Several approaches for improvement of the Direct Volume Rendering in scientific and medical visualization

Nikolay Gavrilov, Alexandra Belokamenskaya, Vadim Turlapov
Laboratory of Computer Graphics
Lobachevsky State University of Nizny Novgorod, Russia
{gavrilov86, alexandra.belokamenskaya, vadim.turlapov}@gmail.com

Abstract

This paper presents Direct Volume Rendering (DVR) improvement strategies, which provide new opportunities for scientific and medical visualization which are not available in due measure in analogues: 1) multi-volume rendering in a single space of up to 3 volumetric datasets determined in different coordinate systems and having sizes as big as up to 512x512x512 16-bit values; 2) performing the above process in real time on a middle class GPU, e. g. nVidia GeForce GTS 250 512 MB; 3) a custom bounding mesh for more accurate selection of the desired region in addition to the clipping bounding box; 4) simultaneous usage of a number of visualization techniques including the shaded Direct Volume Rendering via the 1D- or 2D- transfer functions, multiple semi-transparent discrete iso-surfaces visualization, MIP, and MIDA. The paper discusses how the new properties affect the implementation of the DVR. In the DVR implementation we use such optimization strategies as the early ray termination and the empty space skipping. The clipping ability is also used as the empty space skipping approach to the rendering performance improvement. We use the random ray start position generation and the further frame accumulation in order to reduce the rendering artifacts. The rendering quality can be also improved by the on-the-fly tri-cubic filtering during the rendering process. Our framework supports 4 different stereoscopic visualization modes. Finally we outline the visualization performance in terms of the frame rates for different visualization techniques on different graphic cards.

Keywords: *GPGPU, ray casting, direct volume rendering, medical imaging, empty space skipping, section by arbitrary mesh.*

1. INTRODUCTION

In scientific visualization it is often necessary to deal with some volumetric regular scalar datasets. These data may be obtained by some numerical simulation or via the scanning equipment such as tomographs. The output data of the CT scan is a series of the slices, i.e. two-dimensional scalar arrays of 16-bit integer values. The stack of such slices can be interpreted as a volumetric dataset which can be visualized as a 3D object.

Since the 90s, the Direct Volume Rendering shows itself as an efficient tool for the visual analysis of volumetric datasets [1-4]. Different established approaches [4] make possible the implementation of the real-time volume rendering by using the parallel and high-performance computations on the GPU. The recent progress in GPGPU computations makes the real-time multi-volume rendering possible [1]. In this paper we make a review of general details of our Ray Casting implementation, which allows for the performance and quality improvement of the available visualization methods.

1.1 The framework development motivation

There was a need in a volumetric visualization of the molecular dynamic simulation of the electron bubble transport process. There was a number of different volumetric datasets being obtained during this simulation, representing different scalar and vector fields. These datasets should be visualized together, in a single space.

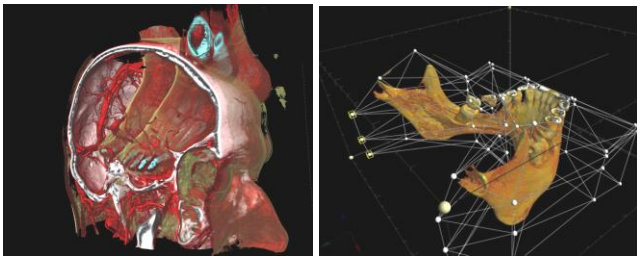

Figure 1: Features that we have implemented in our visualization software: the multi-volume rendering (left); volumetric clipping via the custom polygonal mesh (right).

However, there was no scientific visualization software available, which could satisfy our needs in the visual analysis of the scientific multi-volume time-varying volumetric datasets we had.

We wanted to be able to add several desired specific features in our own visualization framework. For instance, the interactive volumetric section via the arbitrary polygonal mesh can be efficiently used for the interactive and suitable selection of the region of interest, and we have not found any solutions with this feature. There is no multi-volume rendering software available as well, so its development is still in the research domain. Below we make a review of some significant visualization software solutions we have encountered.

