CSC 544
Data Visualization

Joshua Levine
josh@arizona.edu

Lecture 23
Flow Visualization

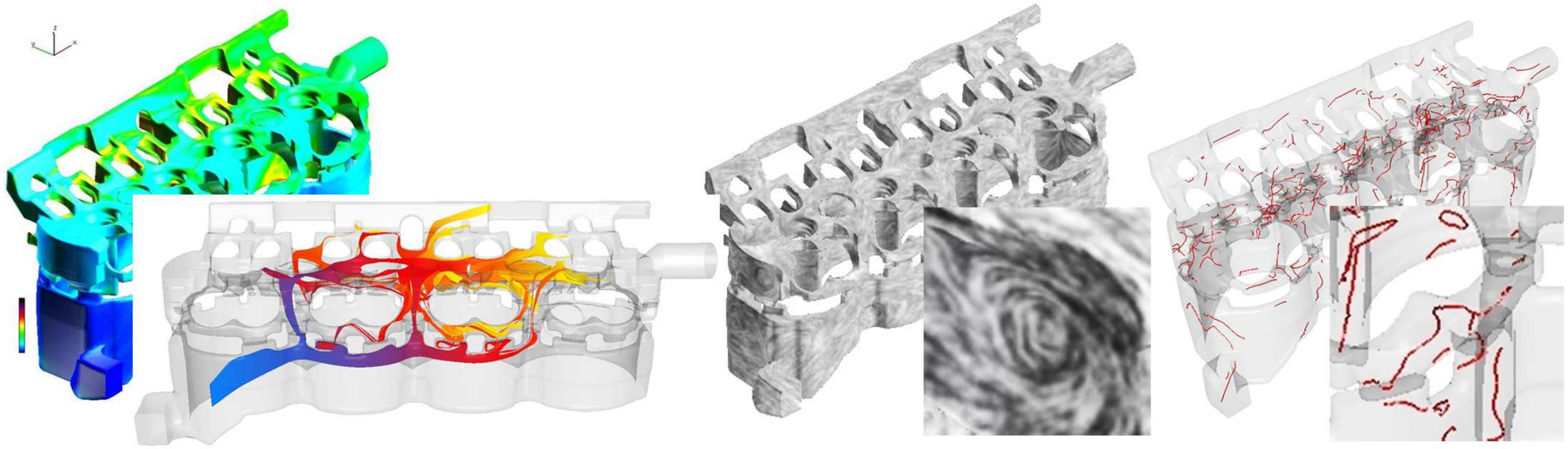
Apr. 10, 2023

Today's Agenda

- Reminders:
  - A05 due! A06 posted (due Apr. 24)
  - P03/P04 questions? (due Apr. 26/May 3)
- Goals for today: Discuss concepts related to geometric and texture-based flow visualization

Classification of Visualization Techniques

- **Direct:** overview of vector field, minimal computation, e.g. glyphs, color mapping
- **Texture-based:** covers domain with a convolved texture, e.g., Spot Noise, LIC, ISA, IBFV(S)
- **Geometric:** a discrete object(s) whose geometry reflects flow characteristics, e.g. streamlines
- **Feature-based:** both automatic and interactive feature-based techniques, e.g. flow topology

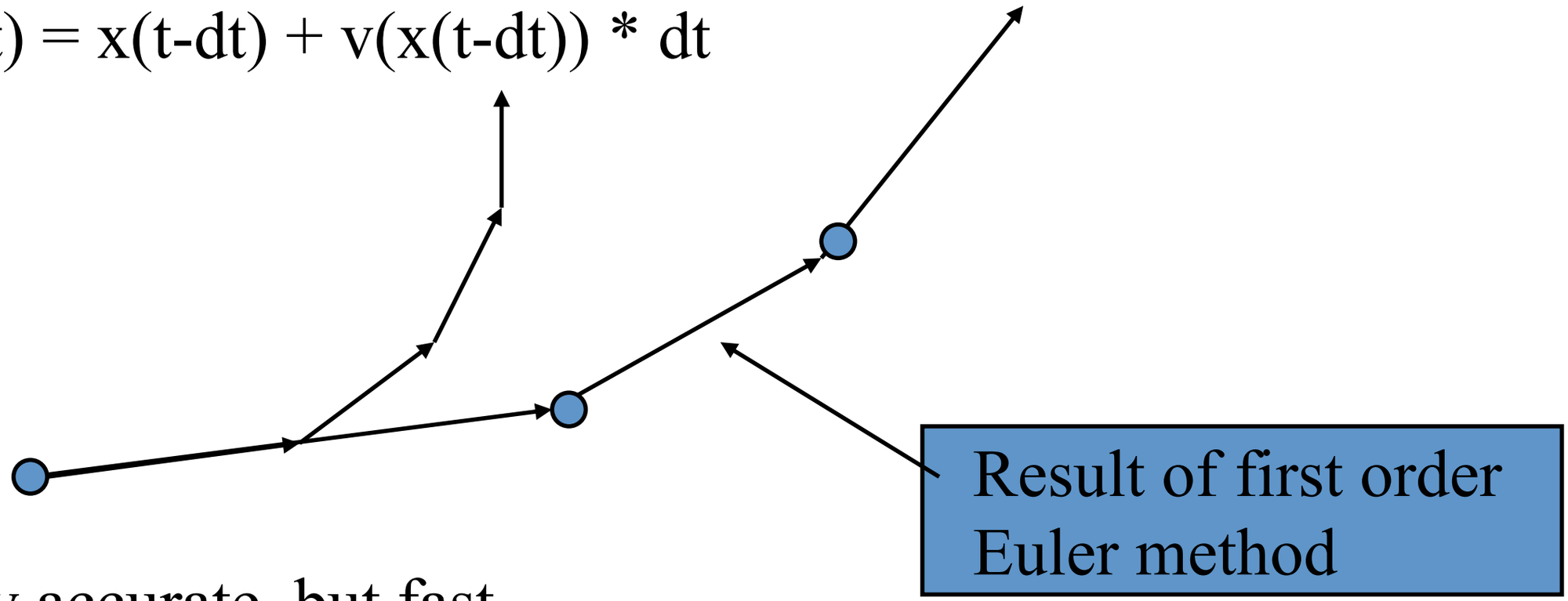


Integration Techniques, Part 2

Numerical Integration

First Order Euler method:

$$x(t) = x(t\text{-}dt) + v(x(t\text{-}dt)) * dt$$



- Not very accurate, but fast
- Other higher order methods are avilable: Runge-Kutta second and fourth order integration methods (more popular due to their accuracy)

Numerical Integration (3)

Standard Method: Runge-Kutta fourth order

$$x(t) = x(t\text{-}dt) + 1/6\ (k_1 + 2k_2 + 2k_3 + k_4)$$

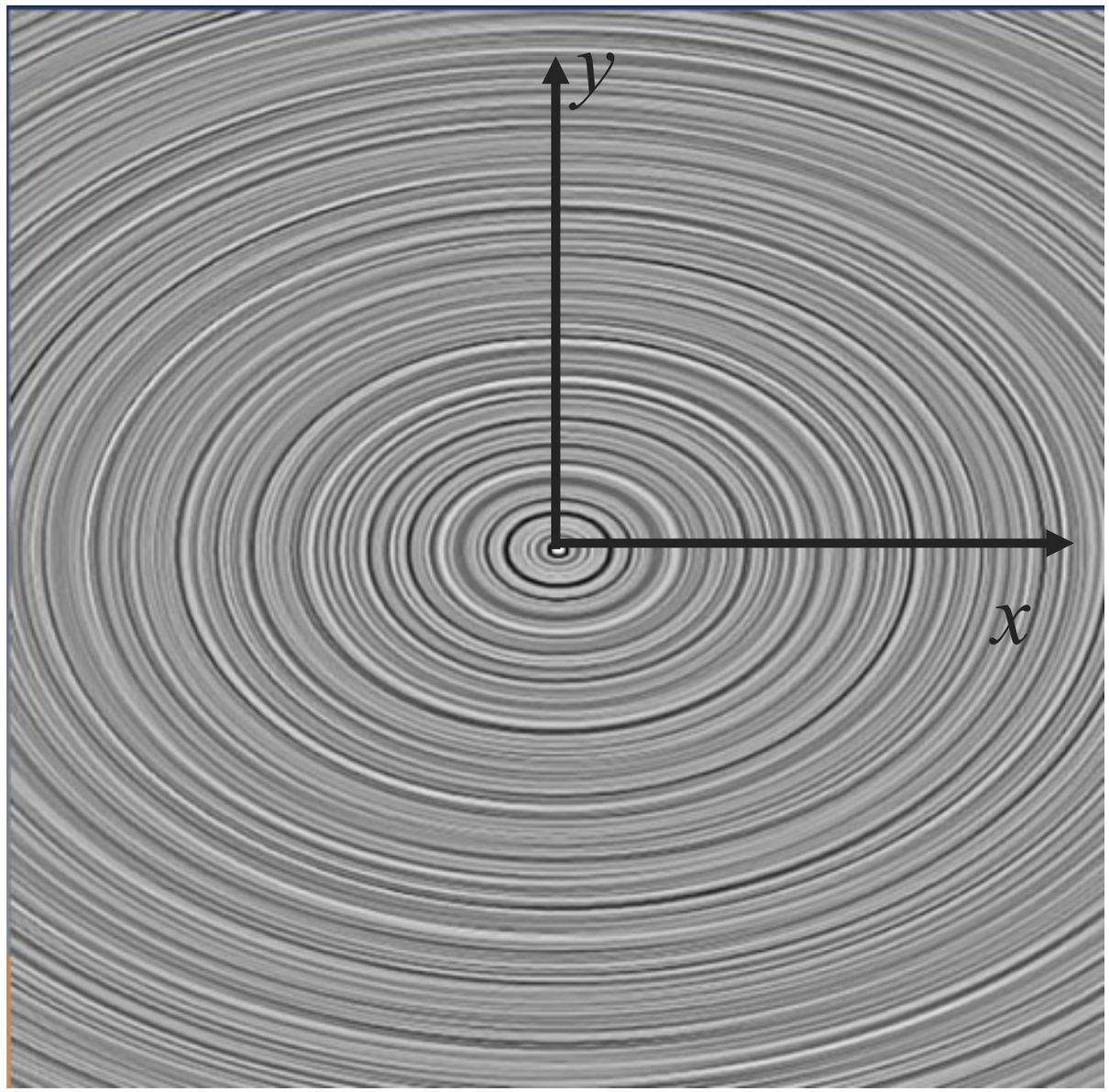
$k_1 = dt * v(t\text{-}dt)$; $k_2 = dt * v(x(t\text{-}dt) + k1/2)$

$k_3 = dt * v(x(t\text{-}dt) + k2/2)$; $k_4 = dt * v(x(t\text{-}dt) + k3)$

- **Numerical integration of stream lines:**

- approximate streamline by polygon $\mathbf{x}_i$

- **Testing example:**
  - $\mathbf{v}(x,y) = (-y, x/2)^T$
  - exact solution: ellipses
  - starting integration from (0,-1)



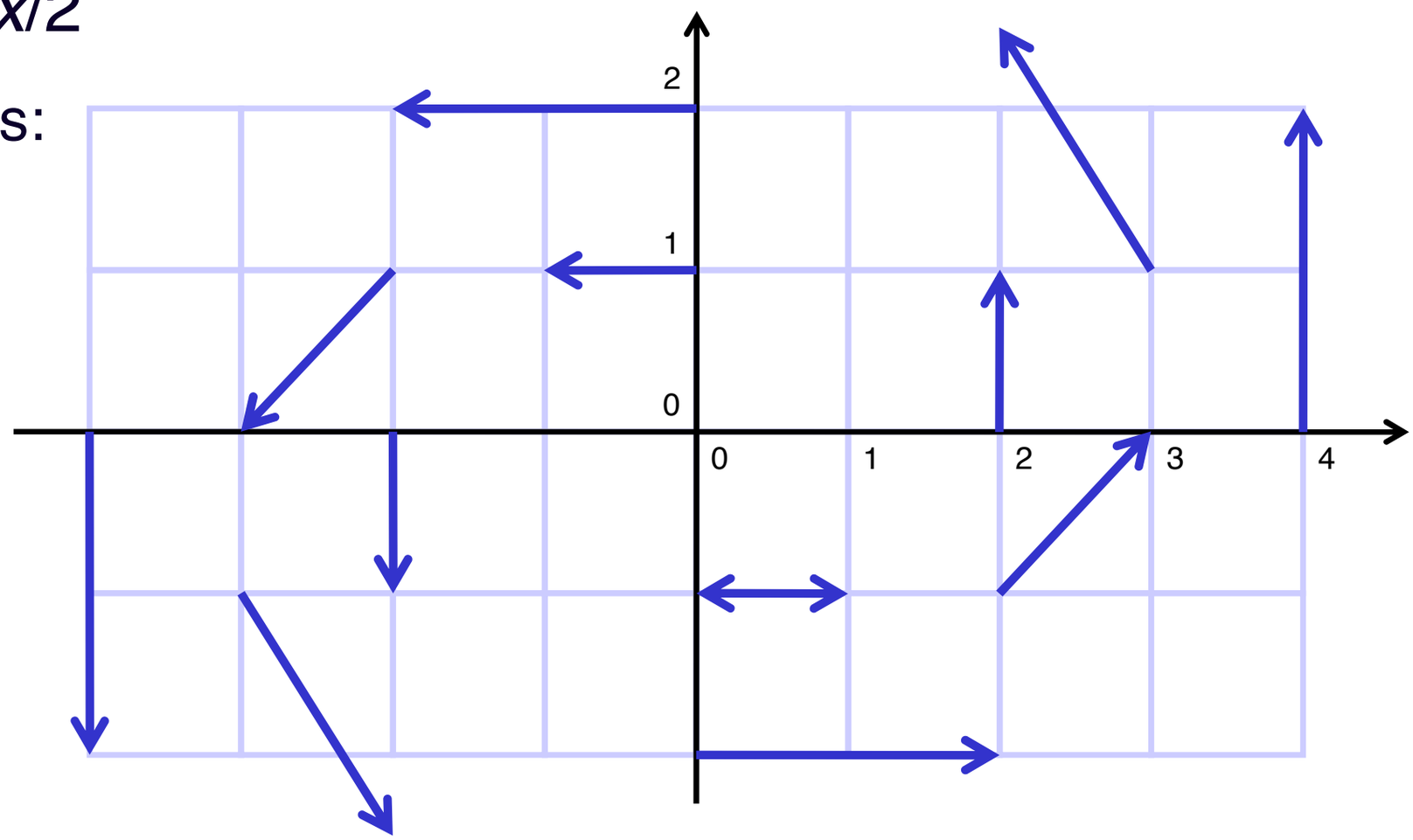
Euler Integration – Example

2D model data:

$v_x = \mathrm{d}x/\mathrm{d}t = -y$

$v_y = \mathrm{d}y/\mathrm{d}t = x/2$

Sample arrows:



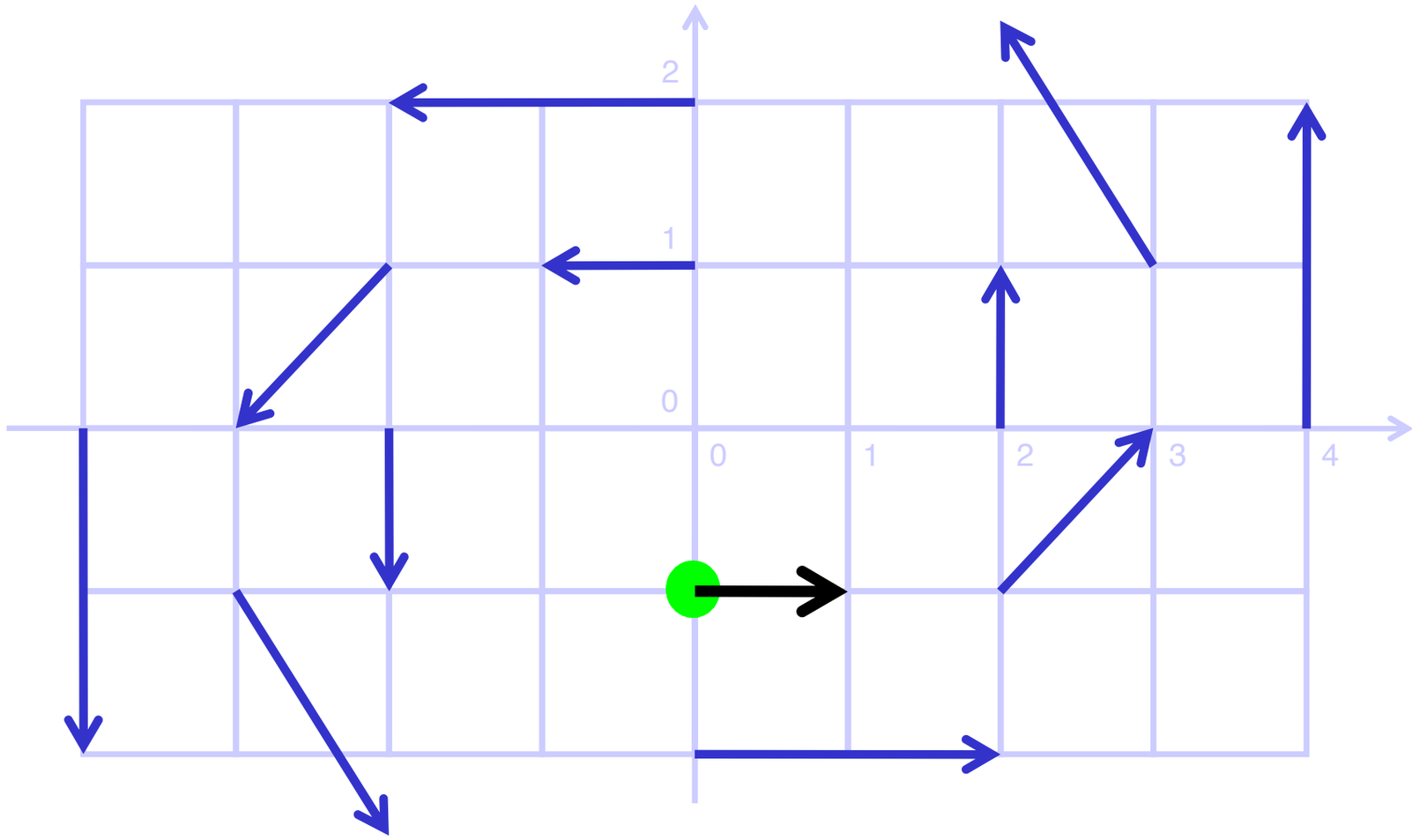
True solution: ellipses.

Euler Integration – Example

- Seed point $\mathbf{s}_0 = (0|-1)^T$;
current flow vector $\mathbf{v}(\mathbf{s}_0) = (1|0)^T$;
$dt = ½$

$v_x = dx/dt = -y$

$v_y = dy/dt = x/2$

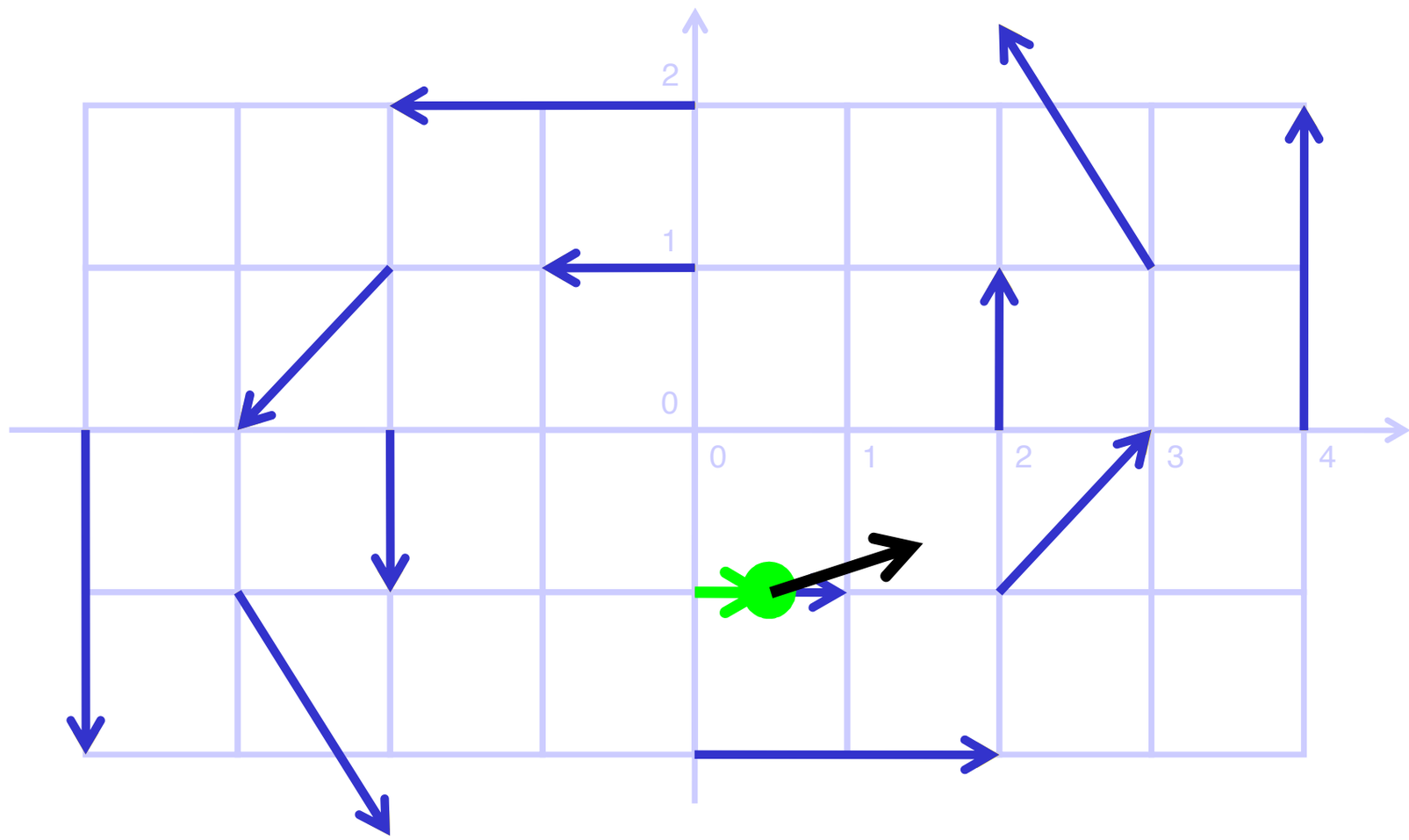


Euler Integration – Example

- New point $\mathbf{s}_1 = \mathbf{s}_0 + \mathbf{v}(\mathbf{s}_0) \cdot \mathrm{d}t = (1/2 \mid -1)^T$; current flow vector $\mathbf{v}(\mathbf{s}_1) = (1 \mid 1/4)^T$;

