CS 247 – Scientific Visualization
Lecture 27: Vector / Flow Visualization, Pt. 6

Markus Hadwiger, KAUST

Reading Assignment #14 (until May 9)

Read (required):

- Data Visualization book, Chapter 6.6
- B. Cabral, C. Leedom:
  Imaging Vector Fields Using Line Integral Convolution, SIGGRAPH 1993

  http://dx.doi.org/10.1145/166117.166151
- Learn how convolution (the convolution of two functions) works:

  https://en.wikipedia.org/wiki/Convolution

Read (optional):

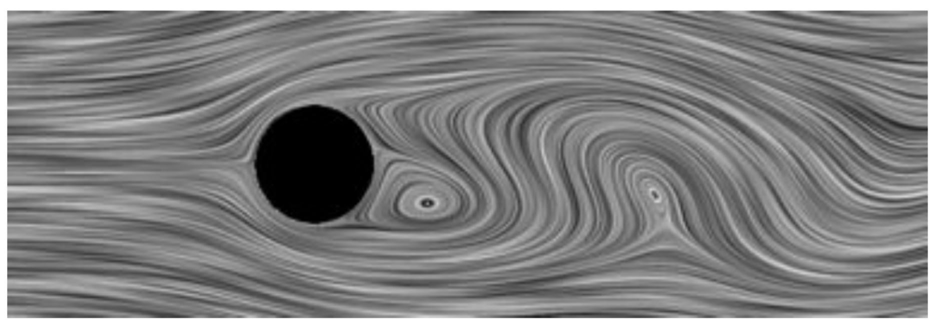
- Paper: Streak Lines as Tangent Curves of a Derived Vector Field,
  Tino Weinkauf and Holger Theisel, IEEE Vis 2010

  http://dx.doi.org/10.1109/TVCG.2010.198

Line Integral Convolution (LIC)

Line Integral Convolution

- Line Integral Convolution (LIC)
  - Visualize dense flow fields by imaging its integral curves
  - Cover domain with a random texture (so called ‚input texture‘, usually stationary white noise)
  - Blur (convolve) the input texture along stream lines using a specified filter kernel
- Look of 2D LIC images
  - Intensity distribution along stream lines shows high correlation
  - No correlation between neighboring stream lines

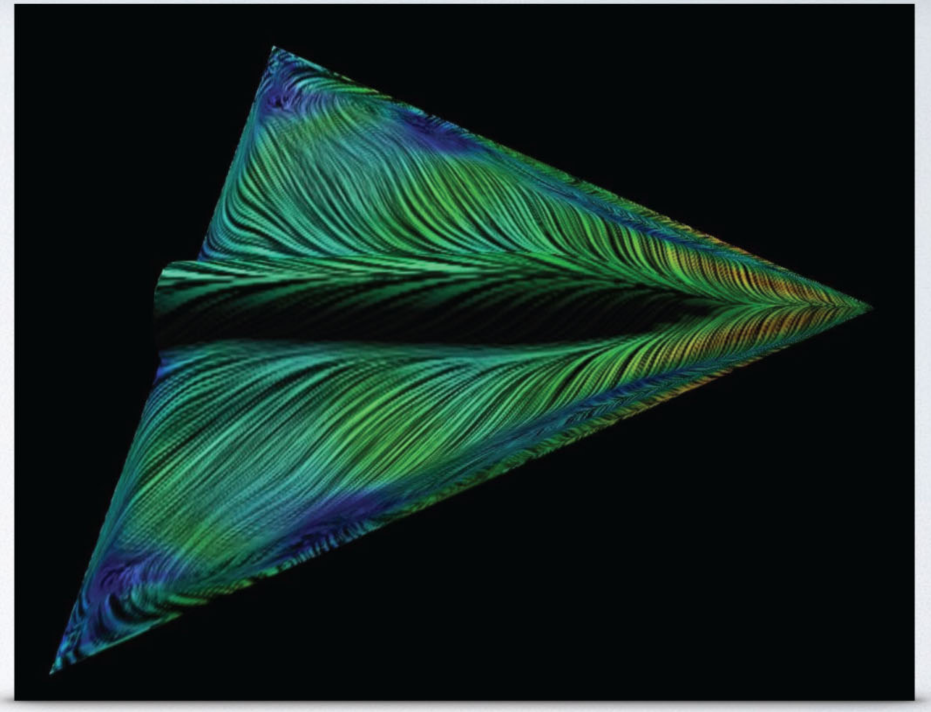


Line Integral Convolution I

- Line Integral Convolution (LIC):
  - goal: general overview of flow
  - approach: use dense textures
  - idea: flow ⇹ visual correlation