The Voreen (http://www.voreen.org/) is an open source volume rendering engine, which can be nicely used for some scientific datasets visualization and the rendering quality is good enough. But when dealing with some big dataset (e.g. 512^3 cells) it appears to be hardly a real-time visualization. Moreover, there is no support for the multi-volume datasets rendering if we want to render several datasets together in one space.

The OsiriX (http://osirix-viewer.com) viewer is designed for the medical staff and has a lot of features, but it is the MacOs-only software. And while it is the medical imaging software, it can be hardly used for the scientific data visualization, because the user may want to obtain some specific data visual representation.

The Fovia's (http://fovia.com) CPU-based Ray Casting efficiency shows that even such GPU-suitable algorithms, like the Ray Casting, are still may be implemented fully on the CPU with the same or better results. However, while the graphical cards' performance, availability and architecture flexibility increases, the CPU-based efficient Ray Casting implementation is a very difficult task. Besides, the Fovia has no the desired features like multi-volume rendering and polygonal mesh based volumetric clipping.

2. METHODS AND ALGORITHMS

In this section we make a brief inspection of the Ray Casting and some other algorithms we have implemented in our system. All of these algorithms and are implemented as the GLSL shaders.

2.1 Rendering methods

Due to the high flexibility of the Ray Casting (RC) method there is a huge amount of different possible visualization techniques. The RC algorithm calculates each pixel of the image by generating (casting) rays on the screen plane and traversing them towards the observer viewing direction. Each of the RC-based rendering algorithms takes the ray start position and its direction as the input parameters, and the pixel color as the algorithm output.

There are six rendering techniques in our framework and each of them supports multi-volume rendering, i.e. these algorithms can handle several volumetric datasets, which are arbitrary located in the world space. The dataset's location is determined by the transformation 3x4 matrix. In addition to the volume's position and orientation, this matrix also defines the spacing by x, y and z components, which allows for the proper volume scaling. So to perform a sampling from the arbitrary located dataset we multiply the sampling point by this matrix, so we will get the sampling point in the volume's local coordinate system.

In order to perform the GPU-based Ray Casting, we use GLSL shaders to calculate the screen plane's pixel colors, like it is done in the Voreen framework. Each of the rendering methods in our framework is defined by some GLSL fragment shader program, i.e. a text file with the GLSL code. These methods differ from each other, but they uniformly handle all visualization parameters, e.g. the observer position, transfer functions, iso-surfaces, anaglyph matrix, etc. So it is easy for us to add new RC-based rendering techniques in our visualization framework.

2.1.1 Maximum Intensity Projection (MIP)

The Maximum Intensity Projection (MIP) is one of the most usable volumetric visualization modes in the medical imaging [3]. It is easy for clinicians to interpret an MIP image of blood vessels.

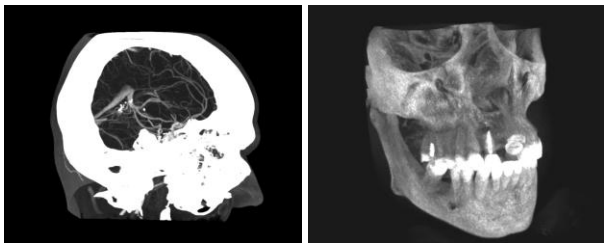

Figure 2: MIP images of the brain vessels (left) and the dental datasets.

The maximal intensities of the volumetric dataset are projected onto the screen plane. Usually the projected intensity determines the pixel's color in the gray-scaled manner (Figure 2). It medical imaging there is a window/level concept. The window is represented by two scalar values – the window width and window center.

2.1.2 The shaded Direct Volume Rendering (sDVR)

The shaded DVR technique is also used in a medical exam, but its usage is more limited [2]. In contrast to the MIP technique here we can use the early ray termination approach, which considerably improves the rendering performance without any image visible changes. This termination can be done because of the DVR algorithm nature – while traversing throw the volume data, the ray accumulates color and opacity, i.e. the optical properties, defined by the transfer function (TF). While the opacity increases, the contribution to the final pixel's color decreases. This opacity accumulation is commonly used to visualize the data as a realistic volumetric object.

