King Abdullah University of Science and Technology KAUST

CS 247 – Scientific Visualization
Lecture 26: Vector / Flow Visualization, Pt. 5

Markus Hadwiger, KAUST

Reading Assignment #14 (until May 11)

Read (required):

- Data Visualization book, Chapter 6.7
- J. van Wijk: *Image-Based Flow Visualization*,
  ACM SIGGRAPH 2002

  `http://www.win.tue.nl/~vanwijk/ibfv/ibfv.pdf`

Read (optional):

- T. Günther, A. Horvath, W. Bresky, J. Daniels, S. A. Buehler:
  Lagrangian Coherent Structures and Vortex Formation in High Spatiotemporal-Resolution Satellite Winds of an Atmospheric Karman Vortex Street, 2021

  `https://www.essoar.org/doi/10.1002/essoar.10506682.2`

- H. Bhatia, G. Norgard, V. Pascucci, P.-T. Bremer:
  The Helmholtz-Hodge Decomposition – A Survey, TVCG 19(8), 2013

  `https://doi.org/10.1109/TVCG.2012.316`

- Work through online tutorials of multi-variable partial derivatives, grad, div, curl, Laplacian:

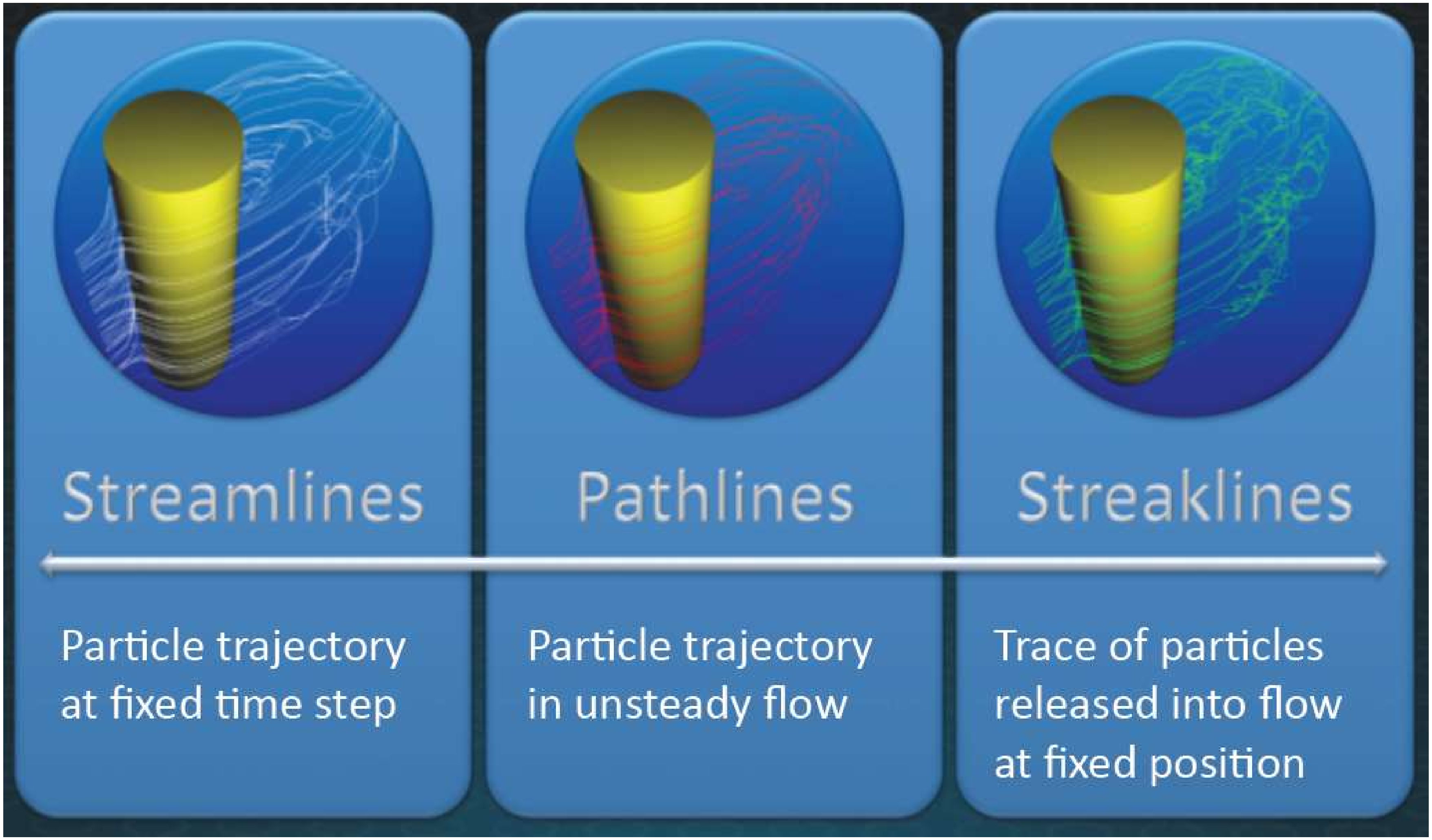
  `https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives`
  `https://www.youtube.com/watch?v=rB83DpBJQsE` (3Blue1Brown)

- Matrix exponentials:

  `https://www.youtube.com/watch?v=O85OWBJ2ayo` (3Blue1Brown)

Integral Curves, Pt. 2

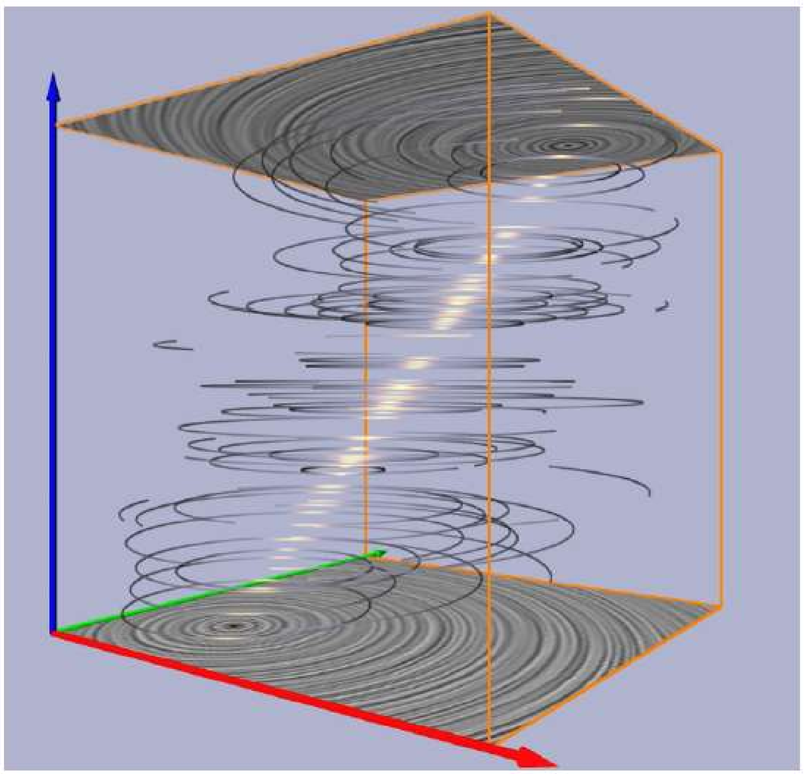
Integral Curves



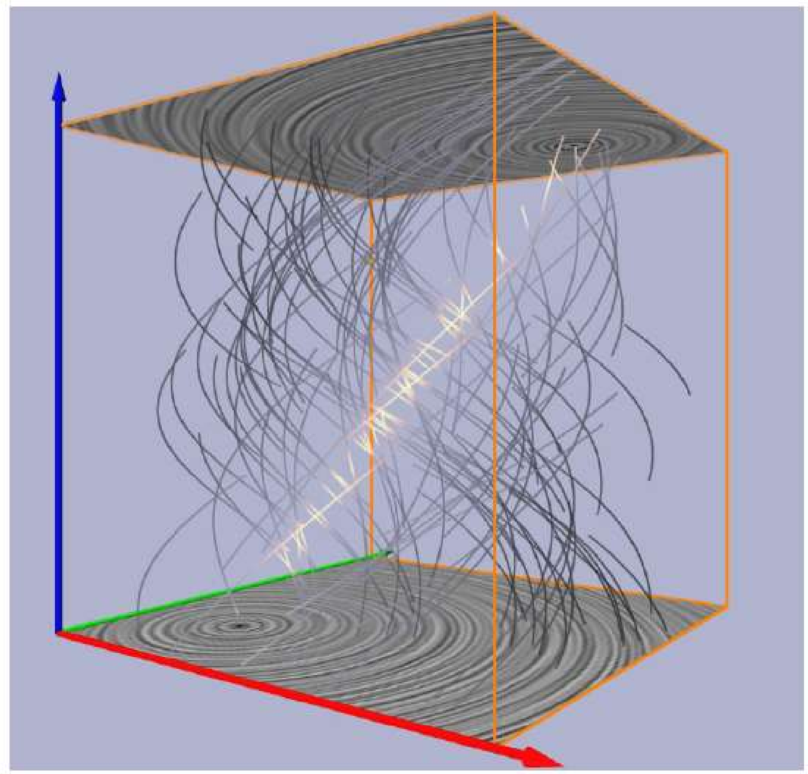
Stream Lines vs. Path Lines Viewed Over Time

