

HANS-CHRISTIAN HEGE AND DETLEV STALLING

# **Fast LIC with Higher Order Filter Kernels**

# Fast LIC with Piecewise Polynomial Filter Kernels

Hans-Christian Hege and Detlev Stalling

## Abstract

Line integral convolution (LIC) has become a well-known and popular method for visualizing vector fields. The method works by convolving a random input texture along the integral curves of the vector field. In order to accelerate image synthesis significantly, an efficient algorithm has been proposed that utilizes pixel coherence in field line direction. This algorithm, called “fast LIC”, originally was restricted to simple box-type filter kernels.

Here we describe a generalization of fast LIC for piecewise polynomial filter kernels. Expanding the filter kernels in terms of truncated power functions allows us to exploit a certain convolution theorem. The convolution integral is expressed as a linear combination of repeated integrals (or repeated sums in the discrete case). Compared to the original algorithm the additional expense for using higher order filter kernels, e.g. of B-spline type, is very low. Such filter kernels produce smoother, less noisier results than a box filter. This is evident from visual investigation, as well as from analysis of pixel correlations. Thus, our method represents a useful extension of the fast LIC algorithm for the creation of high-quality LIC images.

## 1 Introduction

Line Integral Convolution (LIC), introduced by Cabral and Leedom [5], is a particularly powerful and elegant method for synthesizing directional textures. Such textures are useful in visualization and computer art. The algorithm needs a texture and a vector field as input. The output image is computed by convolving the texture along the integral curves of the vector field. This causes anisotropic correlations of pixel intensities: the values are much higher correlated along individual integral curves than in directions perpendicular to field lines. The resulting image thereby clearly depicts the directional structure of the vector field.

The LIC technique can be used either to visualize a vector field or to impress a directional texture to an arbitrary image. In the first case inputs are the vector field one aims to visualize and a noisy, fairly arbitrary texture. In the second case the inputs are the image one wants to modify and some artificial vector field. This field may be generated, e.g., by taking the (rotated) gradient of the smoothed input image.

Elegance and usefulness of the LIC technique inspired many researchers to create variations and extensions of the original algorithm. Since the original algorithm is computationally rather expensive, a much faster algorithm has been developed which provides almost interactive speed [14]. Due to its significant lower computational complexity we call it “fast LIC”. Another feature of this algorithm is that smooth zooms can be produced. This can be utilized to visualize details of vector fields. Furthermore texture animations with constant or spatially varying velocity can be created, in order to portray orientation and strength of the vector field in an intuitive manner. For interactive exploration of large vector fields further acceleration may be necessary. This can be achieved by parallelization. Designs of parallel algorithms for various kinds of computer architectures are presented in [4] and [15].

Another method, derived from LIC, for encoding field direction and orientation has been proposed in [17]. Here a low frequency input texture and a ramp like anisotropic convolution kernel are used. Several algorithmic approaches have been suggested for the generation of LIC images on curved surfaces [8, 9, 1, 16, 12]. In ref. [10] it is shown how surface shapes in volume data can be illustrated using principal direction-driven 3D LIC. By integrating the LIC algorithm into direct volume rendering, dye advection – as used in experimental flow visualization – has been simulated [13]. Strategies for effectively portraying 3D flow using volume line integral convolution are discussed in [11]. An overview on the current status of LIC algorithms and applications of LIC is given in [3].

In this paper we generalize the fastLIC algorithm [14] for use of piecewise polynomial kernels. We employ the new algorithm not only in the plane, but create LIC textures also on arbitrary surfaces, extending our work [1]. Furthermore, we investigate whether the use of higher order filters pays visually. This is done by analyzing the differences between LIC images being generated with different kernels. The examinations are supplemented by a statistical analysis of LIC images.

The material of this paper has been organized as follows. After an introduction in Sect. 2, providing the mathematical background of LIC, a suitable class of filter kernels is introduced, as well as a means for fast convolution (Sect. 3). The fast LIC algorithm for polynomial filters is described in Sect. 4. In the last section we provide a statistical analysis of LIC images based on white noise input textures.

## 2 Line Integral Convolution

Line integral convolution may be performed in flat and curved space of arbitrary dimension. For simplicity we describe the algorithm for the flat 2D case. It will become obvious that the techniques and results can simply be generalized for higher dimensional and curved spaces. In fact we implemented the algorithm also for vector fields on curved surfaces.

## 2.1 Integral Curves

We are working on a domain  $\Omega \subset \mathbb{R}^2$  with some vector field  $\mathbf{v} : \Omega \rightarrow \mathbb{R}^2$ . Let  $\boldsymbol{\tau} : [t_0, t_1] \rightarrow \mathbb{R}^2$  denote the integral curves<sup>1</sup>

$$\frac{d}{dt} \boldsymbol{\tau}(t) = \mathbf{v}(\boldsymbol{\tau}(t)) \quad (1)$$

of the field  $\mathbf{v}$  for initial conditions  $\boldsymbol{\tau}(t_0) = \mathbf{x}_0$ ,  $\mathbf{x}_0 \in \Omega$ . For simplicity we assume that the right hand side locally obeys a Lipschitz-condition such that for all  $\mathbf{x}_0 \in \Omega$  there is a unique solution. Furthermore, we assume that the field vanishes nowhere in  $\Omega$ , i.e. that no critical points exist. All integral curves  $\boldsymbol{\tau}(t)$  can then be reparametrized by arc-length  $s$ , where  $s(t) = \int_{t_0}^t v(t') dt'$ . Using  $s$  as parameter makes it particularly easy to step along a given field line with equi-distant steps. Both assumptions are taken to simplify the analysis. They are not imposed in our implementation, where special care is taken at critical and discontinuous points of the field  $\mathbf{v}$ . Applying the chain rule and the inverse function theorem, then using Eq. (1) and  $ds/dt = v$ , we obtain

$$\frac{d\boldsymbol{\tau}(t(s))}{ds} = \frac{d\boldsymbol{\tau}}{dt} \left( \frac{ds}{dt} \right)^{-1} = \frac{\mathbf{v}(\boldsymbol{\tau}(t(s)))}{v(\boldsymbol{\tau}(t(s)))}.$$

Introducing a new function  $\boldsymbol{\sigma}(s) := \boldsymbol{\tau}(t(s))$ , we get the alternative definition of integral curves

$$\frac{d}{ds} \boldsymbol{\sigma}(s) = \frac{\mathbf{v}(\boldsymbol{\sigma}(s))}{v(\boldsymbol{\sigma}(s))}, \quad (2)$$

with initial condition  $\boldsymbol{\sigma}(s_0) = \mathbf{x}_0$ . Note that the right hand side of Eq. (2) is just the normalized vector field.

## 2.2 Line Convolution

Now, in LIC an input texture  $T$  is convolved along the field lines  $\boldsymbol{\sigma}$  of a given vector field. The intensity at point  $\mathbf{x} = \boldsymbol{\sigma}(s)$  is defined as

$$I(\mathbf{x}) = \int_{s-L}^{s+L} T(\boldsymbol{\sigma}(s')) h(s-s') ds', \quad (3)$$

where  $h$  is an arbitrary filter, normalized to  $\int_{-L}^L h(s') ds' = 1$ . Employing filter kernels  $h$  with finite support  $\text{supp } h = [-L, L]$  and using the notation  $I_{\boldsymbol{\sigma}}(s) := I(\boldsymbol{\sigma}(s))$ , and  $T_{\boldsymbol{\sigma}}(s) := T(\boldsymbol{\sigma}(s))$ , we may also write

$$I_{\boldsymbol{\sigma}}(s) = \int_{-\infty}^{\infty} T_{\boldsymbol{\sigma}}(s') h(s-s') ds' = (T_{\boldsymbol{\sigma}} * h)(s), \quad (4)$$

where  $*$  means convolution.

