COLOR-TABLE ANIMATION OF FAST ORIENTED LINE INTEGRAL CONVOLUTION FOR VECTOR FIELD VISUALIZATION

Siegrun Berger, Eduard Gröller

Institute of Computer Graphics
Vienna University of Technology, Karlsplatz 13/186/2
A-1040 Vienna
Austria
e-mail:groeller@cg.tuwien.ac.at

ABSTRACT

Fast Oriented Line Integral Convolution (FROLIC), which is a variant of LIC, illustrates 2D vector fields by approximating a streamlet by a set of disks with varying intensity. FROLIC does not only show the direction of the flow but also its orientation. This paper presents color-table animation of FROLIC images. Various color-table compositions are discussed in detail. When animating FROLIC images visual artifacts (pulsation, synchronization) must be avoided. Several strategies in this respect are dealt with. Color-table animation of FROLIC has been implemented as a Visual C++ application, whereby the calculation of the dynamical system is performed with Mathematica. This allows researchers from various disciplines to conveniently explore and investigate analytically defined 2D and 3D vector fields.

Keywords: oriented line integral convolution, color-table animation

1. INTRODUCTION

Several texture-based techniques for the visualization of flow fields have been investigated in recent years. With Line Integral Convolution (LIC) [CaLe93] a white noise input texture is filtered along curved streamline segments. The intensity $I(x_0)$ at an arbitrary position x_0 of the output image is calculated by

$$I(x_0) = \int_{s_0 - s_l}^{s_0 + s_l} k(s - s_0) T(\sigma(s)) ds, \qquad (1)$$

where T is the input texture, $\sigma(s)$ is a parameterized streamline through x_0 ($x_0 = \sigma(s_0)$) and k describes the convolution kernel. s_l specifies the length of the streamline segment used in the filter operation. The texture values along the streamline segment σ (s), $(s_0-s_l) \leq s \leq (s_0+s_l)$, are weighted with the corresponding kernel values $k(s-s_0)$ and are accumulated to give the intensity $I(x_0)$ at position x_0. Various kernel functions k() can be used in the filter operation. Animation of the flow field can be achieved by using a ramp-like convolution kernel and phase shifting the kernel in successive images. This gives the impression of flowing ripples, which also encode the orientation of the flow. Extensions and performance optimizations of LIC are investigated in [FoCo95, StHe95, ShJo96, InGr97, ShKa97].

Line Integral Convolution does not encode the orientation of a flow field in still images. Oriented Line Integral Convolution (OLIC) [WeGr97a] overcomes this disadvantage. There are two main differences between LIC and OLIC: First LIC uses typically textures with much higher spatial frequency than OLIC. OLIC uses sparse textures, which consist of randomly distributed, distinct spots. OLIC can be thought of as a set of ink droplets which are smeared by the vector field over an underlying sheet of paper. Ideally the smeared ink droplets are so far apart from each other that they do not overlap. The second difference between LIC and OLIC is that OLIC uses a ramp-like convolution kernel k(). Such a kernel produces streamlets with varying intensity along the trace. The sparse texture and the varying intensity of the streamlets make it possible to recognize the orientation of the underlying flow field. In figure 1 the difference between LIC and OLIC is illustrated. Figure 1(a) shows the LIC image of a circular flow. In this image it is not recognizable if the flow is in clockwise or counterclockwise orientation. Figure 1(b) shows the OLIC image of a circular clockwise flow and figure 1(c) shows the OLIC image of a circular counterclockwise flow. The additional information in the OLIC image is gained at the expense of spatial resolution. To avoid the appearance of undesirable macroscopic patterns in the OLIC image, the starting points of the streamlets must be carefully selected. Positioning the starting points (i.e., ink droplets of the texture) on a regular grid would produce annoying macroscopic patterns. A jittered grid is a better choice. If the distance between the starting points of the streamlets is too large a lot of flow information is not depicted in the result image. On the other hand, if the starting points are too close together the overlapping of the streamlets might be too extensive. Finding an optimal droplet distribution in the input texture so that tight packing of streamlets results in the output image is discussed in [WeGr97b].

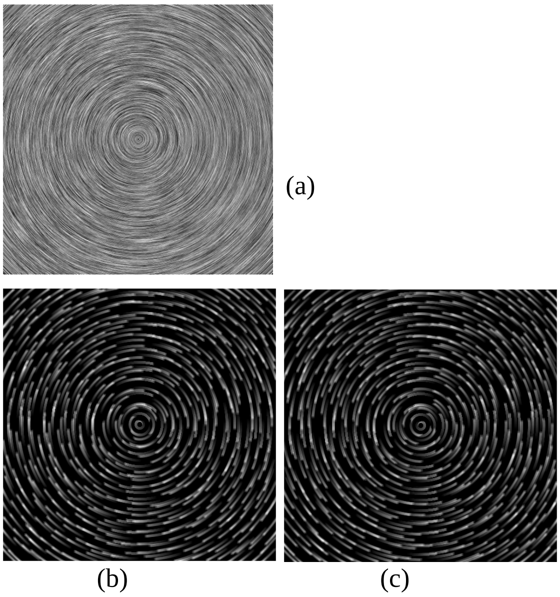


Figure 1: LIC image of circular flow (a), OLIC image with clockwise flow (b), OLIC image with counterclockwise flow (c) [WeGr97b]

