Perspective Contouring in Illustrative Visualization

Sibgrapi paper ID: 57889

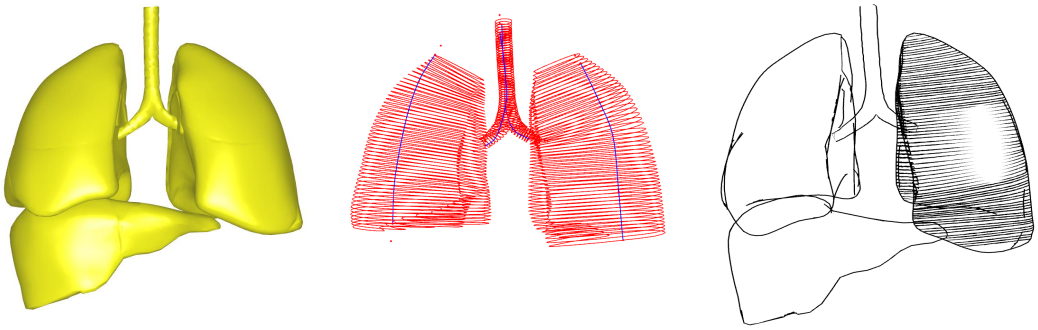

Figure 1. On a 3D model (left), the user defines axes that are used to extract cross sections from the geometry (center). After, the sections are used to render contour lines that create emphasis in a user-defined portion of the model (right). Silhoutte edges improve the shape perception.

***Abstract*—Although traditional line illustrations are very abstract representations, they can communicate the form of an object effectively. This is only possible by the use of perceptual cues, mastered by professional artists. To create effective line illustrations from 3D models, it is important to use these perceptual cues to avoid visual dissonance and ambiguity, and also to create visual effects like shading, emphasis and depth perception. In this paper, we propose the adaptation of a technique called Perspective Contouring, used by illustrators to create emphasis in contour line drawings. The technique uses perceptual principles to manipulate line attributes, like width, spacing and length, around regions of interest determined by the user, creating the illusion that those regions advance towards the observer. The whole process is controlled through an intuitive sketch-based interface.**

***Keywords*-illustrative visualization; focus of attention; line illustration; hatching; non-photorealistic rendering**

I. Introduction

Illustrations are abstractions created by artists for representing the world in a simple and comprehensible manner. Illustration techniques are very useful to create focus of attention, to prevent visual overload, to provide context information, and even guarantee the visibility of important information. These factors are especially important in several areas like medical education for example, where realistic images like photographs can be overwhelming and difficult to understand.

To create these abstractions, artists often use very simple elements like dots (stippling) and lines (hatching). To use these elements in an effective manner, they need to rely on perceptual principles to compose them, bringing unity, balance and emphasis to the illustrations. These principles are what make us see a 3D form or a textured surface out of many, apparently unorganized, lines, and if not used properly can lead to visual artifacts and misinterpretation. These principles often involve: contrast, constancy, color, shapes, patterns and visual attention, among others [1]. When these elements are contradictory they generate visual dissonance that reduces the effectiveness of communication.

To create convincing images of the real world using only lines, the illustrators apply their knowledge about the human visual system. The artists take advantage of how we perceive shapes, tones, contrast, etc., to manipulate the lines and to create effects and illusions so we do not see the lines any more, but the whole that results from the their composition.

Since the illustrative techniques used by artists are so effective in communicating information about structures, they were adopted in computer graphics for the visualization of scientific data, bringing together new applications for this "Illustrative Visualization" [2]. Many works in this area try to simulate the rendering styles used by artists, like hatching, to visualize 3D objects. Line-based illustrations are a very good way to depict the shape of an object and are also very simple and clean but, due to the high level of abstraction, are very difficult to use effectively.

Although many techniques try to replicate the style of line illustrations (related work is presented in Section II), the issue of how to provide emphasis in the illustration has not been well explored. Emphasis, or focus of attention, is used to highlight the most important regions in a scene and catch the observers attention. Without this quality, the drawing looses expressiveness and become inefficient, even ambiguous.

