Illustrative volume visualization for unstructured meshes based on photic extremum lines

Allan Rocha Fábio Markus Miranda Waldemar Celes
Tecgraf/PUC-Rio – Computer Science Department
Pontifical Catholic University of Rio de Janeiro, Brazil
Email: {acr, fmiranda, celes}@tecgraf.puc-rio.br

***Abstract*—Scientific visualization techniques create images attempting to reveal complex structures and phenomena. Illustrative techniques have been incorporated to scientific visualization systems in order to improve the expressiveness of such images. The rendering of feature lines is an important technique for better conveying surface shapes. In this paper, we propose to combine volume visualization of unstructured meshes with direct rendering of illustrated isosurfaces. This is accomplished by extending a GPU-based ray-casting algorithm to incorporate illustration with photic extremum lines, a type of feature lines that captures sudden change of luminance, conveying shapes in a perceptually correct way.**

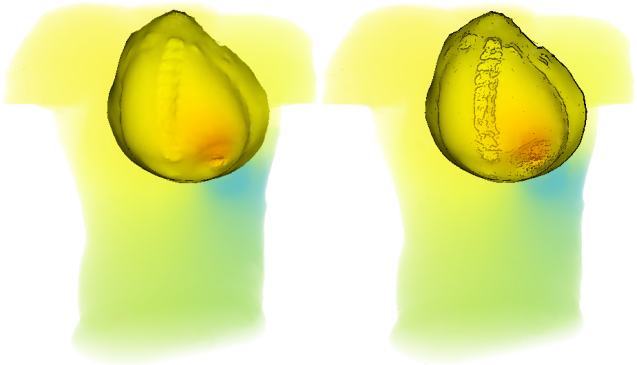
***Keywords*-Illustrative Volume Visualization ; Photic Extremum Lines**

I. Introduction

Throughout history, artists have used illustration to emphasize model features, improving image understanding with simplicity. In computer graphics, non-photorealistic rendering (NPR) [1] tries to mimic illustration techniques, producing images inspired by the way artists work on illustrations. Non-photorealistic rendering techniques do not faithfully represent the models as they really are, but enhance their features to ease shape understanding [2]. Illustration techniques go beyond NPR techniques and use *abstraction* as a key component to reproduce images. Illustrations are effective because deliver images that guide the observer to focus on the essential parts of the model, without losing the context in which the model is inserted [3].

Illustrative visualization then tries to combine the visual abstraction from illustrations with NPR techniques in order to achieve more expressive results. Illustrative visualization is concerned to "what" to render and "how" to render, and then uses both low-level and high-level abstraction techniques [4]. Drawing feature lines is one of the most effective low-level abstraction technique, and is explored by traditional illustrators [5]. Observing an illustration produced by an artist, one can note, besides stippling and hatching, the use of a combination of different line types. Based on this, several researches have been conducted on feature line extraction and rendering for three-dimensional computer generated models.

However, while designing their illustration, artists take into account their visual perception of the model to choose the set of lines to use, instead of relying on a pure geometry analysis.



(a) Conventional (b) Illustrated

Fig. 1. Image of an electrocardiography relating electric potential on the body surface with activities in the heart. The proposed illustration technique clearly depicts the interaction with the vertebral column.

One key ingredient for understanding the shape and features of a model is its illumination. Based on lighting variation (including shadows) artists draw lines to express the object. Based on this, Xie et al. [6] presented a technique to extract *photic extremum lines* (PELs), which emphasize significant variations of illumination over 3D triangle surfaces. Also trying to explore illumination variation on triangle surfaces, Zhang et al. [7] defined Laplacian lines as the zero-crossing points of the Laplacian of the surface illumination.

Ebert and Rheingans [8] introduced the concept of *volume illustration*, combining line extraction with NPR techniques for volume data. Volume illustration derives from the same concepts of visual abstraction, integrated with the flexibility to manipulate a transfer function, to achieve enhanced images. Volume illustration techniques are effective for conveying the structure of the data inside the volume and to reveal their main features [9], [10], [11], [12], [13], [6], [14]. In this context, researchers have extracted feature lines for emphasizing isosurface rendering. In general, the isosurface is previously extracted from the volume, and conventional techniques for triangle meshes are applied.

In this paper, we propose to combine volume visualization of unstructured tetrahedral meshes with illustrated isosurfaces. This is accomplished by extending a GPU-based ray-casting

algorithm to incorporate illustration with photic extremum lines (PELs) [6]. Instead of previously extracting the isosurface, we propose a direct rendering technique, enhancing the isosurface combined with the volume visualization. The PELs are extracted without drastically impacting the performance of the volume visualization algorithm. We show that the rendering of these lines in fact delivers better understanding of the data, revealing important structures and features otherwise hidden. Figure 1 illustrates an achieved result. It shows a torso electrocardiogram that relates electric potential on the body surface with activities in the heart. As can be noted, our illustration technique better reveals the interaction between the electric potential with the vertebral column.

II. Related work

In this section, we briefly review works related to our approach. We first review papers on line drawings and then on volume illustration.

A. Line Drawing

A set of computer-generated lines has been used for illustrative purpose, most inspired on the way artists enhance their drawings. These line-generator algorithms can be classified in two different groups: image space and object space. Object-space algorithms compute geometric surface features to judge the expressiveness of lines. As an example, we can mention the extraction of ridges and valleys, which represent regions of maximum and minimum curvatures [15]. Another example is demarcating curves, which represent the loci of the strongest inflections on the surface [16].

