Interactive Importance-Driven Visualization Techniques for Medical Volume Data

Timo Ropinski, Frank Steinicke, Klaus Hinrichs

Institut für Informatik, Westfälische Wilhelms-Universität Münster, Germany
Einsteinstraße 62, 48149 Münster, Germany
Email: {ropinski, fsteini, khh}@math.uni-muenster.de

Abstract

In this paper we introduce volume visualization techniques developed with the aim to enhance interactive exploration of medical datasets. We present volumetric lenses which can be applied to a volume dataset to interactively focus regions of interest within these datasets. During rendering the parts of a volume dataset intersecting the lens, which is defined by a convex 3D shape, are rendered using a different visual appearance. The lenses proposed in this paper allow to apply non-photorealistic rendering techniques at interactive frame rates to aid comprehension for medical diagnosis. After briefly explaining how convex focus areas can be defined interactively we will explain the proposed lenses and their implementation in detail. Finally, we will give some performance results and conclude the paper.

1 Introduction

Visualization of medical volume datasets provides an essential diagnosis tool for medical applications. With the rise of dedicated graphics hardware it became even more and more important over the last years. Today, medical volume rendering techniques ease health professionals' work during medical diagnosis. Due to the rapid development of high precision medical imaging techniques, e.g., CT, MRI and PET, and their high spatial resolution the amount and complexity of data makes performing diagnostic examinations a challenging task. Since in most medical applications the region of interest, e.g., tumors or arterial structures, is small in comparison to the overall dataset, mechanisms are required to support health professionals to focus on these regions. Furthermore, certain features and properties of anatomical tissues are essential for the diagnosis and need to be preserved, e.g., size and shape of pathologies as well as their spatial position and vicinity to other anatomical structures. Therefore it is important that visualization techniques consider these demands and aid comprehension by visualizing contextual structures and object relations.

Traditionally, most volume visualization techniques only use a single rendering technique for displaying medical volume datasets to depict information. Although current approaches, e.g., transparency, maximum intensity projection, isosurface rendering or usage of multi-dimensional transfer functions, can provide useful insights into volume datasets the visualization often lacks information needed for certain exploration tasks. For example, direct volume rendering (DVR) techniques consider each voxel during rendering and are not sufficient to exclude non-relevant structures from visualization, that may occlude potentially relevant regions.

Thus, recent research tends to combine several volume rendering techniques in a single image in order to provide more information by emphasizing regions of interest [10, 11]. These focus-based approaches minimize elimination of contextual information and enable highlighting of regions of interest to draw the user's attention to. For example, illustrative rendering techniques, e.g., edge-enhancement, toon-shading, etc., can be applied to regions on which health professionals need to focus. All focus-based techniques exploit multiple volume rendering techniques contributing to a single image in order to provide an improved visualization by allowing insights into volume datasets. Thus, objects of interest, contours, surfaces and spatial relations are better perceivable which potentially eases medical diagnosis. Unfortunately, current focus-based rendering techniques have not been developed with the objective to achieve interactive frame

rates [10, 11], which is a major demand to perform medical diagnosis. Hence, to support the diagnosis interactive volume visualization algorithms are needed, which enable the user to visually intrude into a dataset and allow to extract as much information as possible.

The algorithm proposed in this paper allows to interactively define regions of interest within a volume dataset, to which a different visual appearance is applied. The region of interest, sometimes referred to as lens, is given by an arbitrary convex shape and a visual appearance associated with it. Data within the lens volume can be emphasized using arbitrary rendering techniques, e.g., edge-enhancement, isosurface rendering etc. In contrast to recent importance-driven volume rendering techniques, the proposed algorithm is fully accelerated by todays commodity graphics hardware, such that the shape of the lens, its position as well as the visual appearance can be modified interactively. With the proposed approach interactive exploration of volume datasets is enabled to support health professionals during medical diagnosis.

