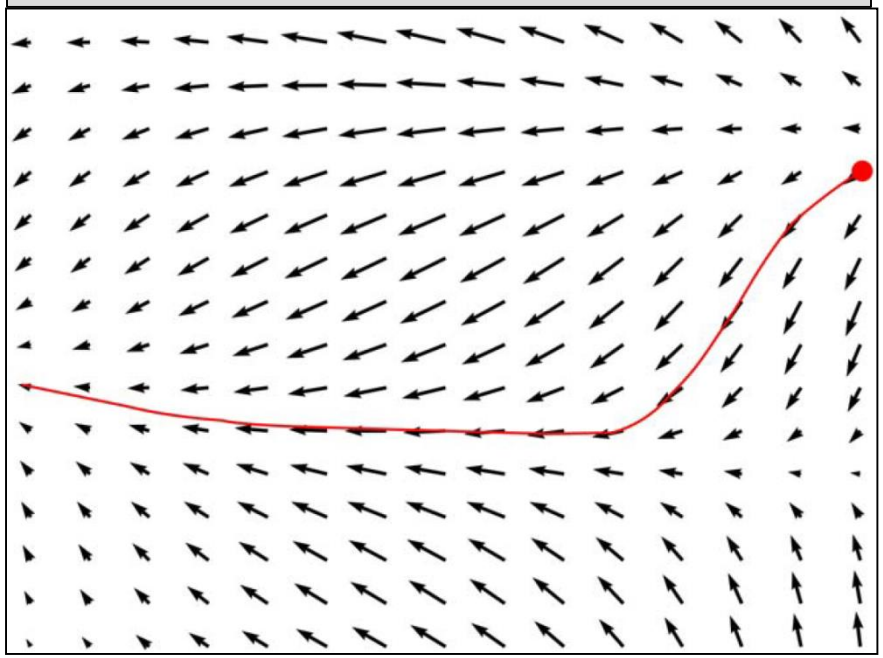
Vector Field Visualization: Introduction

What is a Vector Field?

A simple 2D **steady** vector field



A vector-valued function that assigns a vector (with direction and magnitude) to any given point.

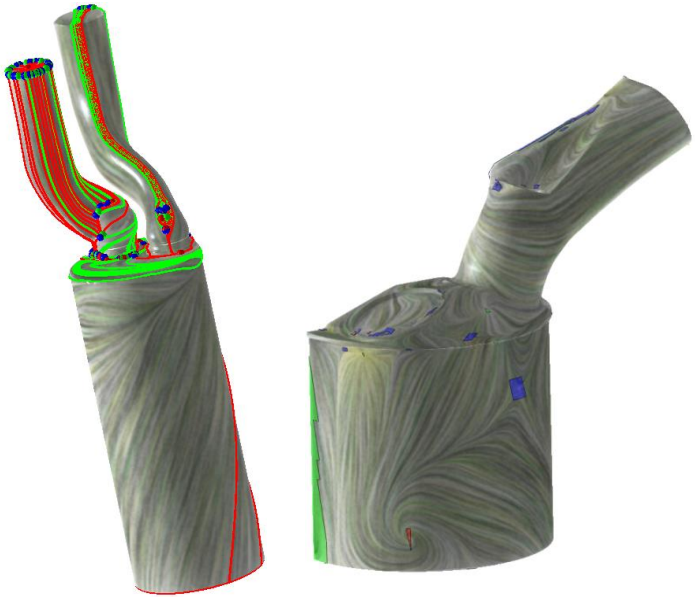
It typically can be expressed as an ordinary differential equation, i.e. ODE.

$$\frac{d\varphi(x)}{dt} = V(x)$$

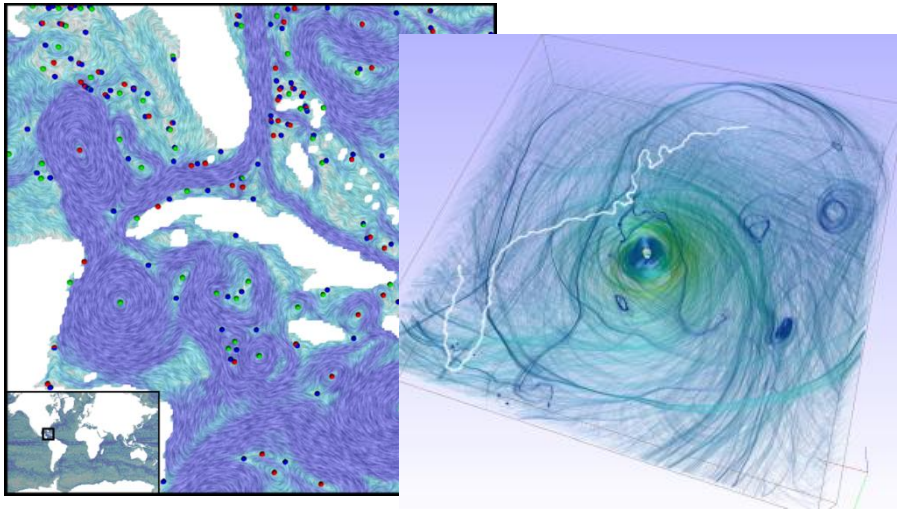
Its solution gives rise of a "**flow**", which consists of densely placed particle **trajectories** (i.e. the red curve shown in the example).

Why It is Important?

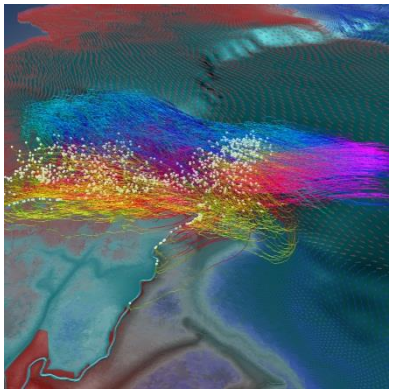
Applications in Engineering and Science

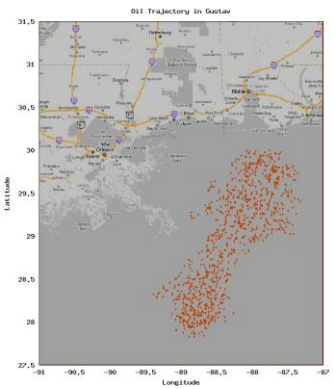


Automotive design
[**Chen** et al. TVCG07,TVCG08]

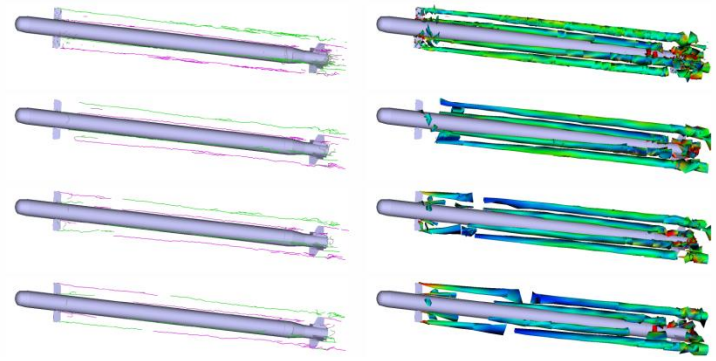


Weather study [Bhatia and Chen et al. TVCG11]



