DISPLAY OF OUTLINES, CONTOURS AND SURFACE FEATURES FROM VOLUME DATA

BY

MARIUS GAVRILESCU*

“Gheorghe Asachi” Technical University of Iași,
Faculty of Automatic Control and Computer Engineering

Received: July 30, 2012
Accepted for publication: September 12, 2012

Abstract. Volume visualization is an ever-growing field of computer graphics with multiple applications in the representation and analysis of 3D scalar data. Such data often comes with a high degree of complexity, making it a challenging task to render a proper 2D image which highlights relevant information while discarding other less significant data. In this paper, we present techniques for the rendering, identification and representation of various features which are meaningful to the human vision system, such as contours, outlines or various surface shapes. We use several approaches, based on the orientation of gradient vectors or on image processing techniques, and show how they can be successfully employed to highlight relevant details from volume data. We illustrate the differences between images obtained through 3D rendering directly and those subjected to contour and feature identification techniques and show how these provide a better visual understanding of the underlying data.

Key words: volume rendering, contour.

2010 Mathematics Subject Classification: 68M99, 68W35.

*Corresponding author; e-mail: marius_official@yahoo.com
1. Introduction

Volume visualization is a field of research which has matured significantly in recent years. The ever-growing need for the efficient and intuitive processing of large bodies of volume data (such as CT and MRI scans (Bushong, 2000; Brown, 2012), ultrasound data (Szabo, 2004), or 3D seismographic surveys (Patel et al., 2010; Gavrilescu & Manta, 2011), together with the rapid development of modern computational means, have greatly stimulated the advancements of 3D computer graphics, particularly the visualization of structured 3D data. Volume visualization has wide applicability in fields such as medical imaging (Kohlmann et al., 2007; Gavrilescu & Manta, 2008), industrial feasibility testing (Malik et al., 2009), or in various environmental engineering disciplines.

The representation and rendering of volume data is vastly used for the visual analysis of three- and multi-dimensional medical scans. By employing various rendering algorithms such as slicing (Yagel et al., 1996), ray casting (Krüger et al., 2003) or marching cubes-based isosurface identification (Montani et al., 1994), a wide assortment of anatomical structures and features can be identified from the underlying data and represented on-screen in a manner which makes it easy to interpret for both doctor and patient. Volume rendering algorithms typically rely on the distribution of color and transparency, depth perception, shadowing, illumination and other visual enhancements, to classify data originating from medical scanners and to generate an image which conveys meaningful information without visual overload. However, the analysis and processing of such data is often a complicated task. For this reason, in the paper we present techniques which visually process and enhance various details found in volume data. We use a variation of the ray casting algorithm to perform 3D volume rendering and incorporate various techniques into the rendering pipeline in order to better highlight various contours, outlines and other such features for a better visual analysis of the resulting images.

2. Volume Rendering Approach

The raw data originating from the scanning device (be it CT, MRI or other) is organized into data sets. These are collections of scalar values which quantify various physical properties of the scanned media in various key positions in 3D space. These positions may be spatially-distributed in various configurations (Jonsson, 2005). In our applications, we rely on data sets arranged in rectilinear structured grids. This means that the scanning device probes the rectangular region of space which circumscribes the scanned object at positions which lie on a 3D rectangular grid. Each region of space corresponding to a grid element is referred to as a voxel. In this configuration, the image of the scanned object is a volumetric construct made up of voxels, in
a manner similar to how a 2D picture is composed of pixels. The concept is illustrated in Fig. 1.

Fig. 1 – Illustration of the volume dataset concept. A volume is an object composed of 3D pixels called voxels. The number of voxels on each axis depends on the resolution of the scan.

In order to represent the data, we rely on direct volume rendering (DVR), where a 2D image is generated from based on the scalar values from the data set, without the need for additional polygonal geometry (as is the case, for instance, in renderers which use slicing). Our method of choice is a variation of the ray casting algorithm (Hadwiger et al., 2005). The volume is first subjected to reconstruction by means of tri-linear interpolation (Carlborn, 1993), a process which generates a continuous representation of the object from the original, discrete data values. Once such a representation is available, rays are emitted from each pixel of the output image into the volume and sets of voxel samples are taken along each ray at a predetermined sampling rate. Subsequently, a classification method (for example, by means of transfer functions (Kindlmann, 2002)) is used to assign optical properties to the sampled voxels. While in the original data set the voxels lack such properties, a transfer function which operates in a non-visual domain (such as densities, gradient magnitudes, curvature values etc.) associates color and opacity values to the voxel samples based on one or more of their non-optical properties.

