BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Tomul LV (LIX), Fasc. 3, 2009
Secția
AUTOMATICĂ şi CALCULATOARE

POST-RENDERING ENHANCEMENT OF VOLUMES

BY

MARIUS GAVRILESCU, VASILE ION MANTA and *WERNER PURGATHOFER

Abstract. The paper presents an approach to visually enhance representations of volume data as a means to improve volume visualization. Direct volume rendering is employed to represent several volume data sets, using the popular Ray Casting algorithm. The result is rendered to a texture via an off-screen framebuffer, which then goes through a post-rendering processing stage. This stage involves the application of image enhancement techniques such as the use of spatial filters, to produce clearer, sharper, and less noisy images of the rendered volume. Depending on the specifics of the volumetric data set, post-rendering enhancement may bring forth more relevant visual information or otherwise improve the overall quality of the resulting images.

Key words: volume visualization, Ray-Casting, post-rendering, feature enhancement, image filtering.

2000 Mathematics Subject Classification: 65D18, 68U05.

1. Introduction

The present decade has seen tremendous improvements in computer graphics. Current rendering techniques make full use of revolutionary next-gen Graphics Processing Unit (GPU) architectures, with features such as fully programmable vertex and fragment units, unified shaders and hardware-supported physics computations, which make it possible to produce photorealistic images and detailed high quality animations. Driven primarily by the development of computer gaming, GPUs have significantly increased in performance in the last few years, and, while normally designed for the fast rendering of 3D scenes, their applications and potential extend much further [1].

GPUs with advanced functionality are of particular use in volume rendering applications, where the developer can take advantage of the parallelism of per-pixel processing stages to implement direct volume rendering

techniques, over-sampling or advanced isosurface extraction methods to produce high quality suggestive renderings.

Volume visualization involves the representation of the parameters, features and details of volumetric data sets using visual cues encoded primarily in the color and opacity of rendered images of a volumetric object. Volume data comes in all shapes and sizes, at various resolutions and containing various details within. Volume visualization techniques are of particular interest in medical imaging, where 3D datasets produced by Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) scanners can be represented through volume rendering approaches, as opposed to the grayscale slice-based views traditionally used in radiology. A classification of the data contained within a data set is required to isolate the information useful to the potential user. After classifying the available data, however, it is often necessary for the resulting information to be displayed in a way that is intuitive, accentuates or otherwise enhances features of interest inside the volume, and is editable in real-time, so as to provide the user with the means to customize the way the data is eventually displayed. Several past enhancement or illustrative approaches relied on modifications during the 3D rendering stage, such as importance driven feature enhancement [2], sphere map-based illustration [3], or the isolation of specific features through the rendering of segmented volume data [4]. In this paper, however, we explore enhancements which take place after the generation of a 2D image of the 3D volumetric object, in what will henceforth be referred to as the *post-rendering* stage of volume visualization. Using methods typical to 2D image processing, the images produced by 3D volume rendering can be further modified or improved, in real time, at the cost of some additional computing power.

2. Volume Rendering Approach

The input data for the volume rendering loop are 8 bit CT data sets where the sampled points are structured into a regular grid. The data is fed into video memory as a 3D texture, where a continuous volume is reconstructed via hardware-supported trilinear interpolation.

For the volumes considered in the present paper, we employ a direct volume rendering approach using the popular Ray Casting algorithm. It is an image-order approach which is extensively used in volume graphics due to its applicability in fragment programs and because it allows the developer to take full advantage of the parallel processing capabilities of the GPU. A ray is projected from each point in image space through the volume, and the volume is sampled along each ray. Through a means of classification, the sampled points for each ray receive colour and opacity values. Each pixel in image space is constructed by compositing the colours and opacities of sampled points along the ray which corresponds to that pixel. Fig. 1 shows the basic principle of the Ray-Casting method.

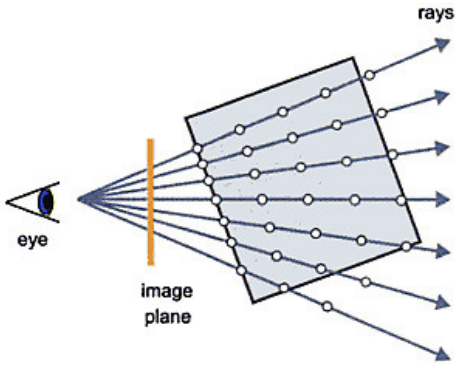

Fig. 1 − The Ray Casting algorithm [5].

