Comparing LIC and Spot Noise

Wim de Leeuw Robert van Liere*

Center for Mathematics and Computer Science, CWI

Abstract

Spot noise and line integral convolution (LIC) are two texture synthesis techniques for vector field visualization. In this paper the two techniques are compared. Continuous directional convolution is used as a common basis for comparing the techniques. It is shown that the techniques are based on the same mathematical concept. Comparisons of the visual appearance of the output and performance of the algorithms are made.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.6 [Computer Graphics]: Methodology and Techniques I.6.6 [Simulation and Modeling]: Simulation Output Analysis.

Additional Keywords: flow visualization, texture synthesis

1 INTRODUCTION

Among the techniques used for the visualization of vector fields, texture based methods are a recent development. By using texture, a continuous visualization of a two-dimensional vector field can be presented. Figure 8 shows a visualization of a slice from a direct numerical simulation using texture. The images show the turbulent flow around a block. These images clearly show the power of texture as a medium for visualization. The visual effect of direction is achieved by line structures in the direction of the vector field. These lines are the result of higher coherency between neighboring pixels in the field direction. Spot noise and line integral convolution (LIC) are two texture based techniques for vector field visualization that make use of this principle.

Before texture was used for data visualization, many papers appeared dealing with the generation of textures [7, 9, 8, 3]. The purpose was artistic or for giving images a realistic appearance. Perlin [9] used directed convolution of random images as a texture synthesis technique to produce images of flames.

Spot noise, introduced by van Wijk [13], was the first texture synthesis technique for the visualization of vector data. A spot noise texture is synthesized by distributing a large number of small intensity functions – called spots – over the domain of the data. Data is visualized by transforming the spot as a function of the underlying vector field. Furthermore the concept of texture animation was introduced for static vector fields. Subsequent texture frames are generated by considering the spots as particles and advecting the spot positions in the vector field.

Line integral convolution, introduced by Cabral and Leedom [1], uses a piece of a streamline as a filter kernel for the convolution of a random texture. Animation can be realized by cyclic shifting of the filter kernel in subsequent frames.

In later papers both LIC and spot noise have been improved and extended. Improvements include increased texture synthesis speeds, generalizations of grid types, usage of the techniques with time dependent vector fields, and zooming in on details.

In this paper we compare both techniques and some extensions. In Section 2, spot noise and LIC are briefly described. We describe the governing algorithms and some extensions. In Section 3 a common basis is given. By using continuous directional convolution as a basis, it is shown that both techniques share a common underlying principle. We also discuss texture synthesis for time dependent vector fields using this common basis. The techniques will be compared with respect to output texture and performance in Section 4. The conclusions will be presented in Section 5.

*CWI, Department of Software Engineering, P.O. Box 94097, 1090 GB Amsterdam, Netherlands. E-mail {wimc|robertl}@cwi.nl

2 LIC AND SPOT NOISE

2.1 Algorithm

A LIC texture is generated by convolution of an input texture with a one-dimensional filter kernel. The shape of the kernel is determined by the shape of the streamline through the pixel. A pixel in the final texture is determined by the weighted sum of a number of pixels along a line in the input texture:

$$C_{out}(i,j) = \sum_{p \subset \tau} C_{in}(p)h(p), \tag{1}$$

where τ is the set of pixels in the input texture used for convolution, $C_{in}(p)$ is the value of the input texture pixel at grid cell p and $h(p)$ is the convolution filter.

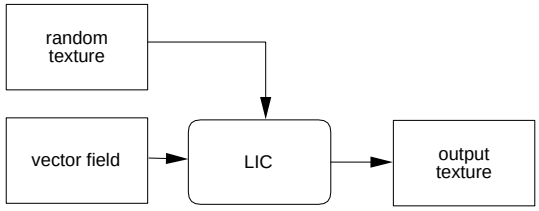

Figure 1: Schematic representation of the LIC-algorithm.

Figure 1 gives an overview of the main components of the algorithm. The inputs of the algorithm are a random texture and the vector field. Streamlines are calculated from the vector field and are used for convolution of the input texture. This results in the output texture.