Plotted with time as third dimension

- Tangent curves to a (n + 1)-dimensional vector field



Stream Lines



Path Lines

Streamline

- Curve parallel to the vector field in each point for a fixed time

Pathline

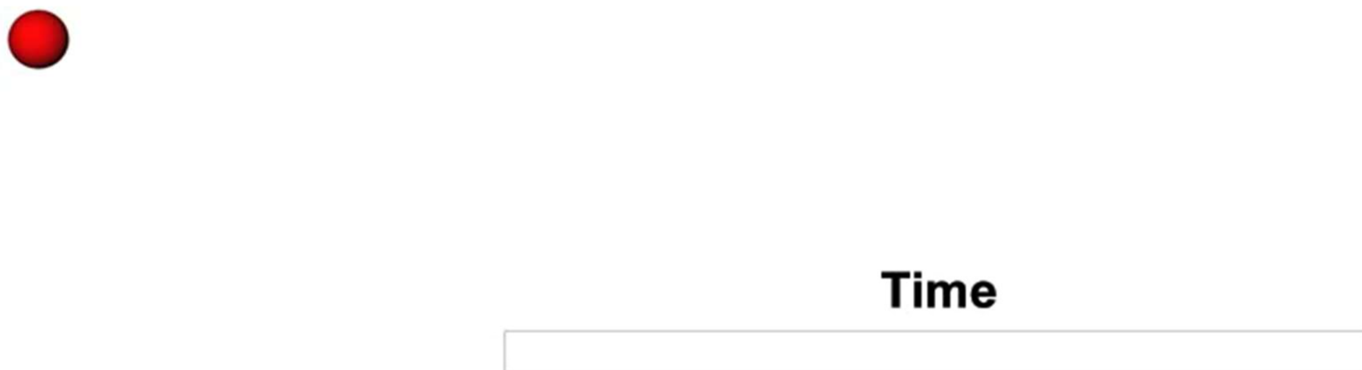
- Describes motion of a massless particle over time

Streakline

- Location of all particles released at a *fixed position* over time

Timeline

- Location of all particles released along a line at a *fixed time*



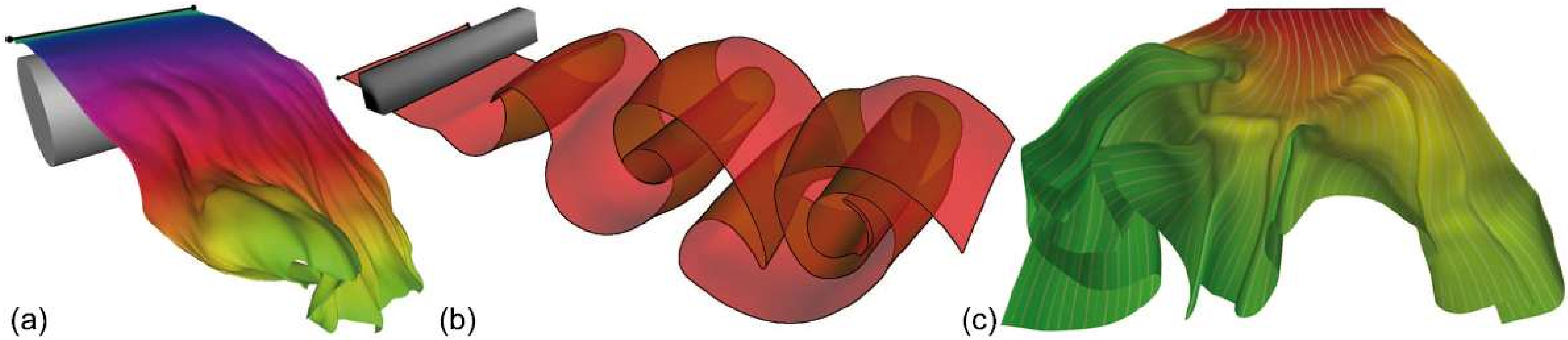
streak line

location of all particles set out at a fixed point at different times

Surfaces Instead of Lines

Seeding from a line instead of from a point

Example: streak surfaces



Volumes: seeding from a surface instead of a line

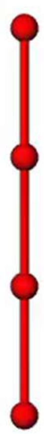
Real “Streak Surfaces”

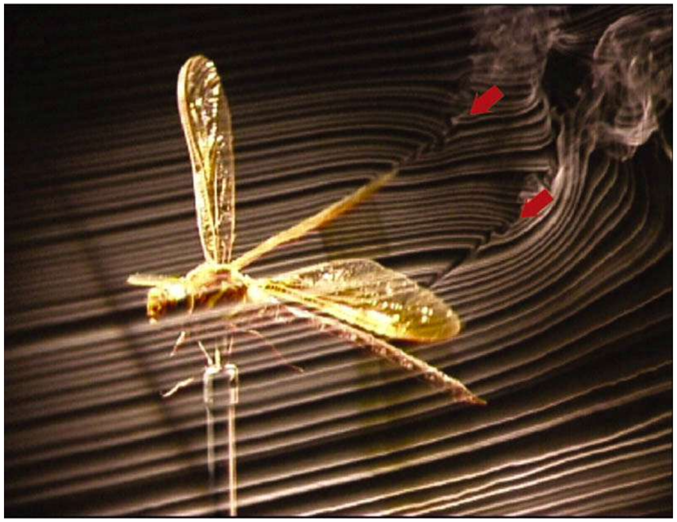
Artistic photographs of smoke

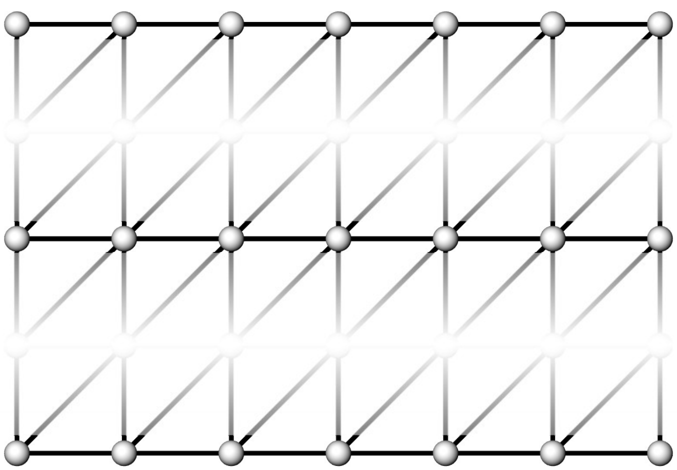
Time

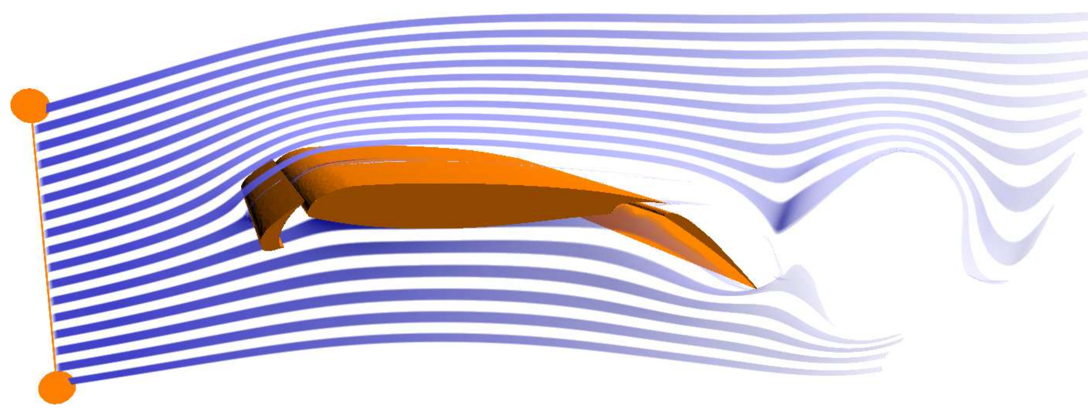
streak line

streak surface





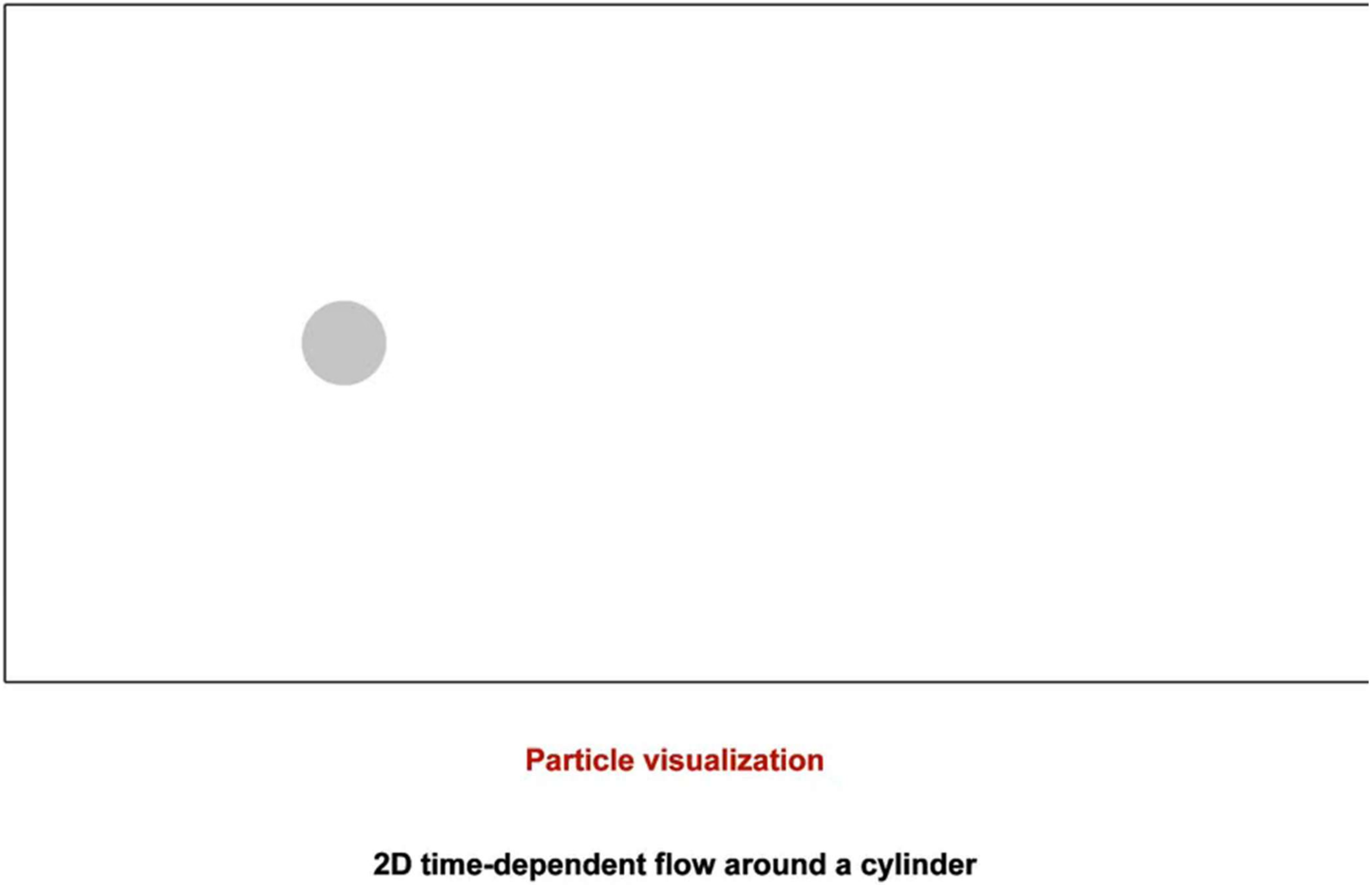




[Data courtesy of Günther (TU Berlin)]