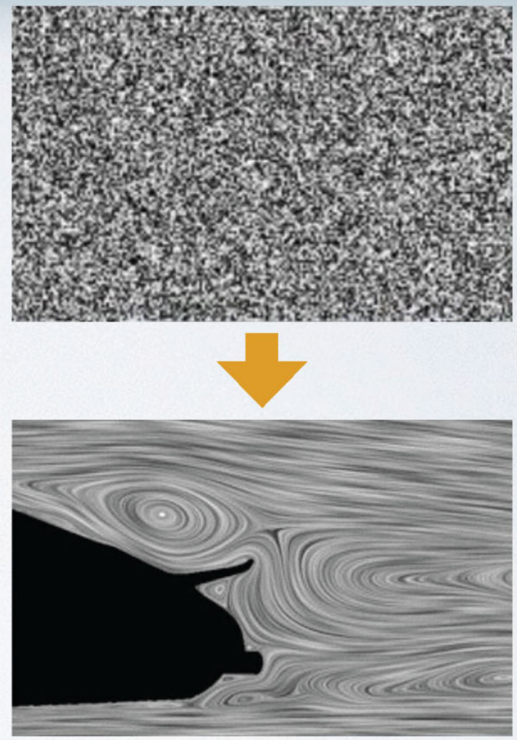
Line Integral Convolution I

- Line Integral Convolution (LIC):
  - goal: general overview of flow
  - approach: use dense textures
  - idea: flow ↔ visual correlation



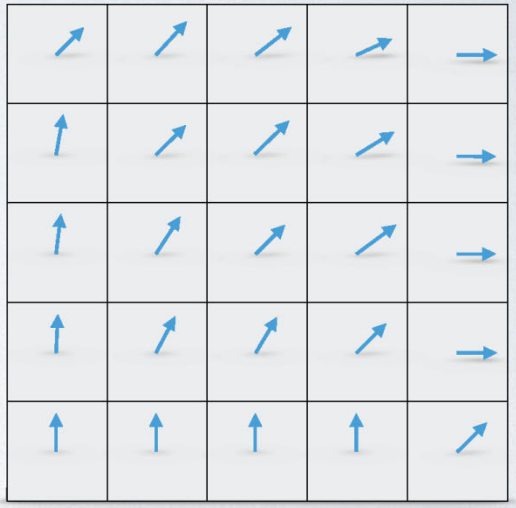
Line Integral Convolution II

- Idea
  - global visualization technique
  - dense representation
  - start with random texture
  - smear along stream lines
- Only for **stream lines!**
  (steady flow, i.e. time-independent fields)



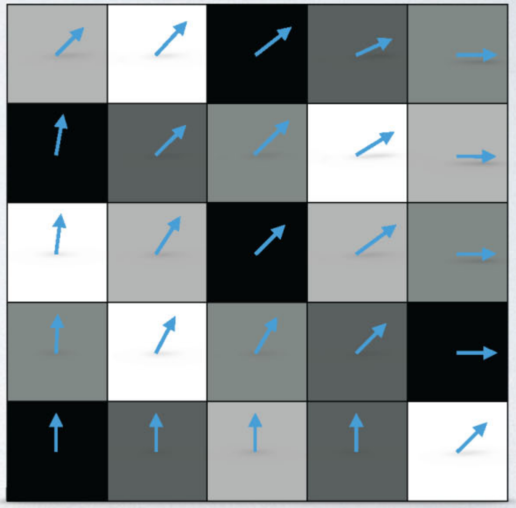
Line Integral Convolution III

- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture
    ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



