Visualizing 3D Flow

Victoria Interrante
Institute for Computer Applications in Science and Engineering

Chester Grosch
Old Dominion University

We discuss volume line integral convolution (LIC) techniques for effectively visualizing 3D flow, including using visibility-impeding halos and efficient asymmetric filter kernels.

An elegant and versatile technique, line integral convolution (LIC)[1,2] represents directional information via patterns of correlation in a texture. Although most commonly used to depict 2D flow, or flow over a surface in 3D, LIC methods can also portray 3D flow through a volume.[1,3] However, the popularity of LIC as a device for illustrating 3D flow has historically been limited by the computational expense of generating and rendering such a 3D texture and by the difficulties inherent in clearly and effectively conveying the directional information embodied in the resulting volumetric output textures.

Here we discuss some factors that may underlie some of the perceptual difficulties encountered with dense 3D displays and describe strategies for more effectively visualizing 3D flow with volume LIC. Specifically, we suggest techniques for

- selectively emphasizing critical regions of interest in a flow;
- facilitating the accurate perception of the 3D depth and orientation of overlapping streamlines;
- efficiently incorporating an indication of orientation into a flow representation; and
- conveying additional information about related scalar quantities such as temperature or vorticity over a flow via subtle, continuous line width and color variations.

Volume LIC

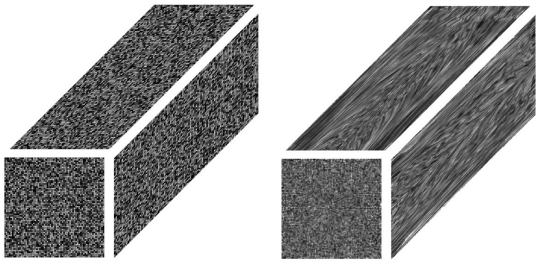
Applying LIC to a solid noise texture using a 3D vector field results in a solid 3D output texture—as shown in Figure 1—in which the values of the voxels are everywhere locally correlated in the direction of the 3D flow. The mechanics of the computation are straightforward, but how can you effectively visualize such data? It can be difficult to mentally reconstruct an accurate perception of the 3D flow from any series of 2D slices viewed sequentially. Defining an appropriate set of surfaces across which the 3D flow information can meaningfully be shown can prove problematic and imaging the data as a set of partial opacity values via direct-volume rendering sacrifices the inner details of the 3D texture. We believe that if appropriately defined and rendered, a 3D LIC texture has considerable potential to provide a full, immediate, and intuitive impression of a 3D flow's global and local characteristics. The challenge is to determine how to achieve such a representation. Shen et al.[3] suggested complementing a volume-rendered LIC texture with simulated dye advection. Here we look at some other options.

Region of interest specification

In certain cases, you can use a scalar function such as temperature or vorticity to identify, a priori, specific critical regions in a flow volume within which (or across which) the depiction of flow information is especially important. When we used LIC in conjunction with a region of interest (ROI) thus defined, we were able to achieve substantially better results by applying the ROI mask as a preprocess to the input texture before LIC rather than as a postprocess to the output afterwards.

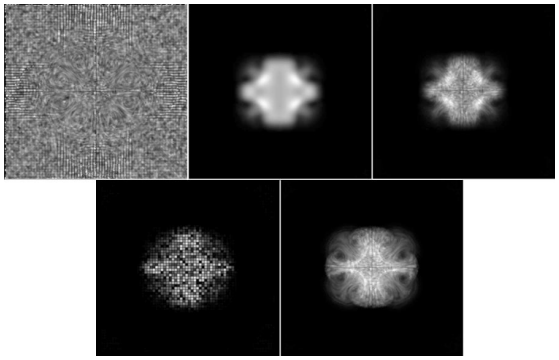
In the first case, the scalar ROI masking function completely determines the flow texture's visible portion. Here, the ROI's boundary generally does not follow the direction of the flow. In the second case, where the ROI function is applied before LIC, the flow guides the apparent ROI segmentation. When this occurs, the effective boundaries of the ROI will be everywhere aligned with the direction of the flow.