$v_x = \mathrm{d}x/\mathrm{d}t = -y$

$v_y = dy/dt = x/2$

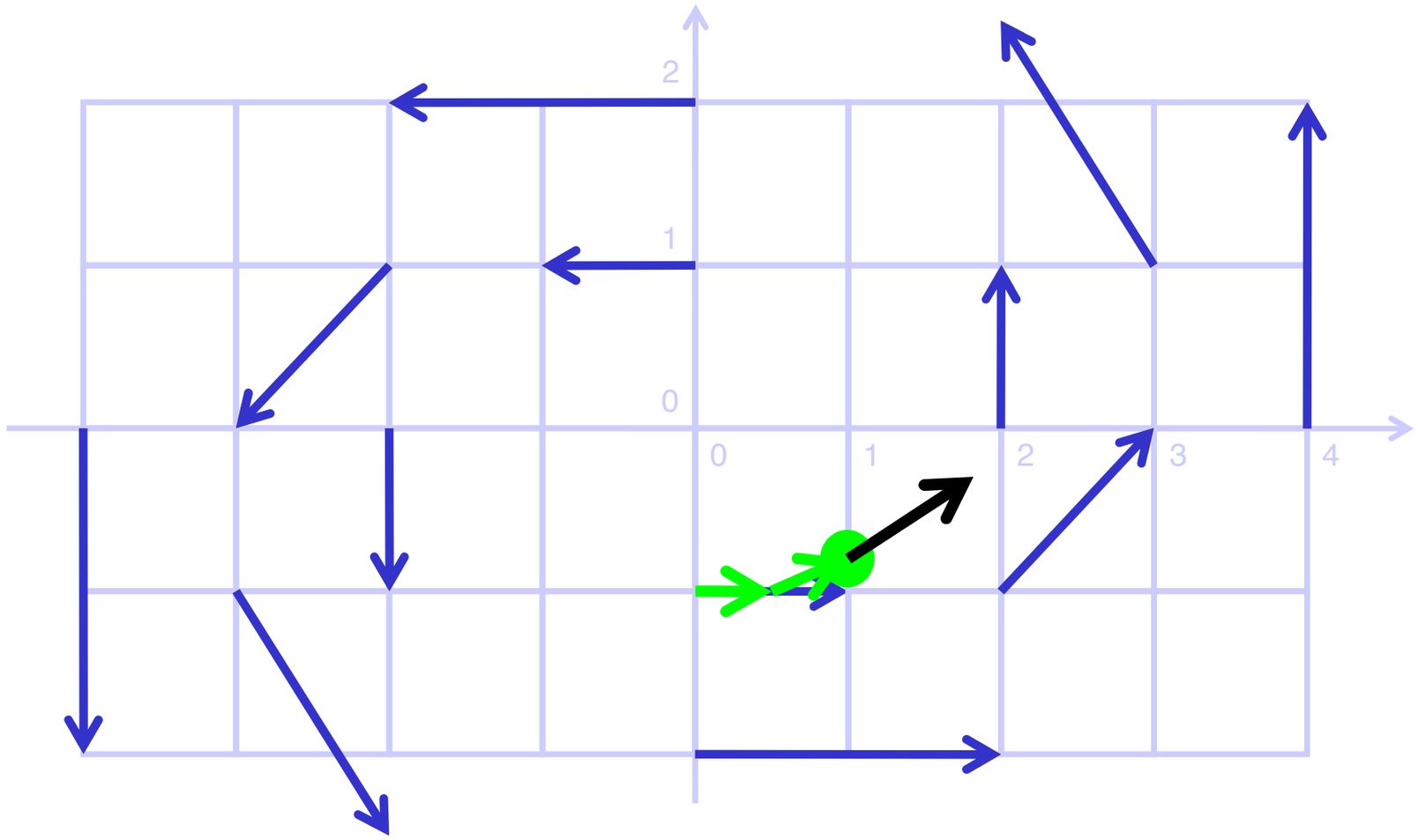


Euler Integration – Example

- New point $\mathbf{s}_2 = \mathbf{s}_1 + \mathbf{v}(\mathbf{s}_1) \cdot \mathrm{d}t = (1 | -7/8)^T$; current flow vector $\mathbf{v}(\mathbf{s}_2) = (7/8 | 1/2)^T$;

$v_x = \mathrm{d}x/\mathrm{d}t = -y$

$v_y = dy/dt = x/2$



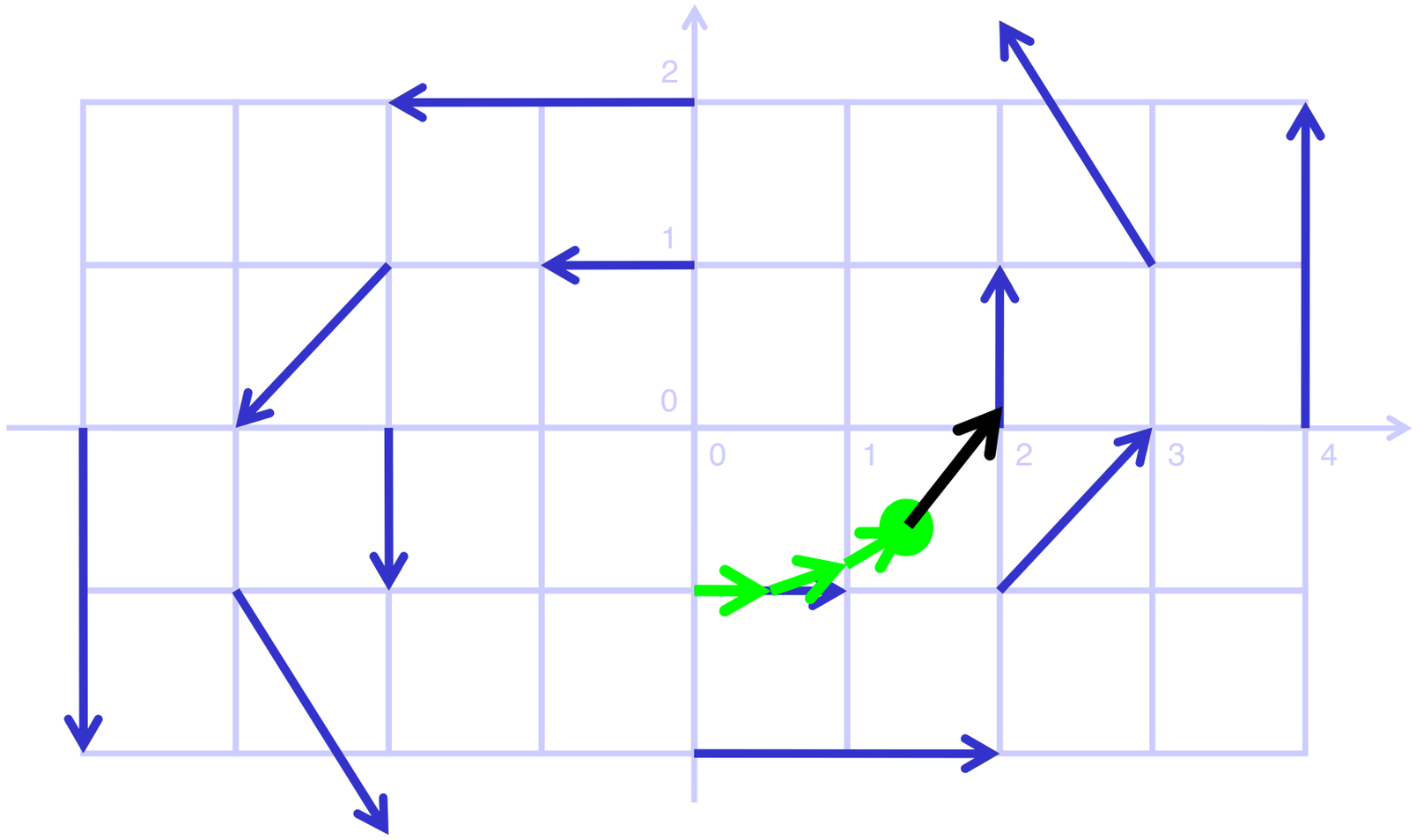
Euler Integration – Example

- $\mathbf{s}_3 = (23/16 | -5/8)^T \approx (1.44 | -0.63)^T$;

$\mathbf{v}(\mathbf{s}_3) = (5/8 | 23/32)^T \approx (0.63 | 0.72)^T$;

$v_x = \mathrm{d}x/\mathrm{d}t = -y$

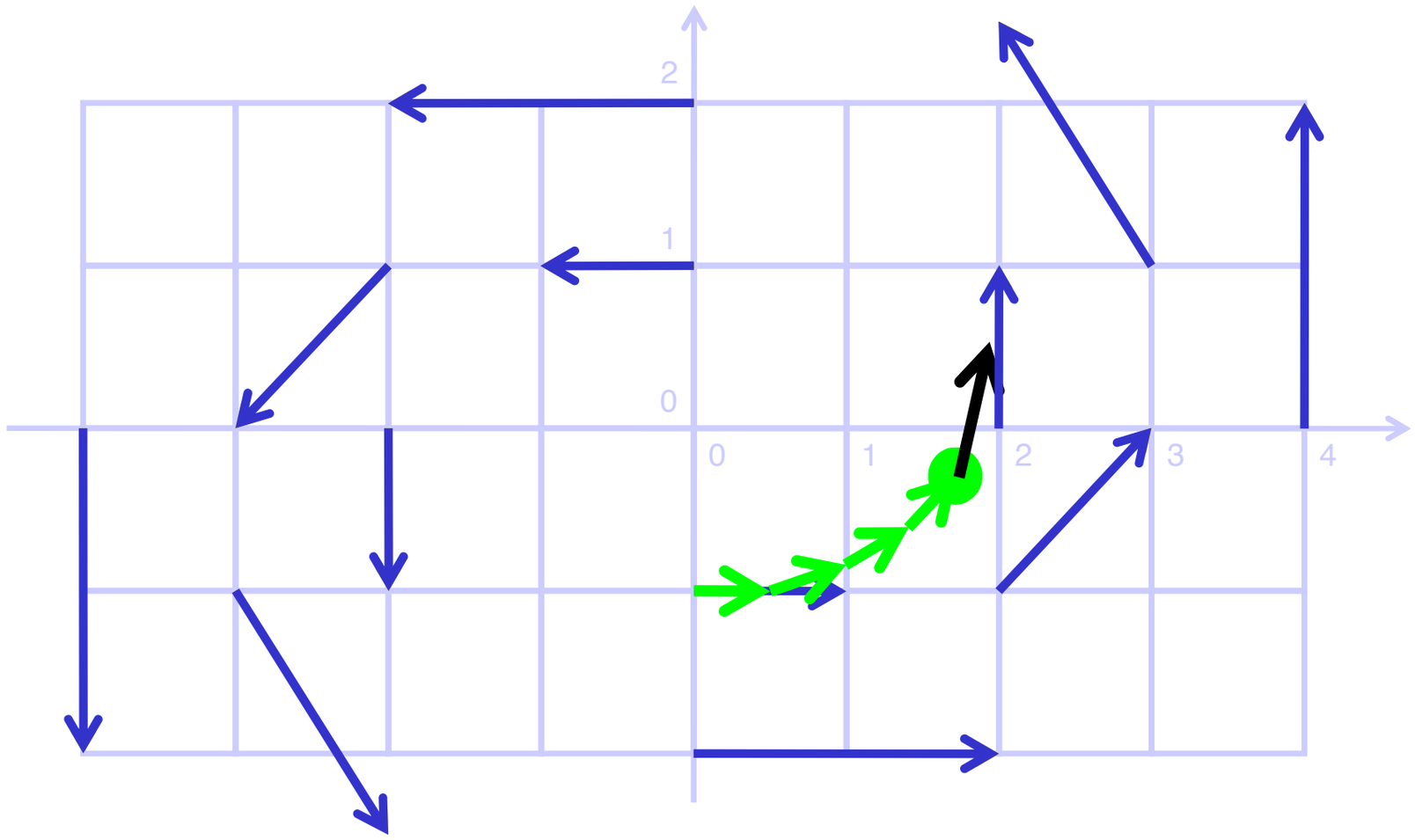
$v_y = dy/dt = x/2$



Euler Integration – Example

- $\mathbf{s}_4 = (7/4 | -17/64)^T \approx (1.75 | -0.27)^T$;

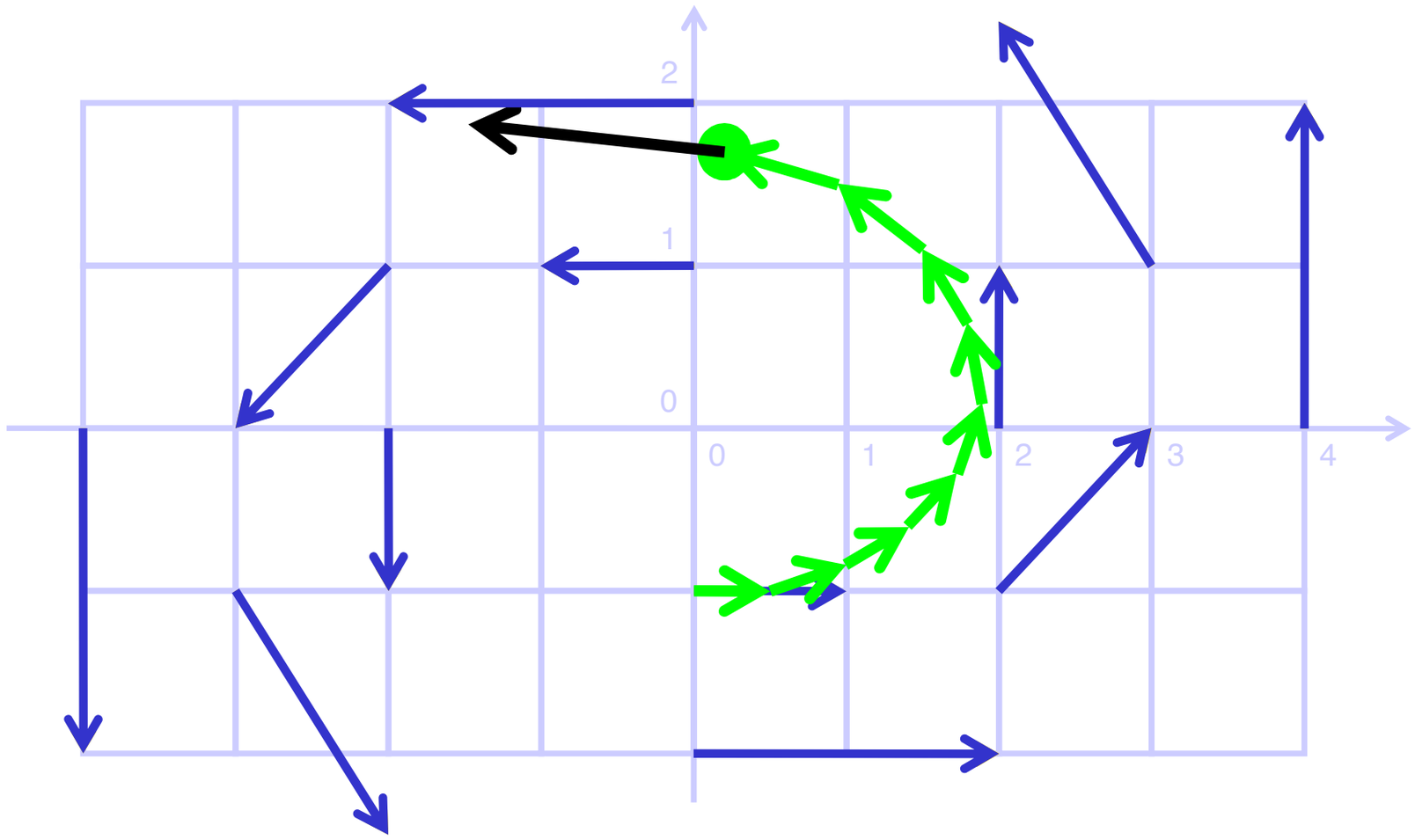
$\mathbf{v}(\mathbf{s}_4) = (17/64 | 7/8)^T \approx (0.27 | 0.88)^T$;



Euler Integration – Example

- $\mathbf{s}_9 \approx (0.20 | 1.69)^T$;

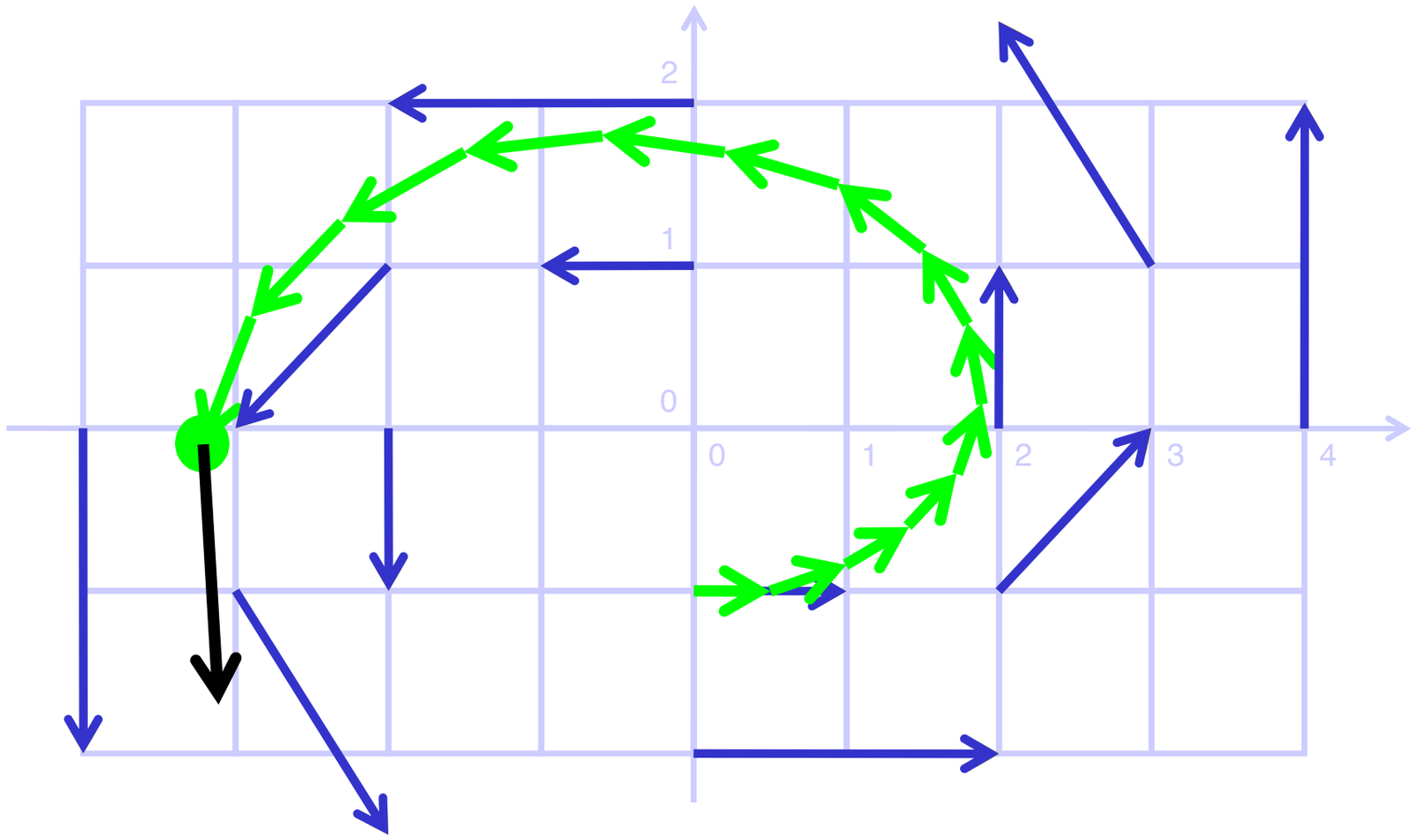
$\mathbf{v}(\mathbf{s}_9) \approx (-1.69 | 0.10)^T$;



Euler Integration – Example

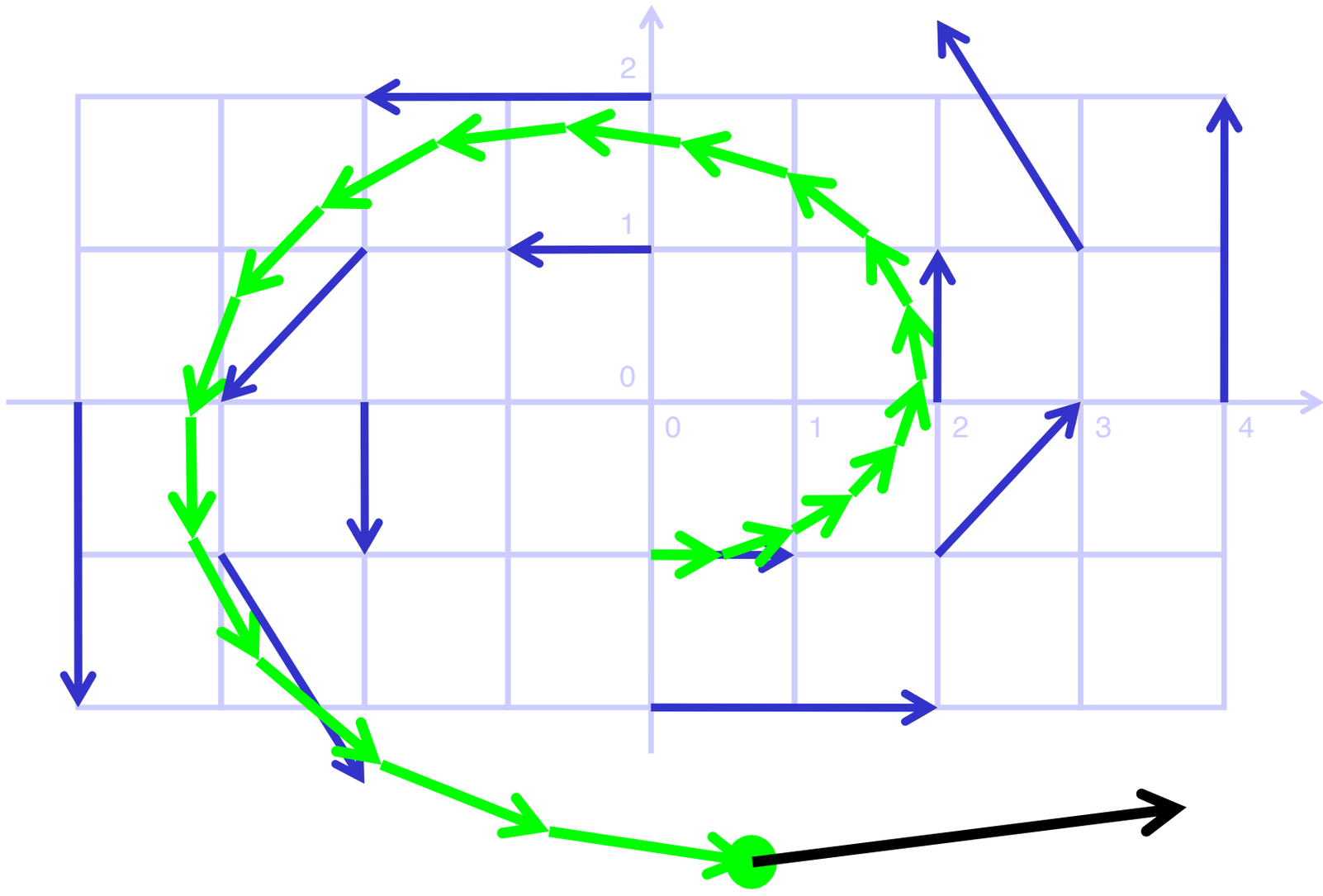
- $\mathbf{s}_{14} \approx (-3.22 | -0.10)^T$;

$\mathbf{v}(\mathbf{s}_{14}) \approx (0.10 | -1.61)^T$;



Euler Integration – Example

- $\mathbf{s}_{19} \approx (0.75 | -3.02)^T$; $\mathbf{v}(\mathbf{s}_{19}) \approx (3.02 | 0.37)^T$; clearly: large integration error, dt too large, 19 steps

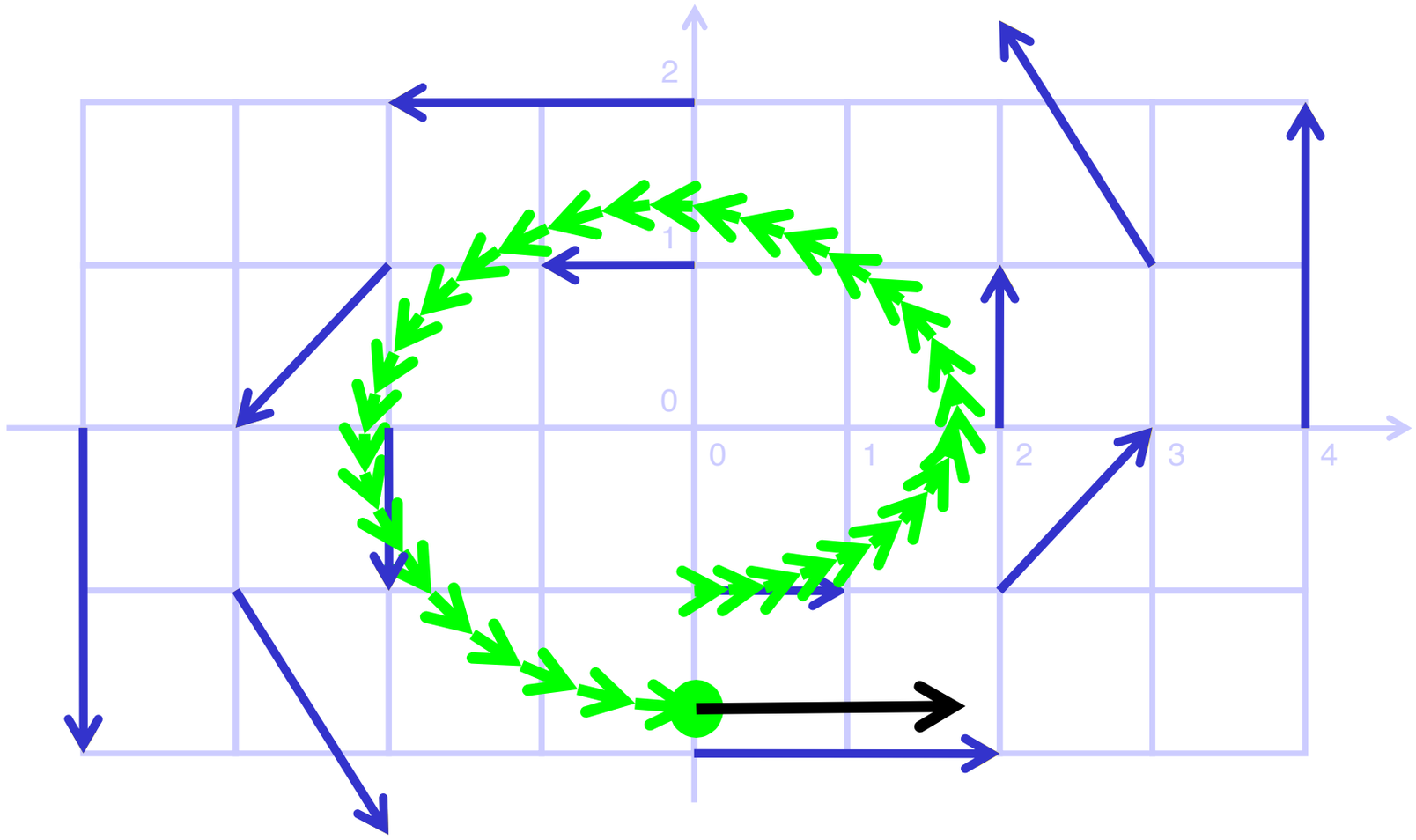


Euler Integration – Example

- d*t* smaller (1/4): more steps, more exact.

