Flow Visualization using Illustrative Line Styles

Maarten H. Everts, Henk Bekker, Jos B.T.M. Roerdink, and Tobias Isenberg
Johann Bernoulli Institute for Mathematics and Computer Science
University of Groningen
maarten@bitpuzzle.com, {h.bekker | j.b.t.m.roerdink}@rug.nl, isenberg@cs.rug.nl

Abstract

We present a flexible illustrative line style model for the visualization of streamline data. Our model partitions view-oriented line strips into parallel bands whose basic visual properties can be controlled independently. We thus extend previous line stylization techniques specifically for visualization purposes by allowing the parametrization of these bands based on the local line data attributes. We demonstrate the effectiveness of our model by applying it to 3D flow field datasets.

1 Introduction

The flow of fluids and gases plays an important role in a wide variety of real-world phenomena. Examples include the aerodynamics of cars, the heat distribution in offices, and the airflow around falling ink droplets. Consequently, flow has been extensively studied, typically through three-dimensional simulations. These simulations yield large amounts of data containing information at multiple scales; for some applications the general structure of the flow is most relevant, for others the small local deviations are the subject of study. Visual representations of flow data help in understanding its behavior and over the years a large number of methods have been developed for this purpose. Initially, most flow visualization methods employed photorealistic rendering techniques, but later-on also methods that borrow principles from scientific illustration were developed.

Inspired by such illustrative visualization techniques [18], we present a flexible method for illustratively depicting streamlines generated from 3D vector fields. We achieve this flexibility by introducing a line style model whose parameters can be interactively manipulated, thus facilitating the interactive exploration of the parameter space of visual streamline representations. This allows the user to select and generate the representations that are most suitable for the data and communication goals at hand.

In order to achieve flexible parametrization of line styles we generalize a previous illustrative approach for line visualization [3], by subdividing the view-oriented line strips that represent the streamlines. These strips are split into bands that are arranged orthogonally to a line's direction, and whose shape, color, relative distance to the viewer, and width can be independently controlled. In addition, we allow these line parameters to individually depend on local data attributes such as temperature or velocity.

In summary, the contributions of this paper are a flexible line style model for use in scientific streamline visualizations and a fast yet flexible implementation of this model on the GPU. We demonstrate the power of our approach for a number of 3D flow datasets that exhibit complex flow patterns.

2 Related Work

In this section we discuss related work in the fields of flow visualization and illustrative visualization.

2.1 Flow Visualization

Being one of the most fundamental subjects for visualization, a broad range of methods have been developed for the visualization of flow datasets. McLoughlin et al. [16] survey both flow visualization in general and integration-based, geometric flow visualization in particular. A key component of integration-based flow visualization methods is the use of geometric objects to depict the properties and structure of the flow. These objects are generated by integrating over the underlying velocity field—starting from a set of seed points.

Lines are the most widely used primitives for this purpose, and in the context of steady flow such trajectories are called streamlines, whereas for unsteady flow streak and pathlines are used. The challenge for the visualization of three-dimensional streamlines is overcoming perceptual problems. Simply rendering large numbers of lines can quickly lead to clutter and occlusion, not to mention the fact that the general thinness of a line makes it hard to convey depth and spatial relationships.

Solutions to deal with these perceptual challenges include careful placement of streamlines through seeding strategies (see McLoughlin et al. [16] for an overview), illuminated streamlines [26], and shaded tubes or ribbons [25]. Particularly relevant for this paper are the approaches that employ textured view-oriented triangle strips to mimic shaded tubes [20, 22]. Such shaded primitives need to have a certain width for the depth perception to work and have only a limited number of visual variables to convey additional information: typically only width and color, although textures can also be used to convey information about the flow [22]. In terms of flexibility in controlling the appearance of flow

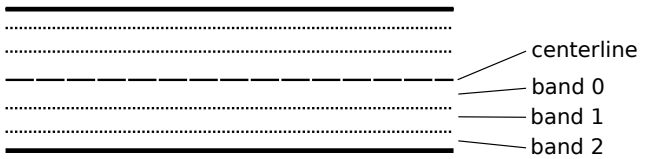

Figure 1: A view-oriented strip subdivided into a number of bands mirrored around the centerline.

streamlines, the approach by Shen et al. [21] that uses 3D flow textures is relevant.