Other approaches consider geometric features with respect to the point of view. These are the view-dependent lines. The classical example is silhouette lines that define object contour [17], but fail to reveal internal structures. Suggestive contours try to overcome this limitation, drawing lines that reveal other regions where the surface bends sharply away from the viewer [18]. Highlight lines, as defined in [19], extend suggestive contours for convex surfaces. Another method, apparent ridges [20], generalizes ridges and valleys lines in a view-dependent fashion.

Noting that illumination variations also reveal information related to the geometry of objects, Xie et al. [6] proposed the photic extremum lines (PELs). PELs are not based on curvature variation, but on illumination variation. Different light sources can be used to better reveal local object structures. Zhang et al. [21] improved PEL extraction by incorporating an enhanced shading technique and eliminating the use of different light sources.

On the other hand, lines can be extracted in image space, as a post-processing phase [22]. These techniques face the challenges imposed by using a two-dimensional and discrete representation of the model, such as occlusion and low resolution. Lee et al. [23] proposed to process the image understanding lines as an abstraction of a shaded image. Jardim and de Figueiredo [24] presented a hybrid method for computing apparent ridges, combining object-space and image-space computations.

B. Volume illustration

Volume illustration represents a new paradigm for rendering volume data [8], [25]. The goal is to combine visual abstraction, NPR, and volume visualization to better reveal important features of the data inside the volume. Bruckner [4] explored volume illustration for medical data. Semantic layers, introduced in [26], allow the mapping of volumetric attributes to one visual style, and Xie et al. [6] proposed an effective illustrative visualization framework, combining PELs and shading on isosurfaces previously extracted from the volume.

As pointed out by Xie et al. [6], for methods that illustrate isosurfaces after extraction, it is difficult to achieve real-time interaction. One has to first extract the triangle mesh representing a given isosurface, perform differential computation on the mesh, and then compose the final image by multiple-rendering passes to combine isosurface and volume rendering. Kindlmann et al. [9] avoided all this process by using curvature-based transfer functions, enhancing the expressive and informative power of direct volume rendering, revealing, for instance, ridges and valleys in the volume. Interrante et al. [27] also proposed a technique for drawing ridges and valleys on a transparent or semi-transparent skin surface. The volumetric rendering system proposed in [13] directly extracts silhouettes and suggestive contours. Ma and Interrante [28] had already presented a discussion on the motivation for extracting and displaying perceptually relevant feature lines from unstructured grids.

In this paper, we propose to combine volume rendering of unstructured meshes with direct rendering of illustrated isosurfaces. We employ a GPU-based ray-casting algorithm that directly extracts the PELs. Our choice for PELs is based on their easiness of computation and their independence from explicit computation of surface curvature. As a result, we are able to illustrate isosurfaces with a small performance penalty.

III. Data representation

In this work, the volume data are represented by unstructured tetrahedral meshes. The scalar field is given at vertices of the tetrahedra. We then assume a piecewise linear variation of the scalar field inside the volume. The gradient of the scalar field represents its variation. On an isosurface, the gradient represents the surface normal. Therefore, to compute the diffuse light on the surface, we need to compute the scalar field gradient at arbitrary points in the volume. This is accomplished by using a linear gradient reconstruction method. Correa et al. [29] recently presented a comparison among linear gradient estimation methods, classifying them in two groups: averaging-based methods and regression-based methods. We have opted for using the Green-Gauss average-based method, which produces good results with a low computational cost [29].

In a pre-processing phase, from the scalar field values sampled at the vertices of the mesh, $f(\mathbf{x})$, we estimate and

store the gradients as part of the volume data. Considering a tetrahedral cell defined by its four vertices $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, we can compute the constant gradient (∇f) associated to the linear scalar field variation inside the cell, by solving the following linear system [29]:

$$\begin{bmatrix} (\mathbf{v}_1 - \mathbf{v}_0)^T \\ (\mathbf{v}_2 - \mathbf{v}_0)^T \\ (\mathbf{v}_3 - \mathbf{v}_0)^T \end{bmatrix} \nabla f = \begin{bmatrix} f(\mathbf{v}_1) - f(\mathbf{v}_0) \\ f(\mathbf{v}_2) - f(\mathbf{v}_0) \\ f(\mathbf{v}_3) - f(\mathbf{v}_0) \end{bmatrix}$$

We assume that all tetrahedra defining the volume are non-degenerated. From the constant gradients associated to the cells, we need to compute smoothed gradients at the vertices; otherwise, the isosurface illumination would have discontinuity along cell interfaces.

Averaging-based methods express the gradient at a given vertex $\mathbf{x}$ by averaging the gradients of neighboring tetrahedra:

$$\nabla f(\mathbf{x}) = \sum_i w_i \nabla f(i)$$

where w_i is a weighting factor, and $\nabla f(i)$ is the constant gradient at the adjacent tetrahedron i. The Green-Gauss approximation uses the volume of each tetrahedron (V_i) as the weighting factor [29]

$$\nabla f(\mathbf{x}) = \sum_i V_i \nabla f(i)$$

Once we have the gradient at each vertex, during ray traversal we can estimate the gradient at any point $\mathbf{x}$ inside a tetrahedron using barycentric interpolation. The barycentric coordinate associated to point $\mathbf{x}$ in a given tetrahedron is computed by:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \mathbf{B}\,(\mathbf{x} - \mathbf{v}_0)$$

where:

$$\mathbf{B} = \begin{bmatrix} \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 & \mathbf{v}_3 - \mathbf{v}_0 \end{bmatrix}^{-1}$$

We compute the matrix $\mathbf{B}$ and associate it to the corresponding tetrahedron in a pre-processing phase. The fourth barycentric coordinate is simply given by: $\lambda_0 = 1 - (\lambda_1 + \lambda_2 + \lambda_3)$.