The most common means of classification employs a transfer function which maps the density values of the points inside the volume to RGBA quadruplets, representing color and opacity. The transfer function can take a variety of forms and allows great flexibility in editing out irrelevant parts of the volume and outlining/colouring features of interest. Fig. 2 shows a rendering of a CT skull data set which employs a piecewise linear transfer function with color values assigned to each of its linear segments. Irrelevant points from the environment surrounding the actual skull are rendered invisible, while high density data such as the jaw or the teeth are shown in red and mid-density values are light green.

Transfer function specification is done manually via a user interface as seen in Fig. 3. In the background, the grey histogram of the volume shows voxel distributions among density values. The user may modify the shape of the function by dragging the points in between its linear segments. The function itself controls opacity, thus points which are “higher up” on the graph will cause the regions of the volume with density values near those points to render more opaque. A colour can be assigned to the range of density values corresponding to each individual segment.

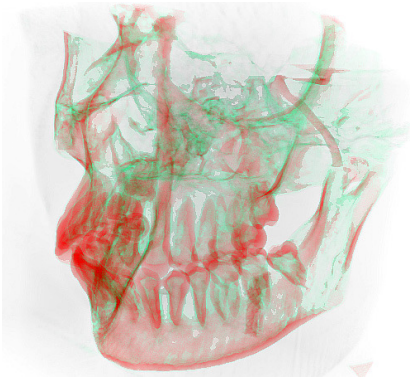

Fig. 2 − Rendering of the skull dataset.

This allows a significant degree of control over what is being displayed, as the parts of the volume which are not needed can simply be rendered invisible, while regions of interest can be emphasized through an appropriate combination of colour and transparency.

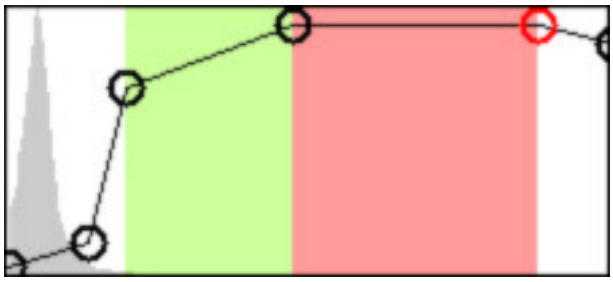

Fig. 3 − Interactive graph for transfer function specification. The horizontal axis contains density values, while the vertical axis corresponds to opacity levels.

Naturally, there are more sophisticated means of specifying a transfer function. This allows, in the case of medical datasets for instance, a concurrent display of several anatomical features, each with its specific colouring and shading [6]. A more advanced means of transfer function specification is provided by Bruckner *et al.* [7], where the data is divided across multiple property domains, and each mapping via the transfer function is done independently for each domain.

One additional approach for volume visualization is the detection and rendering of isosurfaces. An isosurface is formed by the points of the volume which have the same density value. In the case of a medical dataset, this could be regions of bone, skin, the wall of a blood vessel, etc. Isosurfaces are a very common method for displaying volumes, because, in many cases, only the outer layers of the structures inside the volume are of interest. For instance, in the case of the rendering in Fig. 2, the potential user might only be interested in the shape of the skull and not necessarily in its interior structures.

Several approaches for detecting and rendering isosurfaces rely on the Marching Cubes algorithm, which analyses the volume eight points at a time and determines whether an isosurface passes inbetween the points, as well as the polygons needed to construct that isosurface [8].

Our approach for isosurface detection is based on a method developed by Hadwiger *et al.* [9], which makes use of the Ray-Casting algorithm and exploits the possibility to implement adaptive sampling along the projected rays. Thus, for each ray, the sampling points between which the isosurface passes are isolated. Afterwards, several iterations of a refinement algorithm detect the point on the actual isosurface with good precision. The result is visible in Fig. 4, which features the same dataset as in Fig. 2, only in this case, a single isosurface is detected and shaded accordingly. The threshold which

determines the point of detection is adjustable, rendering visible isosurfaces corresponding to various densities.