2.2 Illustrative Visualization and Line Stylization

Illustrative visualization methods [18] use and apply the principles of (scientific) illustrators to achieve the clarity and effectiveness found in traditional illustrations. Naturally, many of these methods employ techniques from the field of non-photorealistic rendering (NPR). NPR methods particularly relevant to our work aim to replicate line drawings and, for this purpose, support different line styles.

Dooley and Cohen [2], for example, use dashing for illustrating geometric models, whereas the difference vectors of Schlechtweg et al. [19] permit a larger palette of styles. Other line style parametrization methods include stroke texturing [10, 11, 17], multi-resolution curves [4], skeletal strokes [7, 8], and programmable line styles [6, 9]. These NPR styles are typically applied to contour and feature lines of 3D objects, aim to replicate marks made by traditional tools, and—if used in an illustration—may carry a meaning (such as parts being hidden). In our work, in contrast, we use line styles to specifically visualize data properties of streamlines in a flow.

One important concept from the field of illustrative visualization important for our work is the use of halos [1,3,22,24] to make objects (including lines) easier to discern from the background, thus improving depth perception. In the context of flow visualization, other illustrative methods related to our work include stroke- and painting-inspired visualizations of 2D flow fields [12, 14], illustrative 3D volume rendering [23], stylized streamlines [15], as well as animated, dashed streamlines [13] and dashtubes [5].

3 Line Styles for Visualization

As we have seen, line-based flow visualizations are problematic due to their limited number of visual variables as well as the occlusion that is introduced if more than a few line primitives are being used. We address these two major issues by introducing an extended line style model for visualization purposes.

3.1 Line Partitioning Into Line Bands

Such an extended illustrative line style model needs to increase the number of visual variables to allow the specification and parameterization of a variety of effects that can be flexibly used in visualization. For this purpose and inspired by halo-based line visualizations [3], we partition the line strips that are used to render the data lines into several bands (Fig. 1). By that separation we provide the granularity that is necessary to allow us to define visually distinguishable styles, each of the bands increasing the number of visual variables that can be controlled. These bands run parallel to the centerline and together define the visualization line style.

Specifically, we represent each line from the 3D dataset by a view-oriented line strip as done in many previous line-based rendering systems. This strip is subdivided into two mirrored sets of bands, one on each side of the line (Fig. 1). The three visual properties that we control per band are color, width, and distance offset w.r.t. the viewer, each of which can be controlled independently. While the distance offset is not actually a *visual* property, it has an effect when used as a depth-dependent halo [3, 24]. In that case the halo line band is folded back, away from the viewer. The effect is that the *visible* width of the halo depends on the difference in distance between two lines w. r. t. the viewer, improving the depth perception. Therefore, our extended line style model can be seen as a generalization of the depth-dependent line halo technique [3].

3.2 Local Attribute Mapping

This basic line model allows us to specify a wide range of visual effects, notably by its capability to convey information about the data in the visualization by mapping data attributes such as temperature, pressure, etc. to a line's visual properties. Specifically, we control each line style band's color and width based on the value of local scalar line attributes by means of mapping functions.

For the color attribute, this mapping is encoded in conventional color maps that assign input values $\in [0,1]$ to RGB colors. We provide a selection of pre-defined color maps from which the user can choose a color map most suitable for that particular attribute type and the desired visual style. Similar to controlling the color of a band a user may adjust the width of a band to convey more information in the visualization. For example, mapping local velocity to band width yields wide bands where the velocity is high and thin bands where the velocity is low. To control this mapping, both a minimum and a maximum value can be set for the band width.

3.3 Flexible Band Shapes

The control of the line width property of a band can also be used to create bands with repeating line shape patterns such as dashes, droplets, etc. Moreover, the local density (or frequency) of these repeating shape patterns can be used to provide additional means for conveying local flow properties; this is particularly useful for the velocity property. For this purpose we employ (1) a *shape mapping function* and (2) a *dedicated line shape attribute*.