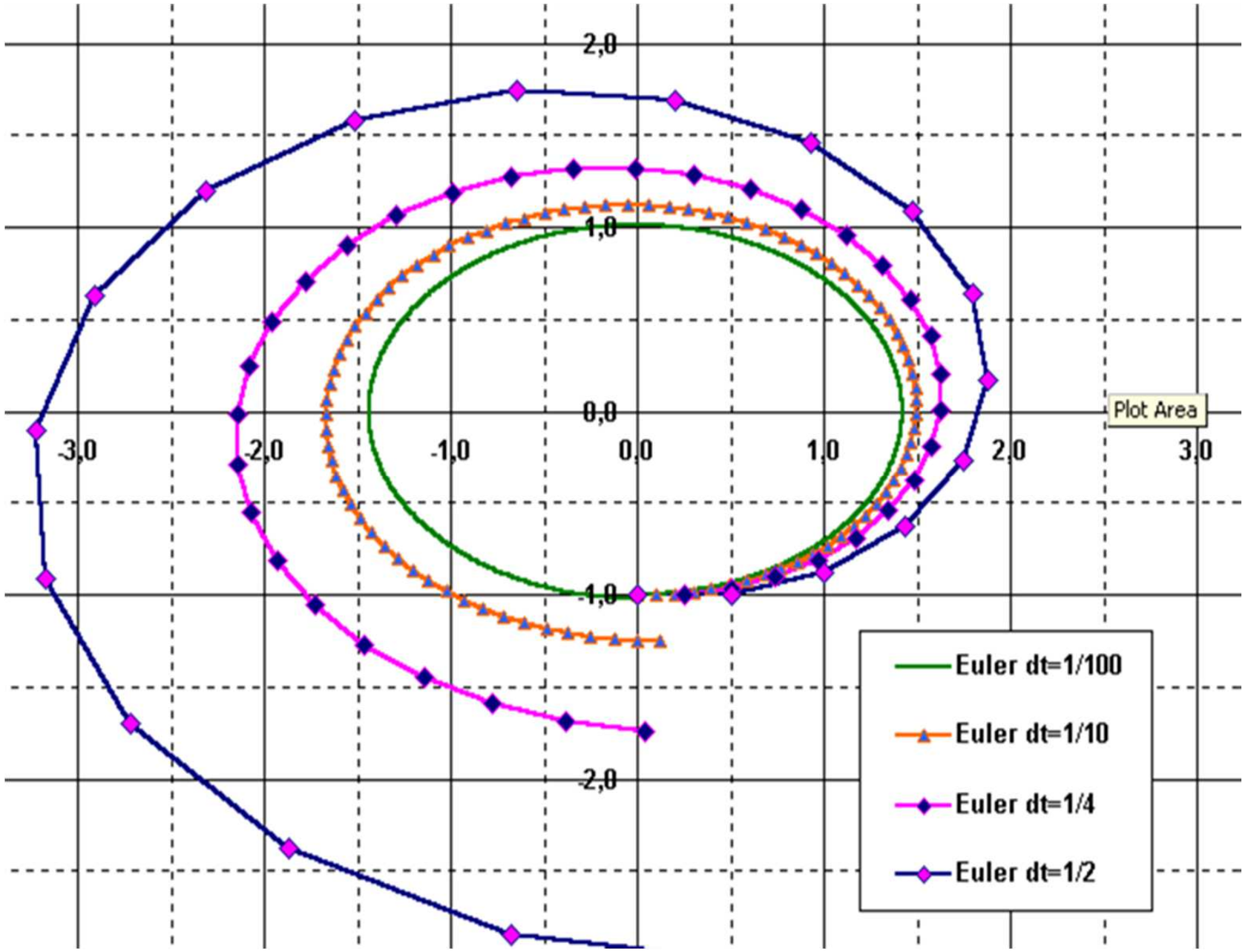
$\mathbf{s}_{36} \approx (0.04 | -1.74)^T$; $\mathbf{v}(\mathbf{s}_{36}) \approx (1.74 | 0.02)^T$;

- 36 steps



Comparison Euler, Step Sizes

Euler
quality is
proportional
to d*t*

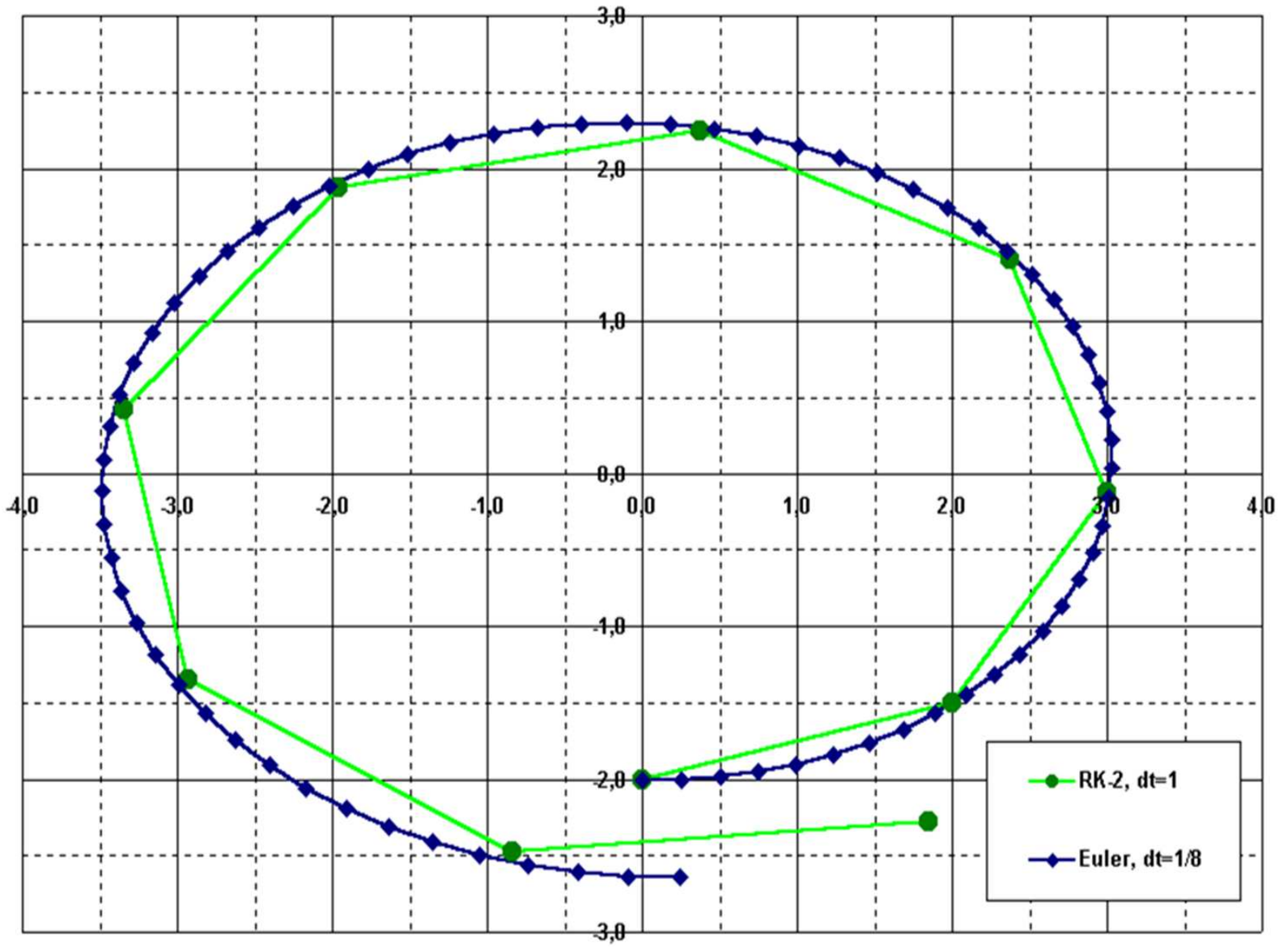


Euler Example – Error Table

d*t*	#steps	error
1/2	19	~200%
1/4	36	~75%
1/10	89	~25%
1/100	889	~2%✓
1/1000	8889	~0.2%

RK-2 – A Quick Round

RK-2: even with dt = 1 (9 steps)
better
than Euler
with dt = 1/8
(72 steps)



RK-4 vs. Euler, RK-2

Even better: fourth order RK:

- four vectors **a**, **b**, **c**, **d**
- one step is a convex combination:
  $\mathbf{s}_{i+1} = \mathbf{s}_i + (\mathbf{a} + 2\cdot\mathbf{b} + 2\cdot\mathbf{c} + \mathbf{d})/6$
- vectors:

  a = d$t\cdot\mathbf{v}(\mathbf{s}_i)$ … original vector

  b = d$t\cdot\mathbf{v}(\mathbf{s}_i+\mathbf{a}/2)$ … RK-2 vector

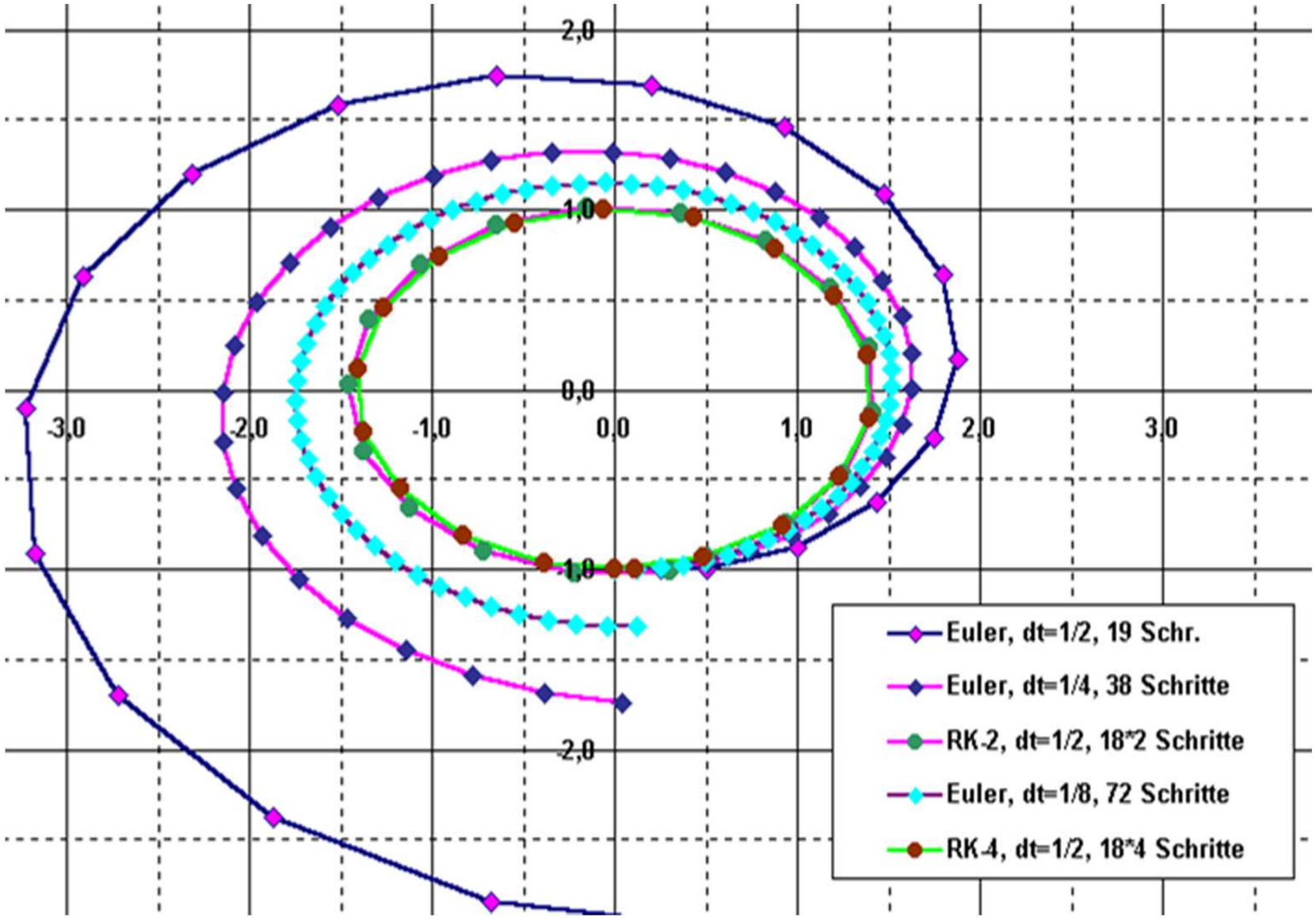
  c = d$t\cdot\mathbf{v}(\mathbf{s}_i+\mathbf{b}/2)$ … use RK-2 …

  d = d$t\cdot\mathbf{v}(\mathbf{s}_i+\mathbf{c})$ … and again

Euler vs. Runge-Kutta

RK-4: pays off only with complex flows

Here approx. like RK-2



Integration, Conclusions

Summary:

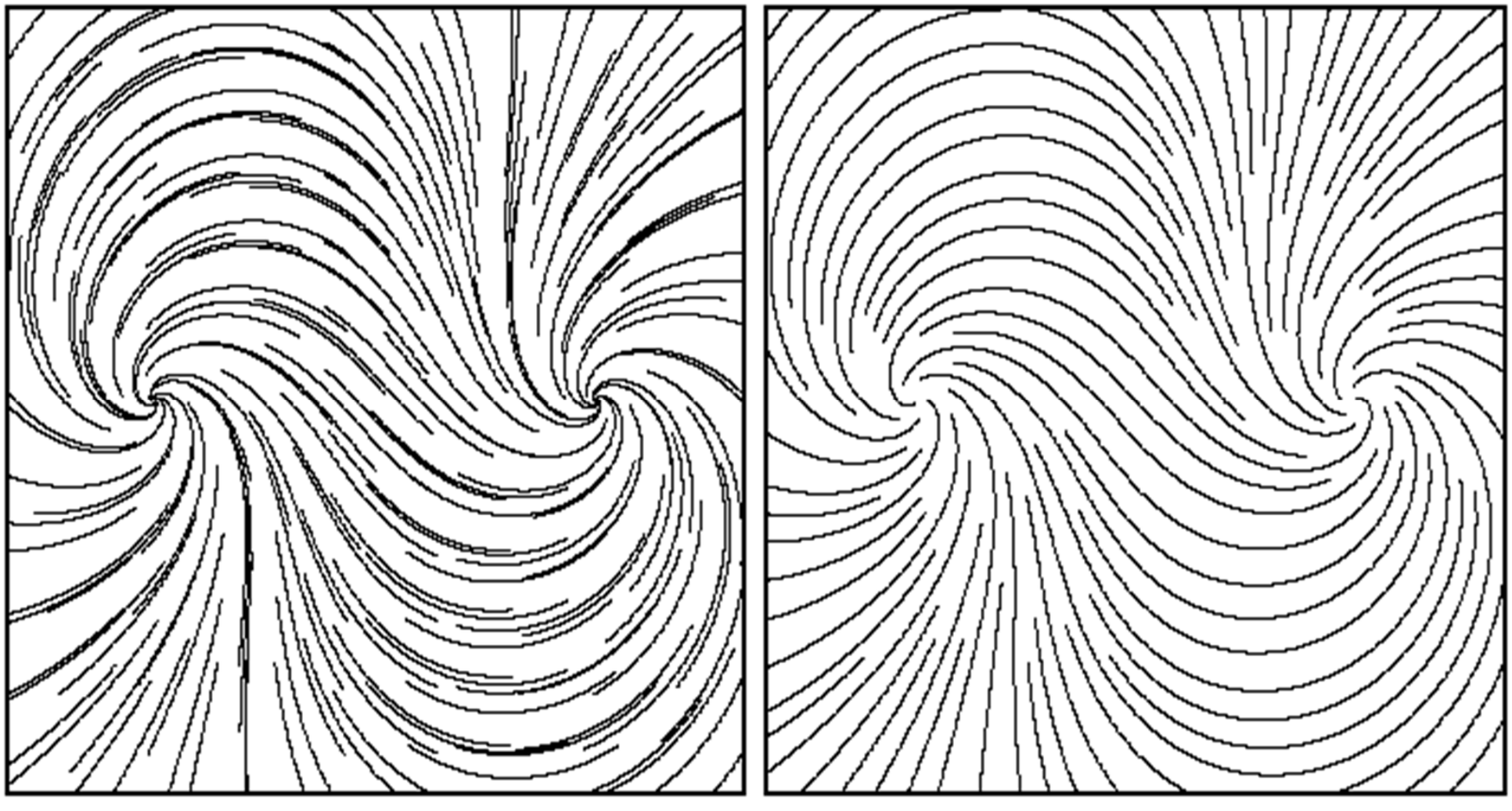
- analytic determination of streamlines usually not possible
- hence: numerical integration
- various methods available
  (Euler, Runge-Kutta, etc.)
  - Euler: simple, imprecise, esp. with small d*t*
  - RK: more accurate in higher orders
  - furthermore: adaptive methods, implicit methods, etc.

Streamline Placement

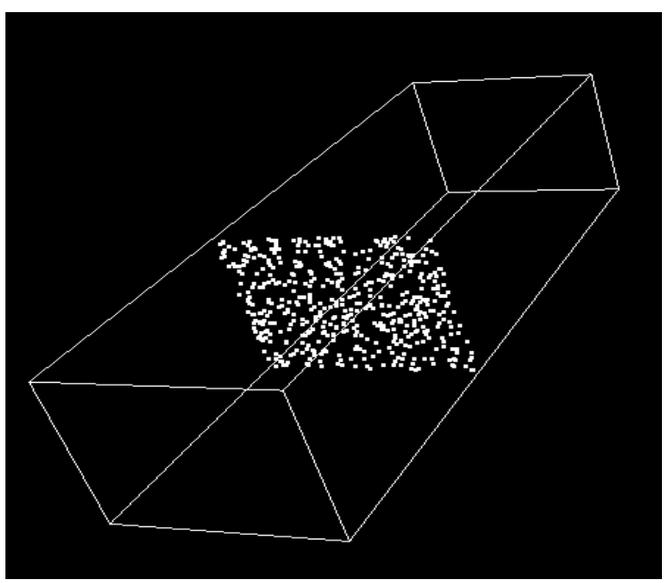
Problem: Choice of Seed Points

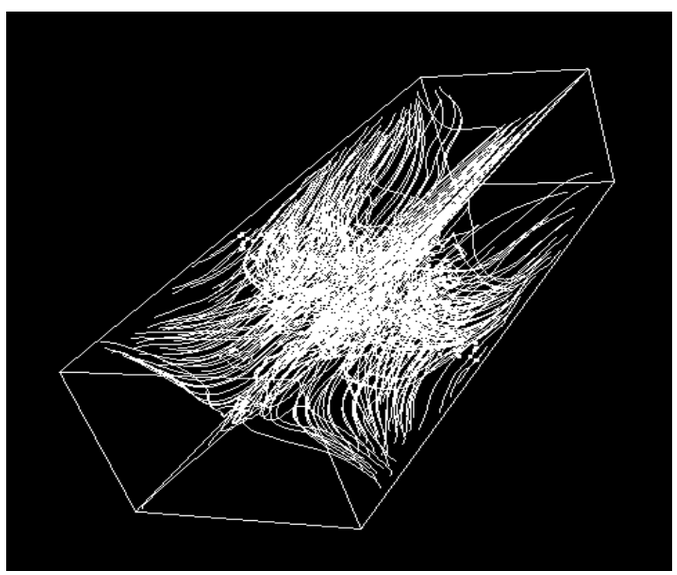
Streamline placement:

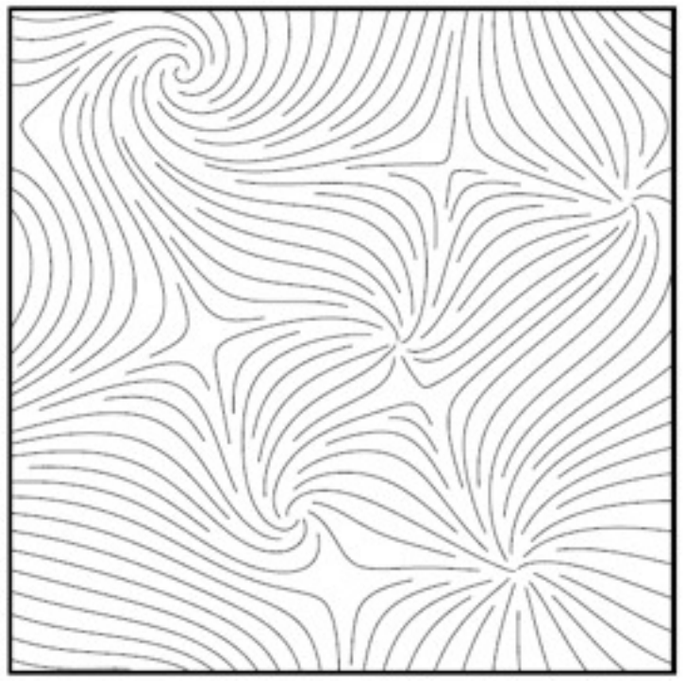
- If regular grid used: very irregular result



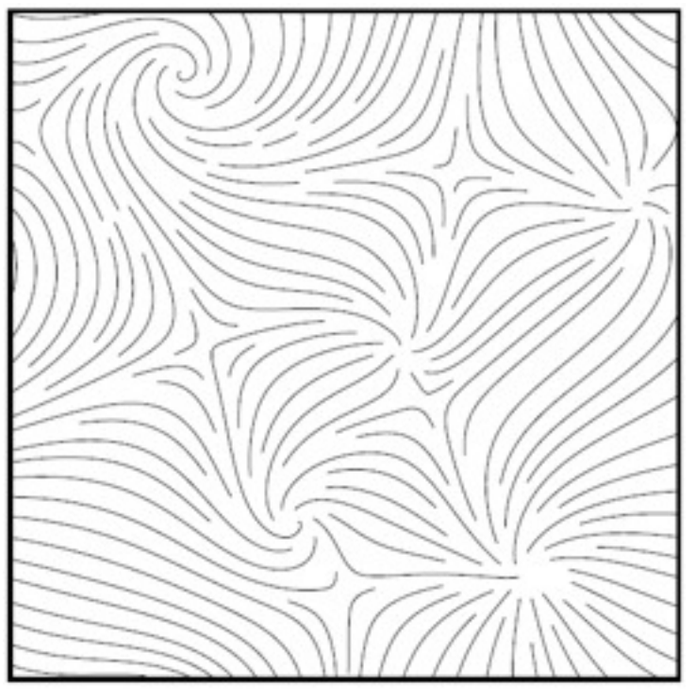
- **Seeding of integral lines:**
- which stream/path/streak/time lines to visualize?
- too few: important details get lost
- too many: overload, visual clutter
- simple approaches:
  - start on regular grid points
  - start randomly
- It has to be the right number at the right places!!!





