Issues on Displaying 3D Data for Scientific Visualization

Thomas Theußl
theussl@cg.tuwien.ac.at

Institute of Computer Graphics
Vienna University of Technology
Karlsplatz 13/186/2,A-1040 Vienna, Austria
http://www.cg.tuwien.ac.at/home/

Abstract

Scientific visualization often deals with three-dimensional data. Either the data is inherently three-dimensional, e.g., flow simulation data or medical data, or 3D is chosen as the biggest reasonable projection space for high-dimensional data, e.g., dynamical systems or databases. Since conventional displays usually are two-dimensional, this data has to be transformed to two dimensions which leads to information loss. Objects in front occlude others lying behind them, lines appear to coalesce into a unified solid surface of indeterminate depth. This paper gives a short overview of some approaches which try to enhance the spatial impression of three-dimensional data displayed as two-dimensional pictures.

Keywords: perception, visualization, line shading, transparent surfaces.

1 Introduction

In the field of computer graphics many techniques and algorithms have been developed to render three dimensional data on two dimensional displays, often trying to do this as accurate as possible. In scientific visualization, however, generating images which provide maximal insight into three-dimensional data is more important than rendering an image in a photo-realistic way.

Particularly when depicting large surfaces occlusion becomes a major problem. Objects that lie behind or within these surfaces cannot be seen. Using transparency improves the situation but still there are problems: transparent surfaces provide little cues on their shape, especially the shape of the front-facing portion usually cannot be perceived accurately. Further, multiple superimposed transparent layers cannot be distinguished at all and it is almost impossible to estimate the distance in depth between two such layers. Small, sparse, opaque marks on a transparent surface facilitate the perception of both the shape and depth of a layered, transparent surface without impairing too much the visibility of objects located behind it. By taking into account the curvature of the surface when applying these marks, the perception of the shape of the surface itself can even be enhanced.

Another important issue is what part of the data should be depicted not to obtain pictures which are overloaded and confusing. Especially when visualizing three-dimensional flow using stream surfaces it is not only necessary to consider how to render these surfaces but also which of the infinite possible surfaces to choose [2]. Also the distribution of streamlines is not trivial [17].

When depicting lines in three dimensional space in general, occlusion is also a problem [17], but more problematic is that these lines when not in motion tend to be flattened, they appear to be in one plane [7]. The use of common lighting models and the thereby achieved shading effects give important cues for the spatial perception, but there are also other approaches that try to enhance the perception of lines in three dimensional space.

2 Enhancing the perception of lines in 3D

To visualize three-dimensional vector fields it is often necessary to depict three-dimensional lines, for example stream lines. Since lines in 3D do not have a unique normal vector, they either have to be flat shaded, impairing the spatial impression of the image, or polygonal tubes have to be used, limiting the number of streamlines that can be displayed in a scene. Another approach would be to use Line Integral Convolution [1, 14], a commonly used algorithm for visualizing vector fields, which is easily extended to 3D. Unfortunately, the results are not as good as in 2D [2], too many details obscure the flow topology. Below three approaches are described which depict lines in three-dimensional space. The first illuminates streamlines, the second tries to ray-cast vector fields and the third describes some strategies to enhance 3D Line Integral Convolution.

2.1 Illuminated Streamlines

Although lines in $\mathbf{R}^3$ do not have a unique normal vector, Zöckler et al. [17] illuminate streamlines using the popular reflection model of Phong. They select from all the normal vectors the one coplanar to the light vector L and the tangent vector T. Let N be this normal vector, L the light direction, V the viewing direction, and R the unit reflection vector (the vector in the L-N-plane with the same angle to the surface normal as the incident light). Then the light intensity at a particular point is given by

$$\begin{aligned} I &= I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \\ &= k_{\text{a}} + k_{\text{d}}\ L \cdot N + k_{\text{s}}\ (V \cdot R)^n \end{aligned} \tag{1}$$

Since light reflection on stream lines increases the spatial impression of the resulting images, lines shaded with this method provide stronger depth cues (see Figure 1). Furthermore, texture mapping capabilities of modern graphics hardware can be exploited to render these lines efficiently, achieving high frame rates even when large numbers of lines have to be rendered.

