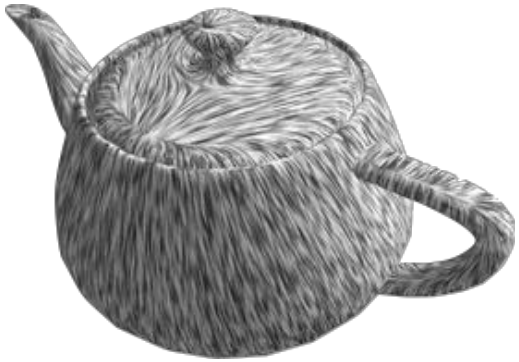
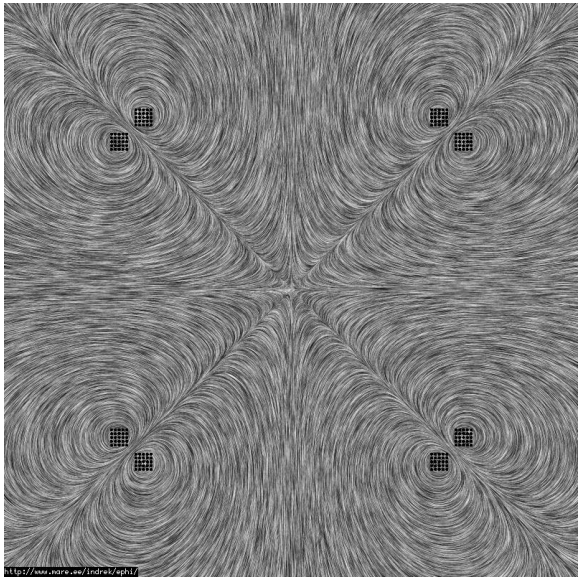
very important

- **LIC – Line Integral Convolution**
  (Cabral/Leedom, Siggraph 1993)

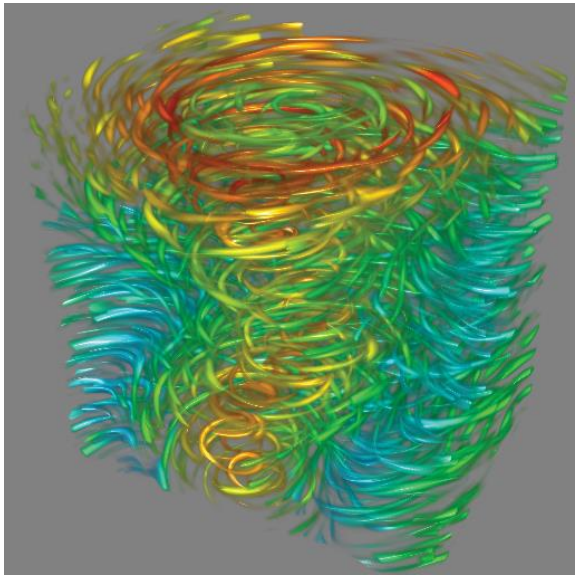
- A global method to visualize vector fields





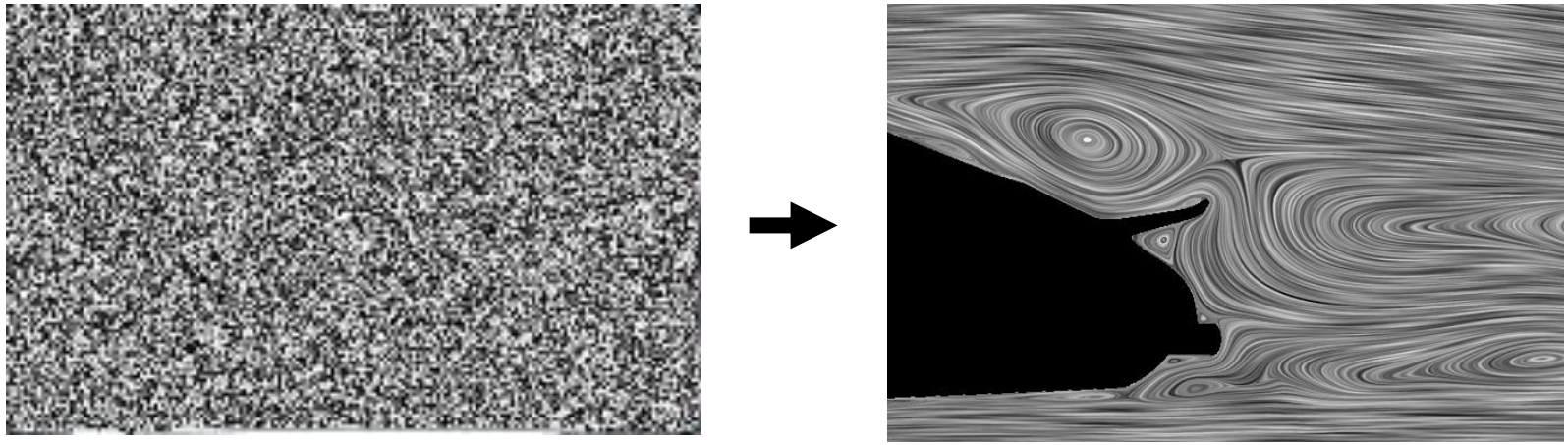
2D vector field

vector field on surface
(often called 2.5D)

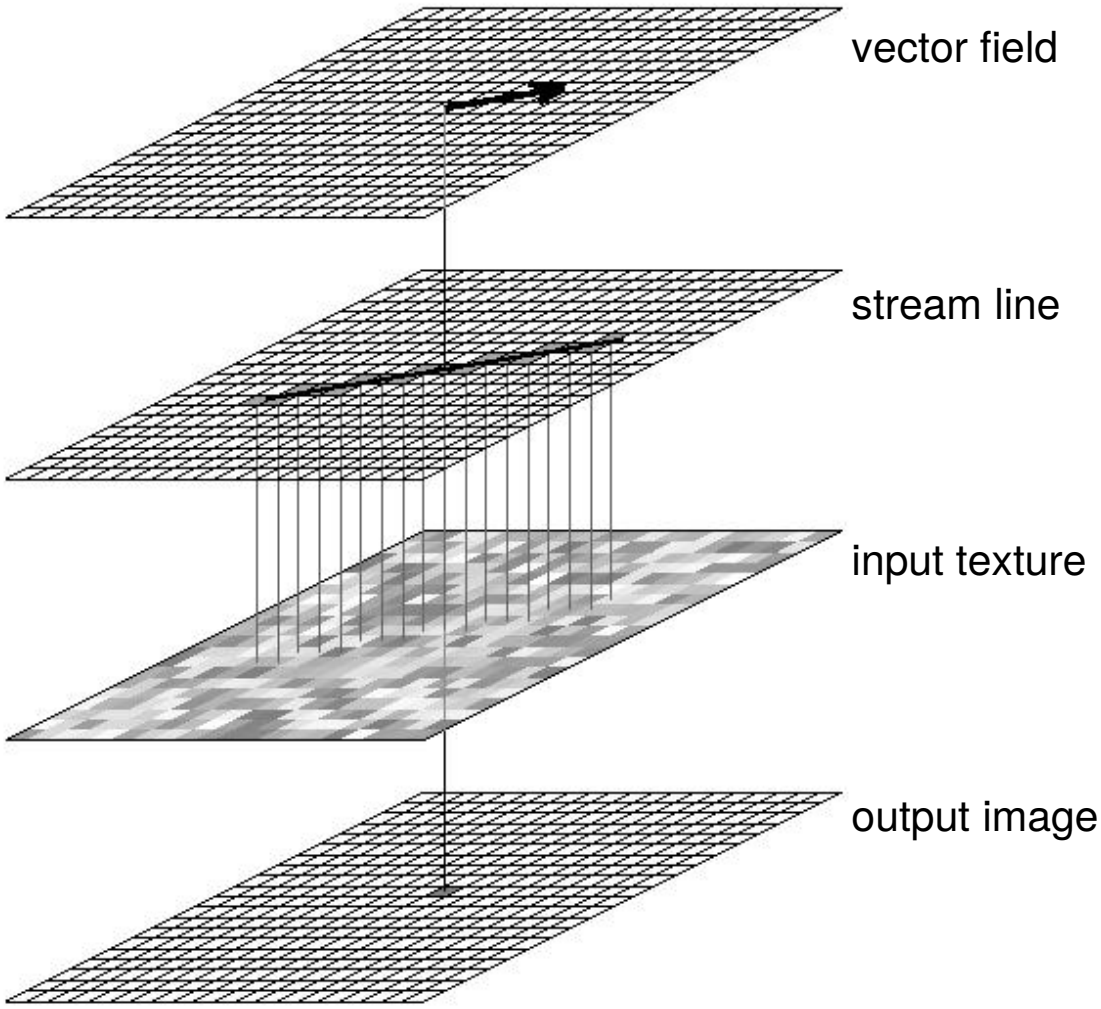


3D vector field

- Idea of Line Integral Convolution (LIC)
  - **Global** visualization technique; uses stream lines
  - Start with a **random texture**
  - **Smear** out this texture along the stream lines in a vector field
    - Results in
      - **low correlation** of intensity values **between** neighboring lines,
      - **but high correlation along** them