fixed zero opacity rows

break connectivity

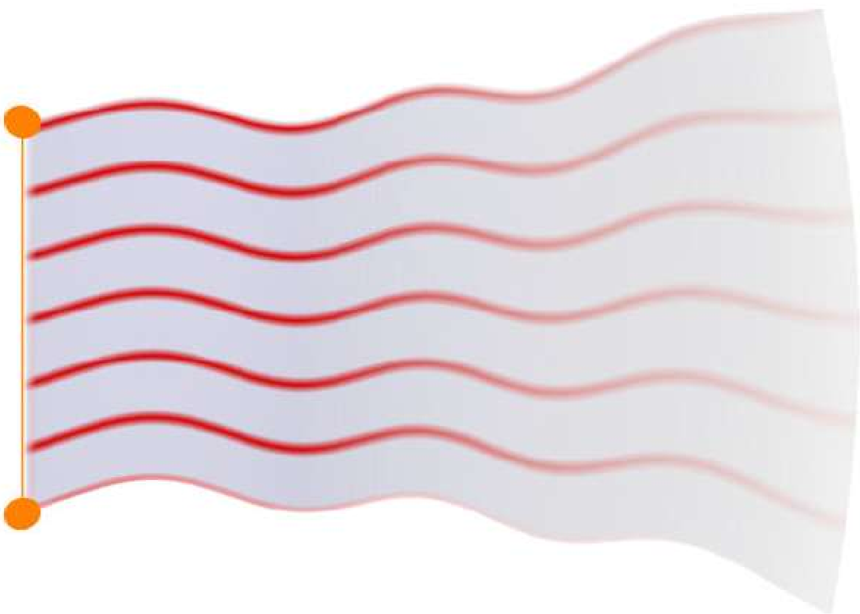


time line

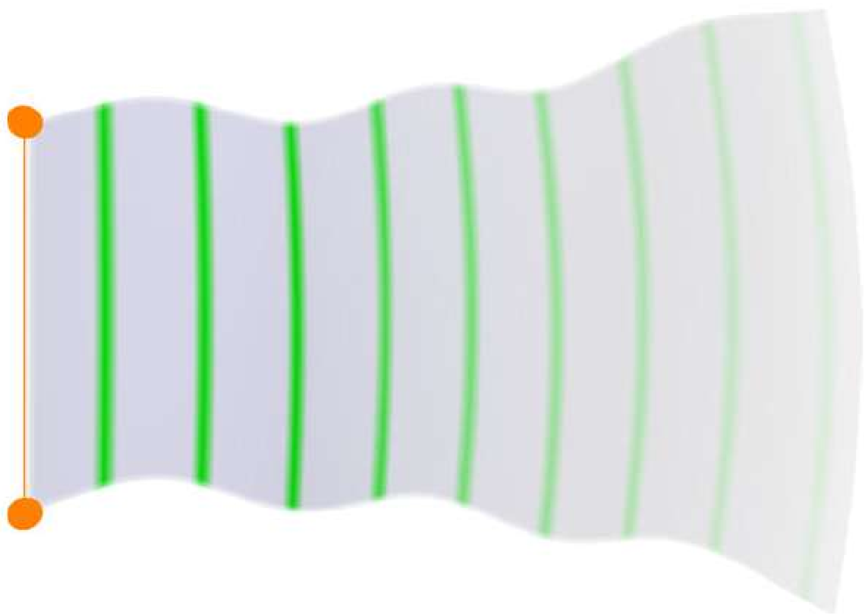
location of all particles set out on a certain line at a fixed time

Streak Lines vs. Time Lines

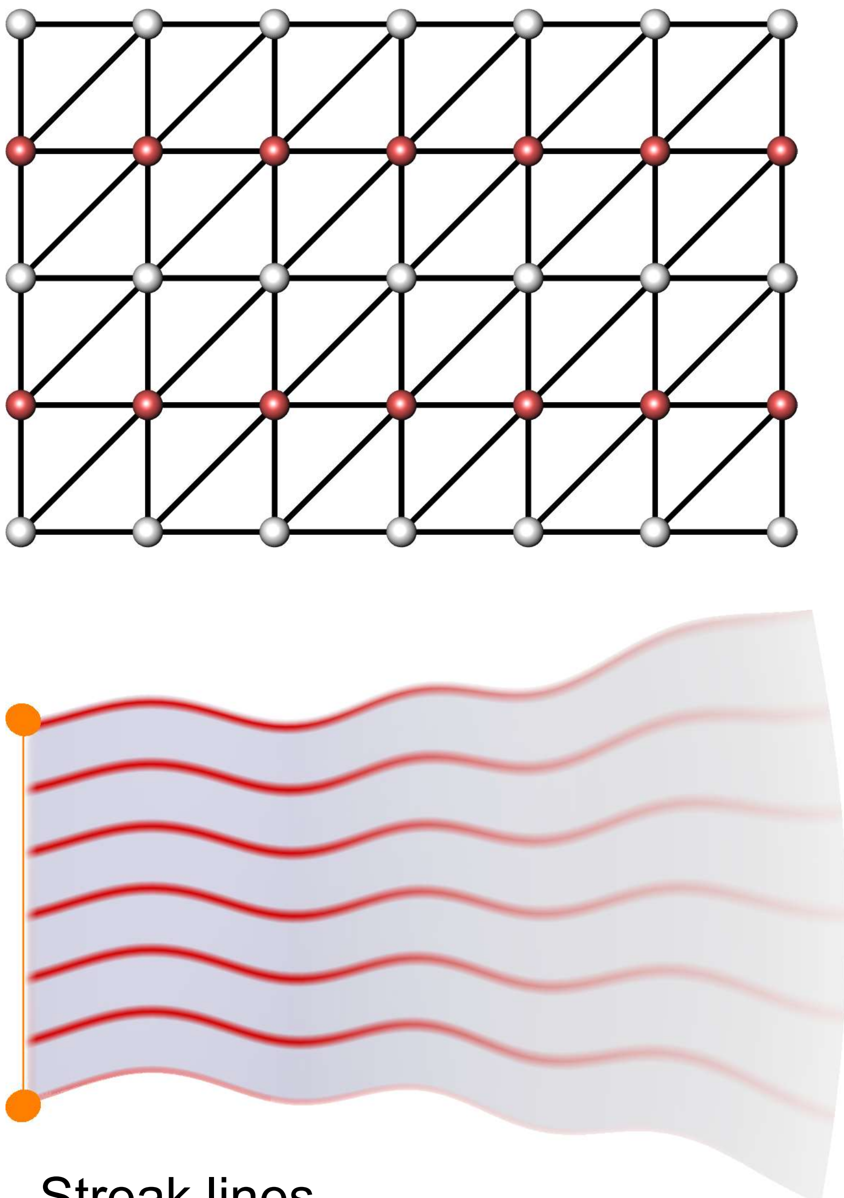
(on a streak surface)



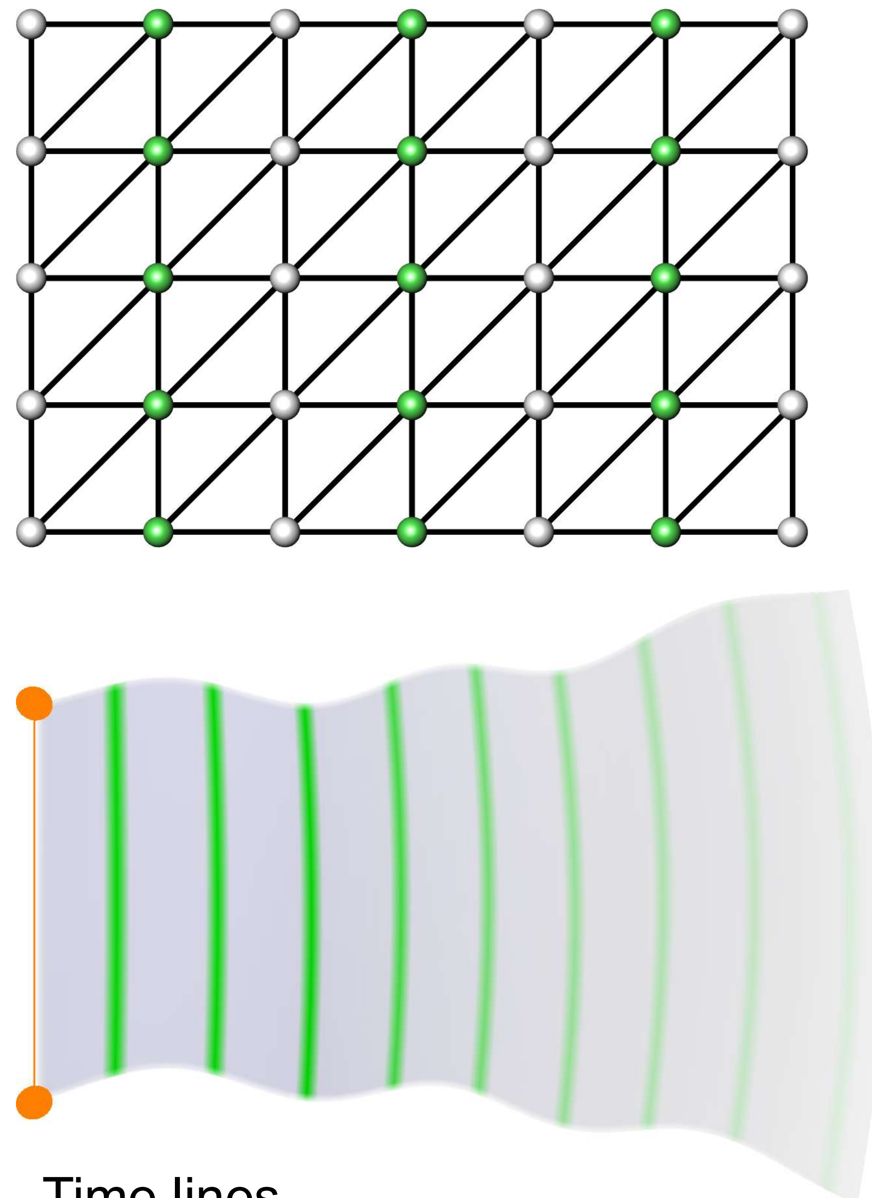
Streak Lines



Time Lines



Streak lines



Time lines

The Flow / Flow Map of a Vector Field (1)