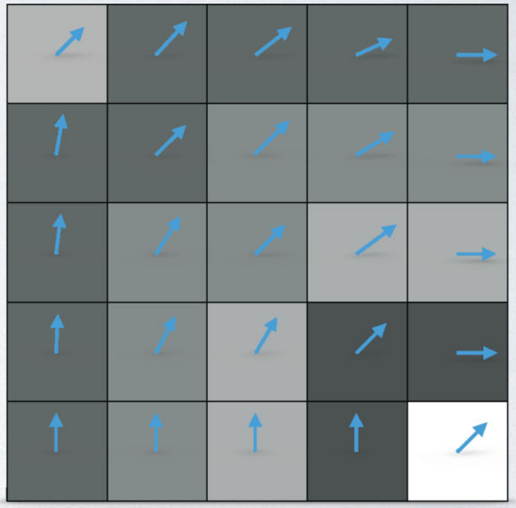
Line Integral Convolution III

- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture
    ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



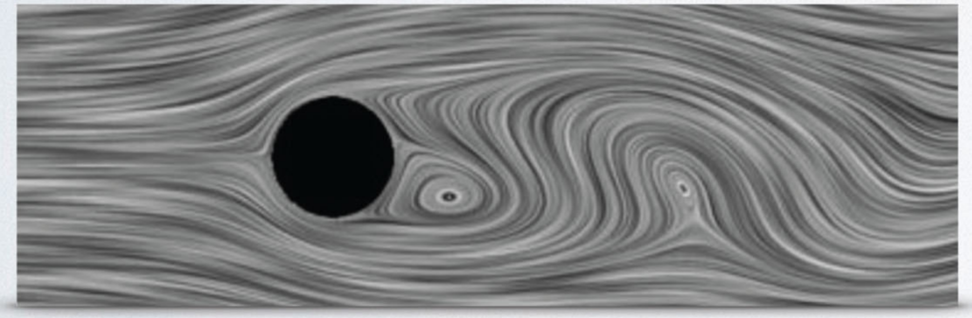
Line Integral Convolution III

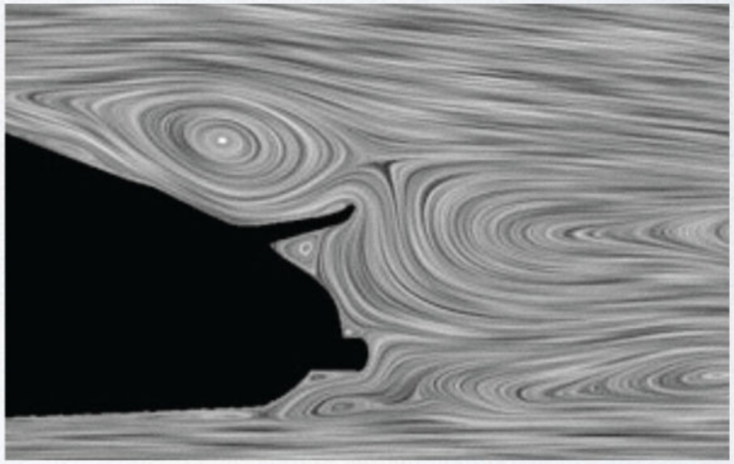
- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture
    ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



Line Integral Convolution IV

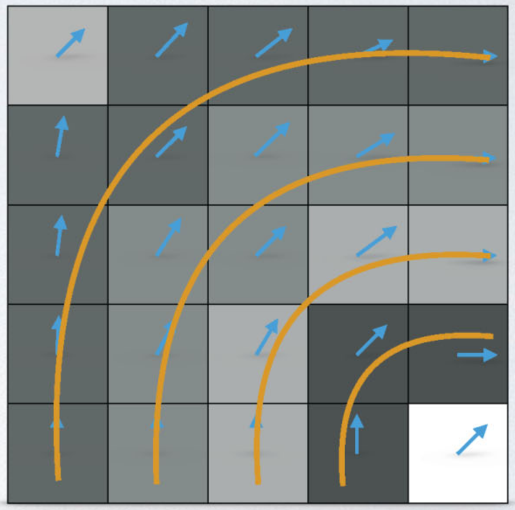
- Look of 2D LIC images
  - intensity along stream lines shows high correlation
  - no correlation between neighboring stream lines