During ray traversal, should the ray intersect the isosurface of interest at point $\mathbf{x}$, we compute the barycentric coordinate and then compute the associated unit normal vector by:

$$\vec{n} = \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$$

with $\nabla f(\mathbf{x})$ given by:

$$\nabla f(\mathbf{x}) = \lambda_0 \nabla f(\mathbf{v}_0) + \lambda_1 \nabla f(\mathbf{v}_1) + \lambda_2 \nabla f(\mathbf{v}_2) + \lambda_3 \nabla f(\mathbf{v}_3)$$

IV. Ray Casting for Tetrahedral Meshes

The light transportation through a volumetric cloud of scalar values is given by [30]:

$$I(D) = I(0) e^{-(\int_0^D \rho(t)dt)} + \int_0^D e^{-(\int_t^0 \rho(u)du)} \kappa(t)\rho(t)dt$$

where D is the ray length inside the volume with ρ and κ representing the attenuation coefficient and luminance, usually mapped to RGBA values through a transfer function.

A major challenge of volume rendering is to evaluate such integral in a fast and accurate way. One of the approaches is based on ray casting. Given an unstructured mesh composed of tetrahedral cells, a set of rays are casted from the eye position to the model mesh boundary. Each ray is then traversed through the cells using an adjacency data structure until the ray leaves the model mesh.

In this work, we use a ray-casting algorithm implemented in CUDA. At each cell traversal, the ray integral value is calculated using an approach proposed by Moreland and Angel [31] and Espinha and Celes [32]. Considering a piecewise linear transfer function, it is possible to pre-calculate the integral and store it on a 2D texture, and use the scalar values from the front and the back of the tetrahedron as texture coordinates to access the pre-computed values.

V. Direct PEL extraction

Photic extremum lines (PELs) capture sudden change of luminance. As proposed by Xie et al. [6], PELs can be extracted considering one or more light sources. For each light source, they proposed to extract PELs considering only diffuse illumination. The diffuse intensity at a given point on a surface can be simply expressed by:

$$I = \max(\vec{n} \cdot \vec{l},\, 0)$$

where $\vec{n}$ is the unit normal at this point, and $\vec{l}$ is the unit vector from the point on the surface toward the light source. The extracted feature lines are directly related to the surface shape, since variation of diffuse light is highly affected by the variation of the normal [6].

Because we are rendering volume isosurfaces, both "back" and "front" faces are relevant. We then compute the diffuse contribution by:

$$I = |\vec{n} \cdot \vec{l}|$$

In their experiments, Xie et al. [6] used a main directional light parallel to the view vector and optional auxiliary local lights. In order to minimize performance penalty, we have opted to use just one light for PEL extraction. We propose to use a local light at the viewer position. We have concluded that a positional light works better in our case, because it better captures small normal variations. As we discretize the volume by tetrahedra, and assume that the gradient varies linearly inside each cell, a directional light would be less sensitive to the diffuse light variation.

A. PEL definition

Photic extremum lines (PELs) are defined as the loci of points on the surface where the variation of the absolute value of the illumination, in the gradient direction, reaches the local maximum [6]. The unit gradient direction of the diffuse contribution is given by:

$$\hat{w} = \frac{\nabla I}{\|\nabla I\|}$$

The points representing local maxima must satisfy:

$$D_{\hat{w}} \|\nabla I\| = 0 \quad \text{and} \quad D_{\hat{w}} D_{\hat{w}} \|\nabla I\| < 0$$

B. Computing PELs

In our algorithm, whenever a ray reaches an isosurface of interest, we check if the intersection point belongs to a PEL on the surface. We do that numerically, using finite difference.

Expressing the point on the surface by $\mathbf{x}$, we first estimate the gradient of the scalar field at this point by barycentric interpolation as described. This gradient represents the normal direction of the surface at this point. We then find a surface-aligned basis, computing the tangential direction $\hat{u}$ and $\hat{v}$ through cross products. The gradient of the illumination function ∇I, on the isosurface, is then computed using central difference:

$$\frac{\partial I}{\partial u}(\mathbf{x}) = \frac{I(\mathbf{x} + \delta\hat{u}) - I(\mathbf{x} - \delta\hat{u})}{2\delta}$$
$$\frac{\partial I}{\partial v}(\mathbf{x}) = \frac{I(\mathbf{x} + \delta\hat{v}) - I(\mathbf{x} - \delta\hat{v})}{2\delta}$$

where δ isto the spacing parameter to evaluate the derivative. Note that the illumination gradient is expressed in tangential coordinates. Points $\mathbf{x} \pm \delta\hat{u}$ and $\mathbf{x} \pm \delta\hat{v}$ may fall outside the current tetrahedron. In such a case, we employ a recursive algorithm that uses the tetrahedron adjacency information to traverse the mesh, until the containing tetrahedron is found, or the external boundary of the model is reached (in which case, we replace the central difference by forward or backward difference). Once the containing tetrahedron is found, we compute the corresponding illumination function.

