Gradient-based Classification and Representation of Features from Volume Data

Marius Gavrilescu, Vasile Manta, Eduard Gröller

***Abstract*— The extraction and representation of information from volume data are important research avenues in computer-based visualization. The interpretation of three- or multi-dimensional data from various scanning devices is important to medical imaging, diagnosis and treatment, reliability and sustainability analyses in various industrial branches, and, in more general terms, information visualization. In this paper, we present several approaches for the classification and representation of relevant information from volume data sets. The techniques are based on the gradient vector, a property directly derived from the original volume data. We show how this property can be computed and subsequently used for classification through gradient-based one- and multi-dimensional transfer functions, as well as for the enhancement of surface features. The described techniques are illustrated through images generated using our volume rendering framework, from Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) data sets. The resulting images show how gradient-based techniques are suited for improved volume classification and the better extraction of meaningful information.**

I. Introduction

Volume visualization is a sub-domain of computer graphics which deals with the exploration, sampling, classification and representation of volume data. Such data most commonly originates from scanning devices based on Computed Tomography (CT), Magnetic Resonance Imaging (MRI), ultrasound etc. When an object is scanned, the result is a volume data set which contains information on the physical properties of the media inside a rectangular region of space around the object. The data is most often arranged in a regular grid of constituents referred to as *voxels* [1]. These are the three-dimensional (3D) equivalents of pixels from two-dimensional (2D) images. However, they do not contain optical information (color and opacity), but rather have an associated scalar value which quantifies the reaction of the corresponding medium to the scanning device. This scalar value is referred to as *density* or *intensity*, depending on the type of device used. For example, in the case of medical CT images, the scalar values constitute an accurate mapping of the densities of various tissues [2]. Bone, for instance, has values from a higher range than muscle or blood vessels. This difference in associated scalar values is a powerful criterion for the classification of volume data sets, as it allows for the separation of various types of materials based on their differing densities. However, as we will explain later in the paper, density alone is sometimes not sufficient for proper classification, and thus other voxel properties have to be obtained and factored into the classification method.

Throughout the visualization process, volume data sets are subjected to various computational stages which are part of the *volume rendering pipeline* [3, 4]. Rendering is the process through which images are produced from abstract data or geometry. In the case of volume rendering, this pipeline involves loading the data into memory, reconstruction of the volume from the data set, sampling, classification, and various other techniques and enhancements. The result is an image which is produced by projecting the 3D data onto the 2D viewing plane [5]. The classification stage is arguably among the most important ones, since it is here that the information of interest is isolated from the rest [6]. In this paper we focus on the type of classification which makes use of the gradient vector to discriminate among voxels. In the following sections we explain what the gradient vector is, how it can be computed, and how it can be used for the purposes of volume visualization.

Section II of the paper briefly describes the volume rendering process used in our framework. Section III presents several approaches used to compute the gradient vector and illustrates their advantages and shortcomings. Section IV presents techniques for gradient-based classification, applied to CT and MRI medical data sets. The final section concludes the paper and details future extensions of the described methods.

Manuscript submitted on 18.06.2011 This work was supported in part by the "BRAIN - An Investment in Intelligence" doctoral Scholarship Program, within the Technical University of Iasi, Romania. This paper was supported by the CNCSIS project IDEI 342/2008, *Nonlinear evolution equations - theoretical aspects and applications to life and environmental sciences*.

Marius Gavrilescu is a PhD student from the Technical University of Iasi, Romania, currently working at the Vienna University of Technology, Austria (e-mail: mariusgv at cg.tuwien.ac.at, mariusgav42 at yahoo.com).

Vasile Manta is professor at the Technical University of Iasi, Romania, Faculty of Automatic Engineering and Computer Control (e-mail: vmanta at cs.tuiasi.ro).

Eduard Gröller is associate professor at the Vienna University of Technology, Austria and adjunct professor of computer science at the University of Bergen, Norway (e-mail: groeller at cg.tuwien.ac.at).

II. Volume Reconstruction, sampling and rendering

This section succinctly explains the methods used by our framework for processing and representing the volume data. For readers not directly associated with volume graphics or computer graphics in general, we feel that these explanations are necessary in order to make other sections easier to follow.

The initially discrete volume data is stored in video memory as a 3D texture [7]. This allows the reconstruction of the continuous volume via hardware-implemented trilinear interpolation [8]. The volume is now continuous and described by a function $V: R^3 \rightarrow$R, which maps scalar values to each position *(x, y, z)*. After reconstruction, scalar values are retrievable via texture look-ups from any such position within the volume. The distribution of the scalar values depends on the physical properties of the materials of the scanned object. Denser materials such as bone or metal have different scalar values from lighter ones such as soft tissues, plastic or various fluids.

The scalar values are the basis for retrieving information from the data. The aim is to extract the desired information from the volume while hiding unwanted data, and to generate an image which illustrates this information.

The rendering approach used by our framework is called Direct Volume Rendering (DVR) [9, 10]. Essentially, it involves sampling and displaying the volume without employing additional geometry or triangle-based surfaces to extract the desired information. As the name suggests, DVR methods work directly on the data points. The algorithm used for sampling and projecting the data is called *ray casting* [7, 11]. The concept behind this approach is illustrated in Fig. 1.

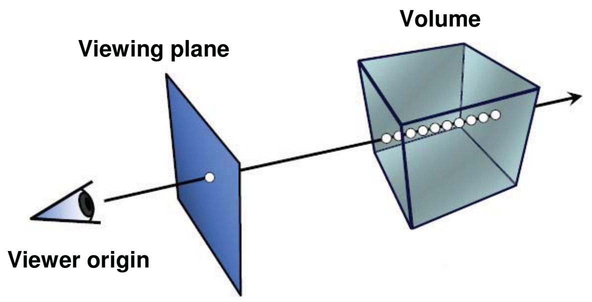

Fig. 1. Sampling of a volume data set through ray casting. A single ray is illustrated, which is emitted from the viewing plane into the box-shaped volume. Inside the volume, the white dots are positions sampled along the ray

Ray casting is a popular algorithm with multiple implementations in volume visualization [12, 13, 14]. The main steps for basic ray casting are as follows:

- for each pixel, emit a ray from the viewing plane through the volume
- sample the volume along the rays (as illustrated in Fig.1)
- assign opacity and color to each sampled point using some classification method
- combine the colors and opacities of the points along each ray (the result is the color of the pixel from which the ray was originally emitted)

The colors of the resulting pixels are combinations of the colors of the sampled points along each respective ray, weighted by the opacities of each point. Less transparent points contribute more color to their corresponding pixel than more transparent ones. Through classification, the distribution of color and transparency is controlled throughout the volume. Regions which contain uninteresting data are typically set to be completely transparent, while useful information is semi-transparent or opaque, and highlighted by a proper combination of color and opacity.
Some of the more frequently used methods for classification are collectively referred to as *transfer functions*. A transfer function maps color and opacity to sampled points in the volume depending on the properties of voxels. The most straightforward of such properties is the scalar value itself, which leads to a category of classification methods called *data-based transfer functions*. Such functions assign color and opacity based on the scalar value. These optical properties are typically RGBA values; RGB is the Red-Green-Blue color space, while A is called *alpha value* and controls opacity. Therefore, transfer functions are $T_f : R \rightarrow R^4$, where $T_f(V)$ = RGBA. These are 1D functions which assign an RGBA quadruplet to each value of the volume function V described earlier. Through careful specification of transfer functions, certain regions in the volume can be emphasized, while others are hidden. Throughout the paper, we define the shape of these functions through user-interactive tools [15, 16, 17], which allows great flexibility for the fine-tuning of the rendered images. In certain cases however, taking only the scalar value into account is not enough, and other voxel properties are needed for a more refined classification. In the following section, we describe such a property and explain its usefulness for the purpose of visualization.

III. The gradient vector

Considering the function V mentioned in Section II, the gradient vector (or *gradient*, for short), is its first derivative. Since V is a three dimensional function, the gradient ∇V is a three-component vector containing the partial derivatives of V in directions, x, y, and z.

The most useful property of the gradient is that its orientation locally indicates the largest degree of change in scalar values, which can indicate a significant surface inside the volume [18]. For instance, the gradient at the interface between bone and muscle tissue is oriented perpendicularly to this interface, and this orientation can be used to identify the surface of the bone. One important application of gradient vectors is the implementation of a local illumination model. Lighting adds a significant degree of realism and quality to a rendered image, and it is featured in nearly every figure in this paper. Local illumination relies on the normal vector, which indicates the local orientation of a surface. The normal vector from triangle- and quad- based graphics can be substituted by the gradient, since, as mentioned, it is perpendicular to significant identifiable surfaces in the volume [19].

There exists a variety of techniques for the numerical estimation of the gradient from discrete data. There are basically two main directions when computing the gradient: pre-computation and dynamic computation.

The idea when pre-computing gradients is to calculate them in a pre-processing step, before any actual rendering takes place. The result is a three- or four-component volume which stores for each voxel the corresponding gradient as three components and, if desired, its magnitude as the fourth component. In the paper, we use the *central differences* approach for gradient estimation. This means that, for a specific voxel, the gradient is computed based on the differences between neighboring voxels in various arrangements. The simplest such arrangement is to use pairs of neighbors from each main axis, located in either direction, for a total of six neighboring voxels. Assuming the discrete volume is stored as a 3D array with components $V_d[i, j, k]$, where i, j, k are indexes used for traversing the array in its three respective dimensions, the gradient is computed as shown in (1).

$$\nabla V = 0.5\begin{pmatrix} V_d[i+1, j, k] - V_d[i-1, j, k] \\ V_d[i, j+1, k] - V_d[i, j-1, k] \\ V_d[i, j, k+1] - V_d[i, j, k-1] \end{pmatrix} \qquad (1)$$

An extension of this method involves using all 27 values from within a 3x3x3 cell around a voxel, i.e., all $V_d[i+i_x,j+j_y,k+k_z]$, where i_x, j_y, k_z = {-1, 0, 1}. This results in a more precisely estimated gradient and a smoother look of the rendered images [20].

Unlike the pre-computational approach, dynamic gradient estimation is incorporated into the volume rendering pipeline and thus is performed in real-time. The gradients are not pre-stored in memory, but rather are calculated when needed during the rendering process.

Similarly to the first method from Subsection III, six neighbors are initially used, two for each axis. The difference is that the gradient estimator often operates on the continuous, reconstructed volume. Thus, neighboring voxels are not restricted to an indexed matrix, but may be chosen by sampling at an arbitrary distance d from the center voxel (2). d is set to a value which is small compared to the size of the volume.

$$\nabla V = \frac{1}{2d}\begin{pmatrix} V(x+d, y, z) - V(x-d, y, z) \\ V(x, y+d, z) - V(x, y-d, z) \\ V(x, y, z+d) - V(x, y, z-d) \end{pmatrix} \qquad (2)$$

The method can be further improved by considering a multitude of neighbors around the center voxel. We refer to this approach as *spherically-computed gradient.* Specifically, we take samples located on the surface of a sphere centered on the voxel in question. The difference is then computed between each sample on one hemisphere and its diametrically opposed correspondent on the complementary hemisphere. Summing up these differences produces the desired gradient. The positions of the samples are calculated using spherical coordinates. Considering a voxel located at position *vPos = (x, y, z)* with the associated scalar value *V(vPos)*, the gradient vector is obtained as depicted in (3), assuming n samples are taken at distance r from the voxel.

$$pos = \begin{pmatrix} \cos\left(\frac{i\pi\sqrt{2/n}}{2}\right)\sin\left(2i\pi\sqrt{2/n}\right) \\ \sin\left(\frac{i\pi\sqrt{2/n}}{2}\right)\sin\left(2i\pi\sqrt{2/n}\right) \\ \cos\left(2i\pi\sqrt{2/n}\right) \\ \end{pmatrix} \qquad (3)$$

$$\nabla V = \sum_{i=0}^{n/2-1}\left(\left(V(vPos + r\cdot pos) - V(vPos - r\cdot pos)\right)\cdot pos\right)$$

The effects of the use of gradients as computed using the aforementioned approaches are shown in Fig. 2. We implemented local illumination using the Phong model [21], where the gradient is used in place of the normal. Using more than six neighbors results in smoother images regardless of whether the computation is carried out in a pre-processing step or dynamically. The advantages and drawbacks of both approaches stem from the often-occurring trade-off between memory and computational power requirements. Pre-computed gradients require as much as four times more memory than the volume data set. For and for large data this may well exceed the available video memory. However, once the gradients are stored, they are obtainable via a single texture fetch, with barely any impact on performance. Dynamic gradients, on the other hand, require barely any memory. Since they are computed on-the-fly and require multiple texture fetches, the impact on performance can be substantial. The spherically-computed gradients have a particularly negative influence on the time required for rendering, but produce better quality images. A good trade-off is the pre-computed 27 value method, which is sufficient for high-quality images, provided enough memory is available.

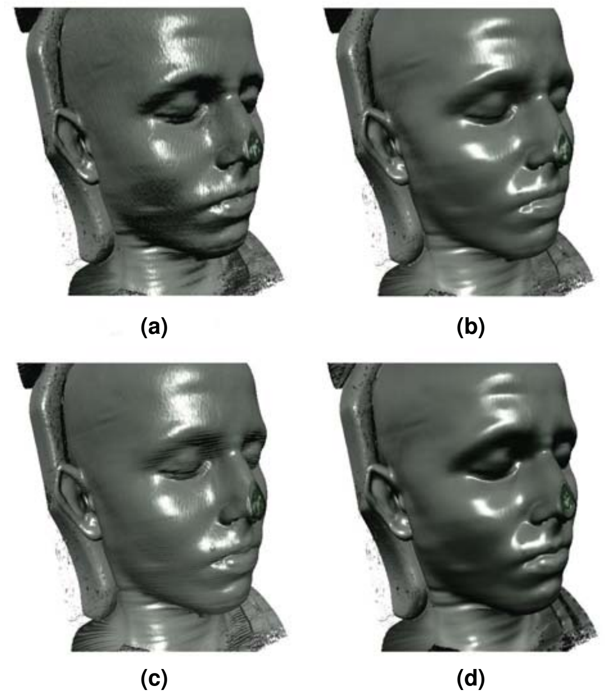

Fig. 2. Illumination of a volume data set using: pre-computed gradient using 6 neighbors (a); precomputed gradient using 27 values (b); dynamically computed gradient using 6 neighbors (c); dynamic spherically-computed gradient (d).

IV. GRADIENT-BASED CLASSIFICATION

For the purposes of volume classification, the most significant application of gradients is the identification and highlighting or elimination of surfaces. The orientation of the gradient indicates the orientation of the surface, while its magnitude indicates the difference between the materials on either side of the surface. We use the gradient magnitude as an additional voxel property (aside from the scalar value) and incorporate it into the classification process. The first classification method involves 1D gradient-based transfer functions. Similarly to data-based transfer functions, this class of functions assigns RGBA values to sampled voxels, and is described by (4).

$$T_f : R \to R^4,\quad T_f\left(\left\|\nabla V\right\|\right) = RGBA \qquad (4)$$

Fig. 3 presents the result of using gradients for classification. Fig.3(a) shows a volume rendered with a 1D data-based transfer function. A 1D gradient-based function is used In Fig.3(b). Various surfaces are clearly identified and rendered semi-transparently, while color is used to separate certain features such as the sinus cavity and the metencephalon. The voxels corresponding to the structures indentified in bright green in Fig 3(b) could not be similarly highlighted using a data-based function (Fig. 3(a)) because they share scalar values with neighboring structures which occlude them. Surface information, however, allows for their separation.

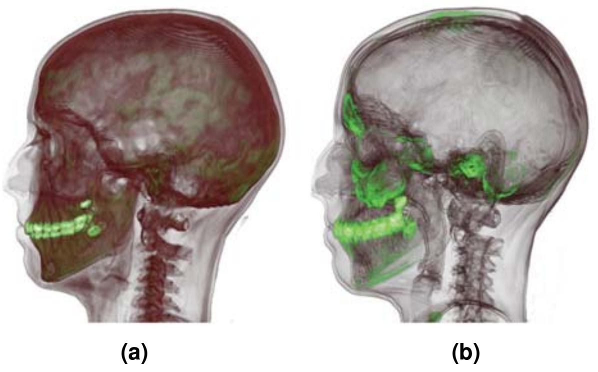

Fig. 3. Classification using: a 1D data-based transfer function (a); a 1D gradient-based transfer function (b).

The identification and highlighting of surfaces is further explored using combinations of data- and gradient-based classification. One approach is to use what we refer to as *thresholded transfer functions*. These are assigned a threshold scalar value to separate data- and gradient-based classification. With thresholded transfer functions, gradient-based classification is applied to voxels whose scalar value is below the threshold, while scalar values are used when they are above this threshold.

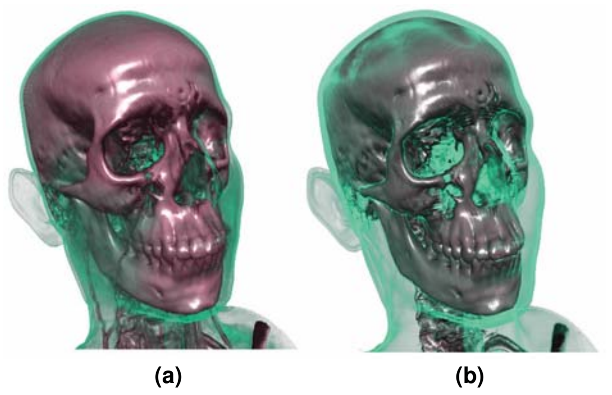

Fig. 4. Surface identification and highlighting using: a 1D data-based transfer function (a); a thresholded transfer function (b)

A result is shown in Fig. 4. The objective is to highlight the outer surface (the skin tissue), while still keeping bone structures visible. In Fig. 4(a), this was attempted using a traditional 1D data-based function. While the outer, transparent surface was correctly identified, it could only be highlighted to a certain extent, before started occluding other structures. Fig. 4(b) shows the same attempt, only this time a thresholded transfer function was used. Bone tissue was still represented from scalar values, but skin was identified using gradient information. The result is that the skin surface can be better highlighted, while still keeping the bone underneath visible.

The flexibility of the classification is further extended into the realm of 2D transfer functions [22]. For an arbitrary voxel, an RGBA quadruplet is generated considering both its properties, the scalar value and the gradient magnitude (5).

$$\begin{aligned} &T_{f2} : R^2 \rightarrow R^4, \\ &T_{f2}\left(V, \|\nabla V\|\right) = RGBA \end{aligned} \tag{5}$$

Two dimensional transfer functions have two independently-generated components: one is data-based (T_{fd}) and the other one is gradient-based (T_{fg}). The 2D function is formed from combinations of these two components, thereby incorporating both voxel properties into the classification process (6).

$$T_{f2} = w_d \; T_{fd} + w_g \; T_{fg} \tag{6}$$

Through a careful specification of the two components, as well as a fine-tuning of the w_d and w_g weights, this form of classification proves more flexible than its one-dimensional counterpart.

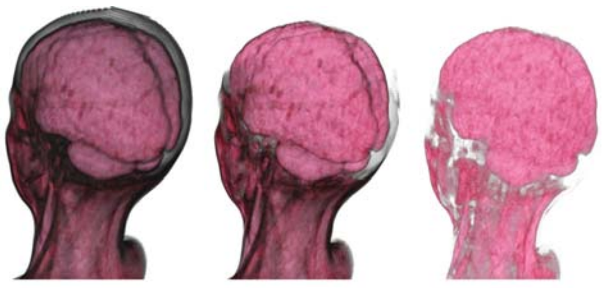

Fig. 5. Visualization of brain tissue from a CT dataset using a 2D transfer function; the layer of skin from (a) is partially removed (b) and completely removed (c).

Fig. 5 shows the application of a 2D transfer function for a CT dataset. The aim is to show soft brain tissue underneath similarly dense skin. These two tissues are inseparable based on scalar values only. The brain and skin are separately identified in Fig. 5(a). Through the manipulation of the aforementioned weights, the opacities associated with certain gradient values can be lowered.

This is reflected in Fig. 5(b), where it can be seen how the outer surface starts to lose visibility and "peel away". The brain beneath it is still visible and largely unaffected. The limit of the technique is shown in Fig. 5(c). The outer surface is completely removed, but some surface information is also lost from the brain region.

We also tried this technique on MRI scans. MRI data is often more problematic than CT, because the scalar values for distinctive materials overlap more frequently and the noise content is substantially higher. However, soft tissue such as brain matter or blood is better identified in MRI images. Unlike the previous CT dataset, the brain can be identified using a 1D data-based transfer function (Fig. 6(a)).

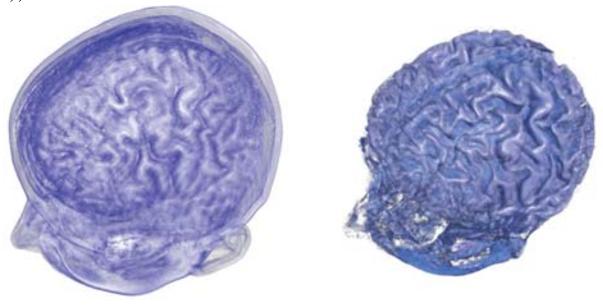

Fig. 6. Visualization of brain tissue from an MRI data set using: a 1D transfer function (a); a curvature-modulated 2D transfer function (b).

As previously, we attempted to remove the outer skin layer, but, due to the noise content and the distribution of scalar values throughout the dataset, the 2D transfer function described by (6) proved insufficient. We therefore have to further refine the classification. The transfer function from (6) is expanded by adding curvature information [23, 24, 25], since the ridges and valleys from the surface of the brain are generally more curved than the outer layer of skin. First, local curvature is estimated as shown in (7). For any vector *vect* we use the notation *vect*$_n$ to denote that *vect* is normalized.

$$k(x,y,z) = \left\| \begin{pmatrix} \nabla V(x+d,y,z)_n \cdot \nabla V(x-d,y,z)_n \\ \nabla V(x,y+d,z)_n \cdot \nabla V(x,y-d,z)_n \\ \nabla V(x,y,z+d)_n \cdot \nabla V(x,y,z-d)_n \end{pmatrix} \right\| \tag{7}$$

The curvature is estimated to be the magnitude of the vector whose components are the dot products of the normalized gradients from each respective axis. Once the curvature is obtained, the transfer function from (6) is modulated accordingly (8). An additional coefficient w_k is also introduced for fine tuning the modulation.

$$T_{f2k} = w_k \; k \left(w_d \; T_{fd} + w_g \; T_{fg} \right) \tag{8}$$

The result can be seen in Fig. 6(b), where the outer layer of skin has been removed almost completely.

V. Conclusions

The paper presented methods for the estimation of the gradient from volume data, and various applications involving classification via transfer functions based partially or completely on the gradient. The rendered images show that incorporating this voxel property into the classification process allows for more classification possibilities. 1D and 2D transfer functions were used for the identification of structures and the highlighting and removal of surfaces, from both CT and MRI data. When 2D functions proved insufficient, curvature information was added in order to refine the classification process. Future work in this direction involves the expansion of the classification possibilities within our rendering framework. We intend to incorporate additional voxel properties, such as occlusion or distance, into the classification pipeline, to allow for even greater flexibility. The aim is to enable the extraction of individual structures within volumes without having to rely on segmentation or costly non-interactive methods.

Acknowledgment

Marius Gavrilescu thanks the members of the Vis-Group at the Institute of Computer Graphics and Algorithms of the Vienna University of Technology for their valuable support and help with research.

References

[1] M. Chen, A. Kaufman, R. Yagel, *Volume Graphics*. Springer, 2000.
[2] J. Hsieh, *Computed Tomography – Principles, Design, Artifacts and Recent Advances*. Bellingham, WA: The International Society for Optical Engineering, 2003.
[3] T. Peterka, R. Ross, Y. Hongfeng, K-.L. Ma, W. Kendall, H. Jian, “Assessing improvements to the parallel volume rendering pipeline at large scale”, in *Proc. Ultrascale Visualization 2008 Workshop*, Austin TX, 2008.
[4] J.E. Vollrath , D. Weiskopf , T. Ertl, “A generic software framework for the gpu volume rendring pipeline”, in *Proc. Vision, Modelling and Visualization*, Erlangen, Germany, 2005, pp. 391-398.
[5] Chen M., Kaufman A., Yagel R., *Volume Graphics*. Springer, 2000.
[6] E.B. Lum, J. Shearer, K-.L Ma, “Interactive multi-scale exploration for volume classification”, *Vis. Comput.*, vol. 22, no. 9, pp. 622-630, 2006.
[7] M. Hadwiger, J. M. Kniss, C. Rezk-Salama, D. Weiskopf, and K. Engel, *Real-time Volume Graphics*. A K Peters, 2006.
[8] H. Nguyen, *GPU Gems 3*, Addison-Wesley Professional, 2007.
[9] G.M. Nicoletti, “Advances in direct volume rendering for visualizing large 3D data sets from scientific and medical applications”, in *Proc. 5th Biannual World Automation Congress*, vol. 13, 2002, pp. 245-250.
[10] V. Solteszova, D. Patel, S. Bruckner, I. Viola, “A Multidirectional Occlusion Shading Model for Direct Volume Rendering”, *Comput. Gr. Forum*, vol. 29, no. 3, 2010, pp. 883-891.
[11] L. Marsalek, A. Hauber, P. Slusallek, “High-speed volume ray casting with CUDA”, in *Proc. IEEE Symposium on Interactive Ray Tracing*, Los Angeles, 2008, pp. 185-190.
[12] M. Weiler, M. Kraus, M. Merz, T. Ertl, "Hardware-based ray casting for tetrahedral meshes," *in Proc. IEEE Visualization 2003*, Seattle, WA, 2003, pp. 333-340.
[13] H. Ray, H. Pfister, D. Silver, T.A. Cook, “Ray casting architectures for volume visualization”, *IEEE Trans. Vis. Comput. Gr.*, vol. 5, no. 3, 1999, pp. 210-219.
[14] F. Rossler, R.P. Botchen, T. Ertl, “Dynamic shader generation for GPU-based multi-volume ray casting”, *Comput. Gr. App.*, vol. 28, no. 5, 2008, pp. 66-74.
[15] S. Bruckner, E. Gröller, “VolumeShop: An interactive system for direct volume illustration”, in *Proc. IEEE Visualization 2005*, Minneapolis, MN, 2005, pp. 671-678.
[16] J. Meyer-Spradow, T. Ropinski, J. Mensmann, K. Hinrichs, “Voreen: A Rapid-Prototyping Environment for Ray-Casting-Based Volume Visualizations”, *IEEE Comput Gr. App.*, vol. 29, no. 6, 2009, pp. 6-13.
[17] M. Gavrilescu, M.M. Malik, E. Gröller, Custom interface elements for improved parameter control in volume rendering, in *Proc. ICSTC 2010*, Sinaia, Romania, pp. 219- 224.
[18] M. Levoy, “Display of surfaces from volume data”, *IEEE Comput. Gr. App.*, vol. 8, no. 3, 1988, pp. 29-37.
[19] C. Rezk-Salama, M. Hadwiger, T. Ropinski, P. Ljung, ”Advanced illumination techniques for GPU volume raycasting”, *SIGGRAPH Course Notes,* 2009.
[20] L. Neumann, B. Csebfalvi, A. König, Eduard Gröller, “Gradient estimation in volume data using 4D linear regression”, *Comput. Gr. Forum*, vol 19, no 3, 2000, pp. 351-358.
[21] T. Whitted, “An improved illumination model for shaded display”, *Communications of the ACM*, vol. 23, no. 6, 1980, pp. 343-349.
[22] J. Kniss, G. Kindlmann, C. Hansen, “Multidimensional transfer functions for interactive volume rendering”, *IEEE Trans. Vis. Comput. Gr.*, vol. 8, no. 3, 2002, pp. 270-285.
[23] J. Hladuvka, A.H. König, E. Gröller, “Curvature-based transfer functions for direct volume rendering”, in *Proc. Spring Conference on Computer Graphics 2000*, vol. 16, 2000, pp 58-65.
[24] G. Kindlmann, R. Whitaker, T. Tasdizen, T. Möller, “Curvature-based transfer functions for direct volume rendering: methods and applications”, in *Proc. IEEE Visualization 2003*, Washington DC, 2003, pp. 67-76.
[25] S. Bruckner, E. Gröller, “Style transfer functions for illustrative volume rendering ”, *Comput. Gr. Forum*, vol. 26, no. 3, 2007, pp. 715-724.