In order to evaluate Eq. (4) at a point  $\mathbf{x} = \boldsymbol{\sigma}(s)$ ,  $\mathbf{x} \in \Omega$ , we need a curve segment centered at  $\mathbf{x}$  and extending a length  $L$  in both directions. For points

---

<sup>1</sup>We will use the terms ‘integral curve’, ‘field line’ and ‘stream line’ synonymously.

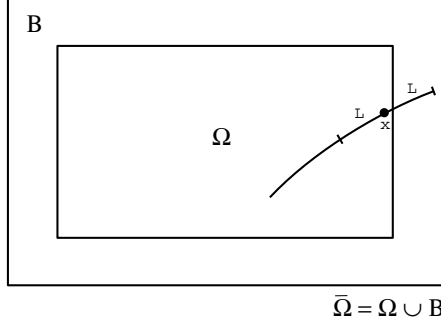


Figure 1: Enlarged domain  $\bar{\Omega} = \Omega \cup B$ , used for calculation of convolution integrals for all points  $x \in \Omega$ .

$\mathbf{x}$  near  $\partial\Omega$  such curve segments typically leave the domain  $\Omega$ . Therefore we pad domain  $\Omega$  by a sufficiently large boundary region  $B$  as shown in Fig. 1 and continue the vector field  $\mathbf{v}$  arbitrarily but smoothly into region  $B$ . This allows us to calculate integral curves of sufficient lengths for all points  $\mathbf{x} \in \Omega$ . The input texture is also defined on the extended domain  $\bar{\Omega} = \Omega \cup B$ , in order to perform convolutions along these line segments. The output image lives on  $\Omega$ .

### 2.3 Raster Images

The domains  $\bar{\Omega}$  and  $\Omega$  are partitioned into sets  $\{\bar{\omega}_i\}$  and  $\{\omega_j\}$  of rectangular pixels. The input texture  $T$  is a predefined raster image or is given procedurally. In either case it is a piecewise constant function  $T : \bar{\Omega} \rightarrow \mathbb{R}$ , assigning a texture value  $T_{\bar{\omega}_i}$  to each pixel  $\bar{\omega}_i$ . The output is also a raster image<sup>2</sup>, i.e. a piecewise constant function  $O : \Omega \rightarrow \mathbb{N}_0$ , assigning an integer grey value  $O_{\omega_i}$  to each pixel  $\omega_i \subset \Omega$ .

First, let us focus on the computation of an intensity value  $I$  for a particular point  $\mathbf{x} = \boldsymbol{\sigma}(s)$  on a field line, regarding the fact that  $T$  is a raster image. According to Eq. (3) all texture values along the curve segment  $\boldsymbol{\sigma}[s-L, s+L]$  contribute to  $I(\mathbf{x})$ . Assume that the curve segment  $\boldsymbol{\sigma}[s-L, s+L]$  passes  $n$  pixels  $\omega_{j_1}, \dots, \omega_{j_n}$ , i.e. consists of  $n$  sub-segments  $\boldsymbol{\sigma}[s_{k-1}, s_k]$  each crossing one pixel  $\bar{\omega}_{j_k}$ . Here  $s_0 = s-L$  and  $s_n = s+L$  are the limits of the convolution integral, whereas the values  $s_1 \leq s_2 \leq \dots \leq s_{n-1}$  correspond to all intersections between the curve segment  $\boldsymbol{\sigma}[s-L, s+L]$  and the pixel boundaries<sup>3</sup>. The one-dimensional texture  $T(\boldsymbol{\sigma}(s'))$  on interval  $[s-L, s+L]$  then is a piecewise constant function with breakpoints  $s_1, s_2, \dots, s_{n-1}$ , that is  $T(\boldsymbol{\sigma}(s')) = T_{\bar{\omega}_{j_k}}$ , for  $s' \in [s_{k-1}, s_k)$  and  $k = 1, \dots, n$ . Therefore equation (3) becomes

$$I_{\boldsymbol{\sigma}}(s) = \sum_{k=1}^n T_{\bar{\omega}_{j_k}} \int_{s_{k-1}}^{s_k} h(s-s') ds'. \quad (5)$$

<sup>2</sup>Note, that the pixels  $\bar{\omega}$  of the input texture contained in  $\Omega$  may differ in size and location from the pixels  $\omega$  of the output image.

<sup>3</sup>For this discussion we may disregard degenerate cases where the intersections are not single points.

Given all breakpoints  $s_k$  and a specific kernel  $h$ , e.g. a polynomial one, the intensity values  $I_\sigma(s)$  can be easily calculated.

The output image must also be a raster image. Therefore a representative intensity value has to be obtained for each pixel  $\omega_i$ . Ideally one should compute a spatial average,

$$O_{\omega_i} = \int_{\omega_i} I(\mathbf{x}) d\mu \quad (6)$$

and, for quantization, cast the result value to the next integer. Here  $d\mu$  defines a suitable measure with  $\int_{\omega_i} d\mu = 1$ .

For performance reasons both quantities,  $I_\sigma(s)$  and  $O_{\omega_i}$  will be computed only approximately, as shown in the next section.

## 2.4 Sampling and Aliasing

In a practical implementation the calculation of convolution integrals must be extremely fast, since for each pixel  $\omega$  of the output image several intensity values have to be calculated. Computing the intersections of the curve segment  $\sigma[s-L, s+L]$  with the pixel boundaries and then evaluating Eq. (5) would be computationally too expensive. Trading some accuracy, the intensity  $I_\sigma(s)$  can instead be approximated by a Riemann sum. Using the notation  $T_l = T_\sigma(l\Delta s)$  and  $h_l = h(l\Delta s)$  we get from Eq. (3) a discrete approximation

$$\hat{I}_\sigma(s) = \frac{1}{2m+1} \sum_{k=-m}^m T_\sigma(s+k\Delta s) h(-k\Delta s) = \frac{1}{2m+1} \sum_{k=-m}^m T_{j-k} h_k \quad (7)$$

where  $m = L/\Delta s$ ,  $j = s/\Delta s$ , and  $k = (s-s')/\Delta s$ . Algorithmically this means that the integrand is sampled pointwise, e.g. starting at point  $\sigma(j\Delta s)$  and then stepping along curve  $\sigma$  with fixed step size  $\Delta s$  in both directions. Of course, the results computed with approximation (7) in general are correct only in the limit  $\Delta s' \rightarrow 0$ . Comparing the approximate expression with Eq. (5), the error sources for finite  $\Delta s$  are obvious: First, not all texture values  $T_{\bar{\omega}_k}$  with  $k = 1, \dots, n$  are taken into account, and second, the integrals in Eq. (5) are approximated only roughly. Rephrased in computer graphics language: using approximation (7), two signals are pointwise sampled, the 1D texture along a curve segment and the terms contributing to the integrals in Eq. (5). Since the distance of two subsequent pixel boundaries  $s_{k-1}$  and  $s_k$  may be arbitrarily small, the Nyquist frequency of the 1D random texture is unbounded – even if  $T$  would be continuous – and aliasing effects are in principle unavoidable. This is moderated by the fact that pixels being missed during sampling typically would receive only small weights according to Eq. (5).

