The Future of Volume Graphics in Medical Virtual Reality

Judith K. Muehl, Bernhard Kainz, Alexander Bornik, Markus Grabner, Stefan Hauswiesner and Dieter Schmalstieg

Technische Universität Graz

***Abstract*— A recent trends in medical virtual reality is to include information from multiple sources, especially about physiology, into one model and one single visualization. Computer graphics must therefore deal with a huge amount of information in real time. The latest developments in computer graphics hardware allows not only to implement direct volume rendering on the graphics processing unit (GPU), but the emerging compute languages enable us to address volume rendering problems of arbitrary complexity without being limited to formulating visualization techniques in an arkward fashion to match the GPU execution model. Utilizing the arising new possibilities to meet next generation's demands in medical visualization**

***Keywords*— volume rendering, raycasting, medical virtual reality, visualization, CUDA**

INTRODUCTION AND RELATED WORK

While techniques for visualization of anatomical 3D reconstructions have been used in clinical practice for years, the future of medical applications is no longer oriented on showing images as acquired by medical technology. Added value arises from combining information from different sources as well as additional knowledge into one single presentation. From this fact arise today's challenges in medical virtual reality (VR) research.

The fundamental tool for modern medical VR is volume rendering, the process of visualizing data stored aligned to a grid. The volume data can either be transformed into a set of geometries and subsequently rendered conventionally, or else the visualization is directly derived from the volume data set, which leads to Direct Volume Rendering (DVR). DVR offers higher quality images and a larger degree of freedom, since no data is lost during a transformation [Lev88].

Early attempts to interactively display volumes used texture mapped geometry [CN94], which is still very fast, but suffers from distortions and limited flexibility [LHJ99]. More advanced GPUs allow to visualize volumes as suggested by [Lev88] using raycasting at interactive frame rates on consumer hardware [KW03, SSKE05].

A common approach to GPU-based raycasting is to render the front and back faces of the volume bounding box in a way such that the color encodes the entry and exit points of the rays in the volume [KW03]. The actual ray traversal is then performed by a fragment shader, which reads the ray coordinates from the two color buffers, fetches volume data from a 3D texture at regular sampling intervals, and applies the transfer function and color accumulation procedure.

DVR also allows efficient implementation data intermixing, a posteriori image fusion [WFZ04] as well as focus and context techniques [HMBG00]. Multi-dimensional transfer functions are proposed in [KNKI02]. DVR can be combined with geometric primitives like streamlines or additional precomputed volumes through in texture advection [LGSH06].

But the end of this development has not been reached yet, and the ever increasing demand for high quality, highly flexible volume imaging has increased also, especially in the medical field. The volume visualization should correctly intersect or blend with geometric parts of the scene, for example to correctly show surgery tools [BPVR08].

The next generation of direct volume rendering tools must be able to deal with multiple huge volumes and multivariate, high-dimensional data sets. This can hardly be achieved by straight forward extensions of the conventional CPU vertex/fragment shader model used in today's implementations. While the fundamental technique for prodicing the visualizations will still be based on raycasting, the use of the emerging GPU compute languages (speficically, CUDA) enable the use of a new class of optimized algorithms. This paper gives an overview of the prototype of a new CUDA-based visualization system currently under development at Technische Universität Graz.

REQUIREMENT FOR VOLUME GRAPHICS

Information fusion from more than one medical imaging source introduces a new set of problems for the visualization. Firstly, multi-volume visualization required registration of the multiple data sets, which is mainly a computer vision problem and not dealt with in this paper. Secondly, data in the different volumes to be visualized together may origin in the same data acquisition procedure, but will often be acquired using multiple imaging technologies and consequently require a multi-modal data approach.

Data might be taken at different points in time. Time resolved medical datasets are for example flow examinations

of the human vascular system or any kind of simulation. The visualization technique must be able to handle higher dimensional datasets with at least 4 dimensions.

The raycaster also has to visualize diverse quantities which differ in their meaning as well as in their data format. These multi-variate datasets are not limited to one scalar attribute in their representation. Vector fields and tensor fields obtained for example in diffusion tensor MRI are common examples.