Creating good illustrations from 3D data can greatly improve the work of professional illustrators by automating

the more tiring tasks. However, artists can not loose the freedom they are used to and such techniques should support the correct use of perceptual principles.

This work presents the core techniques of an environment that allows the creation of illustrations from 3D datasets through contour lines. As main characteristics we can cite:

- The use of perceptual principles to emphasize important regions;
- The use of conceptual marking, a concept familiar to the illustrator;
- The freedom given to the illustrator to easily control in the illustration construction.

To emphasize important regions we adapt a traditional illustration technique called Perspective Contouring [3], which creates focus of attention through the manipulation of the length, width and density of the lines (Section III presents a brief description of this technique). The entire process is controlled by the user through a sketch-based interface.

In section IV we present an overview of the algorithm, which is described in details in Sections V and VI. The results are shown in Section refsec:results, and Section VIII brings the conclusions and future work.

II. Related Work

Early works like the one from Appel *et al.* [4] already used lines with halos to create depth effects, using the perceptual principle of overlapping [3] to improve the visualization of mathematical structures. Kamada and Kawai [5] applies stylization in hidden lines, allowing a better visualization of the structure of geometric objects from a single viewpoint. This subject was explored again by Dooley and Cohen [6], which used illustration rules to render silhouette and hidden lines.

These works, though use line illustration techniques, do not explore the power of combining multiple lines to achieve a hatching effect. Hatching can be used to shade the surface of an object, giving texture and lighting cues, and enhance the visualization of an object.

To create hatching, many works in Non-photorealistic Rendering (NPR) use some measurement on the surface of the model to define the lines. Girshick *et al.* [7] defines a potential field on an image using the principal curvature of the model. The lines are created distributing points through the surface of the models that are integrated in the direction defined by the potential field. Similar approaches are used in [8], [9], [10] and [11].

Some other works, like [12] and [13], use pre-defined textures with different tones and line styles that are mapped to the model's surface. Winkenbach and Salesin [14] describe a method to create line illustrations from smooth surfaces using NURBS patches on the surface to guide the hatching.

Rossl and Kobbelt [15] use the principal curvature fields to create parallel lines in image space. The user must perform a segmentation of the model in image space and, for each segment, parallel lines are parameterized following the maximum curvature direction. The line width is controlled by a tone mapping using a pre-rendered shaded image of the model.

Deussen *et al.* [16] use a hybrid approach to model the lines. They are defined by the intersection of a set of parallel planes with the model geometry. The orientation of each plane is determined by the user through a spline, and the intersection calculations are performed by the graphics pipeline through clipping planes. Although the calculation is performed in object space, the lines are generated in image space.

A recent work by Ritter *et al.* [17] uses contour lines in the visualization of vascular structures. The lines' width is used to indicate the depth of different parts of the structure. Regions that are closer to the viewer have thicker lines while thinner lines indicate a more distant region. Lines are also used to model shadows when two segments overlap, the size of the shadows indicating the distance between the segments.

As pointed out earlier, although the previous mentioned techniques are devoted to build line illustrations, none of them are explicitly based on perceptual principles to provide focus of attention.

III. Perspective Contouring

Being one of the most difficult techniques to master, line illustration demands knowledge about how the human visual perception works to create effects and illusions that allow us to understand abstract drawings.

Line illustrations are very abstract but artists can still manage to create drawings with very few lines and that are very comprehensible. This is possible because of perceptual cues used in such illustrations. These cues often involve contrast, line direction, focus and recognition of shapes, patterns and edges. When these perceptual cues are contradictory, they generate visual dissonance that may cause confusion and reduce the effectiveness of communication. Some examples of dissonance are zebra stripes, when line width and spacing are out of balance, and Moire patterns.