Practical experience shows that using the approximation with a step size  $\Delta s$  of about a third or half a texture pixel width usually delivers visually pleasing results. Hence, this approximation is sufficient for practical purposes.

The pixel averages defined by Eq. (6) are approximated also by discrete sums

$$O_{\omega_i} \approx \frac{1}{n_c} \sum_{j=1}^{n_c} \alpha_j \hat{I}(\mathbf{x}_j) \quad \text{where } \mathbf{x}_j \in \omega_i \quad (8)$$

with finite  $n_c$ . The sampling locations  $\mathbf{x}_j$  and weights  $\alpha_j$  can be chosen according to some anti-aliasing scheme. As will be discussed later, our LIC algorithm is able to produce multiple samples per pixel in a natural way. Although there is only limited control about the exact locations of these samples within a pixel (or optionally a subpixel), by adjusting a parameter  $min_{hit}$  the total number of samples per pixel (or subpixel) is guaranteed to be  $n_c \geq min_{hit}$ . It turns out that by averaging all samples with equal weights  $1/n_c$  results of sufficient high quality are obtained.

A field line  $\sigma$  typically hits many pixels  $\omega_i$  of the output image and therefore can be used to compute intensity values for a multitude of pixels. The fast LIC algorithm [14] avoids the redundancies during intensity calculations for neighbored points of a field line by relating these values mathematically and using such a discretization. For box filters the coefficients  $h_k$  are constant and the relation between  $\hat{I}_\sigma(s)$  and  $\hat{I}_\sigma(s + l\Delta s)$  is obvious. Therefore, calculating the LIC integral only at those locations which serve as sample points during integration, a fast algorithm can be designed. For non-constant kernels Eq. (7) is still valid, but it is unfavorable to use it directly for calculating many intensity values along a field line.

In Sec. 3 and 4 and we will discuss how discrete convolutions can be performed very efficiently for certain types of filter kernels and how this procedure can be combined with the fast LIC concept.

### 3 Convolution Theorem, Filter Kernels

The original fast LIC algorithm utilizes the fact that for box-type filter kernels the convolution integrals are just differences of sums which can be easily updated while stepping along a field line. This fact is a special case of a certain convolution theorem stated below.

Piecewise polynomials provide a rather general function class that allows us to represent a wide variety of kernel shapes. Therefore, we will mainly aim at using this type of functions as filter kernels. We will define piecewise polynomial functions formally, introduce a basis of the corresponding linear space (following de Boor [2]), and then apply the convolution theorem to a linear combinations of these basis functions.

#### 3.1 A Convolution Theorem

The single important theorem which our work is based on is that the convolution of two functions  $f$  and  $h$ , where  $h$  has finite support, is equal to the convolution of the integral  $F$  of  $f$  and the derivative  $h'$  of  $h$ , i.e.

$$f * h(x) = \int_{-\infty}^{\infty} f(y) h(x - y) dy = \int_{-\infty}^{\infty} F(y) h'(x - y) dy = F * h'(x) \quad (9)$$

for all  $x \in \mathbb{R}$ . To show this we consider a definite integral with finite bounds first, apply integration by parts, let the integration bounds move to infinity, and

then use the fact that the filter kernel  $h$  has finite support:

$$\int_{-z}^z F(y) h'(x-y) dy = F(y) h(x-y) \Big|_{y=-z}^{y=z} + \int_{-z}^z f(y) h(x-y) dy \xrightarrow{z \rightarrow \infty} f * h(x)$$

for every finite  $x$ . Assuming that  $f$  is (at least)  $n$  times integrable and  $h$  is (at least)  $n$  times differentiable, we may apply Eq. (9)  $n$  times repeatedly. In order to simplify the notation, we denote the  $n$ th integral of  $f$  by  $F_n$ ,

$$F_n(x) = \int_{-\infty}^x dx_n \int_{-\infty}^{x_n} dx_{n-1} \dots \int_{-\infty}^{x_2} dx_1 f(x_1) \quad \text{for } n = 1, 2, 3, \dots \quad (10)$$

Then we find the relation

$$f * h = F_n * h^{(n)}. \quad (11)$$

This is valid also if the  $n$ -th derivative  $h^{(n)}$  is a delta distribution. Note that if  $h^{(n)}(x) = c \delta(x - \xi)$  then

$$f * h(x) = c \int_{-\infty}^{\infty} F_n(y) \delta(x - \xi - y) dy = c F_n(x - \xi). \quad (12)$$

Hence, convolution with a kernel  $h$  can be calculated by repeated integration if some derivative of  $h$  is a linear combination of delta distributions. A trivial case is a box filter, where  $h^{(1)}$  consists of two delta functions and the convolution amounts to a difference of two integrals.

### 3.2 Piecewise Polynomial Functions

Given a strictly increasing sequence  $\xi := (\xi_i)_{i=1 \dots l}$  of knots  $\xi_i \in \mathbb{R}$  and polynomials  $P_i$ ,  $i = 1 \dots l$ , each of order  $k$  (i.e., of degree  $< k$ ), then we define a *piecewise polynomial function of order  $k$*  by

$$f(x) := \begin{cases} 0, & x < \xi_1 \\ P_i(x), & \xi_i \leq x < \xi_{i+1}, \quad i = 1 \dots l-1 \\ P_l(x), & x \geq \xi_l. \end{cases} \quad (13)$$

The function and its derivatives may or may not be continuous at the knots  $\xi_i$ , as illustrated in Fig. 2.

It is easy to see that the set of piecewise polynomial functions of order  $k$  defined for a fixed knot sequence  $\xi$  generates a linear space. We will call this space  $\mathbb{P}_{k,\xi}$ .

For the time being we allow the function  $f$  to take non-vanishing values right from the last knot  $\xi_l$ . Later, when using the piecewise polynomial functions as filter kernels, we require the right-most polynomial  $P_l$  (dashed in Fig. 2) to be zero, since the kernels must have finite support.



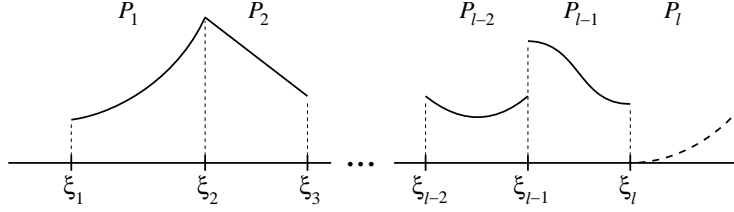


Figure 2: A piecewise polynomial function, defined by a set of  $l$  knots  $\xi_i$  and  $l$  polynomials  $P_i$ .

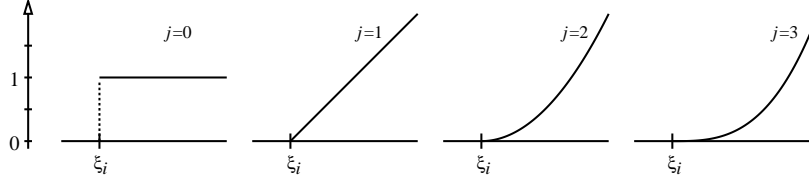


Figure 3: Elements of the truncated power basis  $\phi_{ij} = (x - \xi_i)_+^j / j!$ .