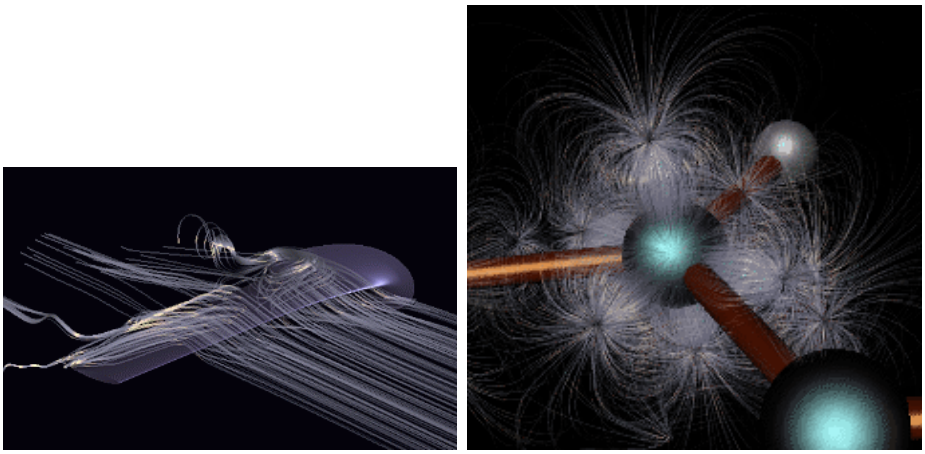

Figure 1: Illuminated Streamlines. The flow around a wing on the left and the electrostatic field of a benzene molecule on the right.

A drawback of this method is that the normal vector is not constant but a projection of the light vector into the normal plane of the stream line. This means that the angle between the light vector and the normal vector is minimized, resulting in a more uniform brightness than we are used to perceive in real world. This effect yet can be compensated by exponentiating the diffuse reflection term in Equation (1).

2.2 Ray-casting Vector Fields

Ray-casting [8] is a direct volume rendering method, where rays are cast into a 3D array of values and for each ray a vector of sample colors and opacities is computed by re-sampling the voxel database at k evenly spaced locations.

When ray-casting a vector field [3], there is exactly one stream line crossing the viewing ray at each sampling point. To shade the sampling color accordingly, there is again the problem that lines in 3D do not have a unique normal vector. Since there is one plane to which the local velocity vector is normal, the vector which points mostly towards the viewer while lying in this plane is chosen as the normal vector. If the light source is located at the viewing position, the shading computation is simplified and flow towards or away from the viewer appears dark and flow normal to the viewing direction appears brightest. However, this approach leads to visualizations that are difficult to interpret if only a still image is used (see Figure 2 on the left). The spatial orientation of the flow is much better perceived if viewpoint animation is used.

To enhance the spatial impression in single images, it makes sense to use pseudo-color to express velocity magnitude (see Figure 2 on the right) or even vector field direction. Therefore, the angle between the streamline normal and the light source is mapped to sample hue, so we can see where the flow is directed towards or away from the light source, i.e., the viewing direction here. The use

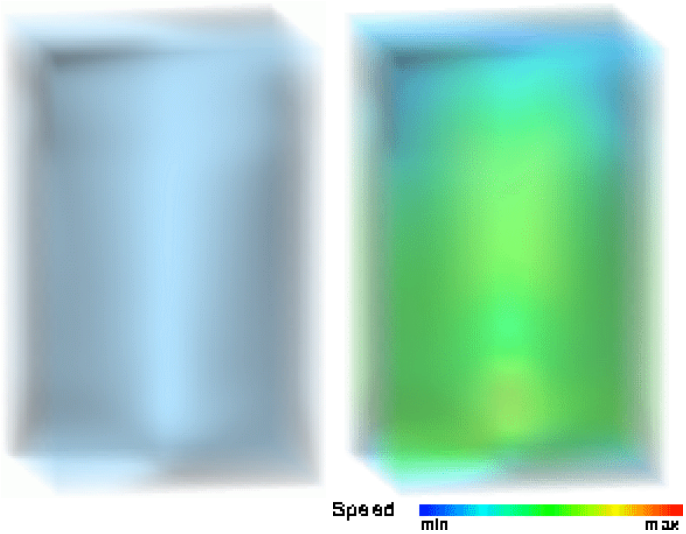

Figure 2: Ray-casted vector fields. On the left streamline shading without color mapping, on the right color is mapped to velocity magnitude. Light source in both pictures at viewing position.

of pseudo-color yet for the visualization of vector field direction is something to which the user first has to become accustomed to.