A spot noise texture is generated by blending together a large number of small intensity functions at random positions on a plane. The shape of the intensity functions is deformed in relation to the vector field. Spot noise is described by the following equation:

$$f(\vec{x}) = \sum a_i h(\vec{x} - \vec{x}_i), \tag{2}$$

in which $h(\vec{x})$ is called the spot function. It is an intensity function which has a non zero value only in the neighborhood of the origin. a_i is a random scaling factor with a zero mean and $\vec{x}_i$ is a random position. The deformation used in [13] was a rotation in the direction of the vector field and scaled by a factor of $(1 + |\vec{v}|)$ in the velocity direction and $1/(1 + |\vec{v}|)$ perpendicular to the field.

Figure 2 shows a schematic representation of the algorithm. The vector field together with a set of random positions and intensities form the input of the algorithm. From these two inputs the positions and shapes of the spots can be determined. This results, after spot blending, in the output texture.

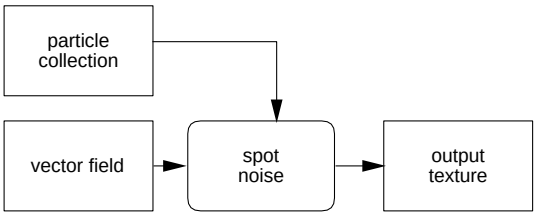

Figure 2: Schematic representation of the spot noise algorithm.

2.2 Extensions

To be a useful tool for the analysis of vector fields, a number of extensions to spot noise and LIC have been proposed. Here we limit our discussion to six extensions:

Non-uniform grids. The data may be defined on a grid with an irregular geometry or topology. Mapping the texture to an curved surface will introduce deformations. If the data has a regular topology the deformation problem can be addressed by transforming the data to a flat geometry. This can be a uniform grid as was described in [2] or a rectilinear grid [6]. Using a rectilinear grid in which the size of the cells matches the cell sizes in the undeformed data results in better mapped textures because the scaling of the texture elements is more uniform.

Performance. Performance is crucial for interactive visualization. Both techniques require substantial computations and performance of the algorithm is therefore important. LIC can be parallelized by partitioning the texture in a number of sub domains [15]. Each sub domain can then be processed in parallel. Furthermore, the generation of LIC texture can be accelerated by using the coherence between successive pixels on a streamline. A substantial performance gain can be achieved by generating long streamlines and reusing them for a large number of pixels. Spot noise can be parallelized because the processing of one spot can be done independently from the other spots. Therefore, processing of all spots can be distributed over a number of processors. A second method to speed up spot noise is by utilizing graphics hardware [6]. Spot rendering and blending can be mapped on functions for which hardware support is available. Both ways to speed up generation have been combined [5].

Animation and time dependent flow. Although the information of a stationary flow is available in a single texture, animation can provide important additional information. Animation is even more important when the flow simulation is time dependent. For stationary flow, a technique was presented where the kernel is shifted for subsequent frames. Animation of time dependent flow can be achieved by UFLIC, described in [10]. Here the values in the texture are deposited along path lines to generate subsequent textures. Using spot noise animation can be achieved by regarding the spots as particles and use advection equations to calculate new spot positions for subsequent frames [13]. More about the use of texture in time dependent flow can be found in Section 3.4.

Zooming. In high resolution simulations flow features can vary three orders of magnitude in size. A single image can impossibly provide all information. Small details in the data can only be perceived if the user is able to magnify a part of the field. In LIC, zooming can be achieved by using high resolution input textures and relating the output texture resolution to the scale at which an image is desired. In spot noise, zooming requires that the texture is regenerated using a smaller part of the data using smaller spots.

3D. Visualization of 3D vector fields is possible with Volume-LIC introduced by Interrante and Grosch [4]. They used the generalization of LIC to 3D in combination with volume rendering for the visualization of 3D flows. Because the presentation of dense volumes is very difficult, selection methods were used to filter the input texture and therefor the area in which the flow is shown. Work on 3D spot noise has not been reported.

Flow direction. Both spot noise and LIC are ambivalent with respect to the direction of the flow. Wegenkittl et al. [14] present a modification of LIC algorithm in which sparse input textures are combined with an oriented filter. In this way the ambivalence in the direction is addressed.

3 COMMON BASIS

In the previous section, we have shown that spot noise and Line Integral Convolution textures are synthesized by considering a neighborhood of a pixel. In this section we will show that both techniques can be described in terms of convolution over a certain region. As an introduction we will use a simplified model to explain the idea. Then, a more formal treatment will be given by using continuous directional convolution.