Once detected, various enhancements may improve the quality of the isosurfaces, such as the implementation of illumination. Per-pixel lighting adds significant visual detail and realism to the resulting images. In Fig. 4, we implemented a local illumination model using Phong shading [10]. More elaborate and optimized lighting approaches may of course be employed, such as interpolated pre-integrated lighting [11].

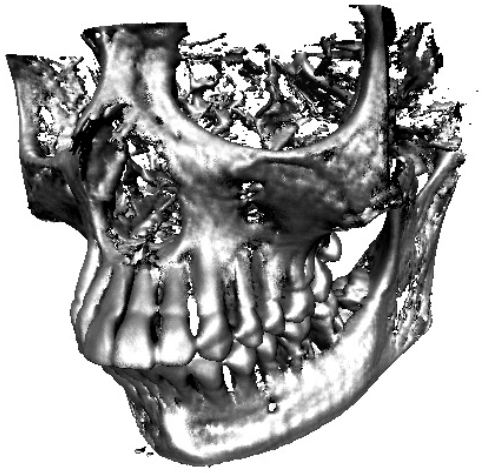

Fig. 4 − Rendering of an isosurface from the skull data set. Local illumination and specularity add to the definition and realism of the resulting image.

3. Post-Rendering Processing Approach

The methods discussed in Section 2 took place within 3D rendering. However, the resulting images can be further manipulated by means of 2D image processing, in the post-rendering stage. Post-rendering enhancements refer to the accentuation or sharpening of features such as edges, boundaries or contrast, to make the resulting images more useful for display or analysis.

First, we render the 3D volume to a texture by means of a framebuffer object and then we use a shader program to extract texel colour values, which in this case are an exact match to the pixels in screen space. Since the texture is updated for each iteration of the 3D rendering loop, this essentially allows real-time dynamic access to the colour values of on-screen pixels, thereby allowing for the "on-the-fly" implementation of many 2D image modifications and enhancements.

Image processing for enhancement purposes can take place in either the spatial domain, where the conventional structure of the image (an array of pixels) is taken into account, or in other domains, the most common of which is

the frequency domain. In the spatial domain, individual pixel values may be combined or compared with other neighbouring pixels in various ways [12]. These enhancements do not add to the information contained in the rendered image, but increase the dynamic range of certain features, which can be detected more easily. Also, most of the 2D image enhancement techniques are empirical and may require several attempts by the user to produce satisfactory results [13]. While, for the purposes of this paper, these enhancement techniques are used in post-processing, they may also serve as pre-processing tools in other kinds of applications, in particular for image restoration or better automated interpretation [14].

One of the most basic spatial image processing approaches is brightness and contrast manipulation. The brightness modifier raises or lowers the overall intensity of pixel colours, while contrast accentuates any differences between neighbouring regions of the image. For images resulting from volume rendering, this often enhances details which are otherwise harder to see.

Both modifiers are combined together into a single formula presented in Eq. 1, where *newColour* is the colour resulting from modifying the original *oldColour* of the pixel. Note that all values are normalized, *i.e*. clamped to the [0, 1] interval. Eq. 1 is applied to each of the three colour channels, Red, Green and Blue (RGB), for every pixel, individually.

$$newColour = (oldColour - 0.5) \cdot contrast + brightness + 0.5 \tag{1}$$

Fig. 5 illustrates the effect that brightness and contrast have on the resulting images.

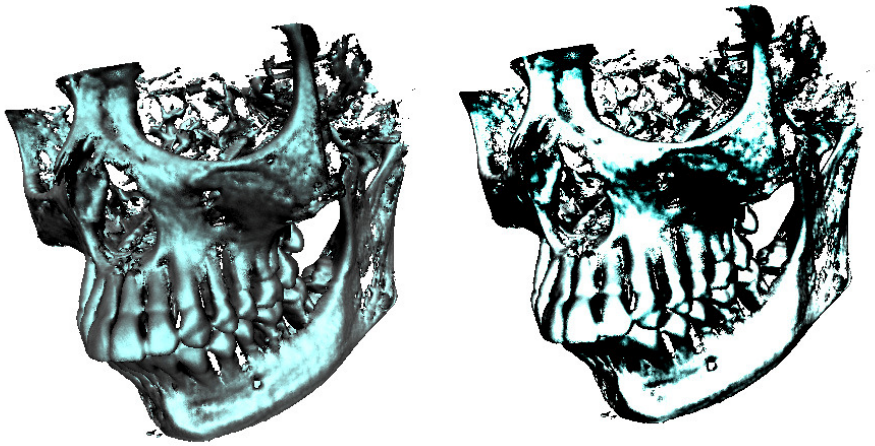

Fig. 5 −Renderings of the same dataset, unmodified (left) and with high post-rendering contrast and brightness enhancement (right).

