Animating Flow Fields: Rendering of Oriented Line Integral Convolution

R. Wegenkittl, E. Gröller, W. Purgathofer
Institute of Computer Graphics
Vienna University of Technology
Vienna, Austria, 1040

Abstract

Line Integral Convolution (LIC) is a common approach for the visualization of 2D vector fields. It is well suited for visualizing the direction of a flow field, but it gives no information about the orientation of the underlying vectors. We introduce Oriented Line Integral Convolution (OLIC), where direction as well as orientation are encoded within the resulting image. This is achieved by using a sparse input texture and a ramp like (anisotropic) convolution kernel. This method can be used for animations, whereby the computation of so called pixel traces fastens the calculation process. In the result section various OLICs illustrating simple and real-world vector fields are shown.

1 Introduction

The visualization of vector fields, e.g., flow visualization or visualization of dynamical systems, is an important part of scientific visualization. Displaying small vectors at discrete points within the vector field (so-called arrow plots) only gives a rough overview of the underlying dynamics. Some of the techniques, e.g., streamlines and streaklines suffer from the disadvantage of showing only flow direction at locally restricted regions of the flow field [7].

An interesting approach towards a global visualization of flows has been done by Jarke J. van Wijk with his spot noise technique [10]. He used a vector field to control and modulate the generation of bandlimited noise. A random texture is convolved with locally varying convolution kernels whose directions are tailored to the tangential directions of the flow. The quality of the results of this method depends to a large extend on the type of the texture used. Furthermore the calculation of spot noise is a time consuming task. The spot noise idea has been extended to follow curved streamlines [5].

In 1993 Brian Cabral and Leith Leedom presented a method called line integral convolution (LIC) [1]. It is a modification of van Wijk's method, as convolution takes place along a (curved) streamline segment which is determined by integrating the vector field.

The method calculates for every point $P_0(x + 0.5/y+0.5)$ in the center of a pixel of the output image a streamline represented by a polyline. This polyline covers l pixels in positive and l pixels in negative integration time. Weighted by the length of the line segment within a single pixel the pixel intensities of an input image are summed. Typically an input image consists of dense white noise. The summing along the polyline is additionally weighted by a so called convolution kernel. For single images a constant convolution kernel gives a good impression of the flow directions. For the production of animations the convolution kernel is shifted for each frame. This gives the impression of flowing ripples, which also encodes the orientation of the flow.

Lisa K. Forssell extended the LIC technique to curvilinear grids [3]. Due to the distortions introduced by curvilinear grids, animating the convolution kernel with equal kernel lengths over the entire grid produces a distorted and misleading flow visualization. Lisa Forssell overcame that problem by adapting the length of the convolution kernel during the calculation of the LIC. Speed encoding is achieved by changing according to the velocity of the vector field the amount of phase-shift of the convolution kernel.

Calculating a LIC is a very time consuming task. Detlev Stalling and Hans-Christian Hege reduced calculation times by introducing the Fast LIC technique [8]. They exploit coherence in the LIC calculation which occurs to a large extent along streamlines. The computation order is not pixel-per-pixel but entire streamlines are processed at a time. This gives an order of magnitude speed-up with the drawback of allowing only simple convolution kernels. Additionally their method allows to continuously zoom into a specific area of the vector field without problems. Thus the input texture and the resulting image do not have

1087-4844/97 $10.00

to be of the same resolution.

Turk and Banks dealt with the problem of distributing small streamline segments over a vector field [9]. Starting with a random placement of small streamlines they optimize the coverage of the vector field with streamlines by moving and connecting streamline segments.

All variations of the LIC method presented until now do not encode the orientation of a flow within a still image. The presented Oriented Line Integral Convolution (OLIC) described in the following sections overcomes this disadvantage by using anisotropic convolution kernels on a sparse texture. In contrast to the dense noise texture used for LIC this sparse texture consists of distinct white spots on a black background as input texture. OLIC also allows velocity encoding in still images and animations. The computation of animated OLICs can be accelerated by precalculating so called pixel traces. In the result section we show some examples of simple and realistic vector fields visualized with the new method.