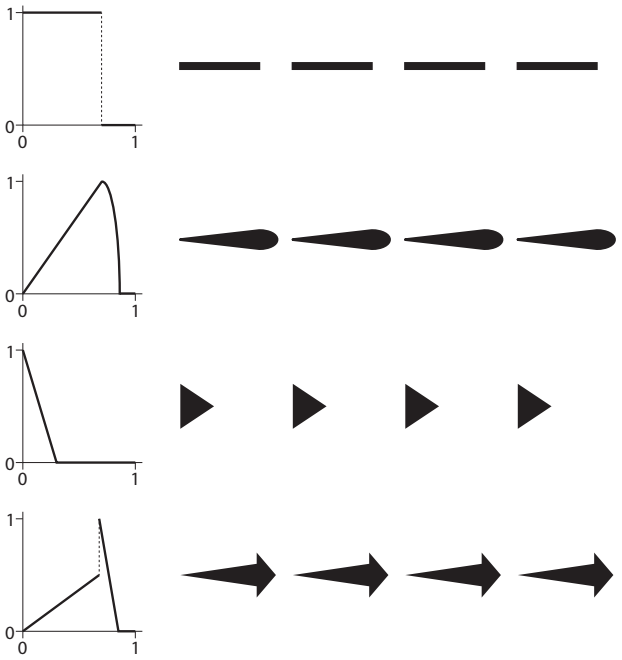

Figure 2: Shape mapping functions and corresponding line styles.

A *shape mapping function* maps a line data attribute ($\in [0,1]$) to the width ($\in [0,1]$) of the band at that point of a line. As such, it defines the shape of a band. We combine this mapping function with a *dedicated line shape attribute*:

$$s_x = \frac{x}{l} \bmod 1, \tag{1}$$

where l defines the size of the shape pattern on the line and x is a line attribute that is monotonically increasing along the line. The modulo operation ensures that the shape is repeated along the line.

The choice of x in Equation 1 determines the local 'density' of the patterns. For example, choosing the distance along the data line to the seed point results in constant size patterns. However, choosing the integration time t makes the frequency of the patterns depend on the local velocity of the field: high velocity will result in a lower frequency of patterns (i. e., elongated patterns), providing additional means for visualizing the local velocity.

Together, the line shape attribute and the shape mapping function provide a flexible way to achieve a wide range of line shapes. Fig. 2 shows a number of examples of mapping functions and illustrates how the mapping function influences the shape of a band and, thus, the line style.

3.4 The Extended Line Style Model for Visualization

Together, the line bands with their color and width control, the means to parametrize the band width with mapping functions, and the repeating shape patterns extend the number of visual variables available for visualization of line data. Most of these visual variables can be combined in one visualization and convey multiple aspects of the streamline data in one image.

4 Implementation

Several design decisions of the conceptual line model were driven by implementation considerations. More specifically, because we aim for the interactive exploration of line styles even when applied to large datasets, the line style model should be suitable for implementation in shaders on modern GPUs. Our implementation consists of two parts, each implemented in a different type of shader. The first is the generation of view-oriented triangle strips (geometry shader) and the second is the application of the line style to the strip (fragment shader).

The transformation of the input lines (stored in GPU memory) into view-oriented triangle strips is done each rendering pass in a geometry shader. The width of these triangle strips is (pre-)calculated by multiplying a global scaling factor with the maximum the of line style widths. The width of a line style is calculated through a summation of the maximum widths of its bands. With the line strips in place as the 'canvas' for the line style, the next step is to apply the style model.

The actual application of the line style is done in a fragment shader. The main goal of this fragment shader is to decide which band of which line style should be applied to the fragment. To determine this it uses the position on the strip, the shape mapping functions, and the values of the relevant line attributes. Then, based on the settings for that band, the color (either from a color map or from a color pattern) and the depth offset of the fragment can be determined.