Flow of a *steady (time-independent)* vector field

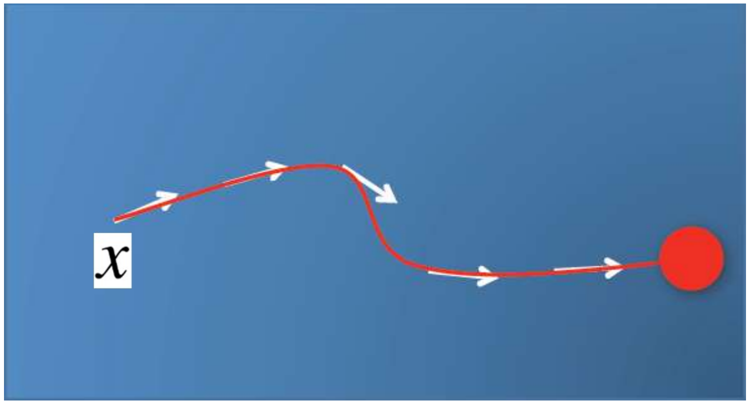
- Map source position x "forward" (t>0) or "backward" (t<0) by time t

$\boxed{\phi(x,t)}$ $\boxed{\phi_t(x)}$ with $\phi_0(x) = x$

$\phi_s(\phi_t(x)) = \phi_{s+t}(x)$

$$\phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n, \quad (x,t) \mapsto \phi(x,t).$$

$$\phi_t : \mathbb{R}^n \to \mathbb{R}^n, \quad x \mapsto \phi_t(x).$$



The Flow / Flow Map of a Vector Field (1)

Flow of a *steady (time-independent)* vector field

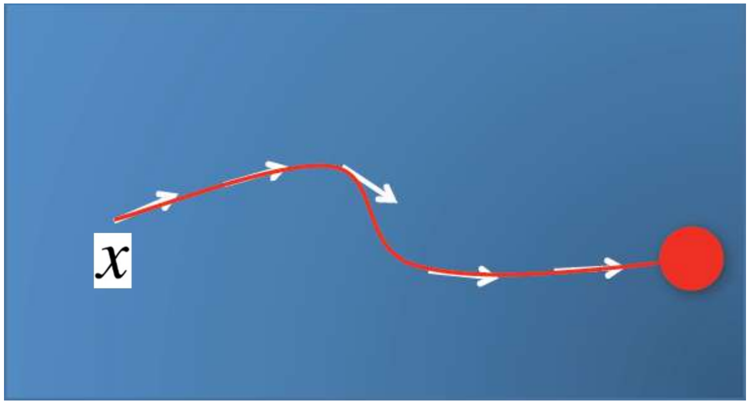
- Map source position *x* “forward” (*t*>*0*) or “backward” (*t*<*0*) by time *t*

$\boxed{\phi(x,t)}$ $\boxed{\phi_t(x)}$ with $\phi_0(x) = x$

$\phi_s(\phi_t(x)) = \phi_{s+t}(x)$

$$\phi : M \times \mathbb{R} \to M, \quad (x,t) \mapsto \phi(x,t).$$

$$\phi_t : M \to M, \quad x \mapsto \phi_t(x).$$



The Flow / Flow Map of a Vector Field (1)

Flow of a *steady (time-independent)* vector field

- Map source position *x* "forward" (t>*0*) or "backward" (t<*0*) by time *t*

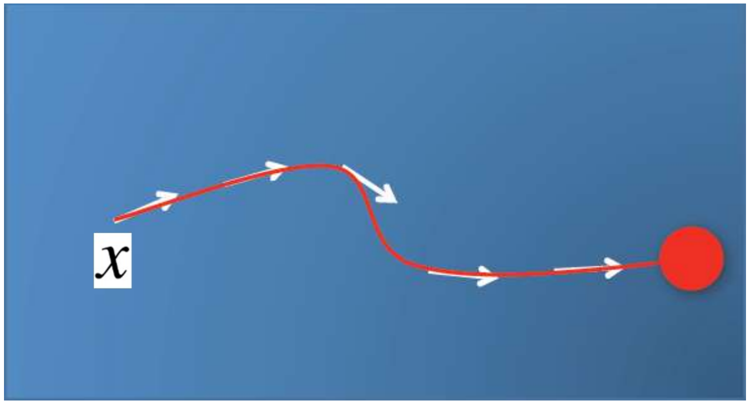
$\boxed{\phi(x,t)}$ $\boxed{\phi_t(x)}$ with $\phi_0(x) = x$

$$\phi_s(\phi_t(x)) = \phi_{s+t}(x)$$

$$\phi : M \times \mathbb{R} \to M, \quad (x,t) \mapsto \phi(x,t).$$

$$\phi_t : M \to M, \quad x \mapsto \phi_t(x).$$

$$\phi(x,t) = x + \int_0^t \mathbf{v}(\phi(x,\tau))\, \mathrm{d}\tau$$



(on a general manifold *M*, integration is performed in coordinate charts)

The Flow / Flow Map of a Vector Field (1)

Flow of a *steady (time-independent)* vector field

- Map source position *x* "forward" (*t>0*) or "backward" (*t<0*) by time *t*

$$\boxed{\phi(x,t)} \qquad \boxed{\phi_t(x)}$$

with

$$\phi_0(x) = x$$

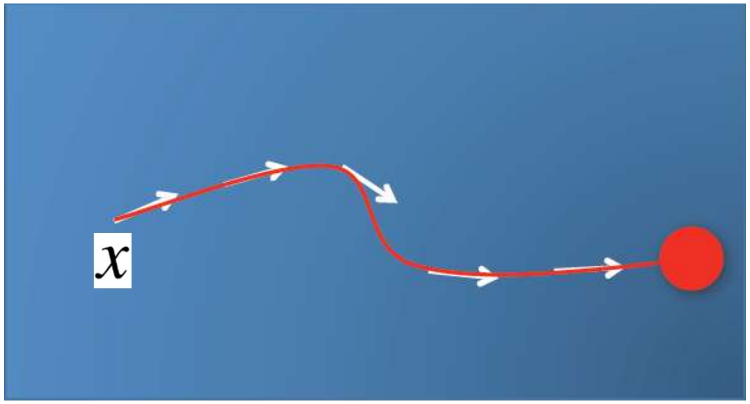
$$\phi_s(\phi_t(x)) = \phi_{s+t}(x)$$

$$\phi : M \times \mathbb{R} \to M, \quad (x,t) \mapsto \phi(x,t).$$

$$\phi_t : M \to M, \quad x \mapsto \phi_t(x).$$

- Unsteady flow? Just fix arbitrary time T

$$\phi(x,t) = x + \int_0^t \mathbf{v}(\phi(x,\tau),\mathrm{T})\,\mathrm{d}\tau$$



(on a general manifold *M*, integration is performed in coordinate charts)

The Flow / Flow Map of a Vector Field (1)

Flow of a *steady (time-independent)* vector field

- Map source position *x* “forward” (*t*>*0*) or “backward” (*t*<*0*) by time *t*

$\boxed{\phi(x,t)}$ $\boxed{\phi_t(x)}$ with $\phi_0(x) = x$

$$\phi_s(\phi_t(x)) = \phi_{s+t}(x)$$

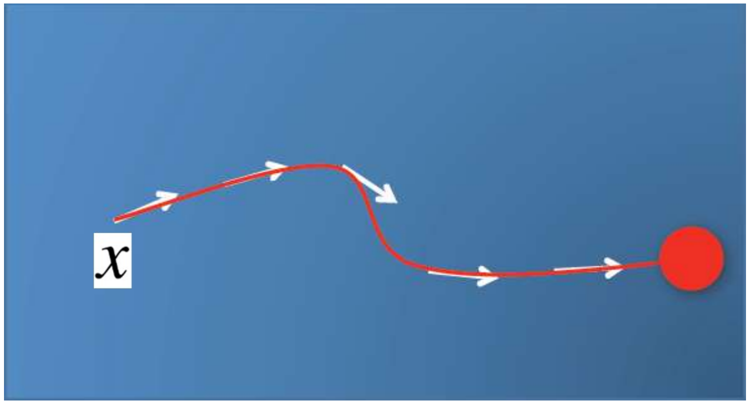
$$\phi : M \times \mathbb{R} \to M, \quad (x,t) \mapsto \phi(x,t).$$

$$\phi_t : M \to M, \quad x \mapsto \phi_t(x).$$

Can write explicitly as function of independent variable *t*, with *position x fixed*

$$t \mapsto \phi(x,t) \qquad t \mapsto \phi_t(x)$$

= stream line going through point *x*



The Flow / Flow Map of a Vector Field (2)

Flow of an *unsteady (time-dependent)* vector field

- Map source position *x* from time *s* to destination position at time *t* (t < s is allowed: map forward or backward in time)

$$\boxed{\psi_{t,s}(x)}$$

with

$$\psi_{t,s}(x) = x + \int_s^t \mathbf{v}\big(\psi_{\tau,s}(x), \tau\big)\,\mathrm{d}\tau$$

$$\psi_{s,s}(x) = x$$

$$\psi_{t,r}\big(\psi_{r,s}(x)\big) = \psi_{t,s}(x)$$

The Flow / Flow Map of a Vector Field (3)

Flow of an *unsteady (time-dependent)* vector field

- Map source position *x* from time *s* to destination position at time *t*
  (t < s is allowed: map forward or backward in time)

$$\boxed{\psi_{t,s}(x)} \qquad \psi_{t,s}(x) = x + \int_s^t \mathbf{v}\big(\psi_{\tau,s}(x), \tau\big)\,\mathrm{d}\tau$$

