GPU-ACCELERATED VOLUME RENDERING TO 3D LENTICULAR DISPLAYS

Daniel Ruijters

Philips Medical Systems, X-Ray Predevelopment
Veenpluis 6, 5680DA Best, the Netherlands
danny.ruijters@philips.com

ABSTRACT

Multi-view lenticular displays offer stereoscopic views, without using additional glasses. In this article a method for hardware accelerated volume rendering of medical data sets to such displays is proposed. Though the generation of multi-view stereoscopic images of large volume rendered medical data sets demands an enormous amount of calculations, the GPU-based implementation offers interactive manipulation throughout. It is shown how the multi-view images are obtained from different focal spot positions, and how they are composed into the final bitmap that is displayed on the lenticular screen.

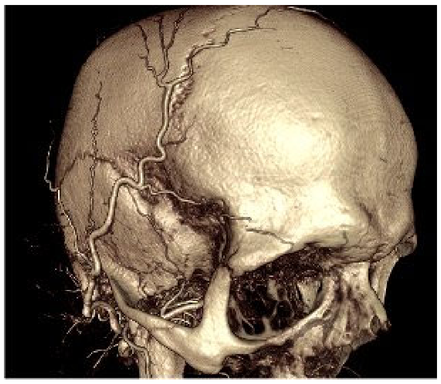

Figure 1: A Direct Volume Rendered image.

I. INTRODUCTION

New developments in medical imaging modalities lead to ever increasing sizes in volumetric data. The ability to visualize and manipulate this 3D data interactively is of great importance in the analysis and interpretation of the data. Direct Volume Rendering is a visualization technique that allows a natural representation, while maximally preserving the information, which is encapsulated in the data (see figure 1). The interactive visualization of such data remains a challenge, since the frame rate is heavily depending on the amount of data to be visualized.

Multi-view lenticular displays [1] allow a stereoscopic view on the data to multiple users, without the use of any additional aid, such as goggles. The additional depth impression that a stereoscopic image offers, allows a natural interpretation of the 3D data. A lenticular display typically offers four to fifteen spatially sequential images, which allows to view the screen from different positions. The fact that the viewer of a lenticular display is not limited to single sweet spot, makes these displays particularly suitable for use in a medical environment.

In this article a method is discussed for generating Direct Volume Rendered images for display on lenticular screens. It is presented how such an algorithm can be implemented to benefit from the processing power of modern graphics hardware. In this way interactive frame rates can be reached, enabling to fully profit from lenticular displays in a clinical environment.

II. METHOD

The Graphics Processing Unit (GPU) is a powerful parallel processor on today's off-the-shelf graphics cards. It is especially capable in performing Single Instruction Multiple Data (SIMD) on large amounts of data. In this paper we intend to sketch a method for rendering volumetric 3D medical datasets to a lenticular display, employing the GPU. Each frame, displayed on the lenticular screen, is composed of multiple different views (nine in our case). The views are separately rendered from slightly different camera positions (see figures 3 and 4), and then the resulting images have to be composited for display on the lenticular screen. Since we harvest the vast processing power of the GPU for the Direct Volume Rendering [2], as well as for the compositing phase, we can perform this whole process at interactive frame rates, even for large datasets (> 100 MB).

III. THE MULTI-VIEW LENTICULAR DISPLAY

The multi-view lenticular display device consists a sheet of cylindrical lenses (lenticulars) placed on top of an LCD in

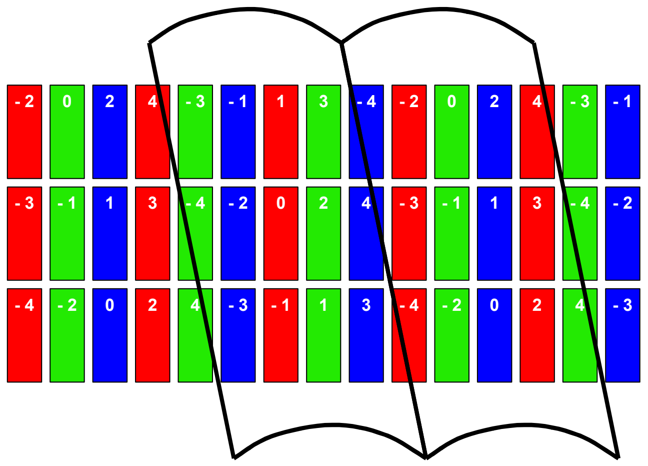

Figure 2: The cylindrical lenses depict every sub-pixel in a different view. The numbers in the sub-pixels indicate in which view they are visible.