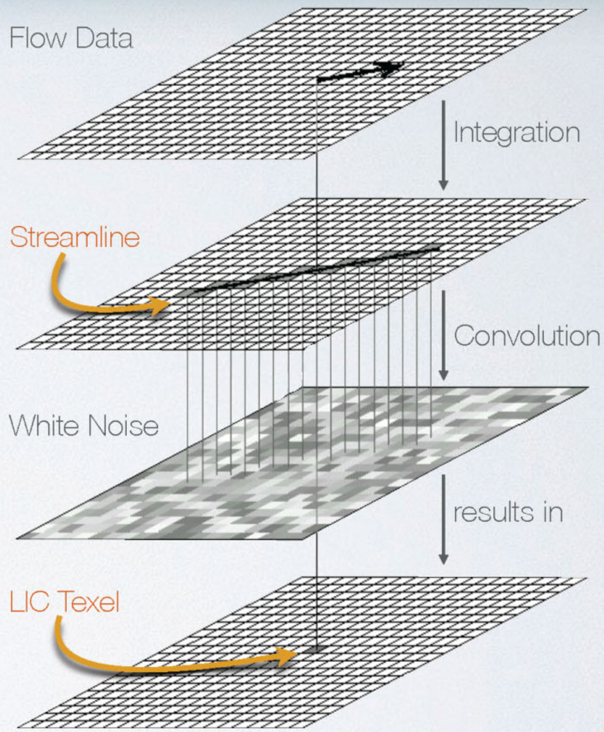
LIC Approach - Goal

- For every texel: let the texture value
  - correlate with neighboring texture values along the flow (in flow direction)
  - not correlate with neighboring texture values across the flow (normal to flow direction)
- Result: along streamlines the texture values are correlated ⇨ visually coherent!



LIC Approach - Steps

- Idea: "smear" white noise (no a priori correlations) along flow
- Calculation of a texture value:
  - follow streamline through point
  - filter white noise along streamline



Convolution Example

Gaussian Blur

en.wikipedia.org/wiki/Gaussian_blur

Cut off filter kernel after an extent of, e.g., 3*standard deviation in each direction

Example:

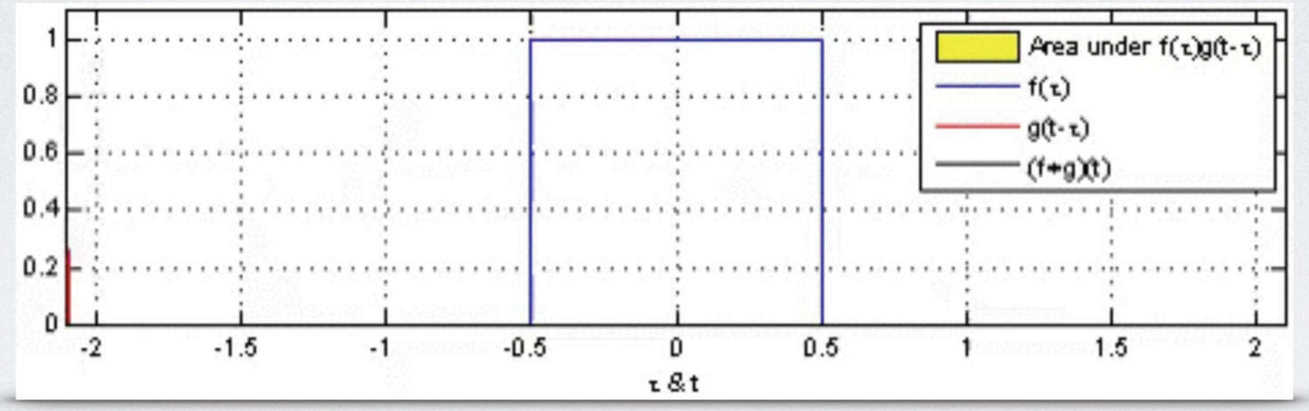
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067
0.00002292	0.00078634	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00038771	0.01330373	0.11098164	**0.22508352**	0.11098164	0.01330373	0.00038771
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067

Note that 0.22508352 (the central one) is 1177 times larger than 0.00019117 which is just outside 3σ.