High contrast results in accentuated edges, while similarly colored regions are made homogeneous. This effect may be combined with curvature-

based volumetric edge detection techniques for a better resulting representation of the volumetric object's outlines. Real-time processing and the ability of the user to change contrast and brightness values dynamically (using a slider based interface, for example) allow for significant customization as far as enhancements are concerned.

A greater degree of freedom in image manipulation is granted through the use of spatial filters. These involve the use of a subimage referred to as a *filter* or *kernel*, essentially a matrix of scalar coefficients which is moved from point to point along the image. At each point, the colour of the corresponding pixel is given by the sum of products of the filter coefficients and the corresponding image pixels from within the area covered by the filter mask [15]. Filters are typically square matrices of odd sizes, the most common being 3x3, 5x5 and 7x7. Filters larger than 5x5 come with significant computational costs and are impractical for real-time applications. Eq. 2 illustrates the operation of filters in the spatial domain for the particular case of a 3x3 filter.

$$q(x,y) = \frac{1}{\sum_{i=-1,j=-1}^{1,1} w(i,j)} \sum_{i=-1}^{1} \sum_{j=-1}^{1} w(i,j) \cdot p(x+i, y+j) \tag{2}$$

In Eq. 2, p is the colour value of a pixel located at position (x, y) somewhere inside the image space and $w(i, j)$ are the values of the 3x3 filter matrix, which range from −1 to 1. Note the division between the sum of all values in the filter, which prevents the brightness of the image from being changed after applying the filter. The result accumulates into the new pixel value, q.

There are numerous filters, each with multiple variations, which can be applied to 2D images. For the purposes of demonstration, we limit the discussion to a few of the more widely used filters. Most commonly, 3x3 filters are best suited to real-time processing, since they come with the lowest computational cost, but 5x5 filters are also useable, depending on what other features and optimizations are implemented. For 7x7 filters and above, the significant impact on performance may not justify the higher precision and control granted by a larger filter mask.

A smoothing filter evens out areas, and bridges gaps in lines or curves, providing a blurred look. Also, for real time applications such as in the present case, smoothing filters are a cheap effective noise removal tool, since other more popular noise removal methods (such as median filters) are often too computationally intensive for real-time use. Fig. 6 shows the results of applying such a filter. Note that image processing is done in real time, right after 3D volume rendering.

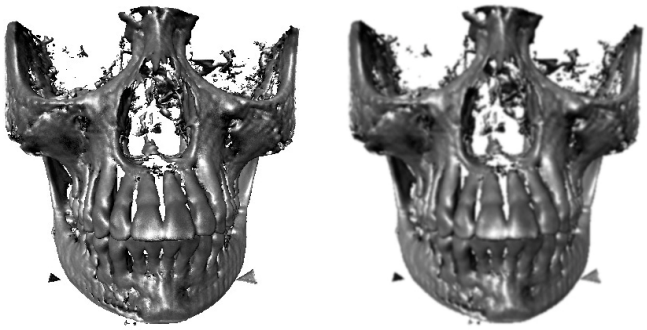

Fig. 6 − Comparative view of the original (left) and the smoothed (right) renderings.

On the opposite end of smoothing filters are sharpening filters. These modifiers accentuate contours and provide a sharper, better defined look, as seen in Fig. 7. Intricate details and curvatures are rendered significantly more visible. However, this filter also accentuates any noise, as seen, for instance, on the two front-most teeth from the right-side skull rendering in Fig. 7.