Variations in data acquisition parameters and data sources lead to variations in resolution as well. Future challenges in medical applications proceed into the direction of multi-scale approaches. The corresponding computer graphics challenge is a multi-resolution approach for the multiple volumes envisioned for the raycaster. Furthermore, polygonal non volumetric objects have to be considered during a volumetric representation if an interactive volume manipulation is desired.

Finally, the images to be processed are often large datasets. Previously discussed requirements, especially the necessity to show multiple overlapping volumes at the same time, increase the memory demand beyond any reasonable borders. Approaches to deal with the arising problems concern out-of-core rendering techniques and real-time decompression of data.

CHALLENGES IN DESIGN AND IMPLEMENTATION

While widely used GPU raycasting using pixel shaders maps well onto the conventional graphics pipeline with programmable fragment processing, its flexibility is restricted due to limitations of the underlying computation framework. Multi-volume rendering or combination with polygonal geometry can be implemented by means of depth peeling [Eve01], but only at the cost of significant memory bandwidth consumption. However, technological progress in graphics hardware over the past years increasingly favors compute-intensive over memory-intensive applications. Since the raycaster kernel (transfer function and accumulation) is relatively simple, the bus traffic will become the bottleneck when aiming at more complex scenes composed from multiple volumes and polygonal objects.

Real-time raycating using CUDA

We investigate the use of CUDA [NVI08] for advanced volume rendering. A CUDA application consists of large number of concurrent threads (typically more than 1000), which are grouped into tightly coupled thread blocks. CUDA offers several benefits which are relevant in our context:

- Within each thread block, data can be cached and exchanged with other threads with extremely high bandwidth (over 1 terabyte/second) and low latency (few nanoseconds). If an application with an existing memory bus bottleneck can be rewritten to utilize these resources, performance can be improved significantly.
- The programmer has detailed control over the execution configuration (number of threads, size of thread blocks, synchronisation mechanisms) and can optimize these parameters for a particular application.
- The CUDA memory model is more sophisticated than its counterpart available in shader languages. The programmer can choose between different memory access units (e.g., texture and linear memory) and select the unit which is best suited for a particular task. Moreover, arbitrary write operations (scattering) are supported. This allows to overcome the rather bizarre algorithms developed in the past to compensate for the lack of the scatter operation in shader languages.

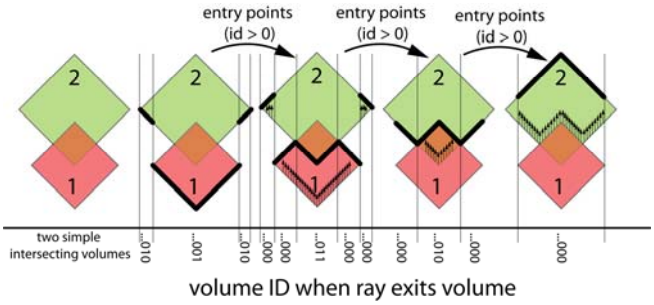

Figure 1: Illustration of the depth peeling step and the subsequent ray casting. Every time a ray leaves a volume the corresponding id of that volume is combined with a logical OR to the currently valid ID. The current ID is then used during the traversal of each ray to determine in which volume the values have to be searched.

Multi-volume raycasting

Unlike earlier multi-volume approaches, which resampled all volumes into a common grid, we support true multi volume in order to avoid double interpolations and save memory in the case of mixed resolution volumes. Our preferred way to cope with overlapping volumes is to extend the commonly known raycasting integral

$$\int_b^a g(s)\exp(-\int_a^s \tau(x)dx)ds$$

by piecewise homogenous ray segments through the volumes. Then the integral can be calculated within these volume segments with improvements like early ray termination

or empty space skipping. To correctly include arbitrary geometry and get homogenous regions, intersection calculations would have to be performed. This is computationally not feasible for real-time applications. Therefore we propose an algorithm performed in several passes to identify unique volume and shape regions. The depth peeling step for multi-volumes is outlined in Figure 1.