Figures 2 and 3 (next page) illustrate the difference between these two approaches. These images represent a numerical simulation of the effect of tabs on jets. The goal of the flow research was to investigate how jet engine noise might be reduced by adding tabs into the

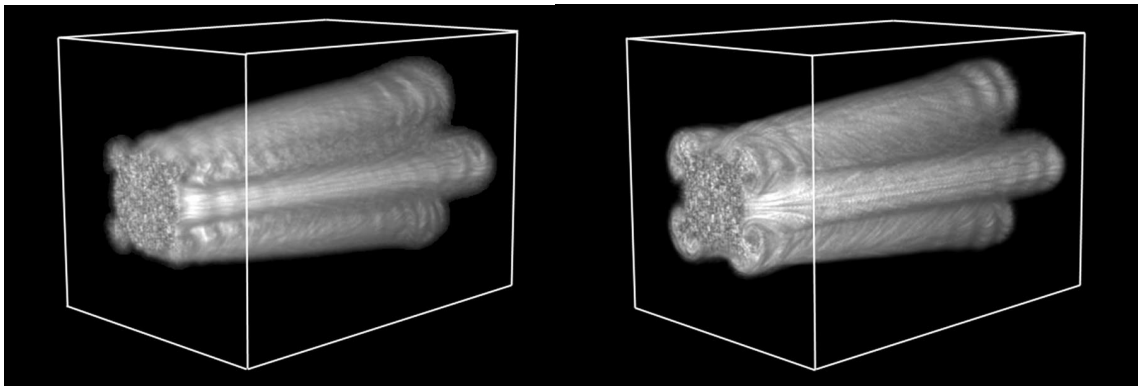


1 **A solid 3D texture of random noise (left). The solid texture after 3D LIC (right).**

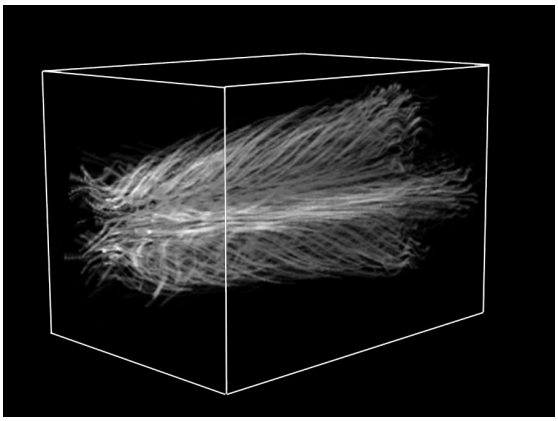
0272-1716/98/$10.00 © 1998 IEEE



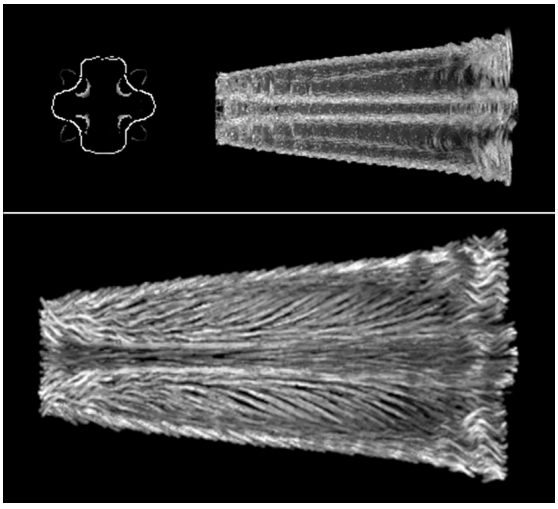
2 A 2D slice from a solid 3D texture generated from a 3D vector field and a solid noise input using volume LIC (top left). The corresponding slice through an ROI mask defined as a function of velocity magnitude (top center). The masked LIC texture (top right). A masked input texture for LIC (bottom left). The result after applying LIC to the masked input (bottom right).



3 A 3D view of the volume-rendered solid textures. Results when the ROI segmentation is applied as a postprocess after LIC (left). Results when ROI masking is applied to the input texture before performing LIC (right).



4 A volume-rendered 3D LIC texture that resembles a collection of scan-converted streamlines. The data was volume shaded according to the direction of the flow, but the relative spatial orientations of the individual lines remain difficult to see.