After discussing related work in the next section, we explain how to distinguish certain regions of a volume dataset during rendering. In Section 4 we propose interactive illustrative lenses supporting medical diagnosis. After giving performance measurements in Section 5 the paper concludes.

2 Related Work

Traditionally, volume rendering spans a wide spectrum of different approaches. One of the standard techniques for visualizing volumetric data is DVR, allowing the sorted composition of visual properties along viewing rays. To improve visualization a transfer function can be specified which maps the encountered voxel data values to optical properties. With the usage of different manually or semi-automatically generated transfer functions the displayed volume dataset can be classified and displayed correspondingly. Indeed, fine-tuning of rendering parameters to emphasize special regions of interest is a time consuming process not suitable for the usage in time-critical clinical diagnosis. However, in general the renderings created with DVR give a good impression of volumetric data. The renderings can be further enhanced when DVR is extended to volume shading, which considers the contribution of light sources. Another approach which allows more expressive volume visualization is the usage of illustrative rendering techniques. By using illustrative rendering techniques certain features of the dataset can be emphasized and an improved perception of structures contained in a volume dataset is enabled [1, 6]. Tietjen et al. have demonstrated the combination of volume visualization techniques with line and surface rendering techniques recently [9].

Hauser et al. [4] have presented a two-level volume rendering approach which allows to combine different volume rendering techniques, e.g., application of different transfer functions, isosurface rendering, etc., into one visualization. However, this technique is constrained to segmented datasets. The segmentation information is evaluated to apply individual rendering techniques to each object. During a global rendering step the techniques are merged into a final visualization. Since rendering processes are linked to objects emphasizing of specific regions is not possible.

In [10] Viola et al. have introduced a non-interactive volume rendering technique based on a view-dependent model for automatic focus and context volume visualization. Therefore, an object importance, i.e., the degree of visibility priority, has to be assigned to a segmented object. According to the importance and viewpoint settings objects are rendered such that important objects are more accentuated than less important ones. Besides the non-interactivity, this approach is limited since importance has to be assigned to objects manually. Furthermore, this approach is as the one presented by Hauser et al. only applicable for pre-segmented objects.

In [11] a method for generating object contours and enhancing volumetric features of a volume dataset is presented. This approach classifies volumetric data whether it lies inside or outside the focal region and uses different rendering methods for both. Different visual appearances of objects in the focal region are ensured through determining object specific transfer functions. Although this approach shows that the combination of different rendering techniques, e.g., illustrative rendering and DVR, provides the viewer a better comprehension of volumetric data, a comfortable volume exploration is not possible since interactive frame rates are not achieved. In addition, the resources and time

consumed during rendering mainly depend on the size of the dataset as well as the size of the focal region. Mainly, this is due to the fact that this focal region-based volume rendering is not hardware accelerated.

The importance-driven volume rendering techniques presented in this paper have been developed for volume ray-casting techniques. The underlying GPU-based volume ray-casting technique, in the following called GPU-based ray-casting, has been introduced by Krüger and Westermann in 2003 [5]. It allows very efficient volume rendering on commodity graphics hardware by casting rays through the volume dataset, which is represented as a 3D texture. To determine the entry and exit points for the rays the bounding box of the volume dataset needs to be rendered. For each ray the entry and exit points at the bounding box of the volume dataset are encoded as RGB color values representing volume texture coordinates. To render the volume dataset based on this color encoding the following steps are necessary:

1. render color coded bounding box's back face to texture $bbox_{back}$,
2. render color coded bounding box's front face,
   (a) fetch ray's exit point from texture $bbox_{back}$,
   (b) cast ray from its entry point, i.e., the current fragment's color, through the bounding box to the exit point, i.e., the texel color fetched in the previous texture fetch.

A major benefit of GPU-based volume ray-casting is that it supports early-ray-termination as well as empty-space-skipping to further accelerate rendering without affecting the quality of the final image. Since in ray-casting a higher sampling rate does not require to send any more proxy geometry down the rendering pipeline it can be used to render high-quality images very efficiently on current graphics hardware.

