Enhanced Spot Noise for Vector Field Visualization

Willem C. de Leeuw

Delft University of Technology
Faculty of Technical Mathematics and Informatics
P.O.Box 356, 2600 AJ Delft, The Netherlands.

Jarke J. van Wijk

Netherlands Energy Research Foundation ECN
P.O.Box 1, 1755 ZG Petten
The Netherlands.

Abstract

Spot noise is a technique for texture synthesis, which is very useful for vector field visualization. This paper describes improvements and extensions of the basic principle of spot noise. First, better visualization of highly curved vector fields with spot noise is achieved, by adapting the shape of the spots to the local velocity field. Second, filtering of spots is proposed to eliminate undesired low frequency components from the spot noise texture. Third, methods are described to utilize graphics hardware to generate the texture, and to produce variable viewpoint animations of spot noise on surfaces. And fourth, the synthesis of spot noise on grids with highly irregular cell sizes is described.

1 Introduction

Although numerous techniques have been proposed [1], vector field visualization remains a challenging subject. Most methods use mappings of the vector field to shape and color. Texture is a visual primitive that has been much less explored. An advantage over other techniques is that texture can give a continuous view of a 2D field opposed to discrete positions, as with arrow plots or streamlines.

This paper is based on earlier work of van Wijk [2]. We will refer to the technique described in that paper as *standard* spot noise. A texture can be characterized by a scalar function f of position $\mathbf{x}$. A spot noise texture [2] is defined as

$$f(\mathbf{x}) = \sum a_i h(\mathbf{x} - \mathbf{x}_i), \qquad (1)$$

in which $h(\mathbf{x})$ is called the spot function. It is a function everywhere zero except for an area that is small compared to the texture size. a_i is a random scaling factor with a zero mean, $\mathbf{x}_i$ is a random position. In non-mathematical terms: spots of random intensity are drawn and blended together on random positions on a plane (figure 1).

The use of a spot as a basis for a texture has two nice consequences. First the shape of the spot determines the

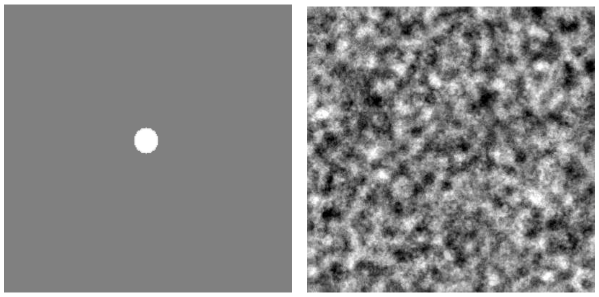

Figure 1: Principle of spot noise: single spot (left) spot noise texture (right)

characteristics of the texture; the appearance of the texture can be controlled by the shape of the spot. Different textures result from different spot shapes. Second, local control of the texture is possible. Vector fields can be effectively visualized, if the shape of the spot is adapted to the data at the position of the spot (figure 2).

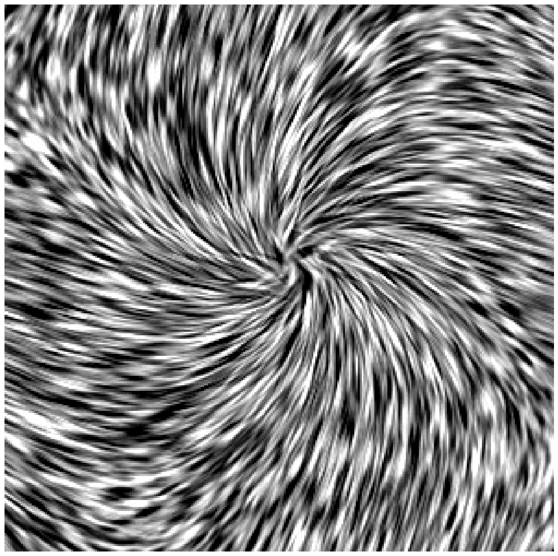

Figure 2: spot noise used to visualize a vector field

Other texture synthesis techniques for vector field visualization are *texture splats* [3] and *Line Integral Convolution* (LIC) [4]. Texture splats are small textures used for volume rendering. They can be adapted to show vector fields by slight disturbance of the reconstruction function used to generate the splat. LIC was proposed by Cabral and Leedom [4] and extended by Forssell [5]. A LIC texture is generated by convolution of an input texture with a one dimensional filter kernel, aligned with the streamline through the filter center.