5 Using LIC to show 3D flow in the vicinity of a surface of interest. Ridges of velocity magnitude define a thin surface between the faster and slower moving regions of the flow (top). The result when we apply 3D LIC with a short filter kernel to an input texture consisting of evenly distributed points among the voxels intersected by this surface (bottom).

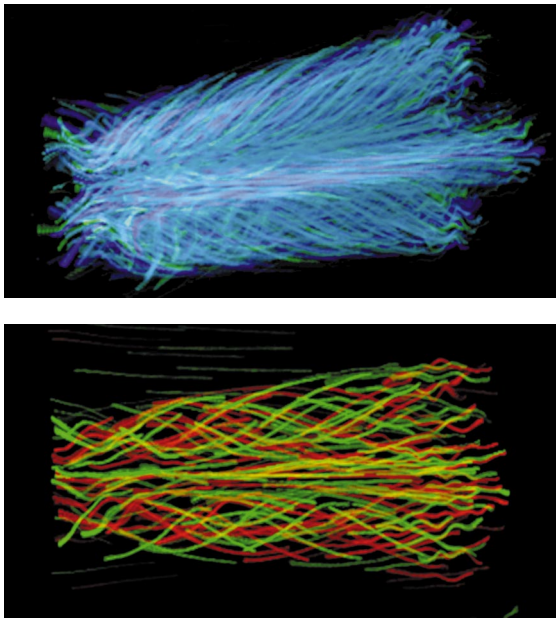
jets to generate vortices that could facilitate the mixing of the hot, supersonic flow with the colder, subsonic coflow. The flow data was generated on a nearly rectangular grid, of resolution $101 \times 101 \times 148$. The resolution of the input and output textures was made twice as large ($202 \times 202 \times 296$) so that the flow's details could be easily seen. The difference between the methods is especially apparent in the vicinity of the four pairs of counter-rotating vortices that were induced by the tabs.

Sparse input textures

When you apply LIC to a solid noise texture, even one that has been masked by an ROI function, the resulting 3D output texture is difficult to visualize as anything other than a solid object with fuzzy boundaries. By applying LIC over an input texture consisting of a sparse set of distributed points[4]—taking care to advect the "empty" space along with the full space—you can produce a solid texture like the kind shown in Figure 4, which represents a scan-converted collection of evenly placed, densely clustered streamlines. We premultiplied the input

texture by the same ROI mask used in Figures 2 and 3 to generate the volume shown in Figure 4. This constrained the streamlines to originate in the more rapidly moving regions of the flow, but allowed them to extend anywhere.

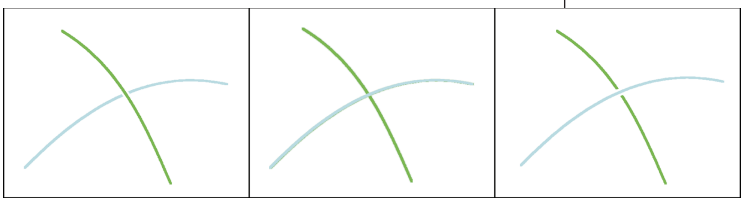
You can also use LIC in conjunction with a sparse input texture to visualize 3D flow in the vicinity of a surface of interest. In Figure 5, we applied a ridge strength function over the velocity magnitude volume shown earlier to define a "boundary surface" between the regions of relatively faster and slower flow. This approach facilitates coherent data segmentation according to the relationships between the values, while allowing the precise definition of "fast" and "slow" to remain fluid. Eliminating the need to project the flow directions onto the surface decreases the likelihood of generating a misleading impression of the flow, a problem addressed by Max et al.[5] in some detail. Because all calculations are performed in 3D, the tufts in the output texture will accurately reflect the flow's local 3D orientation in the immediate vicinity of the surface of interest.