3 Separating Regions of Interest

To apply a different visual appearance inside the lens volume, the voxels contributing to a pixel in image space need to be distinguished whether they are inside or outside the lens volume. We determine these voxels' locations before rendering the dataset, since it results in better performance. This is due to the fact that less per-fragment operations are needed during rendering compared to an approach where the regions are distinguished during rendering.

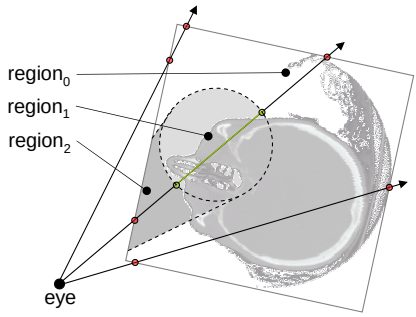

Figure 1: Scheme showing additional entry and exit points at the lens volume.

In our approach a ray cast through a volume dataset which is intersected by a convex lens volume is split into three different sections, i.e., one section in front of the lens, one inside the lens and one behind the lens (see Figure 1). Therefore we distinguish three view-dependent regions:

- $region_0$: voxels behind and next to the lens volume,
- $region_1$: voxels inside the lens volume, and
- $region_2$: voxels in front of the lens volume.

Our algorithm renders these view-dependent regions in three subsequent rendering passes starting with $region_0$. Since the number of samples used for each ray depends on its length, the number of per-fragment operations required for the three sections together is equal to the number of per-fragment operations when a single ray with equal length is processed. Because the view-dependent regions of the volume dataset are determined before accessing the dataset itself, rendering can be performed very fast, since only the usually simple shaped lens geometry and the bounding box of the volume dataset are considered during this computation.

For computing the distinction of the three regions we determine the intersection points of each ray with the lens surface in volume texture coordinates. For a convex lens volume at most two additional intersection points per ray are required, which can be stored in two RGB textures. Since the lens volume can be defined by an arbitrary convex shape, more complex surfaces have to be handled in comparison to GPU-based ray-casting, where the entry and exit

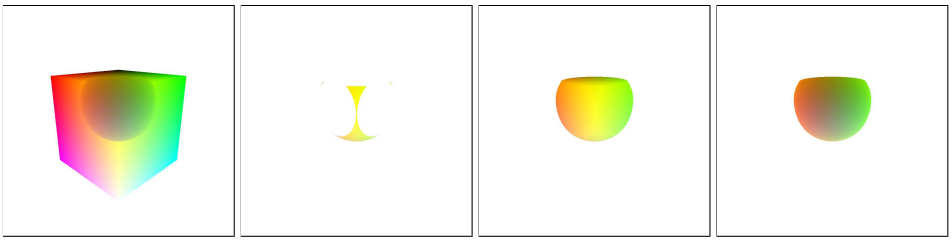

Figure 2: Textures storing intersection points at the transition between $region_1$ and $region_0$, $region_2$ and $region_1$ as well as the intersection points at a spherical lens's front and back face (from left to right). Only those fragments of the lens which are inside the bounding box of the volume dataset contribute to these images (for a colored version see Figure 5).

points are determined by rendering the front resp. back faces of the volume's bounding box.

We exploit image-based CSG rendering techniques and similar image-based rendering techniques to determine these entry and exit points. In contrast to regular CSG rendering techniques, we use the color channels to encode the volume texture coordinates needed during volume traversal. For a spherical lens volume all needed textures encoding the intersection points as RGB colors are shown in Figure 2. From left to right the figure shows the entry points for rays cast through $region_0$, i.e., behind and next to the lens, the exit points of the rays cast through $region_2$, i.e., in front of the lens volume, and the texture encoding the front and the back of the lens volume.