Fig. 2 shows an image rendered using DVR via ray casting. A transfer function maps color and opacity values to the densities of the materials encountered by the rays. This allows the efficient separation of anatomical structures and tissue types based on differences in their density values. The image from Fig. 1 highlights the basic principle of volume classification: meaningful information is visible, meaning that its constituent voxels are assigned non-zero opacity values, while other irrelevant data is completely or almost completely transparent. What constitutes “meaningful information” is a subjective, case-by case matter. In some situations, certain anatomical features might be relevant (a specific tissue type, a certain organ, a tumor), while others might call for a more generalized, “eagle’s view” of the overall data. Color can be employed to further separate structures of similar or neighbouring densities.
For instance, in Fig. 1, bone tissue is rendered white and fully opaque, while other softer tissues are semi-transparent and colored differently. The use of partial opacity values allows the visual inspection of multiple anatomical structures simultaneously. Through the use of an appropriately-specified transfer function, the distribution of color and opacity throughout the volume can be controlled with substantial accuracy. Transfer functions are commonly specified manually, by means of various user interfaces (Bruckner & Gröller, 2005; Meyer-Spradow et al., 2009).

Fig. 2 – A rendering of a Computed Tomography (CT) data set. Various anatomical structures are identified based on differences in density. Harder tissue such as bone is fully opaque and white, while other, softer elements such as muscle, skin and blood vessels are semi-transparent and differently colored.

While a basic ray casting approach allows for the accurate separation of various elements from within the volume data, it often helps to incorporate additional stages into the rendering pipeline in order to better highlight the desired information. The human vision system does not respond well to numeric data or to discrete, quantifiable information, but, rather, it relies on visual cues such as shapes, contours, outlines and highlights to construct images of the object being viewed. Ray casting, as well as most other volume rendering approaches, does not intrinsically contain means of highlighting such visual elements, thereby requiring that they be explicitly implemented. In the following Sections we present several methods to visually enhance various features from volumes. These are easily integrated into the rendering pipeline, do not interfere with regular classification and rendering and are mostly based on overlaying additional visual information over the results obtained through volume processing and rendering. Specifically, we focus on the highlighting of contours, outlines, edges and other similar enhancements.
3. Display of Outlines Using Gradients

The gradient vector or simply “gradient” expresses the variation of scalar values within a volume. Assuming the volume data set is defined by a function $V(x, y, z) = \text{scalar}$, which maps scalar values to each position in 3D space $(x, y, z)$, the gradient is the first derivative of this function. In practice, a common means of computing the gradient employs the central differences approach (eq. (1)). Each component of the gradient is approximated by differentiating two neighbouring function values at a finite distance $d$.

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

(1)

The direction of the gradient indicates the direction of greatest change in the volume, while its magnitude quantifies the amount of change. The gradient has multiple uses in volume rendering, particularly because it is perpendicular to significant surfaces or bordering regions from within the data. Consequently, its immediate applications lie in the implementation of local illumination algorithms and various means of classification (Engel et al., 2006). However, this property can be successfully employed to highlight contours and silhouettes in rendered images. The general idea is illustrated in Fig 3. In the ideal case, where the volume is a perfect sphere, the outer-most regions of the visible portion of the sphere surface can be highlighted based on the angle between the viewer direction and the gradient (Neumann et al., 2000). The regions where this angle is sufficiently close to $90^\circ$ are facing away from the viewer and correspond to a silhouette of the sphere.