Experiments with standard spot noise showed that it gives satisfactory results in many cases, but in some cases problems occur. In areas where the curvature of the vector field is high, the texture does not show the vector field very well. A better representation of highly curved vector fields using spot bending will be described in section 2. In section 3 filtering of spots is discussed, to remove undesired low-frequency components from the texture. Section 4 deals with implementation problems of spot noise. We will describe techniques to accelerate the texture synthesis of spot noise by using graphics hardware. In section 5 we describe a general technique to generate spot noise textures for highly non-uniform grids. Examples of enhanced spot noise are presented in section 6. Finally, the conclusions can be found in section 7.

2 Spot bending

In standard spot noise the spots are rotated and scaled to reflect the local flow. First the spot is aligned with the flow direction by a rotation, then it is scaled. In the direction of the flow it is scaled proportional to $1 + |\mathbf{v}|$ and perpendicular to the flow by $1/(1 + |\mathbf{v}|)$, where $|\mathbf{v}|$ is the velocity magnitude. This scaling ensures that the total area of the spot and thus the average spot density in the texture remains constant. A zero velocity results in a round spot.

The spot influences a region in the texture while its shape is based on data at a single point. If the velocity varies strongly over this region, the shape of the spot does not reflect the data properly. The result is a blurred texture (figure 4). The solution of using smaller spots would only partially solve the problem, for there are always cases in which the velocity gradient is too high to be well represented by a spot of a certain size.

For a better representation of the flow field we need deformation of the spot such that a better representation of the flow field is achieved. Cabral and Leedom suggested using a stream line to warp the major axis of an elliptical spot [4]. This results in a correct representation of the flow along a stream line. However, because the spot is 2D, not only effects of curvature have to be taken into account, but also effects of convergence and divergence of the flow.

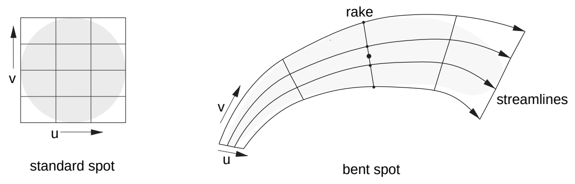

Figure 3: Spot bending

For this deformation of the spot we used a stream surface (figure 3). A stream surface is constructed by integration of streamlines from a rake perpendicular to the flow in the center of the spot. The parameterization of this stream surface using u and v is used to deform the spot polygon that is parameterized in the same way. The effect of this method of deformation is that the spot is 'shaped' by the flow. The problem now is to keep the spot area constant regardless of the flow field; all spots should cover an equal area.

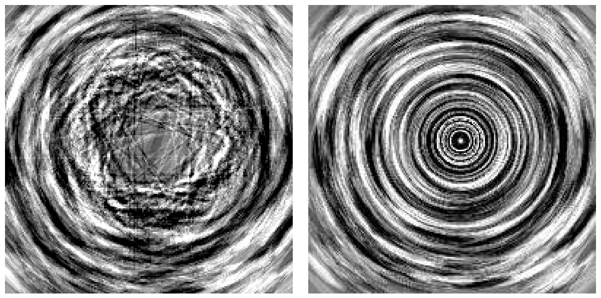

Figure 4: Standard spots (left) and locally adaptive spots (right) compared, using a data set of a vortex with velocity inversely proportional to distance from the center.