This paragraph describes how the textures shown in Figure 2 can be obtained efficiently. Our implementation which uses the OpenGL shading language exploits stencil functionality as well as depth peeling [2] to control which fragments are rendered during the needed passes. In the paragraphs below $bbox_{back}$ and $bbox_{front}$ denote the back face resp. front face of the volume's bounding box, while $lens_{back}$ and $lens_{front}$ denote the back face resp. front face of the lens.

When rendering the $region_0$, i.e., voxels behind and next to the lens volume, the exit points of the rays are given either by the complete or only by parts of $bbox_{back}$. When $lens_{back}$ lies entirely in front of $bbox_{back}$, $bbox_{back}$ represents the exit points for $region_0$. Otherwise only those parts of $bbox_{back}$ which lie behind $lens_{back}$ contribute to the exit points. This is due to the fact that those parts where $bbox_{back}$ lies in front of $lens_{back}$ no voxels are behind the lens and the corresponding rays do not have to be considered when rendering $region_0$. Since $region_0$ contains all voxels behind and next to the lens volume, each entry point for this region is either given by the ray's intersection point with $lens_{back}$ or by the intersection point with $bbox_{front}$. In cases were a particular ray intersects both, the $lens_{back}$ as well as $bbox_{front}$, the intersection point farther away from the viewer is used for further processing. In this step we have to ensure that only those parts of $lens_{back}$ are taken into account, which intersect the volume's bounding box. This is important because only these parts affect the volume dataset, and to efficiently perform volume rendering later on, the parts of the lens not inside the volume's bounding box should not be considered.

The exit points for $region_2$, which consists of voxels lying in front of the lens volume, are given by those parts of $lens_{front}$ which intersect the volume dataset and those parts of $bbox_{back}$, where $lens_{front}$ lies behind $bbox_{back}$. The entry points for $region_2$ are given by $bbox_{front}$ and can be determined in a straightforward way as in GPU-based volume ray-casting (see Section 2).

Since during calculation of the entry points for $region_0$ and the exit points for $region_2$ only those parts of the lens are considered which lie inside the volume's bounding box, at some fragment positions the resulting textures may not define a closed lens surface, i.e., not for each entry point an appropriate exit point is given. Hence we need another approach to determine the entry and exit points for $region_1$, because $region_1$ should be a closed lens volume. To generate such a closed surface, i.e., for each entry point of $region_1$ exists a corresponding exit point, we render the front and the back surface

of $(lens\ volume \cap bounding\ box)$. This can be done quite easily by using a CSG rendering technique, e.g., the SCS algorithm [8].

In the preceding paragraphs we have described how the intersection points for each ray that are required by our algorithm can be obtained. Thus each ray cast through the volume is specified by either two or four intersection points, depending on whether it intersects the lens volume or not. With this knowledge it is quite easy to render $region_0$, $region_1$ and $region_2$ in a back-to-front order using GPU-based volume ray-casting.

4 Interactive Focusing using Illustrative Rendering Techniques

The algorithm described in the previous section enables us to render volume datasets with focus-based regions at interactive frame rates on commodity graphics hardware. In this section we are going to discuss specific lens types from which health professionals can benefit during the exploration of medical datasets. We are going to introduce illustrative rendering lenses as well as occlusion lenses and give examples for their application and explain their implementation.