- Algorithm for 2D LIC
  - **Convolve** a random tex
    along the stream lines

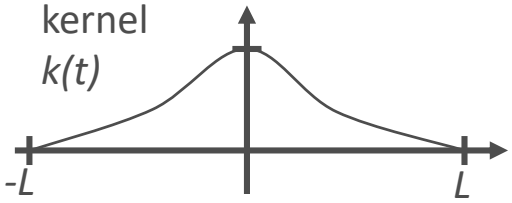


- Algorithm for 2D LIC
  - Let $t \rightarrow \Phi_0(t)$ the stream line containing the point (x_0,y_0)
  - T(x,y) is the randomly generated input texture
  - Compute the pixel intensity as:

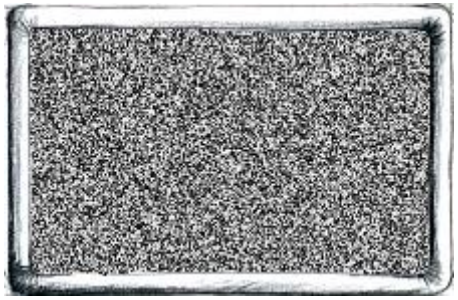
$$I(x_0, y_0) = \int_{-L}^{L} k(t) \cdot T(\varphi_0(t))\, dt$$

*convolution of the input texture T along the stream line $\Phi_0(t)$ with a kernel k(t), which has a length of 2*L.*

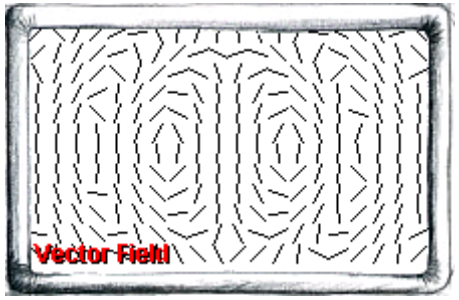
  - Kernel:
    - Finite support [-L,L]
    - Normalized
    - Symmetric



$$\int_{-L}^{L} k(t)\, dt = 1$$

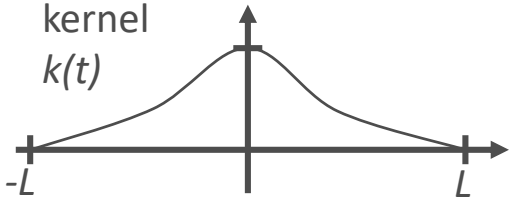


Input texture
(noise)