2.3 Volume Line Integral Convolution

Line Integral Convolution (LIC) is an elegant algorithm for visualizing vector fields. Very good results are achieved for 2D vector fields, but extending it to 3D arises some problems. The pictures get confusing, too many details obscure the flow topology. Some modifications to the original algorithm have been proposed by Interrante and Grosch [7] to achieve better results in three dimensions.

The texture used with 2D LIC would normally be an opaque, uncorrelated white noise. This has the drawback that the orientation of the vector field cannot be perceived. Wegenkittl et al. [16] developed a method, Oriented Line Integral Convolution (OLIC), where they use a low frequency input texture and a ramp like convolution kernel to depict both direction and orientation of the flow. This, further, inspired Interrante and Grosch to enhance 3D LIC by using a sparsely opaque input texture and to use LIC to correlate both color and opacity values in the direction of the flow. They also proposed that the input spots in the volume should be randomly situated according to an approximate Poisson-disk distribution, rather than laid out purely randomly. See Figure 3 for an example.

If the stream lines are shaded as described in Section 2.1, the local orientation of the flow can be clearly depicted but streamlines that are separated in depth but which flow in a similar direction cannot effectively be distinguished. The following techniques have proven useful for clarifying the display of volume textures generated via 3D LIC:

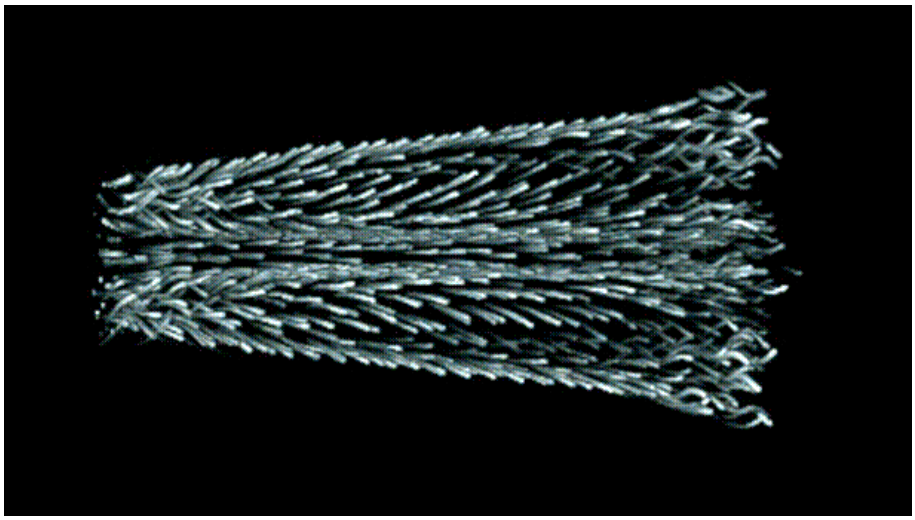

Figure 3: Volume Line Integral Convolution from an input texture of evenly-distributed random point samples. Shading is computed as described in Section 2.1.

- By assigning each of the different input texture elements one of a small number of harmonious yet readily distinguishable different hues, the differentiating of the distinct lines is facilitated. Colors can be randomly distributed among the input points or may alternatively be assigned on the basis of a function computed either over the streamline or its point of origin. Care must be taken with the second approach to prevent the perceptual clumping of elements in homogeneously-valued regions, for example by assigning random luminance variations.

- Artists and illustrators often use gaps to indicate the depth relation if a line crosses behind another. To depict such gaps, a second input texture, identical to the first except that it contains slightly larger spots at the same location as the smaller spots in the original texture, is used during volume rendering to decrease the contribution to the final image of any voxel encountered after a halo, that is the margin that surrounds every line in the second input texture, has been entered and subsequently exited, emphasizing the depth discontinuities in such a way.

- Another approach to facilitate local depth order judgements would be to use stereo, or to animate the lines so that they follow the flow.