The remaining step for our approach is to determine which homogenous volumetric region belongs to which three dimensional volume texture. This is done by a separate assignment of orthogonal coordinates for the volumetric objects and geometry. An intersection of two objects is represented as logical OR of the object's coordinates. A simple example for that coordinate scheme is also outlined in Figure 1. With these assumptions also geometry can be handled. If a ray hits the next volumetric region, and it is a polygonal object, it only has to accumulate the object properties like color to the current ray's value.

The flow chart in Figure 2 outlines the essential parts of the resulting multi-volume rendering approach.

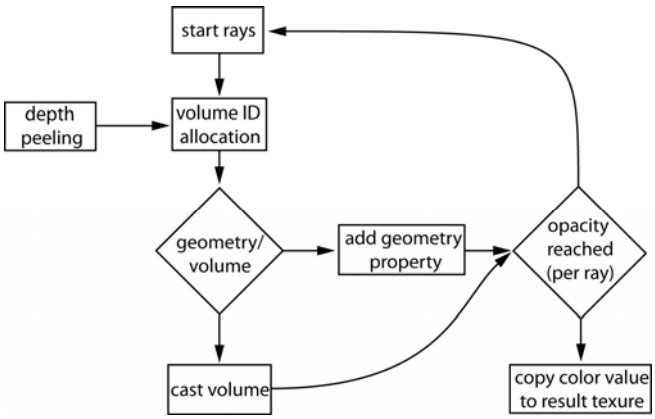

Figure 2: A schematic overview of the core components of our multi volume ray casting algorithm.

Handling multi-variate data

Multi-variate datasets can be visualized using a variety of techniques. Buerger et al. [BuHa08] give a good overview. Possible visualization techniques typically employ rendering of geometry, glyph and direct volume rendering.

In the proposed rendering framework rendering multi variate datasets will be handled in a way similar to multi-volume datasets. Each attribute of the dataset will be assigned a separate ID bit, thus the depth peeling stage described in Figure 1 except for the fact that multiple bits are inverted when entering/exiting the dataset volume, since the attribute images are congruent.

Multi-Resolution

Multiresolution (or level of detail) is a well established concept in computer graphics to deal with data at different scales [LRC+03]. This is relevant in our context for at least two reasons.

Firstly, we want to be able to investigate the data at various degrees of detail. Both a coarse overview and a highly detailed rendering of small features should be obtained from the same data structure. One way to accomplish this goal is by means of the wavelet transform [GLDH97]. More related issues are discussed below.

Secondly, data acquired by different techniques can largely differ in scale (e.g., the geometric resolution of a histogolical image is several orders of magnitude higher than of an MRI dataset). Despite this significant mismatch, we want to display those data sets concurrently to provide focus and context style interaction [HMBG00].

Multi-Dimension

Multidimensional data mainly comes from time-resolved sources. The most obvious way is to update a multidimensional volume at a certain time. Time steps can be derived from the real temporal distance in which the data was generated or real temporal distance multiplied by a consta, if the frame rate does not fulfill the sampling theorem otherwise. The latter will result in a slow motion representation of the dataset.

Most of the multidimensional visualization methods are straight forward. However, the challenge with multidimensional datasets is the huge amount of data that should be handled interactivily. Since these datasets are much larger than conventional three dimensional datasets we have to either provide a fast out of core transfer of the data to the GPU when needed, or an adequate compressed/sub-sampled representation of the data.

Out of core rendering

Massive data requires effective external memory methods, e. g., storage layouts based on space filling curves in order to maximize locality, as described in [Joy06]. Glatter et al. use M-ary balanced search trees and an attribute space Hilbert curve to distribute large multivariate datasets to a number of servers and to efficiently query portions required for rendering based on a particular multidimensional transfer function [GMHG06].

Realtime volume rendering requires all data necessary to render a single image to be present in (graphics-)memory, which can be achieved by emplotying LOD techniques and dynamically manage the active levels and corresponding representations, as done in [LWPL07] in the context of vir-

tual autopsies. There a multi-resolution representation of the dataset is created based on a flat blocking scheme. LOD selection is done based on a screen space error measure calculated after transfer function application. Higher resolution blocks are loaded for those dataset regions with the greatest impact on the output image until no more core memory is available.