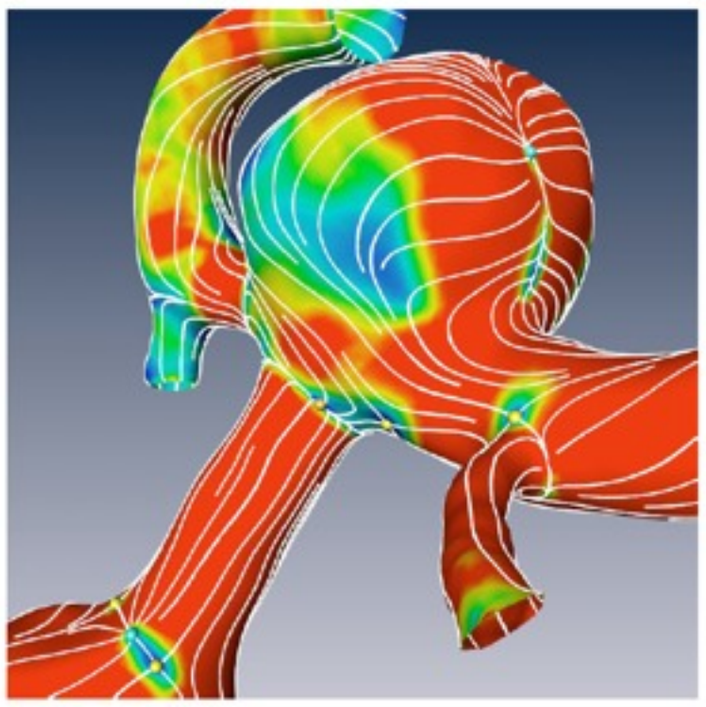


Turk and Banks, 1996



Jobard et al., 1997

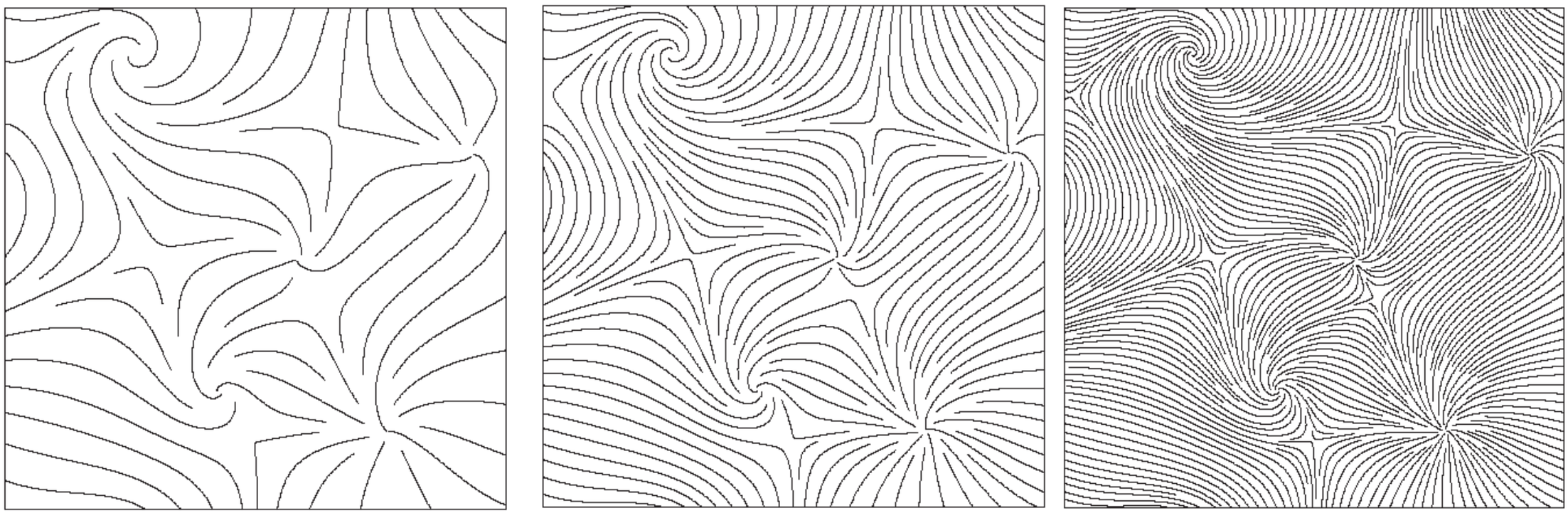
Mebarki et al., 2005



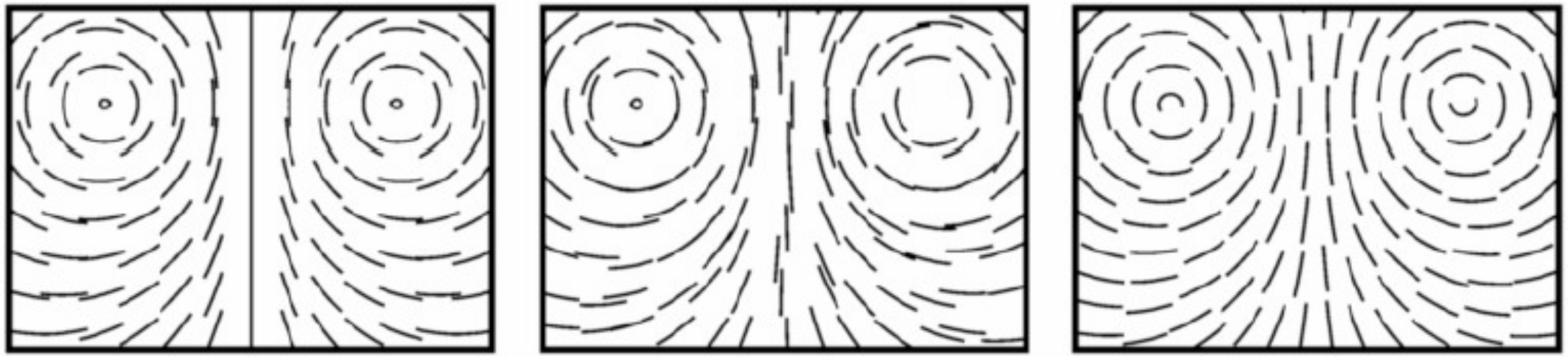
Rosanwo et al., 2009

Streamline seeding

- 2D: evenly spaced stream lines
- Turk/Banks 96:
  - Start with “streamlets” (very short stream lines)
  - Apply a series of energy-decreasing elementary operations: combine, delete, create, lengthen, shorten streamlets
  - Energy: difference between low-pass filtered version of current placements and uniform grey image



Main idea: the distribution of ink on the screen should be even [Turk and Bank 96]



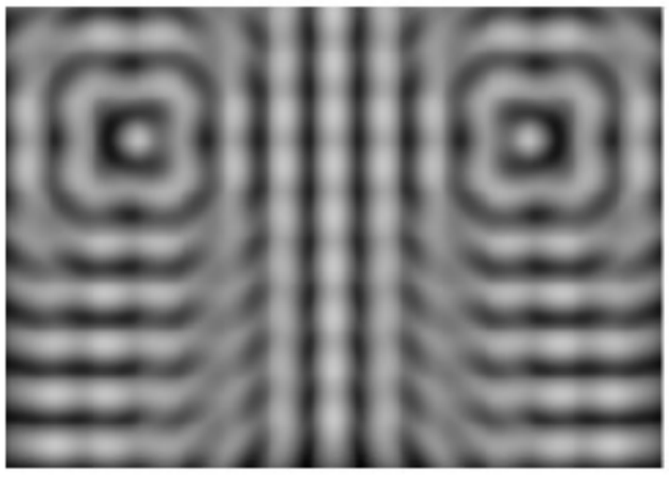


Figure 2: (a) Short streamlines with centers placed on a regular grid (top); (b) filtered version of same (bottom).

Figure 3: (a) Short streamlines with centers placed on a jittered grid (top); (b) filtered version showing bright and dark regions (bottom).

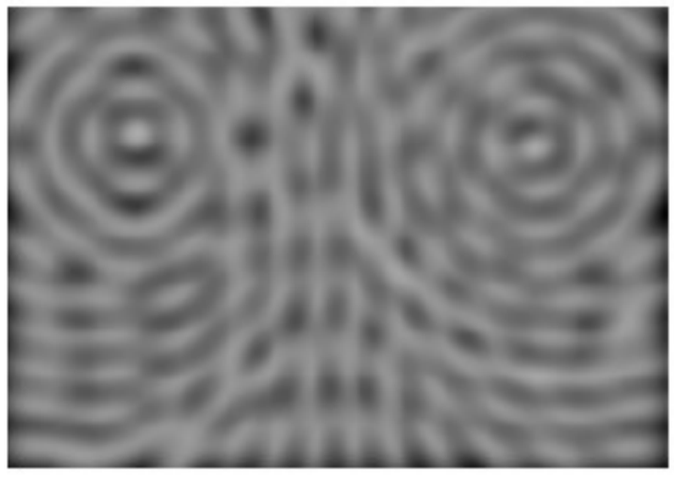
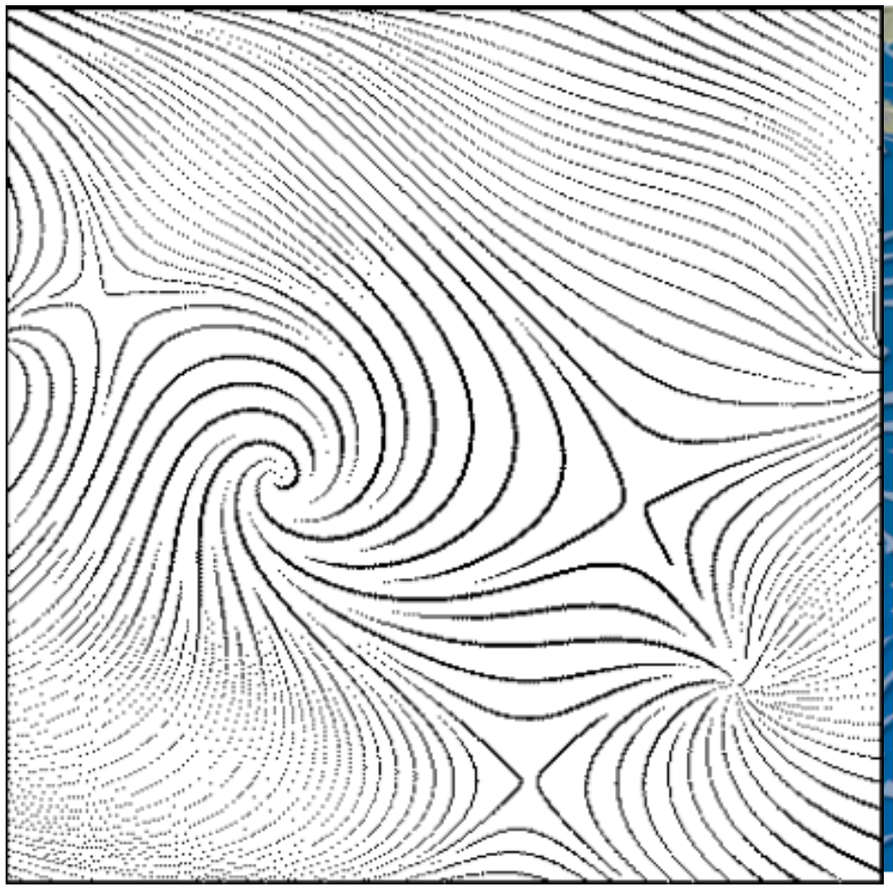
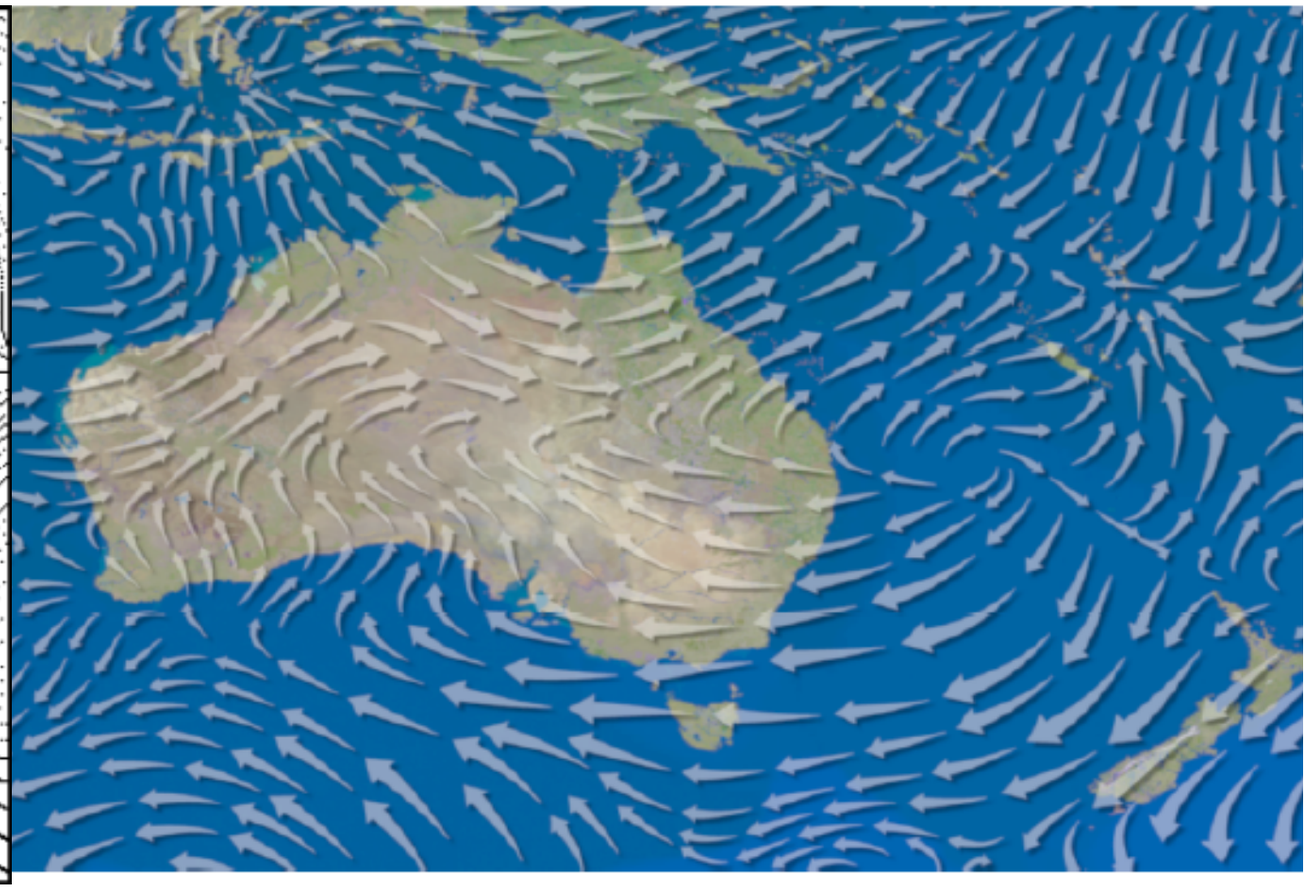


Figure 4: (a) Short streamlines placed by optimization (top); (b) filtered version showing fairly even gray value (bottom).

Results



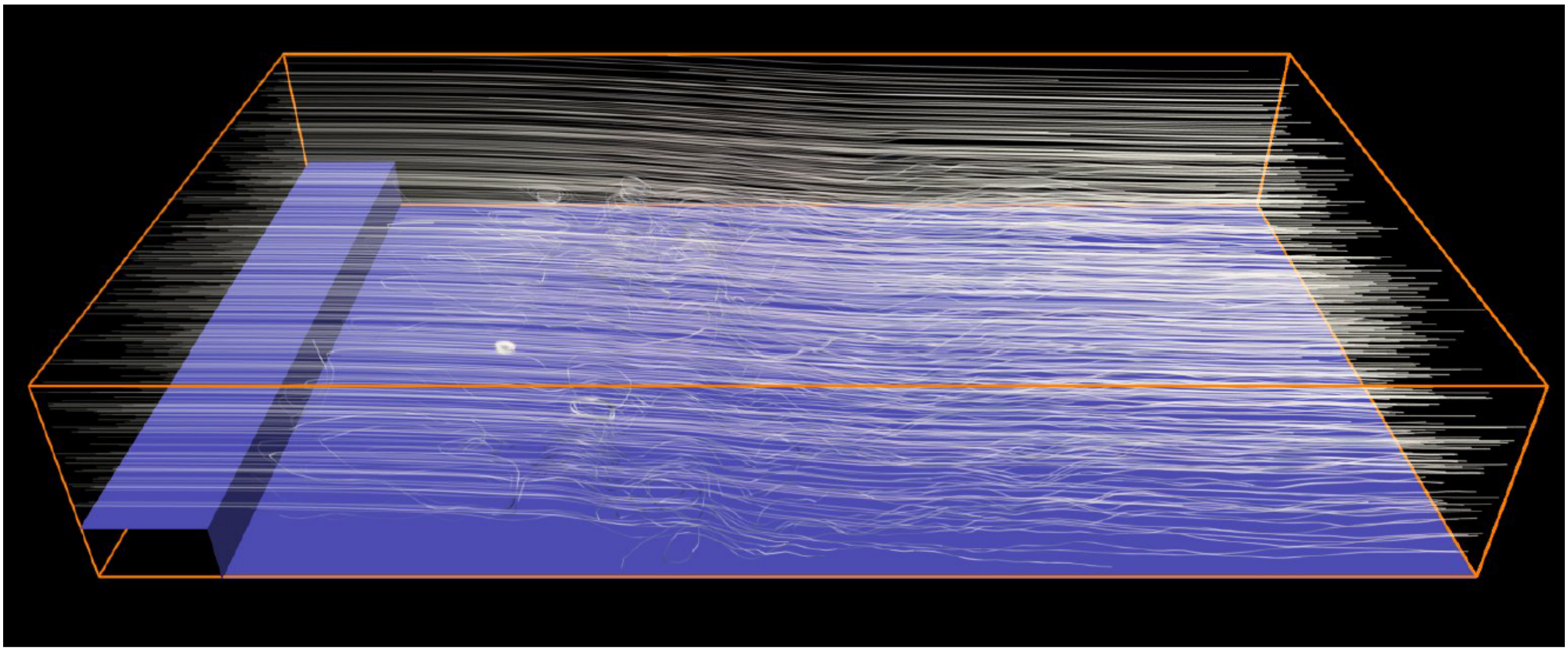
Tapering at streamline ends



Optimized arrow plots

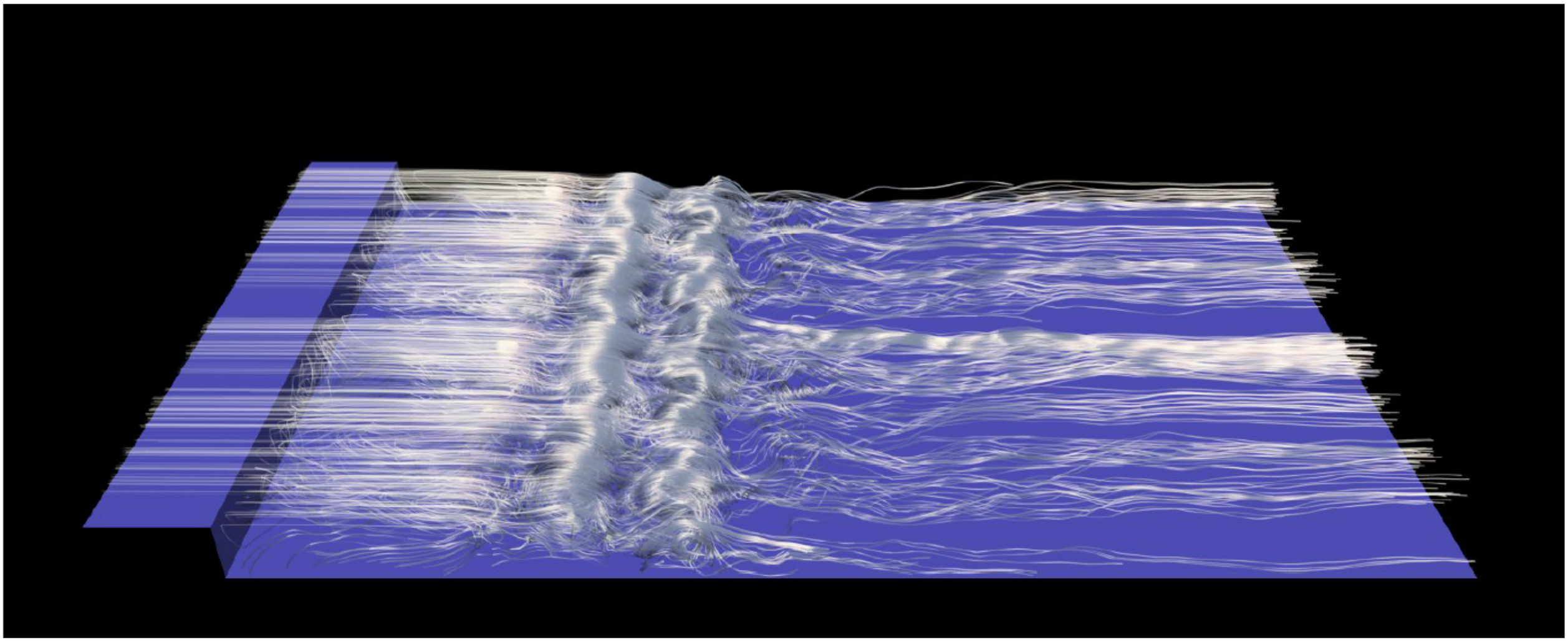
[Turk and Bank ’96]