One important perceptual cue is emphasis or focus of attention. To create focus of attention in contour line illustrations, artists often use a technique called Perspective Contouring. This technique is a particular case of perspective that is very used when representing tubular structures in medical illustrations. The emphasis is achieved by creating regions that seems to advance towards the observer (see Figure 2).

When using this method, the illustrator, instead of using a universal horizon line for the scene, uses an internal horizon line usually perpendicular to the long axis of the object being depicted. The internal horizon line is defined across the region to be emphasized and construction rules are applied

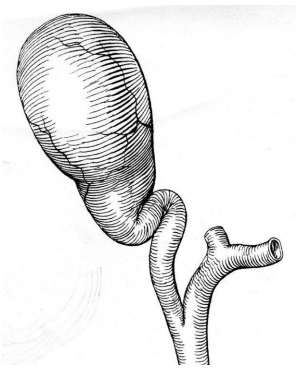

Figure 2. Use of Perspective Contouring to enhance perception of shape (illustration by Bill Andrews).

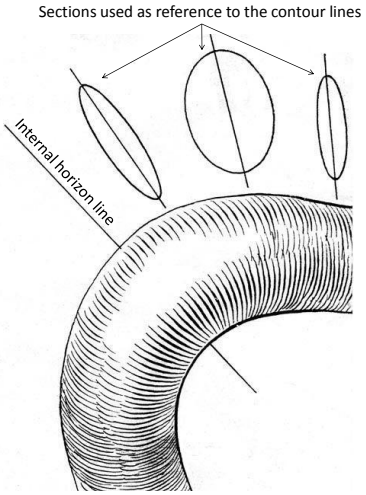

Figure 3. Perspective Contouring. The internal horizon line and some sections used to compose the illustration (adapted from original authored by Gerald Hodge).

to the line drawings based on the distance to this line (see Figure 3). These rules are [3]:

- Lines near the internal horizon line are thicker, shorter and more distant from each other;
- Lines far from the internal horizon line are thinner, longer and closer to each other;
- The silhouette near the internal horizon line is thinner;
- The silhouette far from the internal horizon line is thicker;

This technique facilitates the understanding of the shape of the object as well as directs the attention of the viewer to important regions of the scene. It is clear that the adaptation of this method to Illustrative Visualization techniques can improve the results obtained in several applications.

IV. Overview

Our method allows the user to construct line illustrations from 3D models using Perspective Contouring to create emphasis. The illustrations can be created using two kinds of lines: contour lines, which depict the surface of an object through hatching, and silhouette lines. Both groups of lines are manipulated according to the Perspective Contouring rules to create the illusion of emphasis.

Table I
Illustration process overview

Algorithm steps
1. Silhouette mesh creation (preprocessing step)
2. Definition of axes for cross section extraction (by the user through sketching)
3. Cross section extraction
4. Definition of the ROIs (by the user through sketching)
5. Lines' attributes calculation (Perspective Contouring rules)
6. Definition of lines' attributes outside of the ROIs (by the user through sketching)

The contour lines are modeled by sections extracted from the geometry of the object. This alternative is more expensive than image-space based approaches, but it generates a more exact representation of the contour of the object, allows more control over the lines attributes, and does not create consistency problems in animations. Since the cross sections only need to be extracted once, the object-space modeling for the lines is preferable.

The cross sections are obtained by the intersection of a set of planes with the geometry of the object. To orient the planes, a parametric curve, sketched by the user on the model is used as axis (a pre-calculated model skeleton can also be used, see the work of Cornea et al. [18] for some approaches for extracting skeletons). The planes are perpendicularly and uniformly distributed along this axis and the user can define one or more axis for the same object, each one having its own set of planes.

To model the silhouette lines, we use a special structure called silhouette mesh that keeps all the possible silhouette edges. In each edge of this mesh there is a patch of two triangles that can be rendered to form a silhouette line. The silhouette mesh is created as a preprocessing step based on the object mesh.

With the silhouette mesh created and the cross sections extracted, the user can define the regions of interest (ROIs) where emphasis will be applied. This selection is made through a sketch-based interface, which is also used to determine the form of the highlight used in the ROI. With the ROIs defined, the lines' attributes (spacing and width of the contour lines and width of the silhouette lines) are calculated using the construction rules described in Section III.