3 Enhancing the perception of layered surfaces

Many applications benefit from displaying multiple surfaces. When these surfaces are depicted opaquely, major parts of the model are occluded. This makes it important to render layered surfaces in such a way, that outer structures can be seen and seen through the same time. Using transparency to show what lies behind or within surfaces can be a useful device. Unfortunately, the shape of transparent

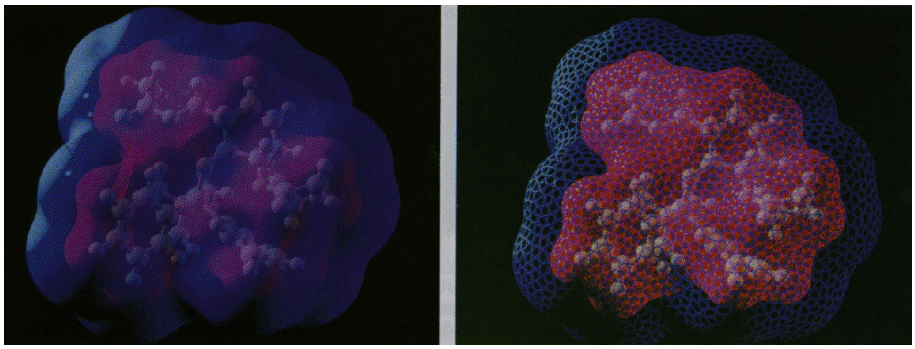

Figure 4: Multiple transparent surfaces on the left and multiple opacity-modulating surfaces on the right

surfaces becomes more difficult to perceive accurately, for ordinary depth cues of shading and occlusion are minimally present on them. Adding partly transparent, partly opaque textures to these surfaces achieves perceptual benefits. Several approaches are described below.

3.1 Opacity-modulating Triangular Textures

Rheingans [13] suggested a straightforward approach which uses conventional texture mapping techniques to apply a pre-computed two-dimensional texture to a surface in 3D. These surfaces are assumed to be composed entirely of triangles, for arbitrary polygon meshes can easily be transformed into triangle meshes.

The texture should have certain characteristics in order to give strong shape cues on surfaces:

- It must be able to be tiled without seams, since such seams might draw the observers attention without providing useful information.
- It must have both enough transparent parts for the objects behind to be seen clearly and enough opaque parts for the surface on which it lies to be clearly perceived. For these are directly at odds with one another, the right balance must carefully be achieved as the case arises.
- It should be easy to generate and quick to render. Therefore, it should be a precomputed image stored in texture memory rather than defined by a procedure evaluated during the shading calculations.

Since the polygons the texture is to be applied to are all triangles, it is obvious to define an equilateral triangular texture element which is easily mapped to the triangles. Furthermore, if the sequence of texture values along each side is identical, and consequently the values at the texture extreme points are the same, any side of the texture element will match seamlessly with any other side.

If the texture element is defined on an equilateral triangle, the textured surface will appear most regular when the triangles on the surface are also equilateral.

Unfortunately, surface extraction methods like the marching cubes algorithm [11] normally do not produce equilateral triangles. So texture pattern regularity must be improved by a preprocessing step which regularizes the polygonal tessellation producing triangles more uniformly sized and closer to equilateral, to avoid disruption of the texture regularity. Rheingans used a re-tiling algorithm developed by Turk [15]. Figure 4 compares multiple transparent surfaces to multiple opacity-modulating surfaces.

3.2 Feature Lines

Using feature lines as a sparse, opaque texture on transparent surfaces [5, 12] was inspired by the ability of gifted artists to define a figure with just a few strokes. The two types of geometric features most often represented in line drawings are discontinuities of depth (silhouettes and contours) and discontinuities of curvature (sharp ridges and valleys). Depicting this small set of meaningful lines gives strong cues on the shape and position of transparent surfaces.

Silhouette and contour curves are the 2D-projection of points on the 3D surface where the direction of the surface normal is orthogonal to the line of sight. Silhouette lines are always visible for they form a closed outline around the projection. Contour curves may be disjoint an can fall within the projective boundary. Although contour curves are important shape descriptors, their use under conditions of stereo and motion are limited. Because they are viewpoint-dependent, they must be recomputed and redrawn every time the viewpoint changes or they will seem to crawl over the surface confusing the perception of the surface data. Furthermore, these contour lines provide little indication of depth distance or surface shape across forward-facing areas.

