

Voxels versus Polygons: A Comparative Approach for Volume Graphics*

Dirk Bartz and Michael Meißner

WSI/GRIS, University of Tübingen,

Auf der Morgenstelle 10/C9,

D72076 Tübingen, Germany

Email: {bartz,meissner}@gris.uni-tuebingen.de

Abstract

The decision to use either the Direct Volume Rendering paradigm or the Indirect Volume Rendering paradigm to visualize a volume dataset is a topical question in Volume Graphics. Unfortunately, it seems that this question has not been sufficiently addressed so far and is not easy to answer.

In this paper, we discuss some of the advantages and disadvantages of one candidate for each of these paradigms; Ray Casting for Direct Volume Rendering and Marching Cubes for Indirect Volume Rendering. The discussion is based on two cartesian grid scalar data fields, a CT scan of a lobster – immersed in resin, and a MRI scan of a human head. These datasets have interesting properties which show different features using different Volume Rendering techniques. Two measurements are considered for our discussion; visual quality and the consumption of resources, such as time and memory.

Keywords: Direct Volume Rendering, Indirect Volume Rendering Ray Casting, Marching Cubes.

1 Introduction

Different Direct Volume Rendering techniques (DVR) are introduced in the literature. The most important is the Ray Casting approach [10, 3], the forward projection approach [2, 13], the splatting algorithm [12], and the 3D texture mapping based projection technique [1]. In this paper, we focus on Ray Casting, since it is so far the only approach which enables adjustable sampling in all dimensions, perspective projections, and gray-level gradients. Although there are several improvements for splatting [7, 6] and for texture mapping-based volume rendering [11], the overall quality and generality does still not match Ray Casting (RC).

So far, RC approaches are mostly software based. In the near future, Ray Casting hardware architectures like Mitsubishi's VolumePro [8] or Vizard [5] will be publicly available, thus decreasing the reconstruction and rendering time.

Generally, a volume dataset is viewed from a viewpoint through a viewplane. Starting from the viewpoint, Ray Casting casts rays for each pixel of the viewplane through the volume. Within the volume, samples are taken by trilinear interpolation from the voxel values of the volume. Each sample is classified using transfer functions

*To appear in the Volume Graphics'99 proceedings

for R, G, B, and α . Thereafter, the samples are shaded and composited. This is done by accumulating samples along the ray corresponding to the line integral. Reconstruction and rendering is performed in one single pass. Hence, each change of the view or of the transfer functions requires a complete re-computation. Furthermore, increasing the size of the viewplane results in a similar increase in the number of rays cast through the volume.

In contrast to the DVR paradigm, there is one dominating technique for Indirect Volume Rendering (IVR), the Marching Cubes algorithm (MC)[4]. Using this algorithm, each cell of eight neighboring voxels is visited. The voxels of the cell are classified as inside, outside, or intersecting with the isosurface. All intersecting, or contributing cells are further examined for the reconstruction of the isosurface. There are 256 cases of combination if the voxels of a contributing cell are below (outside) or above (inside) the isovalue (isosurface). Using a table of these cases, up to five triangles per cell can be generated and shaded by using linear interpolation and central difference gradients for the approximation of the vertex normal. Finally, the reconstructed polygonal isosurface can be rendered using common polygon graphics hardware. In contrast to the RC approach, the reconstruction is considered as a pre-process of the actual rendering. The re-computation of the isosurface is only required after a change of the isovalue. Furthermore, the polygonal rendering is usually not fill-limited, thus the rendering time is not significantly increased if the viewplane is enlarged.

The scope of this paper covers the evaluation of Ray Casting as a DVR technique and Marching Cubes as a IVR technique. We consider aspects of visual quality aspects, such as volumetric representation and isosurface extraction (Section 3.1), and resource consumption, such as memory and time in terms of theoretical complexity and actual usage (Section 3.2). The basis of this evaluation are the two volume datasets, which are introduced in Section 2.

2 Experiments

In this Section, we outline the performed experiments. Specifically, we describe the setup used (datasets and viewing parameter) and the algorithms used. Measurements were performed on a R10000 @ 250 MHz SGI Octane with MXE graphics.