6 Color variations alone are not sufficient to clarify the depiction of overlapping streamlines.

Clarifying depth relations

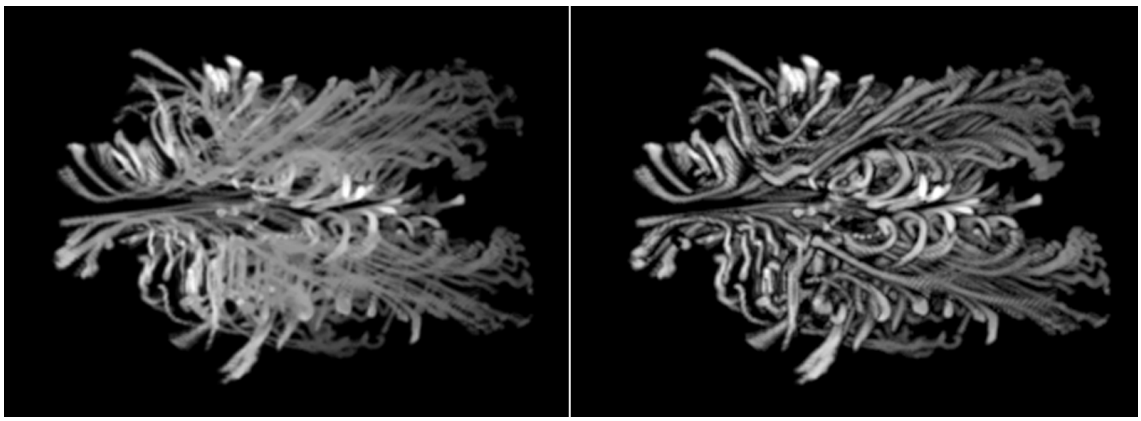
At first glance, the problem of effectively conveying the 3D shape and relative depth relations among the similarly directed, densely clustered streamlines traced by LIC appears to be a problem of simple differentiation: Like-colored lines, existing at different depths but projected onto adjacent pixels in a particular view, appear to coalesce into an indistinguishable clump. Phrased in this fashion, the problem points to a seemingly obvious solution: differentiate the individual lines by rendering them in different colors. Unfortunately, as you can see in Figure 6, merely introducing color variations does not greatly improve the clarity of the depth order relations among the overlapping lines. Figure 7 shows why we should expect this to be the case and suggests a better solution.



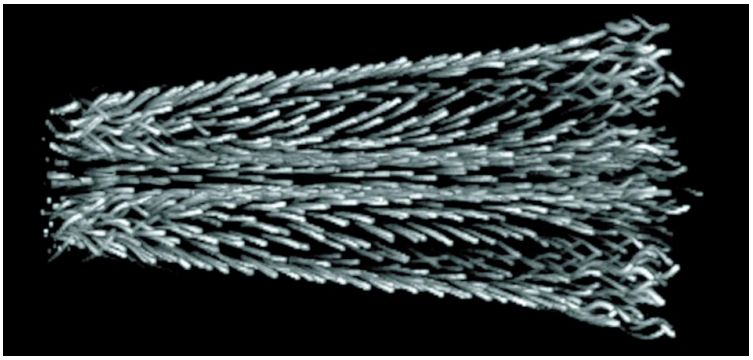
7 Two overlapping lines of roughly equal luminance but different hues. Depth order relations are explicitly emphasized in the leftmost and rightmost images by introducing subtle gaps that flank the foremost line.

In ordinary binocular vision, depth discontinuities generally coincide with the presence of interocularly unpaired regions in the views from each eye.[6] Artists and illustrators have long exploited this correspondence by using gaps to indicate the passing of one object behind another. In 1979, Appel et al.[7] proposed one of the first "haloed" line drawing algorithms for computer graphics.

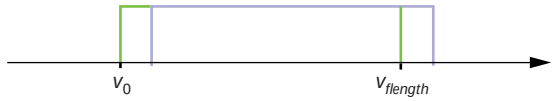
A simple modification to the basic LIC algorithm lets you efficiently compute a matched pair of textures that can be used to generate images like the one in Figure 8, in which the depth discontinuities are highlighted by gaps. We automatically define a subtle and smoothly continuous 3D visibility-impeding "halo" region that fully encloses each streamline in the original 3D texture. To do this, we perform LIC simultaneously over two input textures containing identically located spots of concentric sizes. Because the streamline tracing need only be done once for the pair of volumes, the overhead associated with creating the halos is kept to a minimum. Halos are implemented during raycasting-volume rendering



8 A side-by-side comparison illustrating the impact of visibility-impeding halos in conveying depth discontinuity information and facilitating appreciation of the depth extent in the flow.