Another sparse set of descriptive lines, but which remain fixed on the surface under dynamic viewing conditions, are ridge and valley lines. To detect these feature lines some characteristics of surfaces must be examined. At any non-spherical point on a generic, smooth surface is one direction in which the the surface is curving most strongly. This direction is referred to as the first principal direction and the curvature in this direction is referred to as the first principal curvature. These two can easily be computed at arbitrary points on a smoothly curving surface from the eigenvectors and eigenvalues of the second fundamental form [5].

Valley lines now are the locus of points on a surface where the normal curvature assumes a local minimum in the principal direction associated with the largest, negative curvature and ridge lines are the locus of points on a surface where the normal curvature assumes a local maximum in the principal direction associated with the largest, positive curvature. Every point classified to lie on or near a ridge or valley line is assigned an additional amount of opacity and a slightly different color to better distinguish between these two kinds of feature lines.

If all the points identified by the preceding definition are displayed opaquely, the result is not satisfactory, too much lines are displayed. So steps must be taken to selectively emphasize more important ridge and valley regions and de-emphasize the others:

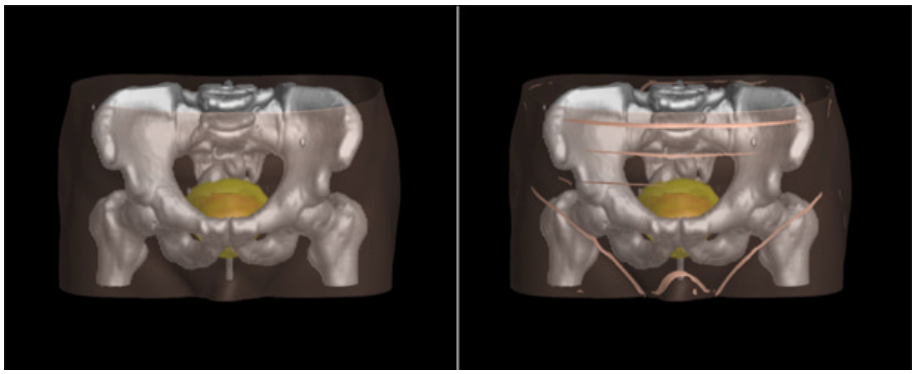

Figure 5: An untextured skin surface on the left and the same skin surface with ridge and valley texture on the right.

- First, the opacity of the ridge and valley points can be defined as a function of the relative magnitude of the normal curvature in the first principal direction. This emphasizes more sharply curved regions.
- Next, only these ridge or valley points are assigned an additional amount of opacity, which magnitude of the maximum principal curvature at that point exceeds a specified cutoff.
- It is possible, that after these two steps very deep, but extremely narrow spurious ridge or valley lines remain visible. The approach to eliminate these lines is to step away from the point in the first principal direction and look if the approximated surface normals begin to realign before a specified minimum distance has been traversed.

Figure 5 shows a treatment plan for a patient with prostate cancer.

3.3 Curvature-directed Strokes

Not all surfaces can be sufficiently characterized by feature lines. There are situations in which a more continuous representation of both areas where the surface shape is changing and areas across which it remains relatively uniform would be desirable. The use of curvature directed strokes [6] was again inspired by artist's use of lines to show shape. Our perception of a surface's form is strongly affected by the choice of line direction used to represent it. Several different basic techniques are commonly used. Strokes uniformly directed across an entire image tend the objects appear to be flattened, vertically-oriented strokes emphasize height and horizontally-oriented strokes emphasize width. But the effect that has been chosen to pursue in this work is to align the stroke direction with the direction of strongest curvature of the surface.