Streamline Seeding in 3D



Weinkauf 2003

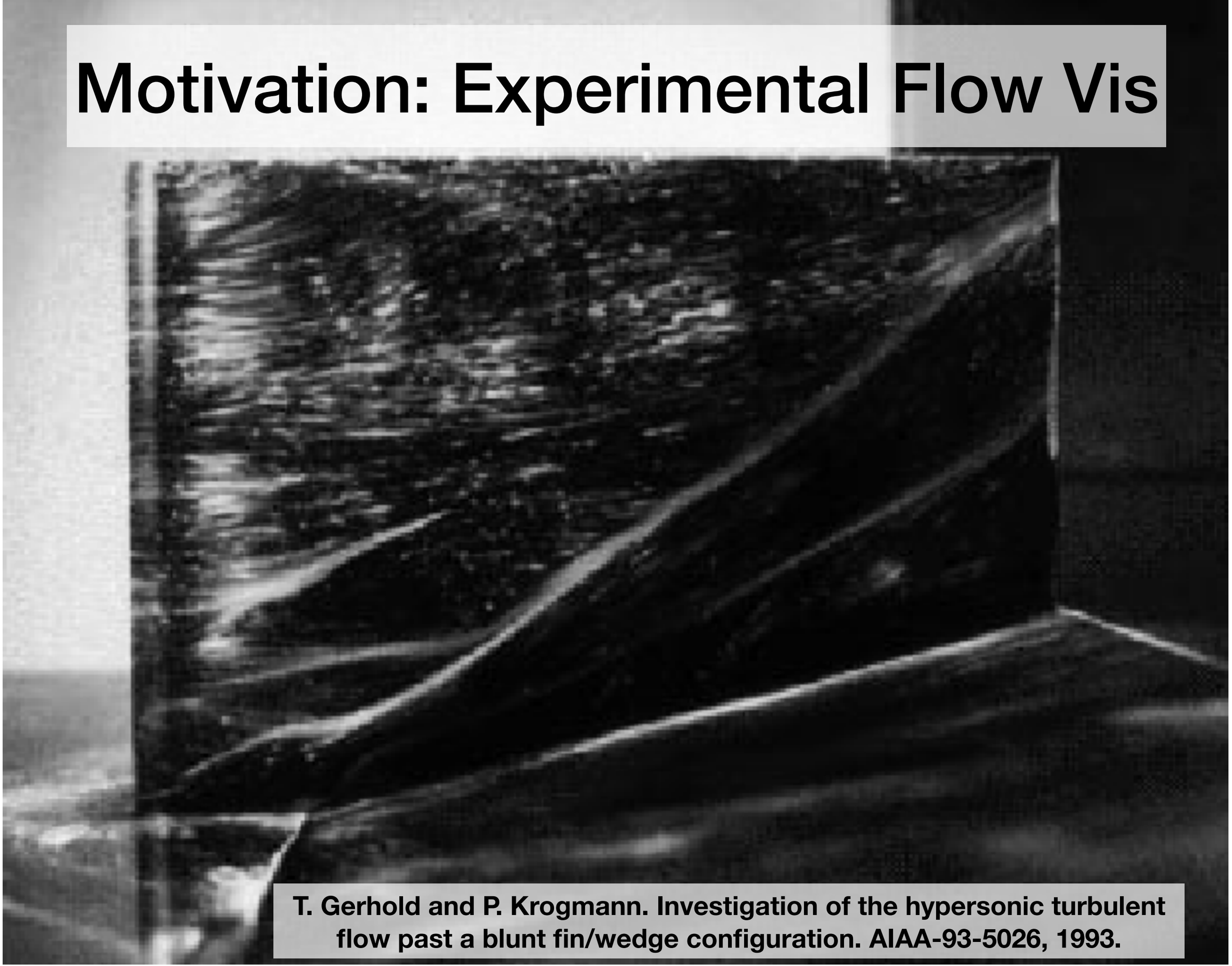
Streamline Seeding in 3D



Weinkauf 2003

Texture-Based Methods

Motivation: Experimental Flow Vis



T. Gerhold and P. Krogmann. Investigation of the hypersonic turbulent flow past a blunt fin/wedge configuration. AIAA-93-5026, 1993.

Spot Noise

- Idea: distribute a set of intensity functions (spots) over domain

- Each spot represents a particle warped over a small time step

- Streak in the direction of local flow from where the particle is seeded

- Repeat for many spots, and then blend together

Spot Noise

- Different textures can be created using different spot shapes, key idea is to align the shape of the spot with the direction of flow

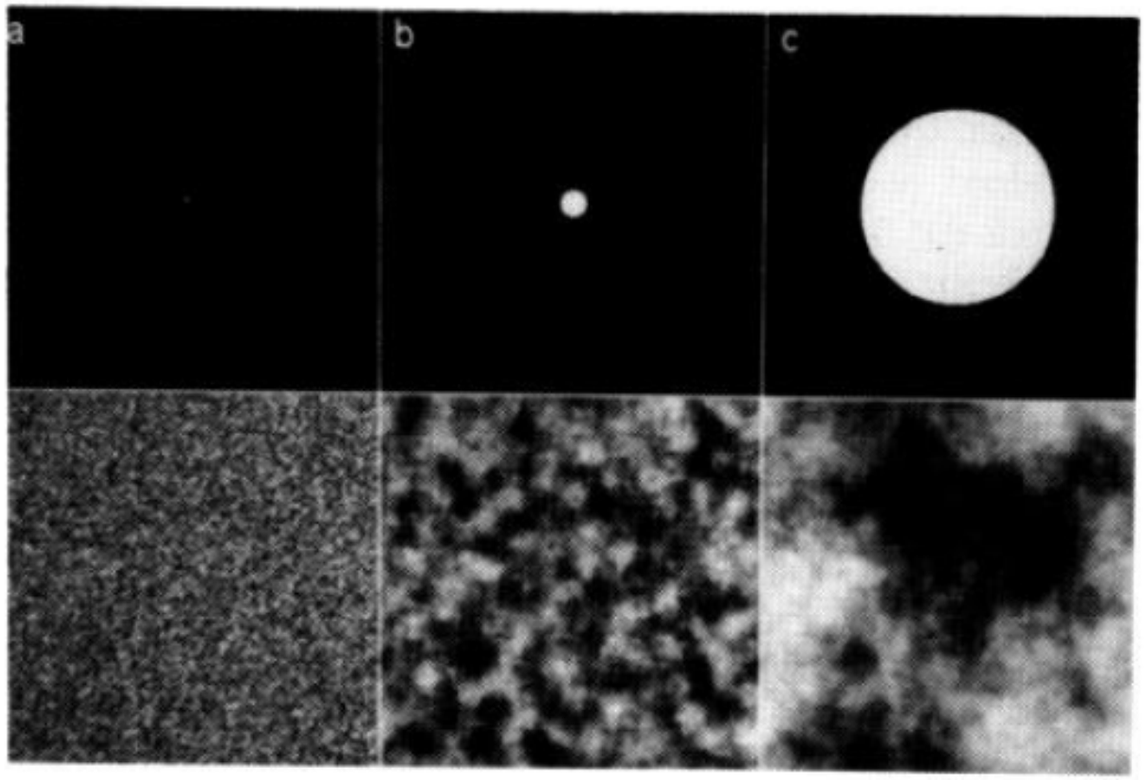


Fig. 2 Different sizes of spot

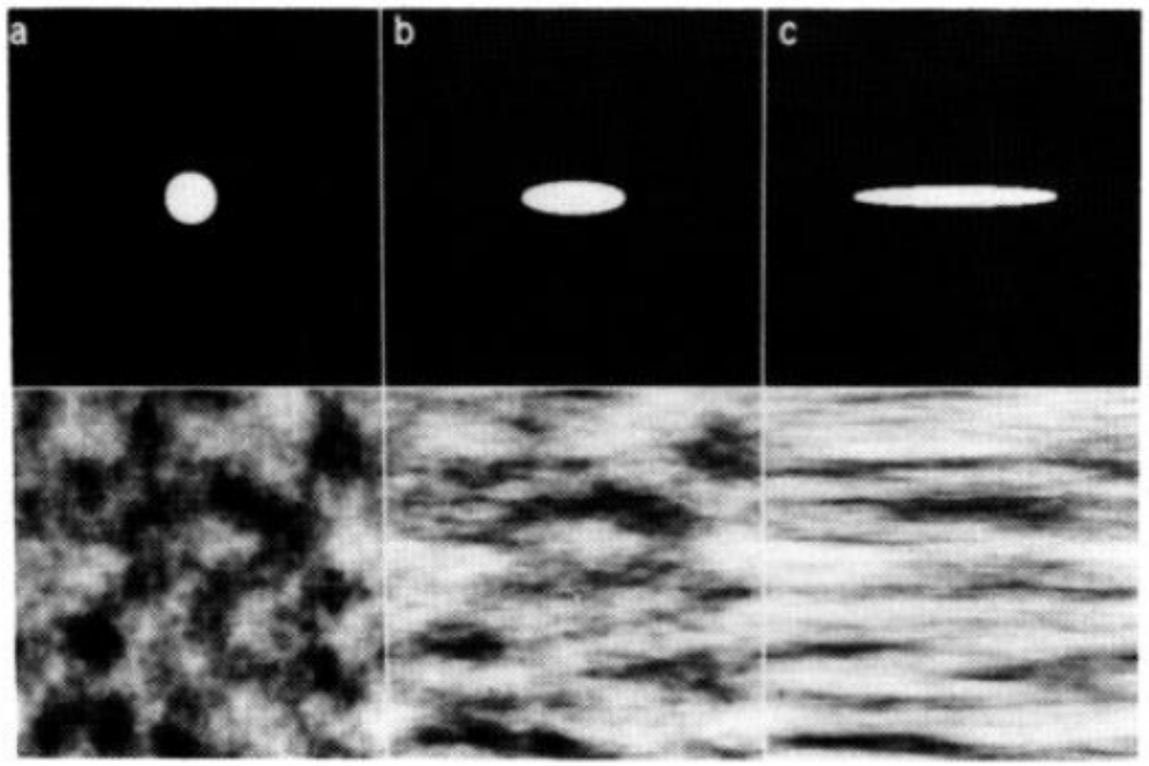
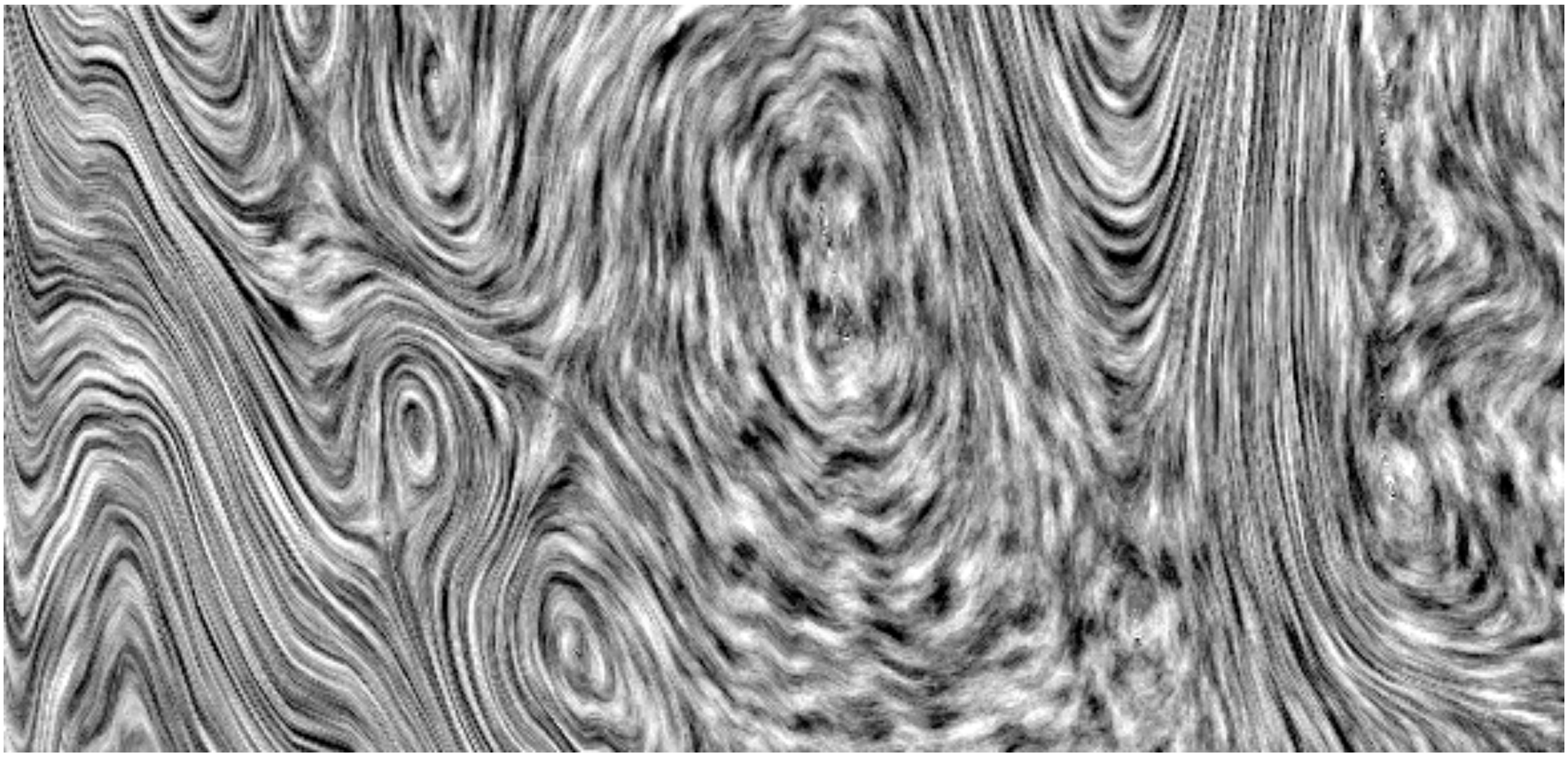


Fig. 6 Non-proportional scaling

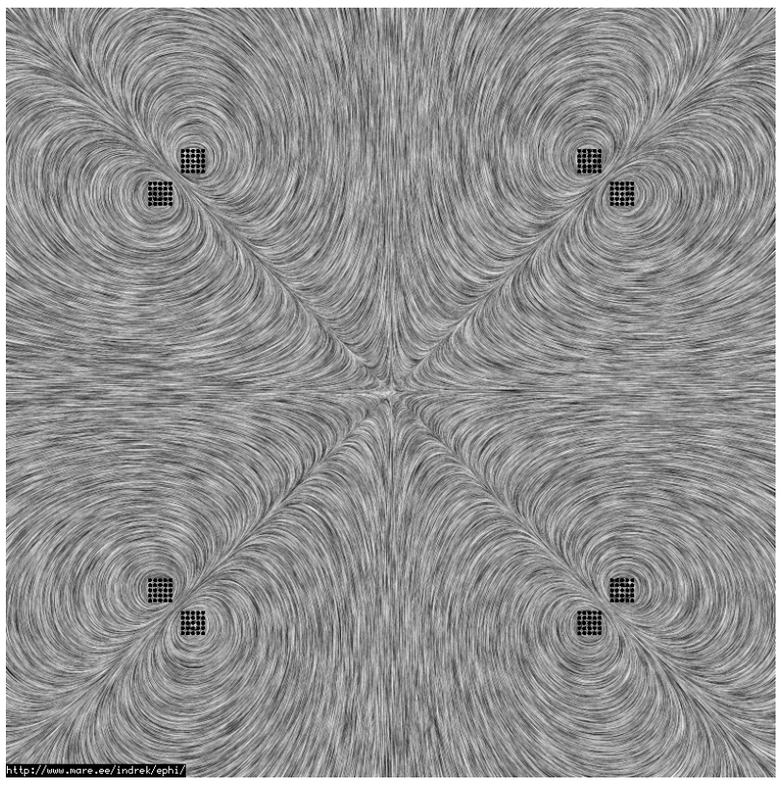
Spot Noise



The velocity of the flow is encoded in the length of the streaks used to smear the texture.

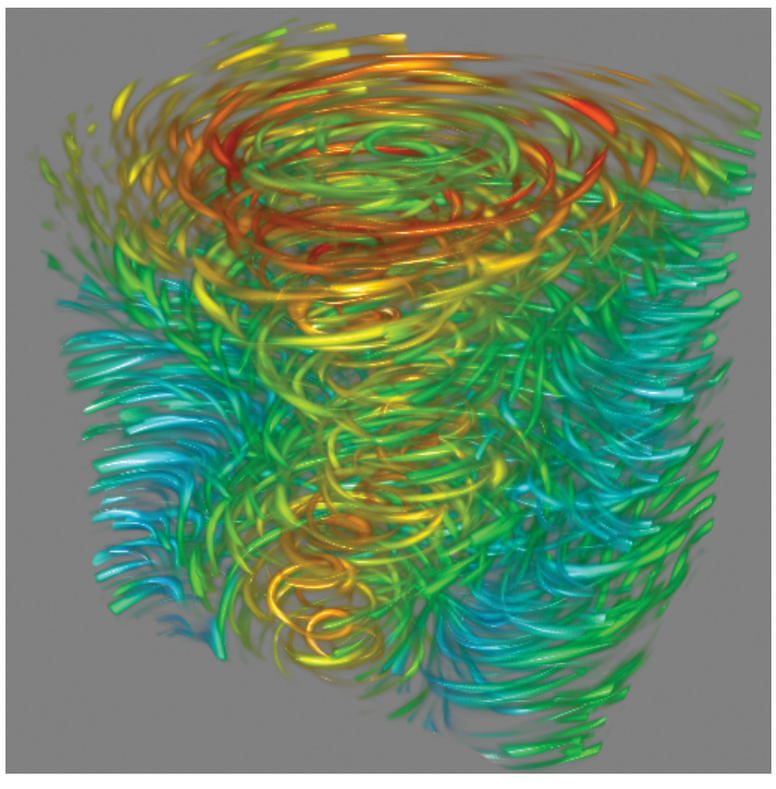
LIC – Line Integral Convolution

- (Cabral/Leedom, Siggraph 1993)
- A global method to visualize vector fields



2D vector field

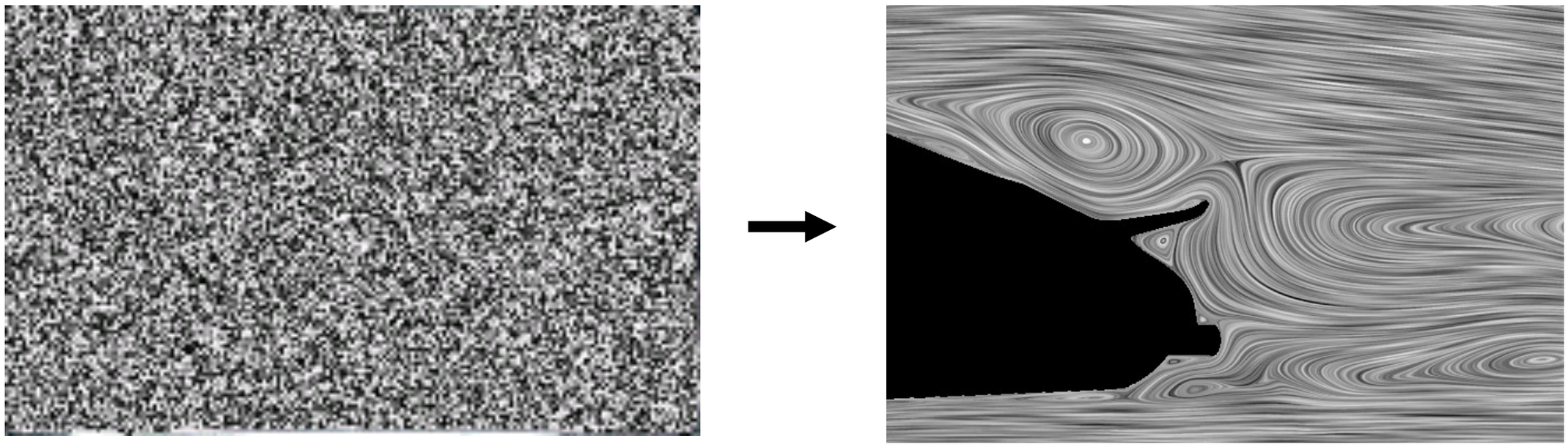
vector field on surface (often called 2.5D)



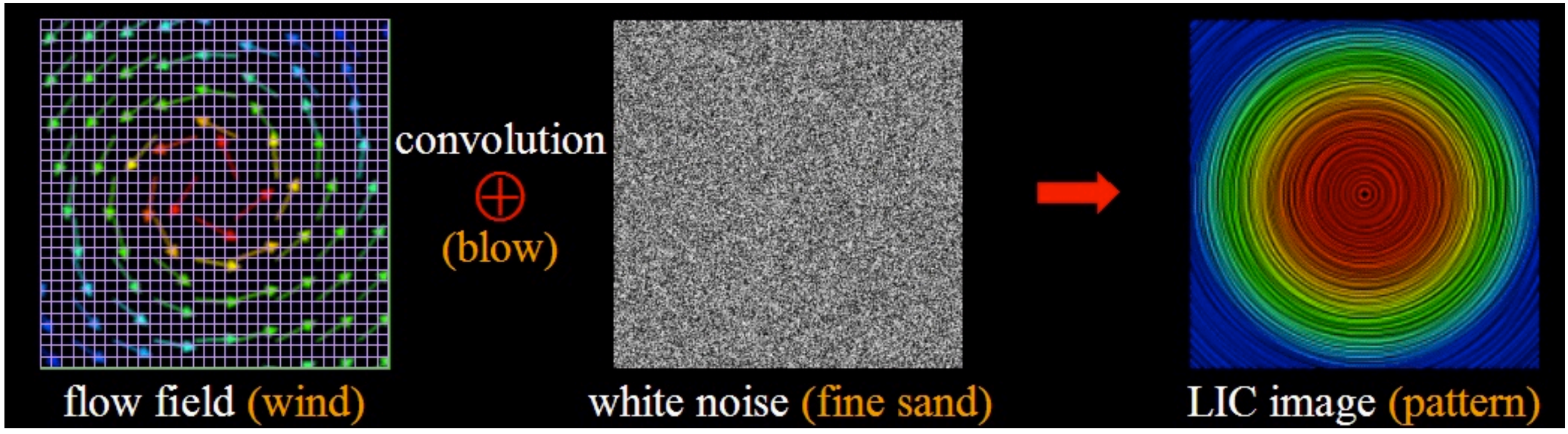
3D vector field

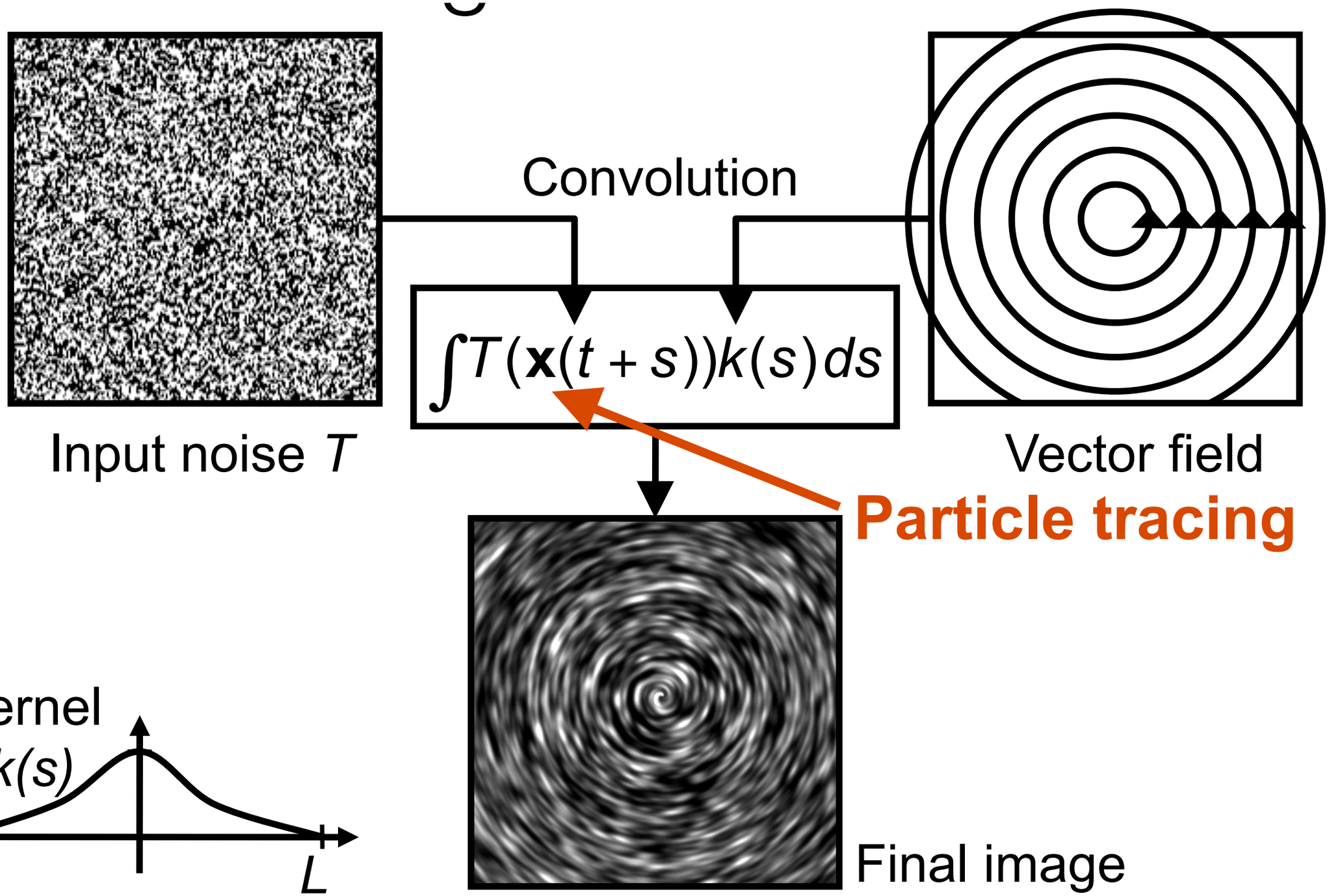
Idea of LIC

- **Global** visualization technique; not only one particle path
- Start with a **random texture**
- **Smear** out this texture along the path lines in a vector field, results in
    - **Low correlation** of intensity values **between** neighboring lines,
    - But **high correlation along** them