This procedure allows us to compute the variation of the illumination function at arbitrary points inside the volume. We need to check if, at point $\mathbf{x}$, the function $\|\nabla I\|$ reaches its local maximum, along the direction of $\hat{w}$ ($= \nabla I / \|\nabla I\|$). This is accomplished by computing the absolute value of the illumination gradient at points $\mathbf{x} + \gamma\hat{w}$ and $\mathbf{x} - \gamma\hat{w}$, where γ is another spacing parameter. We then compute three absolute values of the illumination variation along $\hat{w}$:

$$\begin{aligned} v_0 &= \|\nabla I(\mathbf{x} - \gamma\hat{w})\| \\ v_1 &= \|\nabla I(\mathbf{x})\| \\ v_2 &= \|\nabla I(\mathbf{x} + \gamma\hat{w})\| \end{aligned}$$

Our goal is to evaluate if v_1 is the maximum value of the three. We do that just evaluating:

$$v_1 - \max(v_0, v_2) > \epsilon$$

where ϵ is a tolerance needed due to noise and numerical approximations.

VI. Parameter settings

In general, the effectiveness of algorithms to extract feature lines relies on appropriate settings of parameter values. Our algorithm is also subject to adequate parameter settings. As described, we work with three parameters that must be adjusted to better illustrate the isosurfaces in the volume. We use the spacing parameter δ to evaluate the illumination function gradient on tangential space; we use the spacing parameter γ to sample the function that represents illumination variation along the gradient direction; and we use the numerical tolerance ϵ to compare maximum values. In this section, we discuss how these parameters affect line extraction and propose re-parameterization to ease the choice of appropriate settings.

A. Spacing parameter for gradient evaluation

The spacing parameter δ is used to evaluate the gradient of the illumination function, which is directly related to the surface normal, i.e., to the gradient of the scalar field. As mentioned, the gradient of the scalar field varies linearly inside a tetrahedral cell. As a consequence, if we use a small value for δ, we will probably reach points inside the current tetrahedron, ending up evaluating the illumination variation with a linear variation of normals. This would reveals discontinuities along cell interfaces. On the other hand, if we use a larger value for δ, we smooth the gradient of the scalar field, and the use of too large values would not capture important features.

It is clear that this parameter has to vary in accordance with the size of the tetrahedron containing the point at which the gradient is being evaluated. We have observed that we should use a parameter value that forces the central difference to use points in distinguish tetrahedra, but it should be as small as possible to avoid losing features of the data. We then propose to re-parameterize this spacing distance as follows:

$$\delta = \alpha\, r$$

where α represents the new parameter that replaces δ, and r is the radius of the insphere of the tetrahedron that contains the point, given by:

$$r = \frac{6V}{\|\vec{a}\| + \|\vec{b}\| + \|\vec{c}\| + \|\vec{a} + \vec{b} + \vec{c}\|}$$

with:

$$\vec{a} = (\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)$$
$$\vec{b} = (\mathbf{v}_2 - \mathbf{v}_0) \times (\mathbf{v}_3 - \mathbf{v}_0)$$
$$\vec{c} = (\mathbf{v}_3 - \mathbf{v}_0) \times (\mathbf{v}_1 - \mathbf{v}_0)$$

The value of α is not hard to set. In our experiments, a value from 1 to 5 has worked. It depends on how smooth is the scalar field.

B. Spacing parameter for maximum check

The spacing parameter γ is used to check if the current point is a maximum of the function that represents illumination variation along the illumination gradient direction. This parameter is related to the thickness of the extracted lines. If we set a large value for γ, several pixels, across the line thickness direction, will likely be associated to points of maximum values. This happen because each point will be compared against distant points; in a region of maximum values, all points will present greater value than points outside the region.

We then propose the following re-parameterization:

$$\gamma = \beta \, d_p$$

where β is the new parameter that replaces γ, and d_p represents, in object space, the distance proportional to one pixel on the screen. This distance is approximated by:

$$d_p = \tan\frac{\theta}{h} \, \|\mathbf{x} - \mathbf{v}\|$$

where θ corresponds to the vertical field of view of the camera, h is the height of the viewing surface expressed in pixels, and $\mathbf{v}$ is the position of the viewer. As a consequence, a β value equal to 1 tends to reproduce lines with thickness value also equal to 1. In our experiments, we have opted for using values from 2 to 3. Too thin lines tend to present discontinuities due to noise and numerical precision.

C. Numerical tolerance for filtering lines

The last parameter of our algorithm, ϵ, is used to filter only "strong" maximum values. This is important to avoid capturing all small variation of the illumination gradient, what would result in polluted images. When illustrating surfaces, the goal is to emphasize only the main feature lines. This parameter plays this role. A value equal to zero would capture all the small variations.

This parameter is hard to be re-parameterized. It depends on the variation of the illumination gradient, which cannot be pre-evaluated. We have observed though that this parameter is not hard to set if we work with normalized coordinates. Prior to rendering, we apply a scale to fit the geometry of the model in the unit cube. With this normalization, we have found that values in the interval from 0 to 1 produce good results.