One additional aspect of our implementation is the use of templated shaders. The main reason for this is that the flexibility of our extended line style model yields a large number of options, which without templated shaders would result in a large number of expensive conditional statements in the shader. The templated shaders (implemented using the existing templating library Jinja2[1] allow us to flexibly include only the necessary shader code, based on the chosen style configurations. This approach has the additional benefit of making the shader used for rendering as small as possible.

5 Results

To illustrate the broad range of possible visual representations of lines that can be achieved with our line style model, we apply a number of different line styles to two sets of streamlines. The first set is generated from a snapshot of a numerical simulation of a heat driven cavity (Dataset 1), the other set is generated from a snapshot of a simulation of turbulent flow around a cube (Dataset 2). It is important to note that although the streamlines that we visualize here

[1] http://jinja.pocoo.org/docs/

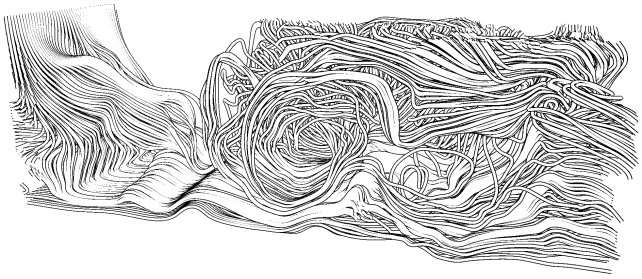

Figure 3: A simple black-and-white line style applied to streamlines from Dataset 1.

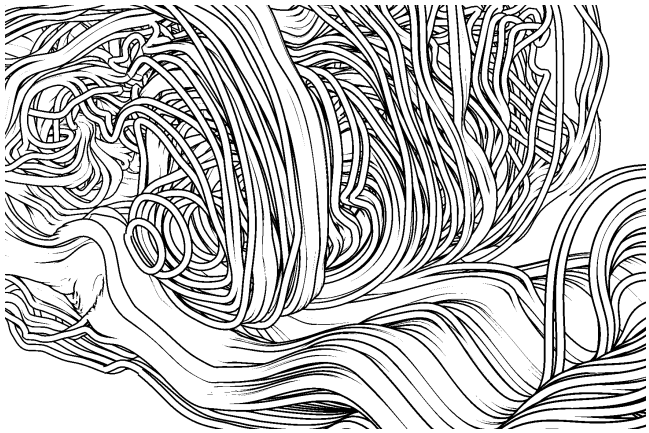

Figure 4: Close-up of streamlines with a simple black-and-white line style applied to them. Notice how the depth-dependent halo (contour) emphasizes collinear streamlines.

may give the impression of a steady flow, they are merely a visualization of the flow field at one particular time step.

We start with the application of a single simple black-and-white line style to Dataset 1 (see Fig. 3). This line style has two bands. The center-most band is white and fairly wide, whereas the outer band is thin and black. In addition, this outer band acts as a depth-dependent halo, although in this case it can also be considered a depth-dependent contour. The first thing to notice in Fig. 3 is how, despite the fact that no color has been used, the spatial relationships of the lines are still clear. Also, the depth manipulation ensures that collinear streamlines (e. g., the laminar flow at the bottom) blend together, emphasizing such collinear structures and yielding a crisper visualization. The close-up of the same dataset in Fig. 4 illustrates this aspect further.

The next step is to employ the visual parameters that our line style model introduces to convey additional information about the flow. Fig. 5 shows the application of a color map to streamlines, using a blue-purple color map to display velocity in Dataset 2. Again, the halo allows us to omit shading and still have good depth perception, making direct application of color maps possible without a potential shading that affects the perception of the colors.

Besides color maps, the other way our line style model

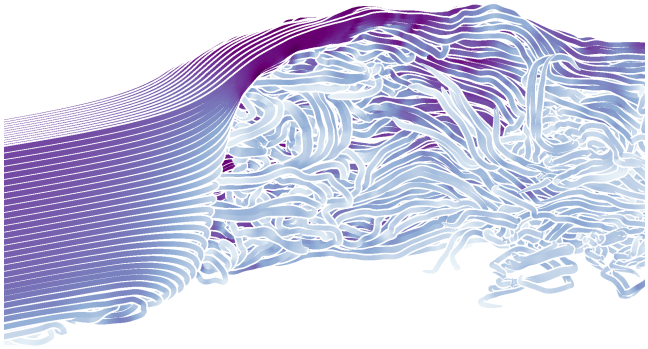

Figure 5: Streamlines depicting flow around a cube, colored with a blue-purple color map to show velocity, combined with white halos for depth perception.