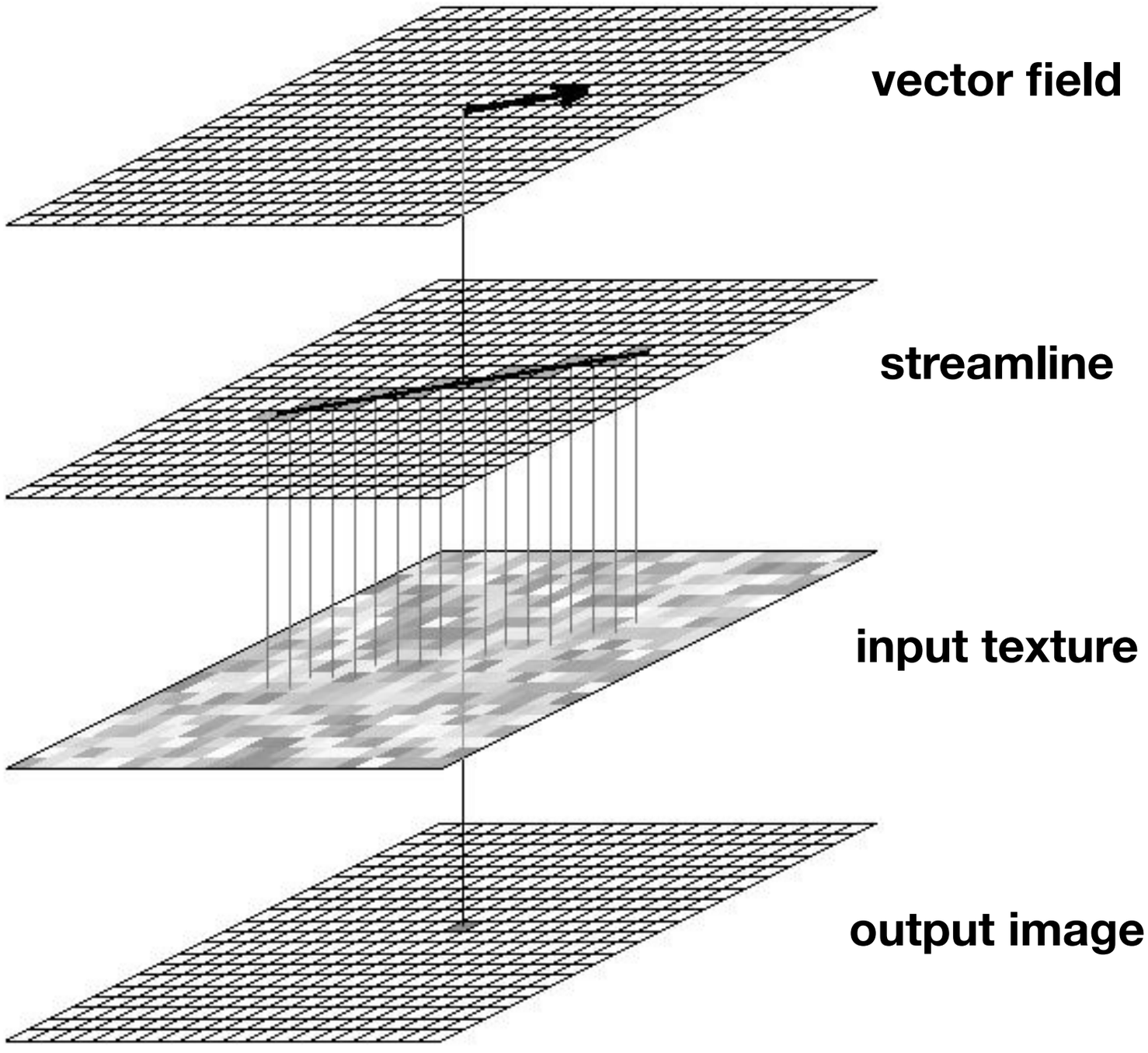
Idea of LIC

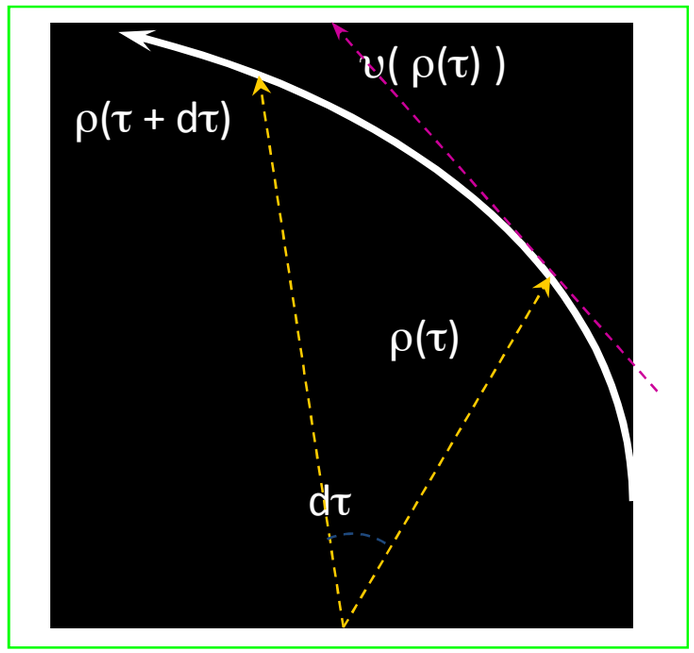

convolution
(blow)
flow field (wind)
white noise (fine sand)
LIC image (pattern)


Convolution
$\int T(\mathbf{x}(t+s))k(s)ds$
Input noise T
Vector field
Particle tracing
kernel
$k(s)$
$-L$
L
Final image

Algorithm for 2D LIC

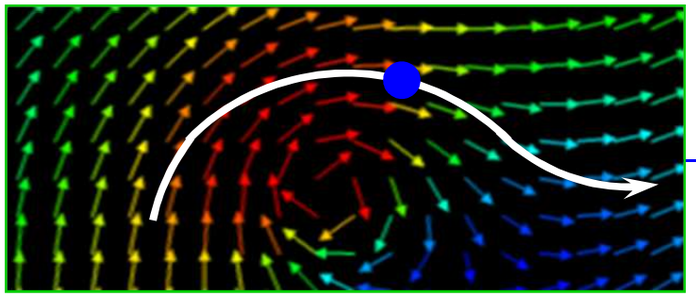
- Convolve a random texture along the streamlines

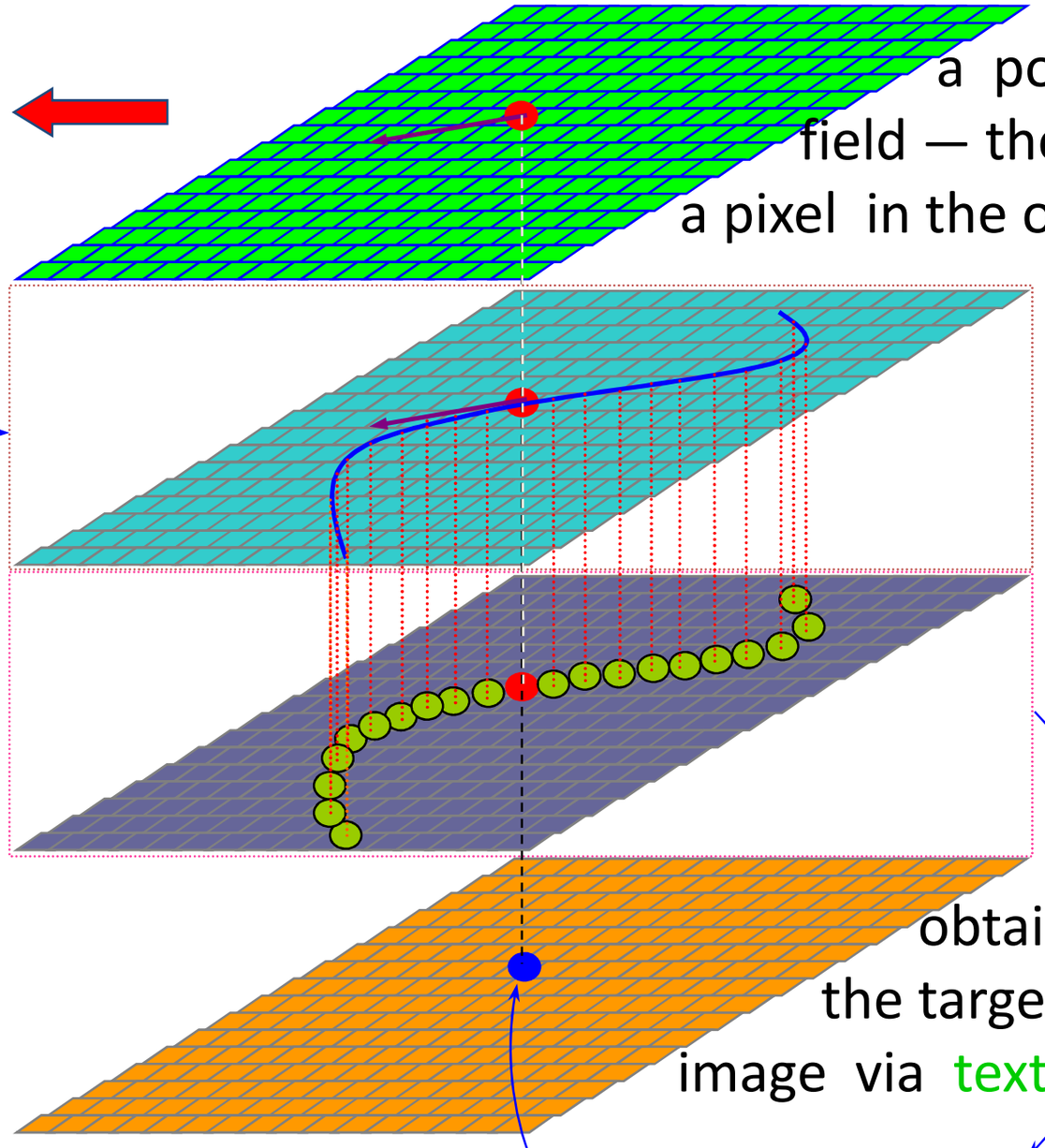




$$d(\rho(\tau)) / d\tau = \upsilon(\rho(\tau))$$

$$\rho(\tau+\Delta\tau) = \rho(\tau) + \int_{\tau}^{\tau+\Delta\tau} \upsilon(\rho(\tau))d\tau$$





a point in the flow field — the counterpart of a pixel in the output LIC image

locate a set of pixels hit by the streamline

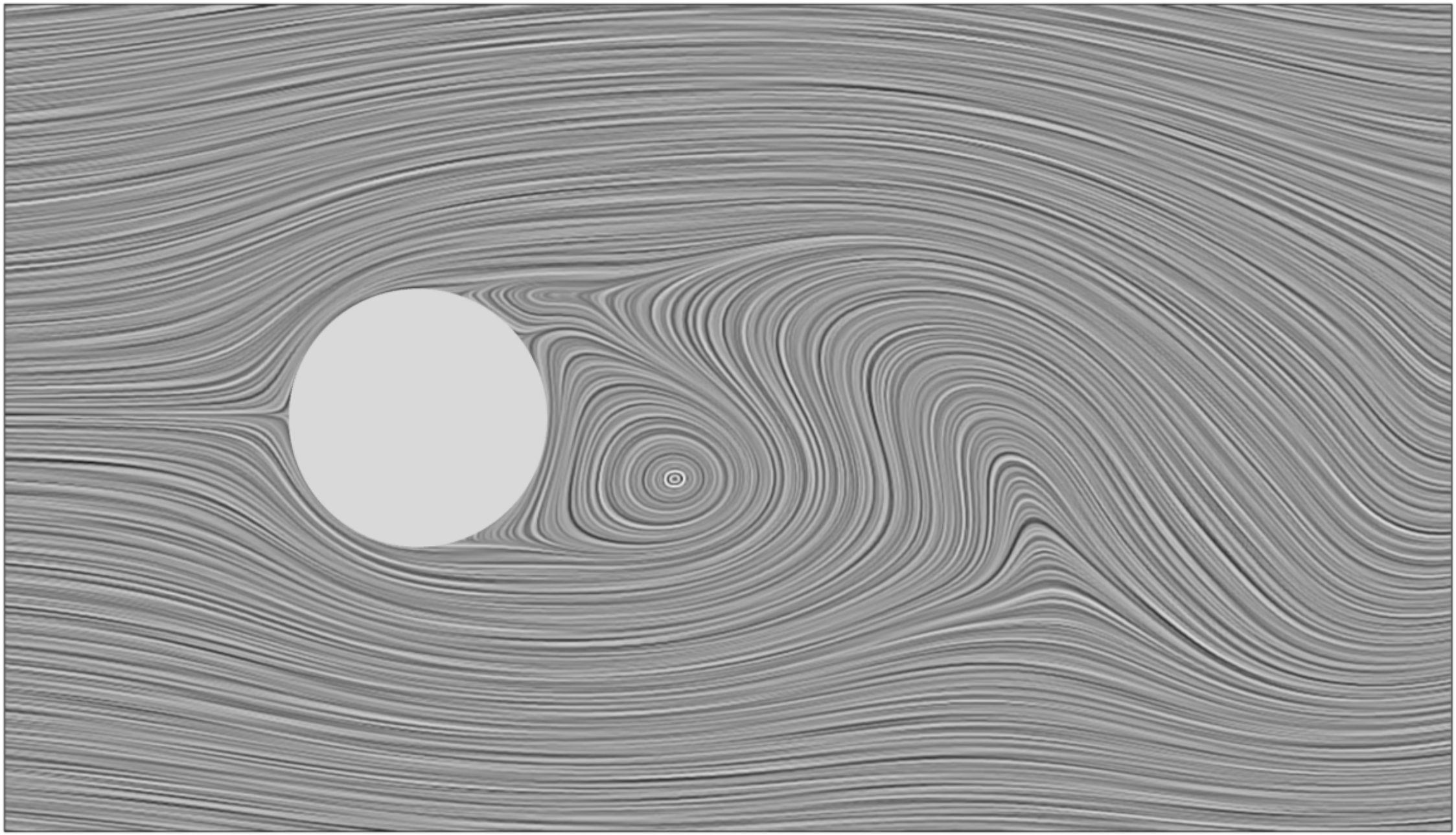
index the input noise for the texture values

obtain the value of the target pixel in the LIC image via texture convolution

$$\frac{\sum (\text{texture}[i] \times \text{weight}[i])}{\sum \text{weight}[i]}$$

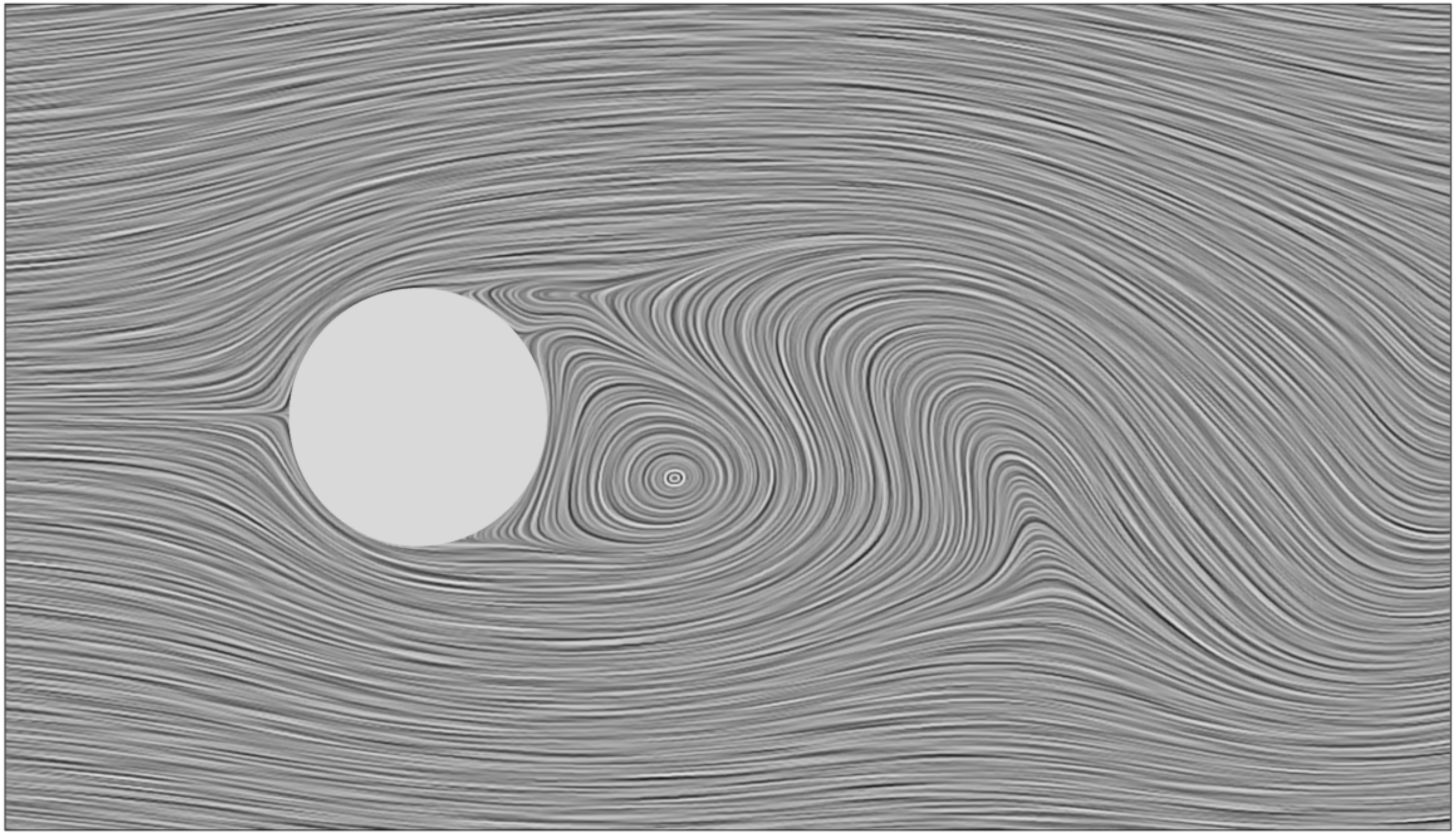
weighting is governed by a low-pass filter

2D flow behind a cylinder



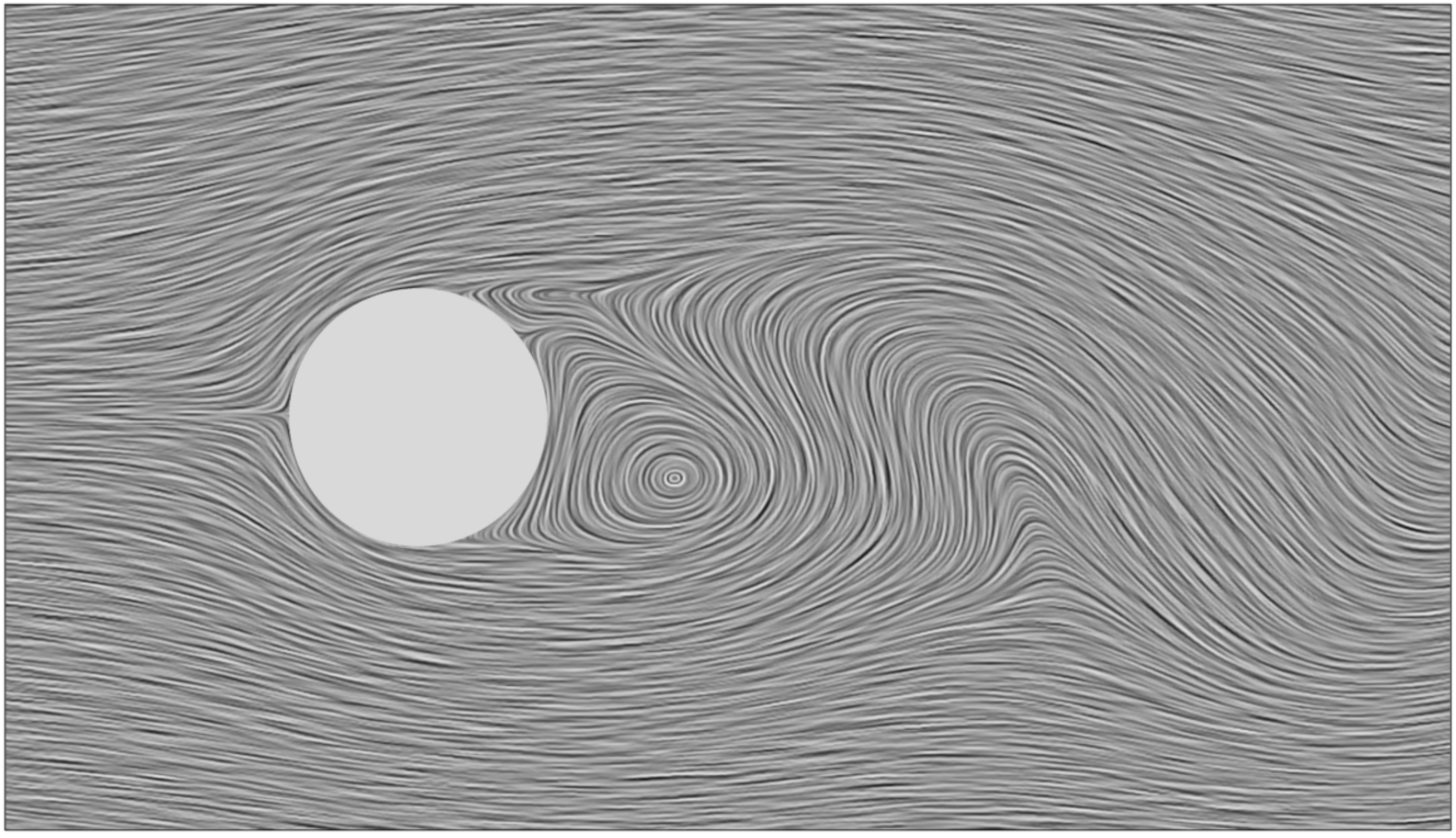
Filter length influences the quality of LIC images
filter length = 100

