ISSN 8756-6990, Optoelectronics, Instrumentation and Data Processing, 2021, Vol. 57, No. 2, pp. 141–148. © Allerton Press, Inc., 2021.
Russian Text © The Author(s), 2021, published in Avtometriya, 2021, Vol. 57, No. 2, pp. 32–40.

ANALYSIS AND SYNTHESIS OF SIGNALS AND IMAGES

A Method for Visualizing Multivolume Data and Functionally Defined Surfaces Using GPUs

S. I. Vyatkin[1]* and B. S. Dolgovesov[1]

[1] *Institute of Automation and Electrometry, Siberian Branch, Russian Academy of Sciences, Novosibirsk, 630090 Russia*

Received December 16, 2020; revised February 19, 2021; accepted March 4, 2021

Abstract—A method for visualizing multivolume data and functionally defined surface using graphics processing units is proposed. The method provides visualization of a large number of volumes, a complex of semi-transparent and functionally defined objects, of complex semi-transparent volumes, including intersections of volumes in constructive solid modeling. Simultaneous rendering of different volumes is more difficult than rendering of a single volume, because both intersecting and blending operations are required. Functionally defined surfaces are well suited for embedding external objects into volumes.

DOI: 10.3103/S875669902102014X

Keywords: *functionally defined surface, perturbation functions, multivolume rendering, ray casting, constructive solid geometry (CSG), CUDA, graphics processing unit (GPU)*

INTRODUCTION

The task of visualization lies in making the objects consisting of data arrays or several connected data sets visible to observer. One of the important problems of volume rendering is the visualization of samples of a three-dimensional scalar field obtained from such sources as x-ray computed tomography or ultrasonic scanners. It is also of current interest to visualize structural features, in particular, the boundaries of regions with identical scalar values different from the scalar values themselves.

The visualization process for sets of different data conceptually includes the three stages of transformation to obtain the images:

1. Classification. This stage transforms the original data set to a new set of classes of objects explicitly containing the information which we want to see.

2. Modeling. Transformation of feature maps of sets to the objects of the visual model.

3. Rendering. Visualization of the objects in the model.

Dependent on the specific realization, the order of these transformations may be changed or their union may be carried out.

At the classification stage the interesting features are separated that can be, for instance, boundaries of regions with identical values. The classification may be binary or non-binary. On the one side, binary classification gives the all-or-nothing results with regard to probabilities of existence. On the other side, non-binary classification may give any value in the range from 0 to 1. The binary and non-binary classifications have their advantages and disadvantages.

At the modeling stage the extracted set of attributes is put in correspondence to the set of attributes of visual information carriers in the visual model. The visual model is the modeled 3D scene in which information-carrying visual primitive 3D objects, the voxels, created with non-binary classifications, are

*E-mail: sivser@mail.ru

placed. A voxel may have different attributes, including non-transparency and emmitance that can have color components.

The third stage of transformations is the object rendering. Ray casting is the most widespread approach to rendering volume due to its flexibility in generating a high-quality image [1, 2]. A large volume of homogeneous data containing in the volume model can be well parallelized in computations using the software-hardware shaders of the graphics processing unit (GPU). Today, a common graphics processing unit is able to map a single volume of high-resolution data with a high framing rate. However, mapping of a single homogeneous volume is not suitable for many applications. Using a combination of different volume data sets becomes popular in medicine applications, where both anatomical and functional structures are required. The multivolume rendering is the rendering of several volumes. Thus, the multivolume rendering is needed at processing several data sets which usually have different orientation and resolution. In medicine visualization such methods for obtaining anatomical images as computed tomography, magnetic resonance imaging (MRI), and ultrasonography and such methods for obtaining functional images as positron emission tomography, single-photon emission computed tomography, and functional MRI are used to examine a patient [3, 4]. Moreover, models of medicine tools and results of segmentation are required for computer assisted surgery. Modern methods have limited capabilities for interactive study of multivolume data in combination with polygonal models. Simultaneous rendering of several volumes in combination with surfaces is more difficult to perform than visualization of individual volumes because of processing volume overlaps and color mixing [5, 6]. Recomputation of volumes in a unified coordinate system is disadvantageous, because it leads to loss in quality or to increasing memory requirements. The combination of volumes with surfaces is the necessary requirement in virtual surgery. Functional models are well suited for embedding extrinsic objects (tools, implants, etc.) into volume data.