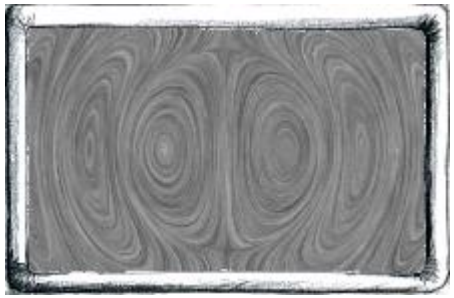


Vector field

Convolution



$$\int_{-L}^{L} k(t)dt = 1$$

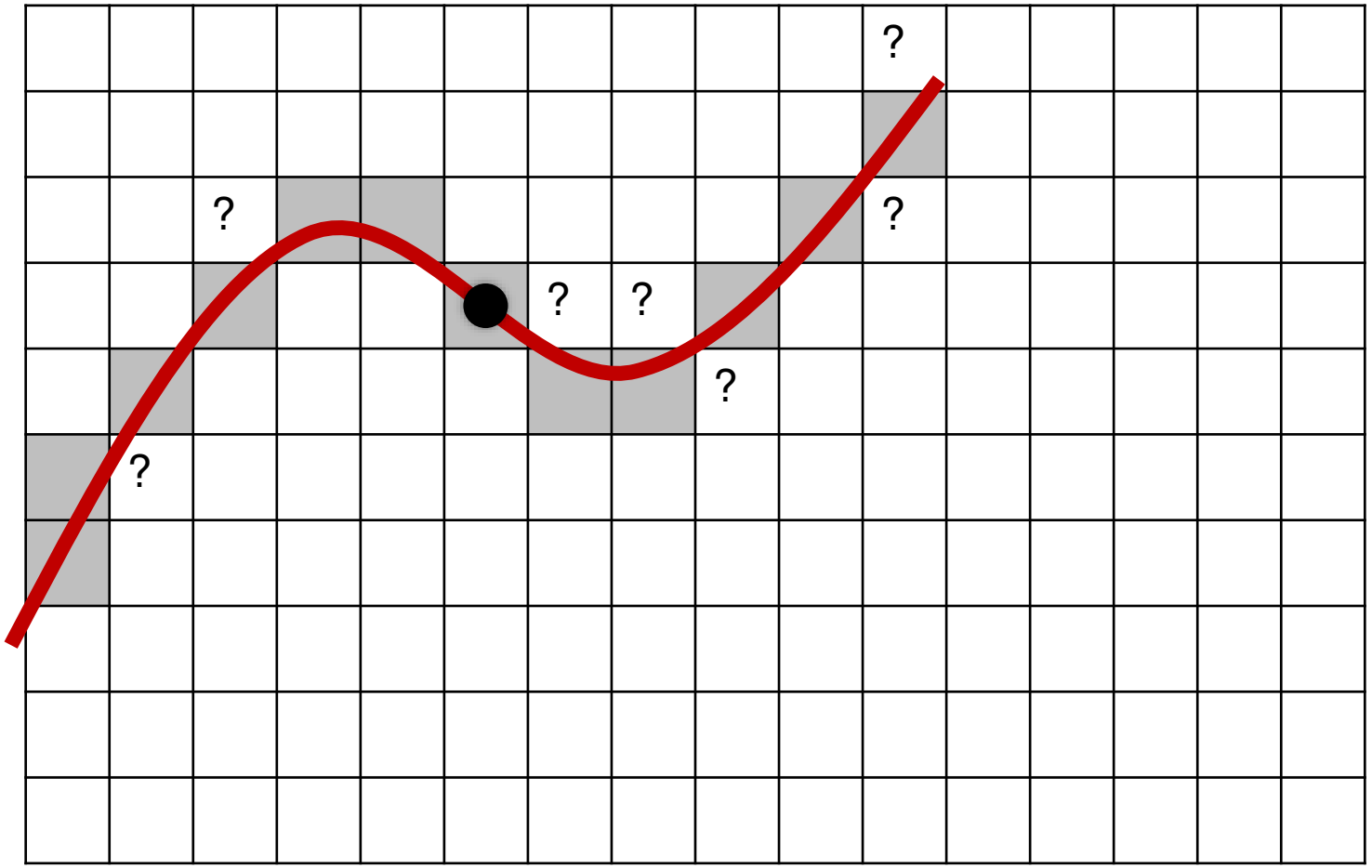


Final image

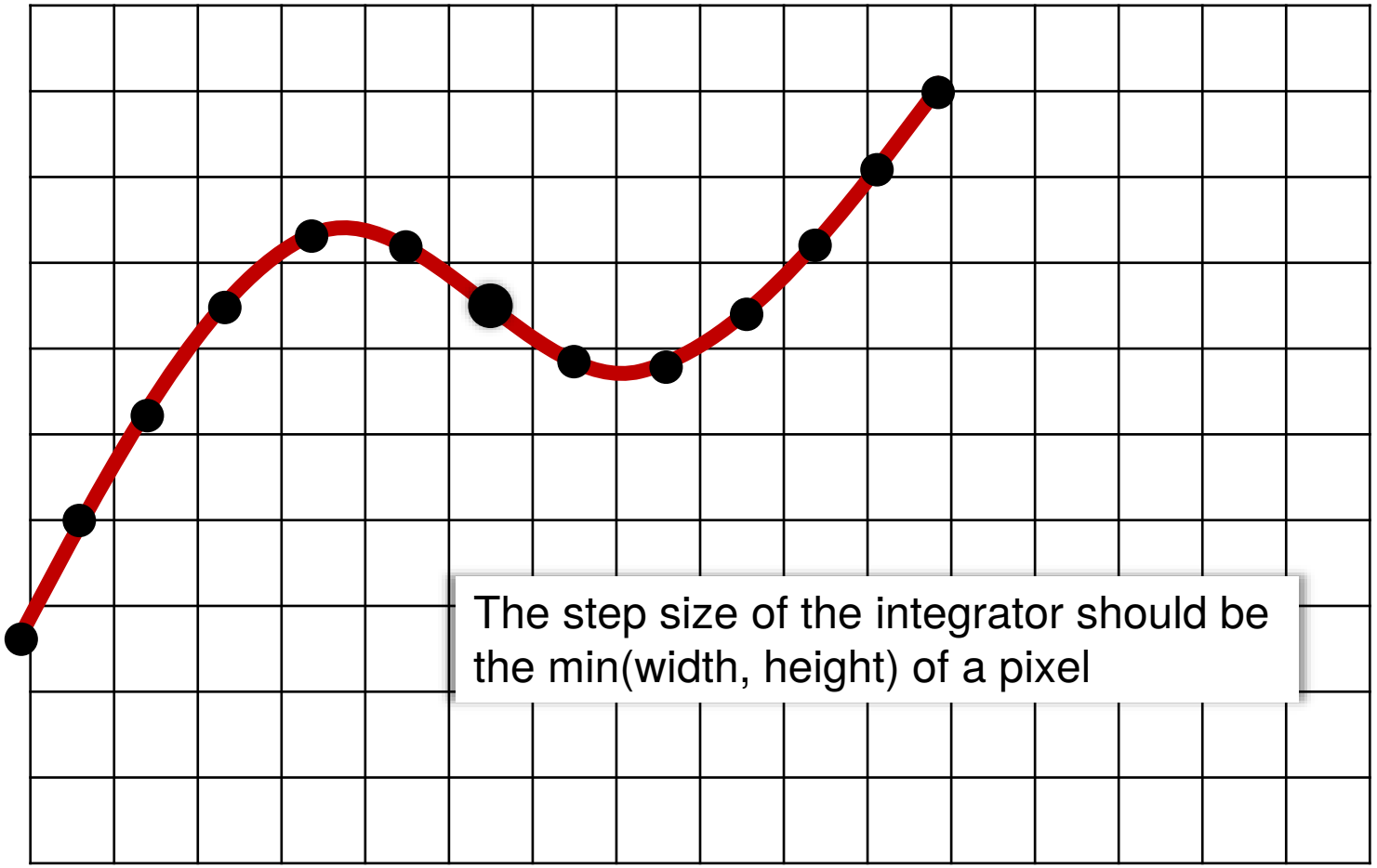
- Technical details:
- Rasterization of the stream line
- Type of kernel
- Length of kernel
- Contrast
- Performance
- Reproducibility

- Technical details: Rasterization of the stream line

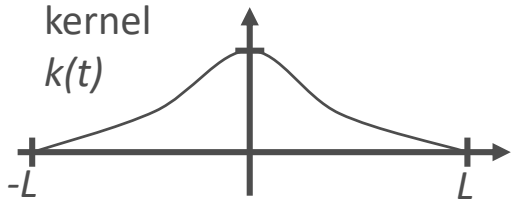
- Technical details: Rasterization of the stream line
  - How to do it?
  - Consider aliasing?



- Technical details: Rasterization of the stream line
  - Easiest way: sample the arc-length-parameterized stream line equidistantly & evaluate the random texture as a bilinear scalar field



- Technical details: Type of kernel

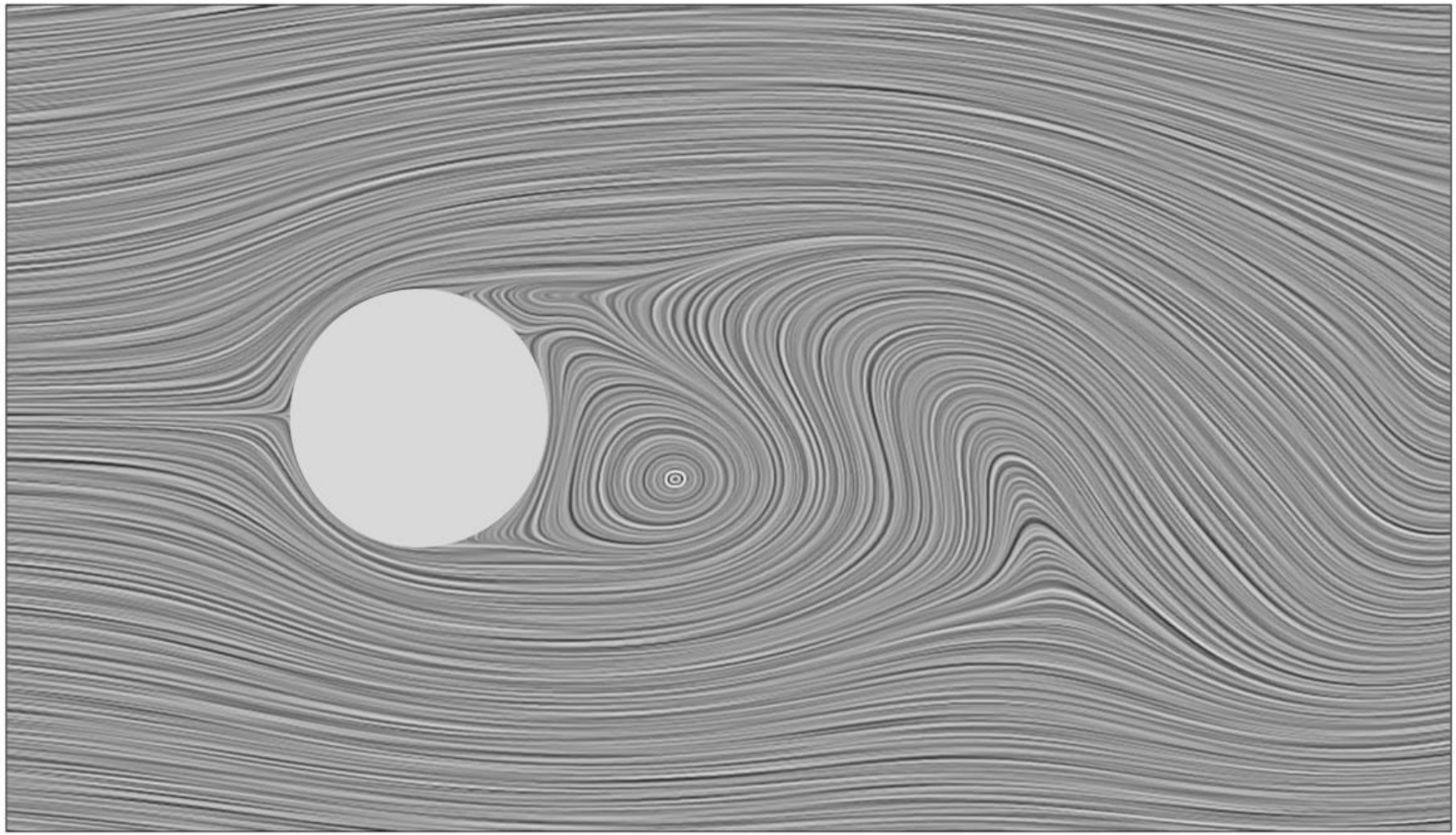


$$\int_{-L}^{L} k(t)dt = 1$$

- Gaussian kernel
- Triangle kernel
- Box kernel
  - Result is the arithmetic mean of all collected pixel values.