![Fig. 3 – Illustration of the gradient-based silhouette algorithm. The gradient vectors are orthogonal to the surface of the volume and form angles of varying degrees with the user direction. The highlighted area corresponds to regions where the local surface faces away from the viewer, thereby indicating a significant silhouette.](image-url)
The silhouette can be rendered, for instance, by decreasing the brightness of the targeted region. In rendered images, this shows up as a more or less subtle shadowing effect, where pseudo-shadows highlight significant contours from the surface under analysis. A result can be seen in Fig. 4, which shows a comparison between a volume rendering without contour highlighting (Fig. 4 a.) and an image of the same data set, where the gradient-based method successfully identifies significant contours and outlines on the visible surfaces. Note that this approach is viewer-oriented, meaning that the evaluation of the contours is adaptable to the viewing direction. The thickness of the contours can be adjusted as shown in eq. (2) (Kindlmann et al., 2003).

\[
mDot = \sqrt{Tk(2 - Tk)}
\]

If the dot product between the viewing vector and the gradient is lower than \( mDot \), then the corresponding surface point is considered to be part of the contour. \( k \) is the local curvature, a measurement which expresses the level of deformation of the surface in a specific point. It can be computed using various approaches (Kindlmann et al., 2003; Bruckner & Größer, 2007) (it may, for instance, be specified in interface elements such as textboxes or sliders).

If the dot product between the viewing vector and the gradient is lower than \( mDot \), then the corresponding surface point is considered to be part of the contour. \( k \) is the local curvature, a measurement which expresses the level of deformation of the surface in a specific point. It can be computed using various approaches (Kindlmann et al., 2003; Bruckner & Größer, 2007) (it may, for instance, be specified in interface elements such as textboxes or sliders).

![Image](image.png)

Fig. 4 – (a) rendered image without contours. Various features such as narrow vasculature or finer details in bone structures do not stand out and may be difficult to perceive. (b) the same data set with gradient-based contours. The blackened regions correspond to areas where the surface faces away from the viewer, i.e. where the gradient and viewing direction form wider angles.

4. Representation of Features Using Image Post-Processing

Image post-processing comprises techniques which may be incorporated into the volume rendering pipeline after an initial 2D image has been obtained
as a result of 3D rendering. Volume rendering algorithms such as ray casting compute pixel values for an output image of a specified resolution. These values are typically RGBA quadruplets: RGB are the color components, while A is the alpha value and is used to compute opacity. The collection of RGBA values is implicitly sent to the video buffer for on-screen display. However, in some cases it may prove limiting to show the output image as it directly results from 3D rendering. Instead, other processing stages may be inserted in the rendering pipeline, before the final output image is produced. We refer to such stages as image post-processing. After an image of the dataset has been generated (such as in Fig. 1), various algorithms may be applied to the result in order to further enhance various details which may otherwise be difficult to see. While there exists a wide assortment of image processing algorithms (Gonzales & Woods, 2008), we mostly use various types of spatial filtering techniques to bring out further details in the generated images.

A spatial filter is a collection of scalar values arranged in a 2D matrix which is applied to an image via a convolution product. A pixel of the output image results from its own, initial value, as well as from contributions of neighboring pixels. Multiple variants of spatial filters may be applied, depending on the values of the filtering matrix. On a relatively high-performance GPU, the filtering computations can easily be carried out in real-time. An image processor component takes as input the pixel values obtained from ray casting (for example as a 2D texture) and subsequently applied a convolution operation across the entire resulting image based on a specified filtering kernel. Fig. 5 shows several images obtained using various types of filters. Even relatively simple filters may have a substantially-beneficial impact on the resulting image, thus allowing the user to better observe details which may be easily-overlooked in the original, unfiltered image.

Fig. 5 – A gallery of renderings with various filter types: a – the original image, as obtained from volume rendering; b – applying a low pass (blurring) filter smoothes the resulting image and reduces noise; c – brightness and contrast enhancement highlights various details from the original image; d – an edge detection filter based on first-order derivatives allows the representation of significant contours.
Based on the above, in order to get better and more accurate results, we often use spatial filters in combination with blending operators. This is in many cases a more flexible solution, allowing for further customization of the result. There are two elements which participate in such an operation: the filter either removes unwanted information or enhances the existing useful one, while the blending factor decides how the filtered result is added to the image. Blending involves a combination between the original image and its filtered version, thereby preserving information from the initial image while adding the enhancements from the modified one.