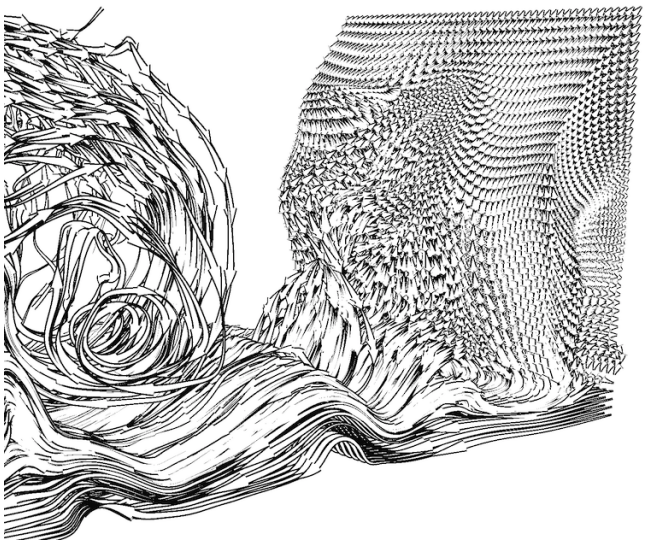

Figure 6: An arrow shape mapping function applied with a simple black-and-white style. The size of the arrow indicates velocity.

can convey additional information is through the size and frequency of shape patterns. Fig. 6 illustrates how an arrow shape can be used to convey both direction and velocity in a black-and-white visualization. Shape and color maps can also be combined, as illustrated in Fig. 7, where light gray arrows are combined with a fairly wide halo to which a color map is applied. Again the size (length) of a pattern indicates the local velocity of the flow. Besides an indication of direction, the arrow shape also gives the visualization a certain feel of motion. A similar effect is achieved with the tadpole shape shown in Fig. 8 where also a color-mapped halo is used, but with a constant shape length.

6 Discussion

As illustrated by the results in the previous section, our parametrization of line styles allows for a wide variety of

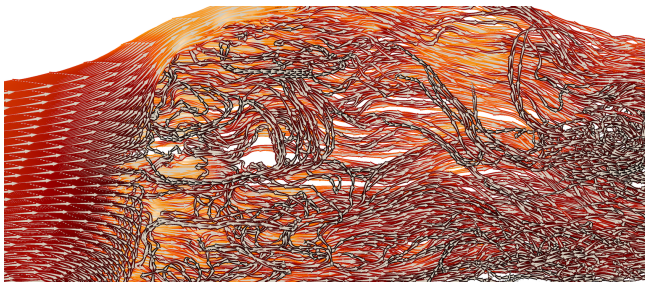

Figure 7: Streamlines depicted through light gray arrow shapes combined with a halos colored with a color map to depict velocity.

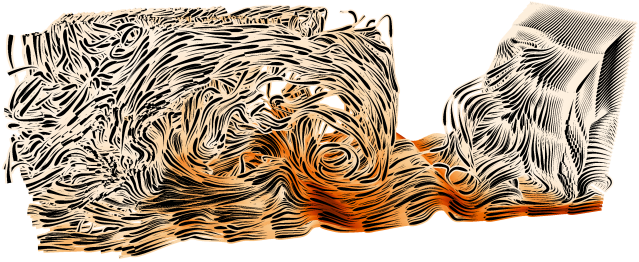

Figure 8: Streamlines depicted with tadpole-shaped, fixed size patterns, combined with a 'halo' colored using a colormap (velocity).

visual representations of lines, accompanied by visual variables to show additional information about the flow. In this section we discuss further aspects, observations, and limitations of our line style model.