The calculation of OLIC images is a rather time consuming task. Therefore Fast Rendering of Oriented Line Integral Convolution (FROLIC) was introduced in [WeGr97b]. FROLIC calculates an approximate solution to the exact convolution result of OLIC. In OLIC each droplet produces a trace with decreasing intensity from head to tail. FROLIC approximates the droplet trace by a sequence of disks with varying intensity. Each disk itself is drawn with a constant intensity. From head to tail the intensities of the disks are decreasing. If n disks are taken to approximate the streamlet, the intensity decreases in n discrete steps. The intensity of adjacent disks is decreasing to simulate the continuous ramp kernel of the OLIC method. Figure 2 shows the difference between droplet traces produced with OLIC and FROLIC. FROLIC allows a much faster calculation than OLIC, because drawing disks is done faster than doing costly convolution operations. Investigations show that FROLIC (without hardware supported rendering) is approximately two orders of magnitude faster than OLIC. This is discussed in detail in [WeGr97b].

Figure 2: Exact trace of a droplet with OLIC (a) and approximated trace of a droplet with FROLIC (b) [WeGr97b]

Animation of OLIC images is realized by simply phase shifting the convolution kernel in successive images. There are two possible ways for achieving animation with FROLIC. The first method phase shifts the convolution kernel the same way as is done to animate OLIC images. The second method uses color-table animation, with which color-table entries are shifted to achieve the impression of motion. A brief and theoretical description of color-table animation for FROLIC images is given in [WeGr97b]. Color-table animation has also been investigated to visualize three-dimensional flow fields [GeWi92].

This paper presents an in-depth investigation of color-table animation for FROLIC. In section 2, color-table animation and several features, like the simulation of various filter functions to avoid undesired effects during animation, are discussed. The FROLIC algorithm and color-table animation is implemented as a Visual C++ application. Implementation details and results are discussed in section 3. The application uses Mathematica to calculate and simulate dynamical systems. Finally in section 4 conclusions are given.

2 COLOR-TABLE ANIMATION OF FROLIC

As mentioned earlier a FROLIC streamlet consists of a set of disks with intensity decreasing from head to tail. All streamlets consist of the same number of disks. Within an image each streamlet is integrated over the same period of time. Therefore speed is encoded in the length of the streamlets. Streamlets are calculated by numerically integrating the underlying flow field. The calculation can be done either with Euler integration or with more elaborate Runge Kutta methods [PrFl88].

Color-table animation of FROLIC is based on the fact that in successive frames of the animation only the color of the disks changes but not their spatial position or shape. Each disk is assigned a short color-table index, which points to a specific entry in the color table. Available intensities or colors are stored in the color table itself. Color-table animation changes the entries of the color table instead of changing the corresponding image. This can be done much faster then changing the image itself. Figure 3 shows how animation of FROLIC images can be achieved with color-table animation.

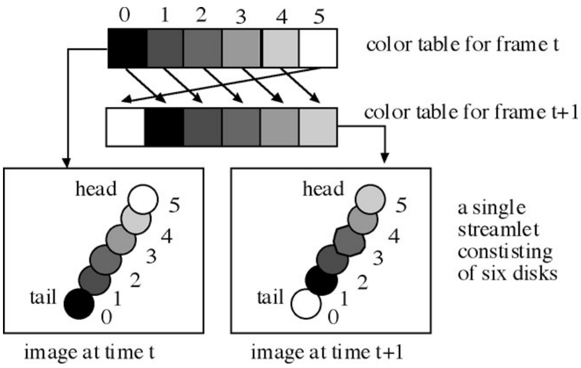


Figure 3: Color-table animation for FROLIC, two consecutive frames, n = 6

A color table ct with n entries (n assumed to be even, e.g., 256) contains values with increasing intensity. Intensity is assumed to be also in the range between 0 and 255. Therefore

$$ct[i] = \left\lfloor \frac{255 * i}{n-1} \right\rfloor \qquad 0 \le i \le n-1 \qquad (2)$$

where ct[i] is the color-table entry at position i (figure 4). If ct_{t-1} is the color table at time t-1 the color table ct_t for the following time step t is calculated recursively (figure 5):

$$\begin{aligned} ct_0[i] &= \left\lfloor \frac{255 * i}{n-1} \right\rfloor \qquad 0 \le i \le n-1 \\ ct_t[i] &= ct_{t-1}[(i-1) \bmod n] \end{aligned} \qquad (3)$$