### 3.3 The Truncated Power Basis

Let us now introduce a basis for the space of piecewise polynomial functions. We will build this basis from so-called truncated power functions, defined by

$$(x)_+^r := \begin{cases} 0, & x < 0 \\ x^r, & x \geq 0 \end{cases} \quad (14)$$

for  $r \in \mathbb{R}_0$ . Using this notation, we define a double sequence of functions corresponding to a knot sequence  $\xi$  by

$$\phi_{ij}(x) = \frac{(x - \xi_i)_+^j}{j!} \quad \text{for } i = 1 \dots l \text{ and } j = 0 \dots k-1. \quad (15)$$

The  $\phi_{ij}$  are piecewise polynomial functions of order  $j+1$  with just one knot  $\xi_i$ . Obviously these functions are elements of  $\mathbb{P}_{k,\xi}$ . Note that this would not be the case if we had required  $P_l(x)$  to be zero in Eq. (13). Plots of various  $\phi_{ij}$  are shown in Fig. 3.

The derivatives of the functions  $\phi_{ij}$  obey

$$\frac{d}{dx} \phi_{ij} = \phi_{i,j-1}, \quad \text{for } j = 1 \dots k-1 \quad (16)$$

and

$$\frac{d^j}{dx^j} \phi_{ij} = \phi_{i,0}, \quad \text{for } j = 0 \dots k-1 \quad (17)$$

where  $\phi_{i,0}$  is a step function which jumps at  $\xi_i$  from 0 to 1.

We will now show that the set of functions  $\phi_{ij}$  is a basis of  $\mathbb{P}_{k,\xi}$ . Let us first look at the linear functionals  $\lambda_{ij}$  that take the difference of the  $j$ th derivative of  $f$  at a breakpoint  $\xi_i$ :

$$\lambda_{ij} f := \text{jump}_{\xi_i} f^{(j)} := f^{(j)}(\xi_i^+) - f^{(j)}(\xi_i^-), \quad (18)$$

with  $i = 1 \dots l$  and  $j = 0 \dots k-1$ . Applying  $\lambda_{ij}$  to  $\phi_{ij}$  and considering Eq. (17) yields

$$\lambda_{ij}\phi_{rs} = \text{jump}_{\xi_i} \frac{d^j}{dx^j} \frac{(x - \xi_r)_+^s}{s!} = \delta_{ir}\delta_{js}. \quad (19)$$

The expression vanishes always except if the knot indices  $i$  and  $r$  as well as the exponents  $j$  and  $s$  are equal. We can use this result to show that the  $\phi_{ij}$  are linear independent. From

$$\lambda_{ij} \sum c_{rs} \phi_{rs} = \sum c_{rs} \lambda_{ij} \phi_{rs} = \sum c_{rs} \delta_{ir} \delta_{js} = c_{ij} = 0. \quad (20)$$

it is obvious that  $\sum c_{ij}\phi_{ij} = 0$  implies  $c_{ij} = 0$ . The dimension of the space  $\mathbb{P}_{k,\xi}$  is  $kl$ , which is equal to the number of functions  $\phi_{ij}$ . Together with linear independence this shows that the  $\phi_{ij}$  really comprise a basis for piecewise polynomial functions. Consequently, every  $f \in \mathbb{P}_{k,\xi}$  has a unique representation of the form

$$f = \sum_{ij} (\lambda_{ij} f) \phi_{ij} = \sum_{ij} (\text{jump}_{\xi_i} f^{(j)}) \frac{(x - \xi_i)_+^j}{j!}. \quad (21)$$

For many applications a more suitable and numerically advantageous basis of  $\mathbb{P}_{k,\xi}$  is given by so-called  $B$ -splines [2, 7]. However, as we will see the truncated power basis allows us to rewrite convolution integrals in an elegant way, thus facilitating the design of a general fast LIC algorithm. Therefore we will expand all filter kernels, even  $B$ -splines, in the truncated power basis.

We note that differentiating Eq. (17) yields the following distributional relationship for the basis functions  $\phi_{ij}$  and all  $i$  and  $j$ :

$$\frac{d^{(j+1)}}{dx^{(j+1)}} \phi_{ij}(x) = \delta(x - \xi_i), \quad (22)$$

with  $\delta$  representing the Dirac delta measure.

### 3.4 Applying the Convolution Theorem

Now suppose the filter kernel is given as a piecewise polynomial function, i.e.  $h = \sum_{ij} c_{ij}\phi_{ij}$ . Differentiating  $\phi_{ij}$  according Eq. (22) until a delta distribution is obtained and applying the convolution theorem (11), the convolution integral may be written as

$$\begin{aligned} f * h(x) &= \int_{-\infty}^{\infty} f(y)h(x-y)dy = \sum_{ij} c_{ij} \int_{-\infty}^{\infty} f(y)\phi_{ij}(x-y)dy \\ &= \sum_{ij} c_{ij} \int_{-\infty}^{\infty} F_{j+1}(y)\delta(x - \xi_i - y)dy \\ &= \sum_{ij} c_{ij} F_{j+1}(x - \xi_i). \end{aligned} \quad (23)$$

Computing convolution integrals therefore amounts to calculating weighted averages of repeated integrals, or – after discretization – of repeated discrete sums.

### 3.5 Filter Kernels of $B$ -Spline Type

For line integral convolution the filter kernel  $h$  should be as simple as possible. Usually there is no reason why to consider unsymmetric kernels. Furthermore, in order to obtain smoother results, the kernel should decrease or even approach zero at its boundaries. (The notion of “smoothness” will be made more quantitative by an intensity correlation analysis in Sect. 5.2.)

A nice class of filters that fulfill these conditions are  $B$ -splines with uniform knot sequences and centered around the origin. The most simple element in this class is a box filter  $b_1$  with knots at  $-L$  and  $L$  and normalized to  $\int b_1 dx = 1$ . Higher order filters can be obtained by repeatedly convolving box filters. Convolution of two box filters we get a triangle filter  $b_2 = b_1 * b_1$ . In contrast to the box the triangle filter is continuous but still has a discontinuous derivative. Even smoother filters are obtained by convolving the triangle with a box again, and so on. In this way we get filter kernels of  $B$ -spline type.

*Theorem.* Let  $b_n$  be the  $n - 1$ th convolution of a box filter with itself.

(i) Function  $b_n$  is given by

$$b_n(x) = \frac{1}{(2L)^n} \sum_{i=0}^n (-1)^i \binom{n}{i} \frac{1}{(n-1)!} (x + (n-2i)L)_+^{n-1}. \quad (24)$$

(ii)  $b_n$  has support  $[-nL, nL]$ .

(iii)  $b_n$  is normalized to  $\int b_n dx = 1$ .

(iv) In the limit  $n \rightarrow \infty$  the functions  $b_n$  converge uniformly to a Gaussian.