2 Oriented Line Integral Convolution (OLIC)

One of the main differences between Line Integral Convolution and Oriented Line Integral Convolution is that OLIC uses sparse textures rather than dense noise textures usually taken for LIC. This naturally implies distinct lines in the resulting image. As a physical justification for this approach one can think of Line Integral Convolution as distributing some drops of ink over a sheet of paper and smear them according to the underlying dynamics of a vector field. These ink droplets produce a sparse texture instead of the dense white noise textures typically used for LIC. Figures 1,2 and 3 show a texture image as it is typically used by LIC and two texture images used by our algorithm as well as the resulting LICs for a simple vector field (saddle point). In both cases the constant function is used as convolution filter kernel.

LIC with dense textures provides directional information for every point in the image. Due to the resulting high frequencies the direction of the flow is, however, not as easy perceptable as in the more structured LIC produced by a sparse texture. So we used a pattern of equidistant white dye droplets as input texture for a standard LIC procedure as shown in figure 2. The much coarser output gives an impression of the flow that is easier to understand, but due to the regular pattern of the texture some undesireable artifacts are introduced.

By perturbing (jittering) the positions of the equidistant droplets in the input texture these artifacts disappear (figure 3) and the resulting image resembles pretty much the LIC in figure 1 albeit with distinct lines. Advantages of this approach are:

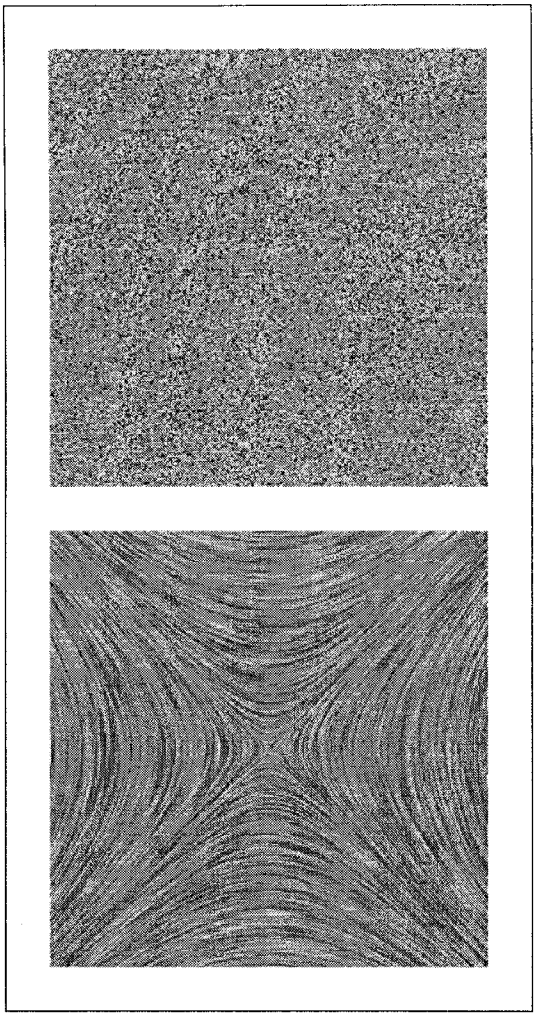

Figure 1: Dense texture image with resulting LIC of a simple vector field (saddle point)

- the dynamics of the vector field can be seen more easily (more structure)
- scaling (e.g. downsizing) the resulting LIC (for example for printouts) does not produce artifacts
- the resulting LIC can be used as a texture for curved surfaces, where the density of the traces of the droplets gives additional depth and curvature cues
- the distinct lines of the resulting images allow velocity encoding by the length of the pixel traces
- the resulting LIC shows distinct traces of droplets. For OLICS we will use these to encode