2D flow behind a cylinder



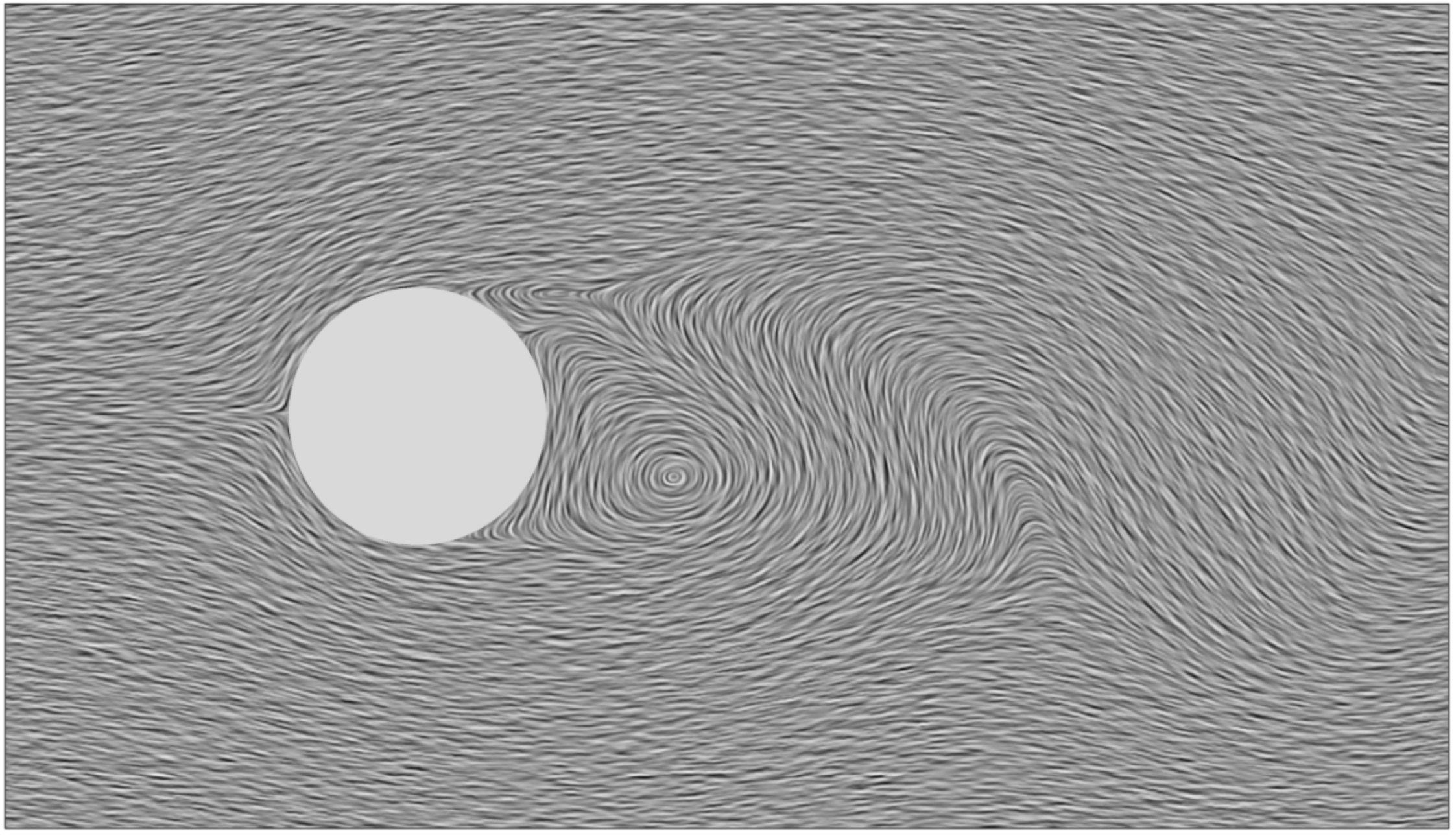
Filter length influences the quality of LIC images
filter length = 50

2D flow behind a cylinder



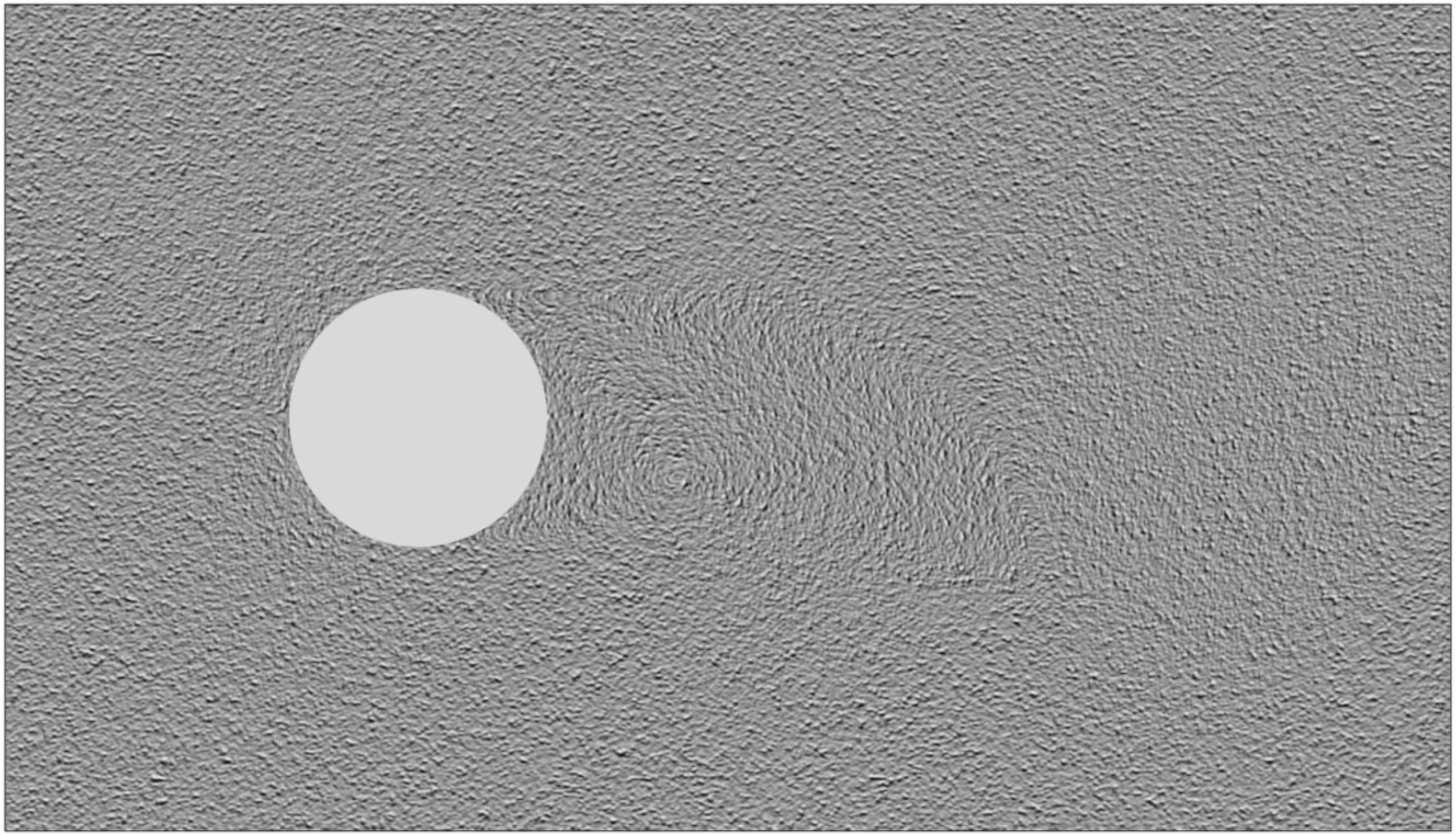
Filter length influences the quality of LIC images
filter length = 25

2D flow behind a cylinder



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

2D flow behind a cylinder



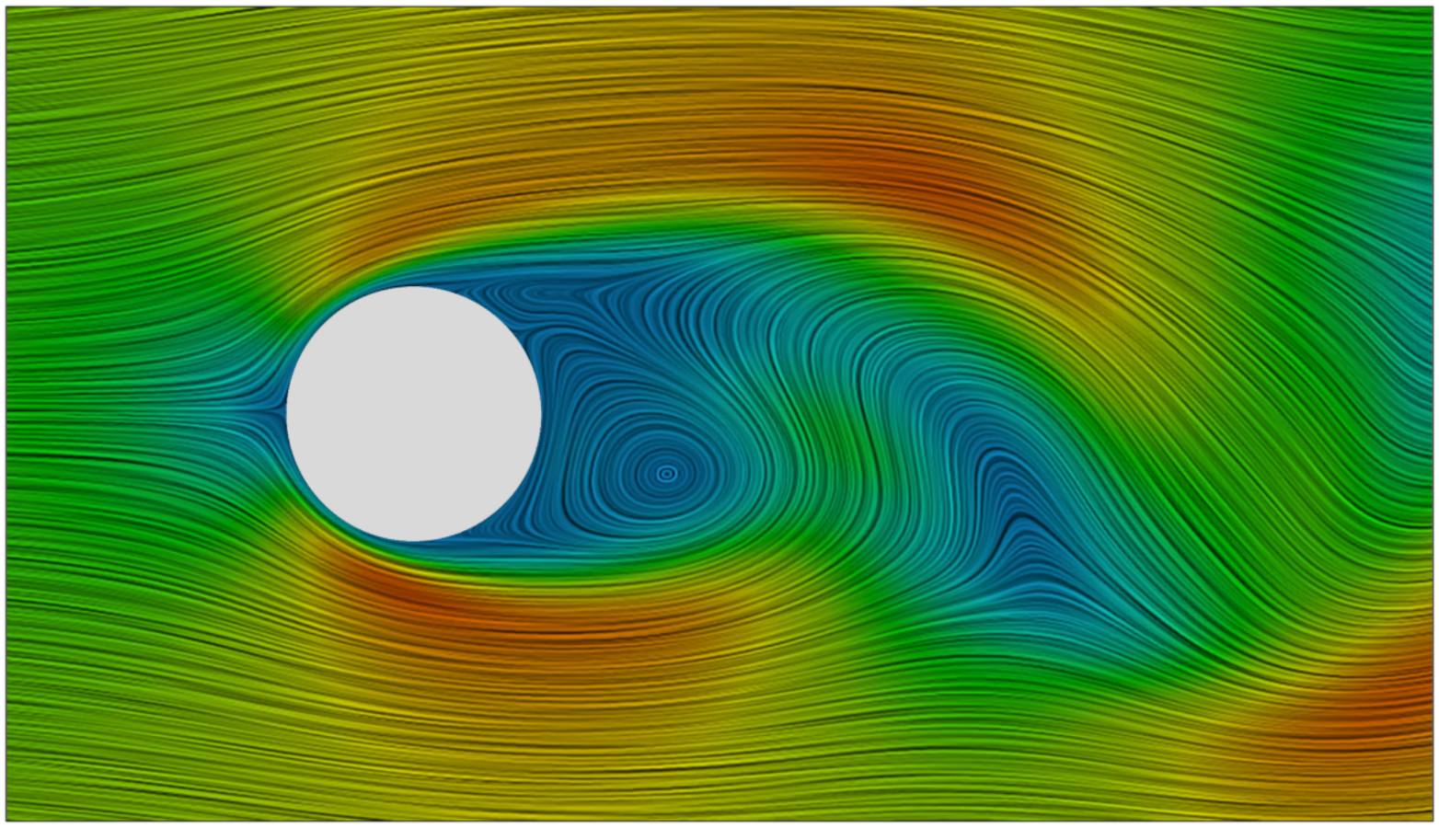
Filter length influences the quality of LIC images
filter length = 1

LIC Summary

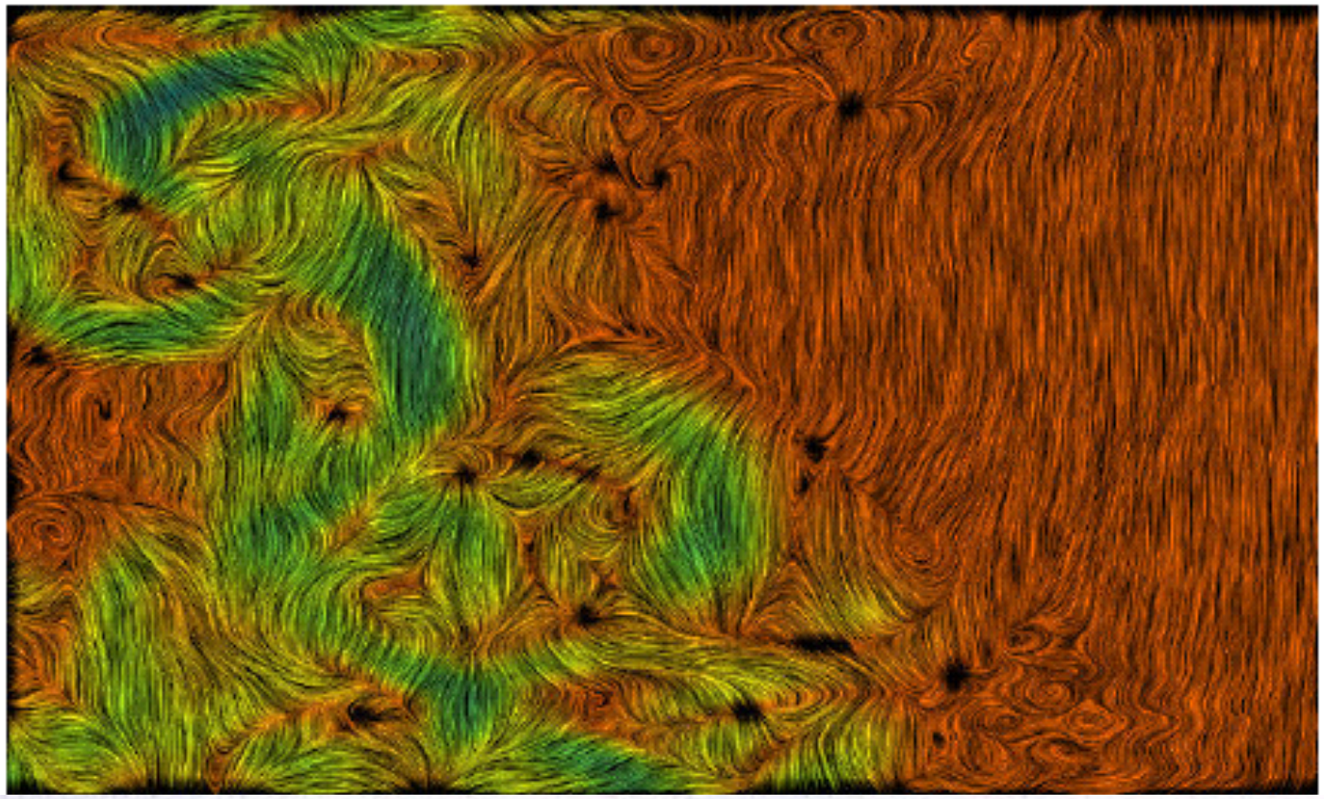
- Improving LIC in the following directions:
  - Combination with color coding
  - Special applications (motion blur…)
  - Adding flow orientation
  - Performance
  - LIC on surfaces
  - LIC for 3D flows
  - LIC for unsteady flows

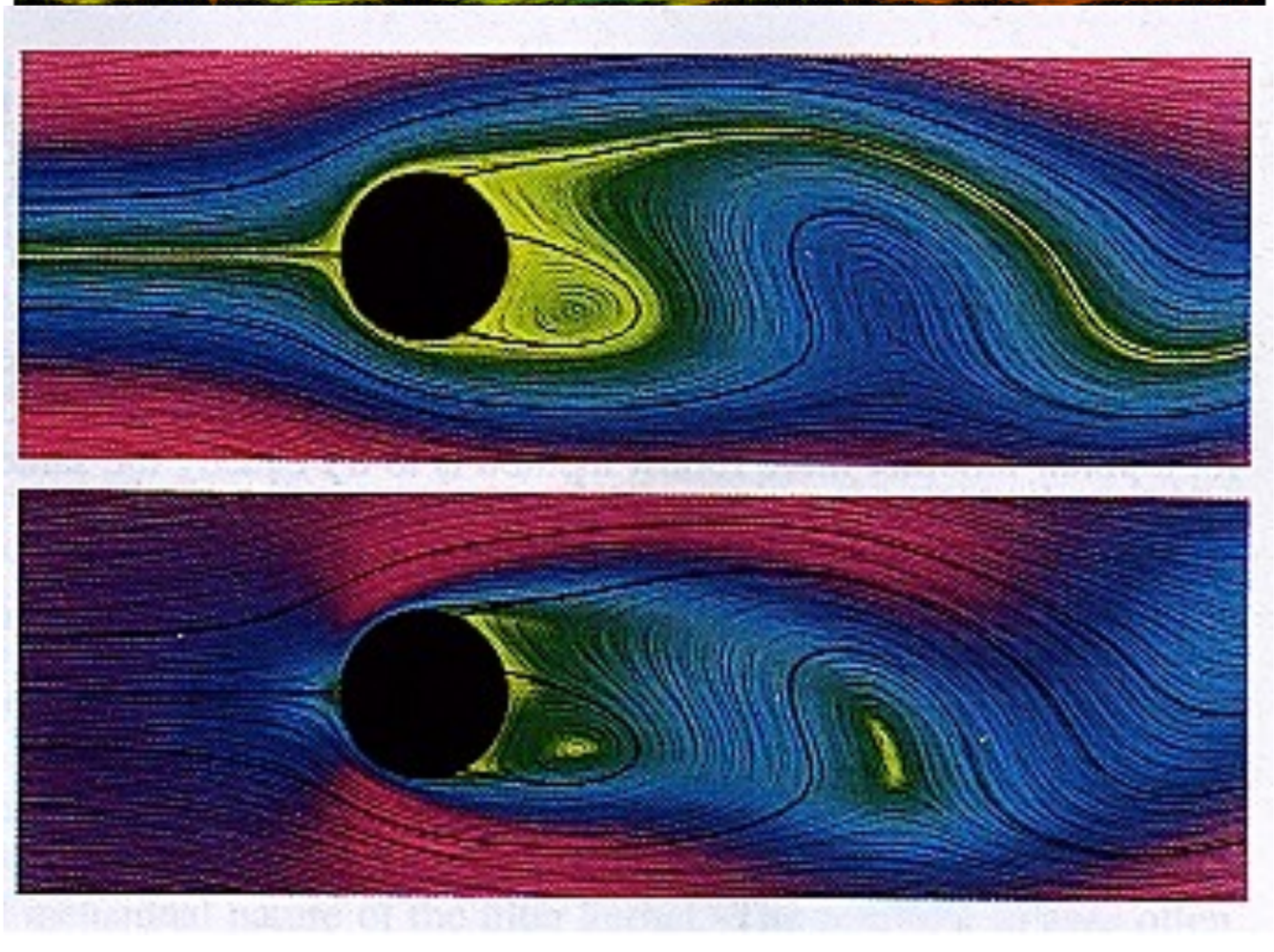
LIC Color Coding

- Usually, LIC does not use the color channel
  - Could use color to encode scalar quantities



Velocity magnitude encoded using color

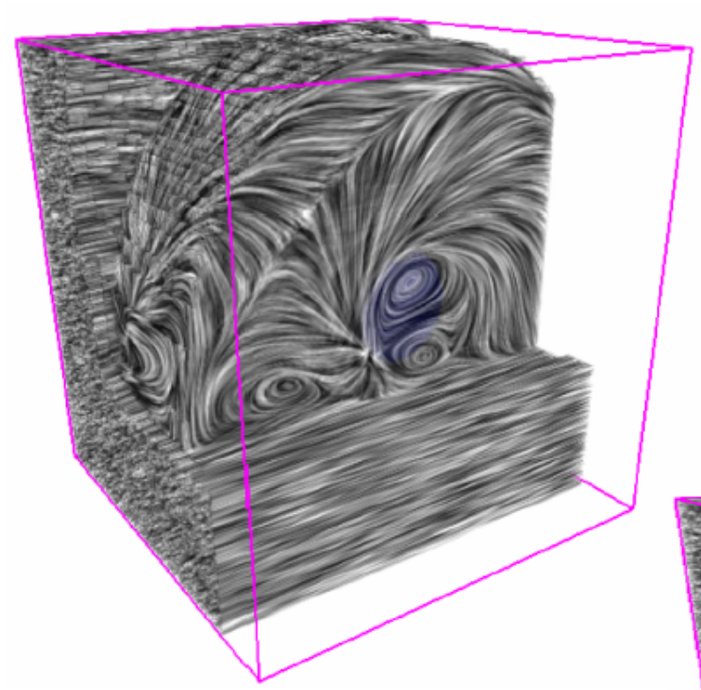


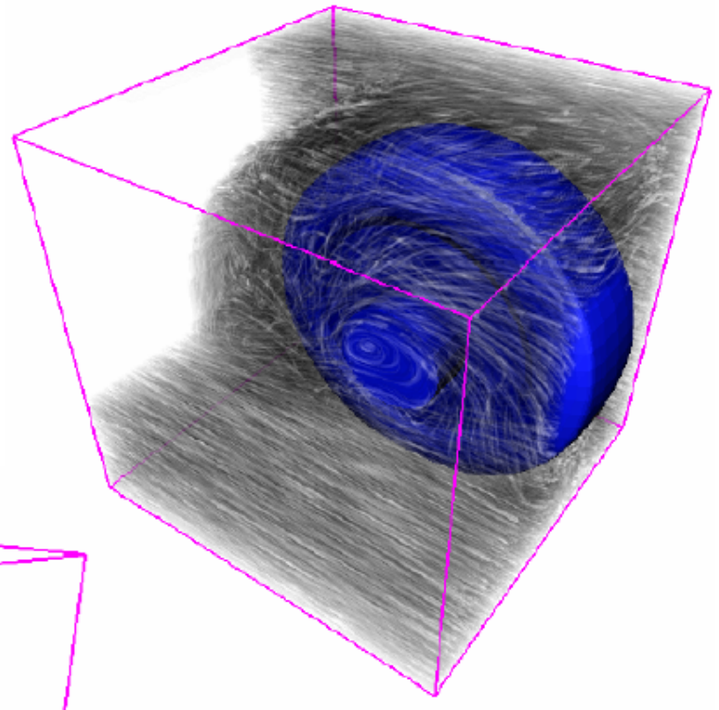


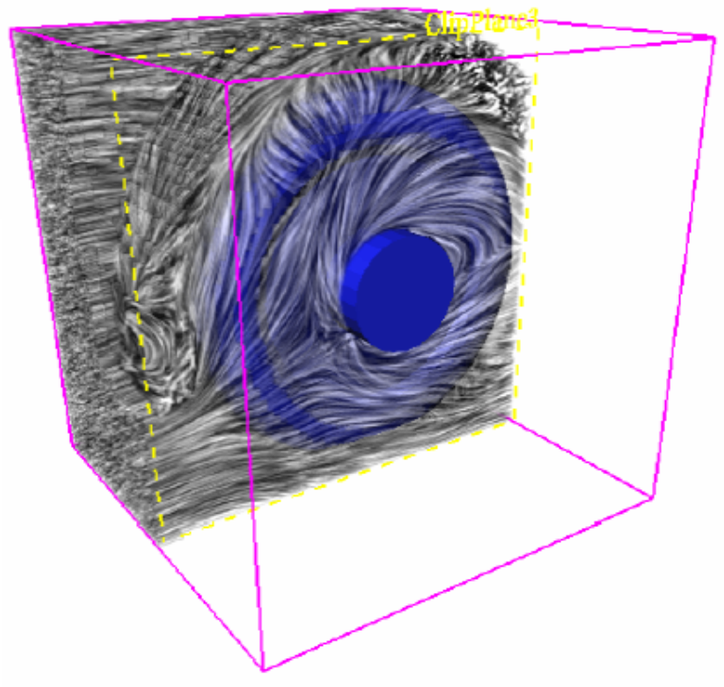
LIC and color coding of velocity magnitude

LIC for 3D Flows

- LIC concept easily extendable to 3D
- Problem: rendering!