Can write explicitly as function of t, *with s and x fixed*

$t \mapsto \psi_{t,s}(x)$ → path line

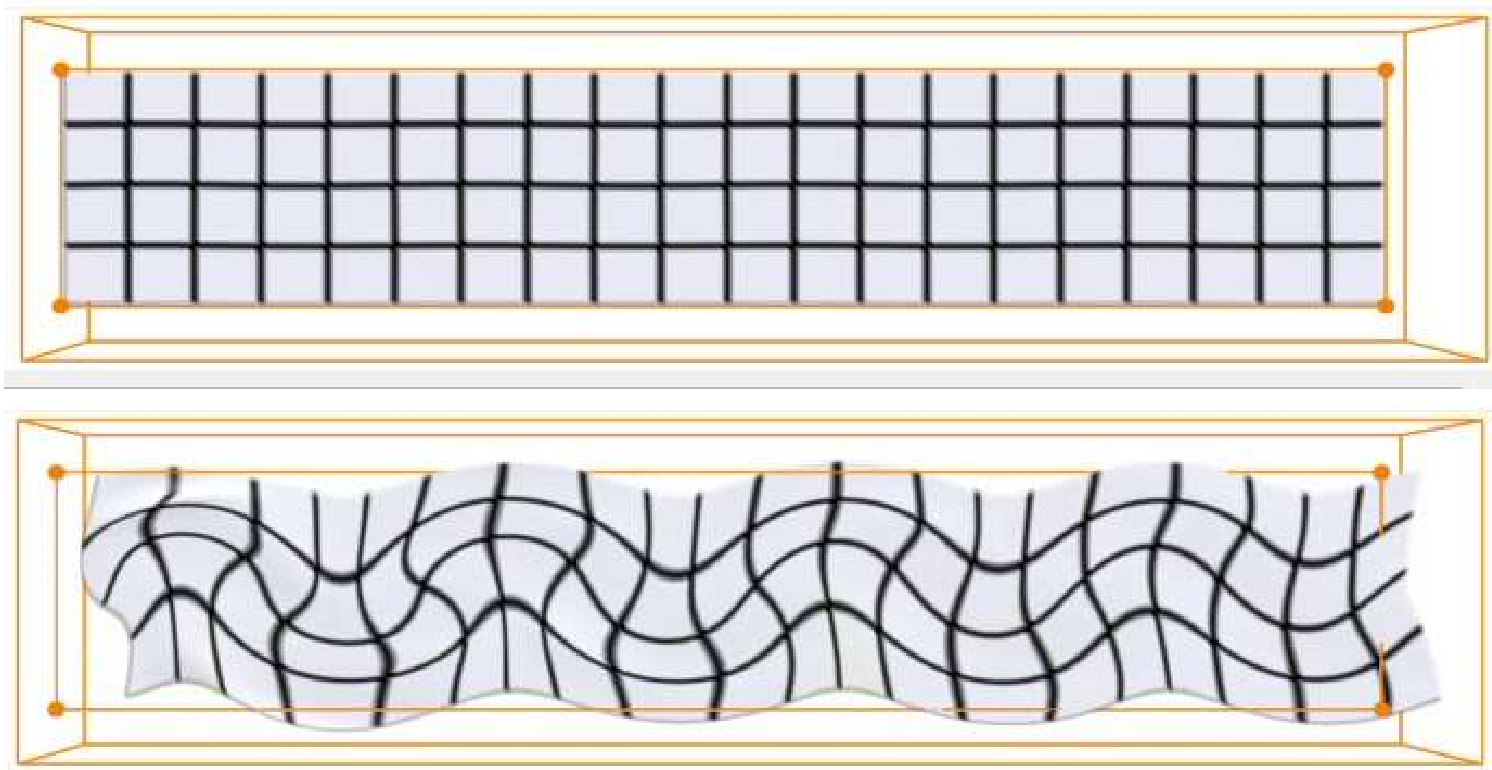
Can write explicitly as function of s, *with t and x fixed*

$s \mapsto \psi_{t,s}(x)$ → streak line

$\psi_{t,s}(x)$ is also often written as ***flow map*** $\phi_t^{\tau}(x)$ (with t:=s and either τ:=t or τ:=t-s)

The Flow / Flow Map of a Vector Field (4)

Can map a whole set of points (or the entire domain) through the flow map (this map is a *diffeomorphism*): $\boxed{t \mapsto \psi_{t,s}(U)}$



U

$= \psi_{s,s}(U)$

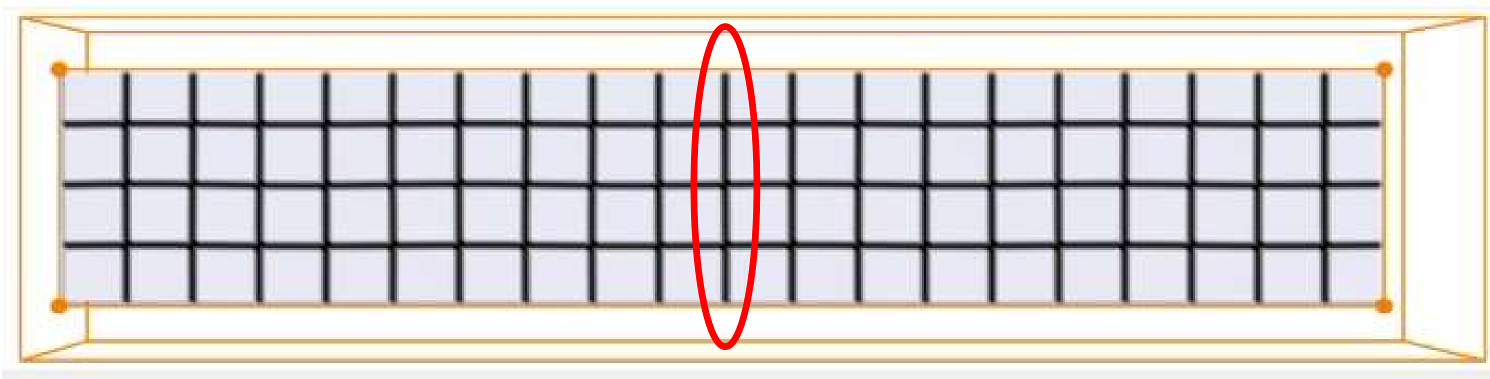
$\psi_{t,s}(U)$

(this is a *time surface!*)

The Flow / Flow Map of a Vector Field (5)

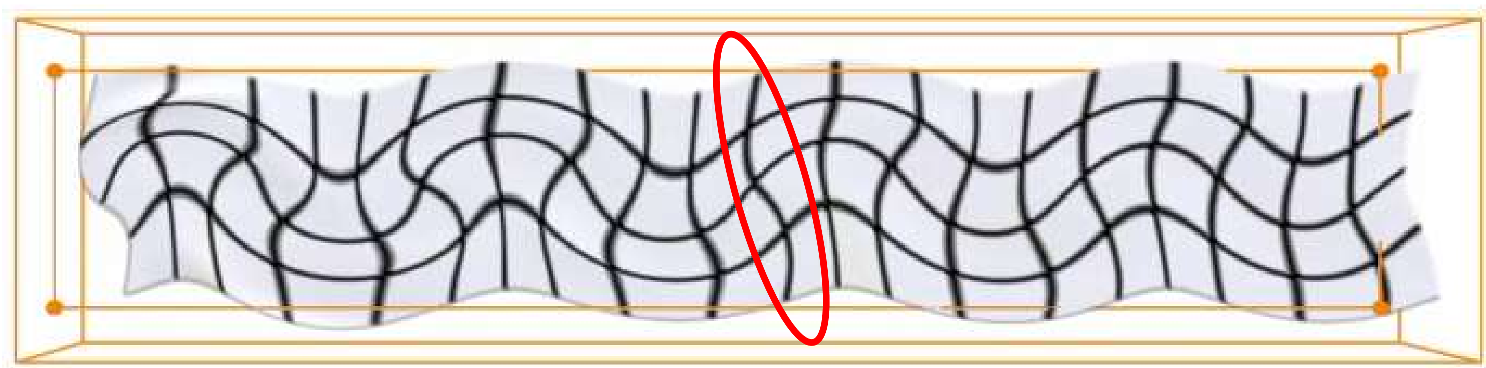
Time line: Map a whole curve from one fixed time (s) to another time (t)