*Proof.*

(i) We will show Eq. (24) by induction. For  $n = 1$  we obtain the box filter itself:

$$b_1 = \frac{1}{2L} \left( (x+L)_+^0 - (x-L)_+^0 \right).$$

Assuming that Eq. (24) is valid for  $b_n$  we get for  $b_{n+1}$ :

$$\begin{aligned} b_{n+1} &= b_n * b_1 \\ &= \frac{1}{(2L)^{n+1}} \sum_{i=0}^n (-1)^i \binom{n}{i} \int_{-\infty}^{\infty} \frac{(x + (n-2i)L)_+^{n-1}}{(n-1)!} \left( (x-y+L)_+^0 - (x-y-L)_+^0 \right) dy \\ &= \frac{1}{(2L)^{n+1}} \sum_{i=0}^n (-1)^i \binom{n}{i} \left( -\frac{(x+L+(n-2i)L)_+^n}{n!} + \frac{(x-L+(n-2i)L)_+^n}{n!} \right) \\ &= \frac{1}{(2L)^{n+1}} \sum_{i=0}^n (-1)^i \binom{n}{i} \left( -\frac{(x+(n+1-2(i+1))L)_+^n}{n!} + \frac{(x+(n+1-2i)L)_+^n}{n!} \right) \end{aligned}$$

Here we have used theorem Eq. (9) to replace the box filter by a delta distribution. Relabeling the index in the first term of the sum and using the equality  $\binom{n}{i-1} + \binom{n}{i} = \binom{n+1}{i}$  we obtain

$$b_{n+1} = \frac{1}{(2L)^{n+1}} \sum_{i=0}^{n+1} (-1)^i \binom{n+1}{i} \frac{(x + (n+1-2i)L)_+^n}{n!}$$

in accordance with Eq. (24).


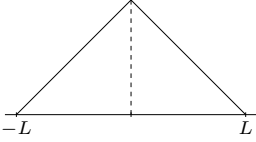
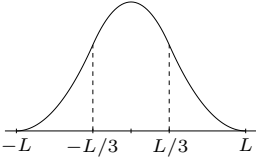
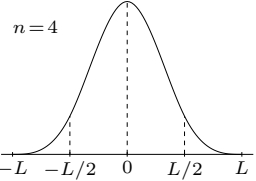
Filter kernel $h$	Definition	Convolution $f * h$
	$P_1 = \frac{1}{2L}$	$\frac{1}{2L} (F(x+L) - F(x-L))$
	$P_1 = \frac{1}{L^2}(x+L)$ $P_2 = \frac{1}{L^2}(L-x)$	$\frac{1}{L^2} (F_2(x+L) - 2F_2(x) + F_2(x-L))$
	$P_1 = \frac{27}{16L^3}(x+L)^2$ $P_2 = \frac{9}{8L} - \frac{27}{8L^3}x^2$ $P_3 = \frac{27}{16L^3}(L-x)^2$	$\frac{27}{8L^3} (F_3(x+L) - 3F_3(x+L/3) + 3F_3(x-L/3) + F_3(x-L))$
$n=4$ 	Convolution with a filter kernel of $n$ th-order $B$ -spline type:	$f * h(x) = \left(\frac{n}{2L}\right)^n \sum_{i=0}^n (-1)^i \binom{n}{i} F_n\left(x + \frac{n-2i}{n}L\right)$

Table 1: Filter kernels of  $B$ -spline type obtained by recursively convolving box filters. All filters  $h$  in this table have support  $[-L, L]$  and are normalized to one. Convolutions  $f * h$  are represented as finite linear combinations of repeated integrals of  $f$ .

(ii) All filters  $b_n$  are normalized since  $\int f dx = \int g dx = 1$  implies  $\int f * g dx = 1$ , and  $\int b_1 dx = 1$ .

(iii) Elementary inspection of Eq. (24) shows that the functions  $b_n$  have support  $[-nL, nL]$ .

(iv) A proof that the functions  $b_n$  converge in the limit  $n \rightarrow \infty$  uniformly to a Gaussian is given in Ref. [6]. ■

In order to achieve  $\text{supp } h = [-L, L]$  we re-scale  $x \rightarrow \frac{x}{n}$ . Furthermore we normalize the filters such that  $\int h(x) dx = 1$ . This yields the following sequence of bell-shaped filters of  $B$ -spline type:

$$\tilde{b}_n(x) = \frac{n}{(2L)^n} \sum_{i=0}^n (-1)^i \binom{n}{i} \phi_{ij}(x) \quad (25)$$

where  $\xi_i = \frac{2i-n}{n}L$  and  $i = 1 \dots n$ . In Table 1 the first few functions of this type are given explicitly in the form of Def. 13. The expressions for  $f * h$  contained in Table 1 result by applying relation (23).

The construction of  $B$ -splines from repeated convolutions has first been described by Curry and Schoenberg in 1947 [6]. In modern spline theory  $B$ -splines are usually defined in a more general context as divided differences on an arbitrary nondecreasing knot sequence, see e.g. [2].

A nice property of a  $n$ th order filter kernel of  $B$ -spline type is that all derivatives except of the  $n-1$ th one are continuous, i.e. the filter is built from exactly  $n+1$  truncated power functions of degree  $n-1$ . Consequently, the evaluation of  $f * h$  involves  $n$ th order integrals only. For general piecewise polynomial filter kernels usually other integrals have to be considered, too. The Catmull Rome Spline e.g. is given by

$$h(x) = 0.5(x+2)_+^3 - 2(x+1)_+^3 + 3(x)_+^3 - 2(x-1)_+^3 + 0.5(x-2)_+^3 - 0.5(x+2)_+^2 + (x+1)_+^2 - (x-1)_+^2 + 0.5(x-2)_+^2.$$

The occurrence of truncated power functions with different degrees makes the computation of  $f * h$  slightly more expensive.

## 4 A General Fast LIC Algorithm

From Eq. (23) we can build a fast LIC algorithm in a straight-forward way. The results of the previous section allow us, like in the original fast LIC algorithm, to exploit the coherence of convolution values along a single field line. After the integrals  $F_k$  along a field line have been computed, we can use this information to quickly obtain the convolution values for a whole bunch of samples on that line.

### 4.1 Discretization

For numerical evaluation the integrals  $F_k$  have to be approximated by sums. For case of simplicity we use left-handed Riemann sums. Then the value of the first integral  $F_1$  at locations  $s = k\Delta s$ , where  $-m \leq k \leq m$  and  $m = L/\Delta s$ , is given by

$$F(k\Delta s) = \int_0^{k\Delta s} f(x) dx \approx \Delta s \sum_{i=0}^{k-1} f(i\Delta s), \quad (26)$$

while the higher order integrals correspond to the repeated sums

$$F_n(k\Delta s) = \int_0^{k\Delta s} F_{n-1}(x) dx \approx \Delta s \sum_{i=0}^{k-1} F_{n-1}(i\Delta s). \quad (27)$$

Note, that we cannot approximate the integrals by centered Riemann sums. These would have to be evaluated at  $(i + \frac{1}{2})\Delta s$ , which destroys the recursive relation in Eq. (27). However, we well might use trapezoid rule. Then the individual contributions would be given by  $\frac{1}{2}(F(i\Delta s) + F((i+1)\Delta s))$ . Since only function evaluations at integer multiples of  $\Delta s$  occur, the higher-order sums can again be computed recursively. However, in practice no difference between left-handed Riemann sums and trapezoid rule is visible. Therefore we use the simpler formulas (26) and (27).