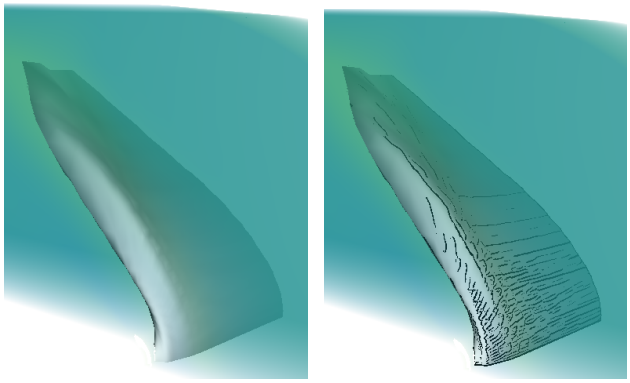
VII. Results

To demonstrate the effectiveness of our proposal, we have tested our algorithm for direct rendering volume data with illustrated isosurface for different models, including medical and engineering data. In this section, we discuss the achieved results.

A. *Conventional* vs. *illustrated volume rendering*

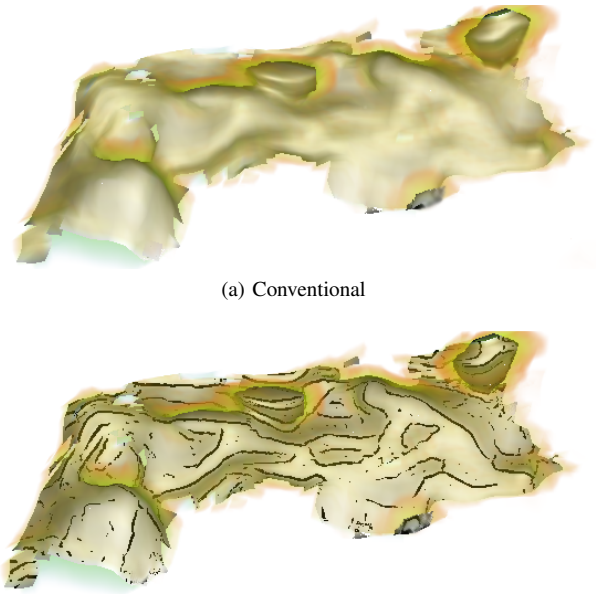
First, we show how the proposed technique does enhance volume visualization by better revealing isosurface shapes. We have already mentioned that our technique was capable of better depicting the interaction between the electric potential with the vertebral column in Figure 1.

We also tested our proposal for other models. Figure 2 shows the achieved result for illustrating an isosurface of the well known *bluntfin* model. The illustrated isosurface reveals small scalar field oscillation along the surface that would be otherwise unnoticed. Figure 3 compares the results of our algorithm with conventional volume rendering for a black-oil reservoir model. Again, the proposed algorithm better conveys isosurface shape.



(a) Conventional (b) Illustrated

Fig. 2. Volume rendering of the *bluntfin* model.



(a) Conventional

(b) Illustrated

Fig. 3. Volume rendering of a black oil reservoir model.

B. Correctness

In order to check the correctness of our proposal to direct extract feature lines, we first apply our algorithm for rendering the surface of a torus. The torus is represented by a synthetic volume data, represented by a regular grid of points whose scalar values were generated by evaluating the torus implicit equation. Each regular grid cell was then subdivided into six tetrahedra. Figure 4 shows the achieved shaded surface (on the left) and the extracted lines rendered in isolation (on the right). A similar experiment was also run by Xie et al. [6], extracting the same PELs. Although PELs is not helpful to convey the shape of a torus, this experiment shows that our proposal correctly computes the PELs for this synthetic surface.

We then compared the lines extracted by our approach on actual data with other different approaches: ridges, apparent ridges, and suggestive contours. These other lines were rendered by first extracting the corresponding isosurface using a

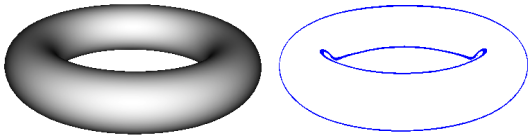


Fig. 4. Rendering of a synthetic volume data representing a torus surface.

marching-tetrahedron algorithm. We then used the software RTSC (the Real-Time Suggestive Countor) from Princeton University [33]. Because the extracted isosurfaces are too coarse, we had to use the functionalities provided by the software to smooth the surfaces. We first call the normal smooth and then the mesh refinement functions; otherwise, the lines were not adequately extracted. This, in fact, illustrates an advantage of our method: smoothness can be achieved by adjusting a single parameter; there is no need to perform geometry computations. Figure 5 and Figure 6 illustrate the results for the torso and the bluntfin models, respectively. As can be noted, the PELs directly extracted by our proposal are equivalent to the feature lines extracted from the previously generated isosurface meshes.

C. Parameter settings

For testing parameter settings, we have run the following experiment. We first chose appropriate parameters for visualizing the data from a torso electrocardiogram that relates electric potential on the body surface with activities in the heart. The lines drawn with our technique better reveal the interaction between the electric potential with the vertebral column, as illustrated in Figure 7b.

For this model, we have judged that the following parameters produce good images: $\alpha = 4.0$, $\beta = 3.0$, and $\epsilon = 0.65$. We then vary each one of these parameters to analyze its effect. Figure 7 illustrates the effect of varying the value of α. As stated, large values smooth the scalar field and important features may be lost. Small values capture discontinuities in the illumination function and result in noisy images. Figure 8 illustrates the effect of varying the value of β. As expected, larger values produce thicker lines. Finally, Figure 9 illustrates the effect of varying the value of ϵ. Again, as discussed, low values capture small variation, resulting in polluted images. Too large value may fail to capture important features.