$$\boxed{t \mapsto \psi_{t,s}(c(\lambda))}$$



$c(\lambda)$

$= \psi_{s,s}(c(\lambda))$

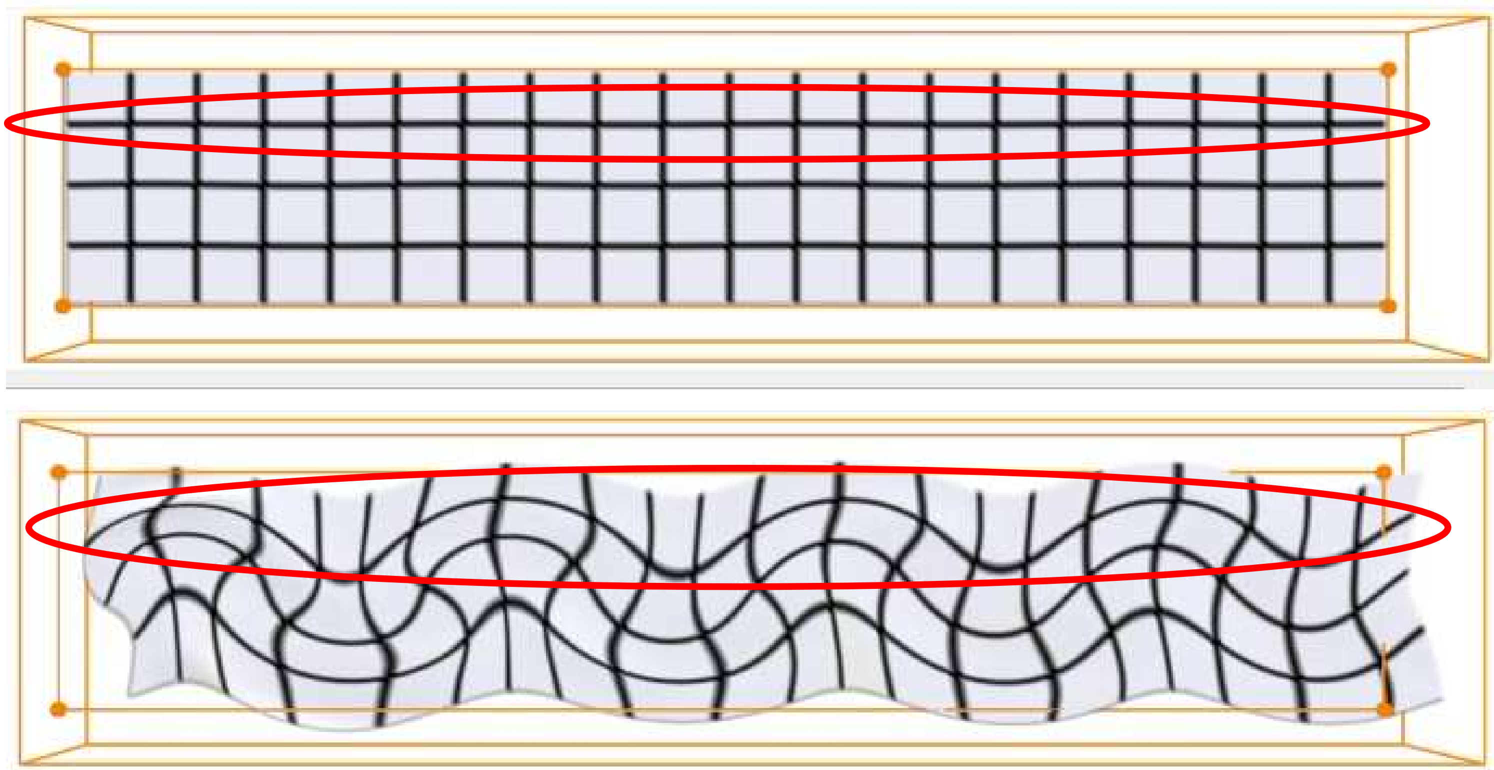


$\psi_{t,s}(c(\lambda))$

The Flow / Flow Map of a Vector Field (5)

Time line: Map a whole curve from one fixed time (*s*) to another time (*t*)

$$\boxed{t \mapsto \psi_{t,s}(c(\lambda))}$$



$$c(\lambda)$$

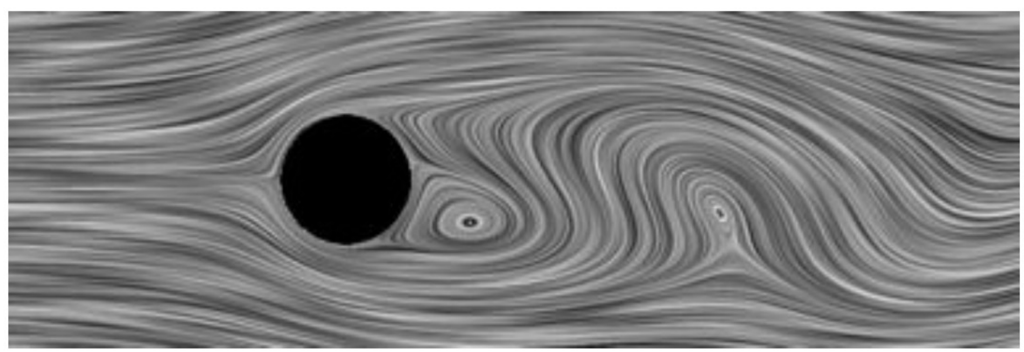
$$=\psi_{s,s}(c(\lambda))$$

$$\psi_{t,s}(c(\lambda))$$

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

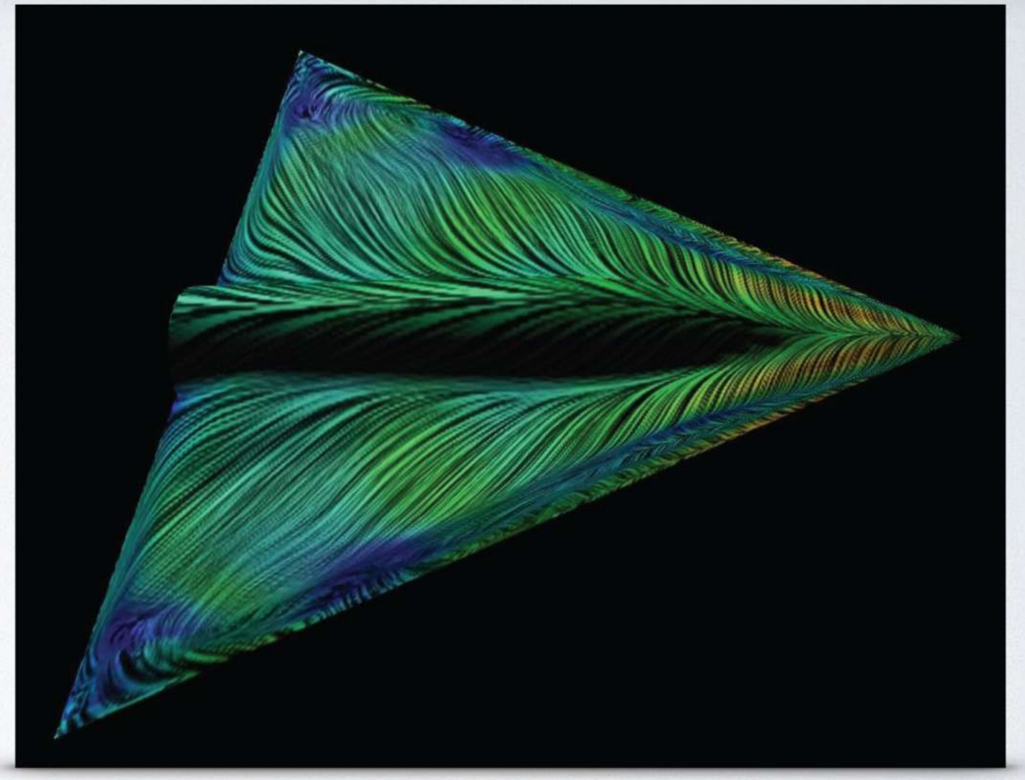


Line Integral Convolution I

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

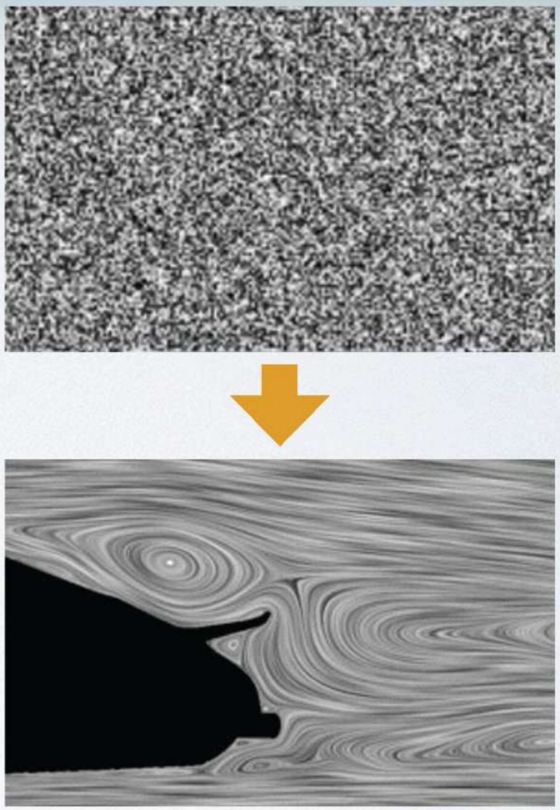
Line Integral Convolution I

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



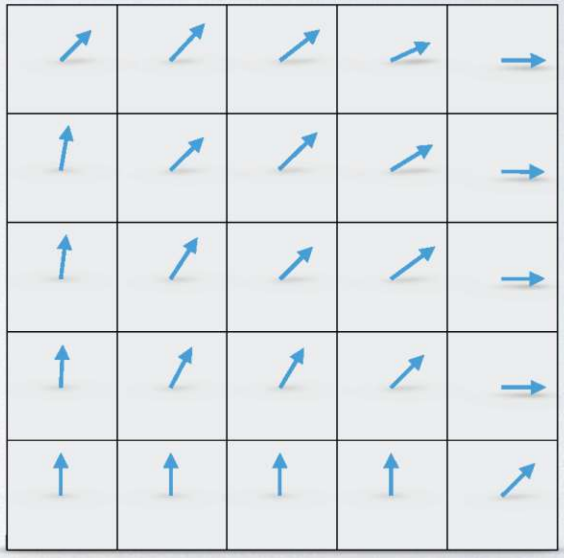
Line Integral Convolution II

- 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)