The stream surface is generated as follows: the velocity $\mathbf{v}$ at the center of the spot is determined. Next, a rake is positioned perpendicular to the velocity. The length of the rake l_r is defined by

$$l_r = w\frac{1}{1 + |\mathbf{v}|}, \tag{2}$$

where $|\mathbf{v}|$ is the velocity magnitude at the center of the spot, and w a factor that determines the size of the spot with respect to the texture. The length of the rake is equal to the width of a standard spot, inversely proportional to the velocity magnitude. From the rake, streamlines are traced forward and backward at regular intervals. While the width of the spots is reduced according to the velocity(equation 2), the length of the spot l_s is increased, such that the total area of the spot is constant:

$$l_r l_s = w^2. \tag{3}$$

Hence the total time interval T for which the stream surface is traced is

$$T = w\frac{1+|\mathbf{v}|}{|\mathbf{v}|}. \tag{4}$$

Note that this is an approximation, since we assume here that the velocity magnitude does not change over the surface of the spot. However due to the conservation of mass the spot area will not change. For example, if the velocity decreases along the stream line the length of the spot will decrease but also it will get wider due to conservation of mass, and thus the area will not change. The streamlines are traced forward and backward for an equal time interval: $T/2$. The mesh of triangles resulting from the stream surface generation is rendered using the spot as a texture.

The effect is clearly visible in the vortical flow in figure 4. If standard spots are used, the center of the vortex looks cluttered and the velocity cannot be deduced from the texture. The picture generated using bent spots at a resolution of 16 in the v-direction and 8 in the u-direction gives a good result even very near to the vortex core.

Synthesis of bent spots is expensive because streamlines have to be traced and a mesh must be rendered for each spot. For performance reasons we implemented an option of conditional spot bending. The difference between a bent spot and a standard spot is determined by the change of velocity over the surface of the spot. In regions where the velocity changes are small, the difference between a standard spot and a bent spot is also small. We use the difference vectors between the velocity vector in the center of the spot and those at the four corners to decide if a bent spot or a standard spot should be used. Bent spots are rendered if the magnitude of the largest difference vector is larger than one tenth of the length of the spot. In experiments this value proved to result in a good balance between image quality and rendering speed.

As shown in [2] spot noise can also be generated by convolution of white noise with the spot function. From this fact it can be seen that LIC with white noise as input is very similar to spot noise using adaptive spots. LIC uses a one dimensional convolution kernel along a streamline. If we reduce the dimension of the spot function perpendicular to the flow to zero and use a spot of fixed length the result would be equal to LIC. In this case the shape of the (one dimensional) spot in spot noise is the equivalent of the shape of the convolution filter in LIC. However, the additional options in spot noise allow us to visualize the magnitude of the flow.

3 Spot filtering

Spot noise pictures often have a coarse, low-frequency component. This component is not related to the shape of the spot, but only to its size. Irrespective if we use circular or a square spots , this low frequency component remains. If we can remove it, changing the shape of the spot will have more effect, and details in the texture will stand out more clearly. Because the power spectrum of a single spot and the resulting texture are equal, removal of the low frequencies from a spot results in textures without the undesired low frequencies. To remove low frequencies from a spot it is prefiltered with a high-pass filter.

As a high-pass filter $f(x,y)$ we used a negative Gaussian with an impulse at the origin (See figure 5):

$$f(x,y) = \delta(x,y) - \frac{c}{\pi}e^{-c(x^2+y^2)}. \tag{5}$$

c is a parameter which controls the width of the Gaussian and thus the cut-off frequency of the filter. The filter function is discretized over an area of finite size. The value of the pixel in the center of the filter is chosen such that the average value over the discretized filter is zero. A Gaussian filter is a suitable choice, that effectively reduces low-pass components. However, experiments showed that the choice is not very critical, other filters which approximate the sinc function can also be applied. The result of filtering a single spot is shown in figure 5.

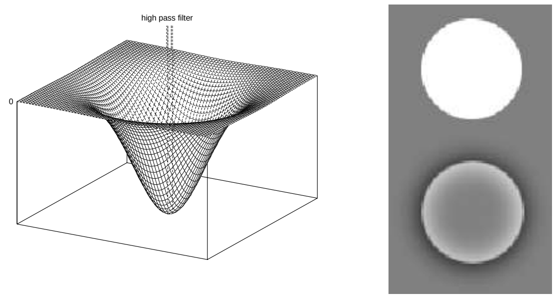

Figure 5: Spot filtering: the shape of the filter function (left) and the result of filtering a single spot (right)

The result of using filtered spots is shown in figure 6. In this figure a vertical slice from a data set of a simulated storm is visualized using spots with a triangular shape. In the left image unfiltered spots are used. The low frequency components can be clearly seen as large dark and light areas. In the right image a Gaussian filter is used. Here the texture is more homogeneous and shows the vector field accurately. Note also that the texture shows the magnitude of the flow. In the center area in the lower part the textures are fine-grained, while in the vortices at the left and right edge the texture is coarser. The use of a triangle as the basic spot shape results in a texture in which also lines *perpendicular* to the flow are present.