3.1 Simplification

To illustrate the commonality between the LIC and spot noise techniques, we start with two simplified variations of LIC and spot noise. For LIC, a straight line segment is used in the direction of the flow at the center of the calculated pixel. This is the DDA (Digital Differential Analyzer) convolution as described in Cabral and Leedom [1]. τ_0 (see equation 1) is the set of pixels determined by the line segment when it is rasterized. If a constant convolution kernel is assumed, then a pixel value is calculated as:

$$C_{out}(i,j) = \sum_{p \subset \tau_0} C_{in}(p) \tag{3}$$

For spot noise, scan converted lines of a fixed length are used. The spots are placed at the center of each pixel in the direction of the flow. Now equation 2 can be rewritten as:

$$C_{out}(i,j) = \sum_{p \subset \tau_1} R_{spot}(p) \tag{4}$$

where $R_{spot}(p)$ is the value of the spot at position p.

These equations produce equivalent output texture because C_{in} is equivalent to R_{spot} and τ_0 is the same as τ_1. The intensity of a spot $R_{spot}(p)$ and the pixel value in the input texture $C_{in}(p)$ are both uniformly distributed random values. That the set of input values τ_0 and τ_1 are the same can be seen in Figure 3. On the left, the kernel and the pixels in the input texture are shown. These pixels (grey region) determine the output pixel (the small box) for DDA-convolution. The right image shows the location of spots that influence the pixel of interest (box). A spot influences the pixel of interest if it covers this pixel. Seven spots (the center is shown by black dots) define τ_1. Dotted lines indicate a spot's extent (to avoid cluttering only two dotted lines are drawn).

3.2 Continuous directional convolution

It is not possible to generate images using streamlines of a constant width and constant density. Due to convergence/divergence

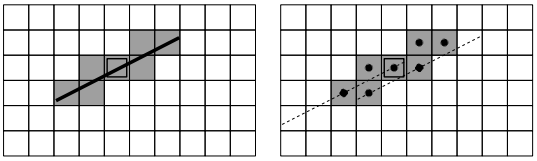

Figure 3: Pixels making up the kernel shape of DDA-Convolution and the location of spots influencing the texture value in simplified spot noise.

the density of lines changes resulting in a higher/lower density. Turks and Banks [12] propose a method which results in an approximation of constant density by calculating the local density of an initial random set of streamlines. This set is iteratively improved by adding or deleting stream lines based on the local density. They also suggest variation of the width of stream lines to get a uniform coverage. Random value based texture techniques also give an approximation of this idea. In texture based techniques the pixels in the direction of the flow do not have equal intensity, but the impression of lines is achieved. This is the result of a higher correlation of pixels in the direction of the flow, compared to perpendicular to the flow. Due to the slow variations which occur, no discontinuities are introduced.

For a more detailed study of this idea, we in introduce continuous directional convolution. The input texture is a continuous function of position: $C_{in}(\vec{x})$ and the convolution equation can be written as:

$$C_{out}(\vec{x}) = \int_{-\infty}^{\infty} C_{in}(\sigma(\vec{s}))k(\vec{x} - \vec{s})d\vec{s} \qquad (5)$$

where $k(\vec{x})$ is the two-dimensional kernel and $\sigma(\vec{x})$ is a two-dimensional deformation function. The function $C_{in}(\vec{x})$ is a two-dimensional white noise signal. Since the kernel is usually non-zero only in a finite region, the integral need only to be evaluated in a region around $\vec{x}$.

The shape of the filter depends on the desired effect. Possibilities are a line with or without a certain width or a more complicated two-dimensional shape (see Figure 4). In terms of frequencies, the