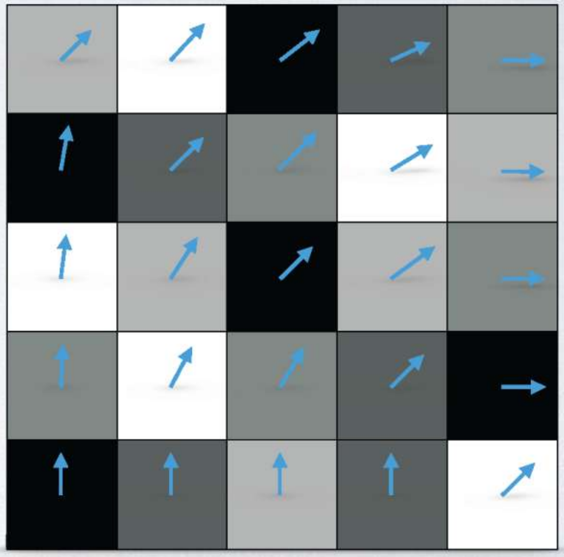
Line Integral Convolution III

- 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



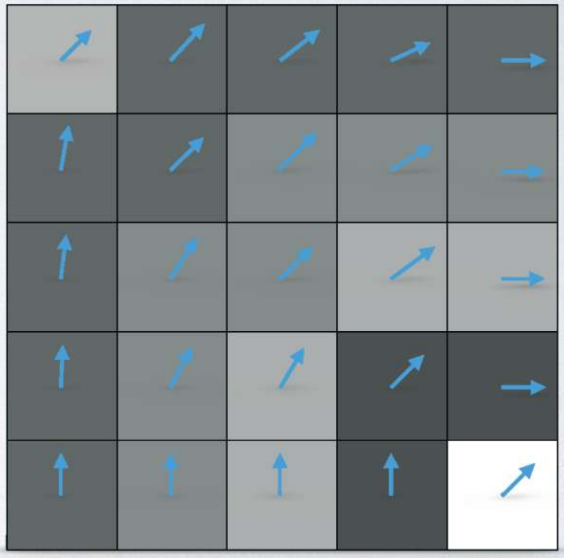
Line Integral Convolution III

- 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



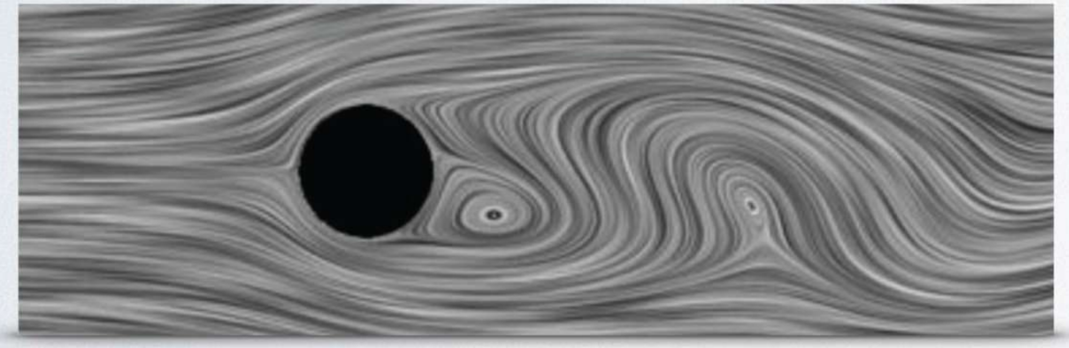
Line Integral Convolution III

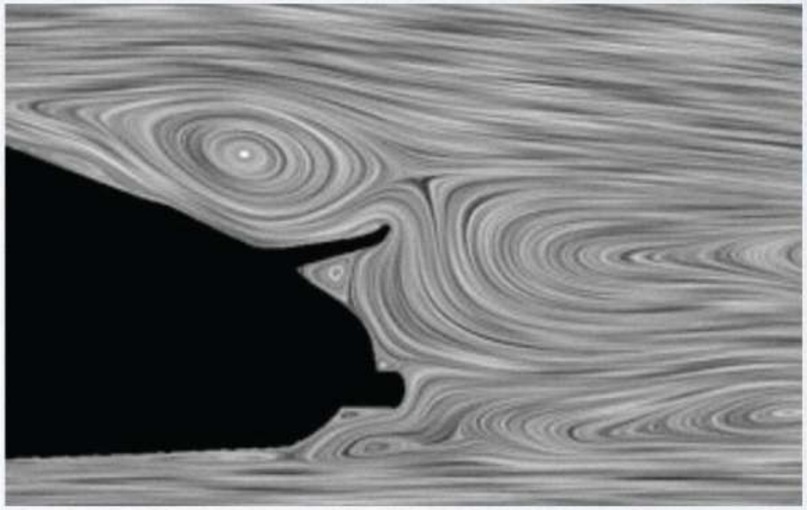
- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture
    ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



Line Integral Convolution IV

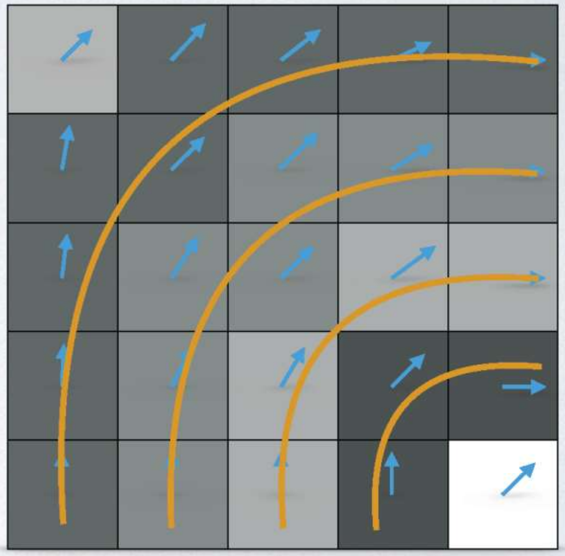
- Look of 2D LIC images
  - intensity along stream lines shows high correlation
  - no correlation between neighboring stream lines





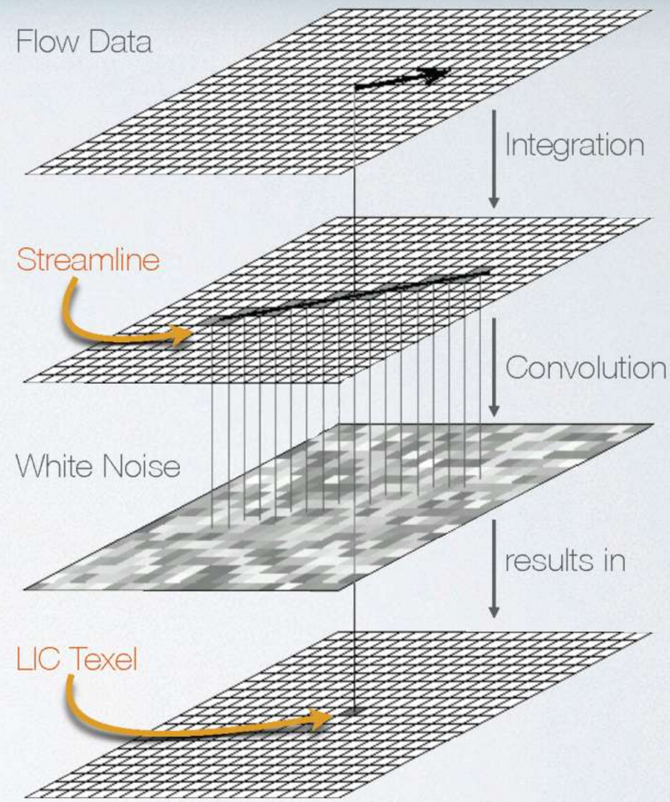
LIC Approach - Goal

- 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:

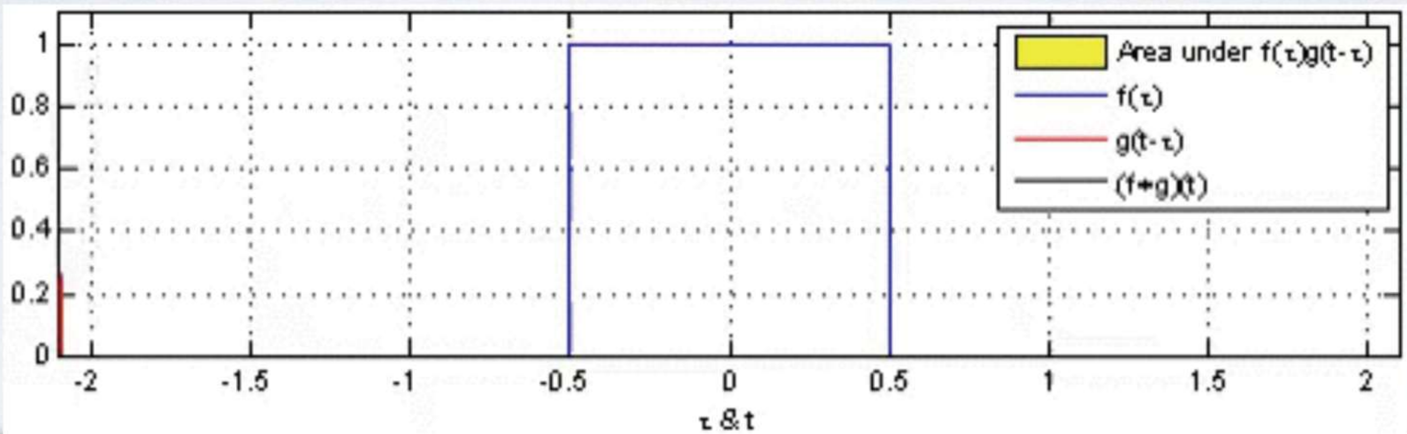
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067
0.00002292	0.00078634	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00038771	0.01330373	0.11098164	**0.22508352**	0.11098164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