As mentioned above medical visualization can be improved by using illustrative rendering techniques. The described lens has been developed with the goal to provide the user insights into datasets by still preserving spatial cues, and to support the user to determine the spatial relationship to the rest of the dataset during exploration. In Figure 3 a lens is applied to a CT scan of a human skull, i.e., the skull is rendered using isosurface shading outside the region of interest while special rendering techniques are applied within the region of interest. The shaded isosurface rendering is a good mechanism to provide the user with cues about the overall structure of objects. It does not contain as much information as, for instance DVR where several voxels contribute to one pixel, but it provides a better overview of the surface structure. Since surface structures are the parts of objects which are perceived in real life, isosurface rendering serves as a good tool to provide spatial relationships. However, due to the fact that only one voxel contributes to each pixel color, isosurface rendering does not allow to achieve insights into a dataset. Therefore, the lens is applied to allow the user to analyze the inside structure of the dataset by still maintaining the contextual information of the isosurface rendering outside the region of interest. In the sample rendering of the skull inside the lens the inner structure of the teeth is rendered using isosurface shading combined with a different transfer function. To draw the user's attention to these structures their silhouette is highlighted with a boundary. Furthermore, we have ensured that the spatial relation of the inner structure can be extracted, since we have rendered the front and the back face of the teeth translucently. Although several techniques are combined within the lens this results in no significant loss of performance compared to other more simple lens styles. This is ensured by processing the section of the ray intersecting the lens only once during the main rendering pass. Thus the three different visual appearances, i.e., translucent rendering, highlighting and isosurface shading, can be combined in a single rendering pass in a front to back order.

Only for the edge detection based highlighting a preprocessing pass is needed. In this pass parts of the volume lying inside the region of interest are rendered as non-shaded isosurfaces; in Figure 3 an appropriate isovalue allowing to extract the inner structure as object of interest has been determined interactively. Since empty-space-skipping as well as early-ray-termination is exploited and no shading needs to be applied, this rendering pass does not result in a significant overhead. The result of this rendering pass is stored in a framebuffer-sized texture which is accessed during the main rendering pass, in which the edge detection required for highlighting objects of interest is performed as initial step. Therefore the previously created texture is accessed four times at the neighboring texels corresponding to the current pixel position. The fetched color values c_{east}, c_{north}, c_{west} and c_{south} are used to determine whether the current fragment belongs to an edge in image space: $fragc = |(c_{east} - c_{west}) + (c_{north} - c_{south})|$. If $fragc$ contains the maximum intensity in all three channels, i.e., it is white, the current fragment belongs to an edge in image space. In this case the current fragment color is set to the desired color used for highlighting objects of interest. The described edge detection mechanism can be easily extended to support highlighting of different objects without needing more rendering passes by considering the different color channels R, G, B and α separately.

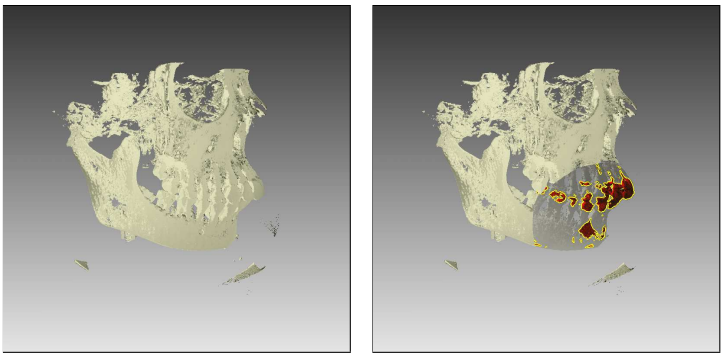

Figure 3: Rotational c-arm x-ray scan of phantom of a human skull. Isosurface shading (left); spherical lens applied giving an occlusion free view to the highlighted inner structures. To provide a better focusing on the region of interest the voxels lying behind the lens have not been rendered (right) (for a colored version see Figure 6).

After the edge detection has been performed, in those cases where the current fragment does not belong to an edge the volume dataset is processed. Therefore, as mentioned above a ray is cast through the parts of the volume which intersect the lens. To achieve the visual appearance shown in Figure 3 an isosurface rendering mode is used to determine the front faces rendered translucently. In cases where the ray intersects the volume, instead of terminating the ray after the isovalue has been encountered on the ray the ray is further processed until an object of interest, i.e., the inner structure, or the last back face of the volume dataset is hit. In both cases an appropriate shading is applied, which ensures the back face is rendered translucently and the object of interest with the corresponding transfer function. Since the ray is further processed after a voxel contributing to the visualization is encountered, the number of samples processed per ray is not larger than when applying DVR inside the lens.

Figure 4 shows a lens applied to an x-ray scan of a foot. The foot is rendered using DVR outside the region of interest to provide an overview of the interior structure. Inside the voxels' gradients are exploited and toon-shading [3] is applied to the extracted isosurfaces of the bones. The toon-shading approach allows a better depth perception, which results in improved comprehension. Furthermore, to let the bones become more apparent a black outline is rendered using the edge detection technique described above.

Besides the visual appearances introduced above the region-based rendering of volume datasets serves additional purposes. For example, we have implemented an *occlusion lens* which renders parts of the volume dataset occluding the region of interest transparently. Therefore we set the degree of transparency proportional to the amount of volume data occluding the region of interest when rendering $region_2$. This effect does not result in any performance overhead compared to DVR, because during ray traversal instead of setting the voxels alpha value based on the corresponding scalar value simply the number of samples already processed is used to determine the alpha value.

Furthermore, our algorithm allows to further enhance rendering performance when applying region based volume rendering. Therefore a different level of detail (LoD) is used inside the region of interest compared to the LoD outside this region. This is achieved by using a different sampling rate. Usually the user is interested in details within the region of interest, where relevant structures are located. The parts outside the region of interest are not in focus and can be rendered using a lower sampling rate therefore. Furthermore, as done in many volume rendering applications the LoD outside the lens can be adaptively increased when resources are available.

Since the three regions of the volume dataset are rendered in subsequent rendering passes, there are different ways how the results of these rendering passes can be combined to get the final image. We will propose two different approaches for this com-

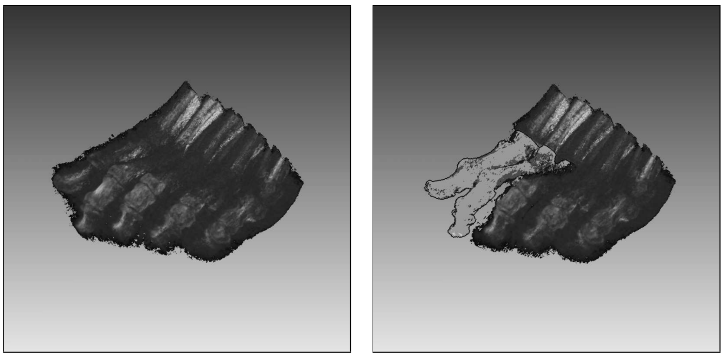

Figure 4: Rotational c-arm x-ray scan of a human foot. DVR (left); cuboid toon-shading lens applied to reveal the bone structure.

bination, which lead to different results. The most obvious way of combination is the replacement of a pixel at the same position generated in subsequent rendering passes. This method may result in images where the parts of a ray intersecting the lens volume are *occluded* by the parts of the ray lying in front of the lens. Thus the visual appearance applied inside the lens volume has no influence on the voxels lying in front of the lens volume. Therefore the fragments representing voxels in front of the lens volume are rendered by considering only the section of each ray being in front of the lens volume. Another approach which better fits into volume rendering is to blend the results of the separate rendering passes in a back-to-front order as it is done with the samples on the ray. When using this blending, the visual appearance used inside the lens volume *shines* through the volume dataset depending on the alpha value of the voxels in front of the lens. This leads to more natural results since the visual appearance used inside the lens volume merges with the rest of the dataset. To further eliminate occlusions the aforementioned occlusion lens technique can be applied.

5 Performance

For measuring the performance of the proposed algorithm for arbitrary convex lens shapes some datasets have been rendered by applying our technique. The skull dataset is shown in Figure 3 and the foot dataset is shown in Figure 4. During the measurements the datasets have been rendered both with and without applying a lens. To show the effect of changing the final image resolution the viewport size has been altered as well. The results are shown in Table 1. For the performance tests an Intel Pentium 4.3 GHz system, running Windows XP Professional, with 1 GB RAM and an *n*Vidia GeForce FX 6800 based graphics board equipped with 256 MB RAM has been used.