The first step is to iteratively select points as evenly distributed as possible over the surface around each texture element will be centered. Next, principal direction, for the direction of the stroke, and principal curvature, for the length of the stroke,

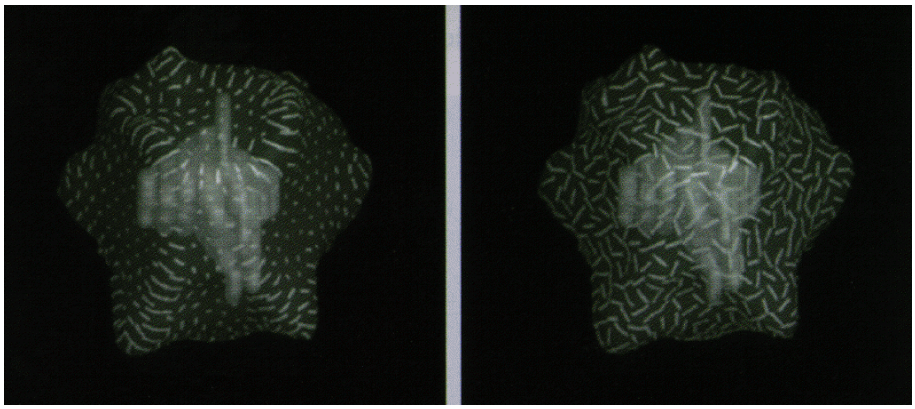

Figure 6: Transparent outer shell with principal direction texture on the left compared to a transparent outer shell textured with randomly-oriented constant length strokes on the right.

are computed at each centerpoint. Each stroke is now modelled as a "slab" with a length dependent of the principal curvature, a height large enough to allow each slab to contain the surface across its fullest possible extend without being so large that it opacifies the surface in unintended areas, and a properly chosen width. The geometrical definition of each individual stroke is now simply passed to the volume renderer (see Figure 6).

3.4 Principal Direction-driven 3D LIC textures

Although the results achieved with the method described in Section 3.3 are quite good, the modelling of the "slabs" as geometric objects is not satisfying. Furthermore, these "slabs" have to be recomputed for each isosurface out of the same set of data, which is cumbersome in cases where it is necessary to view not one but multiple level surfaces through a volume distribution.

Interrante [4] proposes a method where the set of principal directions and principal curvatures is used to define a natural flow over the surface of an object. Since the first principal direction is tangential to the surface with direction of the strongest curvature, each stream line will lie on an isosurface aligned to the curvature of this surface. The set of principal directions can be precomputed for the whole dataset and via 3D Line Integral Convolution it is now possible to generate a solid stroke texture that illustrates the essential shape information of any level surface in the data.

Figure 7 compares a transparent surface to a surface with a texture generated via 3D Line Integral Convolution and the set of principal directions applied, Figure 8 depicts a series of six surfaces textured by the same method.

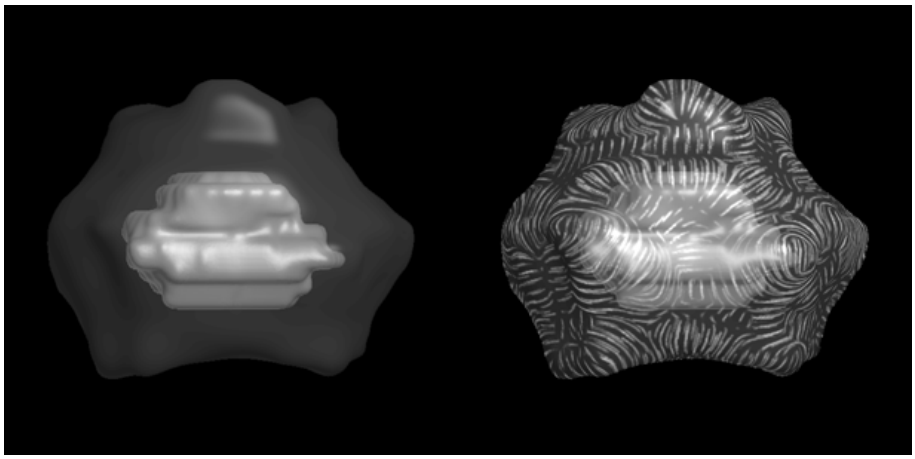

Figure 7: Transparent surface on the left and the same surface on the right with a texture applied generated via principal direction-driven 3D Line Integral Convolution