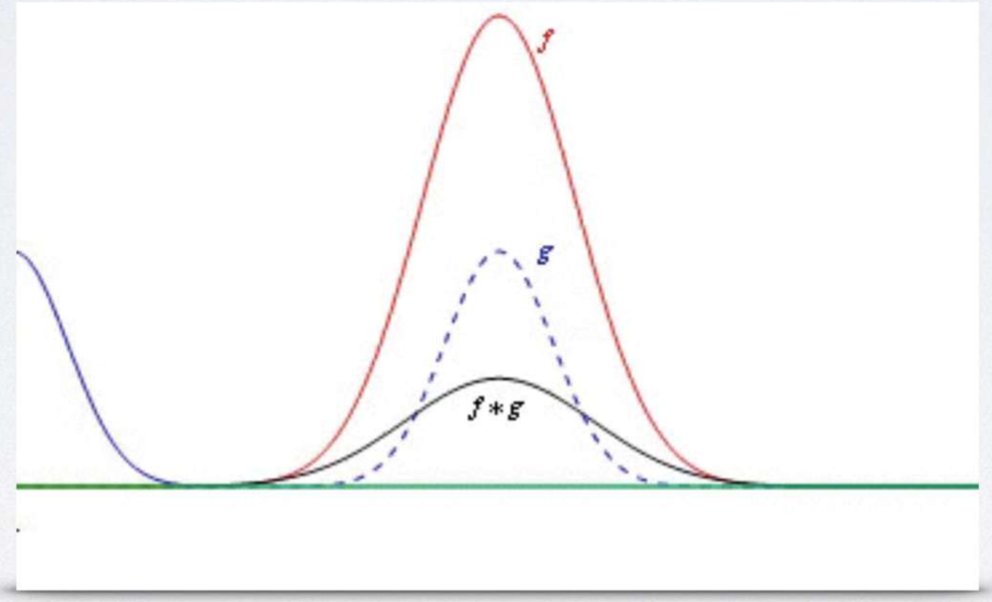
LIC Approach - 1D Convolution I

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



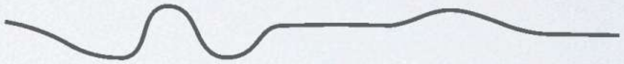
LIC Approach - 1D Convolution III

$k(x)$ 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

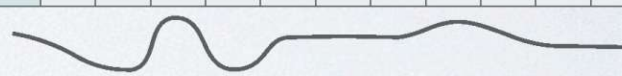
$k(x)$ convolution kernel

¼	½	¼

common section L

$f(x)$ 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

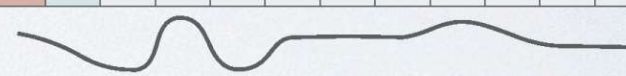
$k(x)$ convolution kernel

¼	½	¼

common section L

$f(x)$ original signal

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



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											

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

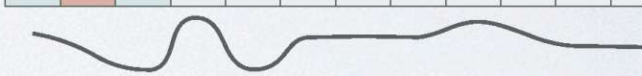
LIC Approach - 1D Convolution III

$k(x)$ 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

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

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

3	2										

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

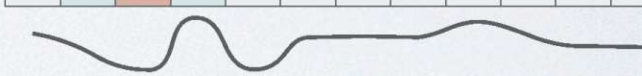
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

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



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

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

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

3	2										

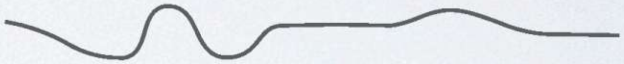
LIC Approach - 1D Convolution III

$k(x)$ 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	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

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

3	2	2½	4								

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

LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

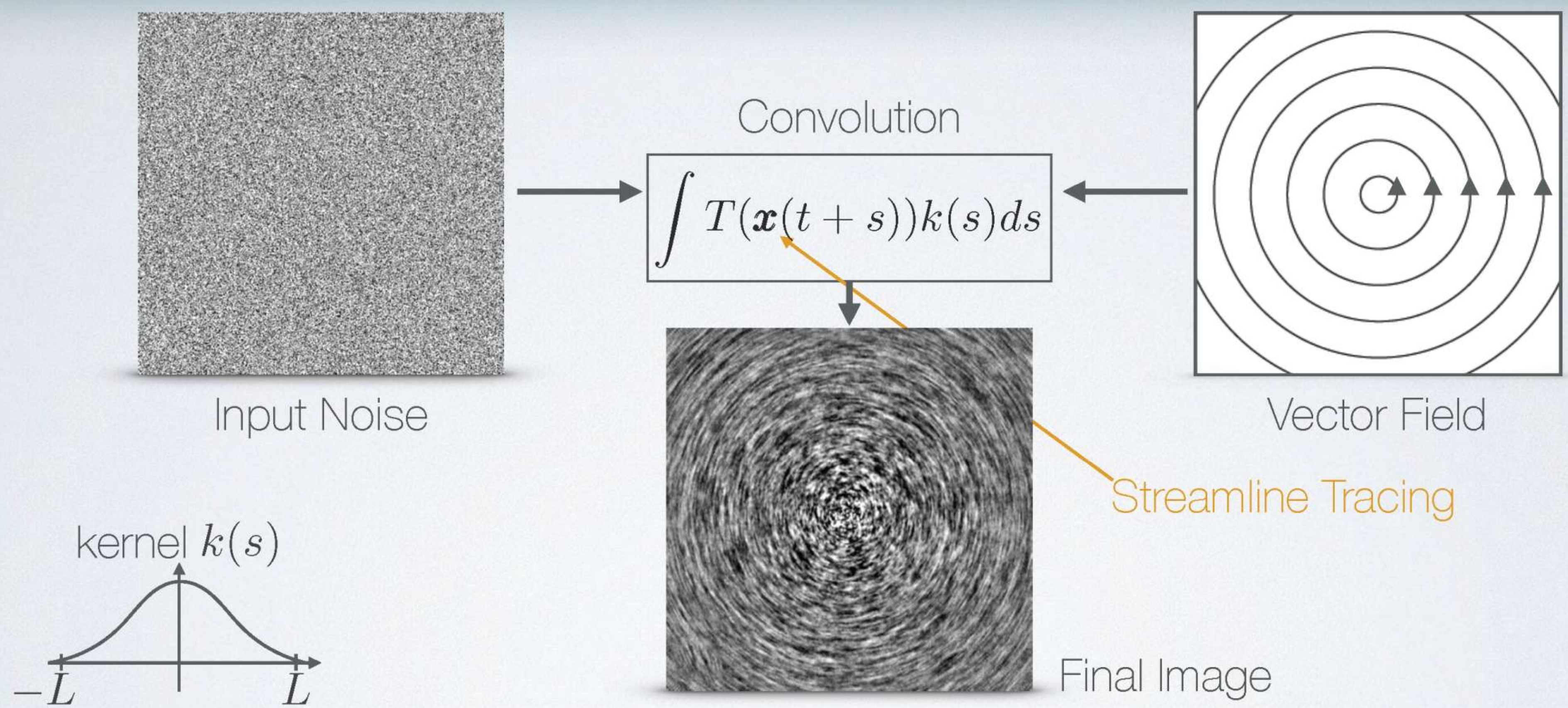
4	2	0	8	0	4	4	4	6	4	2	2	

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

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

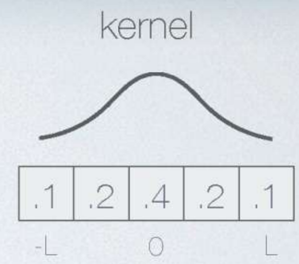
3	2	2½	4	3	3	4	4½	5	4	2½	1½

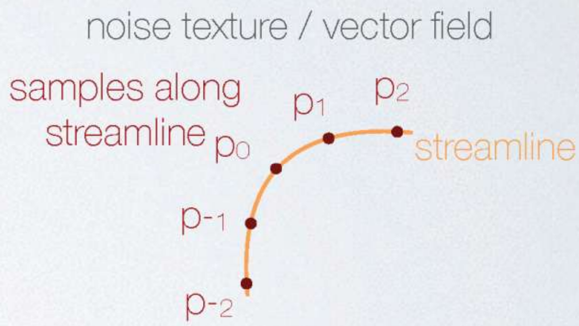
LIC Approach - 1D Convolution IV



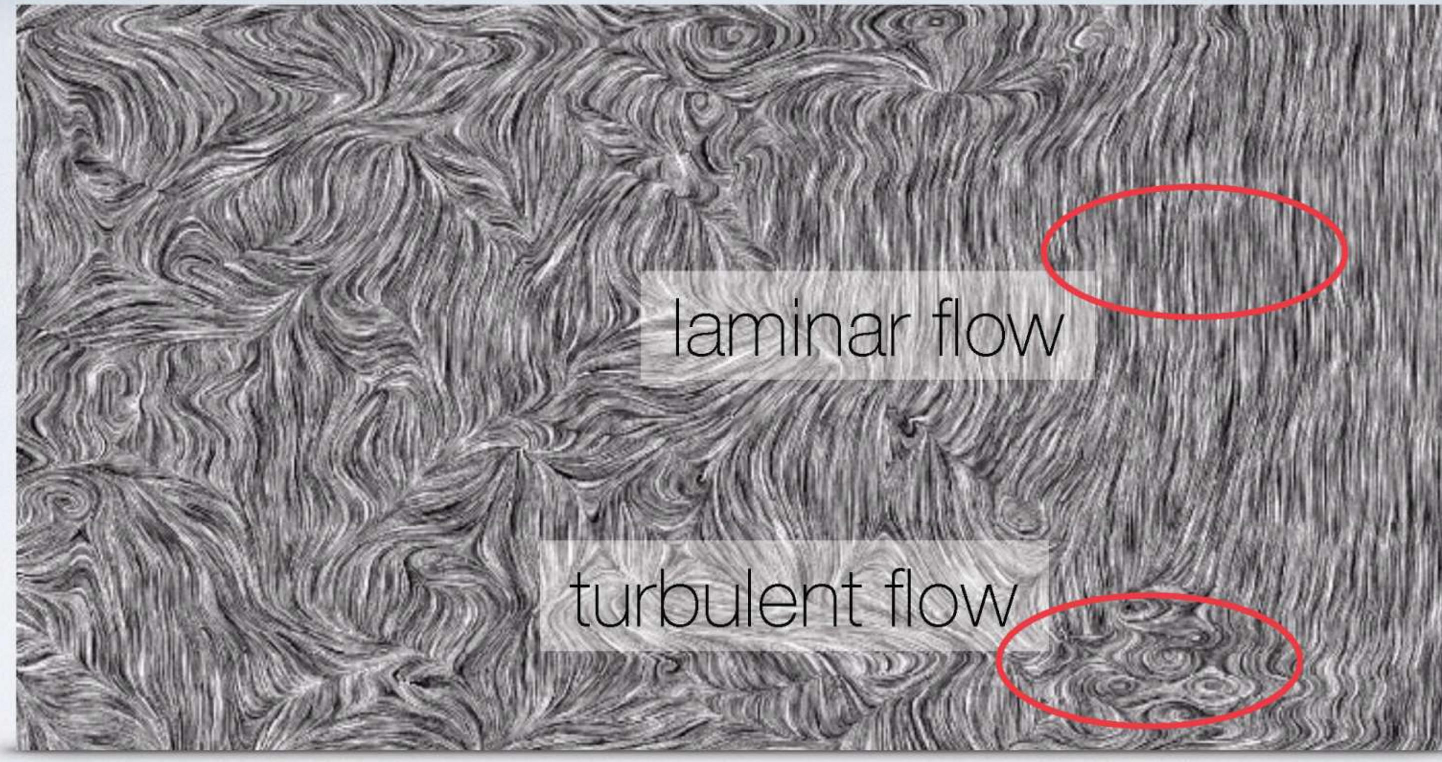
LIC - Algorithm

```
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



Thank you.

Thanks for material

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