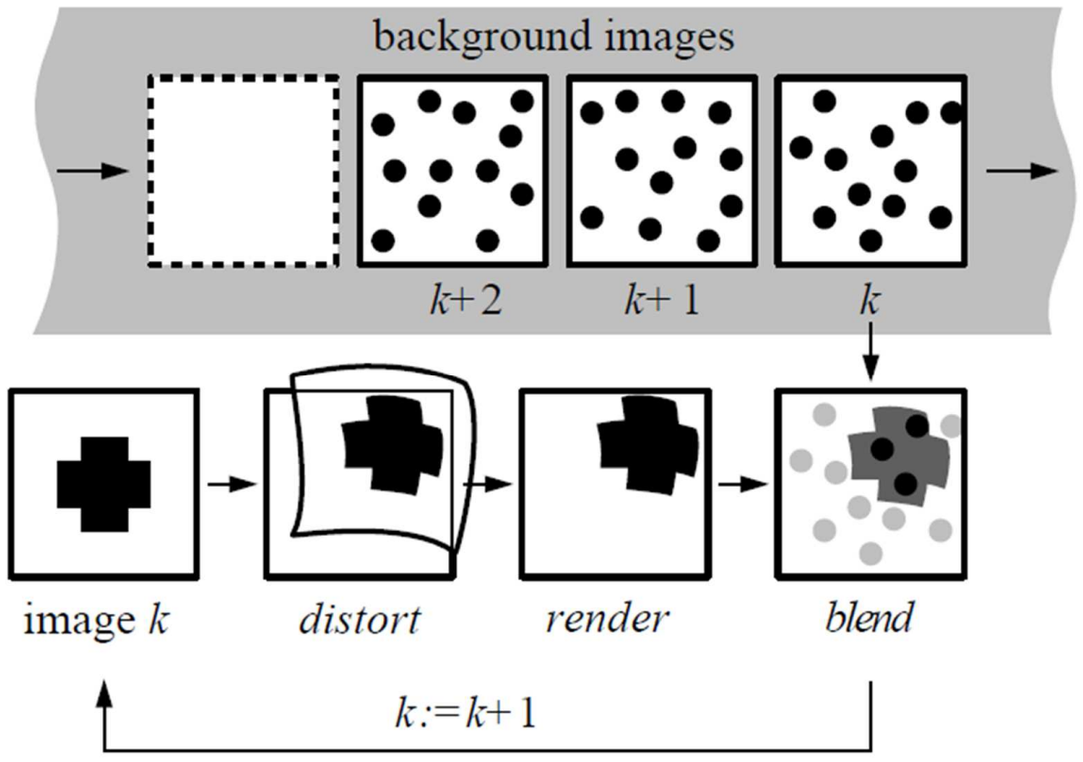


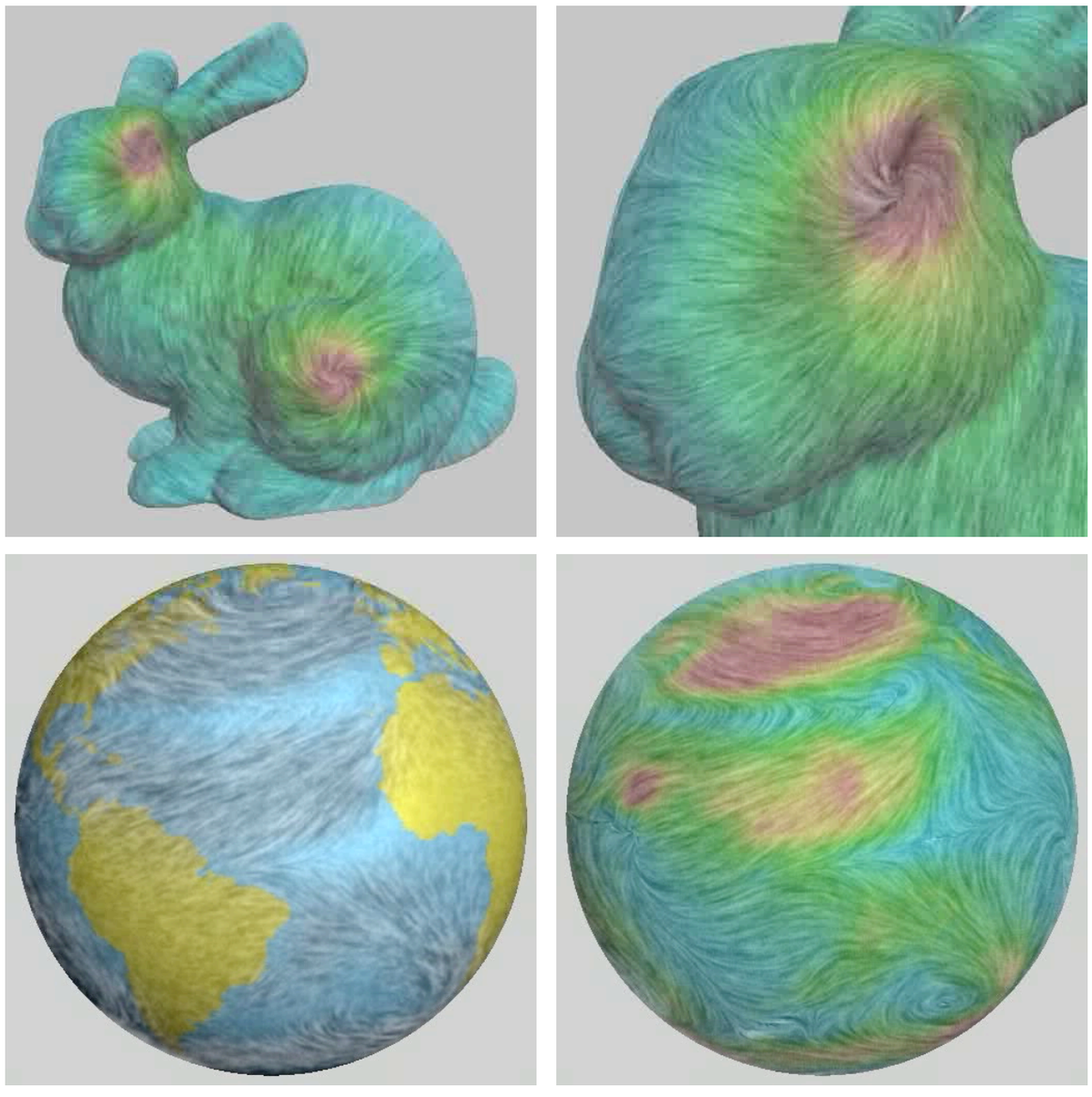


3D LIC can only reveal interesting structures if some data is discarded.

Images courtesy of
[Rezk-Salama et al. 99]

IBFV: Image-Based Flow Visualization (Advect Dye in Image-Space)

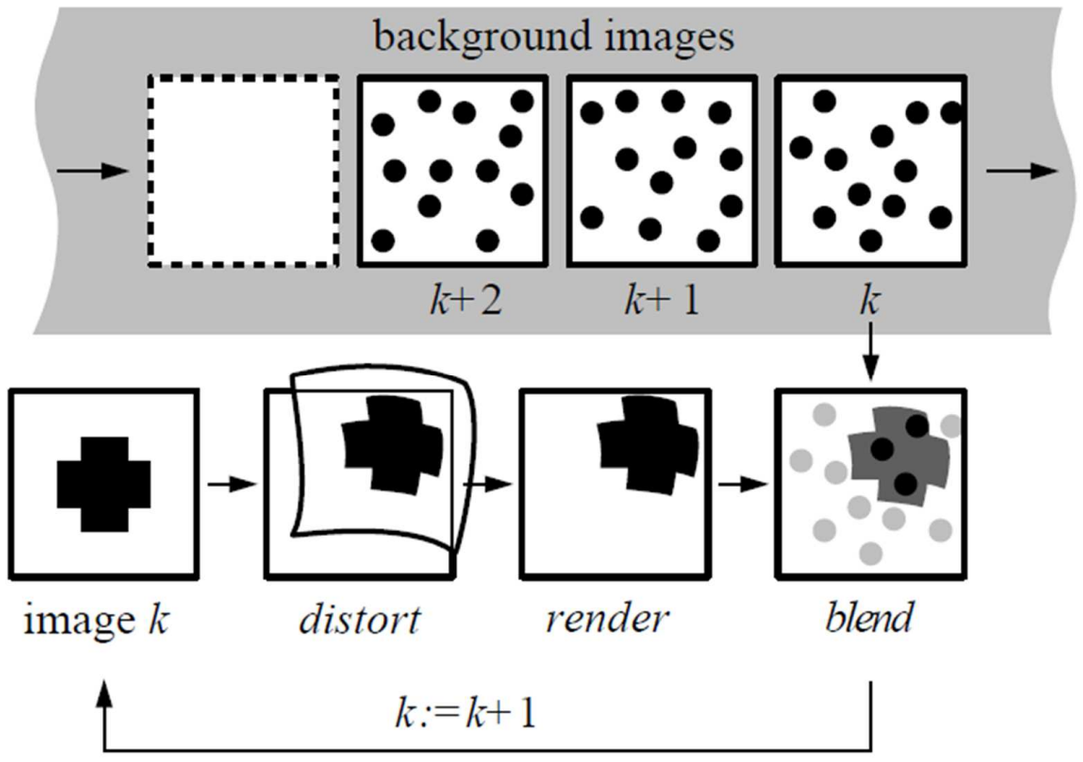


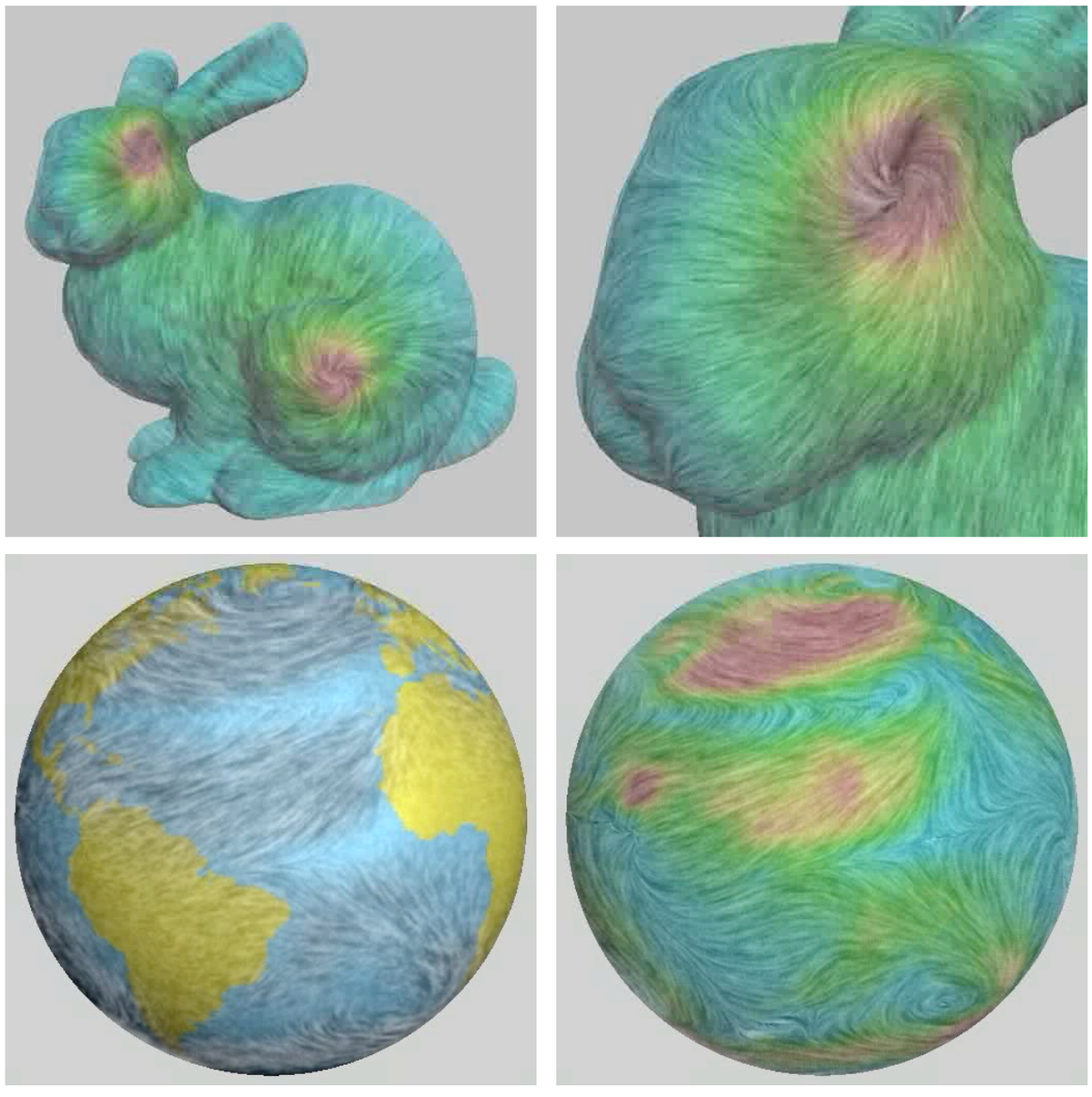


http://www.win.tue.nl/~vanwijk/ibfv/

http://www.win.tue.nl/~vanwijk/ibfvs/

IBFV: Image-Based Flow Visualization (Advect Dye in Image-Space)





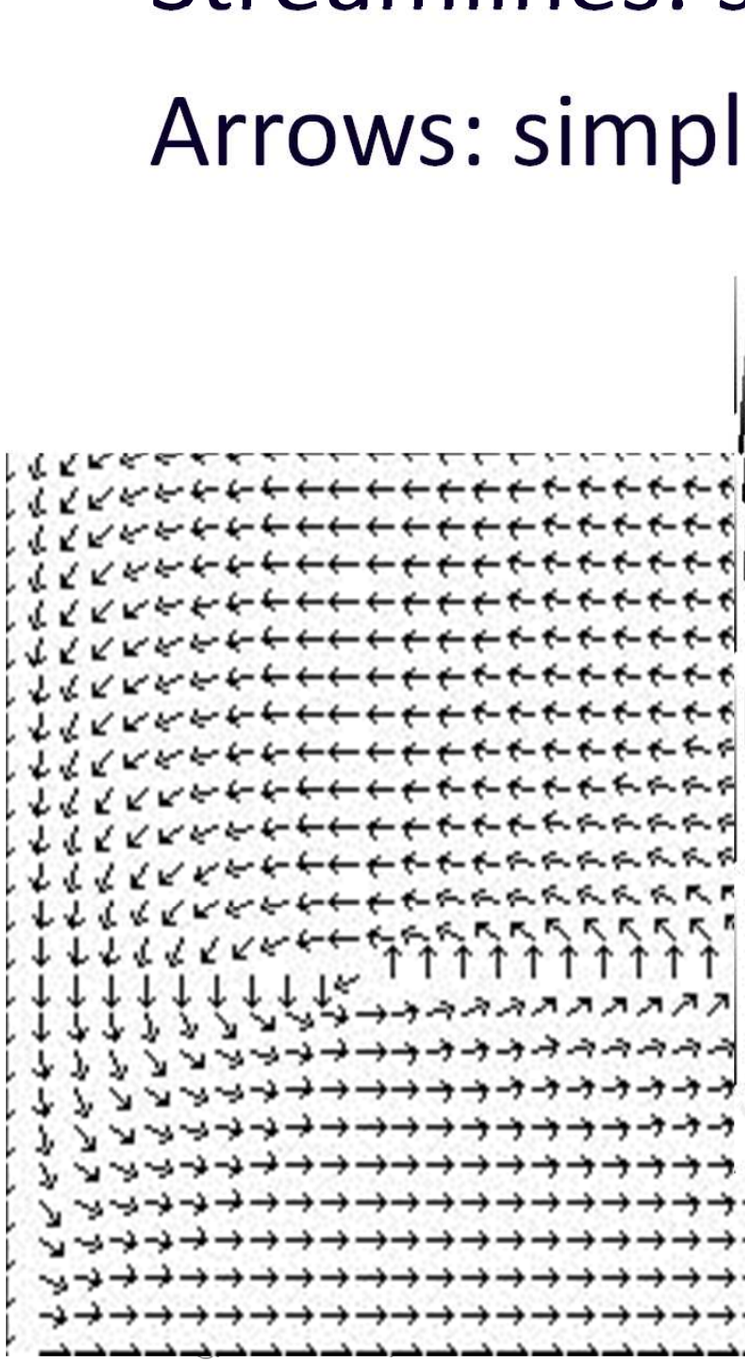
http://www.win.tue.nl/~vanwijk/ibfv/

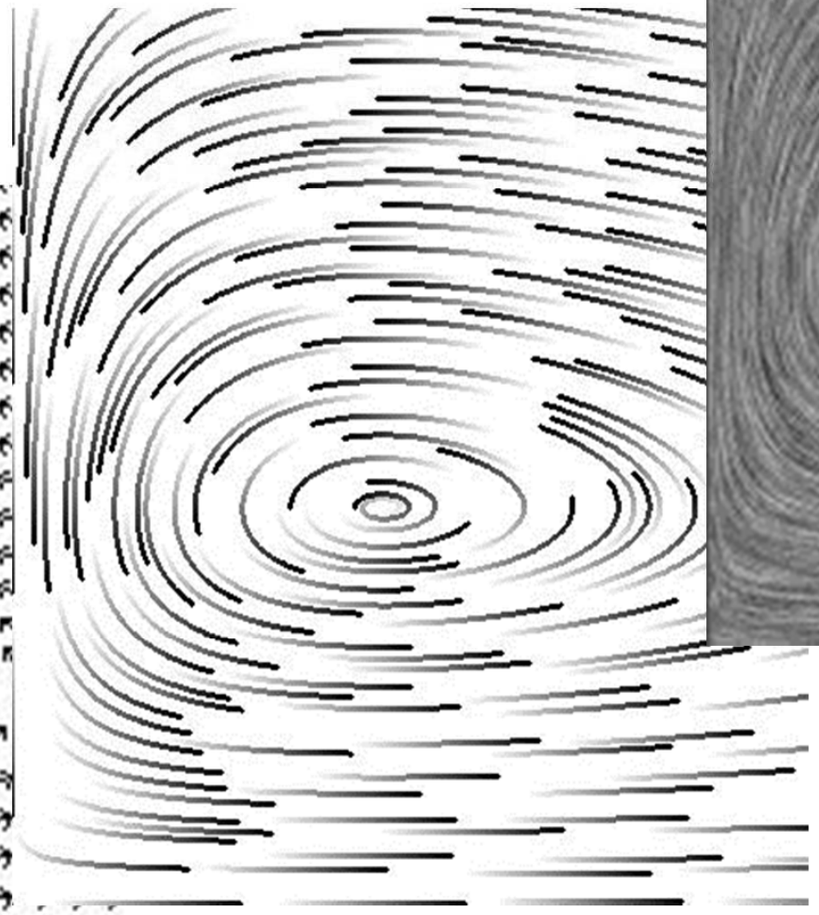
http://www.win.tue.nl/~vanwijk/ibfvs/

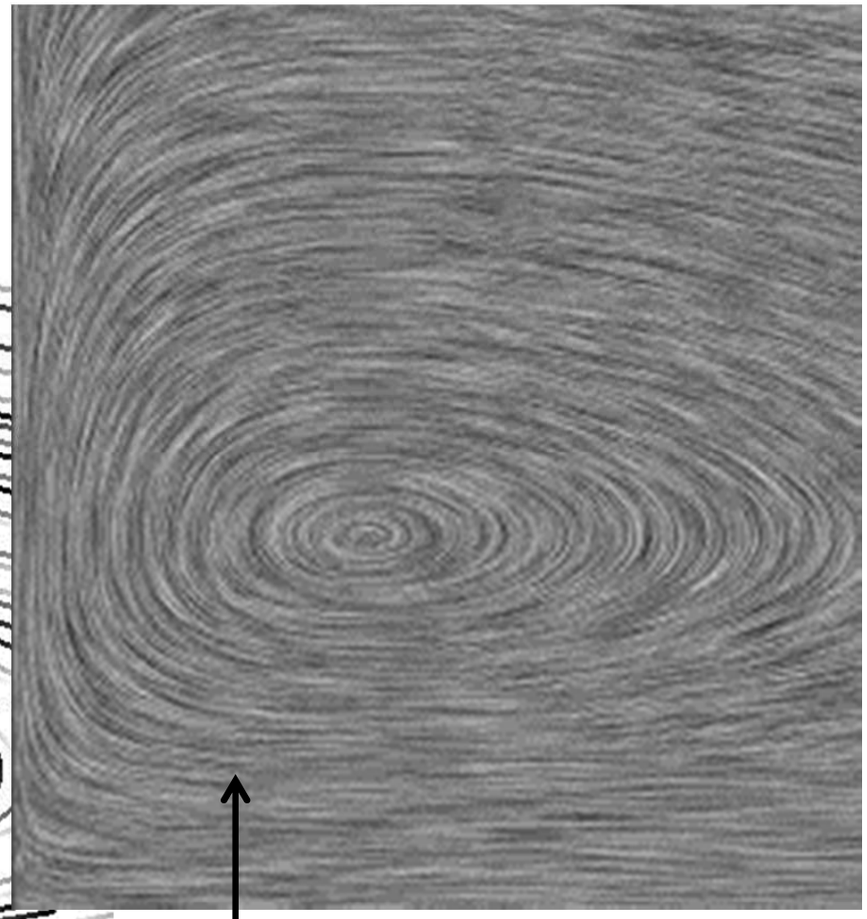
Arrows vs. Streamlines vs. Textures

Streamlines: selective

Arrows: simple







Textures:
2D-filling

Investigating Swirl and Flow Motion

Robert S Laramee
Daniel Weiskopf
Jürgen Schneider
Helwig Hauser

Investigating Swirl and Flow Motion

Robert S Laramee
Daniel Weiskopf
Jürgen Schneider
Helwig Hauser

Investigating Swirl and Flow Motion

Robert S Laramee
Daniel Weiskopf
Jürgen Schneider
Helwig Hauser

Lec24 Reading

- Time and Streak Surfaces for Flow Visualization in Large Time-Varying Data Sets. Harinarayan Krishnan, Christoph Garth, Kenneth I. Joy. IEEE Trans. Vis. Comput. Graph. 15(6): 1267-1274 (2009)

Assignment 06

Assigned: Monday, April 10
Due: Monday, April 24, 4:59:59 pm

Reminder
Project Milestones 03/04

Assigned: Wednesday, March 29
03 (Talk) Due: Wednesday, April 26, 4:59:59 pm
04 (Report) Due: Wednesday, May 3, 4:59:59 pm