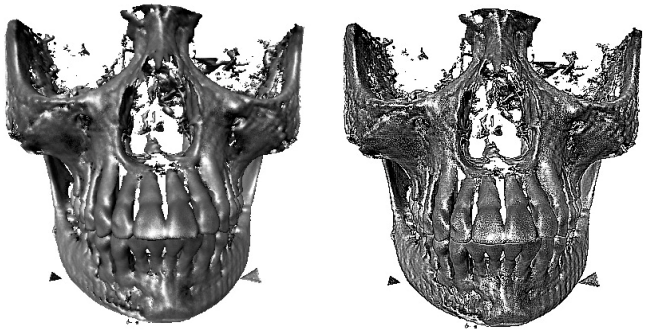

Fig. 7 − Sharpening renders contours more visible on the right-side rendering.

Edge detection can be easily and conveniently achieved using an appropriate "find edges" filter, as demonstrated in Fig. 8.

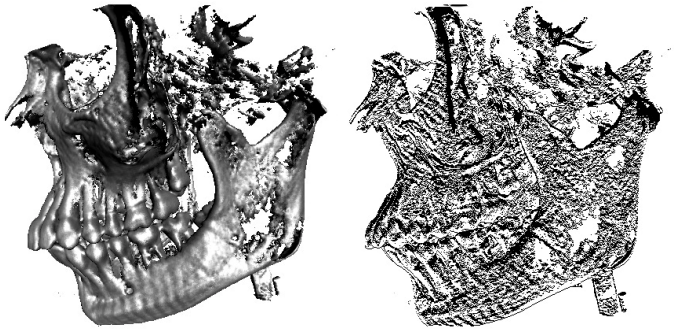

Fig. 8 − The edge detection filter isolates regions of abrupt transition between colour values, as seen on the right-side rendering.

A popular enhancement technique more commonly used for generating bump-maps and for artistic purposes is the emboss filter, which produces a 3D-like shadow effect. The result can be seen in Fig. 9.

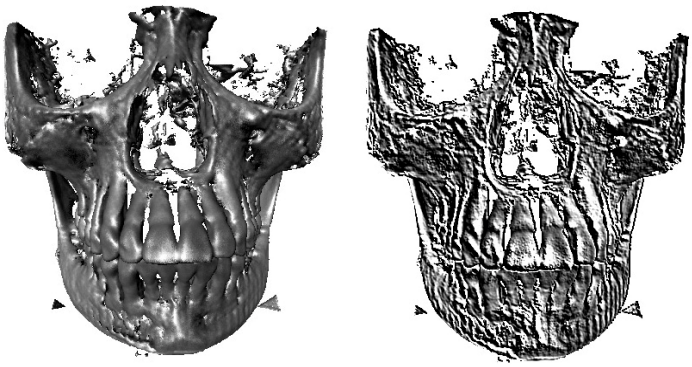

Fig. 9 − The emboss filter generates a bump-map effect in the right-side rendering.

4. Performance Analysis

As an image enhancement technique, 2D filtering may prove a viable alternative to other types of processing particularly found in volume rendering approaches. Still, it is important for volume visualization applications to maintain a real-time component, so as to allow dynamic inspection, editing and customization of the volumes and the resulting rendered images. Table 1 illustrates the performance of the filtering post-rendering stage using various sized filters at various resolutions. The test machine consisted of an Intel Core2Duo at 2.2 GHz, 2GB RAM and a GeForce 8800 GT based video card.

Table 1
Performance of Post-Rendering Processing (Frames per Second)

Resolution	Non-filtered	Filter size		
		3x3	5x5	7x7
800x600	51.4 fps	48.6 fps	38.5 fps	29.3 fps
1024x768	42.3 fps	37.3 fps	28.2 fps	20.5 fps
1280x1024	35.1 fps	29.2 fps	22.7 fps	14.3 fps

Table 1 shows that as the size of the filter increases, performance degrades significantly, which is mostly due to the exponential increase in the count of per-fragment operations. Other issues to take into account when measuring performance are the resolution of the volumetric dataset and whatever volume rendering optimizations are in place (such as empty space skipping, adaptive sampling or early ray termination).

5. Conclusions

The proper enhancement of volume renderings can be a problematic topic, given the subjective nature of what constitutes a "good" image. The usefulness of the approaches discussed in the latter half of the paper depends on what results the developer is aiming for. Since volumetric data sets are very varied, post-rendering enhancement may or may not be a viable option when trying to come up with the most suggestive images. It is therefore up to each individual developer to decide whether the benefits gained from post-rendering image enhancement out-weigh the drop in frame rate.