In detail, our new fast LIC algorithm for piecewise polynomial filter kernels requires the following steps. First, the filter kernel  $h$  has to be expressed in terms

of truncated power functions, i.e.  $h = \sum_{ij} c_{ij} \phi_{ij}$ . Then the repeated sums  $F_n$  at positions  $k\Delta s$  have to be calculated up to the required order. Finally, a discrete approximation  $\tilde{I}$  of the convolution integral Eq. (23) is computed using

$$\tilde{I}(k\Delta s) = \sum_{ij} c_{ij} F_j(k\Delta s - \xi_i). \quad (28)$$

## 4.2 Outline of the Algorithm

Eq. (28) describes how to compute multiple samples on a single field line. In order to compute a full LIC image, we proceed as in the original fast LIC algorithm [14]. For each pixel of the output image we maintain two variables, a hit count and an accumulation variable. We then traverse all pixels in some order. Whenever we encounter a pixel (or optionally subpixel) with a hit count smaller than some user-defined limit  $min_{hit}$ , we start a field line calculation and compute multiple samples on that line. All samples are added to the accumulation variable of the corresponding pixel and the hit count of that pixel is incremented. After all pixels have been processed, the final image is computed by normalizing the accumulation variables against the number of hits per pixel. For the original fast LIC algorithm sophisticated seed point selection strategies or methods for determining the optimal number of samples per field line segment have been developed [15]. These techniques can be applied to the new algorithm without any modification.

In Table 1 various filter kernels of  $B$ -spline type are shown. The convolution value for the most simple filter kernel – the box filter – is given as the difference of the first-order sums at just two different locations. After evaluating these sums, only two operations – one subtraction and one addition – are needed per pixel (except for the first one). This is exactly the same as in the original fast LIC algorithm. Convolutions with piecewise polynomial filters can be evaluated at very little additional costs, by calculating a linear combination of repeated sums  $F_n$  and updating these sums while stepping along a field line (compare Table 1).

## 4.3 Implementation Issues

Some pitfalls of this new LIC techniques should be mentioned, too. The first one is a numerical aspect. The repeated (integer) sums quickly take on very large values, which can cause overflows. In our implementation we use unsigned 32 bit integers to represent these sums. For example, assuming an average value of the 8-bit input texture of 128 we can evaluate the fifth-order sum only for about 100 samples. Then an overflow would occur. However, as will be shown in Sect. 6, visual pleasant results can already be obtained by using the hat or triangle filter, which involves second-order sums only.

Another point which one has to take account of, concerns the inner control points of higher-order filter kernels. For example the third-order filter shown in Tab. 1 has two knots at  $-L/3$  and  $L/3$ . In the discrete case it is necessary that the total number of samples used for approximating the kernel is divisible by 3. Using the nearest integer value of  $L/3$  doesn't work.

## 5 Statistical Analysis of LIC Images

### 5.1 LIC viewed as a Stochastic Process

The input texture  $T(\mathbf{x})$  for LIC can be arbitrarily chosen. In fact, every raster image can be used as input texture. Therefore, only global statements can be made about the statistical properties of LIC images. However, more detailed propositions are possible if the set of input textures is restricted to a class with certain statistical properties.

For vector field visualization the texture values  $T_{\omega_i} \in \mathbb{R}$  normally are chosen as a sequence of random variables  $T_i$ , i.e. as a (discrete) random process. We assume that the random variables  $T_i$  for distinct indices, i.e. for different texture pixels, are mutually independent. Then joint expectation values factorize,  $E(T_i T_j) = E(T_i) E(T_j)$  for any pair  $(i, j)$  with  $i \neq j$ , and the covariance therefore vanishes,  $\text{Cov}(T_i, T_j) = E(T_i T_j) - E(T_i) E(T_j) = 0$ . In accordance with the practice in scientific visualization let us assume that the random variables  $T_i$  are identically distributed with mean value

$$\mu_T = E(T) \quad (29)$$

and finite variance

$$\sigma_T^2 = \text{Var}(T) = E(T^2) - E(T) E(T). \quad (30)$$

We require  $\sigma_T^2 > 0$ , to exclude trivial cases like constant images. Since by definition  $\text{Cov}(T_i, T_i) = \text{Var}(T_i)$ , we get

$$\text{Cov}(T_i, T_j) = \sigma_T^2 \delta_{i,j} \quad (31)$$

for arbitrary pairs  $(i, j)$ .

Some of the following calculations are more elegant, if the pixel position  $i$  is viewed as a continuous variable  $\mathbf{x}$  (although a corresponding random process can physically not be realized). Then many functions have to be considered as distributions; Eq. (31) for example becomes

$$\text{Cov}(T_{\mathbf{x}}, T_{\mathbf{x}'}) = \sigma_T^2 \delta(\mathbf{x} - \mathbf{x}'). \quad (32)$$

A random process  $\{X_i\}$  is called (strictly) *stationary* if shifting in space has no effect on joint distributions, i.e. the distribution of  $X_{i_1}, \dots, X_{i_n}$  is the same as the joint distribution of  $X_{i_1+k}, \dots, X_{i_n+k}$ . Utilizing this fact for  $n = 2$  the autocovariance function

$$\gamma(i, j) := E\{[X_i - E(X_i)][X_j - E(X_j)]\} \quad (33)$$

obviously depends only on  $\tau = j - i$  and may be written as

$$\gamma(\tau) = E\{[X_t - \mu][X_{t+\tau} - \mu]\} = \text{Cov}[X_t, X_{t+\tau}]. \quad (34)$$

More useful is the standardized autocorrelation function

$$\rho(\tau) = \frac{\gamma(\tau)}{\gamma(0)}. \quad (35)$$

The random process  $\{T_i\}$  obviously is stationary. For the autocovariance function we then get

$$\gamma_T(\tau) = \sigma^2 \delta_{0,\tau} \quad (36)$$

and for the autocorrelation function

$$\rho_T(\tau) = \delta_{0,\tau}. \quad (37)$$

Random processes, constituted by a sequence of mutually independent and identically distributed random variables, are called a ‘purely random process’ or ‘white noise’. Equations (29)-(37) represent the simple statistical properties of the random input textures typically used in vector field visualization with LIC. The LIC intensity variables  $I$  depend on these random variables  $T$  and therefore constitute another random process. Its statistical attributes are more interesting. The most characteristic feature of LIC images are the anisotropic correlations: While the intensity values  $I$  are strongly correlated along field lines due to the 1D convolution, they are almost uncorrelated in perpendicular directions.

## 5.2 Statistical Properties of LIC Images Along Field Lines

In order to obtain statistical properties of  $I$  values along a field line, we start with a simplifying assumption: We assume that the texture input grid  $\{\bar{\omega}_i\}$  is fine enough such that distinct texture pixels  $\bar{\omega}_i$  are sampled during the convolution according to Eq. (7).

Is this assumption realistic? In practice one tries to choose a sampling distance along field lines of about one pixel width of the output image. Choosing identical locations, both for  $T$  and  $I$  samples, the condition is fulfilled to a rather good extend, if the spatial resolution of the input texture corresponds at least to that of the output image. In practice this condition usually is met.