In terms of performance, we find that on a fairly modern graphics card (NVIDIA GeForce GTX 285), we can interactively manipulate the line style parameters whilst displaying fairly large datasets, facilitating the interactive exploration of different visual representations of lines. For reference, the two datasets in Section 5 consist of 2500 streamlines (2.5M vertices) and 390 streamlines (250k vertices), respectively.

An additional observation is that in our visualizations where the length of a (shape) pattern depends on the local velocity, the patterns are longer in high velocity areas. Whether this effect is intuitive seems to depend on the people who are asked and the kind of shape being used, as some people correlate high (pattern) frequency with velocity. A related observation is that when the difference in velocity is large, the shape might become too small in low velocity areas, see for example the right side of Fig. 6. Other rendering artefacts are possible, for example when (shaped) line strips overlap in a certain way, resulting in oddly shaped patterns. Also, occasionally there are small artefacts when the view-vector is parallel to the line direction, though the effect is minimal and methods exist to remedy this artifact [22].

Finally, we presented our visualization results to a fluid mechanics expert in an informal discussion. In his initial reaction he commented on the "prettyness" of the images and found the visualizations very suitable for illustration purposes (e. g., classroom usage) because they illustrate well-known phenomena very well. Interestingly though, he liked the simple black-and-white visualizations (such as Fig. 3) best, mainly because of their simplicity and expressive power.

7 Conclusion

We have presented a flexible illustrative line style model for the visualization of streamline datasets. By partitioning line strips into parallel bands whose basic visual properties can be independently controlled, we create a parametrization that allows us to represent a broad range of visual styles for line data visualization. This approach is combined with line attribute mapping functions for color and width to facilitate flexible line shapes and means to convey additional information about the flow.

Future work includes combining our exploration of line styles with interactive streamline seeding strategies to further improve the exploration of flow datasets for visualization and illustration.

8 Acknowledgements

We thank Roel Verstappen and F. Xavier Trias Miquel for the datasets as well as their discussion and helpful feedback.

References

[1] A. Appel, F. J. Rohlf, and A. J. Stein. The Haloed Line Effect for Hidden Line Elimination. *ACM SIGGRAPH Computer Graphics*, 13(3):151–157, Aug. 1979. doi> 10.1145/800249.807437

[2] D. L. Dooley and M. F. Cohen. Automatic Illustration of 3D Geometric Models: Lines. In *Proc. I3D*, pp. 77–82, New York, 1990. ACM. doi> 10.1145/91385.91422

[3] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg. Depth-Dependent Halos: Illustrative Rendering of Dense Line Data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299–1306, Nov./Dec. 2009. doi> 10.1109/TVCG.2009.138

[4] A. Finkelstein and D. H. Salesin. Multiresolution Curves. In A. Glassner, editor, *Proceedings of ACM SIGGRAPH 94 (Orlando, FL, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 261–268, New York, 1994. ACM Press. doi> 10.1145/192161.192223

[5] A. Fuhrmann and E. Gröller. Real-time Techniques for 3D Flow Visualization. In *Proceedings of the conference on Visualization '98*, VIS '98, pp. 305–312, Los Alamitos, 1998. IEEE Computer Society Press. doi> 10.1109/VISUAL.1998.745317

[6] S. Grabli, E. Turquin, F. Durand, and F. X. Sillion. Programmable Rendering of Line Drawing from 3D Scenes. *ACM Transactions on Graphics*, 29:18:1–18:20, Apr. 2010. doi> 10.1145/1731047.1731056

[7] S. C. Hsu and I. H. H. Lee. Drawing and Animation Using Skeletal Strokes. In *Proc. SIGGRAPH*, pp. 109–118, New York, 1994. ACM. doi> 10.1145/192161.192186

[8] S. C. Hsu, I. H. H. Lee, and N. E. Wiseman. Skeletal Strokes. In *Proc. UIST*, pp. 197–206, New York, 1993. ACM. doi> 10.1145/16894.168662

[9] T. Isenberg and A. Brennecke. G-Strokes: A Concept for Simplifying Line Stylization. *Computers & Graphics*, 30(5):754–766, Oct. 2006. doi> 10.1016/j.cag.2006.07.006