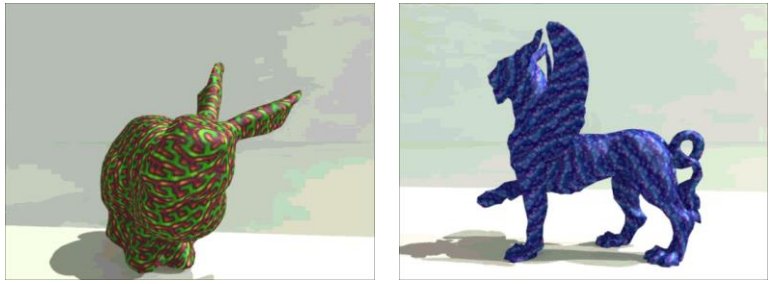


Oil spill trajectories [Tao et al. EMI2010]

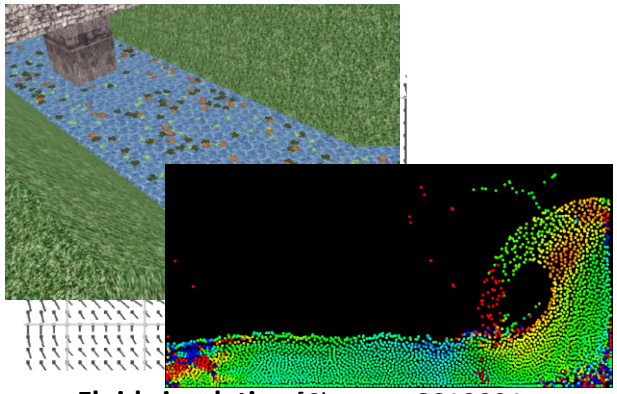


Aerodynamics around missiles [Kelly et al. Vis06]

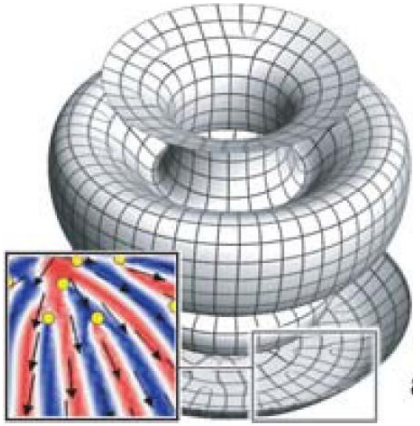
Applications in Computer Graphics



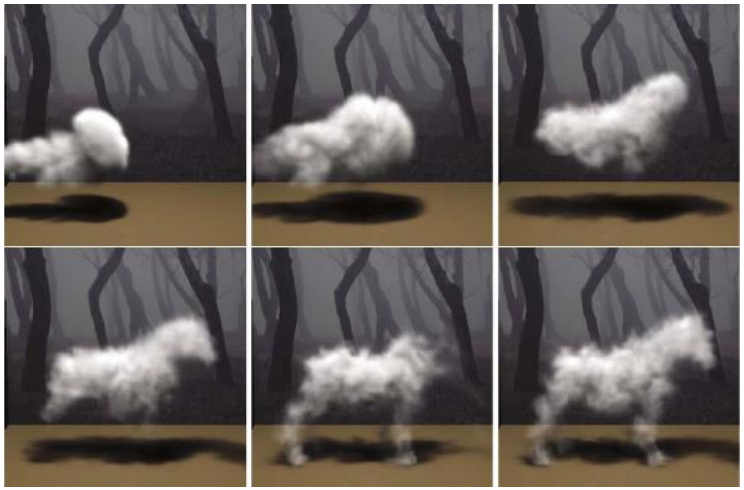
Texture Synthesis [**Chen** et al. TVCG11b]



Fluid simulation [Chenney SCA2004, Cao&Chen 2013]

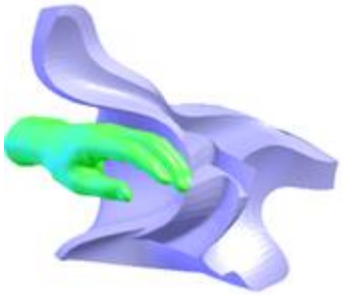


Parameterization
[Ray et al. TOG2006]



Smoke simulation [Shi and Yu TOG2005]

Painterly Rendering [Zhang et al. TOG2006]

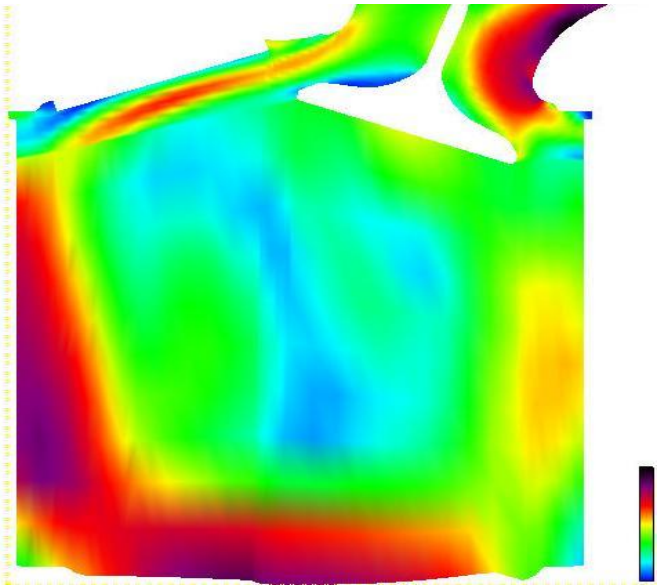


Shape Deformation
[von Funck et al. 2006]

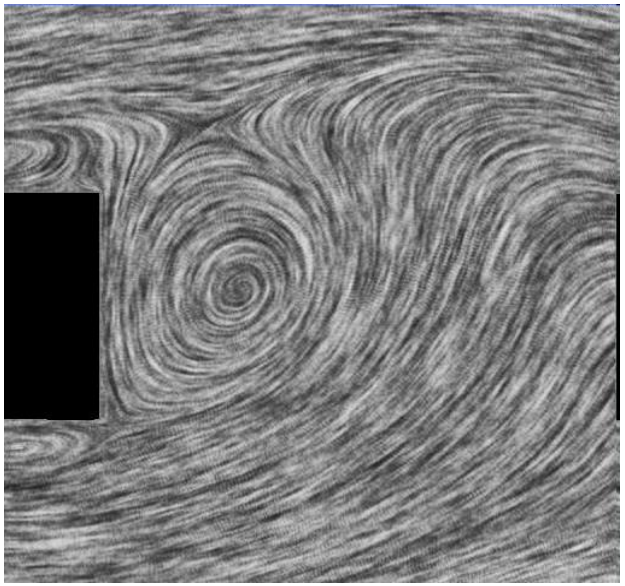
Why is It Challenging?