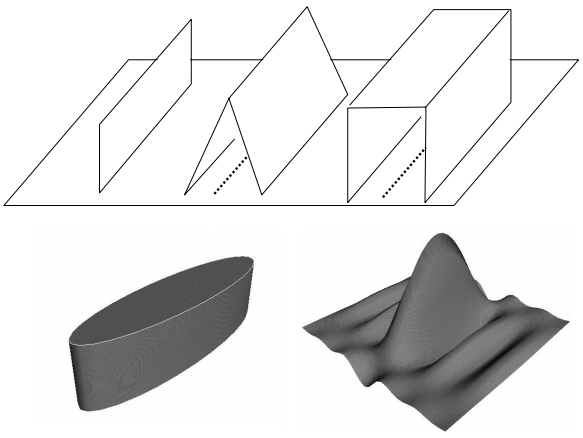

Figure 4: Possible filter shapes for continuous directional convolution, one-dimensional filter, triangle swept along a streamline, swept rectangle, 'spot', and low-pass filter.

purpose of the filter is to achieve an anisotropic filtering of the input where the maximum frequency in the field direction is lower as in the perpendicular direction. According to filtering theory, the best result would be achieved using the an anisotropic low-pass filter (see Figure 4 lower right) with the main axis deformed using a streamline. In practice, however, the convolution must be carried out at a finite resolution. To suppress artifacts due to aliasing introduced by the finite resolution of the texture the same type of anti-aliasing used for the rendering of lines could be used.

It is also possible to encode information regarding the velocity magnitude. This can be done by a parameterizing the filter kernel with the velocity magnitude. In spot noise, the velocity magnitude is encoded in the difference of the frequency in the direction of the flow and the frequency perpendicular to the flow. The same principle could be incorporated in the continuous directional convolution by parameterizing the width of the filter inversely proportional to the velocity magnitude.

Spot noise and LIC are both approximations of continuous directional convolution. The following observations can be made with respect to the approximation:

- In LIC, a scan converted curve is used as the kernel. The kernel has the shape of a connected set of pixels of a particular size. This leads to irregular filtering in the direction perpendicular to the field.
- In spot noise, there are only a finite number of spots. In terms of convolution, the input texture consists of a finite number of randomly placed impulses with a random energy. Since convolution is a linear operator, the convolution integral (equation 5) over this input texture can be evaluated by summation of the separate responses to each impulse (equation 2).

3.3 Conversion between the methods

Continuous directional convolution can be used to 'translate' concepts used by both techniques. Concepts in spot noise can be translated to concepts in LIC and vise versa. For example, in spot noise, the magnitude of the flow is visualized using spot scaling. Because the spot shape performs the same role as the kernel shape in LIC, one might expect that similar results might be obtained in LIC by variation of the length of the kernel. Another example: in spot noise it is easy to generate animations in a stationary flow by spot advection. In LIC this could be realized by advection of the input texture. Each pixel could be regarded as a particle which is advected for some time step. Alternatively, the phase shifting of the kernel technique proposed for LIC could be implemented in spot noise by using a spot shape with a shifting phase in different frames. These examples show that concepts used by the two techniques can be mapped onto each other. The table below lists a number of similar concepts for both techniques.

spot noise	LIC
random spot intensity	random input texture
spot function	kernel shape
spot scaling	kernel length variation
standard spots	DDA convolution
bent spots	streamline convolution
spot advection	texture advection

We could generate spot noise textures using a variant of LIC and, vise versa, LIC textures using a variant of spot noise. The line integral convolution algorithm can be adapted to generate spot noise textures by using a two-dimensional filter domain. The shape of the domain is determined by all positions around the filtered points which, if a spot would be placed there, would cover the filtered point. The spot noise algorithm can be adapted to generate LIC textures. Spots would have the shape of a streamline and would be rendered at the center of each pixel in the texture.

3.4 Time dependent vector fields

A first application of the common basis is the study of texture animation of time dependent vector fields. The challenge of texture animation is to maintain two types of coherence in the textures. To perceive flow, two issues must be addressed. First, spatial coherency (the lower frequency in the field direction as described in the previous section) must be maintained. Second, temporal coherency must be maintained. Temporal coherence is defined as the movement of patterns between texture frames. The impression of movement results when patterns are displaced for a small distance in subsequent frames.

