Animatng Flowfields: Rendering of Oriented Line Integral Convolution

R. Wegenkittl, E. Gröller, W. Purgathofer
[wegenkittl,gröller,purgathofer]@cg.tuwien.ac.at
Institute of Computer Graphics, Vienna University of Technology

Abstract

Line Integral Convolution (LIC) is a common approach for the visualization of 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 low frequency input texture and a ramp like (unisotropic) convolution kernel. This method can be animated, whereby the computation of so called pixel traces fastens the calculation process. In the result section various OLICs using 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 [PoW94].

An interesting approach towards a global visualization of flows has been done by Jarke J. van Wijk with his spot noise technique [Wijk91]. 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 orientation is tailored to the tangential directions of the flow. The quality of the results of this method depends on a large extend on the type of the texture used. Furthermore the calculation of spot noise is a time consuming task.

In 1993 Brian Cabral and Leith Leedom presented their paper introducing the Line Integral Convolution Method (LIC) [CaLe93]. 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 which 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 high frequency 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 changed (shifted) at each frame. This gives the impression of flowing ripples, which also encodes the orientation of the flow.

Lisa K. Forssell extended the Line Integral Convolution to curvilinear grids [Fors94]. 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 the amount of convolution-filter phase-shift according to the velocity of the vector field.

Calculating a LIC is a very time consuming task. Detlev Stalling and Hans-Christian Hege reduced calculation times by introducing their Fast LIC technique [StHe95]. 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 zoom into a specific area of the input texture of the LIC procedure. Thus the input texture and the resulting image do not have to be of the same resolution.

All variations of the Line Integral Method presented until now do not encode the orientation of a flow within a still image. The presented Oriented Line Integral Convolution Method OLIC described in the following sections overcomes this disadvantage by using nonisotropic convolution kernels on low frequency textures. 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 textures of much lower frequency than those usually taken for LIC. This naturally implies a lower frequency in the resulting image as well. As a physical justification for this approach one can think of Line Integral Convolution as distributing some drops of ink over a sheat of paper and smear them according to the underlying dynamics of a vectorfield. These ink droplets produce a texture of much lower frequency than the white noise textures used by LIC. Figure 1 shows a texture field as it is typically used
by LIC and two texture fields used for our algorithm as well as the resulting LICs for a simple vector field (saddle point). In both cases the constant function is used for the filter kernel.

![Figure 1: High Frequency versus Low Frequency Texture Map with Resulting LICs](image)

LIC with high frequency textures provides directional information for every point in the image. Due to the occurring high frequencies the direction of the flow is, however, not as easy perceptible as in the more structured low frequency LIC. So we used a pattern of equidistant white dye droplets as input texture for a standard LIC procedure as shown in the middle column of figure 1. 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 undesirable artifacts are introduced.

By perturbing (jittering) the positions of the equidistant droplets in the input texture these artifacts disappear (right column in figure 1) and the resulting image resembles pretty much the left LIC in figure 1 albeit at a lower frequency. Advantages of this approach are:

- the dynamics of the vectorfield 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 resulting low frequency 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 the orientation of the flow in addition to its direction. The difference of direction and orientation can be seen in figure 2, where two streamlines with equal direction but opposite orientation are shown.

![Figure 2: Two streamlines with equal direction but opposite orientation](image)

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 low frequency 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 figure 3, 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.

The decreasing intensity for each pixel trace can be achieved by using an assymmetric convolution kernel (figure 4). A simple ramp shaped function as shown in figure 4 was used for the images of figure 3.

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 trace corresponds to the speed in the (temporal) middle of each trace. Since this method of encoding flow velocity only gives a feeling for relative speed differences within the flow field and not an absolute speed information this seems to be no major drawback. Figure 5 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.

Figure 5: Flow velocity encoded in the length of a pixel trace

3 Animation using Fast Rendered 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 within an animation. The animation of Oriented Line Integral Convolusions 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 pushing a fast moving spot along the 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 trace beginning 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 is responsible for a pixel trace covering 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 paths can also be used to speed up the calculation of an animation. Due to the low frequency 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 image with pixel traces gives good resulting images).

Figure 6 shows the pixel traces and the corresponding OLIC of a simple vector field. For every white point in the pixel-traces image an information is stored. This information basically contains the identification of the corresponding ink droplet and its initial phase shift offset. 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.

![Figure 6: Pixel traces and corresponding OLIC of a flow field](image)

4 Implementation and Future Work

We implemented an experimental software system within an OpenGL environment [NeDa95]. The windows management is made by GLUT, so the software runs on various different platforms [Kilg96]. For future work an internet page is planned, where a Java applet allows the computation of different types of Line
Integral Convolutions [Flan96]. 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 is stored). The other approach is to implement an adapted type of the Fast Line Integral Convolution method as proposed by Stalling and Hege [StHe95]. This will additionally speed up the calculation of Oriented Line Integral Convolution.

5 Results

All images have been generated with the described OLIC method. Figures 7, 8 use no speed encoding, whereas figure 9 uses the length of the traces to display speed. Figure 7 shows an econometric model describing the interactions of 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 7: OLIC of an econometric model](image)

Figure 8 shows an artificial model similar to a predator-prey model as described by Volterra and Lotka.

In figure 9 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 5).
Figure 8: OLIC of a dynamical system similar to a predator-prey model

Figure 9: OLIC of a simple pendulum model with speed encoding

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

References


