeted hardware, reconstruction, outerloop, and data grid type in Table 4.

Certain approaches have not been examined, or others have had very much emphasis. Backwards view reconstruction image space outer loop on general parallel hardware has had many publications for example [10, 11]. And, of course, not all combinations appear to be meaningful, for example a forward view reconstruction with an outer loop in the image space has not been discovered. An additional interesting discrimination of the taxonomy is that it differentiates between the view reconstruction of an algorithm, and the outer loop which is used for determining control flow. Permutation warping that we have worked on is shown to calculate a backwards view reconstruction, while using the efficiencies of iterating in the object space. And, in a technique known as 3D texture mapping [17], a backwards reconstruction is used in texture mapping, while polygons are drawn in object space to invoke the texture mapping for an object space iteration. Making such view and iteration space differ-

		target hardware				
view rec.	o. loop	graphics (G)	volume (V)	parallel shared add. (PS)	parallel distributed add. (PD)	distributed (D)
forward (F)	Object	[21]	[7]	[20]	[12]	
	Image					
multipass forward (MF)	Object	[1]		[8] A	[15]	
	Image		[14]	[16, 24] D		
backward (B)	Object	[17]			[22] C	
	Image	[25]	[4]	[13]	[10, 11] B	[3]
Fourier ($\mathcal{F}$)	Object					[5]
	Image					

Table 3: Taxonomy of parallel volume rendering using view reconstruction and outer loop (rows) versus targeted hardware (columns)

ences explicit is a valuable insight of this taxonomy. Fourier volume rendering has been only minimally investigated for parallel implementation [5].

4 Conclusions

I presented a comprehensive classification for volume rendering, that can be used to develop taxonomies for the study of the field. I showed one important taxonomy that differentiates approaches that could not be differentiated under other classifications. This was possible because of the use of orthogonal categories, and the consideration of the sequential nature of parallel algorithms. The algorithm control flow features are very important to understanding: view reconstruction and outer loop. The view reconstruction filter was classified by a graph, with data representations and tranformations arcs: backwards (B), forwards (F), multipass forwards (MF), and Fourier ($\mathcal{F}$)). The outer loop determines whether, in the sequential control loop of processing, iterations are done in object (O) or image (I) space.

The use of an example taxonomy showed how – Lacroute's Shear Warp [8], Ma et al.'s Binary Swap technique [10], my permutation warping [22], and Yagel et al. and Schroder et al.'s rendering using templates or by line drawing [24, 16]– can be distinguished. With prior classifications, all four of these algorithms would have been in one category, hybrid. Interestingly, three of these algorithms, all use the same shear warp decomposition of the viewing transformation [8, 24, 16], and combine efficient calculation of interpolation weights for the intermediate sheared volume with the efficiencies of regular access to the aligned base plane intermediate image. Further work needs to be done to clarify if these are actually the same algorithms, implemented on different machines, and described somewhat differently.

The use of a taxonomy helps those in the field find a wide perspective. Such a taxonomy can point out areas that have not been explored, and makes it easier to compare one's work. The work is also valuable for those entering the field to get a fast introduction with the right context. I have denoted the important parameters for a more general classification of parallel volume rendering. I have shown one possible table of approaches. There are hundreds of published algorithms, and hopefully this classification will help researchers to understand and improve upon them.

References

[1] R. A. Drebin, L. Carpenter, and P. Han-

pub.	description	hardware	rec.	outer loop	Data
[1]	Shearing	G/SIMD	MF	O	R
[3]	Ray Casting	D/MIMD	B	I	U
[4]	VIRIM	V	B	I	R
[5]	Fourier	D	$\mathcal{F}$	O	R
[7]		V	F	O	R
[8]	Shear Warp	PS/MIMD	MF	O	R
[10]	Binary Swap	D/MIMD	B	I	R
[11]	Ray casting	PD/MIMD	B	I	R
[13]	Ray Casting	PS/MIMD	B	I	R
[14]	Cube4	V/SIMD	MF	I	R
[15]	Shearing	PD/SIMD	MF	O	R
[16]	Line Drawing	PD/SIMD	MF	O	R
[17]	3D Texture	G/MIMD	B	O	R
[18]	Splatting	G/SIMD	F	O	R
[20]	Cell Projection	PS/MIMD	F	O	U
[22]	Permutation Warping	PD/SIMD	B	O	R
[21]	Projected Tetrahedra	G/MIMD/SIMD	F	O	U
[24]	Template Based	?	MF	O	R
[25]	Ray Casting	G/MIMD/SIMD	B	I	R

Table 4: Some of the algorithms considered in development of taxonomy with key characteristics listed

rahan. Volume rendering. In *Computer Graphics*, pages 65–74, Aug. 1988.

[2] T. T. Elvins. A survey of algorithms for volume visualization. *Computer Graphics (ACM Siggraph Quarterly)*, 26(3):194–201, Aug 1992.