Can do multiple iterations to achieve larger effective filter size

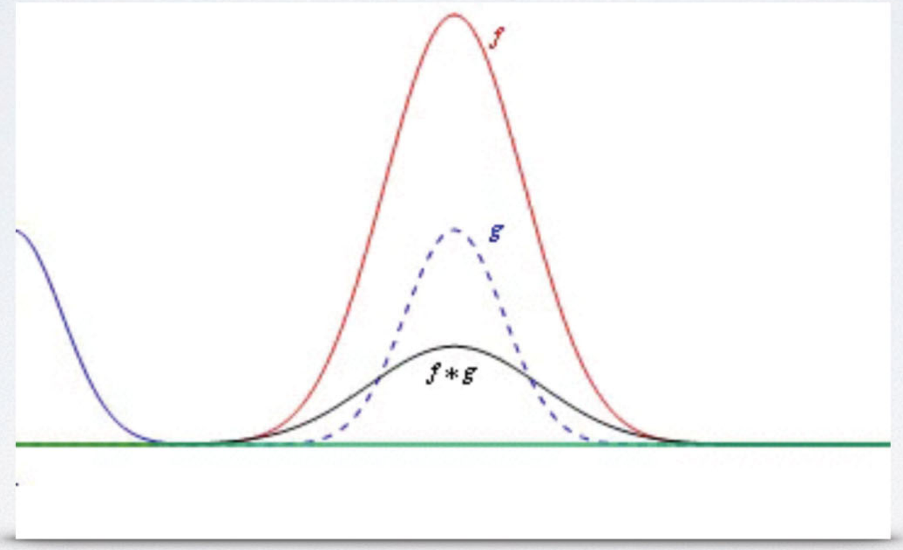
LIC Approach - 1D Convolution I

- Convolution defined as $(f * g)(x) := \int_{\mathbb{R}^n} f(\tau)g(x-\tau)d\tau$



LIC Approach - 1D Convolution II

- Convolution defined as $(f * g)(x) := \int_{\mathbb{R}^n} f(\tau) g(x - \tau) d\tau$



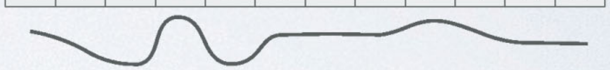
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

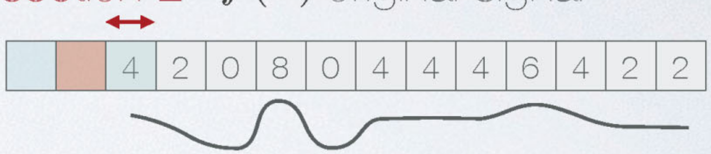
$(f * k)(x)$ smoothed signal

LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

common section L

$f(x)$ original signal



$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x - \tau)d\tau$$

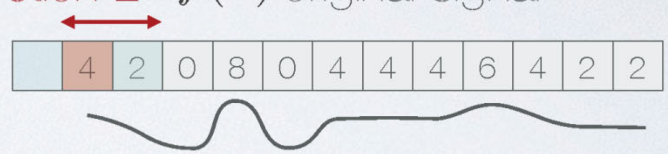
$(f * k)(x)$ smoothed signal

LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

common section L

$f(x)$ original signal



¼ · 0 + ½ · 4 + ¼ · 2

$(f * k)(x)$ smoothed signal

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau) k(x - \tau) d\tau$$

LIC Approach - 1D Convolution III

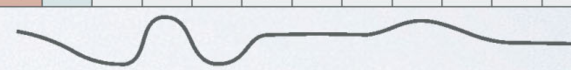
$k(x)$ convolution kernel

¼	½	¼

common section L

$f(x)$ original signal

	4	2	0	8	0	4	4	4	6	4	2	2



$\frac{1}{4} \cdot 0 + \frac{1}{2} \cdot 4 + \frac{1}{4} \cdot 2$

$(f * k)(x)$ smoothed signal

3											

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau) k(x - \tau) d\tau$$

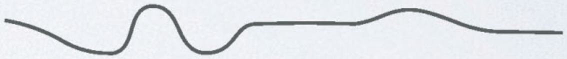
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 4 + ½ · 2 + ¼ · 0