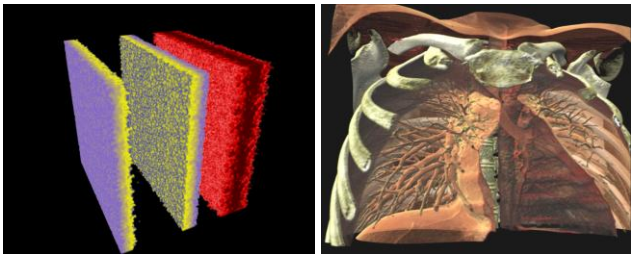

Figure 3: The scientific (left) and medical (right) data visualization by the DVR and sDVR techniques.

2.1.3 Semi-transparent iso-surfaces

Semi-transparent iso-surfaces are usually used for the scientific data visualization, because these surfaces are easier to interpret when examining some new phenomena. One of the main advantages of this method over the DVR's one is a surface's visualization high quality: here we can search for more exact intersection with the surface, a so called Hitpoint Refinement [5], while in DVR there are no surfaces. There are opaque regions in DVR, but theirs accurate visualization requires more difficult algorithm, and its laboriousness considerably reduces the performance of the GPU-implementation.

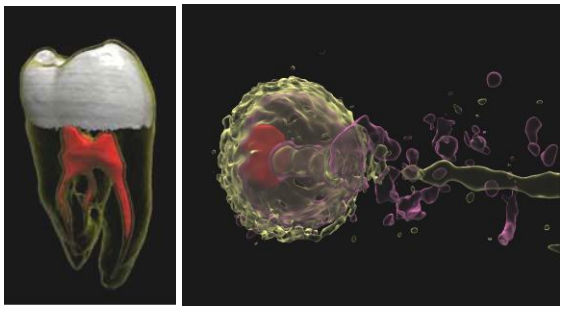

Figure 4: Semi-transparent iso-surfaces via the Ray Casting algorithm. The CT tooth dataset (left) and the simulated electron bubble transport process (right).

This method searches for ray's intersections with the iso-surfaces. When the ray passes across the iso-value, we calculate a more exact intersection point determined by the ray positions and sampled values on the current and previous steps. The linear interpolation is enough to obtain a desired result.

2.2 Optimization strategies

2.2.1 Early ray termination

The Early ray termination technique is a common optimization strategy for a Ray Casting algorithm. It is possible to terminate the RC algorithm for each individual ray if the accumulated opaqueness is close to 1. However, in rendering techniques like the MIP it is necessary to browse the whole ray path until leaving the bounding volume.

2.2.2 Empty space skipping via the volumetric clipping

The custom bounding polygonal mesh can be used either for the rendering acceleration [5] or for the data clipping, as an alternative to the dataset's segmentation. We use OpenGL frame buffers to store the distances from the viewpoint to the mesh front and back faces. The buffers may contain up to 4 ray path segments (in [5] there is only one segment), so that the mesh is not required to be convex. Of course, before using these buffers we should fill them with some relevant data. To do it we draw the bounding mesh by calling common OpenGL instructions and perform rendering into the texture. We use a specific GLSL shader program to fill pixels with the relevant data instead of the standard OpenGL coloring. The program calculates the current distance between the observer and the fragment position. The only varying parameter is the fragment position, i.e. the point on the mesh surface. After the buffers are filled with the relevant distances, the Ray Casting may be performed (Figure 5). For each ray it is known, what segments of the ray path should be traversed.

2.3 Rendering artifacts reduction

Because of the finite steps' number the ray may skip some meaningful features in the dataset, even if the ray step is much less than the voxel's size. As a result the 'wood-like' image artifacts may appear. This wood-likeness appears because each ray starts from the same plane (e.g. from the bounding box face). This artifacts' regularity can be removed by randomization of the ray start positions. The final image will contain 'noisy' artifacts instead of 'regular' ones. If a user does not change the viewpoint and other visualization settings, these random frames can be accumulated by such a way that a user will see an average image that contains no noise (Figure 8).