[3] C. Giertsen and J. Petersen. Parallel volume rendering on a network of workstations. *IEEE Computer Graphics and Applications*, 13(6):16–23, 1993.

[4] T. Gunther, C. Poliwoda, C. Reinhard, J. Hesser, R. Manner, H.-P. Meinzer, and H.-J. Baur. Virim: A massively parallel processor for real-time volume visualization in medicine. In *Proceedings of 9th Eurographics Hardware Workshop*, pages 103–108, Oslo, Norway, Sept. 1994.

[5] D. Junhui and T. Zesheng. Parallel frequency domain volume rendering on workstation cluster. *J. Tsinghua Univ., Sci. Technol. (China)*, 37(9):64–68, Sept. 1997.

[6] A. Kaufman, R. Bakalash, D. Cohen, and R. Yagel. A survey of architectures for volume rendering. *IEEE Engineering In Medicine And Biology*, pages 18–23, Dec. 1990.

[7] A. Krikelis. A modular massively parallel processor for volumetric visualization processing. In *Proceedings of the Workshop on High Performance Computing for Computer Graphics and Visualisation*, pages 101–124, Swansea, UK, July 1995. Springer.

[8] P. Lacroute. Analysis of a parallel volume rendering system based on the shear-warp factorization. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):218–231, Sept. 1996.

[9] M. Levoy. A taxonomy of volume visualization algorithms and architectures. In *Introduction to Volume Visualization*,

pages 6–12. ACM SIGGRAPH, Las Vegas, NV, Jul./Aug. 1991.

[10] K. L. Ma, J. Painter, C. D. Hansen, and M. F. Krogh. Parallel volume rendering using binary-swap compositing. *IEEE Computer Graphics and Applications*, 14(4):59–67, 1994.

[11] C. Montani, R. Perego, and R. Scopigno. Parallel volume visualization on a hypercube architecture. In *Proceedings of 1992 Workshop on Volume Visualization*, pages 9–16, Boston, MA, Oct. 1992.

[12] U. Neumann. Parallel volume rendering algorithm performance on mesh connected multicomputers. In *Proceedings on the Parallel Rendering Symposium*, pages 97–104, San Jose, CA, Oct. 1993.

[13] J. Nieh and M. Levoy. Volume rendering on scalable shared-memory MIMD architectures. In *Proceedings of 1992 Workshop on Volume Visualization*, pages 17–24, Oct. 1992.

[14] H. Pfister and A. Kaufman. Cube-4-a scalable architecture for real-time volume rendering. In *Proceedings 1996 Symposium on Volume Visualization*, pages 47–54, San Francisco, CA, Oct. 1996.

[15] P. Schröder and J. B. Salem. Fast rotation of volume data on data parallel architectures. In *Proceedings IEEE Visualization*, pages 50–57, San Diego, CA, Oct. 1991.

[16] P. Schröder and G. Stoll. Data parallel volume rendering as line drawing. In *Proceedings of 1992 Workshop on Volume Visualization*, pages 25–32, Boston, MA, Oct. 1992.

[17] A. Van Gelder and K. Kim. Direct volume rendering with shading via 3D textures. In *ACM/IEEE Symposium on Volume Visualization*, pages 23–30, San Francisco, CA, October 1996.

[18] L. Westover. Interactive volume rendering. In *Volume Visualization Workshop*, pages 9–16, Chapel Hill, NC, May 1989. Computer Science.

[19] J. Wilhelms. Decisions in volume rendering. In *State of the Art in Volume Visualization*, volume 8, pages I.1–I.11. ACM SIGGRAPH, Las Vegas, NV, Jul./Aug. 1991.

[20] J. Wilhelms, A. V. Gelder, P. Tarantino, and J. Gibbs. Hierarchical and parallelizable direct volume rendering for irregular and multiple grids. In *Proceedings of Visualization*, pages 57–64, San Francisco, CA, Oct. 1996.

[21] C. M. Wittenbrink. Irregular grid volume rendering with composition networks. In *Proceedings of IS&T/SPIE Visual Data Exploration and Analysis V*, page In press, San Jose, CA, Jan. 1998. SPIE.

[22] C. M. Wittenbrink and A. K. Somani. Time and space optimal parallel volume rendering using permutation warping. *Journal of Parallel and Distributed Computing*, 46(2):148–164, Nov. 1997.

[23] R. Yagel. Towards real time volume rendering. In *Proceedings of GRAPH-ICON*, volume 1, pages 230–241, Saint-Petersburg, Russia, July 1996.

[24] R. Yagel and A. Kaufman. Template-based volume viewing. *Comput. Graph. Forum (Netherlands)*, 11(3):C153–C167, Sept. 1992.

[25] T. S. Yoo, U. Neumann, H. Fuchs, S. M. Pizer, T. Cullip, J. Rhoades, and R. Whitaker. Achieving direct volume visualization with interactive semantic region selection. In *Proceedings IEEE Visualization*, pages 58–65, San Diego, CA, Oct. 1991.