$(f * k)(x)$ smoothed signal

3											

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

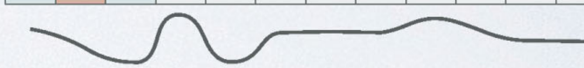
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 4 + ½ · 2 + ¼ · 0

$(f * k)(x)$ smoothed signal

3	2										

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau) k(x - \tau) d\tau$$

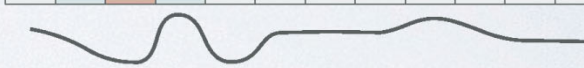
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



$\frac{1}{4} \cdot 2 + \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 8$

$(f * k)(x)$ smoothed signal

3	2										

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau) k(x - \tau) d\tau$$

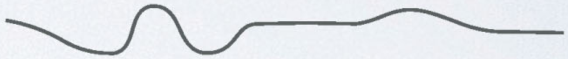
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 2 + ½ · 0 + ¼ · 8

$(f * k)(x)$ smoothed signal

3	2	2½									

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

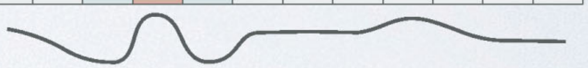
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 0 + ½ · 8 + ¼ · 0

$(f * k)(x)$ smoothed signal

3	2	2½									

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

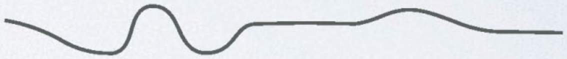
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 0 + ½ · 8 + ¼ · 0

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

$(f * k)(x)$ smoothed signal

3	2	2½	4								

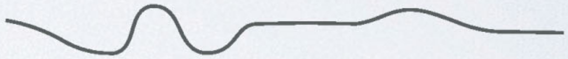
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2	