2.1 Setup

We performed our experiments on two different volume datasets, each with eight bit voxel data. The first dataset is a CT scan of a lobster of resolution $301 \times 324 \times 38$ – with an anisotropic spacing of 1.0, 1.0, 1.5 –, immersed in a cylinder of resin. Three different materials can be classified; resin, meat and shell of the lobster. An image of the dataset is given in Figure 1(a). The second dataset is a CISS-reconstruction of a T2-weight MRI scan of a human head of resolution $258 \times 258 \times 126$ – with an isotropic spacing of 0.9, 0.9, 0.9. Head tissue and brain liquor in the ventricular

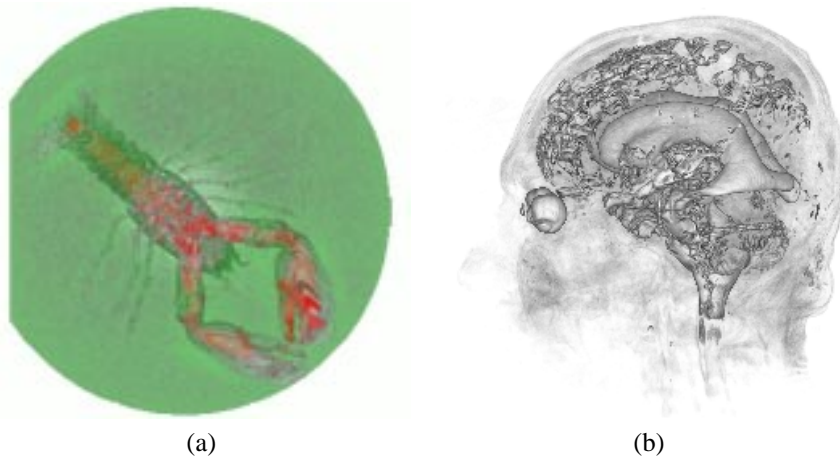


Figure 1: Datasets: (a) CT scan of a lobster (enhanced for b/w printing), (b) MRI scan of a human head.

system¹ of the MRI head are easy to classify, due to the good contrast of the CISS/T2-MRI-reconstruction. An image of the dataset is given in Figure 1(b).

Both datasets have been visualized using RC and MC. For both rendering algorithms, we used a unified environment including one directional light source, similar material properties, and parallel projection for visualization. We chose a viewport of size 324×324 for the CT Lobster dataset, and a viewport of size 258×258 for the MRI Head.

The lobster dataset contains three different materials, as shown in Figure 2(a). We have chosen red for meat, white for shell, green for resin, and the opacity function selected for RC is shown in Figure 2(b). Correspondingly, isovalues, color values, and opacity for the materials have been selected for the MC algorithm as depicted in Table 1.

We assign different opacities for resin using MC and RC (see Figure 2 and Table 1). This is necessary to achieve somewhat similar visual results because the MC algorithm only extracts one layer of triangles (with some exceptions at the center of the cylinder). Therefore, the contribution of the resin is much smaller than the multiple samples taken within the resin using the RC approach. We improve the visual impact of the resin for MC by using a larger opacity for the reconstructed isosurface of the resin.

For the MRI head dataset, the assignment of voxels to materials is shown in Figure 3(a). White has been selected as color for all voxels in the dataset and the opacity function selected for RC is depicted in Figure 3(b). Likewise, the isosurface values, the corresponding material properties, and the opacity function for the MC algorithm are shown in Table 1.

¹The ventricular system of the human brain is responsible for the production and resorption of brain liquor.

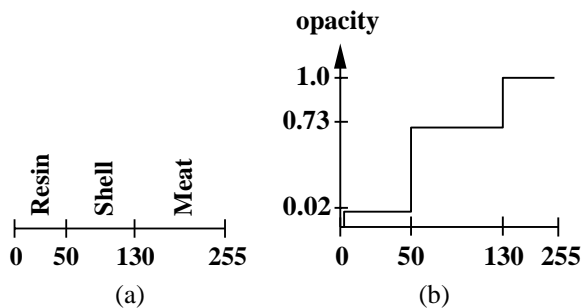


Figure 2: CT Lobster dataset: (a) Material assignment depending on voxel values (b) Opacity function of the voxels for Ray Casting.