Outside the ROIs, the lines' attributes can be controlled by the user through a special sketch-based interface, where the curves sketched determine the behavior of the variables.

TableI resumes the whole process of creating an illustration.

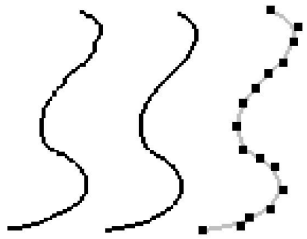

Figure 4. Sketching capture process. Original stroke (left), stroke after Chaikin subdivision (center) and final curve with control points (right) [20]

V. Geometric Structures

In this section, we detail the structures used to represent the geometric elements of the illustration creation process.

A. Sketched Curves

When creating an illustration the user draws curves through a sketch-based interface to define axes for planes used in the extraction of cross sections from the model, the ROIs and the highlights' shape, and also the behavior of the lines' attributes outside the ROIs.

To handle these curves, we first capture all the points from the input device, a mouse for example, in the image plane. These points cannot be used directly to define the parametric curve that we need because: the points are very noisy due to natural jittering in the handling of the input device; the points are irregularly distributed along the curve due to variations in speed of the sketch; there can be a very large amount of points because the input device sends information many times per second. So, to avoid these problems we find a b-spline that can fit in this set of points.

We use Chaikin reverse subdivision [19] to find a noiseless b-spline with a small number of regularly distributed points. Using a reverse subdivision scheme, we can decompose a fine set of points in a sparser approximation. As the Chaikin subdivision is based on a quadratic b-spline, we can assume the sparser information as control points of a quadratic b-spline curve (Figure 4).

If we define the fine points as p_0, p_1, ..., p_n and the sparse points as q_0, q_1, ..., q_m, then, the general case of the Chaikin reverse subdivision has the form:

$$q_j = -\frac{1}{4}p_{i-1} + \frac{3}{4}p_i + \frac{3}{4}p_{i+1} - \frac{1}{4}p_{i+2}, \qquad (1)$$

where the step size of i is two. The cardinality of the sparse points is almost half of the fine points. Each time the subdivision is applied the curve becomes more smooth, but it deviates more from the original curve. In our experiments, we concluded that performing the operation three times is enough to eliminate the noise without deviating too much from the original curve.

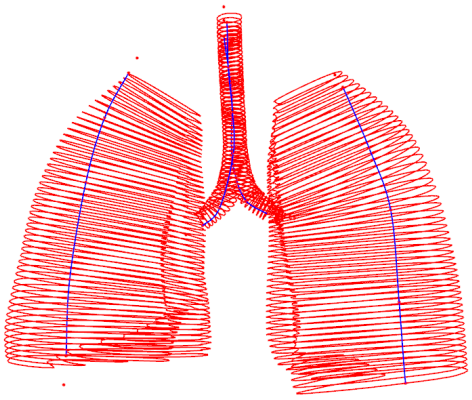

Figure 5. Some of the sections extracted from a lungs dataset. The sketched axes are shown vertically. Control points depicted only for the left lung.

B. Cross sections

To model the contour lines, we extract sections from the object's geometry and represent them as parametric curves. The sections are defined by the intersection of a set of planes and the objects' triangle mesh (Figure 5). We uniformly distribute the planes along each axis defined by the user through the sketch-based interface. The planes are oriented perpendicularly to the axis' tangent vector. The number of planes used in each axis varies with the complexity of the object.

Assuming a triangle mesh with a set of axes curves, the extraction is performed as follows for each axis curve:

- First, we sample points (one for each plane) equally spaced along the axis curve; for each point, we calculate the curve tangent and, so, define a plane perpendicular to the curve;
- For each plane, we calculate the intersection with the mesh edges; each section is modeled as a parametric curve from the intersection points calculated;

When the hatching is composed (see Section VI) and the position of each contour line is calculated using the Perspective Contouring rules, the sections extracted will serve as a reference model for the final lines used in the rendering process. If a contour line lies between two consecutive sections, interpolation is used to find the appropriate curve.