[10] R. D. Kalnins, P. L. Davidson, L. Markosian, and A. Finkelstein. Coherent Stylized Silhouettes. *ACM Transactions on Graphics*, 22(3):856–861, July 2003. doi> 10.1145/882262.882355

[11] R. D. Kalnins, L. Markosian, B. J. Meier, M. A. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes, and A. Finkelstein. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics*, 21(3):755–762, July 2002. doi> 10.1145/566654.566648

[12] R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing Multivalued Data from 2D Incompressible Flows Using Concepts from Painting. In *Proc. IEEE Visualization*, pp. 333–340, Los Alamitos, 1999. IEEE Computer Society. doi> 10.1109/VISUAL.1999.809905

[13] R. S. Laramee and H. Hauser. Geometric Flow Visualization Techniques for CFD Simulation Data. In *Proc. SCCG*, pp. 221–224, New York, 2005. ACM. doi> 10.1145/1090122.1090158

[14] L. Li, H.-H. Hsieh, and H.-W. Shen. Illustrative Streamline Placement and Visualization. In *Proc. PacificVIS*, pp. 79–86, 2008. doi> 10.1109/PACIFICVIS.2008.4475462

[15] L. Li and H.-W. Shen. Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13:630–640, May/June 2007. doi> 10.1109/TVCG.2007.1009

[16] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. *Computer Graphics Forum*, 29(6):1807–1829, Sept. 2010. doi> 10.1111/j.1467-8659.2010.01650.x

[17] J. D. Northrup and L. Markosian. Artistic Silhouettes: A Hybrid Approach. In *Proc. NPAR*, pp. 31–37, New York, 2000. ACM. doi> 10.1145/340916.340920

[18] P. Rautek, S. Bruckner, E. Gröller, and I. Viola. Illustrative Visualization: New Technology or Useless Tautology? *ACM SIGGRAPH Computer Graphics*, 42(3):4:1–4:8, Aug. 2008. doi> 10.1145/1408626.1408633

[19] S. Schlechtweg, B. Schönwälder, L. Schumann, and T. Strothotte. Surfaces to Lines: Rendering Rich Line Drawings. In *Proc. WSCG*, volume 2, pp. 354–361, 1998.

[20] G. Schussman and K.-L. Ma. Scalable Self-Orienting Surfaces: A Compact, Texture-Enhanced Representation for Interactive Visualization of 3D Vector Fields. In *Proc. Pacific Graphics*, pp. 356–365, Los Alamitos, 2002. IEEE Computer Society. doi> 10.1109/PCCGA.2002.1167879

[21] H.-W. Shen, U. Bordoloi, and G.-S. Li. Interactive Visualization of Three-Dimensional Vector Fields With Flexible Appearance Control. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):434–445, July/Aug. 2004. doi> 10.1109/TVCG.2004.13

[22] C. Stoll, S. Gumhold, and H.-P. Seidel. Visualization with Stylized Line Primitives. In *Proc. IEEE Visualization*, pp. 695–702, Los Alamitos, 2005. IEEE Computer Society. doi> 10.1109/VIS.2005.124

[23] N. A. Svakhine, Y. Jang, D. S. Ebert, and K. Gaither. Illustration and Photography Inspired Visualization of Flows and Volumes. In *Proc. IEEE Visualization*, pp. 687–694, Los Alamitos, 2005. IEEE Computer Society. doi> 10.1109/VIS.2005.53

[24] M. Tarini, P. Cignoni, and C. Montani. Ambient Occlusion and Edge Cueing to Enhance Real Time Molecular Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–884, Sept./Oct. 2006. doi> 10.1109/TVCG.2006.115

[25] S.-K. Ueng, C. Sikorski, and K.-L. Ma. Efficient Streamline, Streamribbon, and Streamtube Constructions on Unstructured Grids. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):100–110, June 1996. doi> 10.1109/2945.506222

[26] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive Visualization of 3D-Vector Fields Using Illuminated Stream Lines. In *Proc. VIS*, pp. 107–113, Los Alamitos, 1996. IEEE Computer Society. doi> 10.1109/VISUAL.1996.567777