In a stationary flow, temporal coherence is obtained by using the same principle as is used for spatial coherence. Between successive frames texture values are advected along streamlines. Particle positions on a streamline are calculated by:

$$\vec{x} = \vec{x}_0 + \int_{x_0}^{x} \vec{v}(\vec{x})dx \tag{6}$$

where $\vec{v}(\vec{x})$ is the velocity at position $\vec{x}$ For time dependent flow, temporal coherence is obtained only if particle paths are used to advect the texture. A particle path is expressed as:

$$\vec{x}(t) = \vec{x}(t_0) + \int_{t_0}^{t} \vec{v}(\vec{x}(t))dt \tag{7}$$

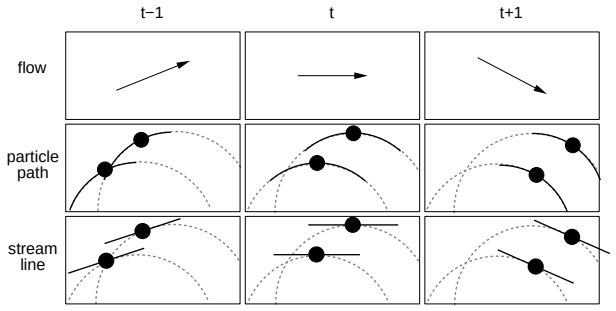

Figure 5: Stream lines or path lines used for coherence in texture

For spatial coherence, streamlines should be used to determine the filtering domain. On the other hand, to get the best possible temporal coherence particle paths should used. This is illustrated in Figure 5. The three columns show different time steps of a flow field. The flow is spatially uniform and the direction varies linear with time, as illustrated in the top row of th Figure. In rows two and three, two kernels are followed over time. Dotted lines indicate particle paths. Bold lines indicate the shape of the kernel. Note that in this particular case streamlines are straight line segments which is not true in general.

Temporal coherence is maintained if particle paths are used, however, it compromises the spatial coherence of the texture. This is because particle paths, and therefore kernels, may intersect, (as is shown in the first column of the Figure) resulting in artifacts in the texture. The crossing of kernels introduces high frequency components in the direction of the flow. Using streamlines compromises the temporal coherence of the texture, as the actual path of the flow may differ from the path suggested by the texture.

Figure 6 compares textures generated using particle paths and streamlines for the kernel shape. The data used for this images is the rotating uniform vector field described in Figure 5. The Figure shows that using particle paths results in high frequencies in all directions and thus compromises spatial coherency in the field direction. Using streamlines high frequencies occur only perpendicular to the field direction.

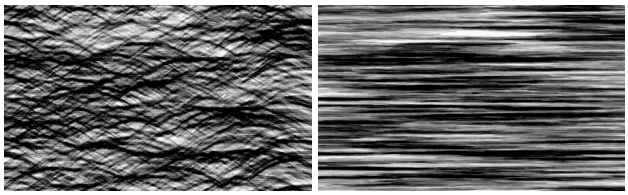

Figure 6: particle paths (left) and streamlines (right) used for texture synthesis in a rotating flow.

There is no perfect solution for the coherency problem. However, useful compromises were taken by UFLIC [10] in which the texture at a certain moment might not represent the current vector field perfectly. Spot noise uses streamlines for the spot shape compromising temporal coherence. These partial solutions are useful as long as the user knows the limitations.

4 COMPARISON

A comparison of the of both algorithms is difficult to realize because the algorithms produce different outputs. It is not possible to adjust the parameters for the methods such that they produce equal textures. Furthermore, there is no metric to compare the information content of texture.

Nevertheless, in this section we will present a metric for comparing textures. We use this metric as an measure to compare the performance of the techniques.

4.1 Output texture comparison

Because both methods do not produce the same texture a metric must be found to compare output textures. We define this metric by introducing the notion of *pixel coverage*. Pixel coverage is defined as the number of random values contributing to a pixel. Textures are defined to be equivalent if the pixel coverage for each pixel is the same.

In the previous section we found that a spot and a kernel are comparable concepts. For a certain kernel a spot can be found such that the area covered is equal to the area of the pixels under a kernel in LIC (see Figure 7). The average area covered be a kernel is the filter length l_f multiplied by the width of a single pixel. If we normalize the width and length of the complete texture to 1 and measure the length of the kernel in pixels, then the area of a kernel A_k is calculated by