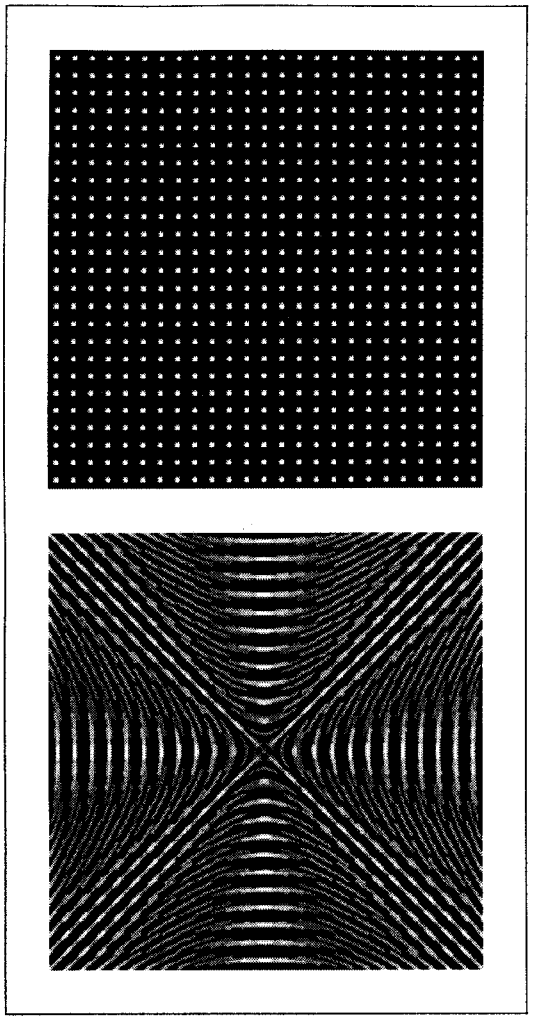

Figure 2: Regular sparse texture image with resulting LIC of a simple vector field (saddle point)

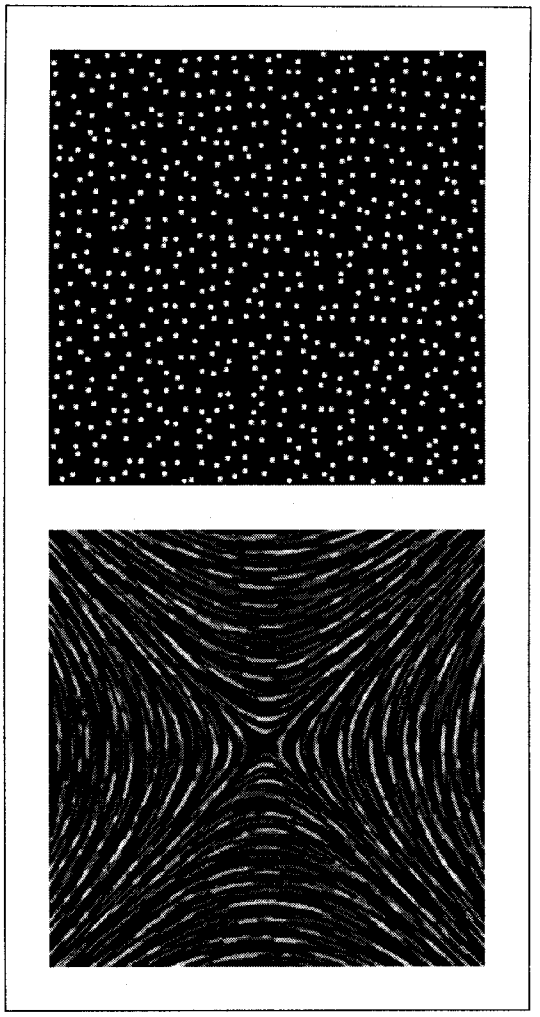

Figure 3: Jittered sparse texture image with resulting LIC of a simple vector field (saddle point)

the orientation of the flow in addition to its direction. The difference of direction and orientation can be seen in figure 4, where two streamlines with equal direction but opposite orientation are shown.

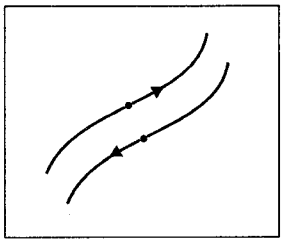

Figure 4: Two streamlines with equal direction but opposite orientation

Especially the last argument is quite advantageous. Until now encoding the orientation of a flow with LIC has only been possible by animation. In reality a researcher would like to know the direction of the flow as well as its orientation even in still images.

Since single traces of pixels are distinguishable in our approach, their intensity can be used easily to encode orientation. Every ink droplet of the sparse input texture thus has a trace comparable to the tail of a comet showing the temporal evolution of the underlying flow. High intensity areas of a pixel trace (set of all pixels covered by an ink droplet moving a short distance according to the underlying vector field) correspond to the current position of an ink droplet. Faded areas of the pixel trace, however, correspond to pixels which have already been crossed by the ink droplet during previous time steps. The principle is illustrated in figures 5 and 6, where two simple rotational flow fields are given, which differ only in the

orientation of the flow. LIC shows no difference for the two vector fields. Images produced with the OLIC technique on the other hand give insight into both the direction and orientation of the flow.