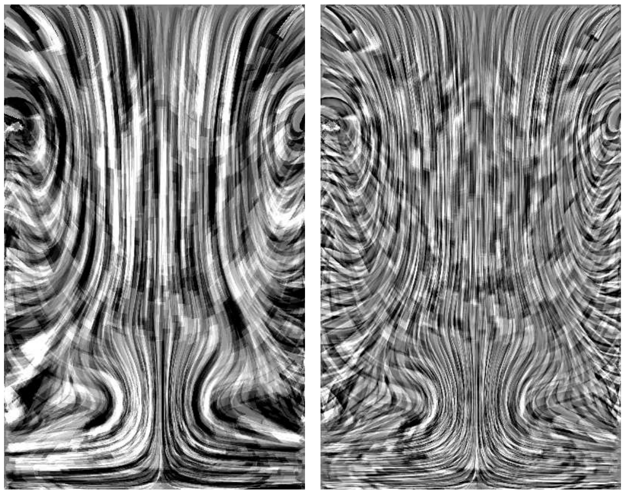

Figure 6: Data set visualized using unfiltered (left) and filtered (right) spots

4 Using graphics hardware

Visualization is an interactive process, therefore fast generation of pictures is essential. We used the graphics hardware available on current high performance graphics workstations such as the SGI Onyx to speed up both the texture synthesis and the image rendering. In a preprocessing step we generate the textures; next these textures are mapped onto polygons for image rendering.

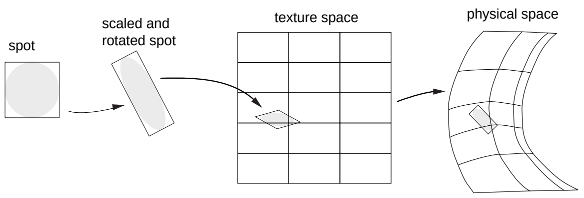

Figure 7: Steps in the synthesis of a spot noise texture

The synthesis and inspection of spot noise textures is done in a sequence of steps, depicted in figure 7. First the spot is transformed to reflect the data; then it is blended with the rest of the spots to produce the texture; finally, this texture is mapped onto the surface.

A typical spot noise texture consists of tens of thousands of spots. The transformation and blending of these spots takes a considerable amount of computation time. However, the nature of spot noise allows us to take advantage of the polygon rendering and texture mapping hardware. We represented the spot as a single texture [6]. This texture is mapped onto a unit polygon centered at the origin. To obtain a final spot in texture space, this polygon is scaled and rotated as a result of adapting the shape of the spot to the vector value. Finally, the spot is translated to its position in texture space and blended. This series of transformations is achieved by using the matrix stack of the graphics hardware normally used for the viewing pipeline.

If bent spots are used, scaling and rotation transformations are not needed, because we generate the stream surface in texture space. The points on the generated stream surface can be used as positions for rendering the mesh which represents the spot. This mesh is rendered with the texture of the spot. The time and stream line number of a particular point are used to determine the corresponding location in the spot texture.

The variation in intensity of the spot is achieved by using the possibility of weighted blending of the rendered object with the picture generated so far. The hardware compositing only supports addition of intensities, but spots can also have a negative intensity, thus subtraction of intensity is needed. Solving this problem without losing the advantage of the hardware is done by splitting equation 1:

$$\begin{aligned} f(\mathbf{x}) &= \sum a_i h(\mathbf{x}-\mathbf{x}_i) \\ &= \sum b_i h(\mathbf{x}-\mathbf{x}_i) - \sum c_i h(\mathbf{x}-\mathbf{x}_i) \\ b_i &= \begin{cases} a_i & \text{if } a_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ c_i &= \begin{cases} 0 & \text{if } a_i \geq 0 \\ -a_i & \text{otherwise} \end{cases} \end{aligned} \tag{6}$$

This way positive-valued spots and negative-valued spots can be accumulated separately by the hardware and only a single subtraction per pixel is needed. Two separate textures are generated, one for positive spots and one for the negative spots. The two resulting textures are then subtracted in software to produce the final texture.