C. Contour lines

The contour lines are used to represent the surface of the object being depicted. The hatching made from these lines can be used to model shading in the illustration.

Each contour line is rendered as a triangle strip created along the correspondent section curve (Figure 6). To create the triangles, a set of uniformly distributed points are sampled from the section curve, and for each point two vertices are created. These vertices are displaced in opposite directions along the binormal vector of the point.

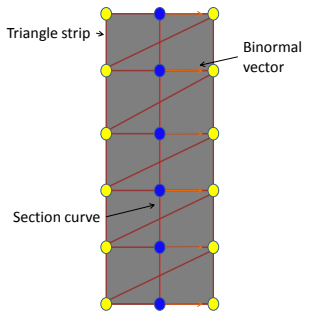

Figure 6. Triangle strip created to represent a contour line. The triangles' vertices are created from points in the (central) section curve.

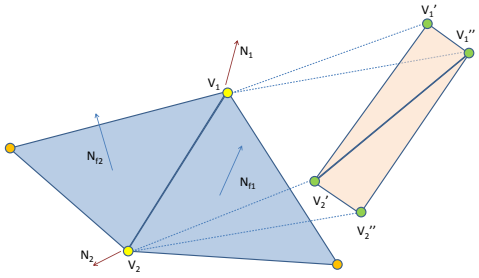

Figure 7. Silhouette edge. For one edge in the original mesh (V_1-V_2) we create four vertices in the silhouette mesh (rectangular patch).

The magnitude of the displacement is defined by the line width value calculated in the hatching composition step (see Section VI).

To avoid patterns, we avoid the regularity of the lines by applying small random variations to their width. This kind of noise also gives a more natural look to the illustration.

D. Silhouette lines

To model the silhouette lines, we use an adaptation of the method proposed by McGuire and Hughes[21]. In a preprocessing step, we create a structure called silhouette mesh that maintains all possible silhouette edges that can be rendered.

To create the silhouette mesh, we go through every edge from the original mesh, and for each one, we create four vertices in the silhouette mesh (Figure 7). These vertices (V_1', V_1'', V_2' and V_2'') have (two by two) the positions of the original vertices (V_1 and V_2) and keep the normal vectors from the two faces that share the edge in the original mesh (N_{f1} and N_{f2}). We also keep the normal vectors of the original vertices (N_1 for V_1' and V_1'' and N_2 for V_2' and V_2''). The new vertices are used to form two triangles that can be rendered as a silhouette line segment.

To determine the relation of each silhouette mesh vertex with the ROIs defined by the user, and thus, be able to calculate the line width for each of them, we save, for each vertex in the silhouette mesh, the closest point in the axes curves.

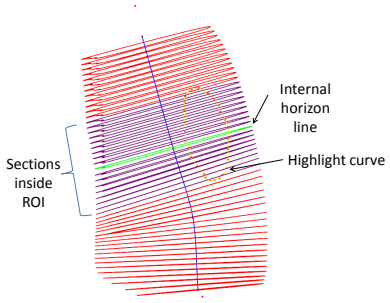

Figure 8. ROI defined in left lung. Some of the sections are shown (the purple sections are intersect by the highlight curve and, thus, belong to the ROI.

VI. Hatching composition

We are concerned with finding out a simple and intuitive manner to let the user determine the ROIs, which correspond to the regions where the internal horizon lines are placed in Perspective Contouring. In traditional illustration, the illustrator often uses lines called "'conceptual markings"' that usually are not part of the final drawing, but are used to help in the creation process. These markings can be used to indicate the basic structure of the object or to help in the construction techniques.

Inspired by this concept, we use a sketch-based interface to allow the user to draw on the model and define the ROIs. The definition of the ROI is combined with the definition of the highlight shape. The highlight is a result from the construction rules of the perspective contouring, since the length of the lines must be smaller when near the internal horizon line.