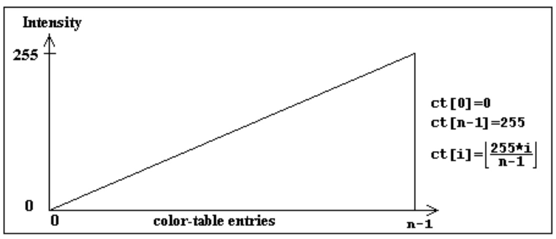


Figure 4: Color table, producing streamlets with orientation

Each streamlet consists of a set of disks, which are drawn with successive color-table entries. Adjacent disks are represented by adjacent color-table indices. The assignment of color-table indices to disks (i.e., drawing disks with corresponding color-table indices) is done only once for initialization and is not changed anymore. Cycling the color-table entries generates consecutive frames of the animation.

In the following we will discuss different compositions of color tables. Using a color table as in figure 4 produces images, which encode orientation of the flow in still images. During animation, however, two undesired effects are produced: the synchronization effect (also dealt with in [GeWi92]) and the pulsation effect. In section 2.1 the composition of a color table, which produces streamlets with orientation, is described and also how to overcome the synchronization and pulsation effects. Another composition of a color table, which does not encode the orientation of the streamlets, is described in section 2.2. Such a color table encodes the orientation of the flow only when the image is animated, but overcomes the pulsation effect.

2.1 COLOR TABLES FOR STREAMLETS WITH ORIENTATION IN STILL IMAGES

To show the orientation of the flow in still images the color table is assigned a ramp function, i.e., a linearly increasing function (see figure 4).

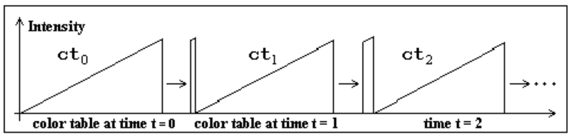


Figure 5: Consecutive color tables at time t = 0, t = 1 and t = 2

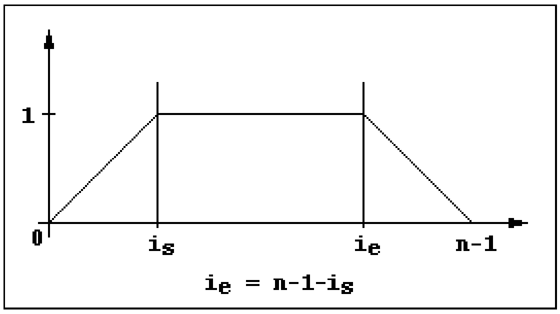


Figure 6: A simple filter f

Cycling a ramp function and starting each streamlet with the same color-table index produces two undesired effects in the resulting image. The first problem can be described as pulsation effect of the animated image. The human visible system is very sensitive to appearing and disappearing bright spots, therefore the disappearance of a bright disk at the head of a streamlet and the simultaneous appearance of a bright disk at the tail of the streamlet is very noticeable and disturbing to the eye. To solve this problem a simple filter f (see figure 6) can be used, which initially increases linearly, is constant in the middle portion, and finally decreases linearly. Filter f[i] is defined as follows:

$$f[i] = \begin{cases} \frac{1}{i_s} * i & 0 \le i < i_s \\ 1 & i_s \le i \le i_e \\ 1 - \frac{1}{n - i_e - 2} * (i - i_e - 1) & i_e < i \le n-1 \end{cases} \qquad (4)$$

The modified color table ct_t' of frame t is calculated by cycling the color table ct_{t-1} of frame t-1 and applying filter f (see figure 7):

$$ct_t'[i] = ct_t[i] * f[i] \qquad with$$
$$ct_t[i] = ct_{t-1}[(i-1) \bmod n], \; 0 \le i \le n-1 \qquad (5)$$

This approach generates a smooth intensity fade-in and fade-out at the head and tail of a streamlet.