After synthesis of the textures, an interactive rendering module displays them in three dimensions. The textures are mapped onto a model description of the surface. The model can be inspected by varying the viewpoint with the mouse. Animated texture is achieved by cyclic display of a number of frames with advected texture. As the mapping from texture to physical space is done on the fly by the hardware the viewpoint can be varied during the animation. Times to generate a spot noise texture range from less than a second for a single texture of reasonable quality, to several minutes for a high-quality sequence of textures for animation purposes.

Color can be used to visualize an independent variable on the same surface. A color map and a spot noise texture are blended by using the value from the spot noise texture as the intensity and the hue from the color map as the hue in the blended picture.

5 Non-uniform grids

As stated in the previous section, spot noise texture is generated on a uniform grid, whereas data often is defined on a non-uniform curvilinear grid. The spot noise textures generated in texture space are transformed to physical space. To prevent distortion of the texture, and thus wrong visualization of the data, the spots have to be predistorted in texture space. The transformation of a cell in computational space to a cell in physical space is specified by the Jacobian matrix. By applying the inverse Jacobian matrix transformation to the spots before they are rendered, the spots will have the desired shape in physical space.

We used linear interpolation to calculate the position and data values between grid points. For this purpose each cell is split into two triangles. Also for the transformation between texture space and physical space we used the decomposition in triangles. Each triangle of the surface has its own transformation. Because many spots have to be rendered all these transformations are precalculated and stored. The performance penalty for using this transformation is very small because it can be appended to the series of transformations needed to map a unit spot to the final spot in texture space as described in the previous section.

The advantage of this method of interpolation is that processing is fast. The surface can be rendered by rendering flat triangles instead of bilinear patches which must be rendered if bilinear interpolation is used. Also the interpolation of values needs less computation compared to bilinear interpolation.

As the result of this triangular mapping from physical space to computational space, the velocity field transformed to texture space becomes discontinuous. This can be seen in figure 8, which shows a constant field on a single trapezium-shaped cell with spot noise rendered onto it. In texture space there is a discontinuity along the diagonal of the square. The velocity in computational space at a grid point depends on the triangle bordering the point chosen.

Because each triangle has its own transformation a problem occurs if we want to apply the predistortion to a standard spot if it covers more than one cell. Several parts of the spot should be transformed differently depending on the cell the part lies in. This problem does not occur if bent spot are used. If a streamline crosses a cell boundary velocity will also change such that the spot will have the desired shape in the final image. In figure 8 bent spots were used.

Many curvilinear grids are highly irregular with respect to cell sizes. In regions where high velocity gradients are expected, such as boundary layers, the simulation is carried out at a higher resolution. If a uniform grid is used for texture synthesis, the pixel density of the generated texture is equal for each cell. If the texture is transformed back into physical space, in small cells many texels (texture elements) are squeezed into a small area while texel-resolution is low in large cells.

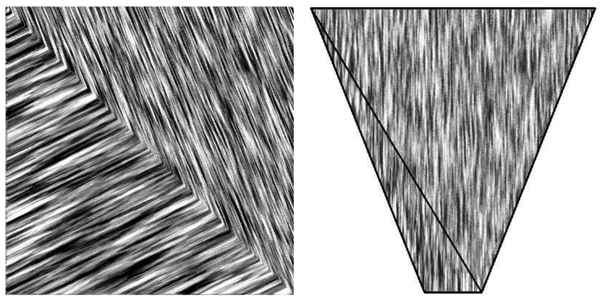

Figure 8: Texture space (left) and physical space (right) for spot noise texture. The field is constant and in the vertical direction

Hence, it is more efficient to adapt the size of the texture cells to the size of the physical cells. We realized this by using a rectilinear texture space instead of a uniform texture space. The distance between two lines in the rectilinear texture space is chosen such that the area of the strip between the two lines has the same proportion to the total area as the equivalent strip in physical space (figure 9).