- Technical details: Length of kernel

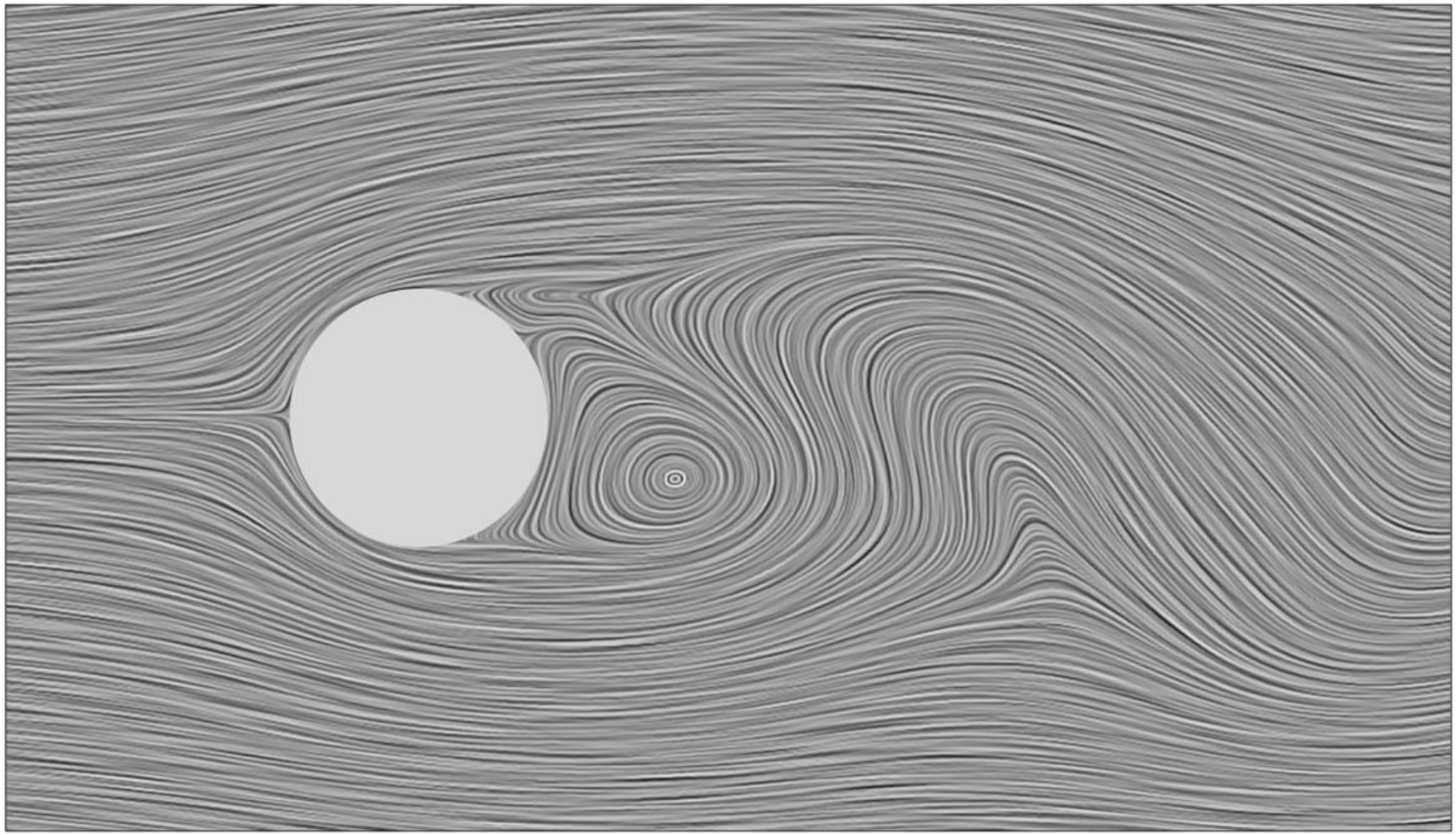
- Longer kernel leads to longer lines, and less contrast
- Smaller kernel leads to shorter lines, and more contrast