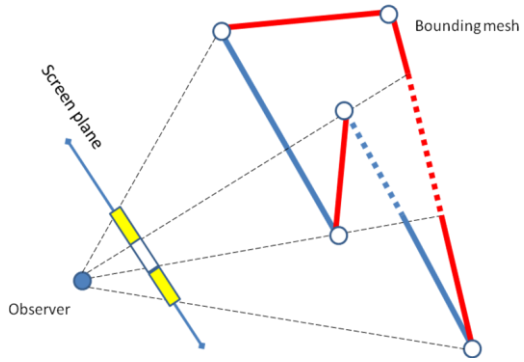

Figure 5: The illustration of the volumetric clipping algorithm. The yellow regions of the screen plane identify pixels, where are only one path segment inside the bounding mesh. In the white region there are two path segments for the rays in the Ray Casting algorithm.

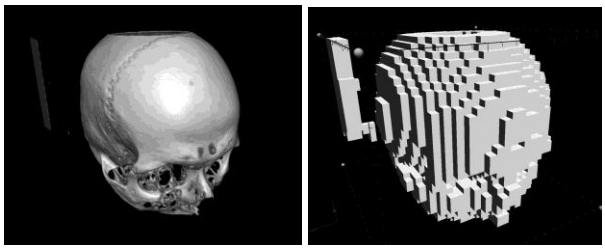

Figure 6: The source dataset (left) and its acceleration structure (right) visualization. Visible 'bricks' identify regions of interest that contain some visible features.

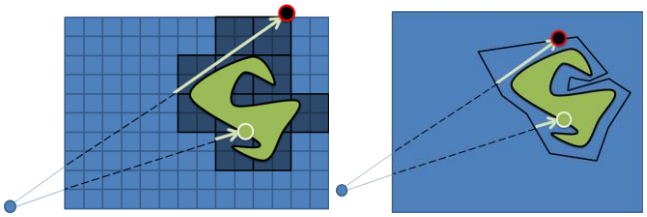

Figure 7: Empty space skipping: via the regular acceleration structure (left); via the bounding mesh (right).

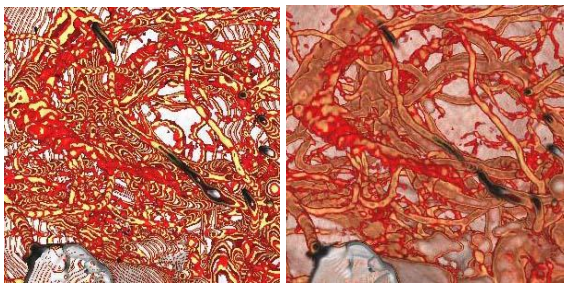

Figure 8: Before (left) and after (right) the DVR 'wood-like' artifacts' reduction by the frames accumulation approach.

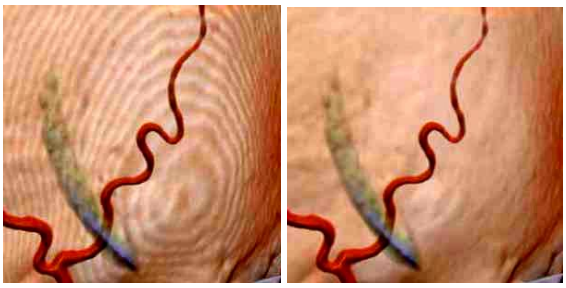

Figure 9: Tri-linear (left) and tri-cubic (right) on-the-fly samplings for the Ray Casting algorithm.

The visualization quality can be improved also via the tri-cubic filtering instead of the common tri-linear one. We have extended the algorithm for the two-dimensional case, presented in [7]. Considering the embedded bi-linear texture filtering ability, we can make only four samplings to calculate the bi-cubic interpolated value. In order to perform a single tri-cubic sampling it is necessary to make 8 basic tri-linear samplings from the same dataset, but in fact we will calculate the data value, determined by 64 nearest voxels' values.