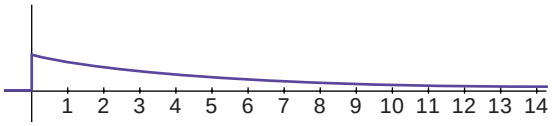


9 Tapered streamlines convey the forward and backward direction of the 3D flow. This depiction of orientation information was inspired by the oriented LIC (OLIC) method previously introduced by Wegenkittl et al.[8]



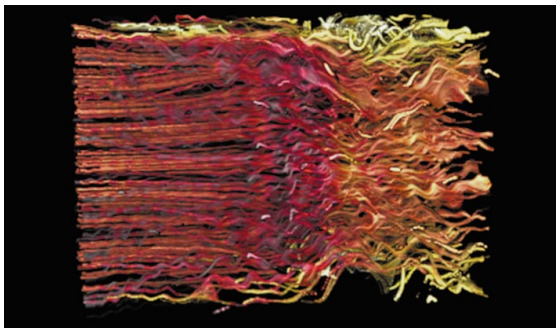
$$I_0' = \sum_{i=0}^{flength} v_i ; \quad I_1' = \sum_{i=1}^{flength+1} v_i ; \quad I_1' = I_0' - v_0 + v_{flength+1}$$

10 Box filter for the fast-LIC method.



$$I_0' = \sum_{i=0}^{flength} v_i \; \mathbf{c}^i, \mathbf{c} < 1$$

11 Exponential filter for the fast-OLIC method.



12 Two different color wash images generated from the same flow data. On the top, color varies from red to yellow with increasing temperature, showing the effect of friction; on the bottom, saturation increases from grey to purple according to the magnitude of the streamwise vorticity, highlighting turbulence.

by decreasing the contribution to the final image of any voxel encountered after a halo has been entered and subsequently exited by an amount proportional to the largest previously encountered halo opacity.

Specifically, the volume renderer takes as input two 3D textures (streamlines and halos) and traces rays through each. As tracing proceeds simultaneously through each volume, color and opacity values accumulate along the rays through the volume of streamlines until the renderer notes the first exit from a halo region. At this point, the accumulated opacity along the ray is increased by an amount proportional to the maximum density encountered in the previously traversed halo region, and tracing continues. Because each line will be everywhere surrounded by its own halo, it is important to let the voxels lying between the entrance and exit points of the first-encountered halo be rendered in the normal fashion. Note that this particular implementation assumes a black background and will not indicate the existence of depth discontinuities between lines whose halos overlap in 3-space, even if the lines themselves do not intersect.

Indicating directional information

Wegenkittl et al.[8] recently showed how basic LIC could be used in conjunction with an asymmetric, triangular-shaped filter kernel and a sparse 2D input texture to produce images in which information about the flow's forward and backward direction is locally conveyed through variation in the intensity of the rendered streamlets. The compelling images produced by this method inspired us to develop a simple modification to Stalling and Hege's fast-LIC algorithm that would let us efficiently compute 3D LIC textures that conveyed directional information in a similar manner (see Figure 9). The fast-LIC method, which we're using for 3D texture generation, gains significant efficiency through incremental calculations to compute the integrated intensity along a streamline. However, as formulated, the fast-LIC approach requires a box filter (see Figure 10).

We quickly realized that we could achieve both the oriented effect of OLIC and the computational advantage of fast-LIC by using an asymmetric filter derived from an exponential function of base less than one (see Figure 11).

In Figures 10 and 11, *flength* represents the length of the filter kernel (or the number of weighted samples in the forward direction along the streamline from each point that combine to determine the value at the corresponding point in the output texture); v_i represents the value of the input texture that lies under the ith sample point; and **c** is a constant that controls the rate of decrease in the contribution of the samples that lie

farther along the streamline. With this filter definition, you can continue to use incremental calculations to derive the integrated output texture values at subsequent voxels along a streamline:

$$I'_1 = (I'_0 - v_0) / \mathbf{c} + v_{flength+1}\mathbf{c}^{flength}$$

$$I'_{-1} = (I'_0 - v_{flength}\mathbf{c}^{flength}) \mathbf{c} + v_{-1}$$

We wrote the equations to reflect the fact that we successively shift new values into an array v of fixed bounds, so that at each step the current origin remains denoted v_0 and the intensity of the next point remains denoted I_{+1} or I_{-1}, depending on the direction of travel along the streamline.