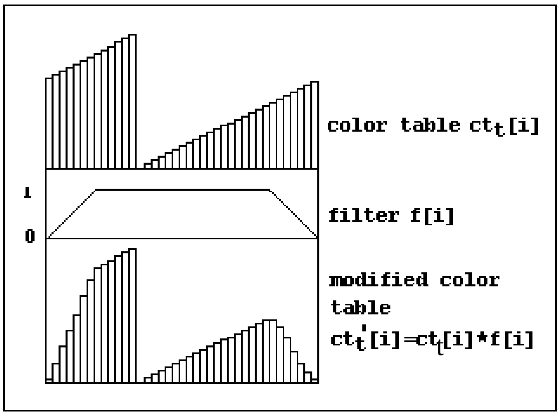


Figure 7: Color table and the filter with the modified color table

The second problem is called synchronization effect in the animated image. All streamlets begin with the same color-table index, i.e. the same intensity values. During animation, the disk with the highest intensity has the same relative position in all streamlets. Bright spots disappear at the head and reappear at the tail of all streamlets simultaneously. Up to now all streamlets are drawn with color-table indices decreasing from n-1 (head) to 0 (tail). Using for each streamlet a random initial offset r (the tail is assigned index r, the head is assigned index (n-1+r) mod n would avoid the synchronization effect. But this approach can not be used together with filter f to avoid also the pulsation effect. Due to the filter f a color-table index can not simultaneously represent a disk in the middle of one streamlet and a disk at the head or tail of another streamlet. To avoid both the pulsation and the synchronization effect the following approach is feasible. The color table is subdivided into m non-overlapping equal-sized subtables ct_0^u. Each subtable is assigned an initial offset and is cycled by its own. Assuming $n \bmod 2 = 0$ and $n \bmod m = 0$, then

$$ct_0[i] = ct_0^u[i] \quad with \; 0 \le u \le m-1, \;\; u * \frac{n}{m} \le i \le (u+1) * \frac{n}{m} - 1 \text{ and}$$
$$ct_0^u[i] = \frac{255}{\frac{n}{m} - 1} * \left(\left(i - u * \frac{n}{m} - u * \frac{n}{m^2} \right) \bmod \frac{n}{m} \right) \qquad (6)$$

The composition of such a color table is shown in figure 8. The number of color-table entries is n and the number of subtables is m. Cycling such a color table (by cycling each of the subtables) and applying filter f to each of the subtables is done according to the following formulas:

$$ct_t' = ct_t[i] * f[i], \quad \text{with}$$
$$ct_t[i] = ct_{t-1}\left[\left(i - u * \frac{n}{m} - 1 \right) \bmod \frac{n}{m} + u * \frac{n}{m} \right], \quad \text{and}$$
$$f[i] = \begin{cases} \frac{1}{i_s} * i \bmod \frac{n}{m} & 0 \le i \bmod \frac{n}{m} < i_s \\ 1 & i_s \le i \bmod \frac{n}{m} \le i_e \\ 1 - \frac{1}{\frac{n}{m} - i_e - 2} * \left(i \bmod \frac{n}{m} - i_e - 1 \right) & i_e < i \bmod \frac{n}{m} \le \frac{n}{m} - 1 \end{cases} \qquad (7)$$

There is a trade off between the number of subtables and the number of color-table entries belonging to a subtable. The more subtables the better the effect of reducing the synchronization effect. On the other hand, a certain number of different intensities per subtable is necessary to avoid too large intensity jumps along streamlets.

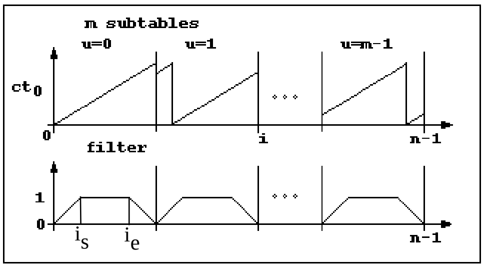


Figure 8: Color table divided into m non-overlapping subtables and corresponding filter

The perceived brightness of a disk depends on the intensity value and the area of the disk. The previous method attenuates the intensity at the head and the tail of a streamlet with filter f. Another approach to overcome the synchronization and pulsation effect modifies the size of the disks. Such a modified streamlet is shown in figure 9. The disk diameter is calculated by reducing the area of the disk by the underlying filter:

$$r[i] = r * \sqrt{f[i]}, \qquad (8)$$

where r[i] is the modified disk diameter and f is the filter value. Using such a filter reduces the impression of the appearance of a bright spot at the tail of the streamlet. To reduce the synchronization effect it is not necessary to divide the color table into subtables, because the intensities in the color table are not filtered with function f. Only one color table can be used and a random initial phase shift is used for each streamlet. If a color table has n entries, then n different phase shifts are possible. Therefore n different kinds of streamlets can be found in the image.

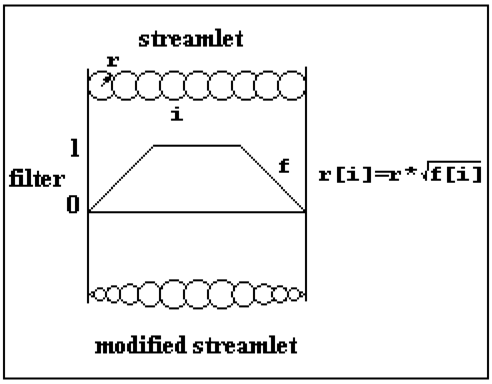


Figure 9: Streamlet with disks of different size

2.2 COLOR TABLES FOR STREAMLETS WITHOUT ORIENTATION IN STILL IMAGES

A method to reduce the pulsation effect of animated FROLIC images is to use a color-table composition as shown in figure 10. For a color table with n entries and one peak in the middle the intensity ct[i] is calculated by:

$$ct[i] = \begin{cases} \dfrac{255}{\frac{n}{2}} * i & 0 \le i < \dfrac{n}{2} \\ 255 - \dfrac{255}{\frac{n}{2}} * \left(i - \dfrac{n}{2} \right) & \dfrac{n}{2} \le i < n \end{cases} \qquad (9)$$