The current work is aimed at creating the method for visualization of multivolume data and functionally defined surfaces for medicine applications.

APPROACH TO DEFINING SURFACES

Visualization technologies for medicine applications apply high-resolution description of object surfaces using the (second-order) functions of deviation from basic quadric [7]. The function is defined by a second-order algebraic inequality with three unknowns x, y, z in the form $F(x, y, z) \geqslant 0$. We consider the surfaces to be closed subsets of Euclidean spaces defined by the circumscribing function $F(x, y, z) \geqslant 0$, where F is a continuous real function, x, y, z is a point in $\mathbb{E}^3$ prescribed by coordinate variables. Surfaces are constructed using quadrics and represented by composition between the basic quadric and perturbations:

$$F'(x, y, z) = F(x, y, z) + \sum_{i=1}^{N} f_i R_i(x, y, z), \tag{1}$$

where f_i is the form factor and $R_i(x, y, z)$ is the perturbation:

$$R_i(x, y, z) = \begin{cases} Q_i^3(x, y, z) & \text{if} \quad Q_i(x, y, z) \geqslant 0; \\ 0 & \text{if} \quad Q_i(x, y, z) < 0, \end{cases} \tag{2}$$

where $Q_i(x, y, z)$ is the perturbation quadric.

The geometric model prepares conditions for constructing objects and their compositions of different complexity. For this purpose, we use the set of geometric operations Φ mathematically defined as follows [8]:

$$\Phi_j: \quad M^1 + M^2 + \cdots + M^n \to M. \tag{3}$$

To generate models of complex objects on the basis of perturbation functions, we use set-theoretic operations. They are carried out using Boolean operations of union and intersection. A binary operation $(n = 2)$ (3) of objects G_1 and G_2 means the operation $G_3 = \Phi_j(G_1, G_2)$ with definition

$$f_3 = \psi(f_1(\mathbf{X}), f_2(\mathbf{X})) \geqslant 0, \tag{4}$$

Fig. 1. Functionally defined implants.

where ψ is the continuous real function of two variables.

The performed experimental investigations showed that the compression coefficients of polygonal objects varied from 10 to 100 and higher in dependence on test models of different degree of refinement [9]. Thus, we may reduce the database of implants (Fig. 1).

RENDERING FUNCTIONALLY DEFINED SURFACES

This task is similar to methods of visualizing volume data which are frequently used in volume visualization (in such approaches the density function is prescribed). Their main difference is in the presence of discrete data. In our case there is an analytically defined density function, which allows more efficiently searching intersection points between rays and surfaces. To make the program working efficiently, we need to take into account peculiarities of the applied hardware. Recently, increase in the computational capacity occurs due to parallel computations. Therefore, the method is adapted for using computational resources of the graphics processing unit. In [10] a method was described for visualizing functionally defined objects without recursive division of the object space over the quaternary tree in which the cube is divided into parts in the plane XY according to the pixels in the image. Such approach was chosen due to specificity of architecture of graphics processing units in which conditional branches are costly operations. However, we here need to scan the entire screen space by rays even if the projection of the object onto the screen is small. In addition to conditional branches, there exists the balancing problem and overhead costs to its solution exceed the gain in performance due to recursive division of the object space over the quaternary tree. Due to a large number of processing units, testing of several rays is performed simultaneously. In most graphics processing units supporting CUDA (software-hardware architecture of parallel computations, which makes it possible to increase the computing performance due to use of the Nvidia graphics processing units), there exists at least 128 scalar computing cores. Consequently, a relatively large part of the cube is removed. Nevertheless, for small and moderate objects such method is not efficient. Moreover, it badly suits the combination rendering of surfaces and volumes, which will be mentioned further.