CONCLUSIONS

In this paper we have outlined a design for the next-generation direct volume rendering engine. It will use CUDA to overcome the limitations of current shader based raycasting schemes, in particular by reducing the memory bandwith introduced by the purely texture based storage schemes. The new approach will handle several advanced requirements simultaneously, among them support for multiple volumes, multi-modal, multi-variate and time-dependent data sets, We have finished an early provide of concept, but the solution is not optimized, so it is too early to give performance results.

ACKNOWLEDGMENT

The presented work is a joint research activity funded by the Austrian Science Fund FWF under contract Y193, the EU under contract number ICT-223877 and the Ludwig Boltzmann Instistute for Clinical Forensic Image Processing and Graphics.

REFERENCES

[Lev88] Marc Levoy. *Display of surfaces from volume data*. IEEE Computer Graphics and Applications, 1988.

[CN94] Timothy J. Cullip and Ulrich Neumann. *Accelerating volume reconstruction with 3d texture hardware*. Technical report, Chapel Hill, NC, USA, 1994.

[LHJ99] Eric LaMar, Bernd Hamann, and Kenneth I. Joy. *Multiresolution techniques for interactive texture-based volume visualization*. In VIS '99: Proceedings of the conference on Visualization '99, pages 355{361, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press

[KW03] J. Kruger and R. Westermann. *Acceleration techniques for gpu-based volume rendering*. In VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), Washington, DC, USA, 2003. IEEE Computer Society.

[SSKE05] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. *A simple and volume rendering framework for graphics-hardware-based raycasting*. International Workshop on Volume Graphics, 0:187-241, 2005.

[BPVR08] R. Brecheisen, B. Platel, A. Vilanova, B.M. ter Haar Romeny, *Flexible GPU-Based Multi-Volume Ray-Casting*, in Proceedings of Vision, Modelling and Visualization 2008, 13th International Fall Workshop, Konstanz (2008)

[Eve01] C. Everitt. Interactive Order-Independent Transparency. Technical report NVIDIA OpenGL applications engineering, 2001.

[BUHA07] R. Bürger, H. Hauser, Visualization of Multi-variate Scientific Data, Eurographics 2007 Star of the Art Reports (STARs), pages 117-134

[NVI08] NVIDIA Corporation. Compute Unified Device Architecture programming guide version 2.0, August 2008.

[WKZ04] A. Wenger, D. F. Keefe, and S. Zhang, Interactive volume rendering of thin thread structures within multivalued scientific data sets, IEEE Transactions on Visualization and Computer Graphics 10, 6 (2004), pages 664-672.

[HMBG00] H. Hauser, L. Mroz, G.-I. Bischi, and E. Groeller, Two-level volume rendering – fusing MIP and DVR, in Proceedings IEEE Visualization 2000 (2000), pages 211-218.

[KNKI02] J. Kniss, J. Kindlmann, and G. Hansen, Multidimensional Transfer Functions for Interactive Volume Rendering, IEEE Transactions on Visualization and Computer Graphics 8, 3 (2002), pages 270-285.

[GMHG06] M. Glatter, C. Mollenhour, J. Huang, J. Gao, IEEE Transactions on Visualization and Computer Graphics 12, 5 (2006), pages 1291-1298.

[LWPL07] P. Ljung, C. Winskog, A. Persson, C. Lundstroem, and A. Ynnerman, Forensic Virtual Autopsies by Direct Volume Rendering, IEEE Signal Processing Magazine, November (2007), pages 112-116.

[Joy06] K. Joy, Massive Data Visualization: A Survey, in Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, T. Moeller, B, Hamann, and R.D. Russel eds, Springer Verlag, Heidelberg, Germany.

[LRC+03] David P. Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufman Publishers, San Francisco, 2003. ISBN 1-55860-838-9.

[GLDH97] M. H. Gross, L. Lippert, R. Dittrich, and S. Häring. Two methods for wavelet-based volume rendering. *Computers and Graphics*, 21(2):237–252, 1997

Author: Dieter Schmalstieg
Institute: Institute for Computer Graphics and Vision
Street: Inffeldgasse 16
City: 8010 Graz
Country: Austria
Email: schmalstieg@tugraz.at