Table 1 shows that interactive frame rates are maintained when applying a lens to the dataset. Furthermore, the table shows that the frame rate drops when a larger viewport is used, since GPU-based ray-casting makes extensive use of per-fragment operations.

6 Conclusion

Effective medical treatment needs an efficient way to perform medical diagnosis. In this paper we have introduced a new interactive approach which improves medical diagnosis based on the exploration of volume datasets. We have demonstrated the application of different rendering techniques allowing to highlight regions of interest interactively to aid comprehension. The introduced techniques have been developed to allow health professionals to explore PET datasets interactively. Some example lenses, which have been developed within the SFB 656 “Molecular Cardiovascular Imaging“ are shown in Figure 7. Within these F_{18} PET datasets, which have been acquired on a Quad-Hidac small animal PET scanner, high scalar values represent high metabolism vitality. In contrast to other importance-driven volume rendering algorithms, our approach yields interactive frame rates

Table 1: Frame rates measured in frames per second which are achieved when rendering various volume datasets with different viewport sizes.

viewport size	512^2		1024^2	
	no lens	lens applied	no lens	lens applied
skull dataset (256^3)	30.1	17.8	19.0	9.6
foot dataset (256^3)	29.7	21.5	14.8	8.2

even on commodity graphics hardware and no expensive graphics workstations are necessary to analyze medical datasets interactively.

Although we have introduced only a few implemented lenses supporting medical diagnosis, several more lenses are possible. For example, it is possible to combine the different visual appearances presented or apply any other volume rendering technique inside a lens.

7 Acknowledgements

The authors thank Gerd Reis for fruitful discussions on volume ray-casting. The PET mouse dataset has been provided by Klaus Schäfers. Furthermore, we acknowledge the valuable comments of the reviewers. This work was partly supported by grants from the Deutsche Forschungsgemeinschaft (DFG), SFB 656 MoBil Münster, Germany (project Z1).

References

[1] David Ebert and Penny Rheingans. Volume Illustration: Non-Photorealistic Rendering of Volume Models. In *Proceedings of IEEE Visualization 2000*, pages 195–202, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[2] Cass Everitt. Interactive Order-Independent Transparency. Technical report, *n*Vidia Corporation, 2002.

[3] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-Photorealistic Lighting Model for Automatic Technical Illustration. *Computer Graphics*, 32(Annual Conference Series):447–452, 1998.

[4] Helwig Hauser, Lukas Mroz, Gian-Italo Bischi, and Eduard Grller. Two-Level Volume Rendering-Fusing MIP and DVR. In *Proceedings of IEEE Visualization 2000*, Washington, DC, USA, 2000. IEEE Computer Society.

[5] Jens Krüger and Rüdiger Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings of IEEE Visualization 2003*, 2003.

[6] Aidong Lu, Christopher J. Morris, David Ebert, Penny Rheingans, and Charles Hansen. Non-Photorealistic Volume Rendering Using Stippling Techniques. In *Proceedings of IEEE Visualization 2002*, Washington, DC, USA, 2002. IEEE Computer Society.

[7] Timo Ropinski and Klaus Hinrichs. Real-Time Rendering of 3D Magic Lenses having arbitrary convex Shapes. In *Journal of the International Winter School of Computer Graphics (WSCG04)*, pages 379–386. UNION Agency - Science Press, 2004.

[8] Nigel Stewart, Geoff Leach, and Sabu John. Improved CSG Rendering Using Overlap Graph Subtraction Sequences. In *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE 2003)*, pages 47–53, 2003.

[9] Christian Tietjen, Tobias Isenberg, and Bernhard Preim. Combining Silhouettes, Surface, and Volume Rendering for Surgery Education and Planning. In *Eurographics Symposium on Visualization (EuroVis05)*, pages 303–310. IEEE, 2005.