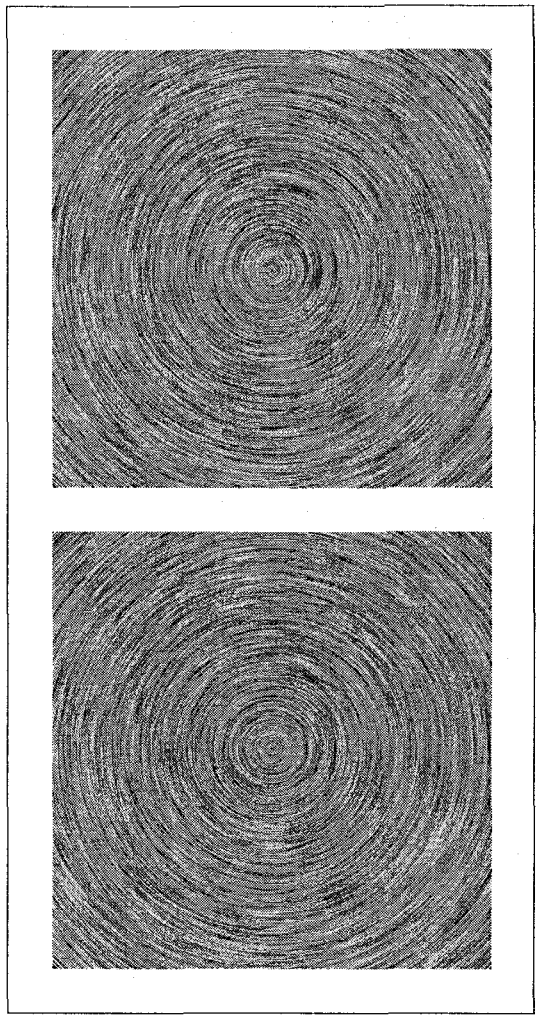

Figure 5: LIC of two flows with same direction but opposite orientation are indistinguishable

The decreasing intensity for each pixel trace can be achieved by using an assymetric convolution kernel (figure 7). A simple ramp shaped function as shown in figure 7 was used for the images of figure 6.

These pixel traces can be seen as paths which are nonuniformly motion blurred due to the speed of the vector field. Thus encoding the speed at a specific pixel by adjusting the length of its trace is a promising approach. The length of the filter kernel has to be adapted correspondingly so that the ramp shaped function exactly covers the length of the trace.

Due to the fact that each pixel is traced in temporal forward and reverse direction, the length of the

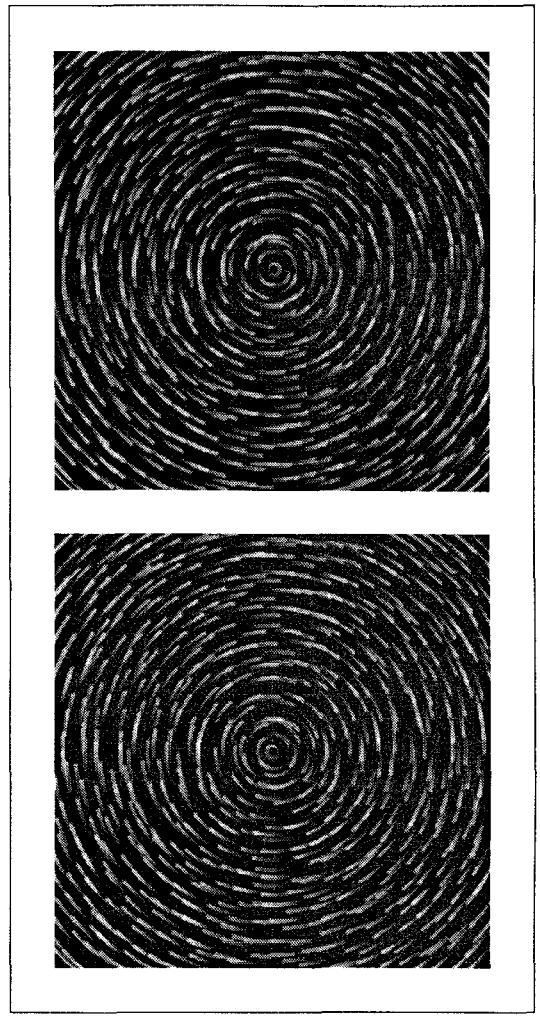

Figure 6: OLIC for two flows with same direction but opposite orientation (same flows as in figure 5

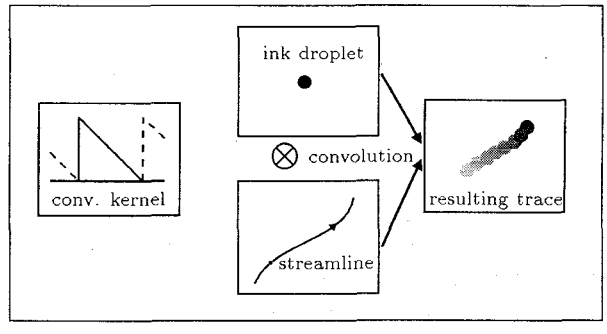

Figure 7: Ramp-like kernel-function and convolution for OLIC

trace corresponds to the speed in the (temporal) middle of each trace. This method of encoding flow velocity gives only a feeling for relative speed differences within the flow field and not an absolut speed infor-

mation. But this seems to be no major drawback. Figure 8 shows another simple vector field with speed encoded in the length of the pixel traces, where the length of the pixel traces vary from 0 to 30 pixels.

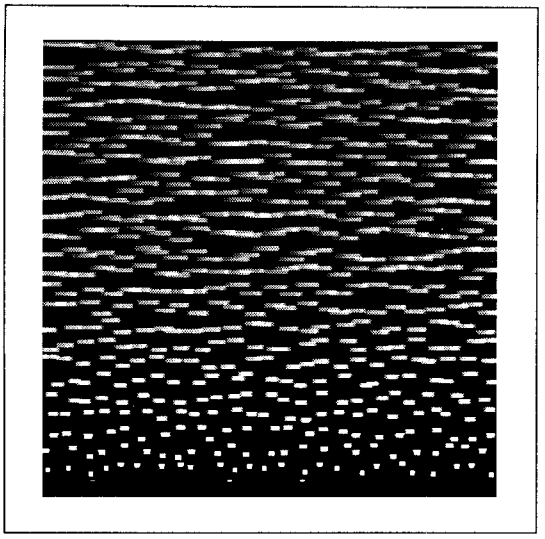

Figure 8: Flow velocity encoded in the length of a pixel trace

3 Animation using Fast OLICs

Oriented Line Integral Convolution encodes direction, orientation and speed of a vector field in a single image. Nevertheless all this information can be interpreted more easily with animation. The animation of Oriented Line Integral Convolutions can be achieved by simply phase shifting the convolution kernel for each frame accordingly. It has to be taken into account, that velocity encoding leads to kernels with different lengths, thus producing an animation that can not be looped. This problem can be overcome by adapting the phase shift to the length of the filter kernel in a way, that each kernel is cycled within the same number of frames. Due to this adaption the visible effect is, that slow parts of the vector field induce short pixel traces with short phase shifts. This results in slow moving spots along the pixel trace. A fast region of the flow induces long pixel traces with big consecutive phase shifts. This translates into a fast moving spot on the long pixel trace.

The result of the above described method is an animation that seems to pulse every cycle it is looped. This effect is due to the fact, that in the first frame every pixel trace has its brightest spot at the beginning of the trace and lower intensity towards the end of the trace. To overcome this unwanted synchronized effect an individual offset for the phase shift of each droplet in the input texture has to be taken into account. Since the output image is calculated pixel by pixel the algorithm has to know which droplet (with corresponding specific initial kernel phase shift) in the input texture produces the pixel trace which covers the current pixel under consideration.

This information must be provided by a precalculation step, where for each pixel the coordinates of the responsible droplet are stored. Now every droplet may have its own phase shift offset and the looped animation looks smooth.

The precalculation of pixel traces can also be used to speed up the calculation of an animation. Due to the distances between distinct droplets of the input texture some pixels of the output image are not covered by pixel traces, thus no calculation has to be done for these pixels. The determination of the information whether a pixel is covered or not is also done by the above mentioned precalculation step. Depending on the density of the input image and therefore on the density of the pixel traces the speed up factor is approximately two to three (since a thirty to fifty percent coverage of the output image with pixel traces gives good results).

Figure 9 shows the pixel traces and the corresponding OLIC of a simple vector field. For every white spot in the image of pixel-traces some information is stored. This information basically contains the identification of the corresponding ink droplet and its initial kernel phase shift. So far only one droplet can be stored for each trace. The errors induced by overlapping traces seem to be negligible. Nevertheless a modified version of the OLIC algorithm is planned which handles this situation correctly.

4 Implementation and Future Work

We implemented an experimental software system within an OpenGL environment [6]. The windows managment is made by GLUT, so the software runs on various platforms [4]. For future work an internet page is planned, where a Java applet allows the computation of different types of Line Integral Convolutions [2]. The idea is, that researchers provide their vector fields to the applet, which then calculates still images and animations of standard LICs and OLICs. This should give researchers the possibility to visualize their vector data easily without having to use any complex visualization tool.

One drawback of Oriented Line Integral Convolution is that overlapping pixel traces are not handled correctly. The problem can be solved by two different approaches. First, the precalculation step could store every droplet that has influence at a specific pixel of the output image (until now, only one droplet

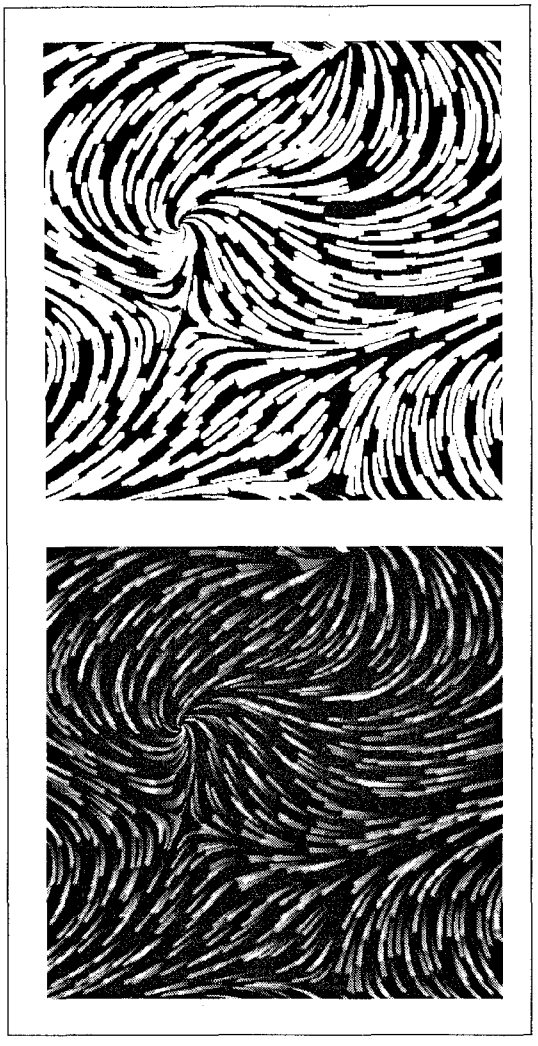

Figure 9: Pixel traces and corresponding OLIC of a flow field

is stored). The other approach is to implement an adapted type of the Fast Line Integral Convolution method as proposed by Stalling and Hege [8]. This will additionally speed up the calculation of Oriented Line Integral Convolution.

5 Results

The images in figure 10, 11 and 12 have been generated with the described OLIC method. Figures 10, 11 use no speed encoding, whereas figure 12 uses the length of the traces to display speed. Figure 10 shows an econometric model describing the interactions of the hight of the budget deficit and the publicity of politicians. Notice that the width of the pixel traces gives additional information whether the flow is divergent or convergent.

Figure 11 shows an artificial model similar to a predator-prey model as described by Volterra and Lotka.

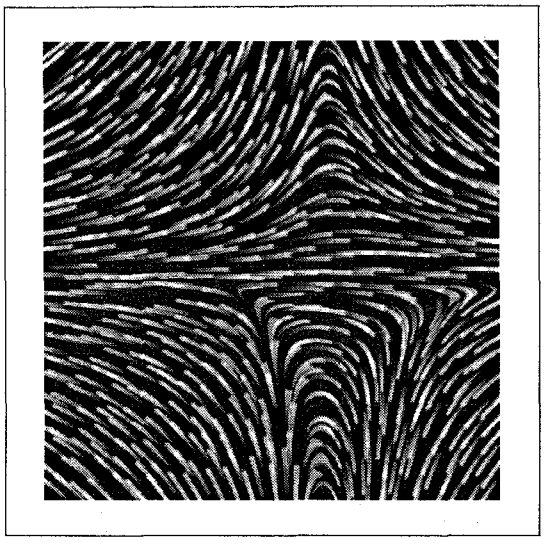

Figure 10: OLIC of an econometric model

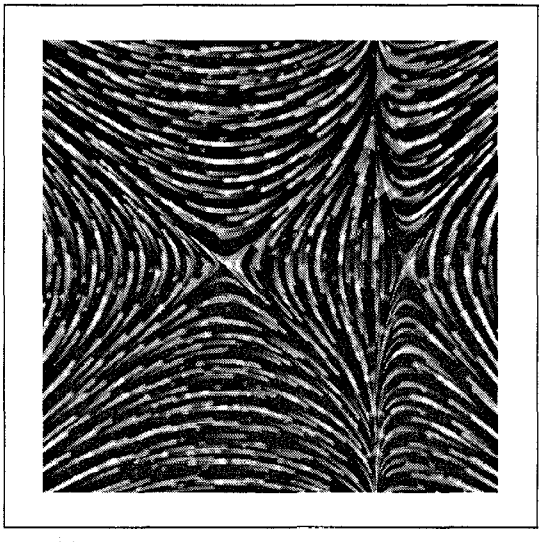

Figure 11: OLIC of a dynamical system similar to a predator-prey model

In figure 12 we used the length of the pixel traces to encode speed. Maximum speed is shown with traces of 60 pixels whereas slow traces have a length of about 12 pixels. It is a good idea to limit the lower bound of the pixel trace length to ensure that no directional information is lost (see also figure 8).

For additional results and to view some animation sequences we refer to our WWW-page:
http://www.cg.tuwien.ac.at/research/vis-dyn-syst/olic/.

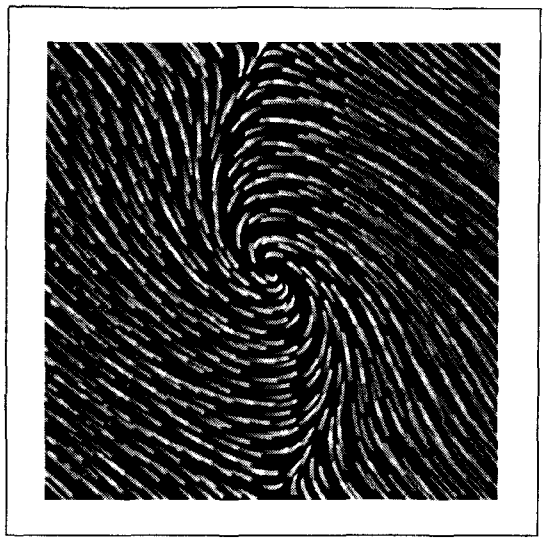

Figure 12: OLIC of a simple pendulum model with speed encoding

Acknowledgments

This work is partially supported by the Platform for Animation and Virtual Reality (PAVR), a EU Training and Mobility of Researchers (TMR) program.

References

[1] B. Cabral, C. Leedom, "Imaging Vector Fields Using Line Integral Convolution", *Computer Graphics Proceedings '93*, ACM SIGGRAPH, pp. 263–270, 1993.

[2] D. Flanagan, *Java in a Nutshell*, O'Reilly & Associates, Inc., 1996.

[3] L. K. Forssell, "Visualizing Flow over Curvilinear Grid Surfaces Using Line Integral Convolution", *Proceedings of IEEE Visualization '94*, pp. 240–247, 1994.

[4] M. J. Kilgard, *The OpenGL Utility Toolkit (GLUT) Programming Interface*, Software Documentation, Silicon Graphics, Inc., 1996.

[5] W. C. de Leeuw, J. van Wijk, "Enhanced Spot Noise for Vector Field Visualization", *Proc. Visualization '95*, IEEE CS Press, 1995.

[6] J. Neider, T. Davis, M. Woo, *OpenGL Programming Guide*, Addison-Wesley, 1995.

[7] F. H. Post, J. J. Wijk, "Visual Representation of Vector Fields: Recent Developments and Research Directions", *Scientific Visualization - Advances and Challenges*, Academic Press Ltd., pp. 367–390, 1994.

[8] D. Stalling, H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution ", *Computer Graphics Proceedings '95*, ACM SIGGRAPH, pp. 249–256, 1995.

[9] G. Turk, D. Banks, "Image-Guided Streamline Placement", *Proc. SIGGRAPH '96*, pp. 453–458, 1996.

[10] J. J. van Wijk, "Spot Noise Texture Synthesis for Data Visualization", *Computer Graphics*, Vol. 25(4), pp. 309–318, July 1991.