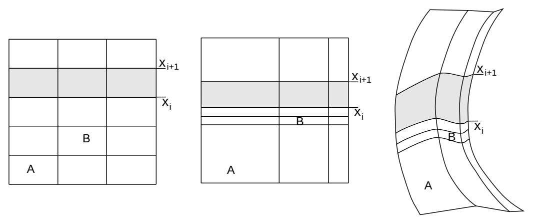

Figure 9: Less irregular scaling results if rectilinear texture space (middle) is used instead of uniform texture space (left) to represent the curvilinear physical space (right).

6 Examples

We present a number of examples of the use of enhanced spot noise in flow data sets.

Stream surfaces are created by advecting particles released along a line in the flow. Visualization of a stream surface gives only gives partial information of the flow on the surface. The magnitude and direction within the plane is lost. Spot noise generated for the curvilinear stream surface can be used to visualize this information. The result of mapping spot noise onto a stream surface is shown in figure 10.

Flow simulations often produce other data along with velocity, such as pressure and temperature. The spot noise texture can be blended with color to show such a quantity.

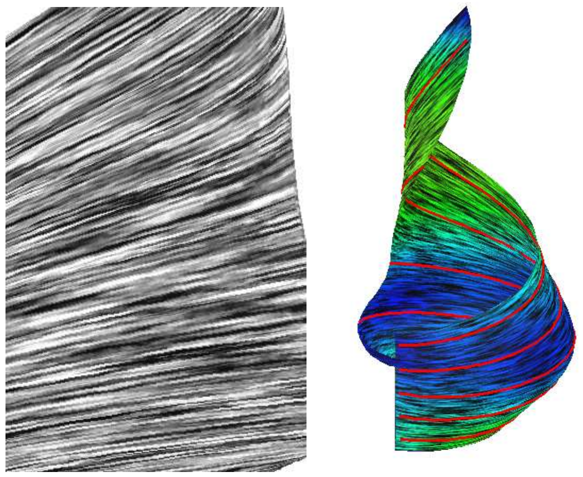

Figure 10: Two views of spot noise on a stream surface in a data set of a storm. Left: detail, right: combination with other techniques, streamlines and velocity magnitude mapped to color. Data courtesy: R. Wilhelmson (NCSA).

In figure 11, pressure is mapped to color. The surface is a slice through a data set of a backward facing step. The use of spot bending is essential here to show the main flow in which the velocity is high together with the recirculation behind the step.

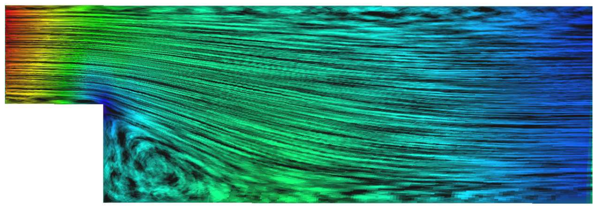

Figure 11: Slice of a backward facing step, color showing pressure. Data courtesy: Dept. of Numerical Mathematics, Delft University of Technology

In wind tunnel experiments the near wall velocity can be visualized using pigmented oil. We simulated pictures from these experiments using enhanced spot noise [7]. As input for the spot noise algorithm we used the wall friction, the derivative of the velocity perpendicular to the wall. This results in pictures which are very similar to the photographs obtained from the described windtunnel experiments. Figure 12 is a spot noise picture of the wall friction on a fin/wedge configuration. Some slight modifications were made to obtain images comparable to the windtunnel picture. Opposed to standard spot noise, only positive spot intensities were used to emulate the white pigment on the black fin surface. Further, the intensity of the spots was scaled by the value of the convergence $(-\nabla \mathbf{v})$ of the wall friction field. to simulate the effect of the transport of oil over the surface during the experiment, and thus accumulation in regions of high convergence.