VIII. Conclusion

In this paper, we have presented a new approach for direct rendering of unstructured volume data combined with illustrated isosurfaces. The illustration is done by extracting photic extremum lines (PELs) during ray traversal. Our algorithm is simple and relatively fast. With the current implementation, we observed that the performance of our conventional volume rendering algorithm was impacted in about 20 to 30%.

We showed that our proposal is helpful to convey isosurface shapes for different models. We demonstrated the correctness of our approach by comparing the achieved results with other proposals for feature line rendering on triangle meshes, which were previously extracted from the volume data. We have also discussed the algorithm parametrization, and have presented the influence of each parameter on the extracted lines.

Acknowledgment

We thank CNPq (Brazilian National Research and Development Council) for the financial support to conduct this research. This work was done in the Tecgraf laboratory at PUC-Rio, which is mainly funded by the Brazilian oil company, Petrobras.

References

[1] G. Winkenbach and D. H. Salesin, "Computer-generated pen-and-ink illustration," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 91–100.

[2] Gooch, Bruce, Gooch, and Amy, *Non-Photorealistic Rendering*. Natick, MA, USA: A. K. Peters, Ltd., 2001.

[3] P. Rautek, S. Bruckner, E. Gröller, and I. Viola, "Illustrative visualization: new technology or useless tautology?" *SIGGRAPH Comput. Graph.*, vol. 42, pp. 4:1–4:8, August 2008.

[4] S. Bruckner, "Interactive illustrative volume visualization," Ph.D. dissertation, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 3 2008.

[5] F. Cole, "Line drawings of 3D models," Ph.D. dissertation, Princeton University, Jun. 2009.

[6] X. Xie, Y. He, F. Tian, H.-S. Seah, X. Gu, and H. Qin, "An effective illustrative visualization framework based on photic extremum lines (pels)," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1328–1335, November 2007.

[7] L. Zhang, Y. He, X. Xie, and W. Chen, "Laplacian lines for real-time shape illustration," in *Proceedings of the 2009 Symposium on Interactive 3D graphics and games*, ser. I3D '09. New York, NY, USA: ACM, 2009, pp. 129–136.

[8] D. Ebert and P. Rheingans, "Volume illustration: non-photorealistic rendering of volume models," in *Proceedings of the conference on Visualization '00*, ser. VIS '00. Los Alamitos, CA, USA: IEEE Computer Society Press, 2000, pp. 195–202.

[9] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-based transfer functions for direct volume rendering: Methods and applications," in *Proceedings of the 14th IEEE Visualization Conference 2003 (VIS'03)*, ser. VIS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 513–520.

[10] S. Bruckner and M. E. Gröller, "Volumeshop: An interactive system for direct volume illustration," in *Proceedings of IEEE Visualization 2005*, H. R. C. T. Silva, E. Gröller, Ed., Oct. 2005, pp. 671–678.

[11] S. Bruckner and M. E. Groller, "Enhancing depth-perception with flexible volumetric halos," Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Tech. Rep. TR-186-2-07-04, Apr. 2007.

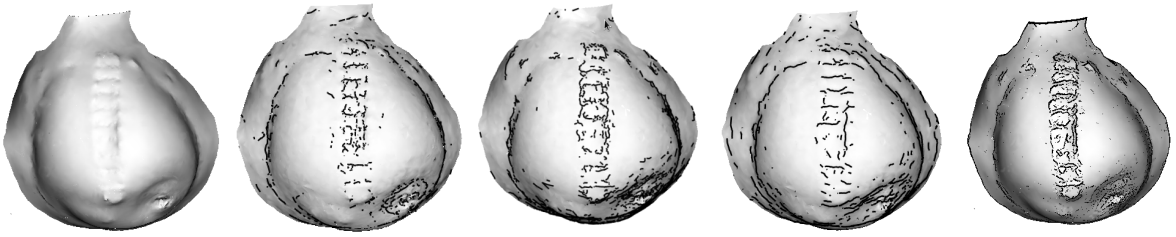
[12] ——, "Style transfer functions for illustrative volume rendering," *Computer Graphics Forum*, vol. 26, no. 3, pp. 715–724, Sep. 2007.

[13] M. Burns, J. Klawe, S. Rusinkiewicz, A. Finkelstein, and D. DeCarlo, "Line drawings from volume data," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 24, no. 3, pp. 512–518, Aug. 2005.

[14] F. d. M. Pinto and C. M. D. S. Freitas, "Volume visualization and exploration through flexible transfer function design," *Computers and Graphics*, vol. 32, no. 5, pp. 420–429, August 2008.