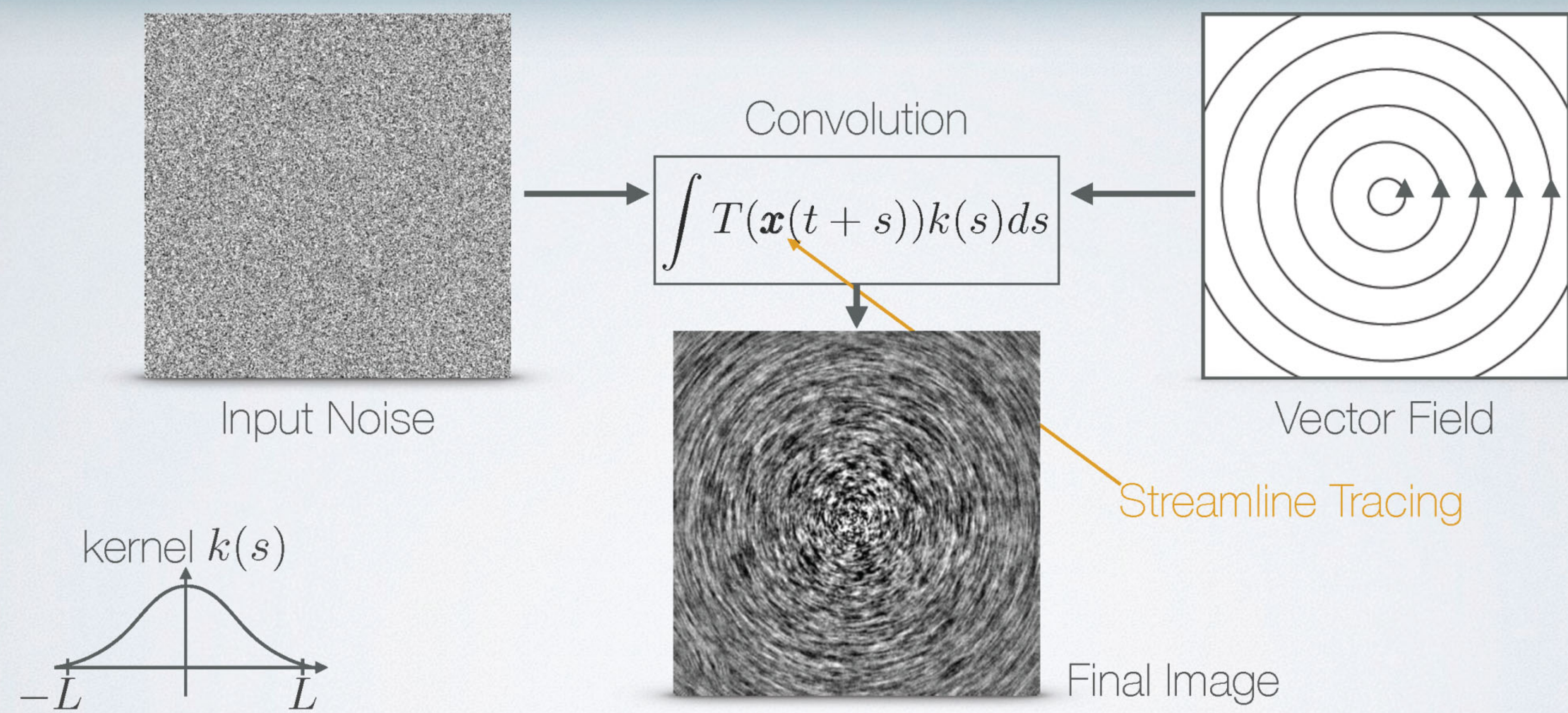


$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

$(f * k)(x)$ smoothed signal

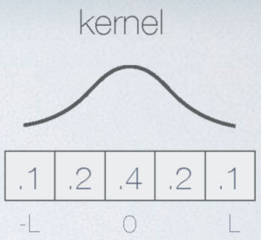
3	2	2½	4	3	3	4	4½	5	4	2½	1½

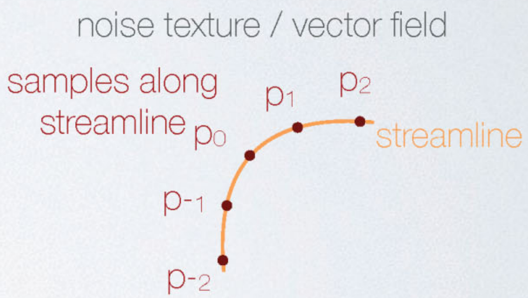
LIC Approach - 1D Convolution IV



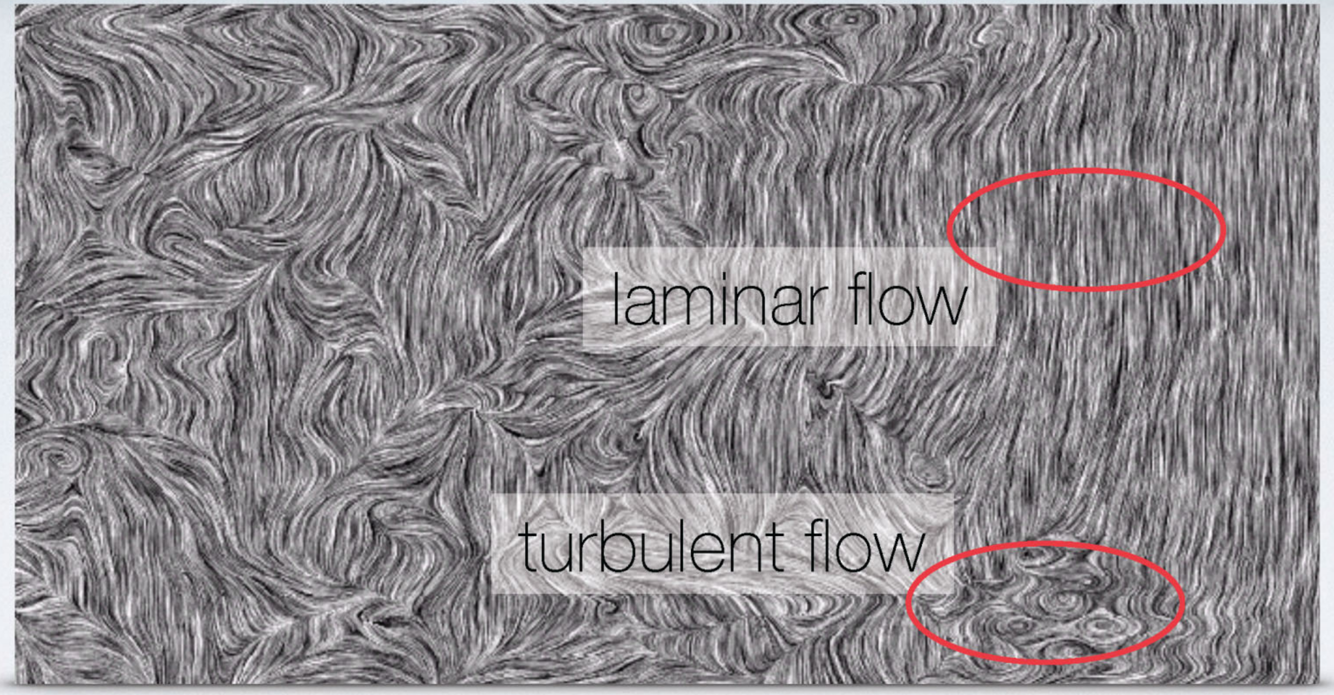
LIC - Algorithm

```
for each pixel //perfect fit for fragment shader

  t = texture( position, noise_texture );

  smoothed_value = kernel_value(center) * t;
  P+ = p- = position;

  for 1 to L // loop over kernel

    v+ = texture( p+, vector_texture );
    p+ = streamlineIntegration(p+, v+);
    smoothed_value +=
        kernel_value * texture( p+, noise_texture );

    v- = -texture( p-, vector_texture );
    p- = streamlineIntegration(p-, v-);
    smoothed_value +=
        kernel_value * texture( p-, noise_texture );
```





LIC - 2D Example



Linear Algebra Approach (1)

- Toeplitz matrix: constant diagonals

$$\mathbf{T} := \left(t_{ij}\right) \text{ with } t_{ij} := t_{i-j}$$