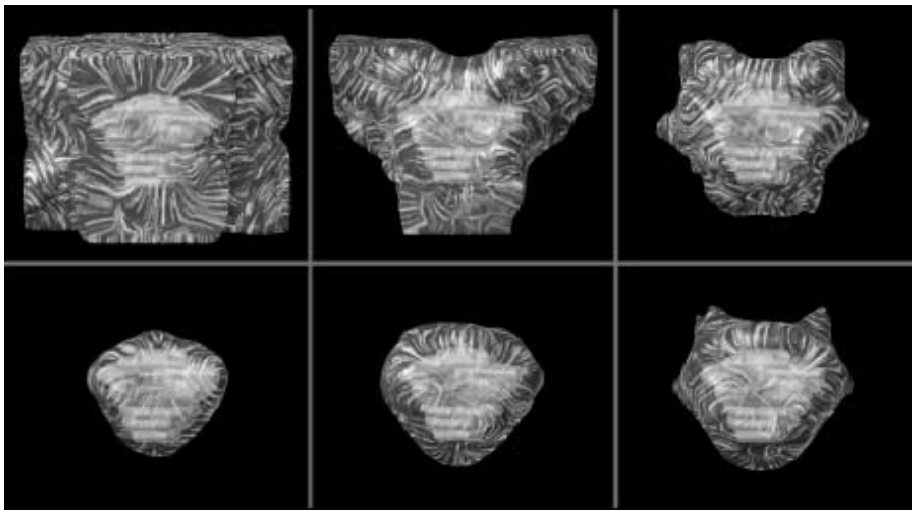

Figure 8: A series of six level surfaces with the same texture applied generated via 3D Line Integral Convolution and the set of principal directions.

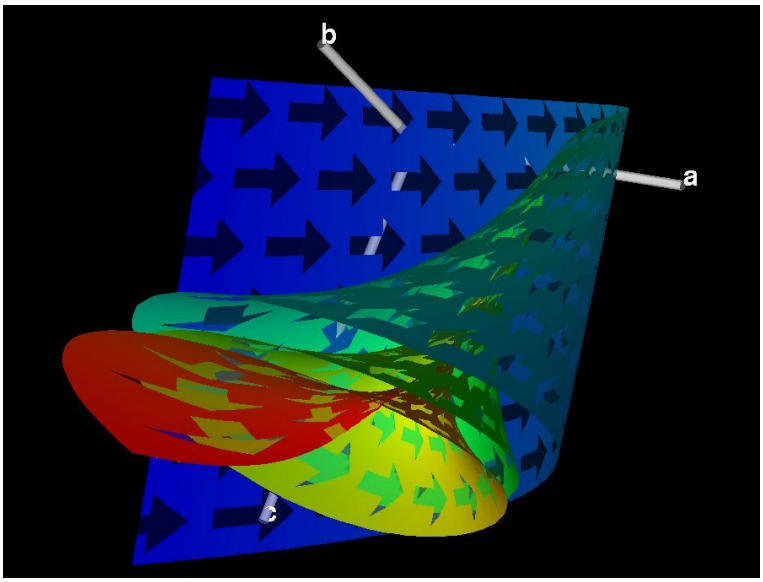

Figure 9: Stream arrows on a stream surface of a dynamical system.

3.5 Stream Arrows

In Section 2 several approaches were discussed which try to visualize three-dimensional flow by depicting lines in space. Another approach would be to visualize dynamical systems by depicting stream surfaces. A major problem thereby is that such surfaces often occlude major parts of the model. Again, transparency alone does not solve the problem for transparent surfaces provide little depth cues. Furthermore, since flow has a direction it would be desirable to visualize this direction too.

An extension to standard stream surfaces that on the one hand lets one see through the surfaces and on the other hand depicts the direction of the flow is the use of stream arrows [10]. Therefore, arrow-shaped textures are mapped to the surface, which both depict the direction of the flow and can be made transparent to show what lies behind them (however, it is also possible to make the remaining surface transparent and depict the stream arrows opaquely). Figure 9 shows a stream surface with stream arrows which, since they are transparent, let one clearly see the inner structure of the dynamical system.