With ROIs and highlights defined, the construction rules of the Perspective Contouring are used to define the width and spacing of the contour lines around each ROI, as well as the width of the silhouette. The user can control the behavior of the lines attributes outside the ROIs through the sketching of function curves.

A. ROI definition

To define a ROI, the user draws a closed curve on the surface of the object. Internally, the ROI is defined by the set of cross sections extracted from the object (see Section V.B) that are covered by the curve (Figure 8). Within the ROI, the position of the internal horizon line is defined as the position of the central cross section.

The curve drawn for the selection of the ROI is also used to determine the shape of the highlight effect caused by the smaller length of the contour lines around the internal

horizon line. To determine the length of the contour lines, the intersection points of the curve of highlight and the sections within the ROI are calculated and stored to serve as endpoints to the triangle strips created to render the contour lines (see Section V.C).

B. Line creation

The lines' attributes within the ROIs (width and spacing for contour lines and width for silhouette lines) are set to achieve emphasis according to the Perspective Contouring rules. Outside the ROIs, the lines' attributes can be manipulated directly by the user to compose the desired illustration.

1) Inside the ROIs: The positions of the contour lines are calculated from the internal horizon line (position of the ROI's central section) towards its boundaries (first and last section of the ROI). To model the construction rules of the Perspective Contouring, we use linear functions that have as input the distance from the current section's position (represented by the point in the axis curve that generated the plane used to extract the section) to the internal horizon line and as output, values of width and spacing between two consecutive lines.

To calculate the width for the silhouette lines, the input for the function is the distance from the closest point in the axis curve, which holds the sections in the current ROI, to the internal horizon line.

The functions for spacing and width of the contour lines follow the same behavior and have the form:

$$value = s - d, \tag{2}$$

where, d is the distance to the ROI center and s is a scale factor. The silhouette width function requires a different behavior in the form:

$$value = sd, \tag{3}$$

From the ROI center, the contour lines are iteratively created with the distance d being updated. In each step, this distance is also used to find the previously extracted section that match the desired position for the new line. If there is no section defined in a certain distance, the two closest sections are interpolated. The intersection points with the highlight curve are also interpolated.

For each contour line to be calculated the steps below are followed (Figure 9):

- Calculate new width;
- Update distance to ROI center by half the calculated width;
- Find section using current distance to ROI center;
- Update distance to ROI center by half the calculated width;
- Calculate new spacing;
- Update distance to ROI center by the new spacing;

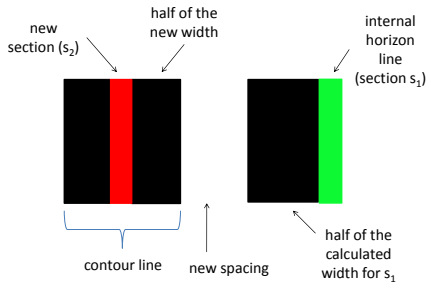

Figure 9. Calculation of a new contour line. From a previous calculated section (s_1), half the width is added to the distance to the internal horizon line (green). With a new spacing calculated and added to the total distance, the new width is calculated and also added. Finally, the new section (s_2) can be found among the extracted sections, and the whole process repeats.

The steps above are repeated for both sides of the internal horizon line until the extremes of the ROI are reached.

As the silhouette constantly changes due to the user interaction with the visualization, it is necessary to set the width of all edges of the silhouette mesh. Which edges must be rendered is calculated by a vertex shader in the graphics processing unit (GPU).

In the vertex shader, we test each vertex of the silhouette mesh to see if it is part of the current silhouette. The test consists in verifying if the dot products of the two normal vectors from the faces that share the edge to which the vertex belongs in the original mesh (in Figure 9, N_{f1} and N_{f2}) with the eye vector have different signs.