Materials	Isovalue	Color (RGB)	Opacity
CT Lobster Dataset			
Resin (R)	2	0.0, 1.0, 0.0	0.13
Shell (S)	50	0.8, 0.8, 0.8	0.27
Meat (M)	130	1.0, 0.1, 0.1	1.0
MRI Head Dataset			
Tissue (T)	30	1.0, 1.0, 1.0	0.02
Liquor (L)	70	1.0, 1.0, 1.0	1.0

Table 1: Overview of dataset material properties for Marching Cubes.

2.2 Two Algorithms for Volume Graphics

Indirect Volume Rendering

For the reconstruction of the isosurfaces, we use an adapted version of the MC algorithms implemented in VTK [9]. All cells of the dataset are visited and classified if they are contributing cells. The gradients are only computed for these contributing cells. In order to prevent ambiguities, a full 256 case set is used. The generated vertices are checked for multiple generation and stored in a bucketsort-like datastructure. This reduces the consumed memory significantly below the upper bound estimated in Section 3.2. Finally, triangle strips or triangle fans were not generated.

Rendering time of polygonal isosurfaces depends very much on the used graphics subsystem. Therefore, we only consider the reconstruction time of the MC algorithm. The rendering is performed using OpenInventor on a SGI Octane with MXE graphics. Depending on the rendering complexity of the extracted isosurfaces (number of polygons and opacity), the actual rendering ranged from approximately 0.5 seconds to approximately five seconds for the MRI Head dataset (similar for the CT Lobster dataset).

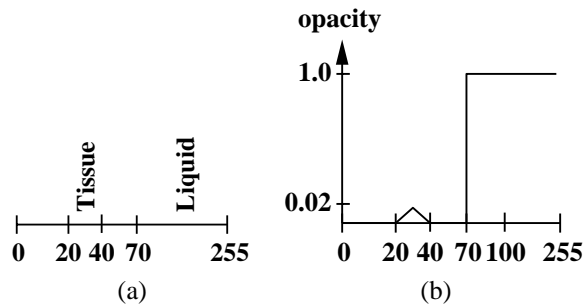


Figure 3: MRI Head dataset: (a) Material assignment depending on voxel values (b) Opacity function of the voxels for Ray Casting.

Direct Volume Rendering

There are several techniques to accelerate RC or MC. These are mostly methods which require some pre-processing like hierarchical datastructures or rough object approximations. For the Ray Casting algorithm, we only exploit early ray termination as the basic acceleration technique, where the ray is not processed any further, once the pixel opacity has accumulated to 98%, which means that there is practically no possible further contribution to this pixel. Furthermore, only the gradients of the contributing samples (opacity is larger than zero) are computed and shaded. In contrast to rendering of MC-generated isosurfaces, RC only depends on the processing performance of the used computer. Furthermore, reconstruction and rendering cannot be distinguished using RC, since it is a single pass algorithm. Therefore, these timings are inherently included in the measurements.

3 Triangles versus Samples

3.1 Visual Quality

RC is commonly considered as a technique for directly visualizing volumetric data, while the MC algorithm extracts isosurfaces from volumetric data which can then be displayed using commonly available polygon graphics hardware. The properties of both algorithms seem to be very obvious but as it can be seen in Section 3.2, the impact to resource consumption can be quite severe. To enable a proper understanding of the differences, it is necessary to have a closer look at the features of both techniques with respect to Volume Graphics. In the following, we will discuss the two approaches and highlight their features in order to describe their strengths and weaknesses. Hereby we will try to answer the following questions:

1. How accurate can smallest structures be displayed?
2. How good is the 3D understanding from generated images?

3. How good can depth be understood from generated images?

Display of Volumetric Information

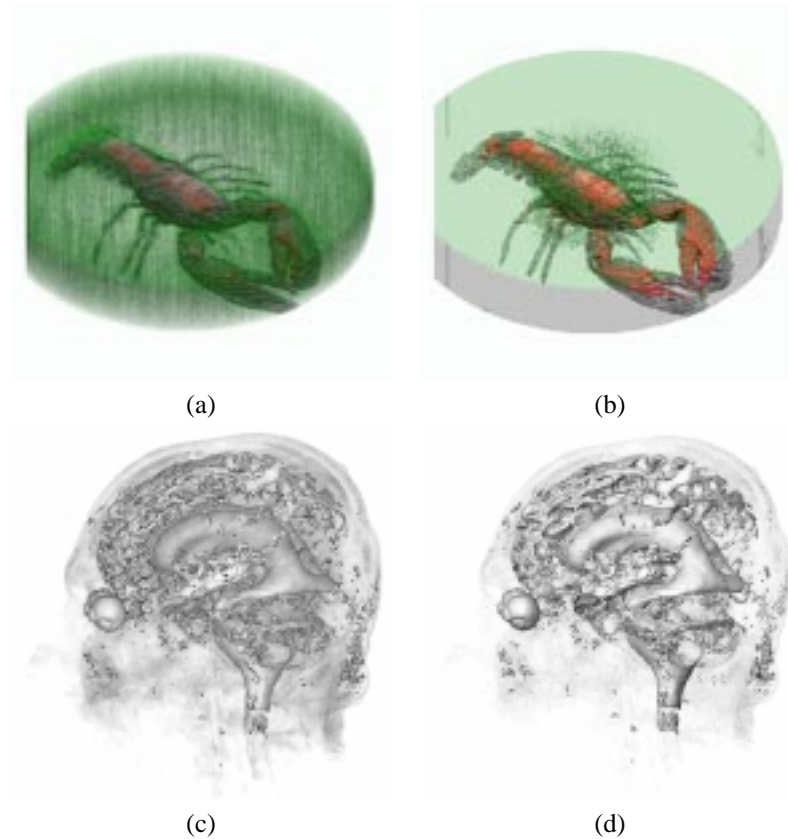


Figure 4: (a) Lobster dataset rendered using RC, classifying resin, shell, and meat. (b) same as (a) using MC. (c) Head dataset rendered using RC, classifying tissue and liquor. (d) same but still, the position of the lobster within the resin can be conceived as (c) using MC.

As mentioned earlier, the MC algorithm extracts isosurfaces from volumetric data. Hence, real thickness, or volumetric dimension of an object, cannot be represented. This is inherent to the isosurface paradigm which is designed to extract infinite thin isosurfaces from volumetric data. As an example, images generated by looking onto an object with different thickness will always look alike. Therefore, the depth of highly transparent objects and the position of objects within highly transparent materials cannot be determined without rotating the object, see Figure 4(b).

In contrast, RC samples volumetric data along rays and accumulates the obtained information. Therefore, the thickness of a material is represented, since absorption

and emission of light are calculated along the ray. The thickness of the material is therefore noticeable by the accumulated color of the individual pixels. Hence, cloudy objects like fog, which do not necessarily have a surface, can be displayed in a very realistic way, but is not possible using MC. This effect can be seen in Figure 4(a), where the resin is rendered as a cloudy (transparent) object but still, the position of the lobster within the resin can be conceived.

Display of Isosurfaces

When it comes to displaying real surfaces within volumetric data, the MC algorithm is capable of precisely extracting a surface which is represented by a certain isovalue. As long as objects do have a surface, specifiable by an isovalue, MC extracts the accurate isosurface. However, the surface of an object might not be specifiable by a single isosurface. This very much depends on the volume dataset, i.e., the overall surface of the object resin – including the surface inbetween lobster and resin – can only be captured applying two isovalues. For MC in general, the number of isosurfaces which can be displayed, mainly depends on the memory limitations and on the polygonal render performance of the graphics subsystem. Hence, MC is capable of extracting smallest structures preserving their nature, as long as the structures can be specified by isovalue(s).

Using RC, volumetric data is sampled along rays and the samples are interpreted using transfer functions. These are used to assign a quadruple (R, G, B, α) which is used to color the sample and blend it corresponding to the volume rendering line integral. In contrast to MC, RC is not dealing with specific isovalue(s), but with the classification of all generated samples using the transfer functions. The sampling rate is therefore very important to achieve sufficient image quality and to prevent aliasing artifacts². As a result of the Ray Casting approach, structures can appear smaller since the samples generated at the border of a structure might be classified as *outside*. Only the rays sampling the structure *inside* will detect the structure. This can be seen in Figure 5(a) where the legs and antennas of the lobster are not as well defined than in Figure 5(b).

Furthermore, it is obvious that due to this effect very small structures which are just one voxel wide might be missed depending on sampling and classification. Therefore, classification is the most crucial part of any DVR. Better results can be achieved using already segmented data or applying higher sampling rates, whereas the images in this paper were generated rendering the original dataset without any oversampling.

3.2 Quantitive Comparison

In this Section, we describe our measurements and complexity considerations of the two different methods, Ray Casting (RC) as a DVR technique, and Marching Cubes (MC) as an IVR technique. In particular, we examine memory and time consumption (see Tables 2 and 3).

²Techniques to adapt Ray Casting for accurately computing isosurfaces are available. However, we only examine the “standard” approach of Ray Casting.

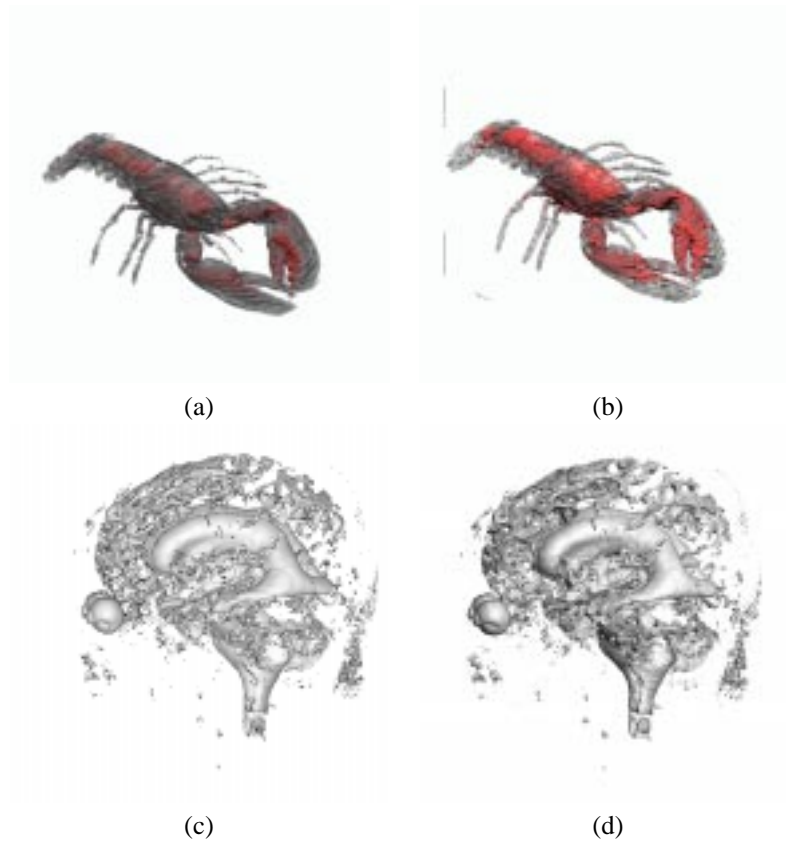


Figure 5: (a) Lobster dataset rendered using RC, classifying shell and meat. (b) same as (a) using MC (c) Head dataset rendered using RC, classifying liquor only. (d) same as (c) using MC.

Memory Consumption

Memory consumption of DVR applications is limited to the actual representation of the volume. Some acceleration structures like octrees or BSP-trees do need significant additional amounts of memory. However, our investigated RC algorithm only uses the standard representation as a 3D array of voxel values. Therefore, the memory consumption is limited to the number of voxels. Considering eight bits per voxel, we use approximately n^3 bytes for the volume representation – where n is the number of voxels in each dimension in a cubic volume –, four times 256 bytes (eight bits per voxel) for the (R, G, B, α) transfer functions, and $x * y * 4$ bytes for a 24+8 bits/pixel viewplane. For reasonably large volumes – which is the case for most applications –, the 3D voxel array will dominate the memory consumption.

The memory consumption of MC applications depends on the volume resolution

as well. Similar to RC, we have a cubic space complexity, because potentially, all cells of the volume might be contributing. In those cases, up to five triangles³ with three vertices each are generated. Each vertex is described by six floats for position and normal vector. Therefore, we have an upper bound of 360 ($= 5 * 3 * 6 * 4$) bytes per cell⁴.

Material	Time /[s]	#Visited Samples	#Contributing Samples
CT Lobster Dataset, 4,126,840 bytes used			
M	10.9	4,803,006	6,965
S	11.9	4,868,610	109,091
R	31.9	5,064,276	3,431,673
M, S	10.5	4,739,096	87,622
M, S, R	30.4	4,682,680	3,318,655
MRI Head Dataset, 8,654,344 bytes used			
T	24.1	8,453,628	1,193,236
L	11.4	6,890,357	18,480
T, L	17.5	6,890,357	841,856

Table 2: Resource consumption of Ray Casting: R denotes resin, M denotes the meat of the lobster, S denotes the shell of the lobster, T denotes the tissue of the MRI Head, and L denotes the liquor of the MRI Head. Memory consumption is constant for all classifications of the same datasets.