[10] Ivan Viola, Armin Kanitsar, and Meister Eduard Gröller. Importance-Driven Volume Rendering. In *Proceedings of IEEE Visualization 2004*, pages 139–145, 2004.

[11] Jianlong Zhou and Klaus D. Tönnies. Focal Region-based Volume Rendering. In *Proceedings of the 9th International Workshop on Systems, Signals and Image Processing*, pages 394–397. IEEE, 2002.

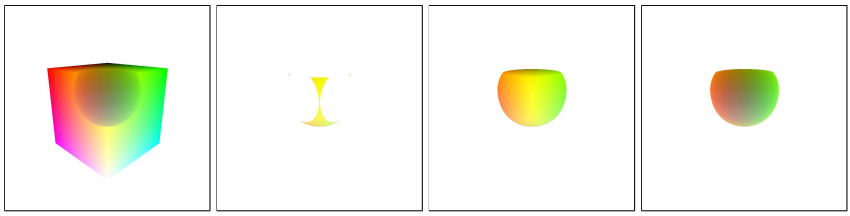

Figure 5: Color coded textures storing intersection points needed for ray-casting (see Figure 2).

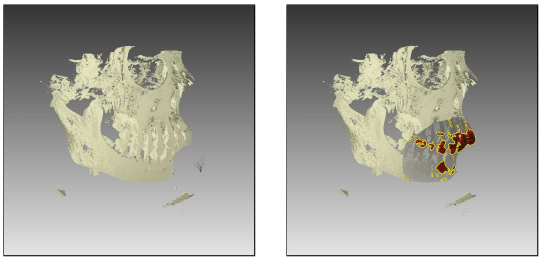

Figure 6: Rotational c-arm x-ray scan of phantom of a human skull isosurface shaded and with an illustrative lens applied (see Figure 3).

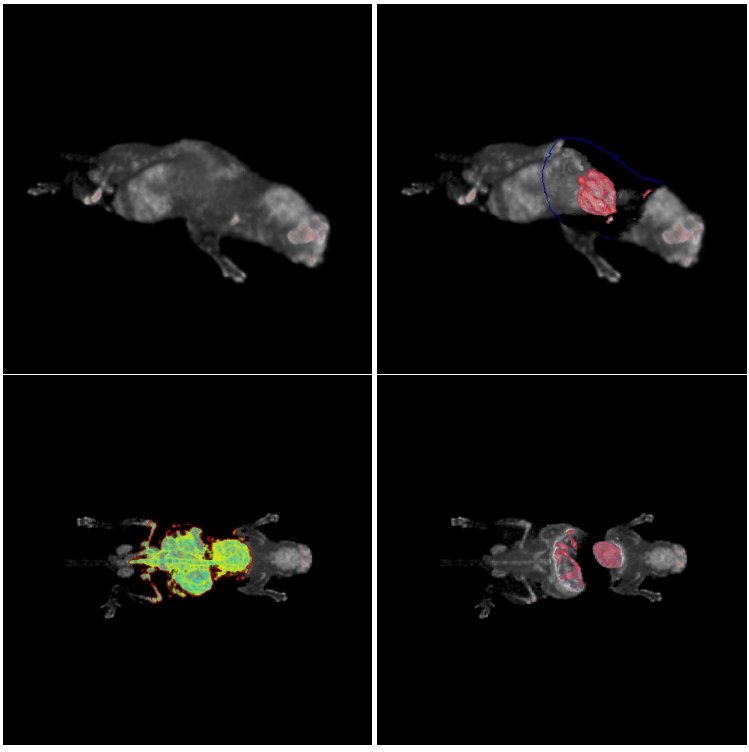

Figure 7: Applications of different lenses applied to a F_{18} PET scan acquired using a QuadHidac small animal PET scanner.