Hence, another method was developed [11]. The rasterization process is divided into two stages and is distributed between the central processing unit (CPU) and the graphics processing unit. The CPU executes division of the object space into quaternary tree to a cell of a certain size. The advantage of this approach consists in the possibility of removing large parts of the cube without the given object at earlier stage. The primitives of intermediate description are fragments of intersection between geometric objects and cells. Under fragments we understand parts of surfaces (1) and volumes belonging to a particular cell. All remaining computations are executed on the graphics processing unit.

SEMI-TRANSPARENT VOLUMES

In the course of scanning of the scene over the z coordinate (scanning of the volume in depth), the algorithm works until the volume is completed or a certain value of opacity larger than some threshold value is accumulated. The difference from the ray tracing algorithm is that we do not track secondary rays in order to increase the computation speed. In addition, in modeling the light passage through semi-transparent volumes, we ignore refraction and decay of secondary rays. Just the reflection and decay of light at its passage from the object to the eye of observer remain in this model.

Fig. 2. Examples of volume images with different priorities.

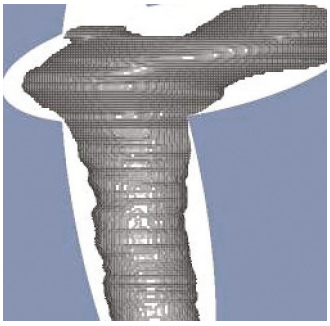

Fig. 3. Three-dimensional texture bounded by a shell in the form of union of two quadrics.

The formula for computing the pixel color may be expressed as [12]

$$P_\lambda = \sum_{n=0}^{N} I_{\lambda n} \Omega_n \prod_{m=0}^{n-1} (1 - \Omega_m), \tag{5}$$

where P_λ is the final color of pixel, λ is r, g, or b color (red, green, or blue color, respectively), $I_{\lambda n}$ is the intensity in the nth voxel computed by the Phong lighting model, Ω_n is the opacity of the nth voxel, $I_{\lambda 0}$ is the reflected light from the first point on the scanning ray, $I_{\lambda n}$ is the background color, and $\Omega_n = 1$. Passing the threshold may be traced as follows: if at the kth step the total transparency $(1 - \Omega_0)(1 - \Omega_1) \ldots (1 - \Omega_{k-1})$ becomes less than some ε, then it means that the contribution of all voxels following after the kth voxel is small; therefore, scanning may be terminated.

There exists a problem with intersecting transparent objects due to necessity of choice of the spatial region belonging to both bodies. How can we choose the material of the intersection region without solving the problem of logical construction? The solution to this problem consists in assigning priorities to the objects of the scene (Fig. 2).

In this work we use quadrics and unions of quadrics as bounding shells for volume data (Fig. 3). How the shell is automatically constructed is the task similar in its complexity to the segmentation task and is outside the scope of this work.

HYBRID RENDERING OF VOLUMES AND SURFACES

The method allows visualizing several volumes in combination with functional surfaces. We apply the visualization method of [11] as the basic method adapted to volumes. As we have described above, the rasterization process is divided into two stages. The first stage is terminated when the considered part corresponds to a cell of a certain size. The primitives of the intermediate description are fragments of intersection between the geometric objects and cells (of the equation of surfaces). The second stage of computations is in processing a list of objects and determining the visibility and color of pixels. The ray casting of the image with complex intersections between volume data and surface is applied and the hierarchical traversal of the space is used for computing the color in the pixel. In other words, the process of space processing is as follows: the volume is traversed over the quaternary tree (in the screen space, the CPU function) whose leaves are binary subtrees (it corresponds to rays directed inside the screen, the GPU function).