such a way that the LCD image plane is located at the focal plane of the lenses [3]. The effect of this arrangement is that LCD pixels located at different positions underneath the lenticulars fill the lenses when viewed from different directions. Provided that these pixels are loaded with suitable stereo information, a 3D stereo effect is obtained, in which the left and right eye see different, but matching information. The screen we used offered nine distinct views, but our method is applicable to any number of views.

The fact that the different LCD pixels are assigned to different views (spatial multiplex), leads to a lower resolution per view than the resolution of the LCD grid [4]. In order to distribute this reduction of resolution over the horizontal and vertical axis, the lenticular cylindrical lenses are not placed vertically and parallel to the LCD column, but slanted at a small angle.

The resulting assignment of a set of LCD pixels is illustrated in figure 2. Note that the red, green and blue color channel of a single pixel are depicted in different views.

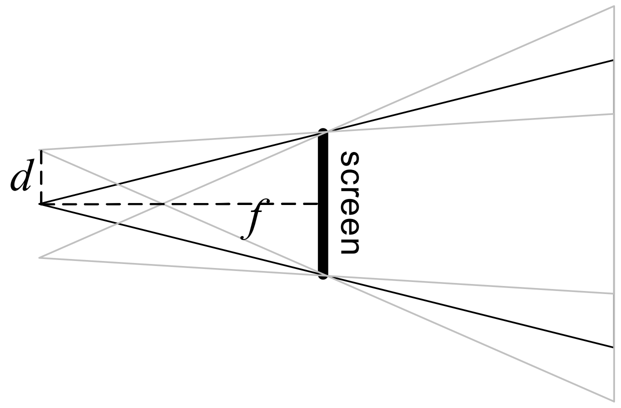

Figure 3: The frustums resulting from three different view points.

IV. THE DIFFERENT VIEWS

The frustums that result from the different focal spot positions, are illustrated in figure 3. The viewing directions of the frustums are not parallel to the normal of the screen, except for the center one. Therefore the corresponding frustums are asymmetric [5,6]. The view port coordinates of such parallel axis, asymmetric frustum perspective projections can be determined as follows:

$$\left(\frac{(x - n \cdot d) \cdot f}{f - z} + n \cdot d \ , \ \frac{y \cdot f}{f - z} \right) \tag{1}$$

Whereby f denotes the focal distance, n the view number and d the distance between the view cameras.

Figure 4 illustrates the images that result from rendering the scene from focal spot positions with an offset to the center of the screen.

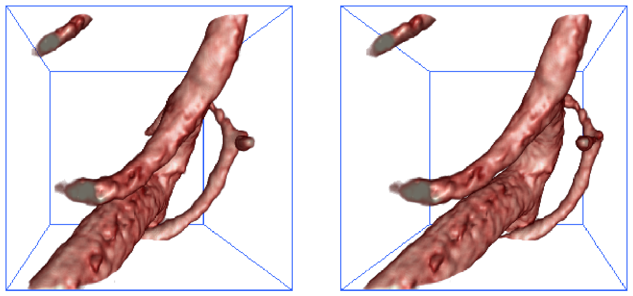

Figure 4: The same scene rendered from the most left and most right view point.

V. DIRECT VOLUME RENDERING

For each view the volumetric data set has to be rendered, using the appropriate frustum perspective projection. Since every view has an effective resolution of $(1/\sqrt{\#views})$ of the resolution of the LCD matrix in the horizontal and vertical direction, the individual views can be drawn in a view port, with a corresponding the resolution.

In order to use the GPU when rendering the datasets, the data has to be loaded in the texture memory of the graphics card. To obtain textures which are better suited for the memory architecture of the graphics card and to be able to deal with data set sizes exceeding the available texture memory, the data set is divided into so-called bricks.