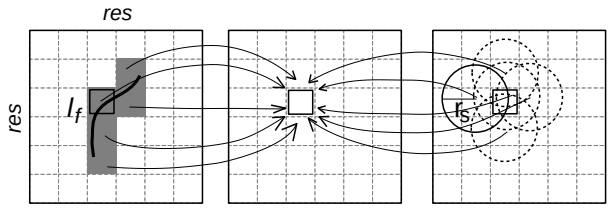

Figure 7: Equal coverage of a pixel: the number of input values which influence a pixel is equal for LIC (left) and spot noise (right)

$$A_k = \frac{l_f}{res^2} \tag{8}$$

where res is the resolution of the texture. A disc shaped spot with a comparable surface has a radius r_s of

$$r_s = \sqrt{\frac{A_k}{\pi}} = \sqrt{\frac{l_f}{\pi res^2}} \tag{9}$$

Each pixel must be covered by l_f spots therefore the number of spots N to be used is

$$N = \frac{l_f}{\pi r_s^2} = res^2 \tag{10}$$

In Figure 8 the same vector field is visualized using both techniques. The image on the left shows the field using LIC while spot noise is used for the right image. The data is a slice from a direct numerical simulation of a turbulent flow around a block and is defined on a rectilinear grid. The resolution of the data is 316×538. The flow is from the bottom to the top of the image. The visualization shows the vortex shedding in the wake behind the block. The resolution of both textures is 512×512. For the LIC texture a kernel length of 20 pixels was used. Using Equations 9 and 10 we obtained values giving an equal pixel coverage for the spot noise image.

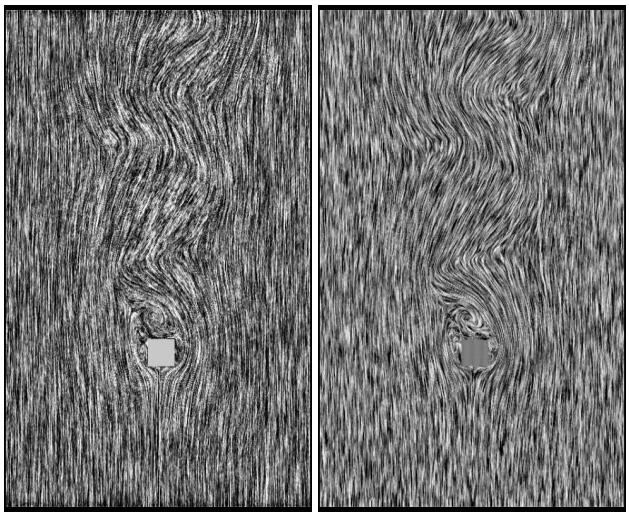

Figure 8: LIC (left) and spot noise (right) images of turbulent flow around a block.

Because the large majority of grid cells in the data is smaller than a texel the amount of detail which could be visible is limited by the texture resolution. For further investigation we used a detail behind the block. This is shown in In Figure 9. In the lower part of the images the block is visible. The data resolution of the section shown is 166×144. Using 512×512 textures the smallest grid cells in the data are about 4 texels in size. In the top left image the filter length used for LIC is 20 pixels while in the bottom left image a filter length of 40 used. The spot noise images have equal pixel coverage using parameters calculated by Equations 9 and 10.

From this side by side comparison a number of differences can be noted. In the LIC image rotation centers are visualized more accurately. For example: the LIC images clearly show the two distinct rotation centers in the rotation area in the top left of the field. The velocity in the rotation centers is relatively low and therefor the spot noise textures become almost isotropic. In addition to displaying velocity direction information, the spot noise image shows the magnitude of the field; e.g. the upper right corner of the image

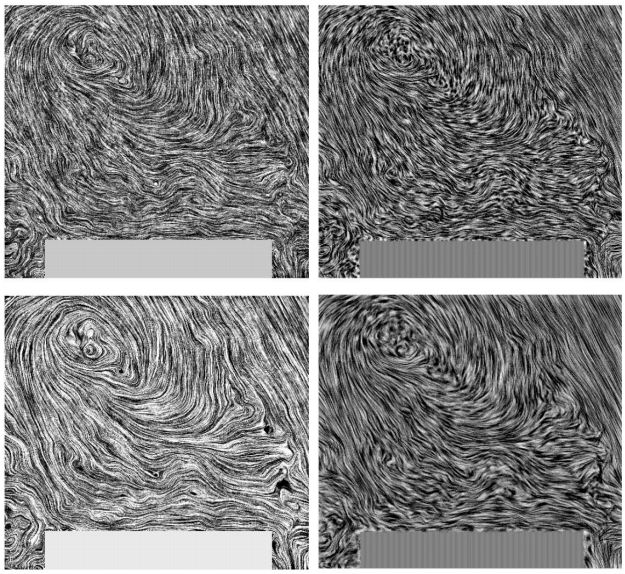

Figure 9: LIC (left) and spot noise (right) images with equal pixel coverage using a filter length of 20 and a spot radius of 0.005. (top) and using a filter length of 40 and a spot radius of 0.007. (bottom)