If the number of representable intensities on a screen is limited, it may make sense to use a color table which contains more than one peak. Using m evenly distributed peaks the intensity ct[i] is calculated by:

$$ct[i] = \begin{cases} \dfrac{255}{\frac{n}{2*m}} * \left(i \bmod \dfrac{n}{m} \right) & 0 \le i \bmod \dfrac{n}{m} < \dfrac{n}{2*m} \\ 255 - \dfrac{255}{\frac{n}{2*m}} * \left(i \bmod \dfrac{n}{m} - \dfrac{n}{2*m} \right) & \dfrac{n}{2*m} \le i \bmod \dfrac{n}{m} < \dfrac{n}{m} \end{cases} \qquad (10)$$

To overcome the synchronization effect, each streamlet is started with a random initial color-table index. As mentioned the disadvantage of the method is that flow orientation is not encoded in still images until the animation is started.

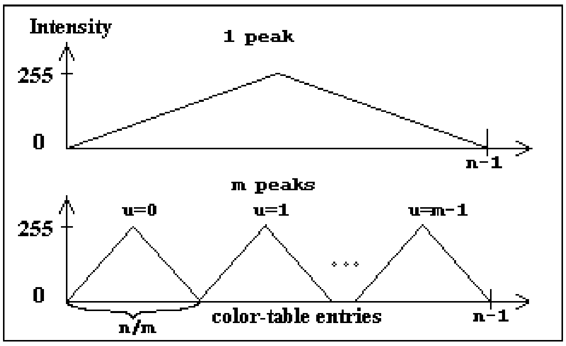


Figure 10: Color tables, producing streamlets without orientation

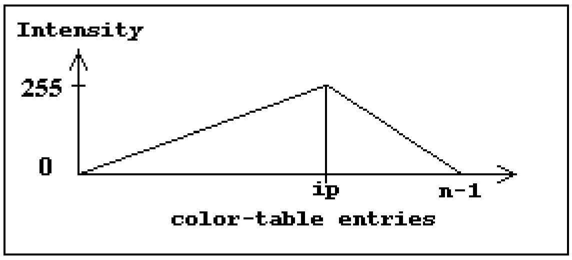


Figure 11: Color table which produces streamlets with orientation, and reduces the pulsation effect

Using a color table as shown in figure 11 reduces the disadvantage that flow orientation is not encoded in still images, but the pulsation effect is still reduced. To overcome the synchronization effect, each streamlet is assigned a random initial color-table index. The intensity ct[i] is calculated by:

$$ct[i] = \begin{cases} \dfrac{255 * i}{i_p} & 0 \le i < i_p \\ 255 - \dfrac{255 * (i - i_p)}{n - i_p} & i_p \le i < n \end{cases} \qquad (11)$$

3 IMPLEMENTATION AND RESULTS

In this section an implementation of FROLIC using Mathematica and Visual C++ [Schil96] under Windows NT is described. The calculation of the dynamical system is done with Mathematica [Wolf97] and the visualization is performed by the Visual C++ application. Both, the Mathematica application and the Visual C++ application can be accessed at http://www.cg.tuwien.ac.at/research/vis/dynsys/ct/. Figure 12 shows the data flow from Mathematica to the Visual C++ application. A Mathematica Notebook, which is a document that handles interaction between the user and Mathematica, is used to specify the dynamical system (set of differential equations). A module written in

Mathematica numerically integrates the dynamical system and writes the result on a file. The result is given as numerical approximations of streamlets of the underlying dynamical system. Using Mathematica for calculating the dynamical system has the big advantage of utilizing a powerful formula parser. The Visual C++ application reads the result file and visualizes the vector field.

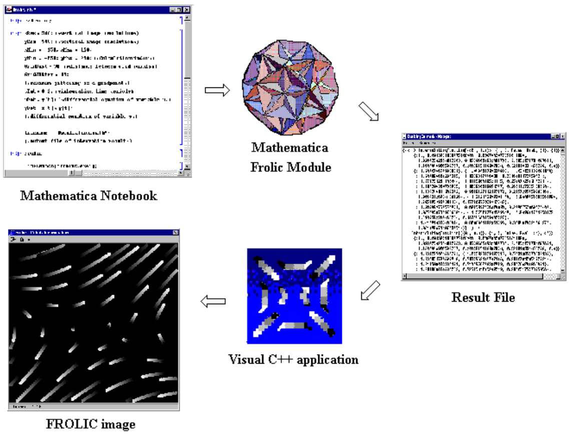


Figure 12: Data flow from Mathematica to the Visual C++ application

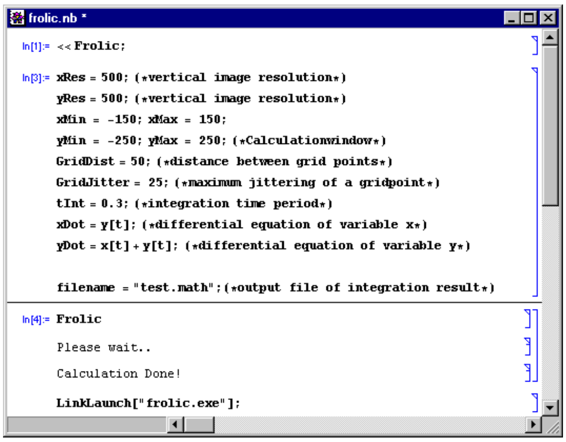


Figure 13: Mathematica Notebook for the specification of a dynamical system

A Mathematica Notebook is shown in figure 13. This Notebook can be used to calculate two-dimensional dynamical systems. Before using the Mathematica module, it must be loaded into the Mathematica workspace. The command “<<Modulename” reads in a particular Mathematica package (e.g., Frolic). After successfully loading the module the parameters, which specify the dynamical system, can be altered. The following parameters can be modified: *xRes* and *yRes* determine the vertical and horizontal resolution of the result image. *xMin*, *xMax*, *yMin* and *yMax* determine the calculation-window, i.e, the area of phase space where the dynamical system must be calculated. *GridDist* is the distance between grid points, *GridJitter* determines the maximum jittering of a grid point. *tInt* is the integration period and determines the length of the streamlets. *xDot* and *yDot* are the differential equations of variable x and variable y for the independent variable t (time). Finally *filename* is the name of the output file which contains the integration result of the dynamical system. After the parameters are specified, “functionname” (e.g, Frolic) starts the calculation. With “LinkLaunch[“frolic.exe”];” the Visual C++ application can be started within Mathematica.

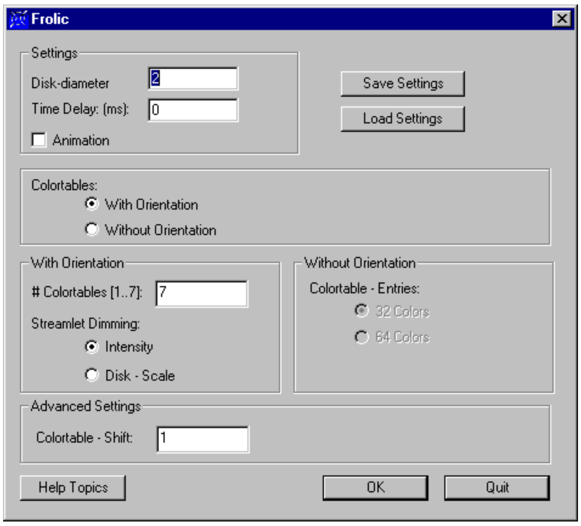


Figure 14: Dialog of the Visual C++ application

The Visual C++ application allows the visualization and animation of the dynamical system previously calculated with Mathematica. The result file, which holds the data for the vector field, is read in and the streamlets are drawn. Various settings can be chosen for the visualization of vector fields. The dialog of the Visual C++ application is shown in figure 14. The *Disk-Diameter* determines the thickness of the streamlets. Two different color tables can be chosen and are described in detail in section 2: a color table producing streamlets with orientation and a color table producing streamlets without orientation. If a color table, which produces streamlets with orientation, is selected, the number of subtables can be chosen. Using such a color table and one subtable produces useful still images. Figure 15(a) shows such an image. Animating this vector field would produce the synchronization effect, because all streamlets begin with the same color-table index, i.e., the same intensity values as described earlier in section 2. To avoid the synchronization effect during animation a random offset (using more than one subtable) is required. Figure 15(b) uses 7 subtables within a color table. Using more than one subtable produces still images, that do not show the orientation very well. For each streamlet in figure 15(a), the disk with the highest intensity has the same relative position, whereas seven different random phase shifts, producing seven different types of streamlets, are given in figure 15(b). Figure 16(a) shows a FROLIC image with streamlets that do not show the orientation of the flow. Figure 16(b) shows the same dynamical system, but orientation is encoded by using a color table which produces streamlets with orientation.

The speed of the animation is adjustable in two ways. Firstly, with *Time Delay* and secondly with increasing the *Colortable-Shift*. A timer is a clock application that notifies an application at regular intervals. *Time Delay* specifies the time in milliseconds after which a timer message is sent periodically. Each timer message causes a color-table shift. The smaller the *Time Delay* the faster the animation. The speed of the animation is limited by the calculation overhead. To increase the speed of the animation, it is possible to increase the step size, with which the color-table entries are shifted. To avoid the pulsation effect, it is possible to choose between two different methods. Scaling the intensity of the color-table entries with a filter and scaling the disk-diameter of the streamlets. Scaling the intensities of the color-table entries is done before the system palette is updated and slows down the animation. Scaling the disk-diameter is done as an initial step, when the streamlets are drawn. Figure 17(a) shows a FROLIC image with streamlets using scaled disks. The FROLIC image in figure 17(b) shows streamlets consisting of equal-sized disks.

4 CONCLUSION

Fast Oriented Line Integral Convolution (FROLIC) illustrates two-dimensional flow fields by approximating a streamlet through a set of disks with varying intensity. This paper describes various aspects of color-table animation for FROLIC images. Color-table animation is a very efficient algorithm. Each disk of the streamlets is assigned a short color-table index, which points to a specific entry in the color table. Various different compositions of color tables are discussed: color tables producing streamlets with orientation and color tables producing streamlets without orientation. Two undesired effects, the pulsation and the synchronization effect, are discussed and various techniques to reduce the two effects are described. FROLIC color-table animation is implemented as a Visual C++ application, whereby the calculation of the dynamical system is performed with Mathematica. Mathematica has the advantage that it offers a powerful formula parser and it offers a great flexibility concerning the specification of dynamical systems.

REFERENCES

[CaLe93] B. Cabral, C. Leedom, “Imaging Vector Fields Using Line Integral Convolution”; Computer Graphics Proceedings ‘93, ACM SIGGRAPH, 1993, pp. 263-270.

[FoCo95] L. K. Forssell and S. D. Cohen. “Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows”. IEEE Transaction on Visualization and Computer Graphics, 1(2): 133-141, June 1995.

[GeWi92] A. VanGelder, J. Wilhelms, “Interactive Animated Visualization of Flow Fields”, ACM Workshop on Volume Visualization, pp. 47-54, ACM 1992.

[InGr97] V. Interrante and Ch. Grosch. “Strategies for effectively visualizing 3D flow with volume lic”. In IEEE Visualization ’97 Proceedings, pages 421-424. IEEE, October 1997.

[PrFl88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. “Numerical Recipes in C”. Cambridge University Press, 1988.

[PoWa93] F. H. Post, T. Walsum, “Fluid Flow Visualization”, In H. Hagen, et. al. (eds) Focus on Scientific Visualization, Springer, 1993, pp. 1-40.

[Schil96] Schildt, Herbert, „MFC – Programming from the Ground UP“: Version 4, Osborne McGraw-Hill, 1996.

[ShJo96] H.-W. Shen, Ch. R. Johnson, and K.-L. Ma. “Visualizing vector fields using line integral convolution and dye advection”. In 1996 Volume Visualization Symposium, pp. 63-70. IEEE, October 1996.

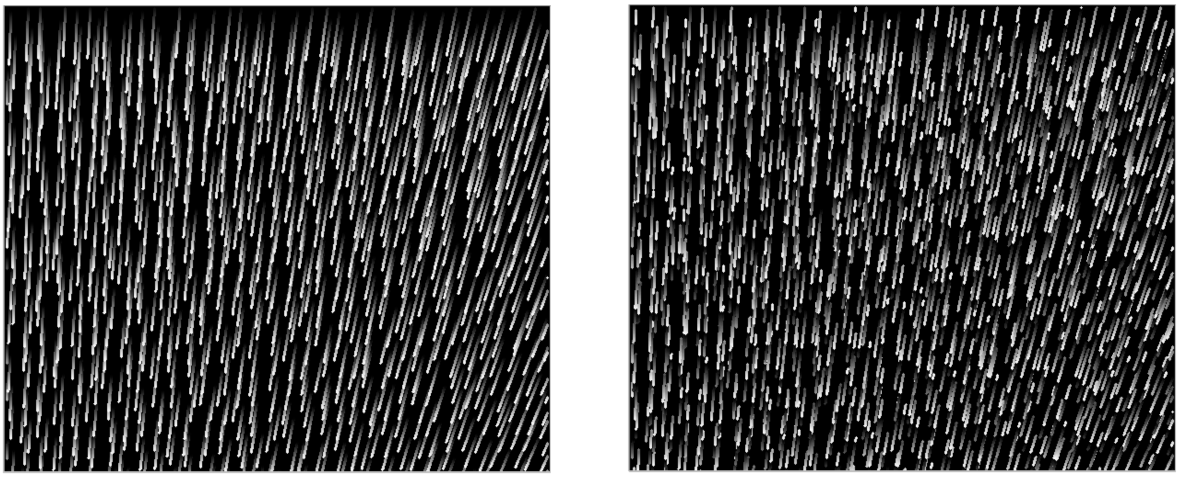
[ShKa97] H.-W. Shen and D. L. Kao. UFLIC: “A line integral convolution algorithm for visualizing unsteady flows”. In IEEE Visualization ’97 Proceedings, pp 317-322. IEEE Computer Society, October 1997.