- Need to effectively visualize both *magnitude + direction,* often simultaneously
- Additional challenges:
  - large data sets
  - time-dependent data

magnitude only

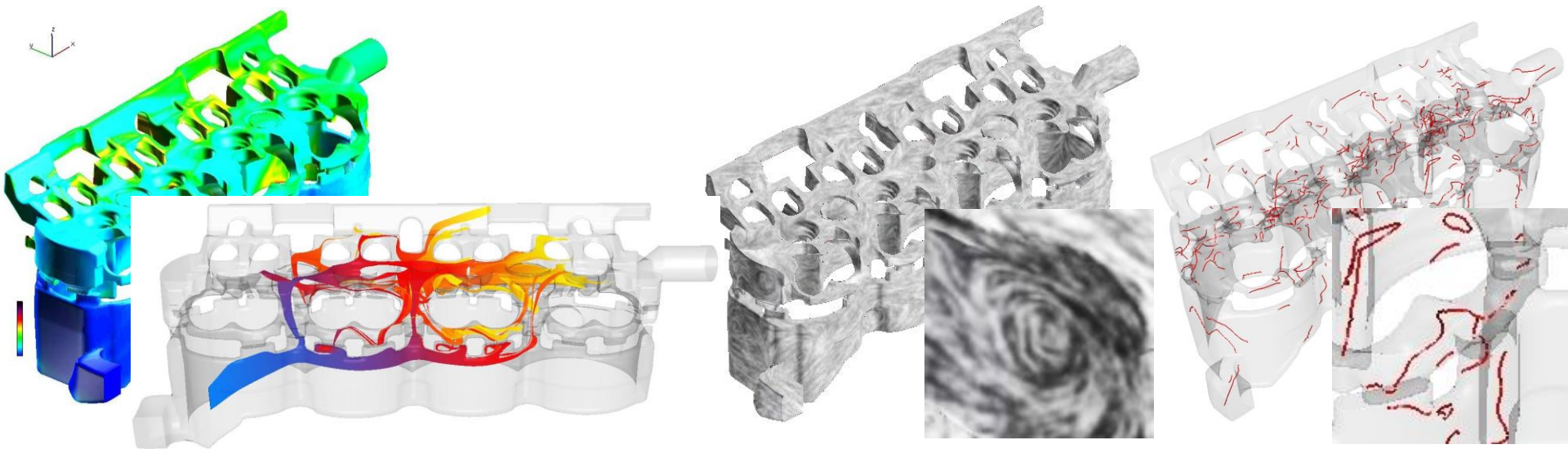


direction only



Classification of Visualization Techniques

- **Direct method:** overview of vector field, minimal computation, e.g. glyphs, color mapping.
- **Texture-based:** covers domain with a convolved texture, e.g., Spot Noise, LIC, LEA, ISA, IBFV(S).
- **Geometric:** a discrete object(s) whose geometry reflects (e.g. tangent to) flow characteristics, e.g. integral curves.
- **Feature-based:** both automatic and interactive feature-based techniques, e.g. flow topology, vortex core structure, coherent structure, LCS, etc.



Flow Data

Data sources:

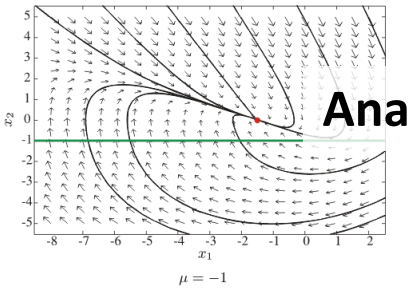
- flow simulation:
  - airplane- / ship- / car-design
  - weather simulation (air-, sea-flows)
  - medicine (blood flows, etc.)
- flow measurement:
  - wind tunnels, water channels
  - optical measurement techniques
- flow models (analytic):
  - differential equation systems (dynamic systems)

Simulated

Source: simtk.org

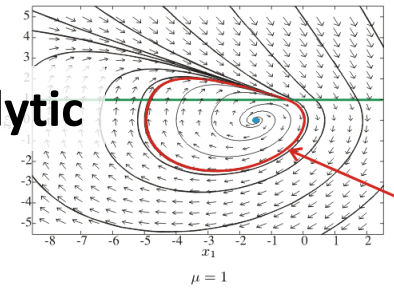
Measured

Source: speedhunter.com



$\mu = -1$

Analytic



$\mu = 1$

$\dot{x}_1 = x_2$

$\dot{x}_2 = -x_2 - \frac{3}{2}|x_2 - \mu| - x_1$

equilibrium: $x_1 = -\frac{3}{2}|\mu|, \quad x_2 = 0$

limit cycle (attracting)

Source: zfm.ethz.ch

Flow Data

Simulation:

- flow: estimate (partial) differential equation systems (e.g. a physical model)
- set of samples (3/4-dims. of data), e.g., given on a curvilinear grid
- most important primitive: tetrahedron and hexahedron (cell)
- could be adaptive grids

Analytic:

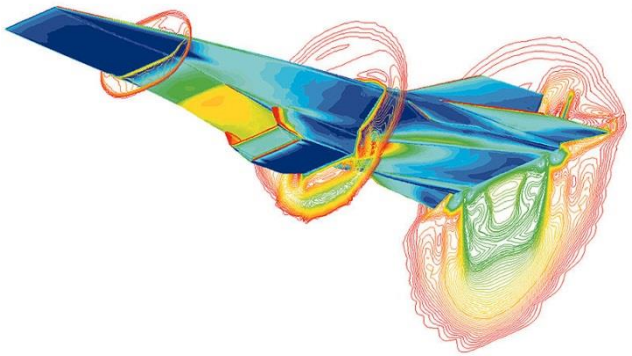
- flow: analytic formula, differential equation systems d**x**/dt (dynamical system)
- evaluated where-ever needed (e.g. making plots of flow in MatLab)

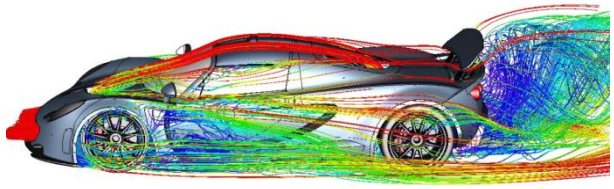
Measurement:

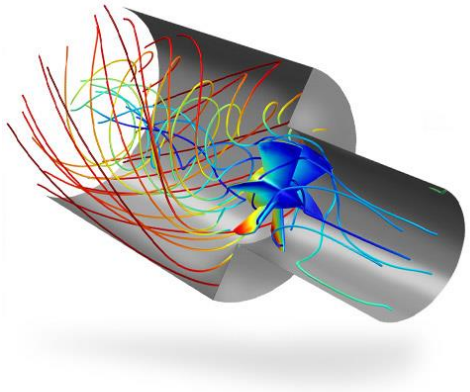
- vectors: taken from instruments, often estimated on a uniform grid
- optical methods + image recognition, e.g.: PIV (particle image velocimetry)

Notes on Computational Fluid Dynamics

- We often visualize Computational Fluid Dynamics (CFD) **simulation** data
- CFD is the discipline of predicting flow behavior, quantitatively
- data is (often) the result of a simulation of flow through or around an object of interest







some characteristics of CFD data:

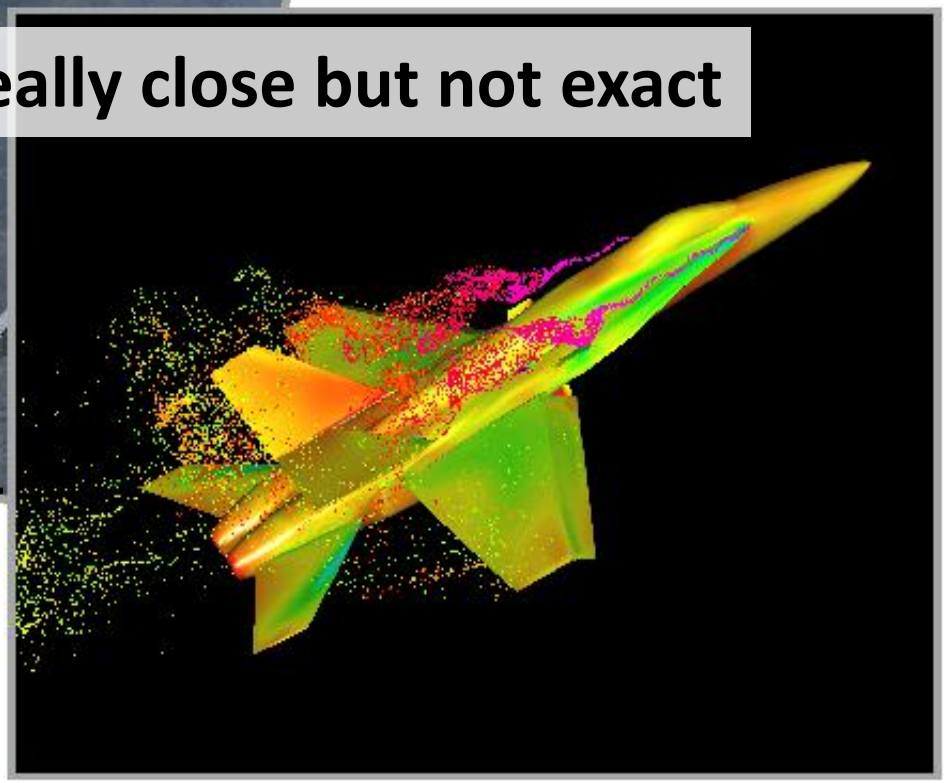
- large, often gigabytes
- Unsteady, i.e. time-dependent
- unstructured, adaptive resolution grids
- Smooth field

Image source: Google images

Comparison with Reality

Experiment

Really close but not exact



Simulation

2D vs. 2.5D Surfaces vs. 3D

2D flow visualization

- R^2 flows
- Planes, or flow layers (2D cross sections through 3D)

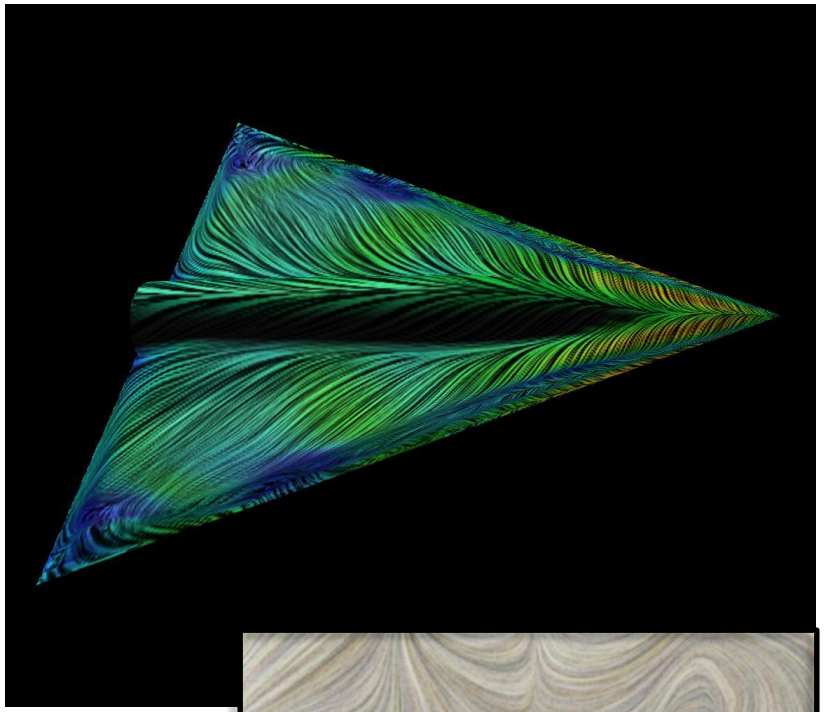
2.5D, i.e. surface flow visualization

- 3D flows around obstacles
- boundary flows on manifold surfaces (locally 2D)

3D flow visualization

- R^3 flows
- simulations, 3D domains

2D/Surfaces/3D – Examples



Surface

2D

3D

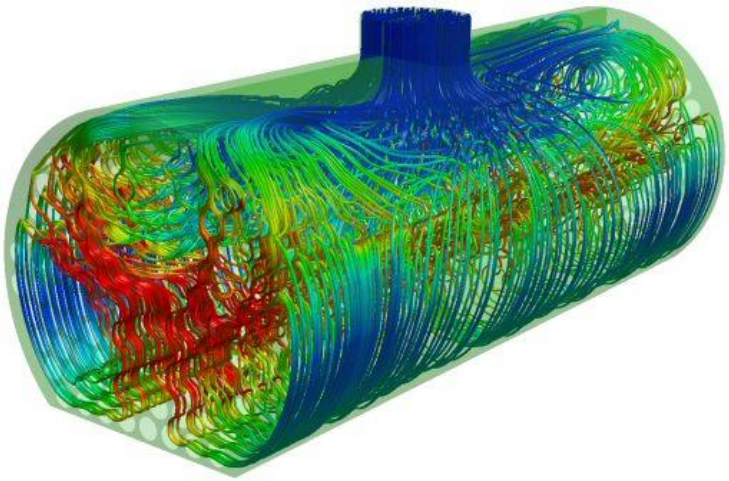


Image source: Alpha solutions

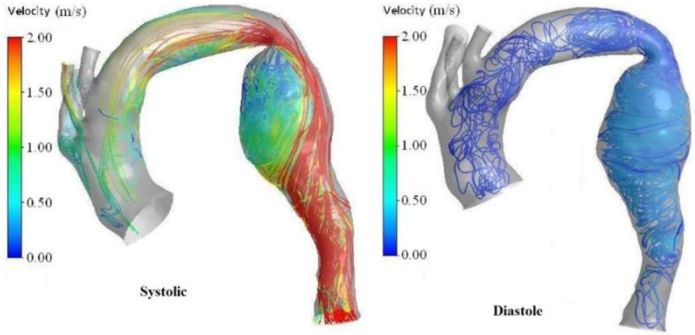
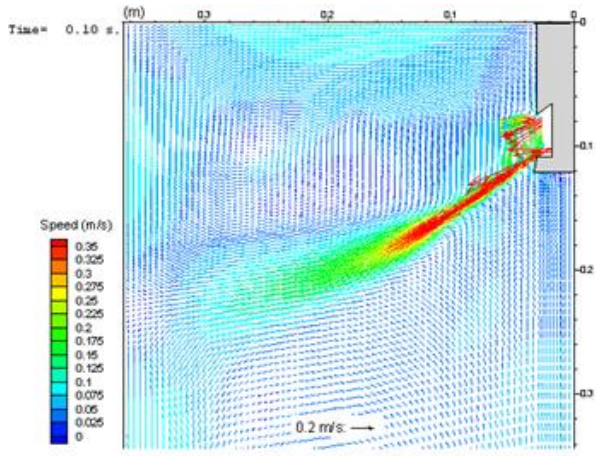


Image source: www.intechopen.com

Steady vs. Time-dependent

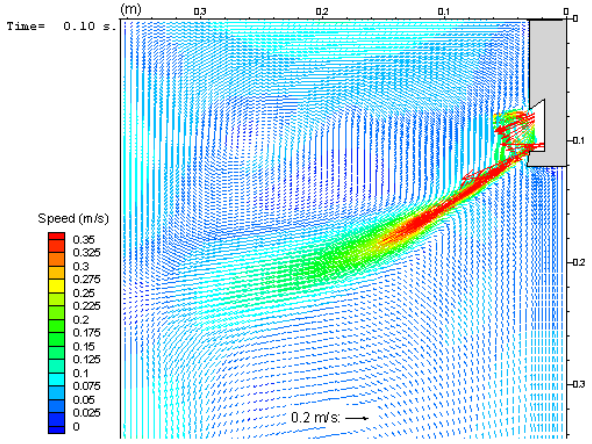
Steady (time-independent) flows:

- flow itself constant over time
- **v**(**x**), e.g., laminar flows
- simpler case for visualization
- well understood behaviors and features

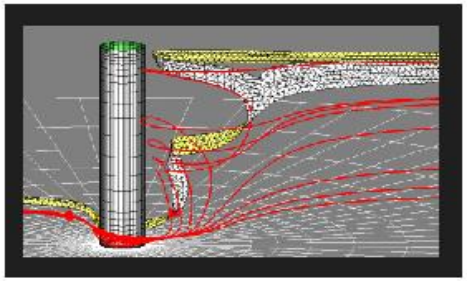


Time-dependent (unsteady) flows:

- flow itself changes over time
- **v**(**x**,***t***), e.g., combustion flow, turbulent flow, wind field
- more complex cases
- *no unified theory to characterize them yet!*

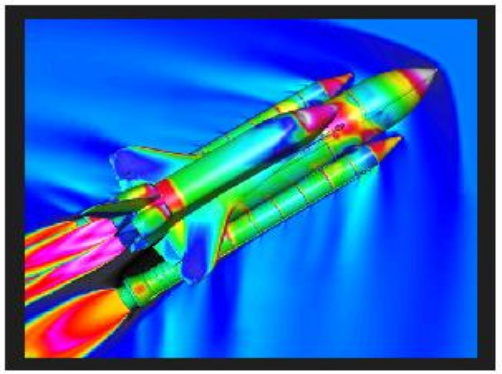


Time-independent (steady) Data



Single Zone
100K Nodes
4 MB

(1985)



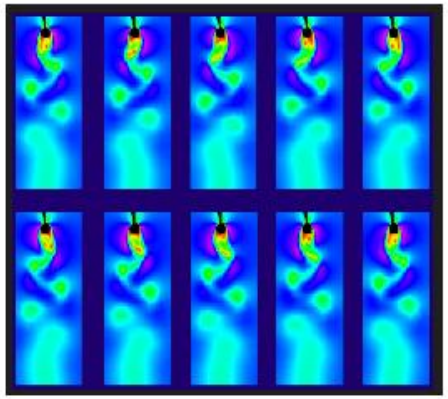
128 Zones
30M Nodes
1080 MB

(1996)

- Dataset sizes over years (old data):

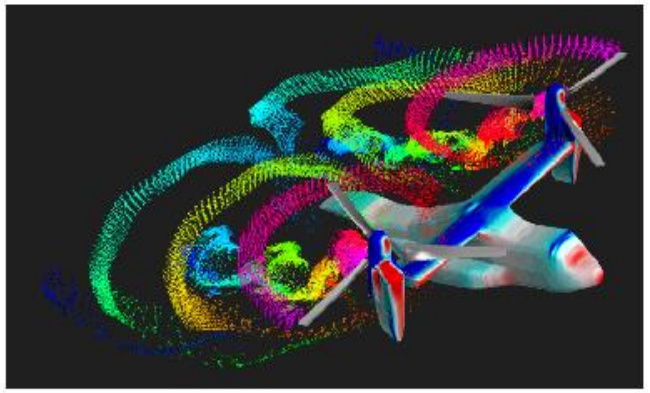
Data set name and year	Number of vertices	Size (MB)
McDonnell Douglas MD–80 '89	230,000	13
McDonnell Douglas F/A–18 '91	900,000	32
Space shuttle launch vehicle '90	1,000,000	34
Space shuttle launch vehicle '93	6,000,000	216
Space shuttle launch vehicle '96	30,000,000	1,080
Advanced subsonic transport '98	60,000,000	2,160
Army UH–60 Blackhawk '99	100,000,000	~4,000

Time-dependent (unsteady) Data



Single Zone
128K Nodes
1 GB

(1990)



25 Zones (9 Moving)
2.8M Nodes
300 GB

(1996)

- Dataset sizes over time:

Data set name and year	# vertices	# time steps	size (MB)
Tapered Cylinder '90	131,000	400	1,050
McDonnell Douglas F/A−18 '92	1,200,000	400	12,800
Descending Delta Wing '93	900,000	1,800	64,800
Bell−Boeing V−22 tiltrotor '93	1,300,000	1,450	140,000
Bell−Boeing V−22 tiltrotor '98	10,000,000	1,450	600,000

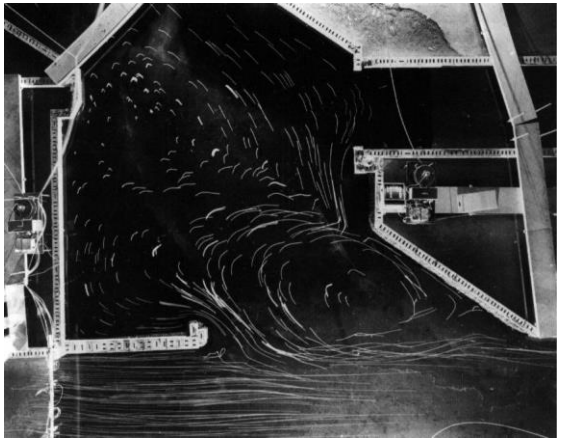
Experimental Flow Visualization

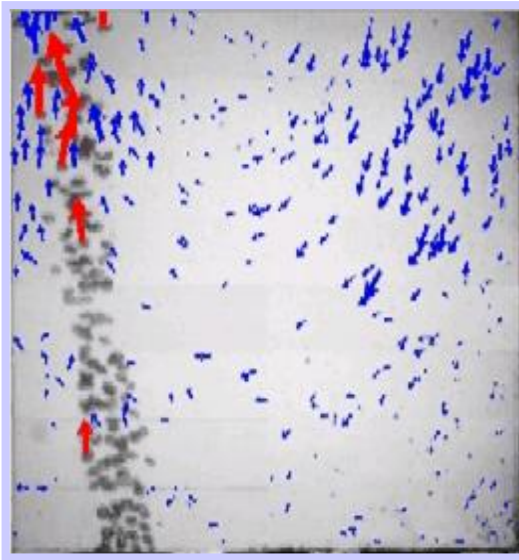
Typically, optical Methods.

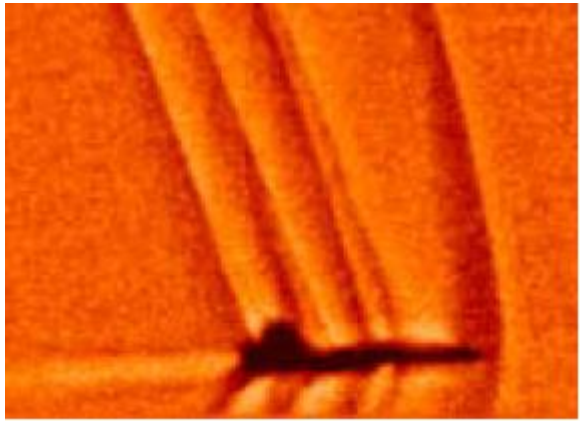
Understanding this experimental methods will help us understand why certain visualization approaches are adopted.