Filter length influences the quality of LIC images
filter length = 100

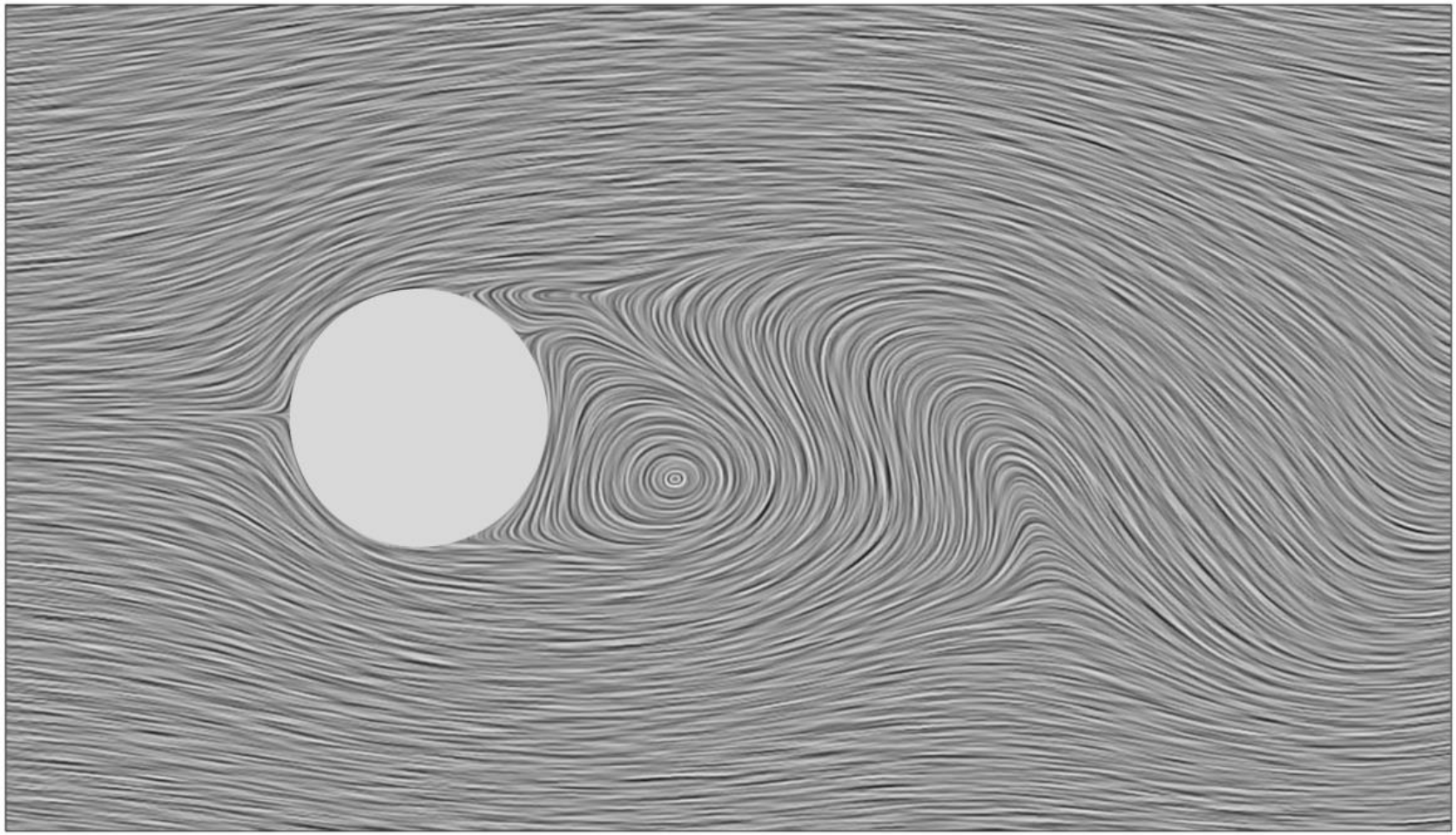
2D flow behind a cylinder

show example in Amira



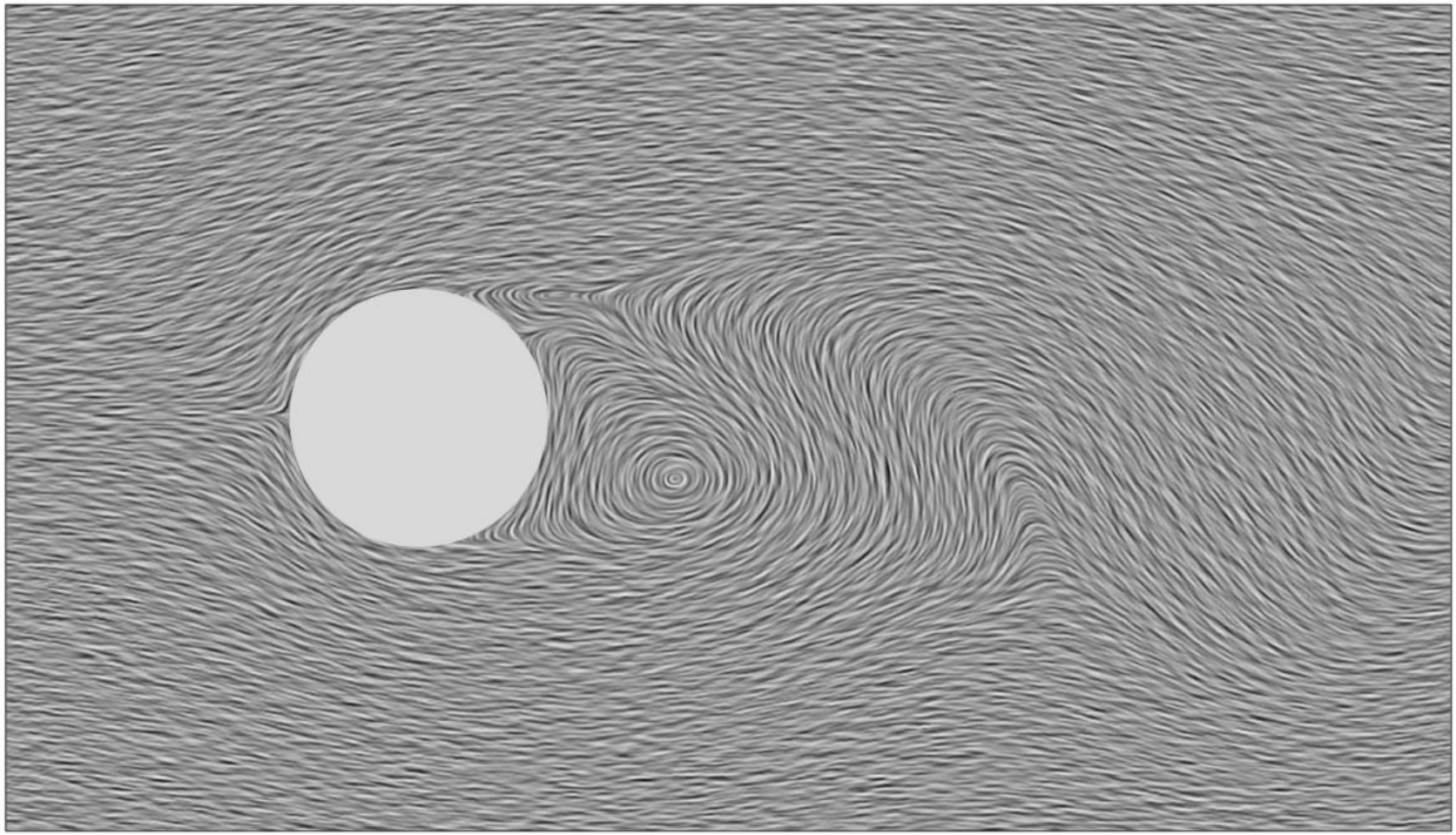
Filter length influences the quality of LIC images
filter length = 50

2D flow behind a cylinder



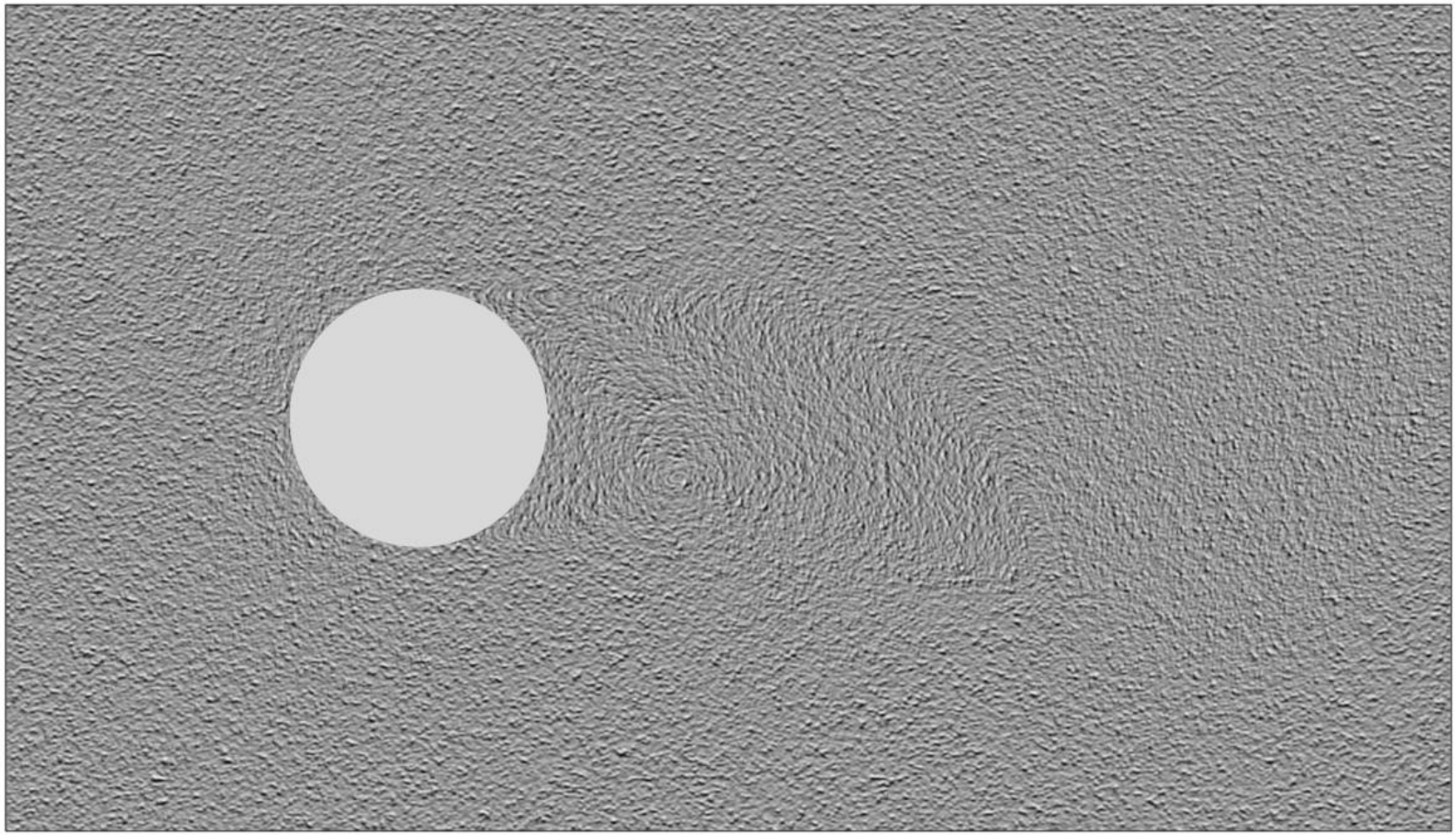
Filter length influences the quality of LIC images
filter length = 25