Our complexity sketch shows that although the space complexity of RC and MC is both cubic, the actual memory consumption of MC can be significantly larger than RC (Tables 2 and 3). Most important is that the memory consumption of RC does not depend on the number of samples taken for the final image; the memory consumption – considering a fixed viewplane size – is constant. In contrast, the memory consumption of MC depends very much on the number of contributing cells. If this number is high, the number of generated triangles (and their respective vertices) is high as well. Therefore, the size of the memory needed is directly related to the number of contributing cells.

Time Consumption

In the previous paragraphs, we showed that memory consumption of RC applications is dependent on the resolution of the volume dataset, but not on the number of samples taken. However, the number of samples determines reconstruction and rendering time of these applications. In order to provide some operation quantification, we estimated

³On average, two triangles per contributing cell of the examined datasets were generated.

⁴As mentioned earlier, VTK eliminates vertices which are generated more than once. Therefore, the actual number of vertices is only a fraction of the estimated upper bound. Tables 2 and 3 shows only the memory used by the volume itself, the pointers of the generated triangles to the actual stored vertices, and these vertices themselves.

Material	Time /s]	Memory /bytes]	#Vertices	#Triangles	#Contributing Cells
CT Lobster Dataset, 3,585,300 visited cells, 3,705,912 voxels					
M	5.2	7,044,496	71,207	139,968	72,478
S	9.6	10,800,408	147,470	296,268	144,102
R	18.3	15,617,640	248,297	496,050	252,855
M, S	14.8	10,483,080	218,677	436,236	216,580
M, S, R	57.2	22,294,808	466,974	932,286	479,435
MRI Head Dataset, 8,256,125 visited cells, 8,387,064 voxels					
T	120.2	70,978,920	1,318,170	2,579,648	1,374,041
L	23.8	21,899,520	285,513	555,012	293,528
T, L	165.3	84,491,376	1,603,683	3,134,660	1,667,569

Table 3: Resource consumption of Marching Cubes: R denotes resin, M denotes the meat of the lobster, S denotes the shell of the lobster, T denotes the tissue of the MRI Head, and L denotes the liquor of the MRI Head. Memory consumption represents the actual used memory.

the number of operations for both approaches (see Tab. 4). Additional to the mentioned operations are numerous branching (if) and loop (for) constructs which are not considered in this time complexity sketch.

Generally, sampling along the rays first calculates the sample (one trilinear interpolation). Thereafter, this sample is classified (one access to the opacity function table). If the sample has a contribution to the final image (its opacity is not equal zero), its classification is completed (three accesses in the transfer function tables) and it is gradient shaded – which requires a gradient approximation using central differences (including 48 accesses to voxels which surround the sample⁵), and one trilinear interpolation of a vector. Finally, the sample is composited with the already accumulated samples.

The number of rays cast through the volume depends on the pixel resolution of the viewplane. However, from a theoretical point of view of sampling theory, this pixel resolution is closely related to the volumetric resolution of the dataset. According to the sampling theorem, at least four rays (four pixels of the viewplane) should be cast through each voxel. Assuming a volume with a n^3 voxel dimension, this gives a quadratic complexity of the number of rays ($O(n^2)$). Furthermore, we sample through the volume. Again, taking the sampling theorem into account, we should sample with a step size of at least half the smallest spacing of the volume. Overall, we have a viewplane resolution which depends on the volume resolution, and we have the sampling distance which depends on the volume resolution. Therefore, we can describe the time complexity as cubic.

For the reconstruction and rendering times using RC (see Table 2), the time costs

⁵ Although all voxel accesses are only in a 32-voxel-neighborhood, no optimizations are used.