The problem with this approach is that the arrows are equal-sized in texture-space so they tend to become too big or too small in areas where stream surfaces spread over regions of high divergence or convergence. To generate stream arrows that are almost equal-sized in the final image it is better to use hierarchical stream arrows [9]. Therefore, a stack of stream arrows textures is defined, where the scale relation of the arrows between successive levels is defined by a constant factor. The ratio between the size of a mesh in texture space and in phase space is then used to find the most appropriate level in the stack, which makes the arrows almost equal-sized in the rendered picture. It would not be useful to define a continuous size function for this purpose because the ability to represent local divergence or

convergence would be lost then.

4 Conclusions

When visualizing three-dimensional data on two-dimensional displays there are several points that have to be considered.

Lines in 3D do not have a unique normal vector, integrating over all normal vectors cannot be done efficiently, so one vector has to be chosen to shade the lines accordingly, for shading very much increases the spatial impression of the resulting images. Illuminated stream lines, where the vector coplanar to the light and tangent vector is chosen as normal vector, give a much better impression of the vector field structure than flat shaded stream lines. However, care must be taken to select the seed points in such a way that the resulting images do not get overloaded and confusing. Detecting characteristic structures and depicting them properly with selected stream lines reduces occlusion and enhances the perception of the vector field structure.

Occlusion yet becomes a major problem when depicting surfaces. Transparency is useful to let one see through surfaces but impairs the spatial impression of the surface itself and the relative depth of multiple layered surfaces. The general approach is to use a sparse, opaque texture so the surface can be seen and seen through the same time. Standard hardware texture-mapping routines can be used to apply a opacity-modulating texture to a surface. This increases very much the perception of the shape of the surface itself but often impairs the perception of the inner structures. Other approaches try to emphasize characteristic parts of the surface, e.g., by depicting ridge and valley lines or by defining strokes dependent of the curvature of the surface. When depicting stream surfaces it is desirable to visualize the direction of the flow too what is nicely done by stream arrows, arrow-shaped textures mapped to the stream surfaces.

References

[1] Brian Cabral and Leith (Casey) Leedom. Imaging vector fields using line integral convolution. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 263–272, August 1993.

[2] Wenli Cai and Pheng-Ann Heng. Principal stream surfaces. In *IEEE Visualization '97*, October 1997.

[3] Thomas Frühauf. Raycasting vector fields. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.

[4] Victoria Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Computer Graphics, Annual Conference Series*, pages 109–116, 1997.

[5] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *IEEE Visualization '95*, pages 52–59, 1995.

[6] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Illustrating transparent surfaces with curvature-directed strokes. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.

[7] Victoria Interrante and Chester Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings of Visualization '97*, pages 421–424, 1997.

[8] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[9] Helwig Löffelmann, Lukas Mroz, and Eduard Gröller. Hierarchical streamarrows for the visualization of dynamical systems. In Wilfrid Lefer and Michel Grave, editors, *Visualization in Scientific Computing '97*, pages 155–163. Springer, 1997.

[10] Helwig Löffelmann, Lukas Mroz, Eduard Gröller, and Werner Purgathofer. Streamarrows: Visualizing multiple layers of streamsurfaces. *The Visual Computer*, 13(8):359–369, 1997.

[11] William E. Lorensen and Harvey E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In M. C. Stone, editor, *SIGGRAPH '87 Conference Proceedings (Anaheim, CA, July 27–31, 1987)*, pages 163–170. Computer Graphics, Volume 21, Number 4, July 1987.

[12] Kwan-Liu Ma and Victoria Interrante. Extracting feature lines from 3D unstructured grids. In *Proceedings of Visualization '97*, pages 285–292, October 1997.

[13] Penny Rheingans. Opacity-modulating triangular textures for irregular surfaces. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.

[14] Detlev Stalling and Hans-Christian Hege. Fast and resolution independent line integral convolution. In *ACM SIGGRAPH Computer Graphics Proceedings*, Annual Conference Series, pages 249–256, August 1995.

[15] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):55–64, July 1992.

[16] Rainer Wegenkittl, Eduard Gröller, and Werner Purgathofer. Animating flowfields: Rendering of oriented line integral convolution. In *Computer Animation '97*, pages 15–21, June 1997.

[17] Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Interactive visualization of 3d-vector fields using illuminated streamlines. In *Proceedings of IEEE Visualization '96, San Francisco*, pages 107–113, October 1996.