2D flow behind a cylinder



Filter length influences the quality of LIC images
filter length = 10

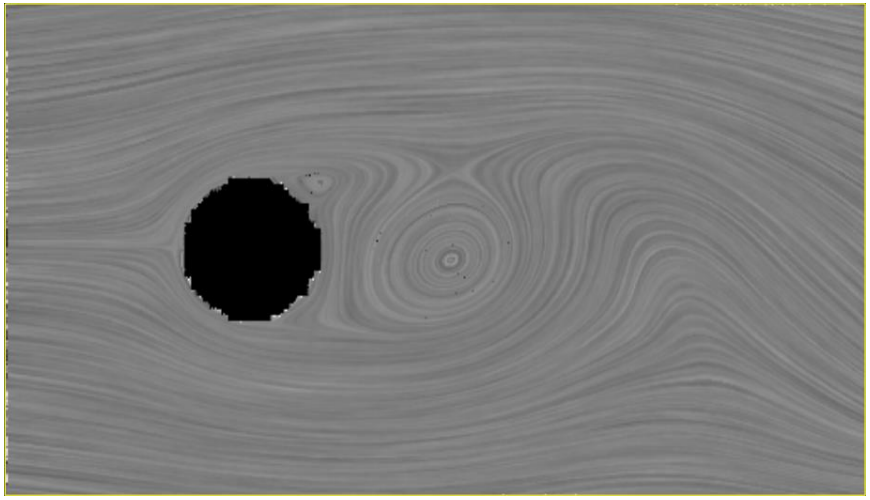
2D flow behind a cylinder



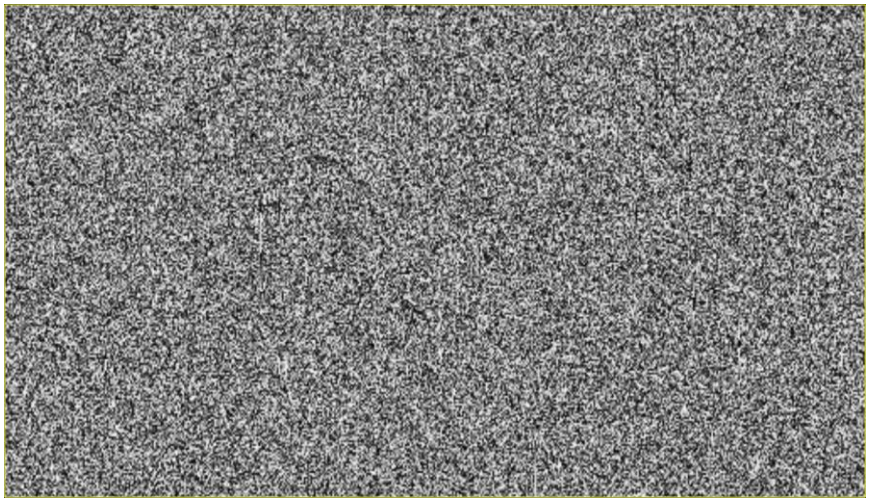
Filter length influences the quality of LIC images
filter length = 1

2D flow behind a cylinder

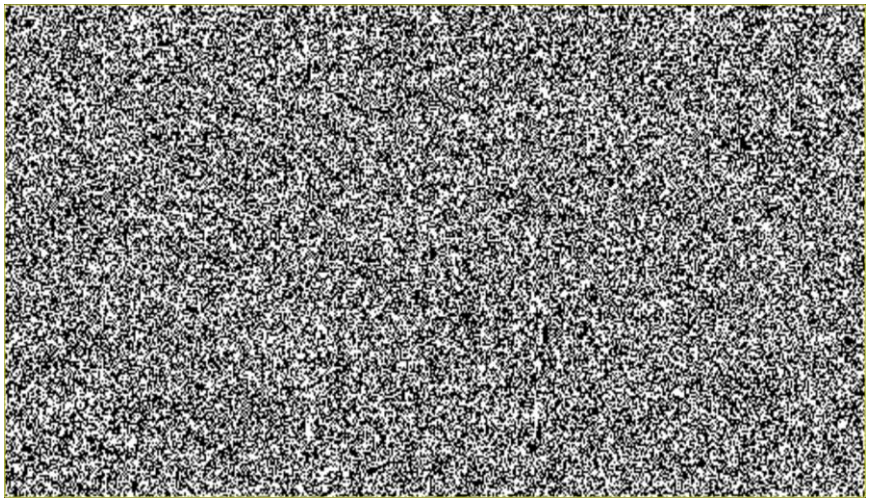
- Technical details: Contrast
- Problem:
  Convolution reduces contrast of the grayscale image
- Solutions:
  - Use black-white image as input
  - Enhance contrast after convolution



- Technical details: Contrast

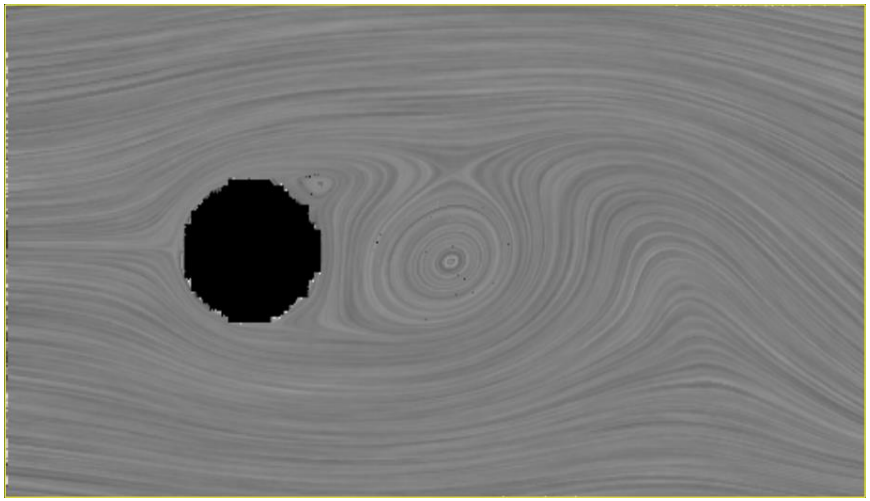


grayscale input texture

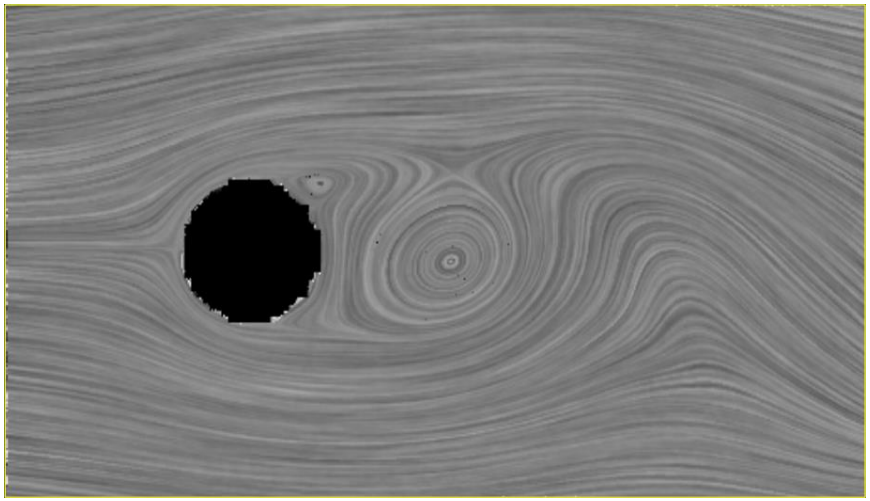


black-white input texture

- Technical details: Contrast

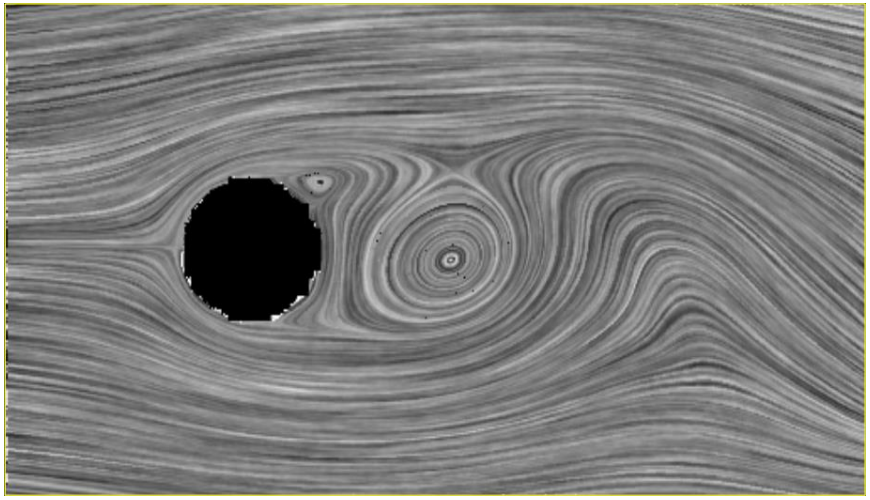


grayscale input texture



black-white input texture

enhanced contrast



- Technical details: Enhancing the contrast
- Compute mean and standard deviation of the convolved texture

only for all non-black pixels !