High-Level Operation	Mult	Add	Others
Trilinear Interp.	7	14	
Linear Vertex Interp.	7	14	
Gradient Approx.	18	12	
Transfer/Voxel Access	1	1	
Cell Access (eight voxels)	8	20	
Normalization (per vertex)	6	2	1 sqrt
Compositing	7	4	
Summary			
MC Contributing Cell	362/ 236	386/ 224	15/6 sqrt's 8 bit op's
MC Non-contrib. Cell	8	20	8 bit op's
RC Contributing Sample	169	132	
RC Non-contrib. Sample	8	15	

Table 4: Number of multiplications (mult) and additions (add) for each high-level operation, and for the respective cell/sample types. For the number of operations for MC Contributing Cells, we consider the worst case of five triangles per contributing cells (first value) and the measured average case of two triangles per contributing cell (second value).

do not simply depend on the numbers of “visited samples”, which are – more or less – equal. More important are the samples which do contribute to the final image; these are the samples with a non-zero opacity. These samples cause most of the computational costs (gradient-shading and compositing) of RC. If the opacity of large areas of the volume is low (i.e., resin), there will be a good chance that a large share of the area will contain contributing samples. Therefore, the total number of these samples will be higher, resulting not in an early but late ray termination and, consequently many shading operations (see experiment R and M,S,R in Table 2).

The time consumption of the reconstruction process using the MC algorithm depends on the number of cells. As mentioned earlier, each cell is visited and classified, as inside, outside, or intersecting with the isosurface, which requires a cell lookup and eight bit operations. For all the contributing cells, we need one multiplication to calculate the positions with respects to the grid spacing⁶. The cell gradients for shading are determined similar to RC by using eight vector central differences (no optimizations used), which includes 48 voxel accesses. Finally, up to five triangles – depending on the classification case – are generated. Each vertex of these triangles involves a linear vertex interpolation of position and normal, and one vector normalization. Note that especially the final estimate is only an upper bound which is usually never met in practice. In our measurements, on average two triangles per contributing cell were generated (see Table 3).

⁶One multiplication is needed for each slice, for each scanline within a slice, and for each cell of a scanline. This results on average in approximately one multiplication for each cell.

The complexity is determined by the number of cells of the volume dataset, because all cells are visited by the VTK implementation of the MC algorithm (thus the similarity between voxels and the number of visited cells). Consequently, the complexity is cubic, similar to the space complexity of RC. For each cell, we can give a constant upper bound of five triangles (and 15 vertices) for the costs of gradient, position and normal computation.

Table 3 shows the timing of the reconstruction process using the MC algorithm. The VTK implementation of MC checks if a vertex was already generated by a previous triangle of the current or a neighboring cell. This checking overhead consumes a significant share of the overall time costs. For the extraction of the isosurface of the MRI Head dataset, the overhead accounts for approximately 50% (isosurface of L) up to 68% (isosurfaces T and L) of the total extraction time. Furthermore, this overhead grows faster than the actual number of vertices. Therefore, the time spent for the generation of multiple isosurfaces is higher than the sum of each individual isosurface. The reconstruction times of the different isosurfaces does vary much, due to the significantly larger numbers of contributing cells, hence the larger numbers of generated triangles (see Table 3).

Marching Cubes vs. Ray Casting

Generally, the memory costs of MC – dominated by the triangles and vertices – are significantly larger than the constant memory costs for RC – dominated by the volume itself.

For the time costs, RC was faster than MC on the MRI Head dataset, while MC was faster on the smaller CT Lobster dataset. The isotropic MRI Head dataset has a similar number of contributing cells and contributing samples. Consequently, the number of operations of MC is larger than for RC. Furthermore, the datastructure overhead of MC accounts for more than 50% of the time costs for this dataset.

The anisotropic CT Lobster dataset has a significantly smaller number of contributing cells or samples for opaque isosurfaces (i.e., M or S) than the previous dataset. Hence the datastructure overhead of MC is not as severe as for the MRI Head, resulting in lower time costs. Additionally, the number of visited cells is smaller than the number of visited samples, which increases the time costs for RC more than for MC. Furthermore, the semi-transparent samples classified as resin cause the rays of RC to travel through-out most of the dataset (experiment R and M,S,R in Table 2), thus increasing the costs for RC. Once opaque samples are added (experiment M,S and M,S,R), the rays are attenuated faster, which results in lower time costs. Transparent visualization of MC generated isosurfaces is done during the rendering stage. Therefore, it is not affecting the reconstruction stage.

Overall, MC is very efficient for single isosurfaces with a low number of contributing cells (and triangles), while RC is faster for large volume datasets. This is especially true for opaque structures, which can be exploited for early ray termination.