shows a region of higher velocity. Unfortunately, this extra information decreases resolution of the directional information. The frequency range needed to encode the velocity magnitude reduces the frequency at which the directional information is displayed.

4.2 Performance comparison

The performance can be compared in several ways. First we will do some order estimation of the performance of the algorithms. Second, we will look at extensions to the algorithms for increased performance. Finally, we will present some measurements.

In unaccelerated LIC the time needed to generate a texture increases linear with the number of pixels in the output texture and linear with the length of the kernel. For spot noise the time needed increases linear with the number of the spots and linear with the area of the spots. If the comparable textures are generated, such as described in the previous section, it is easy to see that the order of generation time of the algorithms is equal. However, the basic operations which have to be carried out are different. For spot noise scan conversion operations are needed and for LIC convolution and stream line integration operations are needed.

The previous analysis is valid for the original algorithms. Several ways have been proposed to speed up the algorithms. In the algorithm for LIC proposed by Stalling and Hege [11] the algorithm consists of two steps. In the first step stream lines are calculated and in the second step the convolution is carried out by successive processing of pixels along streamlines where results are reused. In this way, the complexity of the algorithm becomes independent of the filter length. For spot noise, graphics hardware can be utilized to speed up scan conversion an blending of the spots [6]. Although this does not change the order of complexity, substantial gains can be achieved in the generation time. Parallelization is another way to speed up the algorithms. LIC can be parallelized by dividing the texture in tiles [15]. Parallelization of spot noise is possible by distributing the spots over the processors. The combination of hardware acceleration and parallelization for spot noise is possible if the available processor power is matched by the speed of the graphics hardware [5].

As a test case to compare the speed the algorithms we used the detail of the DNS described in the previous section (see Figure 9). The tests were run on a SGI Indigo^2 workstation equipped with a 250 MHz R4400 processor and a High Impact graphics board. For LIC the original implementation as described in [1] was used. For spot noise an implementation taking advantage of graphics hardware was used. The results for this unfair comparison were 93.5 seconds for LIC and 6.7 seconds for spot noise. For the LIC image a filter length of 20 was used, the spot noise image was generated with a spot radius of 0.005. For the images with a filter length of 40 and a spot radius of 0.007 the times were 182.3 and 6.6 respectively.

We get a better comparison if use the results of accelerated LIC with the times presented for spot noise. The timing results presented in [11] indicate that 4.6 seconds are needed for the generation a similar LIC texture on slightly slower hardware using the acceleration techniques described in the paper. This would suggest that LIC is slightly faster than spot noise.

In this comparison we used textures with equal pixel coverage. However, a number of parameters in spot noise allow trading quality for speed. Figure 10 shows spot noise images in which less spots were used. Compared to the spot noise textures in Figure 9, 20 percent of the number of spots were used in the left image, and 5 percent in the right image. The times needed to generate these images were 1.3 and 0.35 seconds.

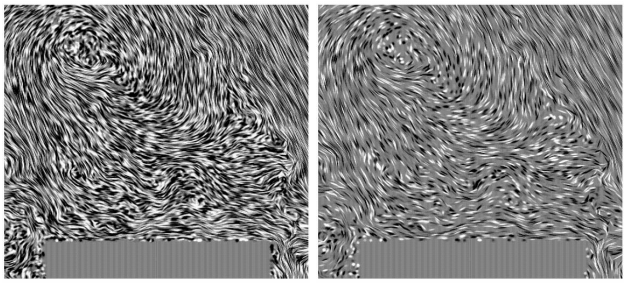

Figure 10: Trading quality for speed in spot noise. Texture using 50000 spots (left) and 12500 spots (right)

5 CONCLUSION

In this paper LIC and spot noise were compared. Both techniques use texture synthesis for vector field visualization. Directional information in textures is encoded by coherence between neighboring pixels. Due to differences between the techniques and the concepts used to discribe them, a direct comparison would be very difficult. By using continuous directional convolution as a model, the similarity in the underlying mathematical basis becomes clear, and similar concepts in the techniques can be found.