$$\mu = \frac{\sum_i^n p_i}{n} \qquad P = \sum_i^n p_i^2 \qquad \sigma = \sqrt{\frac{P - n\mu^2}{n-1}}$$

arithmetic mean *standard deviation*

- Adjust the mean and standard deviation to desired values

$$\sigma' \, , \, \mu' \quad \Leftarrow \text{ new desired values}$$

$$f = \frac{\sigma'}{\sigma} \qquad p_i' = \mu' + f(p_i - \mu)$$

stretching factor
(restrict to a maximum value !)

new grayscale pixel values

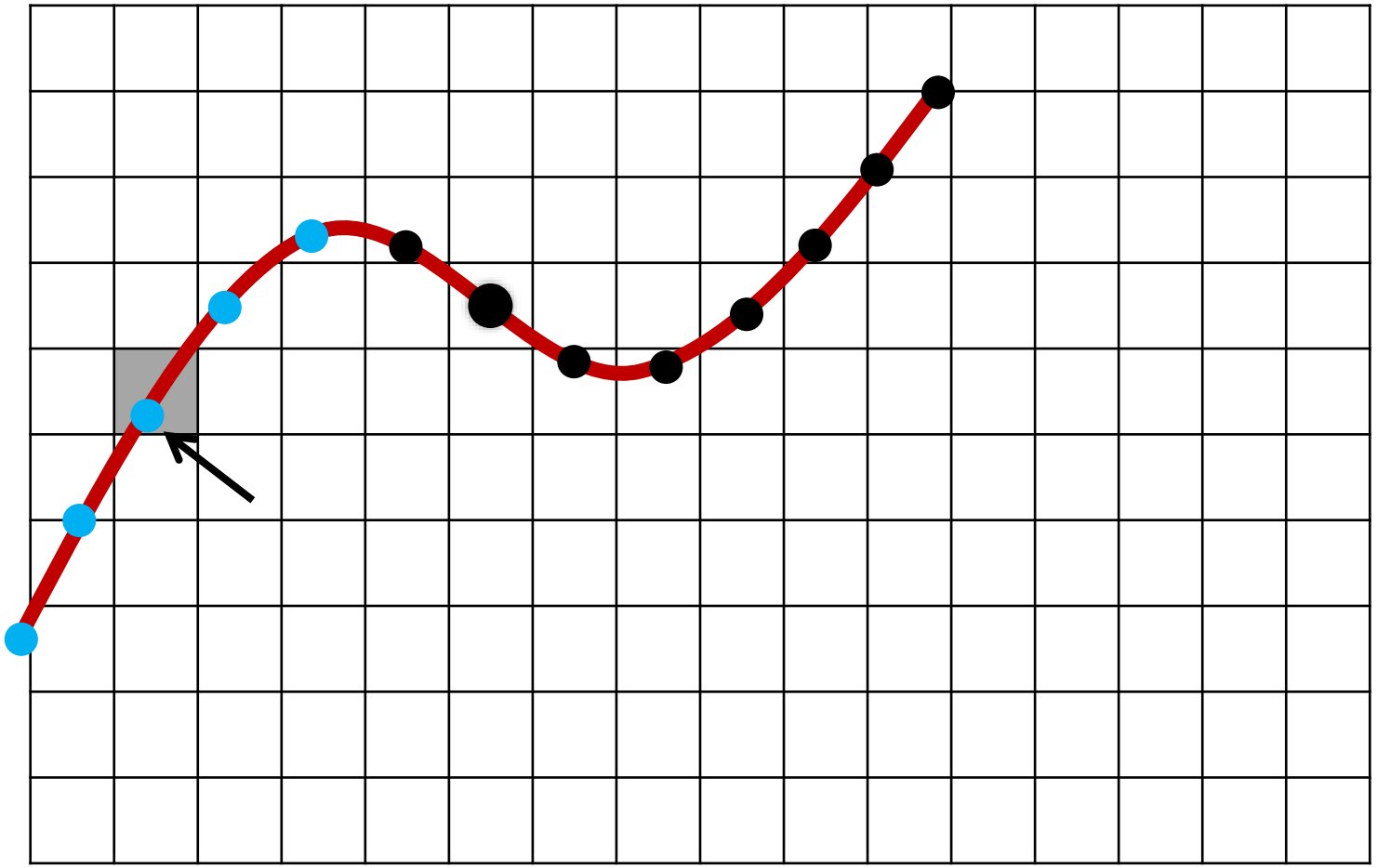
- Technical details: Enhancing the contrast
- Good defaults for desired mean and standard deviation:
  - considering a range [0, 1] with 0=black and 1=white
  - mean ➔ 0.5
  - standard deviation ➔ 0.1

- Technical details: Performance ➔ FastLIC
- Fast Line Integral Convolution (FastLIC) Stalling/Hege 1995
- ➔ Increase performance of LIC
  - apply convolution on all pixels on a particular stream line
  - store all pixels for which convolution is carried out
  - for untouched pixel: start a new tangent curve from there
- Significant speed up in comparison to original LIC
- Current implementations of LIC follow these ideas instead of the original algorithm.

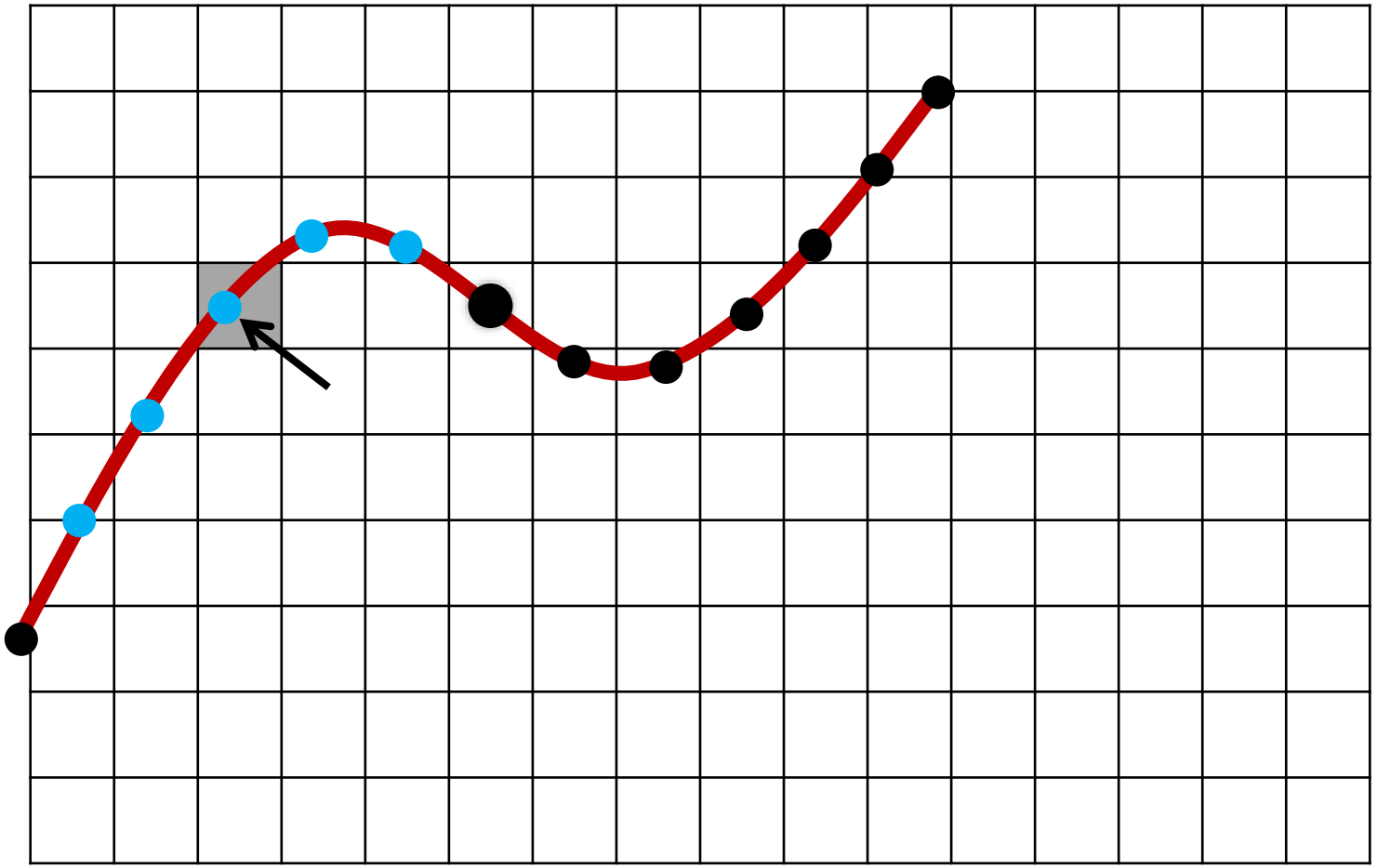
- Technical details: Performance ➔ FastLIC
- General idea: exploit the coherence along the stream line