We can therefore safely assume that the random variables  $T_{\sigma,j-k}$ , with  $k \in \{-m, \dots, m\}$ , being used to calculate a specific  $I$  variable according to Eq. (7) are independent. For simplicity we drop the index  $\sigma$  in this equation and write for the discrete case

$$\hat{I} = \sum_{k=-m}^m T_{j-k} h_k \quad \text{with} \quad \sum_{k=-m}^m h_k = 1. \quad (38)$$

and for the continuous case:

$$I = \int_{-L}^{+L} T(s-s') h(s') ds \quad \text{with} \quad \int_{-L}^L h(s') ds' = 1. \quad (39)$$

Using these relations we find immediately:

$$\mathbb{E}(\hat{I}) = \mathbb{E}(I) = \mathbb{E}(T), \quad (40)$$

and, since the variables  $T$  are independent,

$$\text{Var}(I) = \sigma_T^2 \sum_k h_k^2 \quad (41)$$



in the discrete case, or

$$\text{Var}(I) = \sigma_T^2 \int_{-L}^L h^2(s') ds' = \sigma_T^2 (h * h)(0) \quad (42)$$

in the continuous case. This shows that LIC leaves the average brightness of an image constant, but changes contrast. Using Eq. (31) for the autocovariance we have in the discrete case

$$\gamma_{\hat{I}}(\tau) = \text{Cov}(\hat{I}_t, \hat{I}_{t+\tau}) = \begin{cases} \sigma_T^2 \sum_{k=-m}^{m-\tau} h_k h_{k+\tau}, & \tau = 0, \dots, 2m \\ 0, & \tau > 2m \end{cases} \quad (43)$$

and using Eq. (32)

$$\gamma_I(\tau) = \text{Cov}(I_t, I_{t+\tau}) = \begin{cases} \sigma_T^2 \int_{-L}^{L-\tau} h(s) h(s+\tau), & 0 \leq \tau \leq 2L \\ 0, & \tau > L \end{cases} \quad (44)$$

in the continuous case. Therefore the autocorrelation is

$$\rho_{\hat{I}}(\tau) = \frac{\sum_{k=-m}^{m-\tau} h_k h_{k+\tau}}{\sum_{k=-m}^m h_k^2} \quad \text{for } \tau = 0, \dots, 2m \quad (45)$$

and in the continuous case

$$\rho_I(\tau) = \frac{\int_{-L}^{L-\tau} h(s) h(s+\tau)}{\int_{-L}^L h^2(s)} \quad \text{for } 0 \leq \tau \leq 2L. \quad (46)$$

Since  $\rho_I(\tau) = \rho_I(-\tau)$  this yields the simple relation

$$\rho_I(\tau) = \frac{h * h(\tau)}{h * h(0)}. \quad (47)$$

Hence, with increasing  $\tau$  the autocorrelation of LIC intensity values along a field line generated with a B-spline filter  $\tilde{b}_n$  drops like a B-spline  $\tilde{b}_{2n}$ .

### 5.3 The Effective Filter Length

For a meaningful comparison between different filter kernels their lengths have to be adjusted. Usually, the larger the filter length  $L$ , the larger the feature size along a field line and the less the contrast of the resulting LIC image. In our case contrast can simply be defined as the variance of the image's overall intensity distribution.

When comparing e.g. a box filter and a triangle filter of equal length, it is clear that the triangle filter has smaller feature size and higher contrast. The reason is that pixels close to the boundaries of the triangle filter get smaller weights. Therefore the *effective* filter length is smaller for the triangle filter.

In a comparative study we want to choose equal effective filter lengths. We determine these filter lengths such that the variance of the resulting intensity distribution is equal to the one of a corresponding box filter. Applying Eq. (42) and taking the ratio of the variance of a higher-order filter and a box filter we get the following results for the filter kernels listed in Table 1:

$$\begin{aligned} \text{Triangle filter:} & \quad L_{\text{eff}} = \frac{4}{3}L_{\text{box}} \\ \text{3rd-order filter:} & \quad L_{\text{eff}} = \frac{33}{20}L_{\text{box}} . \end{aligned}$$

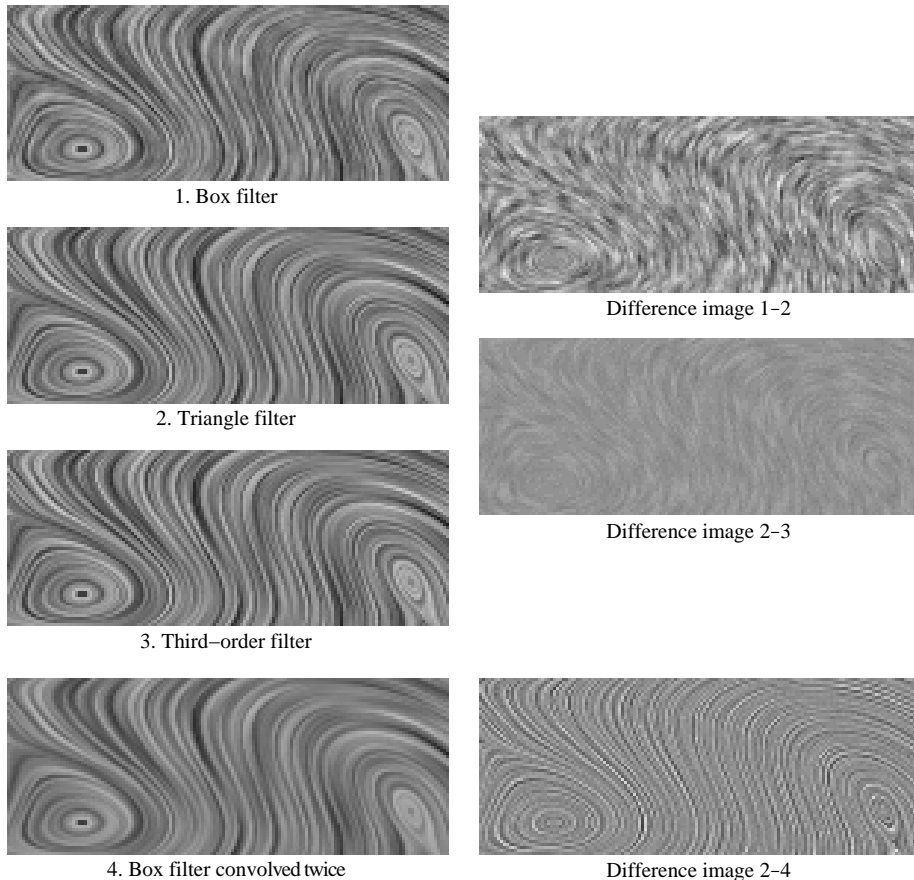


Figure 4: LIC images obtained with different filter kernels. The contrast of the three difference images has been increased equally for better visibility. On a computer monitor the differences between the images on the left are much more obvious.

## 6 Results

Fig. 4 illustrates the effect of different filters kernels on LIC images. All images have been computed using the fast LIC algorithm outlined in the previous section. The differences between the box filter and higher-order filter kernels are well noticeable on a computer monitor. In order to make these differences

visible in the printed book, i.e. after a complex reproduction process, the images are magnified.

In Color Plates 5 and 6 on page 20 in the Appendix we depict solutions of ODEs, using LIC with triangular filter kernels. Furthermore, contrast enhanced difference images of LIC images with triangular respectively box filter kernels are shown. Note that the error structures are much smaller than the characteristic feature length of the LIC texture. Therefore, box filter images appear noisier on the screen than triangular filter images.

We conclude that higher order filter kernels can be combined with the fast LIC algorithm and therefore are well suited for interactive visualization. Statistical analysis of pixel correlations as well as visual investigation show that higher higher order filter kernels lead to smoother, less noisy images. For practical purposes LIC images produced with triangular or quadratic B-spline filters are sufficient. The differences between these and higher order filters in general are hardly visible.