However, it is important to note that rendering of MC generated polygonal isosurfaces is much faster and the reconstruction can be viewed as a pre-processing step. Furthermore, the polygonal isosurface is a continuous representation, and its rendering usually is not fill-rate limited. Therefore, a larger viewport can be used at almost

no additional rendering costs. This is different using RC. If a larger viewplane is used, more rays are cast through the volume. These additional rays significantly increase the reconstruction and rendering time.

4 Conclusion and Future Work

We presented a comparison of the Ray Casting approach (DVR) and the Marching Cubes approach (IVR) for volume graphics. Not surprisingly, neither of the methods is preferable for all situations. Furthermore, this is a multi-dimensional problem and only a few aspects are presented in this paper. However, our results show that DVR techniques can have significant resource consumption advantages compared to IVR techniques.

The presented Ray Casting images often give better insights and depth perception than the Marching Cubes generated images. However, to display accurate surfaces, the latter algorithm can preserve very small details, while they can get lost in the standard Ray Casting approach. Applying oversampling can solve this problem but is very cost intensive. In contrast, the Marching Cubes algorithm cannot correctly represent volumetric features, i.e., cloud-like semi-transparent objects like resin in our example. This is inherent to the infinitely thin polygonal representation of volumetric aspects of the dataset.

Finally, the presented performance measures need to be interpreted carefully. Where as storage requirements of Ray Casting are constant while using the same dataset, memory consumption of Marching Cubes can become very costly, depending on the number of contributing cells.

The time costs for the reconstruction using the IVR technique need only to be spent once per isosurface, while the rendering time will later on only depend on the rendering performance of the graphics hardware used. This is different using DVR, because reconstruction and rendering is performed as a single pass and is currently not possible at interactive frame rates. However, once DVR accelerators become available [8, 5] near real time performance will be possible for DVR as well.

Future work will focus on the evaluation of other DVR approaches. Furthermore, we are interested in the evaluation of acceleration techniques and special features, such as cut planes and their impact on complexity.

Acknowledgments

This work has been supported by the MedWis program of the German Federal Ministry for Education, Science, Research and Technology, and by project 382 of the German Research Council (DFG).

The MRI head dataset was provided by the Department of Neuroradiology of the University Hospital Tübingen. Finally, we would like to thank Michael Doggett for proof-reading and Edelhard Becker for help using Latex.

References

- [1] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *Proc. of Symposium on Volume Visualization*, pages 91–98, 1994.
- [2] G. Frieder, D. Gordon, and R. Reynolds. Back-to-Front Display of Voxel-based Objects. *IEEE Computer Graphics and Applications*, 5(1):52–359, 1985.
- [3] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.
- [4] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proc. of ACM SIGGRAPH*, pages 163–169, 1987.
- [5] M. Meißner, U. Kanus, and W. Straßer. VIZARD II, A PCI-Card for Real-Time Volume Rendering. In *Proc. Eurographics/Siggraph Workshop on Graphics Hardware*, pages 61–68, 1998.
- [6] K. Mueller and R. Crawfis. Eliminating Popping Artifacts in Sheet Buffer-Based Splatting. In *Proc. of IEEE Visualization*, pages 239–246, 1998.
- [7] K. Mueller and R. Yagel. Fast Perspective Volume Rendering with Splatting by Utilizing a Ray-Driven Approach. In *Proc. of IEEE Visualization*, pages 65–72, 1996.
- [8] R. Osborne, H. Pfister, H. Lauer, N. McKenzie, S. Gibson, W. Hiatt, and T. Ohkami. EM-Cube: An Architecture for Low-Cost Real-Time Volume Rendering. In *Proc. Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 131–138, 1997.
- [9] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Prentice Hall, Upper Saddle River, NJ., 2nd edition, 1998.
- [10] H. Tuy and L. Tuy. Direct 2D Display of 3D Objects. *IEEE Computer Graphics and Applications*, 4(10):29–33, 1984.
- [11] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *Proc. ACM Siggraph*, pages 169–177, 1998.
- [12] L. Westover. Footprint Evaluation for Volume Rendering. In *Proc. of ACM SIGGRAPH*, pages 367–376, 1990.
- [13] J. Wilhelms and A. Van Geldern. A Coherent Projection Approach for Direct Volume Rendering. In *Proc. of ACM SIGGRAPH*, pages 275–284, 1991.