Conveying additional information

You can use several devices to display additional scalar variables over a 3D flow represented by volume LIC. As previously demonstrated,[4] you can vary line width to reflect values in an accompanying scalar field. Color, however, remains the most powerful means to nonintrusively convey additional information over a flow. The volume-rendering approach makes it easy to vary the color definition across any predefined 3D texture. Figure 12 shows two different color washes over a flow through a rectangular aperture.

Future work

We demonstrated several new strategies for illustrating 3D flow with volume LIC, including visibility-impeding halos to emphasize the discontinuity in depth between overlapping lines and an asymmetric filter kernel in combination with fast-LIC for efficiently computing 3D flow textures that reveal directional information. We're continuing to investigate methods to more efficiently generate smooth, cyclic animations of 3D LIC textures along streamlines in steady flow data. We're also actively working on an extension to our 3D LIC algorithm to visualize unsteady flow data. ■

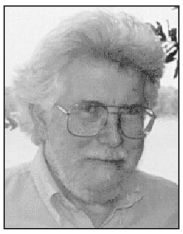
Acknowledgments

This work was supported by ICASE under NASA contract NAS1-19480. We're grateful to Marc Levoy for providing the original volume rendering platform that we built upon, and to Kwan-Liu Ma, Tom Crockett, David Banks, Hans-Christian Hege, and the anonymous reviewers for insightful comments and suggestions for improving this work.

References

1. B. Cabral and C. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Computer Graphics* (Proc. Siggraph 93), ACM Press, New York, 1993, pp. 263-269.
2. D. Stalling and H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution," *Computer Graphics* (Proc. Siggraph 95), ACM Press, New York, 1995, pp. 249-256.
3. H.-W. Shen, C.R. Johnson, and K.-L. Ma, "Visualizing Vector Fields Using Line Integral Convolution and Dye Advection," *1996 Symp. Volume Visualization*, ACM Press, New York, 1996, pp. 63-70.
4. V. Interrante, "Illustrating Surface Shape in Volume Data via Principal Direction-Driven 3D Line Integral Convolution," *Computer Graphics* (Proc. Siggraph 97), ACM Press, New York, 1997, pp. 109-116.
5. N. Max, R. Crawfis, and C. Grant, "Visualizing 3D Velocity Fields Near Contour Surfaces," *Proc. IEEE Visualization 94*, IEEE CS Press, Los Alamitos, Calif., pp. 248-255.
6. S. Shimojo and K. Nakayama. "Real-World Occlusion Constraints and Binocular Rivalry," *Vision Research*, Vol. 30, No. 1, 1990, pp. 69-80.
7. A. Appel, F.J. Rohlf, and A. Stein, "The Haloed Line Effect for Hidden Line Elimination," *Proc. Siggraph 79*, ACM Press, New York, pp. 151-157.
8. R. Wegenkittl, E. Gröller, and W. Purgathofer, "Animating Flowfields: Rendering of Oriented Line Integral Convolution," *Proc. Computer Animation 97*, IEEE CS Press, Los Alamitos, Calif., June 1997, pp. 15-21.

Victoria Interrante *is a staff scientist at the Institute for Computer Applications in Science and Engineering (ICASE), a center of research in applied mathematics, numerical analysis, and computer science operated by the Universities Space Research Association at the NASA Langley Research Center. Her current research focuses on the application of insights from perceptual psychology, art, and illustration to the design of more effective techniques for visualizing 3D data. She received a PhD in computer science from the University of North Carolina at Chapel Hill in 1996.*



Chester Grosch *holds joint appointments as professor of oceanography and professor of computer science at Old Dominion University in Norfolk, Virginia. He received a PhD in 1967 from Stevens Institute of Technology in Hoboken, New Jersey. While at the Pratt Institute, he was chair of the Department of Computer Science and director of the Computer Center. In 1989, he was a Royal Society guest research fellow at the University of Cambridge, UK. Grosch has done extensive professional consulting and currently serves as a consultant for ICASE.*

Contact Interrante at the Institute for Computer Applications in Science and Engineering, Mail Stop 403, NASA Langley Research Center, Hampton, VA 23681-0001, e-mail interran@icase.edu.