- Classic LIC:
  Visit every pixel, Integrate stream line, Convolve
- FastLIC:
  Visit every pixel that has not yet been visited
  Integrate stream line for as long as possible
  Convolve for every pixel covered by the stream line

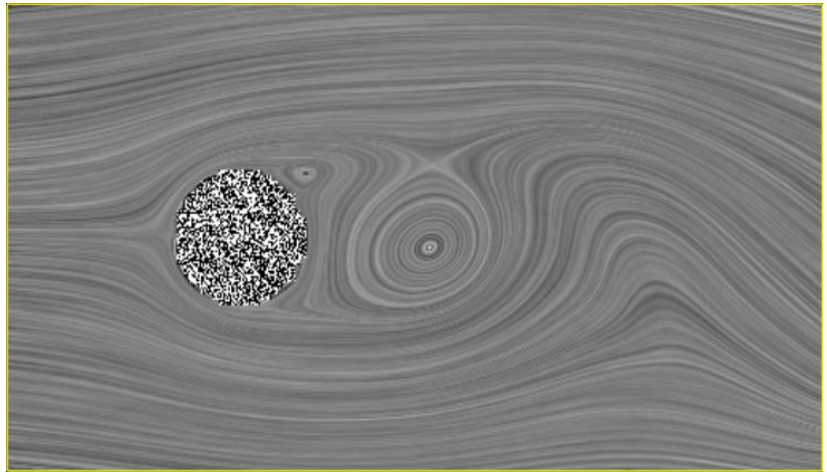
- Technical details: Performance ➔ FastLIC



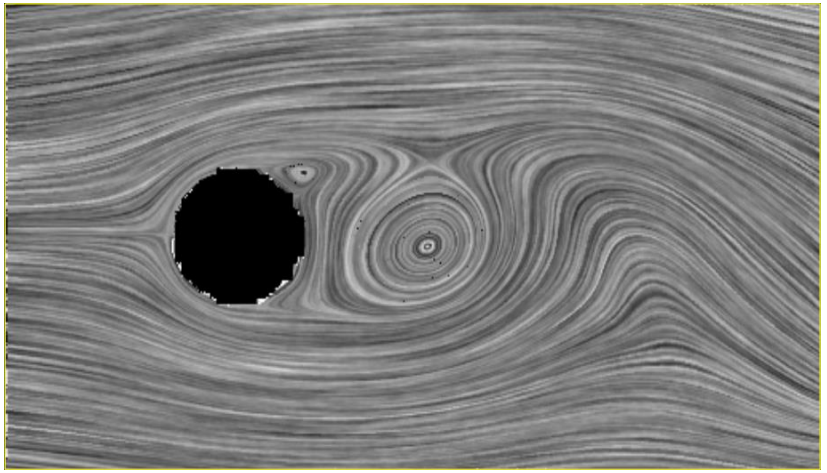
- Technical details: Performance ➔ FastLIC
  - Shifting the kernel: fast update possible for most kernels



- Technical details: Performance ➔ FastLIC
- Significantly fewer stream lines required
- Significantly better anti-aliasing



131072 stream lines were required to compute the classic LIC texture. 9 seconds for 512x256 texture.

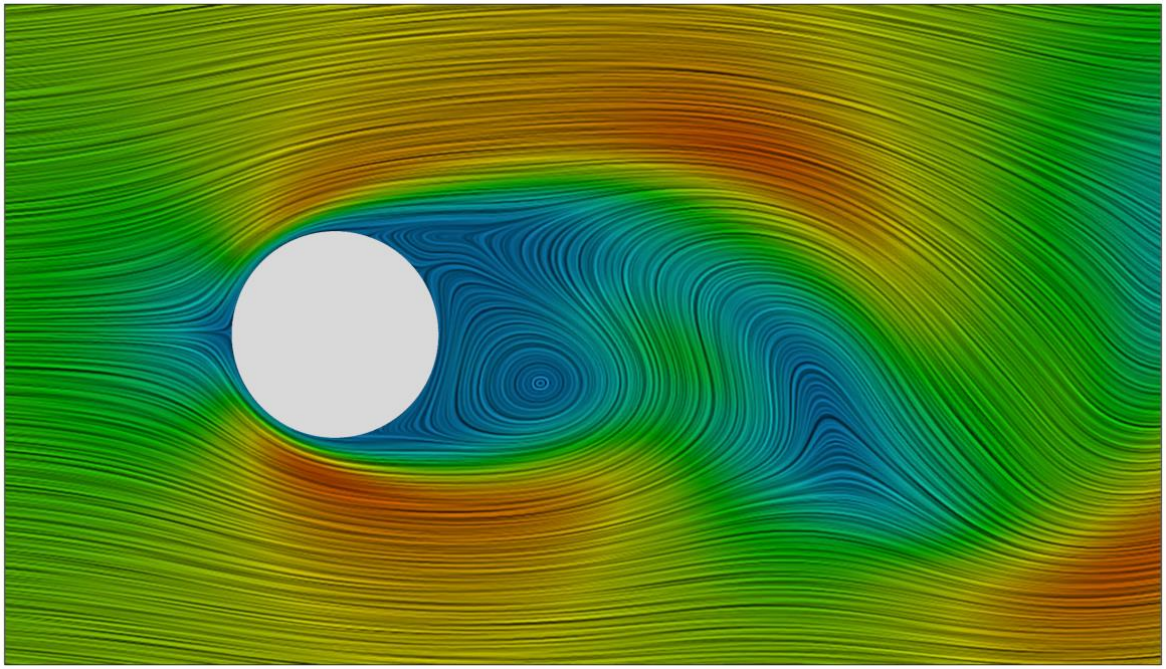


6186 stream lines were required to compute the FastLIC texture. <1 second for 512x256 texture.

- Technical details: Reproducibility
- Using a random texture makes it difficult to reproduce the result from a previous run of the program
- ➔ Initialize the random number generator using a seed.
  - The seed could be a parameter given by the user.
  - srand() is the corresponding C function.
  - Then, the same random texture is generated with each run of the program. Unless you change other parameters such as the size of the texture.

- **LIC – Line Integral Convolution**
- Improving LIC in the following directions:
- ➔ combination with color coding
- ➔ special applications (motion blur…)
- ➔ adding flow orientation
- ➔ LIC on surfaces
- ➔ LIC for 3D flows
- ➔ LIC for unsteady flows

- **LIC – Line Integral Convolution**
- Combination with color coding
- Usually, LIC does not use the color channel
  - ➔ Use color to encode scalar quantities



Velocity magnitude encoded using color

2D flow behind a cylinder