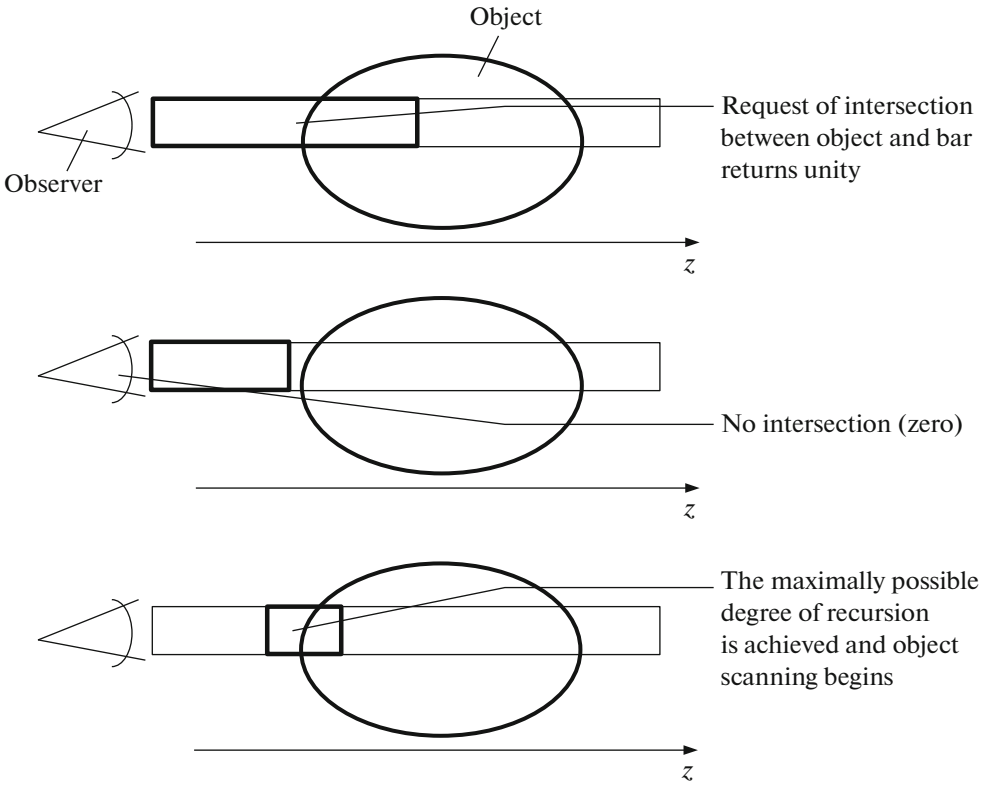

Fig. 4. Binary division of a ray.

In the course of binary traversal, we determine the priority over the z coordinate in the case of quadrics and for accurate computation of the perturbation factor in the case of functionally defined surfaces. In the course of division, large parts of empty space are rather efficiently sorted out at the levels of both the quaternary division and the binary division (Fig. 4). After determining the intersection with the shell, its internal part filled with the 3D texture is scanned.

The volume data are stored as a three-dimensional texture [13]. We use the 3D texture map for volume data sets. The texture map is a single large 3D texture which is assigned to a single texture block. The maximum size of the texture map depends only on available graphical memory. All computations are executed on the graphics processing unit.

The graph of scene includes operations of constructive solid geometry and functionally defined surfaces with volume textures and material properties. The inner vertices of the tree are related with operations of solid geometry, and the leaves refer to the shell bounding the volume, to the volume texture, and to the material with its properties. In Fig. 5 we show the union of an oblique object (implant) with several semi-transparent volumes. Processing fragments is carried out in parallel-pixel mode for cells with a size of 8×8 pixels and traces the passage of rays through the volumes presented as 3D textures. To increase the computation speed, we use the cache of coherency of textures. The rendering system passes the scene once in contrast to multipass methods [14, 15]. In the course of the binary division of a ray, we determine the object nearest to the observer and the nearest point of intersection with the surface, thus providing the sorting of objects nearest to the observer. As a result, the color in a pixel is accumulated from the close objects to the further ones, that is, in the direct priority order, in the case when the objects are semi-transparent. Oblique surfaces veil objects further from the observer. The final ray tracing and shadowing require blending of several intersecting volume objects and efficient accumulation of individual volume contributions along the rays.

RESULTS

To estimate the performance, we compare the proposed approach with the known approaches to multivolume visualization [14, 15] using the graphics processing unit GTX 470. In Table 1 we compare

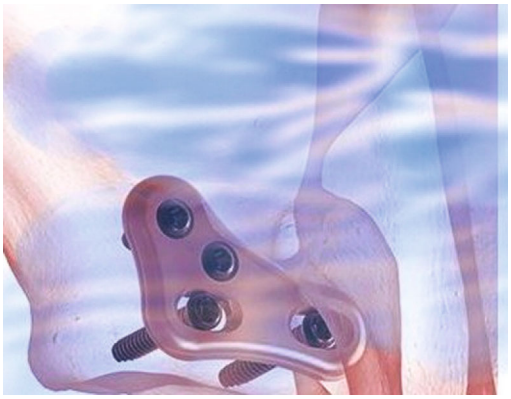

Fig. 5. Visualization of combined anatomical 3D model (tissues and bone structure are given by volume data (3D texture) and the implant is given by functionally defined surfaces).

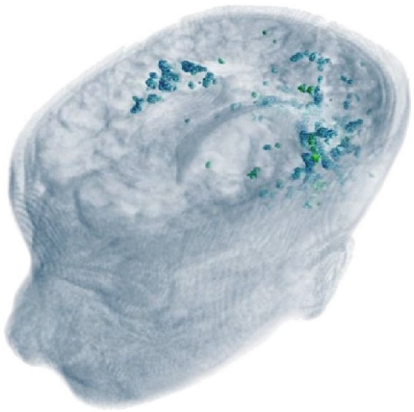

Fig. 6. Multivolume data: skull, brain, tumor (upper part of brain), and head tissues.

the framing rates (the number of frames per second) reached for medicine scenes with several volumes which use several modes and scans for different body parts. Tests 1 and 2 show the resolution of volumes and the time of anatomical scanning (Fig. 6). Test 3 is a combination of anatomical scanning of high-resolution volumes and isosurfaces for methods of [14, 15]. In the proposed method we use a combination of the volume and functional surface. Test 4 is a combination of volumes and a polygonal model with a size of 3.87×10^3 as an analog of the functional surface.

Comparing to the known approaches, the main advantages of the proposed method include the following ones:

1. Generality: using this method, we can process not only functionally defined surfaces, but also semi-transparent volumes.

Table 1. Table 1

Test number	Volumes	Polygons	[14]	[15]	Proposed method
1	1×512^3	–	14–46	9–17	30
2	2×256^3	–	24–38	6–12	37
3	4×256^3	–	5–9	1–3	25
4	10×64^3	3.87×10^3	–	–	16

Functionally defined objects are considered in the proposed method; there are five such objects in test 4.

2. Complex models: the method provides visualization of a large number of volumes, a complex of semi-transparent and functionally defined objects, and complex semi-transparent volumes, including intersections of volumes.

3. Efficiency: using the functionally defined surfaces allows reducing the database of implants and decreasing the time of rendering.

4. Simplicity: the method is relatively simple in realization and has no problems with reliability or degeneration of surfaces.

5. Well approximation: preliminary application of the method for various test samples shows well approximation with a limited error at the boundaries of intersections between volumes and functionally defined surfaces.

CONCLUSIONS

In this work we created a new approach to visualizing several semi-transparent volumes and functionally defined surfaces. The method, including the rasterization stage, was implemented on CUDA, which allows completely controlling the memory hierarchy, in particular, the access to the high bandwidth capacity and to the low latency time of distributed memory. An interactive framing rate at simultaneous visualization of more than hundred arbitrarily overlapping volumes is provided. Computation takes several tens of milliseconds on processing units Intel Core2 CPU E8400 3.0 GHz and GPU GTX 470.

The method is topical for a wide spectrum of application of the volume rendering of complex scenes, including intersecting volumes and surfaces for medicine applications. As shown in this work, the method is simple in its implementation and allows using it to solve many problems.

REFERENCES

1. J. Kruger and R. Westermann, "Acceleration techniques for GPU-based volume rendering," in *Proc. of the 14th IEEE Conf. on Visualization, Seattle, USA, 2003*, pp. 38–43. https://doi.org/10.1109/VISUAL.2003.1250384
2. S. Stegmaier, M. Strengert, T. Klein, and T. Ertl, "A simple and flexible volume rendering framework for graphics-hardware-based ray casting," in *Proc. of the 4th Eurographics, IEEE VGTC Conf. on Volume Graphics, New York, USA, 2005*, pp. 187–195. https://doi.org/10.1109/VG.2005.194114
3. B. B. Avants, N. J. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee, "A reproducible evaluation of ANTs similarity metric performance in brain image registration," Neuroimage **54**, 2033–2044 (2011). https://doi.org/10.1016/j.neuroimage.2010.09.025
4. R. J. Killiany, T. Gomez-Isla, M. Moss, R. Kikinis, T. Sandor, F. Jolesz, R. Tanzi, K. Jones, B. T. Hyman, and M. S. Albert, "Use of structural magnetic resonance imaging to predict who will get Alzheimer's disease," Ann. Neurology **47**, 430–439 (2000). https://doi.org/10.1002/1531-8249(200004)47:4<430::AID-ANA5>3.0.CO;2-I
5. A. Leu and M. Chen, "Modelling and rendering graphics scenes composed of multiple volumetric datasets," Comput. Graph. Forum. **18**, 159–171 (1999). https://doi.org/10.1111/1467-8659.00366
6. S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Groller, "Flexible direct multi-volume rendering in interactive scenes," in *Proc. of the Vision, Modeling, and Visualization Conference (VMV 2004), Stanford, USA, 2004*, pp. 386–379.
7. S. I. Vyatkin, "Complex surface modeling using perturbation functions," Optoelectron., Instrum. Data Process. **43**, 226–231 (2007). https://doi.org/10.3103/S875669900703003X
8. S. I. Vyatkin, "Conversion of functionally defined forms," Software Syst. Comput. Methods, No. 4, 489–499 (2014). https://doi.org/10.7256/2305-6061.2014.4.13982
9. S. I. Vyatkin and B. S. Dolgovesov, "Compression of geometric data with the use of perturbation functions," Optoelectron., Instrum. Data Process. **54**, 334–339 (2018). https://doi.org/10.3103/S8756699018040039

10. S. I. Vyatkin, "Method of binary search for image elements of functionally defined objects using graphics processing units," Optoelectron., Instrum. Data Process. **50**, 606–612 (2014). https://doi.org/10.3103/S8756699014060090
11. S. I. Vyatkin, "Recursive search method for the image elements of functionally defined surfaces," Optoelectron., Instrum. Data Process. **53**, 245–249 (2017). https://doi.org/10.3103/S8756699017030074
12. G. Knittel, "VERVE: voxel engine for real-time visualization and examination," Comput. Graph. Forum **12** (3), 37–48 (1993). https://doi.org/10.1111/1467-8659.1230037
13. S. I. Vyatkin and B. S. Dolgovesov, "A 3D texture-based recursive multi-level ray casting algorithm," in *Proc. of the 2nd IASTED Int. Multi-Conf. on Automation, Control, and Information Technology Software Engineering (ACIT 2005), Novosibirsk, Russia, 2005*, pp. 92–97.
14. F. Roessler, R. P. Botchen, and T. Ertl, "Dynamic shader generation for GPU-based multi-volume ray casting," IEEE Comput. Graph. Appl. **28** (5), 66–77 (2008). https://doi.org/10.1109/MCG.2008.96
15. R. Brecheisen, B. Platel, A. Vilanova, and B. T. H. Romenij, "Flexible GPU-based multi-volume ray-casting," in *Proc. of the Vision, Modelling and Visualization Conf., Konstanz, Germany, 2008*, pp. 1–6.

Translated by E. Oborin