In one such example, the first step involves a high-pass filter, defined in eq. (3), where \( m \) is user-adjustable (Petrou & Petrou, 2010). This type of filter has the advantage that it brings out finer details between bordering regions in the image, but, as a downside, it also enhances noise.

\[
\mathbf{h} = \begin{pmatrix}
  -m/8 & -m/8 & -m/8 \\
  -m/8 & m & -m/8 \\
  -m/8 & -m/8 & -m/8 
\end{pmatrix}
\]

The second step uses a **screen blending** operator to combine the original color \( c_0 \) (i.e. the color in the unfiltered image) with the color \( c_1 \) obtained after filtering (eq. (4)).

\[
\text{finalColor} = 1 - (1 - c_0) (1 - c_1)
\]

A result can be seen in Fig. 6. The original image is first subjected to a high pass filter, revealing only the highest frequencies. By blending this with the original, the resulting image is substantial enhanced as far as the visibility of details is concerned, but context information from the initial image is still retained.

![Image](image_url)

**Fig. 6** – Examples of the use of filtering and blending: \( a \) – the original, unfiltered image; \( b \) – a high-pass filter reveals only the highest frequencies in the image (i.e. the areas where there is substantial change in color); \( c \) – the result obtained by screen blending \((a)\) and \((b)\). Many details from the original image are more easily noticed.
5. Conclusions

We presented several approaches for the visualization and display of contours and features from volume data. These methods were incorporated into the volume rendering pipeline, of which the major component was the rendering algorithm itself. To this end, we used a ray casting approach to render images from structured medical data sets, where various anatomical features could be concurrently displayed through a precise control of color and opacity. We then incorporated various contour and outline extraction algorithms. One such method was based on the orientation of gradients computed from the scalar values of the data set. Gradients are orthogonal to significant surfaces inside the volume and their orientation in relation to the viewing direction indicates which way the surface faces away from the viewer. Based on these considerations, silhouettes could be identified in the on-screen output. Another approach involved the use of image processing techniques, which were added to the visualization pipeline after the volume rendering stage. Through use of various filters and operators, we were able to further refine the images and show additional details and significant outlines. Future work in this direction involves mainly the development of other, more efficient filtering methods, other algorithms for the optimal extraction of contours as well as the better incorporation of these methods into the visualization process.

REFERENCES

Lucrarea prezintă tehnici de identificare și reprezentare a diverselor elemente vizuale care fac obiectul renderizării volumetrice, gen contururi, margini, muchii, precum și a diverselor trăsături ce permit o mai bună analiză vizuală a datelor volumetrice. Într-o primă parte introductivă, se prezintă contextul științific și practic al domeniului visualizării volumetrice, precum și al tehnicilor prezentate ulterior. A doua parte prezintă algoritmul de renderizare volumetrică utilizat pentru obținerea imaginilor din lucrare. Metoda utilizată este o variantă a algoritmului ray casting; aceasta presupune mai multe etape, care includ eșantionarea volumului, reconstrucția, clasificarea și reprezentarea sa pe ecran. Acestei metode de renderizare i se adaugă diverse alte etape,
care constituie implementarea unor procedee descrise în părŃile 3 şi 4 ale lucrării. O primă astfel de abordare presupune utilizarea vectorilor gradient, calculaŃi pe baza datelor din volum. GradienŃii sunt normali la suprafeŃele semnificative din interiorul volumului. Algoritmul prezentat funcŃionează în baza faptului că unghiul dintre direcŃia de privire şi gradient indică orientarea suprafeŃei faŃă de observator. Regiunile pentru care acest unghi are valori apropionate de 90º conŃin contururi semnificative de pe suprafeŃele analizate.

O a doua abordare implică încorporarea de tehnici de procesare de imagini în procesul de renderizare. Odată obŃinute imagini din seturi de date 3D, acestea sunt prelucrate folosind diverse tipuri de filtre spaŃiale, precum şi asocieri dintre astfel de filtre şi operatori de combinare. Se poate astfel obŃine o gamă foarte variată de imagini, în care se pot observa mai uşor diverse detalii: muchii, contururi, margini şi alte trăsături. Imaginile prezentate în figurile din cadrul lucrării ilustrează tehnicile prezentate şi realizează compararea imaginilor neprelucrate cu rezultatele obŃinute în urma procesărilor descrise în lucrare.