The difference in signs shows that the vertex is part of the silhouette and, hence, should be drawn. For that to happen, we displace the vertex in the direction of the normal from the original vertex of the model's mesh (in Figure 9, N_1 and N_2). Doing this, the triangles formed from these vertices will become visible.

If the test fails, we do not displace the vertex and avoid its rendering by modifying the alpha value of its color.

2) Outside the ROIs: Outside the ROI, the construction of the contour and silhouette is the same, but with different functions. In the contour line creation process, the lines are created from the borders of the ROIs following a direction away from the ROI center until the end of the axis curve or until half the distance to a neighbor ROI.

The functions used to configure these lines are defined by the user, through a curve sketched in a special panel (Figure 10) in the user interface. The panel consists in two parts, one for each side of the ROI (left and right), the user being able to draw one curve in each side. The curves act as usual function curves, where the horizontal axis represents the extension of the axis curve from the borders of the ROI to an extreme or to a mid-point between two neighbor ROIs, and the vertical axis represents the value being considered (width

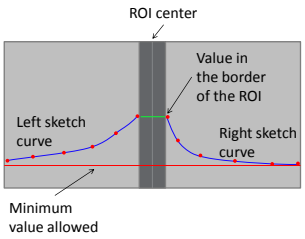

Figure 10. Sketch-based interface used to control the lines' attributes outside the ROIs.

or spacing for contour lines and width for silhouette lines). The user can change which attribute is being represented in the panel at any time, and panels belonging to neighbor ROI can be visualized side by side to allow continuity when drawing the curves.

VII. Results

The algorithm was implemented in C++ and OpenGL using GLSL for the shaders. The models used are represented as triangle meshes.

The interface consists in a main window where the user can see a shaded rendering of the model as well as interact with it (through translations and rotations) and sketch the axes for section extraction and the highlight curves for ROI definition, and a secondary window, inside the main window where the user can sketch the function curves to control the lines' attributes outside each ROI. The main window also has some menus to help in the organization of the creation process.

Figure 12 show an illustration created from the lung dataset. Figure 13 shows our technique applied to a torus.

VIII. Conclusion

This work presented a new approach in the creation of digital line illustration through the utilization of perceptual principles to allow the user to emphasize regions of interest and create focus of attention in the visualization.

Our technique allows the creation of contour line illustrations and the application of emphasis as the user wishes to improve the interpretation of the image. The user has also the power to control how the lines outside the regions of interest behave to create a smoother and more harmonious image.

We developed a sketch-based interface so the user can accomplish the main tasks of the process of creating an illustration, like defining axis curves used to orient the contour lines, regions of interest, highlights and function curves to control the lines' attributes, in an intuitive manner.

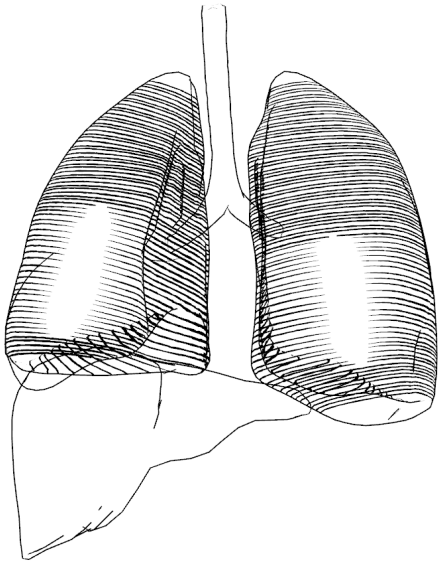

Figure 11. Illustration from the lungs dataset. Emphasis applied on both lungs and silhouette lines used to give context information.

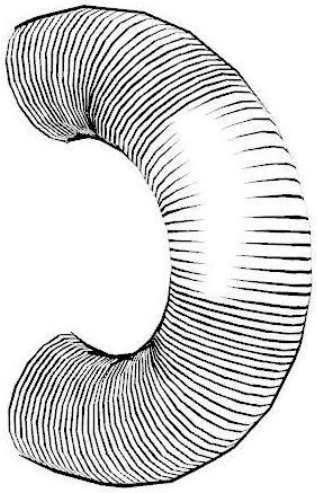

Figure 12. Illustration created from a torus.