3. PERFORMANCE ACCELERATION RESULTS

In table1 we have outlined obtained rendering performance improvement by using the space skipping technique based on the clipping via bounding polygonal mesh (Figure 6). The Acceleration 1 and 2 in table1 mean that tri-linear and tri-cubic on-the-fly samplings were used in experiments.

Data	Size	Acceleration 1	Acceleration 2
Head	512^3	1.6	3
Dental	512x512x331	1.7	3.2
Feet	512x512x250	2	3.6

Table 1: Rendering performance improvement by volumetric polygonal clipping approach.

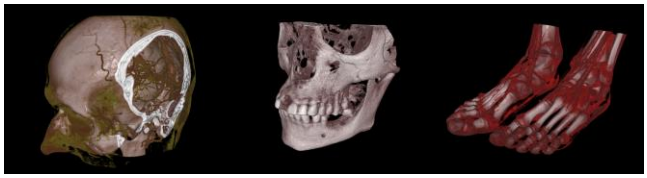

Figure 10: Test CT datasets (left to right): head, dental, feet.

4. CONCLUSIONS

The proposed framework can be used for the multi-modal medical data examination. A user can load datasets of the CT and MRI modalities as DICOM slices. Each dataset is determined by its own coordinate system and a user can properly place datasets together in the space via the control points, so that the user sees the comprehensive 3D-picture of the medical examination. The real-time rendering is performed on consumer graphic cards.

The framework is also good for stereo demonstrations via the stereo-pare of projectors, anaglyph, interlaced rendering or stereo-monitors (like Zalman with passive polarized glasses) or with shutter glasses via the nVidia 3D Vision technology.

The volumetric clipping option is performed via the arbitrary polygonal mesh. The triangles number is not sufficient and does not impair the rendering performance, so the mesh may have a rather complex structure and may be fitted to the visible features in the space. The clipping may be used either for the performance improvement or for the interactive manual data segmentation in order to select the volumetric region of interest. We define the bounding mesh as the triangulated visible cells of the regular acceleration structure. This acceleration technique improves the rendering performance up to four times.

5. AKNOLEDGMENTS

This work was supported by the federal target program "Research and scientific-pedagogical cadres Innovative Russia" for 2009-2013, state contract № 02.740.11.

6. REFERENCES

[1] Kainz B. et al, 2009. Ray Casting of Multiple Volumetric Datasets with Polyhedral Boundaries on Manycore GPUs. Proceedings of ACM SIGGRAPH Asia 2009, Volume 28, No 152.

[2] Lundström C., 2007. Efficient Medical Volume Visualization: An Approach Based on Domain Knowledge. Linköping Studies in Science and Technology. Dissertations; No. 1125.

[3] Geoffrey D., 2000. Data explosion: the challenge of multidetector-row CT. In IEEE Transactions on European Juornal of Radiology. Vol. 36, Issue 2, pp 74-80.

[4] Klaus E. et al, 2004; *Real-Time Volume Graphics*, A.K. Peters, New York, USA.

[5] Scharsach H., 2005. Advanced GPU raycasting. In Central European Seminar on Computer Graphics, pp. 69–76.

[6] Gordon L., 1999. Semi-automatic generation of transfer functions for direct volume rendering. A Thesis Presented to the Faculty of the Graduate School of Cornell University in Partial Fulfillment of the Requirements for the Degree of Master of Science.

[7] Daniel R. et al, 2008. Efficient GPU-Based Texture Interpolation using Uniform B-Splines. Journal of Graphics, GPU, & Game Tools, Vol. 13, No. 4, pp 61-69.

[8] Grimm S. et al, 2004. Memory Efficient Acceleration Structures and Techniques for CPU-based Volume Raycasting of Large Data. Proceedings of the IEEE Symposium on Volume Visualization and Graphics 2004. pp 1 – 8.