The actual Direct Volume Rendering process consists of evaluating the volume rendering equation for every ray of light. We use one ray per pixel, which is defined by the focal spot and the pixel position. The volume rendering equation takes color and opacity value as a function of the position as input. Therefore the scalar values of the medical data set have to be mapped to color and opacity values. The volume render integral can be approximated by the following summation:

$$i = \sum_{n=0}^{N} (\alpha_n c_n \cdot \prod_{n'=0}^{n} (1 - \alpha_{n'})) \qquad (2)$$

whereby i denotes the resulting color of a ray, α_n the opacity at a given sample n, and c_n the color at the respective sample.

This summation can be broken down in N iterations over the so-called over operator [7], whereby the rays are traversed in a back to front order:

$$C_{n+1} = \alpha_n \cdot c_n + (1 - \alpha_n) \cdot C_n \qquad (3)$$

Here C_n denotes the intermediate value for a ray. For Equation 3 standard alpha blending, offered by DirectX or OpenGL, can be used. In order to execute Equation 3, a set of textured slices, containing the volumetric medical data, are blended into each other, whereby the volumetric data is sliced from back to front [2]. The intermediate results C_n are written in the frame buffer.

Our GPU-based Direct Volume Rendering implementation is capable of rendering highlights, diffuse and ambient lighting, which enhances the depth impression. Further it can handle any perspective projection matrix.

The rendered views are stored locally on the graphics card. They are put vertically next to each other, in a single wide rectangular texture map, which is denoted as *texture1*. The fact that the entire 3D scene has to be drawn multiple times, is partially compensated by the fact that the individual views have a lower resolution than the output window. Tests show that we can generate up to 15 frames per second for dataset consisting of 256^3 voxels on a nine view lenticular display.

VI. COMPOSITING

To composite the final image, which will be displayed on the lenticular screen, the red, green and blue component of each pixel has to be sampled from a different view (see figure 2). The view number stays fixed all the time for each sub-pixel. Thus, if the blue component of a certain pixel samples from e.g. view 3, then that will never change in time. Therefore this information is pre-calculated once, and then put in a static texture map, called *texture0*.

In the compositing phase, all the pixels in the output image are parsed by a GPU program. For each pixel, *texture0* will deliver the view numbers that have to be sampled for the red, green and blue components. The respective views are then sampled in *texture1*, delivering the appropriate Direct Volume Rendered pixel value.

VII. CONCLUSIONS

In this article a method for accelerated direct volume rendering to multi-view lenticular displays has been presented. Due to the GPU-acceleration, interactive frame rates can be reached, which allows intuitive manipulation of the rendered scene. Since both the volume rendering and the compositing take place on the graphics hardware, the requirements for the other components of the PC system are rather modest. Thus the realization of the proposed high performance system can be very cost effective.

The fact that viewers do not need to wear any additional glasses, and are not limited to a sweet spot, as well as the fact that large data sets can be manipulated interactively, make this method very suitable for a clinical environment.

VIII. REFERENCES

[1] C. van Berkel, D.W. Parker, and A.R. Franklin, “Multiview 3D LCD,” *Proc. SPIE - Volume 2653, Stereoscopic Displays and Virtual Reality Systems III*, pp. 32-39, April 1996.

[2] D. Ruijters and A. Vilanova, “Optimizing GPU Volume Rendering,” *Journal of WSCG 2006, Volume 14, No. 1-3*, pp. 9-16, January 2006.

[3] C. van Berkel, “Image Preparation for 3D-LCD,” *Proc. SPIE – Volume 3639, Stereoscopic Displays and Virtual Reality Systems VI*, pp. 84-91, May 1999.

[4] N.A. Dodgson, “Autostereo displays: 3D without glasses,” *EID: Electronic Information Displays*, 1997

[5] D. Maupu, M.H. Van Horn, S. Weeks, and E. Bullit, “3D Stereo Interactive Medical Visualization,” *IEEE Computer Graphics and Applications, Volume 25, No. 5,* pp. 67-71, September-October 2005.

[6] P. Bourke, http://astronomy.swin.edu.au/~pbourke/opengl/stereogl/

[7] T. Porter, T. Duff, “Compositing Digital Images,” *ACM Computer Graphics Volume 18, No. 3,* pp. 253-259, July 1984