The images generated show how focus of attention can be created through line attribute manipulation, like width, spacing and length. The perspective contouring technique can create the illusion that a determined region advance towards the observer.

The concepts involved in the technique developed herein are vastly used by traditional illustrators and we have shown that they can be adapted to the digital visualization of data. The work developed is a step in this direction, where the artistic knowledge, already consolidated for many centuries, can contribute even more to the visualization research.

As future work, we intend to make a full evaluation of the algorithm with a group of professional illustrators. Some improvements in the algorithm can be achieved as the inclusion of an automatic skeleton extraction procedure to orient the contour lines, which can complement the axis curves defined by the user and can be modified by him. The interactive edition of the curves is also something that can greatly help the creation process, allowing the user to refine the curves or correct minor details without having to redraw the entire curve.

REFERENCES

[1] C. Ware, *Information visualization: perception for design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000.

[2] I. Viola, M. E. Grller, K. Bhler, M. Hadwiger, B. Preim, D. Ebert, M. C. Sousa, and D. Stredney, "Illustrative visualization," *IEEE Visualization 2005 Tutorial*, 2005.

[3] I. Viola, M. C. Sousa, D. Ebert, B. Andrews, B. Gooch, and C. Tietjen, "Illustrative visualization for medicine and science," *Eurographics 2006 Tutorial*, 2006.

[4] A. Appel, F. J. Rohlf, and A. J. Stein, "The haloed line effect for hidden line elimination." in *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1979, pp. 151–157.

[5] T. Kamada and S. Kawai, "An enhanced treatment of hidden lines," *ACM Trans. Graph.*, vol. 6, no. 4, pp. 308–323, 1987.

[6] D. Dooley and M. F. Cohen, "Automatic illustration of 3d geometric models: lines," in *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*. New York, NY, USA: ACM, 1990, pp. 77–82.

[7] A. Girshick, V. Interrante, S. Haker, and T. Lemoine, "Line direction matters: an argument for the use of principal directions in 3d line drawings," in *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*. New York, NY, USA: ACM, 2000, pp. 43–52.

[8] V. Interrante, "Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution," in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 109–116.

[9] G. Elber, "Line art illustrations of parametric and implicit forms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 71–81, 1998.

[10] ——, "Interactive line art rendering of freeform surfaces," *Proceedings of EuroGraphics'99 (Milano, Italy, sep 1999)*, vol. 18, no. 3, pp. 1–12, sep 1999.

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

[12] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin, "Interactive pen-and-ink illustration," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1994, pp. 101–108.

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

[14] ——, "Rendering parametric surfaces in pen and ink," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1996, pp. 469–476.

[15] C. Rössl and L. Kobbelt, "Line-art rendering of 3D models," in *Proceedings of Pacific Graphics 2000.*, 2000. [Online]. Available: citeseer.ist.psu.edu/rossl00lineart.html

[16] O. Deussen, J. Hamel, A. Raab, S. Schlechtweg, and T. Strothotte, "An illustration technique using hardware-based intersections and skeletons," in *Proceedings of the 1999 conference on Graphics interface '99*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 175–182.

[17] F. Ritter, C. Hansen, V. Dicken, O. Konrad, B. Preim, and H.-O. Peitgen, "Real-time illustration of vascular structures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 877–884, 2006.

[18] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

[19] R. H. Bartels and F. F. Samavati, "Reversing subdivision rules: Local linear conditions and observations on inner products," *Journal of Computational and Applied Mathematics*, vol. 119, pp. 29–67, 2000.

[20] J. J. Cherlin, F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling with few strokes," in *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*. New York, NY, USA: ACM, 2005, pp. 137–145.

[21] M. McGuire and J. F. Hughes, "Hardware-determined feature edges," in *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*. New York, NY, USA: ACM, 2004, pp. 35–47.