[StHe95] D. Stalling, H.-C. Hege, “Fast and Resolution Independent Line Integral Convolution”, Computer Graphics Proceedings ‘95, ACM SIGGRAPH, 1995, pp. 249-256.

[WeGr97a] R. Wegenkittl, E. Gröller, W. Purgathofer, “Animating Flowfields: Rendering of Oriented Line Integral Convolution”. Proceedings of Computer Animation ’97. IEEE Computer Society, pp. 15-21.

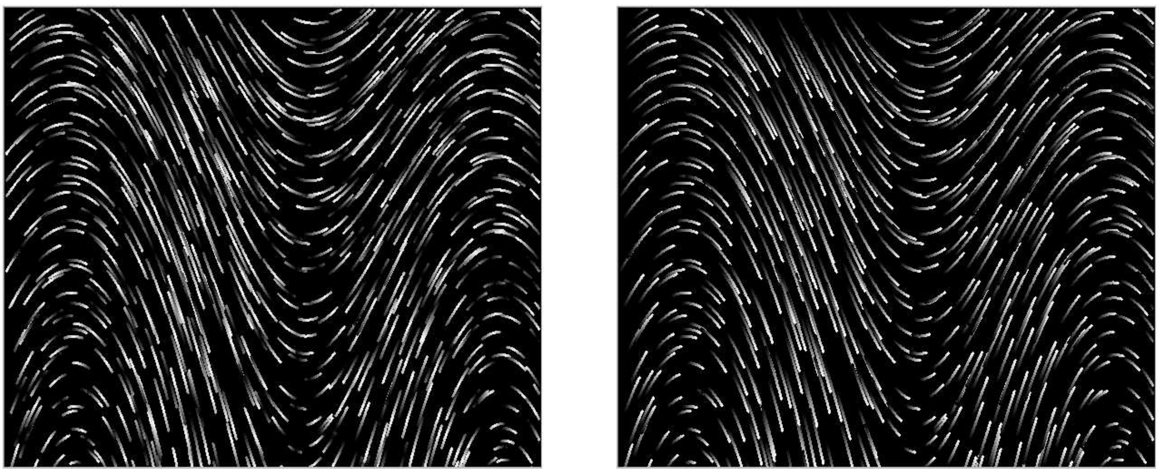
[WeGr97b] R. Wegenkittl, E. Gröller, “Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet”, IEEE Visualization ’97 Proceedings, pp. 119-125.

[Wolf97] Wolfram, Stephen: „Das Mathematica Buch“: „Mathematica Version 3 / Stephen Wolfram“. 5th edition, Addison-Wesely-Longman, 1997



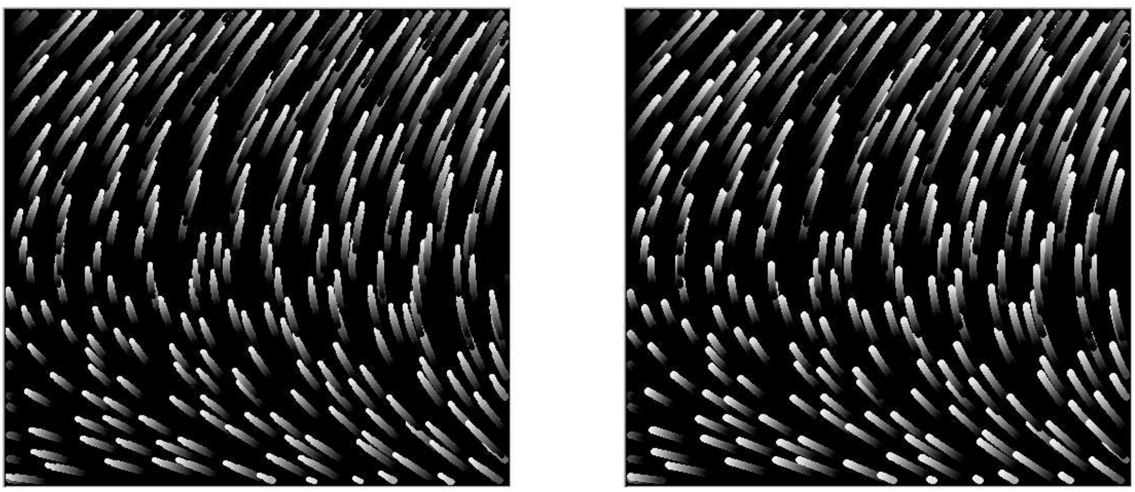
(a) (b)

Figure 15: FROLIC image with one subtable (a) and with seven (b) subtables per color table



(a) (b)

Figure 16: FROLIC image and color table producing streamlets without (a) and with (b) orientation



(a) (b)

Figure 17: FROLIC image with streamlets with scaled disks (a) and with equal-sized disks (b)