However, in many situations, for purely illustrative, feature enhancement or noise removal purposes, 2D image processing techniques applied in the post rendering stage may prove useful, if not indispensable, and may very well cost less computational power than 3D based counterparts. Using various combinations of spatial or even frequency-domain filters may accentuate important details in the final rendering. Thus, the combination of 2D and 3D approaches in the graphical processing of volume data, as a solution for volume visualization, may reveal information not easily visible with only one of either approach.

A c k n o w l e d g e m e n t s. This paper was elaborated within the project *BRAIN-Doctoral Scholarships, an Investment in Intelligence* at the "Gheorghe Asachi" Technical University of Iaşi, Romania.

The paper was supported by the CNCSIS project ID_342/2008, Romania.

Received: July 8, 2009

"Gheorghe Asachi" Technical University of Iaşi,
Department of Computer Engineering
e-mail: vmanta@cs.tuiasi.ro

**Vienna University of Technology*
Institute of Computer Graphics and Algorithms
e-mail: wp@cg.tuwien.ac.at

R E F E R E N C E S

1. Weiskopf D., *GPU-Based Interactive Visualization Techniques.* Springer-Verlag Berlin Heidelberg, 2006.
2. Viola I., Kanitsar A., Gröller E., *Importance-Driven Feature Enhancement in Volume Visualization.* IEEE Transactions on Visualization and Computer Graphics, **11,** 408−418 (2005).
3. Bruckner S., Gröller E., *Style Transfer Functions for Illustrative Volume Rendering,* Computer Graphics Forum, **26,** 715−724 (2007).
4. Hadwiger M., *High-Quality Visualization and Filtering of Textures and Segmented Volume Data on Consumer Graphics Hardware.* Ph. D. Diss., Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2004.
5. Engel K., Hadwiger M., Kniss J.M., Rezk-Salama C., Weiskopf D., *Real-Time Volume Graphics.* A K Peters, Wellesley, Massachusetts, 2006.

6. Gavrilescu M., Manta V., *Volume Visualization Applied in Medical Imaging.* Bul. Inst. Polit. Iaşi, **LIV(LVIII)**, *3-4*, 43−52 (2008).
7. Bruckner S., Kohlmann P., Kanitsar A., Gröller E., *Integrating Volume Visualization Techniques Into Medical Applications.* Proceedings of 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 820−823, Paris, May 2008.
8. Dietrich C.A., Scheidegger C.E., Schreiner J., Comba J.L.D., Nedel L.P., Silva C.T., *Edge Tranformations for Improving Mesh Quality of Marching Cubes.* IEEE Transactions on Visualization and Computer Graphics, **15,** 150−159 (2009).
9. Hadwiger M., Sigg C., Scharsach H., Buhler K., Gross M., *Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces.* Computer Graphics Forum, **24,** 303−312 (2005).
10. Rost R.J., *OpenGL Shading Language.* 2nd Editon, Addison-Wesley Professional, 2006.
11. Lum E., Wilson B., Ma K., *High-Quality Lighting and Efficient Pre-Integration for Volume Rendering.* In Eurographics/IEEE Symposium on Visualization, 2004.
12. Russ J.C., *The Image Processing Handbook. Third Edition*, CRC Press, 1998.
13. Jain K.J., *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.
14. Tinku A., Ray A.K., *Image Processing – Principles and Applications.* John Wiley & Sons, Hoboken, Jew Jersey, 2005.
15. Gonzales R.C., Woods R.E., *Digital Image Processing.* 3rd Edition, Prentice Hall, 2007.

ACCENTUAREA VOLUMELOR ÎN FAZA DE POST-RENDERIZARE

(Rezumat)

Lucrarea prezintă tehnici de accentuare a reprezentărilor de date volumetrice, cu scopul de a îmbunătăți vizualizarea volumetrică. Se utilizează renderizarea volumetrică directă prin intermediul algoritmului *ray casting* pentru a reprezenta *data set*-uri volumetrice. Rezultatul este renderizat într-o textură, care apoi este supusă unor procesări ulterioare. Acestea implică utilizarea de tehnici de procesare de imagini, cum ar fi aplicarea filtrelor spațiale, pentru a produce imagini mai clare şi cu mai puține perturbații. Funcție de natura *data set*-ului reprezentat, această abordare poate scoate în evidență informații utile sau îmbunătăți calitativ imaginile rezultate.