$$\mathbf{T}^{N\times N} := \begin{bmatrix} t_0 & t_{(-1)} & t_{(-2)} & \cdots & t_{(-(N-2))} & t_{(-(N-1))} \\ t_1 & t_0 & t_{(-1)} & \cdots & t_{(-(N-3))} & t_{(-(N-2))} \\ t_2 & t_1 & t_0 & \cdots & t_{(-(N-4))} & t_{(-(N-3))} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{N-2} & t_{N-3} & t_{N-4} & \cdots & t_0 & t_{(-1)} \\ t_{N-1} & t_{N-2} & t_{N-3} & \cdots & t_1 & t_0 \end{bmatrix}$$

Linear Algebra Approach (2)

- Circulant matrix: special case of Toeplitz matrix

$$\mathbf{C} := (c_{ij}) \text{ where } c_{ij} := c_{(i-j)\, mod\, N}$$

$$\mathbf{C}^{N\times N} := \begin{bmatrix} c_0 & c_{N-1} & c_{N-2} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{N-1} & \cdots & c_3 & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_4 & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-2} & c_{N-3} & c_{N-4} & \cdots & c_0 & c_{N-1} \\ c_{N-1} & c_{N-2} & c_{N-3} & \cdots & c_1 & c_0 \end{bmatrix}$$

- Periodic convolution: multiply **C** with (periodic) signal in column vector
- The Fourier transform *diagonalizes* circulant matrices

Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama