Imaging Vector Fields Using Line Integral Convolution

Brian Cabral
*Leith (Casey) Leedom**

Lawrence Livermore National Laboratory

ABSTRACT

Imaging vector fields has applications in science, art, image processing and special effects. An effective new approach is to use linear and curvilinear filtering techniques to locally blur textures along a vector field. This approach builds on several previous texture generation and filtering techniques[8, 9, 11, 14, 15, 17, 23]. It is, however, unique because it is local, one-dimensional and independent of any predefined geometry or texture. The technique is general and capable of imaging arbitrary two- and three-dimensional vector fields. The local one-dimensional nature of the algorithm lends itself to highly parallel and efficient implementations. Furthermore, the curvilinear filter is capable of rendering detail on very intricate vector fields. Combining this technique with other rendering and image processing techniques — like periodic motion filtering — results in richly informative and striking images. The technique can also produce novel special effects.

CR categories and subject descriptors: I.3.3 [Computer Graphics]: Picture/Image generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.4.3 [Image Processing]: Enhancement.

Keywords: convolution, filtering, rendering, visualization, texture synthesis, flow fields, special effects, periodic motion filtering.

1. INTRODUCTION

Upon first inspection, imaging vector fields appears to have limited application — confined primarily to scientific visualization. However, much of the form and shape in our environment is a function of not only image intensity and color, but also of directional information such as edges. Painters, sculptors, photographers, image processors[16] and computer graphics researchers[9] have recognized the importance of direction in the process of image creation and form. Hence, algorithms that can image such directional information have wide application across both scientific and artistic domains.

Such algorithms should possess a number of desirable and sometimes conflicting properties including: accuracy, locality of calculation, simplicity, controllability and generality. Line Integral Convolution (LIC) is a new technique that possesses many of these properties. Its generality allows for the introduction of a completely new family of periodic motion filters which have wide application (see section 4.1). It represents a confluence of signal and image processing and a variety of previous work done in computer graphics and scientific visualization.

* Authors' current e-mail addresses are: *cabral@llnl.gov* and *casey@gauss.llnl.gov.*

2. BACKGROUND

There are currently few techniques which image vector fields in a general manner. These techniques can be quite effective for visualizing vector data. However, they break down when operating on very dense fields and do not generalize to other applications. In particular, large vector fields (512x512 or greater) strain existing algorithms.

Most vector visualization algorithms use spatial resolution to represent the vector field. These include sampling the field, such as with stream lines[12] or particle traces, and using icons[19] at every vector field coordinate. Stream lines and particle tracing techniques depend critically on the placement of the "streamers" or the particle sources. Depending on their placement, eddies or currents in the data field can be missed. Icons, on the other hand, do not miss data, but use up a considerable amount of spatial resolution limiting their usefulness to small vector fields.

Another general approach is to generate textures via a vector field. Van Wijk's *spot noise* algorithm[23] uses a vector field to control the generation of bandlimited noise. The time complexity of the two types of implementation techniques presented by Van Wijk are relatively high. Furthermore the technique, by definition, depends heavily on the form of the texture (spot noise) itself. Specifically, it does not easily generalize to other forms of textures that might be better suited to a particular class of vector data (such as fluid flow versus electromagnetic).

Reaction diffusion techniques[20, 24] also provide an avenue for visualizing vector fields since the controlling differential equations are inherently vector in nature. It is possible to map vector data onto these differential equations to come up with a vector visualization technique. Here too however, the time complexity of these algorithms limit their general usefulness.

Three-dimensional vector fields can be visualized by three-dimensional texture generation techniques such as texels and hypertextures described in [11, 15]. Both techniques take a texture on a geometrically defined surface and project the texture out some distance from the surface. By definition these techniques are bound to the surface and do not compute an image for the entire field as is done by Van Wijk[23]. This is limiting in that it requires a priori knowledge to place the surface. Like particle streams and vector streamers these visualization techniques are critically dependent on the placement of the sampling surface.

The technique presented by Haeberli[9] for algorithmicly generating "paintings" via vector-like brush strokes can also be thought of as a vector visualization technique. Crawfis and Max[5]

describe a three-dimensional variation on this in which blurred cylinders represent three-dimensional brush strokes whose directions and colors are controlled by a three-dimensional vector field. Both techniques represent a conceptual extension of traditional icon placement, where the icons are more sophisticated shapes. However, these techniques break down as the density of the field increases since they require spatial resolution to work.

What is needed is a technique that can image dense vector fields, is independent of both predefined sampling placement constraints and texture generation techniques and can work in two and three dimensions. Such a technique would be very general and have wide application.

3. DDA CONVOLUTION

One approach is a generalization of traditional DDA line drawing techniques[1] and the spatial convolution algorithms described by Van Wijk[23] and Perlin[14]. Each vector in a field is used to define a long, narrow, DDA generated filter kernel tangential to the vector and going in the positive and negative vector direction some fixed distance, *L*. A texture is then mapped one-to-one onto the vector field. The input texture pixels under the filter kernel are summed, normalized by the length of the filter kernel, *2L*, and placed in an output pixel image for the vector position. Figure 1, illustrates this operation for a single vector in a field.

This effectively filters the underlying texture as a function of the vector field. The images in figure 2 are rendered using the DDA convolution algorithm. On the left is a simple circular vector field; to its right is the result of a computational fluid dynamics code. The input texture image in these examples is white noise. Although the description above implies a box filter, any arbitrary filter shape can be used for the filter convolution kernel. It is important to note that this algorithm is very sensitive to symmetry of the DDA algorithm and filter. If the algorithm weights the forward direction more than the backward direction, the circular field in figure 2 appears to spiral inward implying a vortical behavior that is not present in the vector field.

3.1 LOCAL FIELD BEHAVIOR

The DDA approach, while efficient, is inherently inaccurate. It assumes that the local vector field can be approximated by a straight line. For points in vector fields where the local radius of curvature is large, this assumption is valid. However, where there are complex structures smaller than the length of the DDA line, the local radius of curvature is small and is not well approximated by a straight line. In a sense, DDA convolution renders the vector field unevenly, treating linear portions of the vector field more accurately than small scale vortices. While this graceful degradation may be fine or even desirable for special effects applications, it is problematic for visualizing vector fields such as the ones in figure 2, since detail in the small scale structures is lost.

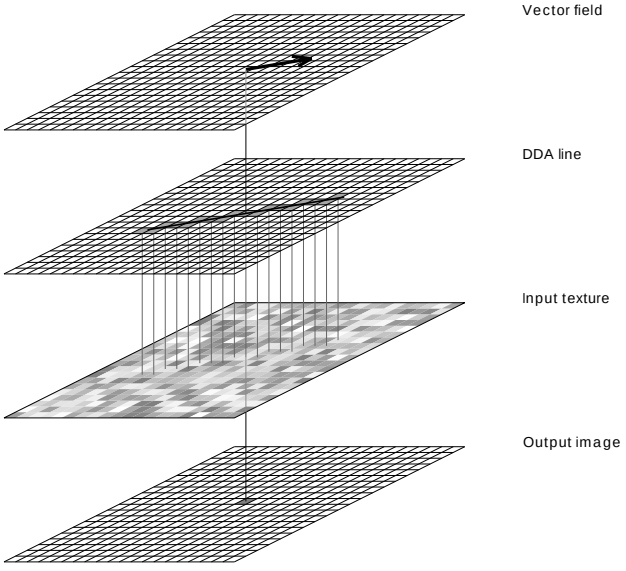

Figure 1: The mapping of a vector onto a DDA line and input pixel field generating a single output pixel.

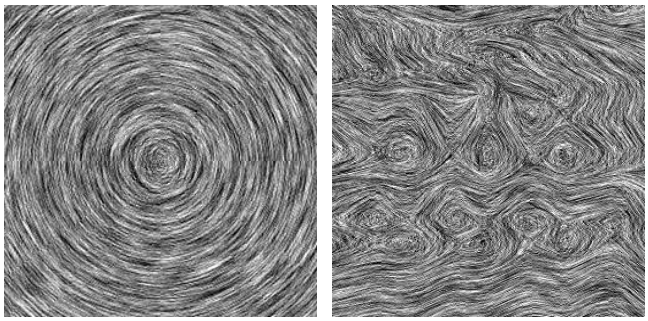

Figure 2: Circular and turbulent fluid dynamics vector fields imaged using DDA convolution over white noise.

Van Wijk's spot noise algorithm[23] also suffers from this problem since the spots are elliptically stretched along a line in the direction of the local field. If the ellipse major axis exceeds the local length scale of the vector field, the spot noise will inaccurately represent the vector field. An accurate measure of local field behavior would require a global analysis of the field. Such techniques currently do not exist for arbitrary vector fields, would most likely be expensive to calculate[13] and are an area of active research[7].

4. LINE INTEGRAL CONVOLUTION

The local behavior of the vector field can be approximated by computing a local stream line that starts at the center of pixel (*x*, *y*) and moves out in the positive and negative directions.[1] The forward coordinate advection is given by equation (1).

$$P_0 = (x + 0.5, y + 0.5)$$

$$P_i = P_{i-1} + \frac{V(\lfloor P_{i-1} \rfloor)}{\| V(\lfloor P_{i-1} \rfloor) \|} \Delta s_{i-1} \qquad (1)$$

$$V(\lfloor P \rfloor) = \text{the vector from the input vector field at lattice point } (\lfloor P_x \rfloor, \lfloor P_y \rfloor)$$

$$s_e = \begin{cases} \infty \text{ if } V \| e \\ 0 \text{ if } \dfrac{\lfloor P_c \rfloor - P_c}{V_c} < 0 \\ \dfrac{\lfloor P_c \rfloor - P_c}{V_c} \text{ otherwise} \end{cases} \quad \text{for } (e, c) \in \begin{Bmatrix} (top, y) \\ (bottom, y) \\ (left, x) \\ (right, x) \end{Bmatrix} \qquad (2)$$

$$\Delta s_i = \min (s_{top}, s_{bottom}, s_{left}, s_{right})$$

[1] Vector field lattice and image coordinates are usually specified in a left-handed coordinate system while vector components are usually specified in a right-handed coordinate system. In this case, the *y*-component of the lattice coordinate in equation (1) must be reflected about the vertical center of the lattice to operate in a consistent coordinate system. This reflection has been omitted to preserve simplicity of presentation.

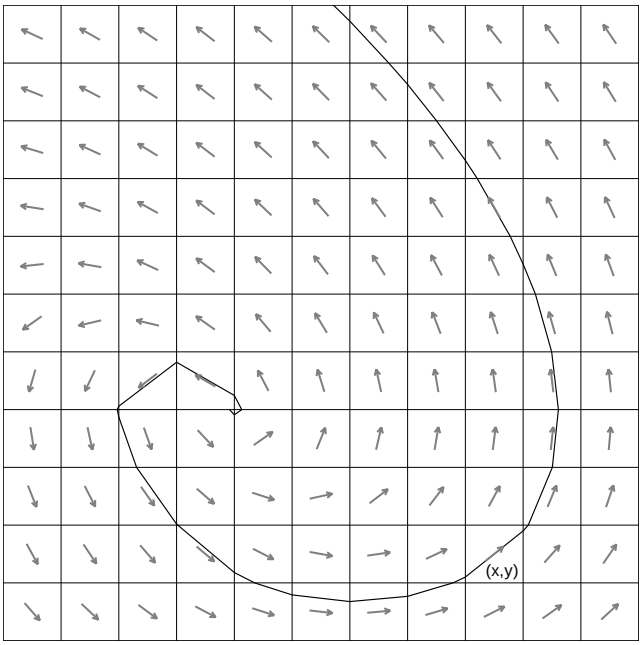

Figure 3: A two-dimensional vector field showing the local stream line starting in cell (*x*, *y*). The vector field is the upper left corner of the fluid dynamics field in figures 2 and 4.

Only the directional component of the vector field is used in this advection. The magnitude of the vector field can be used later in post processing steps as explained in section 4.3.1. Δs_i is the positive parametric distance along a line parallel to the vector field from P_i to the nearest cell edge.

As with the DDA algorithm, it is important to maintain symmetry about a cell. Hence, the local stream line is also advected backwards by the negative of the vector field as shown in equation (3).

$$P'_0 = P_0$$
$$P'_i = P'_{i-1} - \frac{V(\lfloor P'_{i-1} \rfloor)}{\| V(\lfloor P'_{i-1} \rfloor) \|} \Delta s'_{i-1} \quad (3)$$

Primed variables represent the negative direction counterparts to the positive direction variables and are not repeated in subsequent definitions. As above $\Delta s'_i$, is always positive.

The calculation of Δs_i in the stream line advection is sensitive to round off errors. Δs_i must produce advected coordinates that lie within the i+1th cell, taking the stream line segment out of the current cell. In the implementation of the algorithm a small round off term is added to each Δs_i to insure that entry into the adjacent cell occurs. This local stream line calculation is illustrated in figure 3. Each cell is assumed to be a unit square. All spatial quantities (e.g., Δs_i) are relative to this measurement. However, the cells need not be square or even rectangular (see section 6) for this approximation to work. So, without loss of generality, descriptions are given relative to a cubic lattice with unit spacing.

Continuous sections of the local stream line — i.e. the straight line segments in figure 3 — can be thought of as parameterized space curves in *s* and the input texture pixel mapped to a cell can be treated as a continuous scalar function of x and y.[2] It is then possible to integrate over this scalar field along each parameterized space curve. Such integrals can be summed in a piecewise $\mathbf{C^1}$ fashion and are known as line integrals of the first kind (LIFK)[2]. The convolution concept used in the DDA algorithm can now be combined with LIFK to form a Line Integral Convolution (LIC). This results in a variation of the DDA approach that locally follows the vector field and captures small radius of curvature features. For each continuous segment, *i*, an exact integral of a convolution kernel $k(w)$ is computed and used as a weight in the LIC as shown in equation (4).

$$h_i = \int_{s_i}^{s_i + \Delta s_i} k(w)\, dw \quad (4)$$

where
$$s_0 = 0$$
$$s_i = s_{i-1} + \Delta s_{i-1}$$

The entire LIC for output pixel *F'(x, y)* is given by equation (5).

$$F'(x,y) = \frac{\sum_{i=0}^{l} F(\lfloor P_i \rfloor)\, h_i + \sum_{i=0}^{l'} F(\lfloor P'_i \rfloor)\, h'_i}{\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h'_i} \quad (5)$$

where

$F(\lfloor P \rfloor)$ is the input pixel corresponding to the vector at position $(\lfloor P_x \rfloor, \lfloor P_y \rfloor)$

$$l = i \text{ such that } s_i \le L < s_{i+1} \quad (6)$$

The numerator of equation (5) represents the line integral of the filter kernel times the input pixel field, *F*. The denominator is the line integral of the convolution kernel and is used to normalize the output pixel weight (see section 4.2).

The length of the local stream line, *2L*, is given in unit pixels. Depending on the input pixel field, *F*, if *L* is too large, all the resulting LICs will return values very close together for all coordinates (*x*, *y*). On the other hand, if *L* is too small then an insufficient amount of filtering occurs. Since the value of *L* dramatically affects the performance of the algorithm, the smallest effective value is desired. For most of the figures, a value of 10 was used.

Singularities in the vector field occur when vectors in two adjacent local stream line cells geometrically "point" at a shared cell edge. This results in Δs_i values equal to zero leaving *l* in equation (6) undefined. This situation can easily be detected and the advection algorithm terminated. If the vector field goes to zero at any point, the LIC algorithm is terminated as in the case of a field singularity. Both of these cases generate truncated stream lines. If a zero field vector lies in the starting cell of the LIC, the input pixel value for that cell, a constant or any other arbitrary value can be returned as the value of the LIC depending on the visual effect desired for null vectors.

Using adjacent stream line vectors to detect singularities can however result in false singularities. False singularities occur when the vector field is nearly parallel to an edge, but causes the LIC to cross over that edge. Similarly, the cell just entered also has a near parallel vector which points to this same shared edge. This artifact can be remedied by adjusting the parallel vector/edge test found in equation (2), to test the angle formed between the vector and the edge against some small angle *theta*, instead of zero. Any vector which forms an angle less than *theta* with some edge is deemed to be "parallel" to that edge. Using a value of 3° for *theta* removes these artifacts.

The images in figure 4 were rendered using LIC and correspond to the same two vector fields rendered in figure 2. Note the increased amount of detail present in these images versus their DDA counterparts. In particular the image of the fluid dynamics vector field in figure 4 shows detail incorrectly rendered or absent in figure 2.

[2] Bilinear, cubic or Bezier splines are viable alternatives to straight line segments. However, these higher order curves are more expensive to compute.

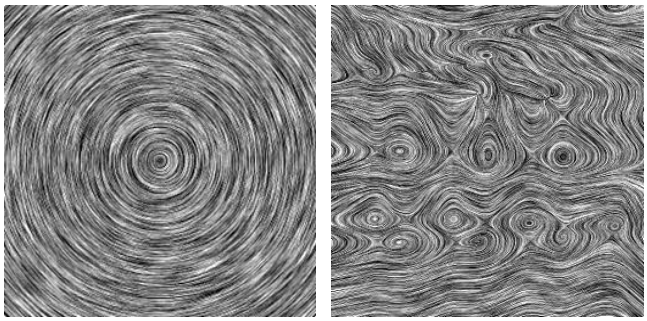

Figure 4: Circular and turbulent fluid dynamics vector fields imaged using LIC over white noise.

The images in figure 5 show the effect of varying L. The input texture is a photograph of flowers. The input vector field was created by taking the gradient of a bandlimited noise image and rotating each of the gradient vectors by 90°, producing vectors which follow the contours of the soft hills and valleys of the bandlimited noise. With L equal to 0, the input image is passed through unchanged. As the value of L increases, the input image is blurred to a greater extent, giving an impressionistic result. Here, a biased ramp filter[10] is used to roughly simulate a brush stroke.

Figures 2, 4, 8, 9 and 11 were generated using white noise input images. Aliasing can be a serious problem when using LIC with a high frequency source image such as white noise. The aliasing is caused by the one-dimensional point sampling of the infinitely thin LIC filter. This aliasing can be removed by either creating a thick LIC filter with a low-pass filter cross section or by low-pass filtering the input image. This second alternative is preferable since it comes at no additional cost to the LIC algorithm. The images in figure 6 show the effect of running LIC over 256x256 white noise which has been low-pass filtered using a fourth order Butterworth filter with cutoff frequencies of 128, 84, 64, and 32.

It is worth noting that Van Wijk's spot noise algorithm[23] can be adapted to use the local stream line approximation to more accurately represent the behavior of a vector field. Instead of straight line elliptical stretching, each spot could be warped so that the major axis follows the local stream line. Furthermore, the minor axis could either be perpendicular to the warped major axis or itself could be warped along transverse field lines. However, an algorithm to perform this task for an arbitrary local stream line would be inherently more expensive and complex than the LIC algorithm.

Sims[18] describes an alternative technique which produces results similar to LIC. This alternative approach warps or advects texture coordinates as a function of a vector field. The similarity between the two techniques is predictable even though the techniques are quite different. The dilation and contraction of the texture coordinate system warping has the visual effect of blurring and sharpening the warped image. This is due to the resampling and reconstruction process necessary when warping from one coordinate system to another. Thus, for regions where the source image is stretched along the vector field an apparent blurring will occur similar to those seen with LIC. However, the techniques are completely different in two fundamental ways. First, LIC is a local operator, meaning no information outside of a fixed area of interest is needed. Warping even when done locally requires maintaining global consistency to avoid tearing holes in the warped image. This increases the complexity of the warping operation when compared to LIC. Second, LIC is a spatially varying filtering operation and does not warp or transform any texture coordinates.

4.1 PERIODIC MOTION FILTERS

The LIC algorithm visualizes local vector field tangents, but not their direction. Freeman, et al[8] describe a technique which simulates motion by use of special convolutions. A similar technique is used by Van Gelder and Wilhelms[22] to show vector field flow. This technique can be extended and used to represent the local vector field direction via animation of successive LIC imaged vector fields using varying phase shifted periodic filter kernels.

The success of this technique depends on the shape of the filter. In the previous examples (figures 2 and 4), a constant or box filter is used. If the filter is periodic like the filters used in [8], by changing the phase of such filters as a function of time, apparent motion

Figure 5: Photograph of flowers processed using LIC with L equal to 0, 5, 10 and 20 (left to right, top to bottom).

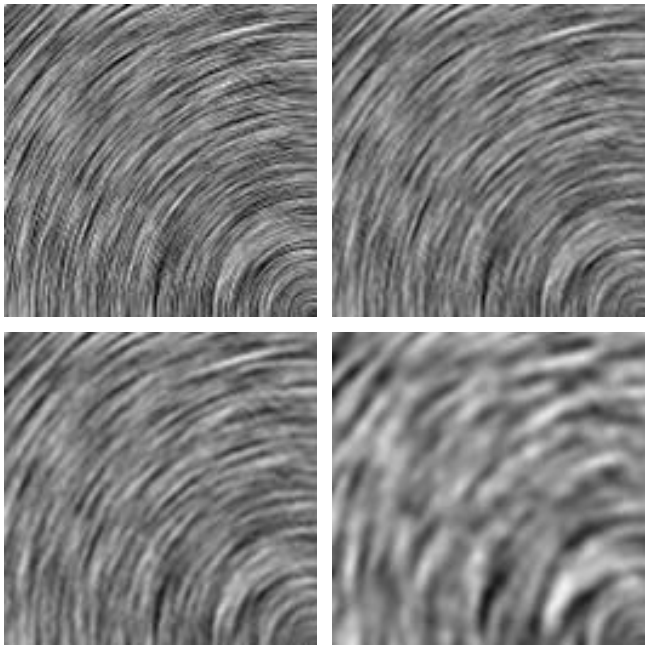

Figure 6: The upper left hand quarter of the circular vector field is convolved using LIC over Butterworth low-pass filtered white noise with cutoff frequencies of 128, 86, 64, and 32 (left to right, top to bottom).

in the direction of the vector field is created. However, the filters used in [8] were, by design, high-pass Laplacian edge enhancing filters. Using this filter over a bandlimited noise texture produces very incoherent images since the high frequency components of the noise are accentuated. Instead, it is possible, and desirable, to create periodic low-pass filters to blur the underlying texture in the direction of the vector field. A Hanning filter, $1/2(1 + \cos(w+\beta))$, has this property. It has low band-pass filter characteristics, it is periodic by definition and has a simple analytic form. This function will be referred to as the *ripple* filter function.

Since the LIC algorithm is by definition a local operation, any filter used must be windowed. That is, it must be made local even if it has infinite extent. In the previous section we used a constant filter implicitly windowed by a box of height one. Using this same box window on a phase shifted Hanning filter we get a filter with abrupt cutoffs, as illustrated in the top row of figure 7.

This abrupt cutoff is noticeable as spatio-temporal artifacts in animations that vary the phase as a function of time. One solution to this problem is to use a Gaussian window as suggested by Gabor[4].[3] By multiplying, or windowing, the Hanning function by a Gaussian, these cutoffs are smoothly attenuated to zero. However, a Gaussian windowed Hanning function does not have a simple closed form integral. An alternative is to find a windowing function with windowing properties similar to a Gaussian and which has a simple closed form integral. Interestingly, the Hanning function itself meets these two criteria. In the bottom row of figure 7, the five phase shifted Hanning filter functions in the top row are multiplied by the Hanning window function in the middle row. The general form of this function is shown in equation (7). In this equation c and d represent the dilation constants of the Hanning window and ripple functions respectively. β is the ripple function phase shift given in radians. The integral of $k(w)$ from a to b used in equation (4) is shown in equation (8).

$$k(w) = \frac{1+\cos(cw)}{2} \times \frac{1+\cos(dw+\beta)}{2} \tag{7}$$

$$= \frac{1}{4}\left(1+\cos(cw)+\cos(dw+\beta)+\cos(cw)\cos(dw+\beta)\right)$$

$$\int_a^b k(w)\,dw \tag{8}$$

$$= \frac{1}{4}\left(\begin{array}{c} b-a+\dfrac{\sin(bc)-\sin(ac)}{c} \\ +\dfrac{\sin(bd+\beta)-\sin(ad+\beta)}{d} \\ +\dfrac{\sin(b(c-d)-\beta)-\sin(a(c-d)-\beta)}{2(c-d)} \\ +\dfrac{\sin(b(c+d)+\beta)-\sin(a(c+d)+\beta)}{2(c+d)} \end{array} \right)$$

As mentioned above, both the Hanning window and the Hanning ripple filter function can be independently dilated by adjusting c and d to have specific local support and periodicity. The window function has a fixed period of 2π.

Choosing the periodicity of the ripple function represents making a design trade-off between maintaining a nearly constant frequency response as a function of phase shift and the quality of the apparent motion[3]. A low frequency ripple function results in a windowed filter whose frequency response noticeably changes as a function of phase. This appears as a periodic blurring and sharpening of the image as the phase changes. Higher frequency ripple functions produce windowed filters with a nearly constant frequency response since the general shape of the filter doesn't radically change. However, the feature size picked up by the ripple filter is smaller and the result is less apparent motion. If the ripple frequency exceeds the Nyquist limit of the pixel spacing the apparent motion disappears. Experimentation shows that a ripple function frequency between 2 and 4 cycles per window period is reasonable. One can always achieve both good frequency response and good feature motion by increasing the spatial resolution. This comes, of course, at a cost of increased computation[16].

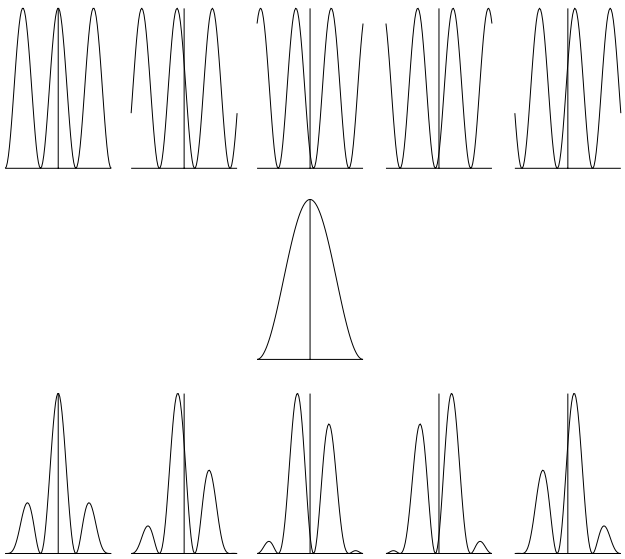

Figure 7: Phase shifted Hanning ripple functions(top), a Hanning windowing function(middle), and Hanning ripple functions multiplied by the Hanning window function(bottom).

4.2 NORMALIZATION

A normalization to the convolution integral is performed in equation (5) to insure that the apparent brightness and contrast of the resultant image is well behaved as a function of kernel shape, phase and length. The numerator in equation (5) is divided by the integral of the convolution kernel. This insures that the normalized area under the convolution kernel is always unity resulting in a constant overall brightness for the image independent of the filter shape and LIC length.

Because the actual length of the LIC may vary from pixel to pixel, the denominator can not be precomputed. However, an interesting effect is observed if a fixed normalization is used. Truncated stream lines are attenuated which highlights singularities. The images in figure 8 a show another section of the fluid dynamics vector field imaged with variable and constant kernel normalization. The implementation of the LIC algorithm uses precomputed sum tables for the integral to avoid costly arithmetic in the innermost loop.

A second normalization may be done to insure the output image retains the input image's contrast properties. The LIC algorithm reduces the overall image contrast as a function of L. In fact, in the case of the box filter, as L goes to infinity the entire output image goes to the average of the input image. This can be ameliorated by amplifying the input or contrast stretching the output image as a function of L. Clearly as L goes to infinity the amplification or con-

[3] D. Gabor in 1946 created a localized form of the Fourier transform known as the Gabor transform. This transform is the Fourier transform of an input signal multiplied by a Gaussian window translated along the signal as a function of time. The net result is a signal which is spatially and frequency localized. Wavelet theory is based on a generalization of this type of spatial and frequency localization.

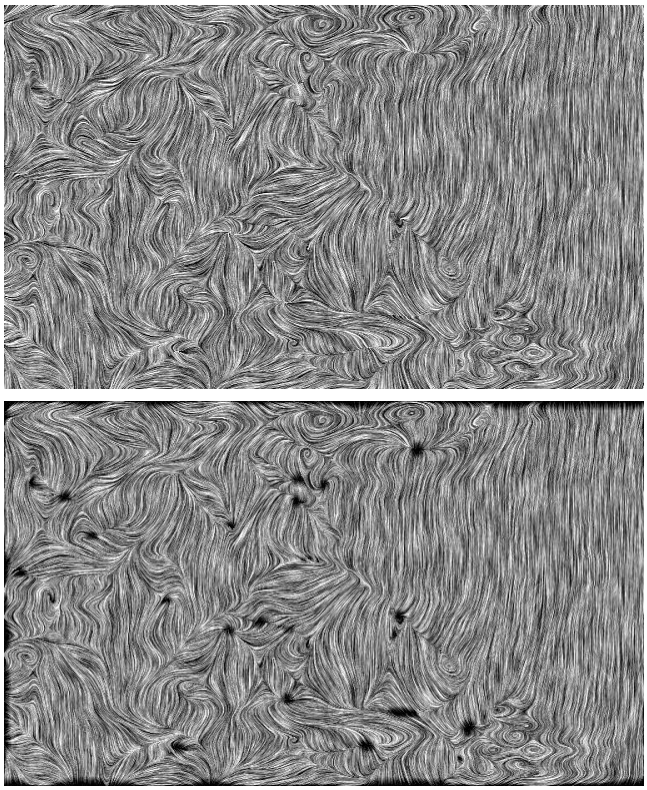

Figure 8: White noise convolved with fluid dynamics vector field using variable normalization (top) versus fixed normalization (bottom).

trast stretching must go to infinity as well. The images in all the figures are contrast stretched.

4.3 IMPLEMENTATION AND APPLICATION

The LIC algorithm is designed as a function which maps an input vector field and texture to a filtered version of the input texture. The dimension of the output texture is that of the vector field. If the input texture is smaller than the vector field the implementation of the algorithm wraps the texture using a toroidal topology. That is, the right and left edges wrap as do the top and bottom edges. If the texture is too large it is cropped to the vector field dimensions. Careful attention must be paid to the size of the input texture relative to that of the vector field. If too small a texture is used, the periodicity induced by the texture tiling will be visible. For scientific applications this is unacceptable. One must insure

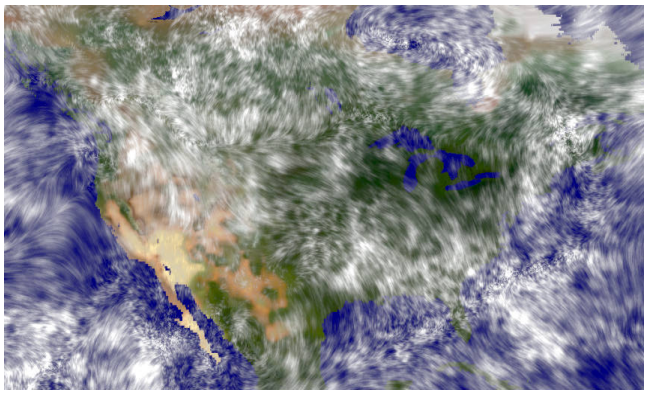

Figure 10: A wind velocity visualization is created by compositing an image of North America under an image of the velocity field rendered using variable length LIC over $1/f$ noise.

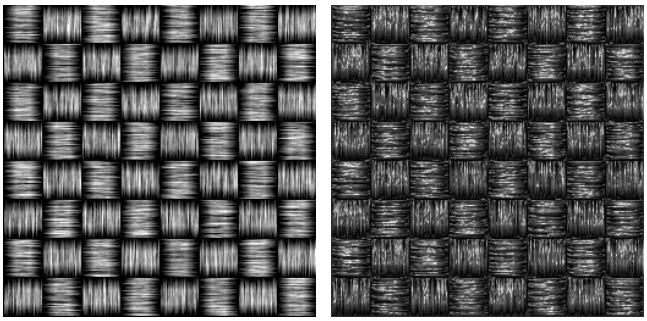

Figure 9: White noise convolved with checkerboard vector field using fixed normalization (left), and then gradient shaded (right) to give the appearance of a rough woven surface texture.

that the input texture is large enough so that the periodicity induced by the coordinate wrapping is not apparent.

The algorithm can be used as a data operator in conjunction with other operators much like those of Sims[17] and Perlin[14]. Specifically, both the texture and the vector field can be preprocessed and combined with post processing on the output image. The LIC implementation is a module in a data flow system like that found in a number of public domain and commercial products. This implementation allows for rapid exploration of various combinations of operators.

4.3.1 POST PROCESSING

The output of the LIC algorithm can be operated on in a variety of ways. In this section several standard techniques are used in combination with LIC to produce novel results.

An interesting example of constant kernel normalization is shown in figure 9. A simple basket weave pattern is generated by alternating vector directions in a checkerboard fashion. Each checker is surrounded by null vectors. This vector field is then used to convolve white noise. The LIC is truncated as it nears the edges of the checkers which results in a gradual attenuation. When that output is gradient shaded, the basket weave becomes very realistic. While other techniques could be used to generate such a texture, the simplicity of the source data illustrates the versatility of LIC.

A surface wind velocity field is imaged in figure 10 using LIC to blur $1/f$ noise. The resulting image is composed over an image of North America to present scale and location. The LIC algorithm is slightly modified to image vector magnitude by varying the length of the line integral, $2L$, as a function of the vector field magnitude. In figure 10 this effect is seen as clumpiness in $1/f$ cloud-like structures where the wind velocity field is small.

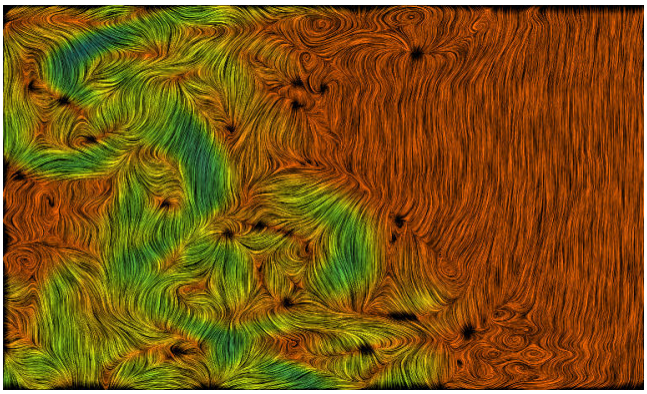

Figure 11: The fixed normalization fluid dynamics field imaged in figure 8 is multiplied by a color image of the magnitude of the vector field.

Figure 12: A photograph (top) of the Flavian Amphitheater is used to create a vector field. The field is then used to image a "painted" version of the same photograph (bottom).

Another method to add vector magnitude information is seen in figure 11. The [fixed normalization] fluid dynamics field of figure 8 is multiplied by a color image of the vector magnitude. The advantage of this approach over variable length LIC is that the fine grained detail generated by fixed length LIC is retained even in low magnitude areas.

The LIC algorithm can be used to process an image using a vector field generated from the image itself. In figure 12, a vector field is generated from the input image by low-pass filtering the image, taking the gradient of the resulting image and rotating the vectors by 90°.

The LIC algorithm can also be used to post process images to generate motion blur. A rendering algorithm or paint system can easily specify a pixel by pixel velocity field for objects. By using a biased triangle filter[10] and variable length LIC the input image can be motion blurred in the direction of apparent motion. This has precisely the desired results for motion blurring as seen in figure 13.

Figure 13: The original photo on the left shows no motion blurring The photo on the right uses variable length LIC to motion blur Boris Yeltsin's waving arm, simulating a slower shutter

4.4 THREE-DIMENSIONAL LIC

The LIC algorithm easily generalizes to higher dimensions. Equations (1), (3) and (5) trivially extend to three dimensions. In the three-dimensional case, cell edges are replaced with cell faces. Both the input vector field and input texture must be three-dimensional. The output of the three-dimensional LIC algorithm is a three-dimensional image or scalar field. This field is rendered using volume rendering techniques such as those found in [21] and [6].

Figure 14 is a three-dimensional rendering of an electrostatic field with two point charges placed a fixed distance apart from one another. In this volumetric rendering, the magnitude of the vector field is used to control the opacity transfer functions. Great efficiency gains can be achieved if the LIC algorithm exploits this by avoiding rendering for vector field cells whose magnitude is outside of the volume renderer's min/max threshold window.

5. PERFORMANCE

There is a distinct performance and quality trade-off between the DDA convolution algorithm and LIC. LIC is roughly an order of magnitude slower than the DDA method. Both algorithms were timed using cells processed per second (CPS) as the figure of merit. The tests were run on an unloaded IBM 550 RISC 6000. The DDA algorithm averages about 30,000 CPS while LIC averages about 3,000 CPS.

The three-dimensional algorithm only marginally degrades in performance with the increase in dimensionality, processing some 1,200 CPS. Since the algorithm remains one-dimensional in nature, the cost per cell only increases by a factor of three as a function of dimension. Using the thresholding described above, the performance of the three-dimensional LIC algorithm has exceeded 30,000 CPS.

6. FUTURE WORK

A number of research directions relating to LIC remain outstanding.

Currently no methods exist for determining the accuracy of a vector field representation, such as those created by LIC or any other method. These accuracy metrics would necessarily be related

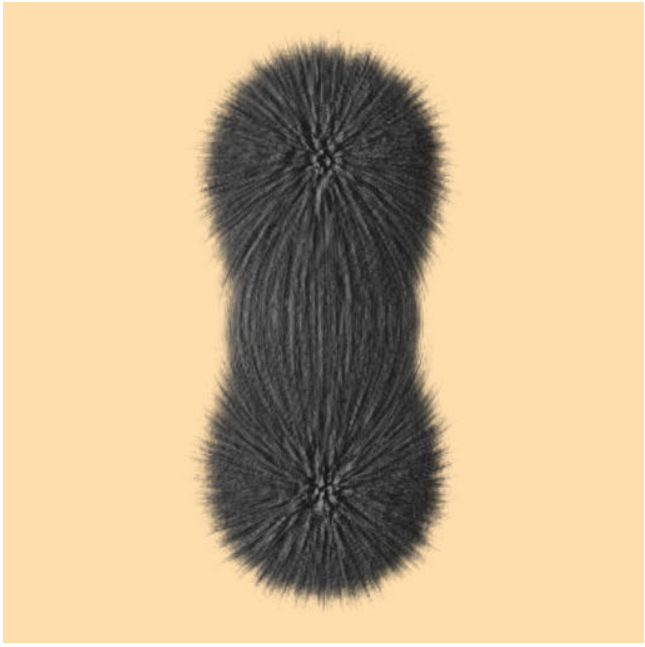

Figure 14: A three-dimensional 512^3 electrostatic field is imaged by volumetrically ray tracing a three-dimensional scalar field produced using LIC over white noise.

to the differential topology of the entire vector field. As mentioned above, much work in theoretical and applied mathematics has been done in this area. This work needs to be studied and applied to efficient vector field imaging algorithms.

LIC is conceptually independent of the advection algorithm used to define the parametric support used by the convolution operation. The method described here might be best characterized as a variable step Euler's method. Other techniques such as a fourth order Runge-Kutta could produce differing or improved results. A thorough investigation into this issue is beyond the scope of this paper. It does, however, represent an area deserving special attention.

Visualizing the orthogonal complement of a two-dimensional vector field is accomplished by rotating the individual vectors 90°. However, in three-dimensional vector fields the orthogonal complement of a vector is a plane. This suggests that a generalization of the one-dimensional LIC filter would be a two-dimensional surface filter. This filter would have as its geometric support a differential surface whose normals would be defined by the vector field, thus creating a Surface Integral Convolution (SIC). As with the LIC, an arbitrary two-dimensional filter could then be used to filter the three-dimensional input image.

Another direction for generalization is to develop versions of the algorithm which operate directly on curvilinear and arbitrarily grided vector fields without resampling the input data. The LIC algorithm could easily be modified to handle arbitrary line intersections and topologies of both type of grids. As with the rectilinear LIC, it would have an analogous three-dimensional generalization. Two additional problems remain however: generating curvilinear and arbitrarily girded textures and output resampling.

One possible image processing application of LIC is the deblurring of motion blurred images. Images acquired with a moving CCD camera often exhibit such blurring. If the CCD frequency response curves and the camera motion are known, one-dimensional deconvolution techniques could be used in conjunction with LIC to deblur the images.

The local nature of the LIC algorithm suggests a parallel implementation. Such an implementation could, in principle, compute all pixels simultaneously. This would allow for interactive generation of periodic motion animations and special effects.

7. SUMMARY

Line integral convolution represents a new and general method for imaging two- and three-dimensional vector fields. The algorithm filters an input image along local stream lines defined by an input vector field and generates an output image. The one-dimensional filter shape is independent of either input and can be arbitrary. To indicate directional flow of the vector field, a whole family of continuous motion filters has been introduced. These filters give apparent motion in the direction of the vector field. The technique can also be used to create special effects. Additionally, the local nature of the algorithm lends itself to efficient and simple implementations.

8. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract W-7405-ENG-48. The SIGGRAPH '93 reviewers provided many helpful comments and suggestions. Thanks to Nelson Max who suggested using higher order functions within a cell and who provided critical assessment all along the way. Roger Crawfis deserves special thanks for various conversations over the past couple of years on the topic of vector visualization. Chuck Grant provided helpful suggestions clarifying the language used to discuss periodic motion filters. John Bell and Jeff Greenough provided the turbulent computational fluid dynamics data used in figures 2, 4, 8 and 11 and for using the algorithm in their work. Dean Williams and Jerry Potter provided the North America wind velocity data. Lastly, thanks to John Zych who helped with the rendering of the North America image.

REFERENCES

1. Bresenham, J. Algorithm for Computer Control of a Digital Plotter. In *IBM Systems Journal* 4, 1 (1965), 25-30.
2. Bronstein, I. and Semendyayev, K. *Handbook of Mathematics*. Van Norstrand Reinholt (1985), 291-293.
3. Chang, S. *Fundamentals Handbook of Electrical Engineering and Computer Engineering*. John Wiley & Sons, Inc. (1982), 264-266.
4. Chui, K. *An Introduction to Wavelets*. Academic Press, Inc. (1992), 49-60.
5. Crawfis, R. and Max, M. Direct Volume Visualization of Three-Dimensional Vector Fields. *Proceedings of the Workshop on Volume Visualization*, Kaufman and Lorensen Eds (1992).
6. Drebin, R., Carpenter, L. and Hanaran, P. Volume Rendering. *Computer Graphics* 22, 4 (August 1988), 65-74.
7. Dumortier, F., Roussarie, R., Sotomayor, J. and Zoladek, H., Study of Field Bifurcations. *Lecture Notes in Mathematics*, Springer-Verlag (1991).
8. Freeman, W., Adelson, E. and Heeger, D. Motion without Movement. *Computer Graphics* 25, 4 (July 1991), 27-30.
9. Haeberli, P. Paint By Numbers: Abstract Image Representation. *Computer Graphics* 24, 4 (August 1990), 207-214.
10. Heckbert, P. Filtering by Repeated Integration. *Computer Graphics* 20, 4 (August 1986), 315-321.
11. Kajiya, J. and Kay, T. Rendering Fur with Three Dimensional Textures. *Computer Graphics* 23, 3 (July 1989), 271-280.
12. Kenwright, D. and Mallinson, G. A 3-D Streamline Tracking Algorithm Using Dual Stream Functions. *IEEE Visualization '92 Conference Proceedings* (October 1992), 62-68.
13. Max, Nelson. Personal Communication (1992).
14. Perlin, K. An Image Synthesizer. *Computer Graphics* 19, 3 (August 1985), 287-296.
15. Perlin, K. Hypertexture. *Computer Graphics* 23, 3 (July 1989), 253-262.
16. Pratt, W. *Digital Image Processing*. 2nd ed. John Wiley & Sons, Inc. (1991), 243-245.
17. Sims, K. Artificial Evolution for Computer Graphics. *Computer Graphics* 25, 4 (August 1991), 319-328.
18. Sims, K. Choreographed Image Flow. *The Journal of Visualization and Computer Animation* 3, 1 (January-March 1992), 31-43.
19. Tufte, E. The Visual Display of Quantitative Information. *Chesire, CT: Graphics Press* (1983).
20. Turk, G. Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion Textures. *Computer Graphics* 25, 4 (July 1991), 289-298.
21. Upson, C. and Keeler, M. V-Buffer: Visible Volume Rendering. *Computer Graphics* 22, 4 (August 1988), 59-64.
22. Van Gelder, A. and Wilhelms, J. Interactive Animated Visualization of Flow Fields. *Proceedings of the Workshop on Volume Visualization*, Kaufman and Lorensen Eds. (1992).
23. Van Wijk, J. Spot Noise Texture Synthesis for Data Visualization. *Computer Graphics* 25, 4 (July 1991), 309-318.
24. Witkin, A. and Kass, M. Reaction-Diffusion Textures. *Computer Graphics* 25, 4 (July 1991), 299-308.