## References

- [1] H. BATTKE, D. STALLING, AND H. HEGE, *Fast line integral convolution for arbitrary surfaces in 3D*, Visualization and Mathematics (H. HEGE AND K. POLTHIER, eds.), Springer, 1997, pp. 181–195.
- [2] C. DE BOOR, *A practical guide to splines*, Applied Mathematical Sciences, vol. 27, Springer, New York, Heidelberg, Berlin, 1978.
- [3] B. CABRAL, H.-C. HEGE, V. INTERRANTE, K.-L. MA, AND D. STALLING, *Texture synthesis with line integral convolution – course notes*, Siggraph 97, ACM Siggraph, Cambridge, 1997.
- [4] B. CABRAL AND C. LEEDOM, *Highly parallel vector visualization using line integral convolution*, Seventh SIAM Conference on Parallel Processing for Scientific Computing, February 1995, pp. 802–807.
- [5] B. CABRAL AND L. C. LEEDOM, *Imaging vector fields using line integral convolution*, Computer Graphics (Siggraph '93 Proceedings) (J. T. KAJIYA, ed.), vol. 27, August 1993, pp. 263–272.
- [6] H. CURRY AND I. SCHOENBERG, *On spline distributions and their limits: the pólya distribution functions*, Bull. Amer. Math. Soc. **53** (1947), 1114.
- [7] P. DEUFLHARD AND A. HOHMANN, *A First Course in Scientific Computation*, Verlag de Gruyter, Berlin, 1995.
- [8] L. K. FORSELL, *Visualizing flow over curvilinear grid surfaces using line integral convolution*, Visualization '94, IEEE Computer Society, 1994, pp. 240–247.
- [9] L. K. FORSELL 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** (1995), 133–141.

- [10] V. INTERRANTE, *Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution*, , Computer Graphics Proceedings, Annual Conference Series, ACM Siggraph, Addison Wesley, August 1997, held in Los Angeles, California, 3-8 August 1997, pp. 109–116.
- [11] V. INTERRANTE AND C. GROSCH, *Strategies for effectively visualizing a 3D flow using volume line integral convolution*, ICASE, Technical Report TR-97-35, July 1997.
- [12] X. MAO, M. KIKUKAWA, N. FUJITA, AND A. IMAMIYA, *Line integral convolution for arbitrary 3D surfaces through solid texturing*, Proceedings of Eighth Eurographics Workshop on Visualization in Scientific Computing, 1997 (to appear).
- [13] H. SHEN, C. R. JOHNSON, AND K. MA, *Visualizing vector fields using line integral convolution and dye advection*, 1996 Volume Visualization Symposium, IEEE, October 1996, pp. 63–70.
- [14] D. STALLING AND H. HEGE, *Fast and resolution independent line integral convolution*, , Annual Conference Series, ACM Siggraph, Addison Wesley, August 1995, held in Los Angeles, California, 6-11 August 1995, pp. 249–256.
- [15] D. STALLING, M. ZÖCKLER, AND H. HEGE, *Parallel line integral convolution*, Parallel Computing **23** (1997), 975–989.
- [16] C. TEITZEL, R. GROSSO, AND T. ERTL, *Line integral convolution on triangulated surfaces*, Proceedings of WSCG '97, The Fifth International Conference in Central Europe on Computer Graphics and Visualization '97 (N. M. THALMANN AND V. SKALA, eds.), vol. 3, University of West Bohemia Press, February 1997, held in Plzen, Czech Republic, February 1997, pp. 572–581.
- [17] R. WEGENKITTL, E. GRÖLLER, AND W. PURGATHOFER, *Animating flow-fields: Rendering of oriented line integral convolution*, Vienna University of Technology, Institute of Computer Graphics, Technical Report TR-186-2-96-23, December 1996.

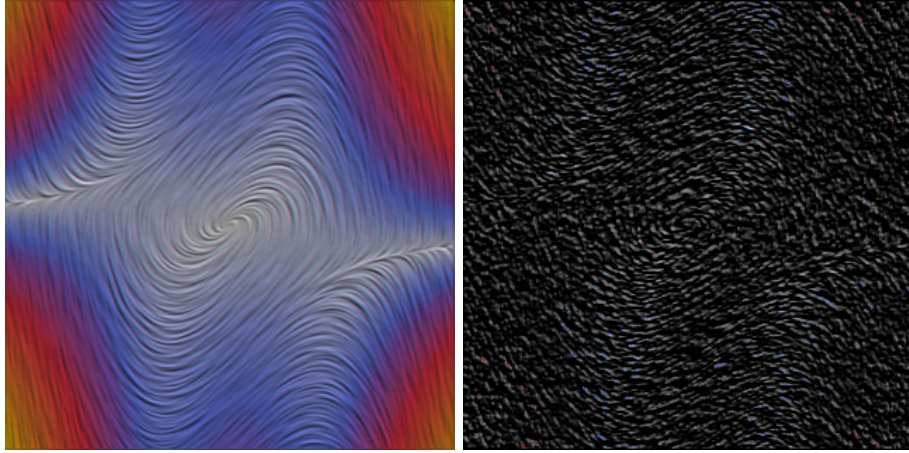


Figure 5: The left image shows the solution of the ODE  $x' = y, y' = (1-x^2)y-x$  using line integral convolution with a triangular filter kernel. This filter produces smoother results than a box filter. Right: contrast enhanced difference between the left image and one generated with a box filter (width of the box filter is  $3/4$  of the triangle filter). Note, that the error structures are much smaller than the characteristic feature length of the LIC texture. Therefore, LIC images generated by box filters appear noisier.

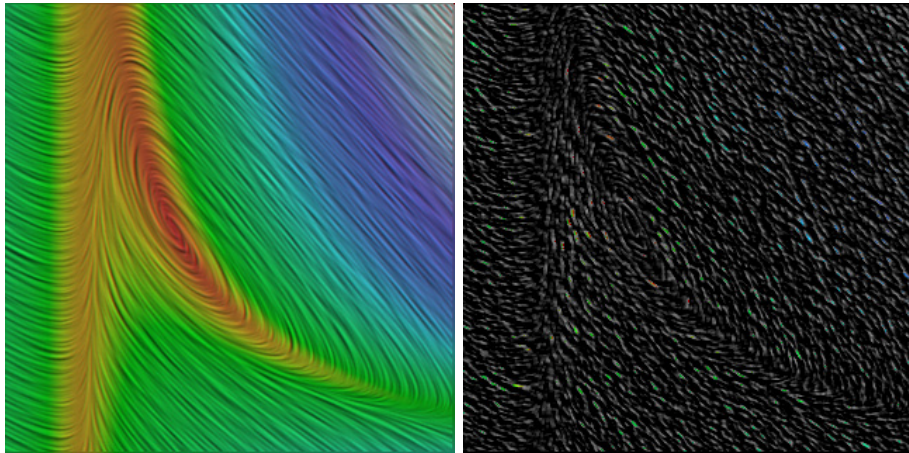


Figure 6: Like above, a LIC image generated with a triangular filter kernel is shown (left) and compared with a LIC image generated with a box filter by depicting a contrast enhanced difference image (right). The image on the left depicts the solution curves of the ODE  $x' = 1 + x^2y - 4x, y' = 3x - x^2y$ . Color encodes magnitude of the vector  $(x', y')$ .