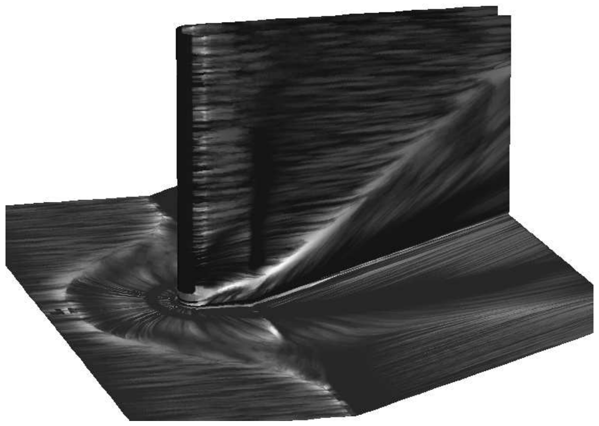

Figure 12: Wall friction data on blunt fin visualized using spot noise. Data courtesy: T. Gerhold and H.G. Pagendarm (DLR Göttingen, Germany)

7 Conclusion

In this paper we described several enhancements of spot noise for surface vector field visualization. The enhancements we proposed are spot bending and spot filtering. Spot bending is useful for the visualization of vector fields with high curvature. Prefiltering of spots can be used to generate more homogeneous textures. The use of graphics hardware for faster image generation was described, as well as methods for texture synthesis on data sets defined on a curvilinear grid. As a result of the enhancements described, spot noise can be applied to a large variety of vector field visualization problems.

If we compare enhanced spot noise with LIC, it has the advantage that the vector magnitude is visible in a single image. Although it is possible to show magnitude using LIC by varying the kernel length [4], it has a negative effect on the image quality. Another advantage is that animations of (time dependent) flow are easily generated by treating a spot as a particle which is advected by the flow. No special precautions are needed to show local variations in speed. Direct speed comparison between spot noise and LIC is not possible because the end results are not equal. For LIC images, a function has to be evaluated for each pixel in the texture. In spot noise the generation time depends on the number of spots used to generate the texture. By chosing the number of spots a trade-off can be made between image quality and rendering speed.

In this paper spot noise is only used in two dimensions.

However, the concept can be extended to three dimensions, using a density function as a spot and volume rendering to display results.

We think texture is a useful visual primitive for the visualization of vector fields, and spot noise with the enhancements presented above is an efficient and accurate technique for the generation of such textures.

Acknowledgements

We thank Frits Post for useful discussions on spot noise, and comments on earlier versions of the paper. The fin data and many valuable ideas with regard to the visualization of wall friction were provided by Hans-Georg Pagendarm. We would also like to thank Remko Vaatstra for his help with the implementation of the algorithm. This work is supported by the Netherlands Computer Science Research Foundation(SION), with financial support of the Netherlands Organization for Scientific Research (NWO).

References

[1] F.H. Post and J.J. van Wijk. Visual representation of vector fields: Recent developments and research directions. In L.J. Rosenblum et al., editor, *Scientific Visualization: Advances and Challenges*, pages 367–390. Academic press, 1994.

[2] J.J. van Wijk. Spot noise – texture synthesis for data visualization. In T.W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 263–272, July 1991.

[3] R.A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In G.M. Nielson and D. Bergeron, editors, *Proceedings Visualization '93*, pages 261–265. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[4] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In J.T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 263–272, August 1993.

[5] L.K. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In D. Bergeron and A. Kaufman, editors, *Proceedings Visualization '94*, pages 240–247. IEEE Computer Society Press, 1994.

[6] N. Max, R. Crawfis, and C. Grant. Visualizing 3D velocity fields near contour surfaces. In D. Bergeron and A. Kaufman, editors, *Proceedings Visualization '94*, pages 248–255. IEEE Computer Society Press, 1994.

[7] W.C. de Leeuw, H.G. Pagendarm, F.H. Post, and B. Walter. Visual simulation of experimental oil-flow visualization by spot noise images from numerical flow simulation. In R. Scateni, J.J. van Wijk, and P. Zanarini, editors, *Visualization in Scientific Computing '95*, pages 135–148. Springer Verlag, 1995.