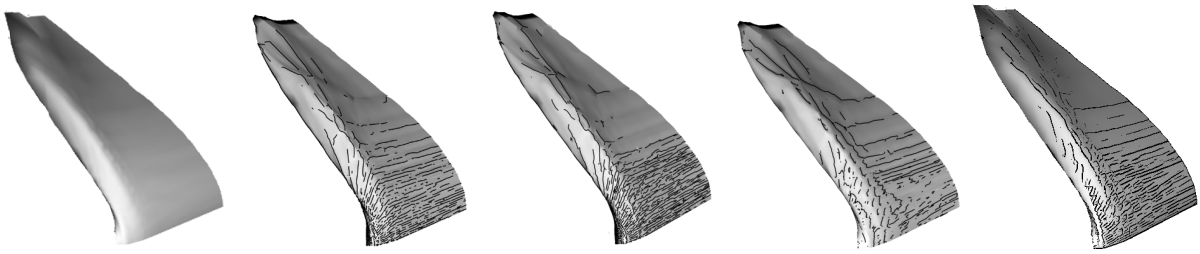
[15] Y. Ohtake, A. Belyaev, and H.-P. Seidel, "Ridge-valley lines on meshes via implicit surface fitting," *ACM Trans. Graph.*, vol. 23, pp. 609–612, August 2004.

[16] M. Kolomenkin, I. Shimshoni, and A. Tal, "Demarcating curves for shape illustration," *ACM Trans. Graph.*, vol. 27, pp. 157:1–157:9, December 2008.



(a) Diffuse illumination (b) Rigdes (c) Apparent ridges (d) Suggestive contours (e) Our technique

Fig. 5. Comparison among different techniques to extract feature lines for the torso model.



(a) Diffuse illumination (b) Rigdes (c) Apparent ridges (d) Suggestive contours (e) Our technique

Fig. 6. Comparison among different techniques to extract feature lines for the bluntfin model.

[17] A. Hertzmann and D. Zorin, "Illustrating smooth surfaces," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 517–526.

[18] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Trans. Graph.*, vol. 22, pp. 848–855, July 2003.

[19] D. DeCarlo and S. Rusinkiewicz, "Highlight lines for conveying shape," in *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, Aug. 2007.

[20] T. Judd, F. Durand, and E. Adelson, "Apparent ridges for line drawing," *ACM Trans. Graph.*, vol. 26, July 2007.

[21] L. Zhang, Y. He, and H.-S. Seah, "Real-time computation of photic extremum lines (pels)," *Vis. Comput.*, vol. 26, pp. 399–407, June 2010.

[22] T. Saito and T. Takahashi, "Comprehensible rendering of 3-d shapes," *SIGGRAPH Comput. Graph.*, vol. 24, pp. 197–206, September 1990.

[23] Y. Lee, L. Markosian, S. Lee, and J. F. Hughes, "Line drawings via abstracted shading," *ACM Trans. Graph.*, vol. 26, July 2007.

[24] E. Jardim and L. de Figueiredo, "A hybrid method for computing apparent ridges," in *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*, 30 2010-sept. 3 2010, pp. 118 –125.

[25] F. de Moura Pinto and C. Freitas, "Importance-aware composition for illustrative volume rendering," in *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*, September 2010, pp. 134 –141.

[26] P. Rautek, S. Bruckner, and M. E. Gröller, "Semantic layers for illustrative volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1336–1343, Oct. 2007, to be presented at IEEE Visualization 2007.

[27] V. Interrante, H. Fuchs, and S. Pizer, "Enhancing transparent skin surfaces with ridge and valley lines," in *Proceedings of the 6th conference on Visualization '95*, ser. VIS '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 52–. [Online]. Available: http://portal.acm.org/citation.cfm?id=832271.833836

[28] K.-L. Ma and V. Interrante, "Extracting feature lines from 3d unstructured grids," in *Proceedings of the 8th conference on Visualization '97*, ser. VIS '97. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997, pp. 285–ff. [Online]. Available: http://portal.acm.org/citation.cfm?id=266989.267085

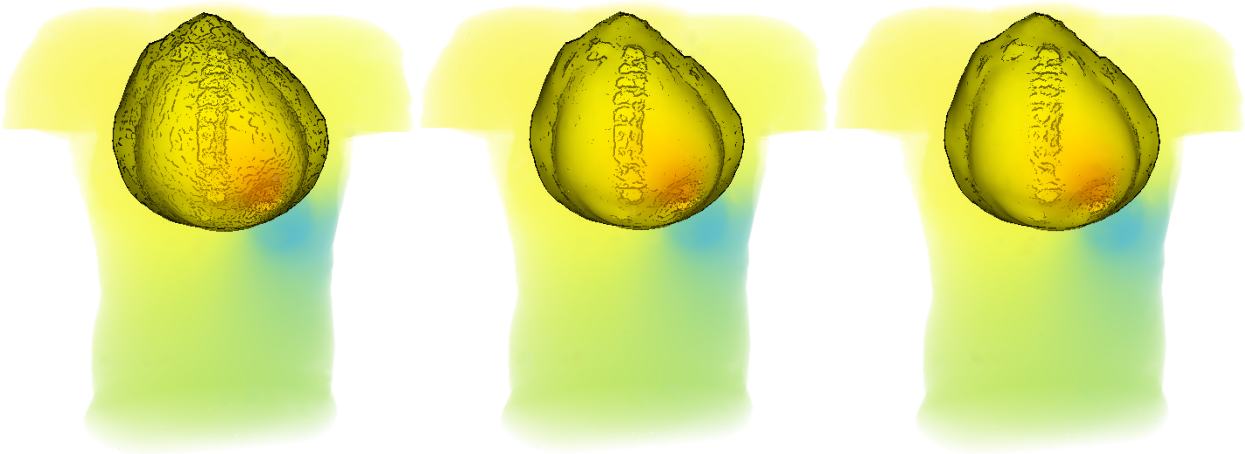
[29] C. D. Correa, R. Hero, and K.-L. Ma, "A comparison of gradient estimation methods for volume rendering on unstructured meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 305–319, May 2011.

[30] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 99–108, June 1995.

[31] K. Moreland and E. Angel, "A fast high accuracy volume renderer for unstructured data," in *Proceedings of the 2004 IEEE Symposium on Volume Visualization and Graphics*, ser. VV '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 9–16.

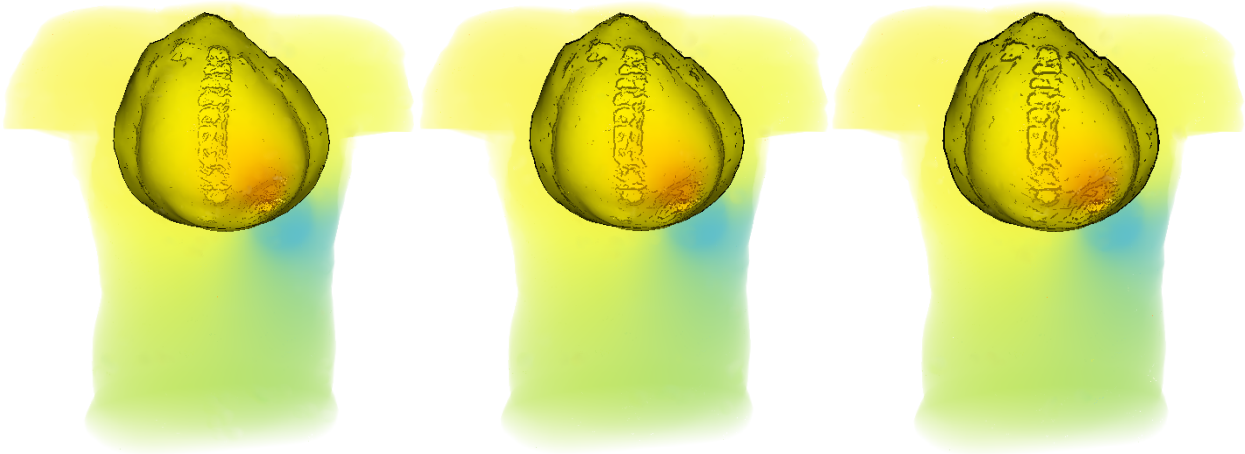
[32] R. Espinha and W. Celes, "High-quality hardware-based ray-casting volume rendering using partial pre-integration," in *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 273–280.

[33] S. Rusinkiewicz. Rtsc, the real-time suggestive contour. [Online]. Available: http://www.cs.princeton.edu/gfx/proj/sugcon/



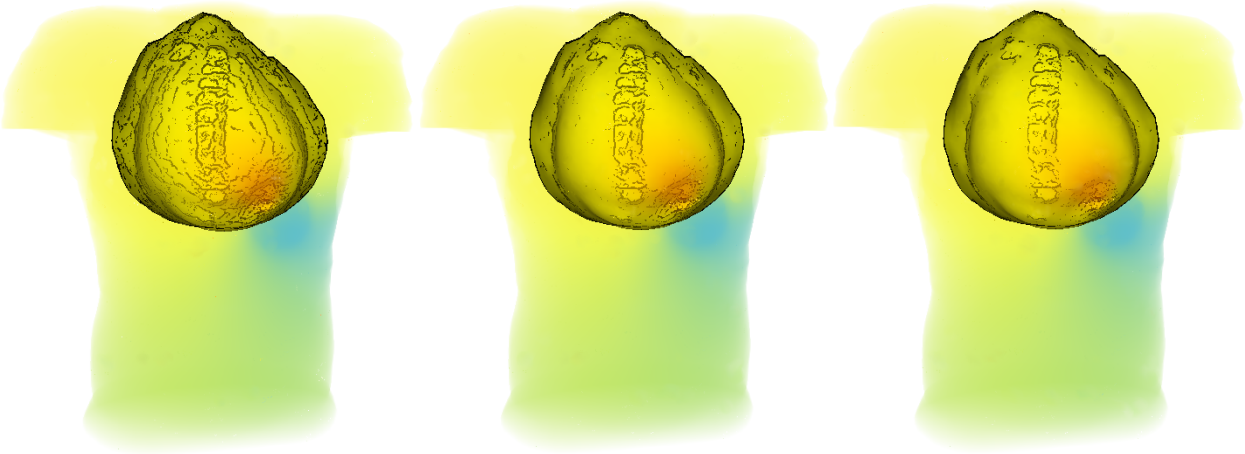
(a) $\alpha = 1.0$ (b) $\alpha = 4.0$ (c) $\alpha = 8.0$

Fig. 7. Effects of varying the parameter α; other parameters are fixed: $\beta = 3.0$ and $\epsilon = 0.65$



(a) $\beta = 2.0$ (b) $\beta = 3.0$ (c) $\beta = 4.0$

Fig. 8. Effects of varying the parameter β; other parameters are fixed: $\alpha = 4.0$ and $\epsilon = 0.65$



(a) $\epsilon = 0.05$ (b) $\epsilon = 0.65$ (c) $\epsilon = 1.30$

Fig. 9. Effects of varying the parameter ϵ; other parameters are fixed: $\alpha = 4.0$ and $\beta = 3.0$