The diffences in information presented by both techniques are due to the fact that LIC does not encode velocity magintude. Therefore, the spatial resolution for presenting directional information is higher than for a comparable spot noise texture. If the acceleration scemes proposed for the techniques are taken into account the differences in performance of LIC is slightly better than of spot noise. Spot noise is more flexible with respect to trading texture quality for generation speed.

Continuous Directional Convolution is used to show that for texture animation of time dependent flow, it is not possible to fully satisfy the requirements of spatial coherence in the texture and temporal coherence in frames. We believe that continuous directional convolution can serve as a basis for future study and improvements of texture synthesis techniques for flow visualization.

Acknowledgments

Thanks to Arthur Veldman and Roel Verstappen University of Groningen for using their data. We are grateful to the reviewers who gave valuable ideas for improvements of the paper. This work is partially funded by the Dutch foundation High Performance Computing and Networking (High Performance Visualization project).

References

[1] B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series, pages 263–272. ACM SIGGRAPH, August 1993.

[2] L.K. Forssell and S.D. Cohen. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-speed Animation, and Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

[3] P. Haeberli. Painting by Numbers: Abstract Image Representations. In *Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, volume 24, pages 207–214, July 1990.

[4] Victoria Interrante and Chester Grosch. Strategies for Effectively Visualizing 3D Flow with Volume LIC. In R. Yagel and H. Hagen, editors, *Proceedings of Visualization '97*, pages 421–424, Los Alamitos (CA), 1997. IEEE Computer Society Press.

[5] W.C. de Leeuw and R. van Liere. Divide and Conquer Spot Noise. In *Proceedings Super Computing '97 (http://scxy.tc.cornell.edu/sc97/program/TECH/DELEEUW/INDEX.HTM)*, 1997.

[6] W.C. de Leeuw and J.J. van Wijk. Enhanced Spot Noise for Vector Field Visualization. In G.M. Nielson and D. Silver, editors, *Proceedings Visualization '95*, pages 233–239, Los Alamitos (CA), 1995. IEEE Computer Society Press.

[7] J-P Lewis. Texture Synthesis for Digital Painting. In *Computer Graphics (SIGGRAPH 84 Conference Proceedings)*, volume 18, pages 245–251, July 1984.

[8] D. R. Peachey. Solid Texturing of Complex Surfaces. *Computer Graphics (SIGGRAPH 85 Conference Proceedings)*, 19(3):279–286, July 1985.

[9] K. Perlin. An Image Synthesizer. In *Computer Graphics (SIGGRAPH 85 Conference Proceedings)*, volume 19, pages 287–296, July 1985.

[10] H.-W. Shen and D.L. Kao. UFLIC: A Line Integral Convolution Algorithm for Visualizing Unsteady Flows. In R. Yagel and H. Hagen, editors, *Proceedings Visualization '97*, pages 317–322, Los Alamitos (CA), 1997. IEEE Computer Society Press.

[11] D. Stalling and H.C. Hege. Fast and Resolution Independent Line Integral Convolution. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 249–256, August 1995.

[12] G. Turk and D. Banks. Image-Guided Streamline Placement. In *SIGGRAPH 96 Conference Proceedings*, pages 453–460, July 1996.

[13] J.J. van Wijk. Spot Noise – Texture Synthesis for Data Visualization. In *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, volume 25, pages 263–272, July 1991.

[14] R. Wegenkittl and Eduard Gröller. Fast Oriented Line Integral Convolution for Vector Field Visualization. In R. Yagel and H. Hagen, editors, *Proceedings of Visualization '97*, pages 309–316, Los Alamitos (CA), 1997. IEEE Computer Society Press.

[15] M. Zöckler, D. Stalling, and H. Hege. Parallel Line Integral Convolution. In A. Chalmers and F.W. Jansen, editors, *Proceedings First Eurographics Workshop on Parallel Graphics and Visualization*, pages 249–256. held in Bristol, UK, 26-27 September, 1996.