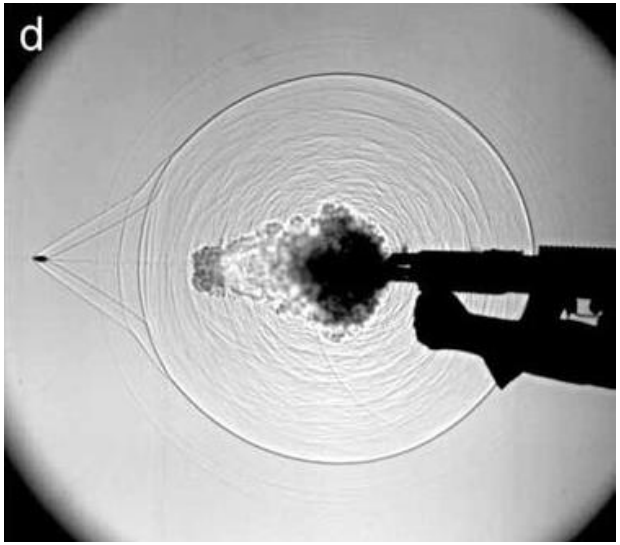
With Smoke or Dye

- Injection of dye, smoke, particles
- Optical methods:
  - transparent object with complex distribution of light refraction index
- Streaks, shadows

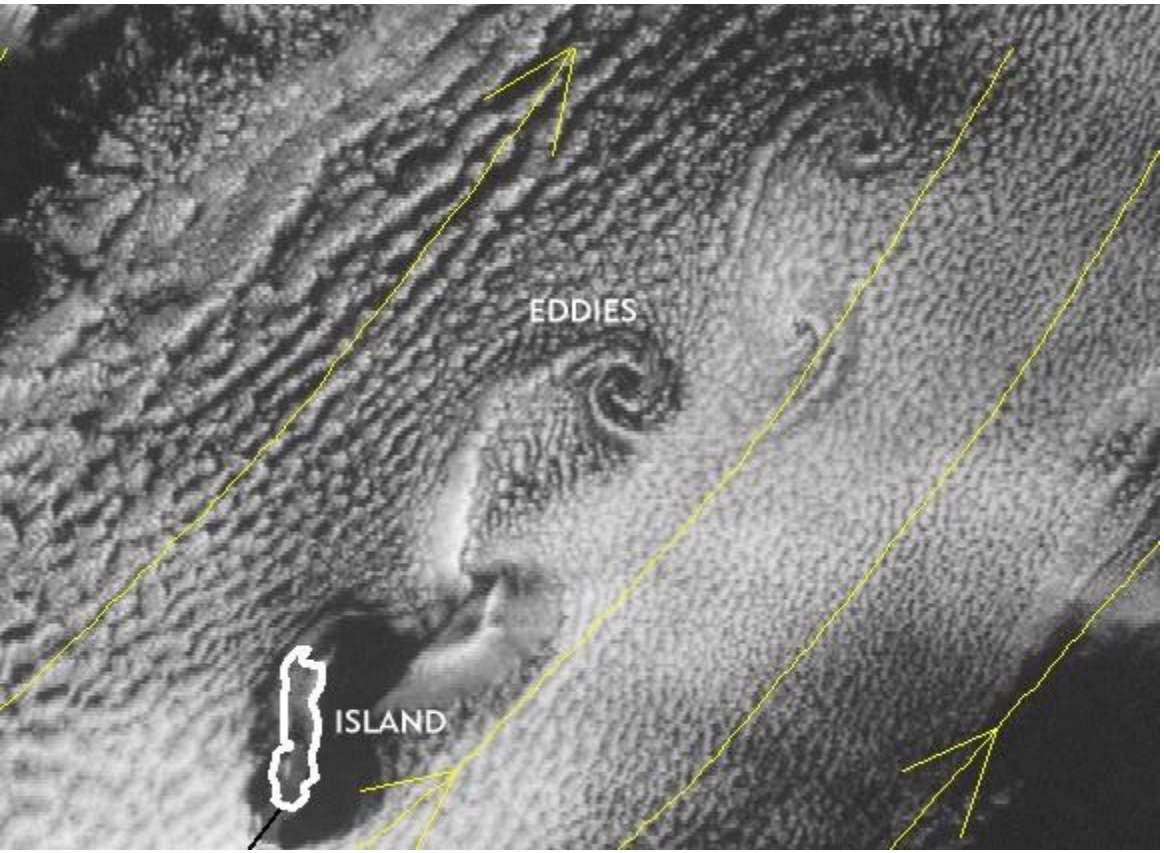








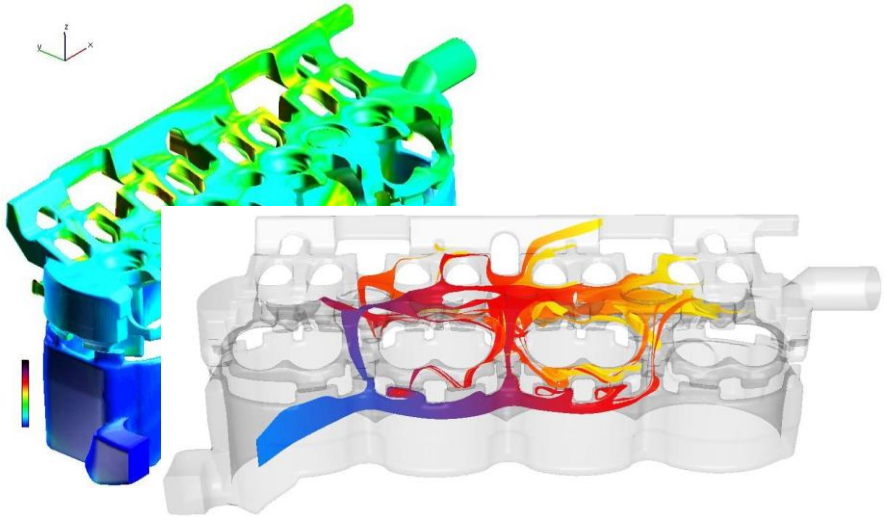
Large Scale Dying

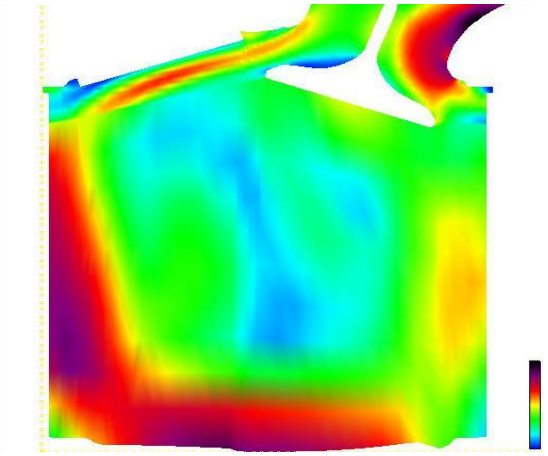


Source: weathergraphics.com

Source: ishtarsgate.com

Direct Methods

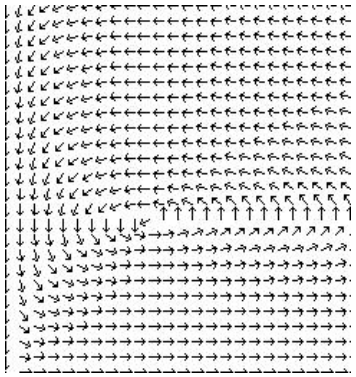


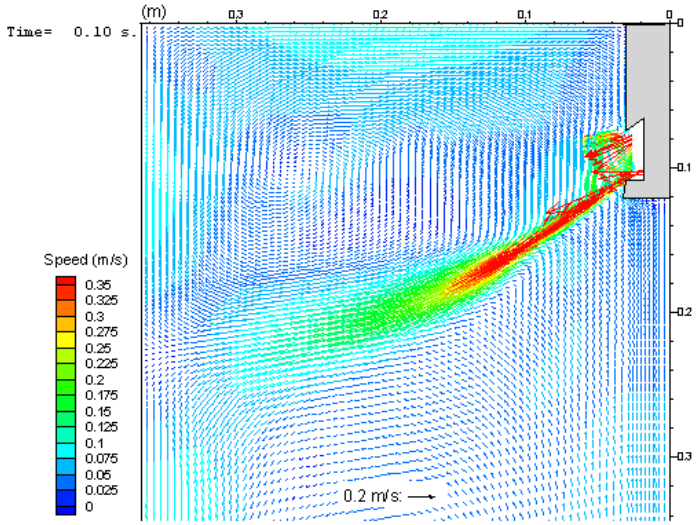


Direct FlowVis with Arrows

Properties:

- direct FlowVis
- <u>frequently used!</u>
- *normalized* arrows vs. velocity coding
- 2D: quite useful,
  3D: often problematic
- often difficult to understand in complex cases, mentally integrate to reconstruct the flow





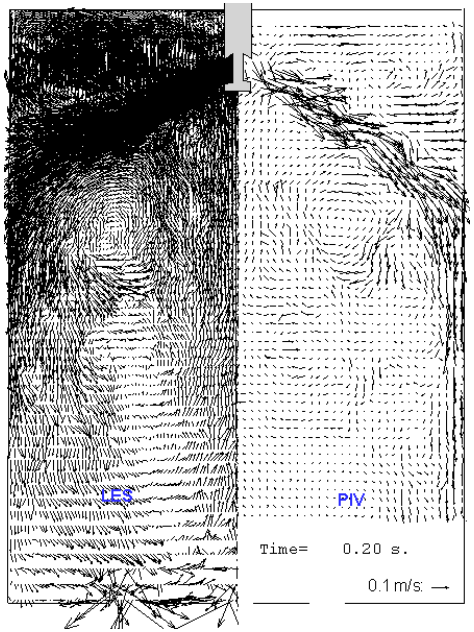
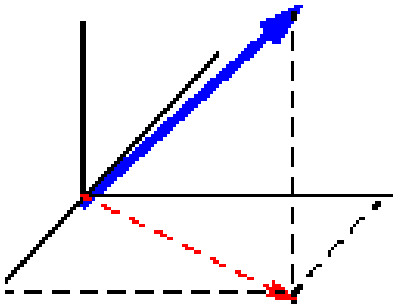


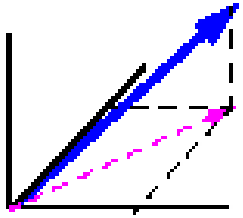
Image source: tms.org

Issues of Arrows in 3D

Common problems:

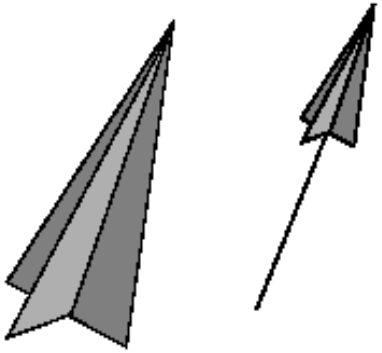
- Ambiguity
- Perspective shortening
- 1D objects generally difficult to grasp in 3D





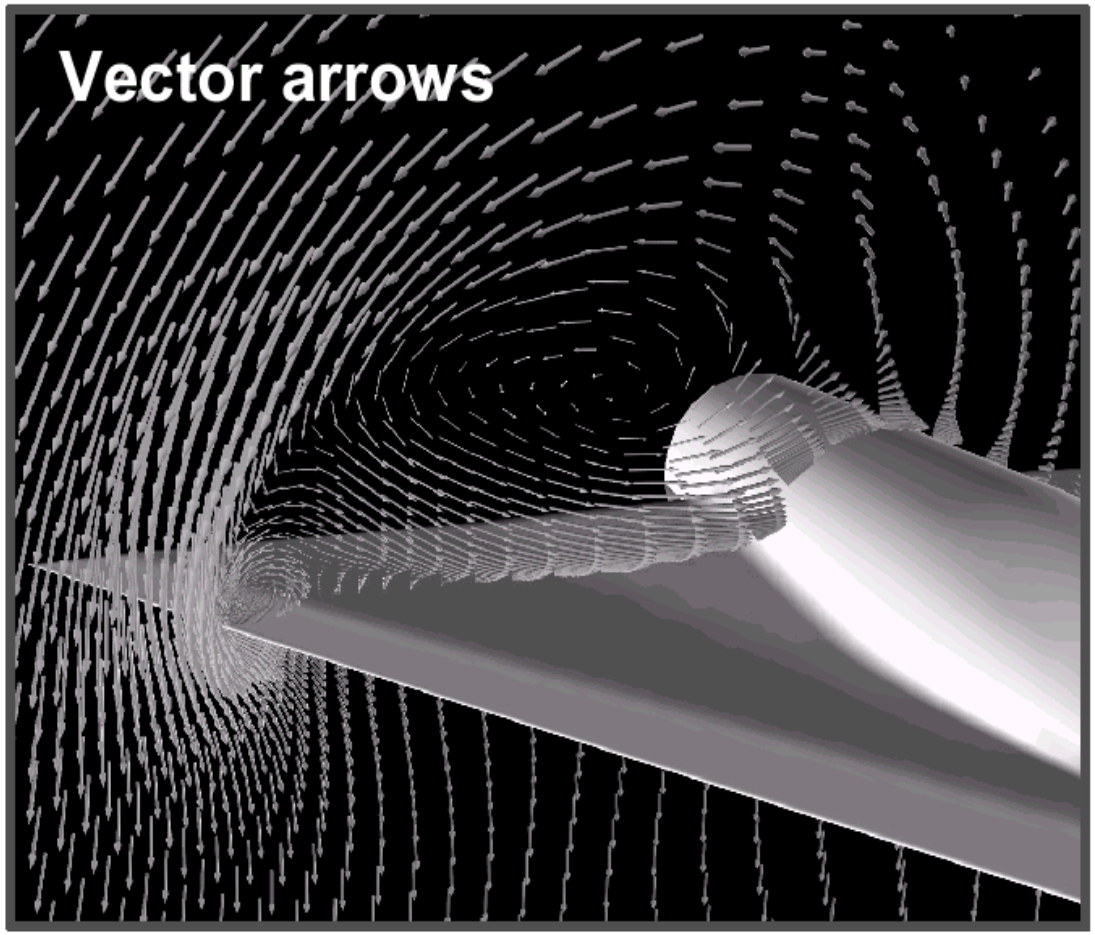
Remedy:

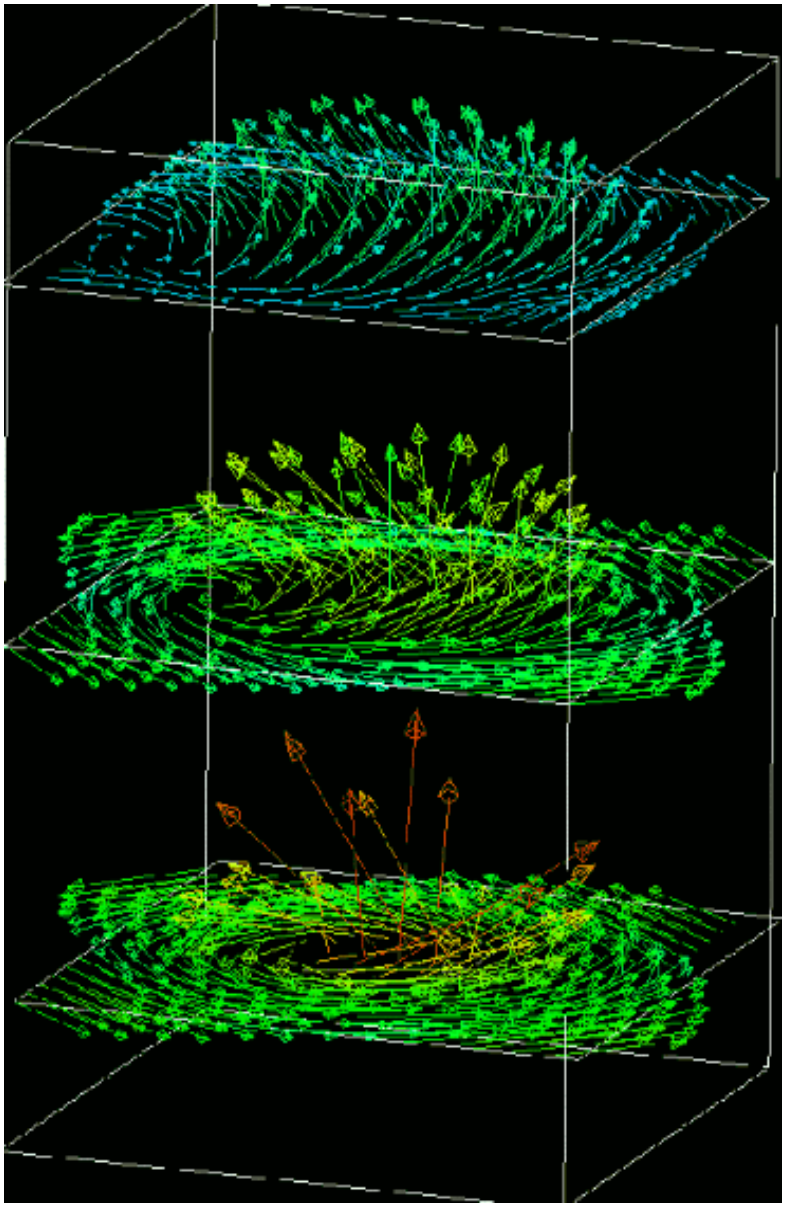
- **3D**-Arrows
  (are of some help)



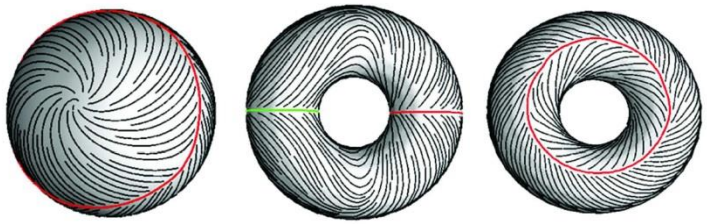
Arrows in 3D – Examples

Compromise:
arrows only in layers

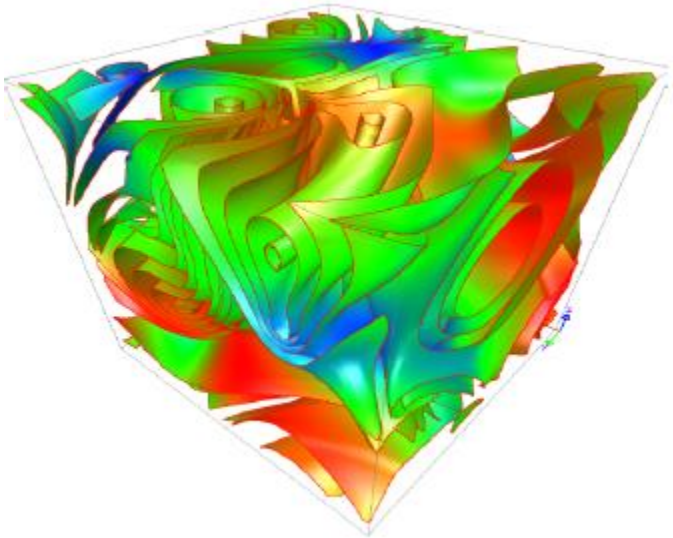


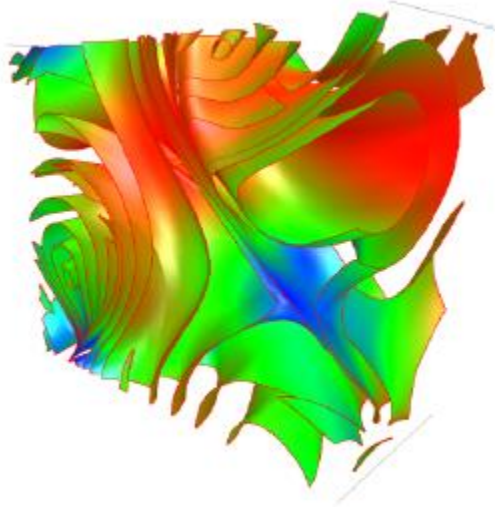


aching/csIVU/-vis/



Geometric-based Methods: Integral curves and surfaces

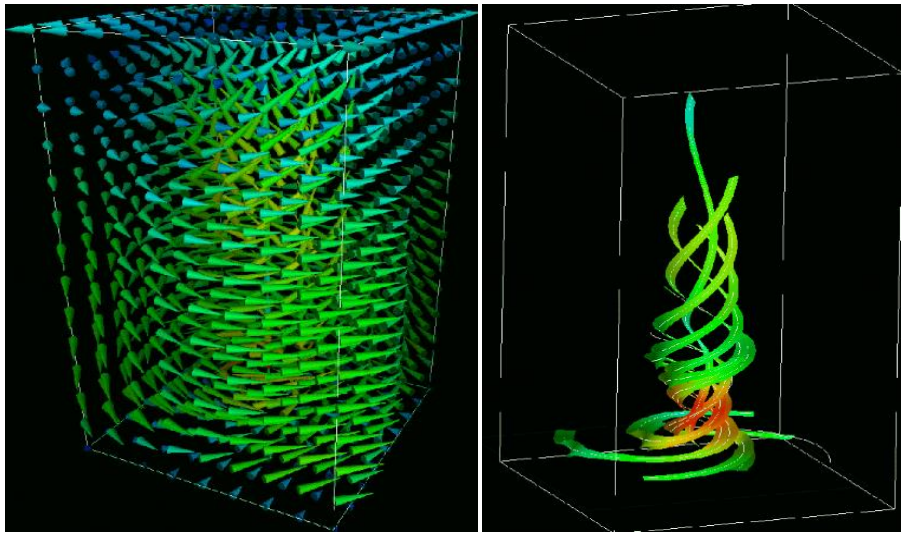




Direct vs. Geometric FlowVis

Direct flow visualization:

- overview of current state of flow
- visualization with vectors popular
- arrows, icons, glyph techniques



Geometric flow visualization:

- use of intermediate objects,
  e.g., after vector field integration over time
- visualization of development over time
- streamlines, stream surfaces
- *analogous to indirect (vs. direct) volume visualization*

Streamlines – Theory

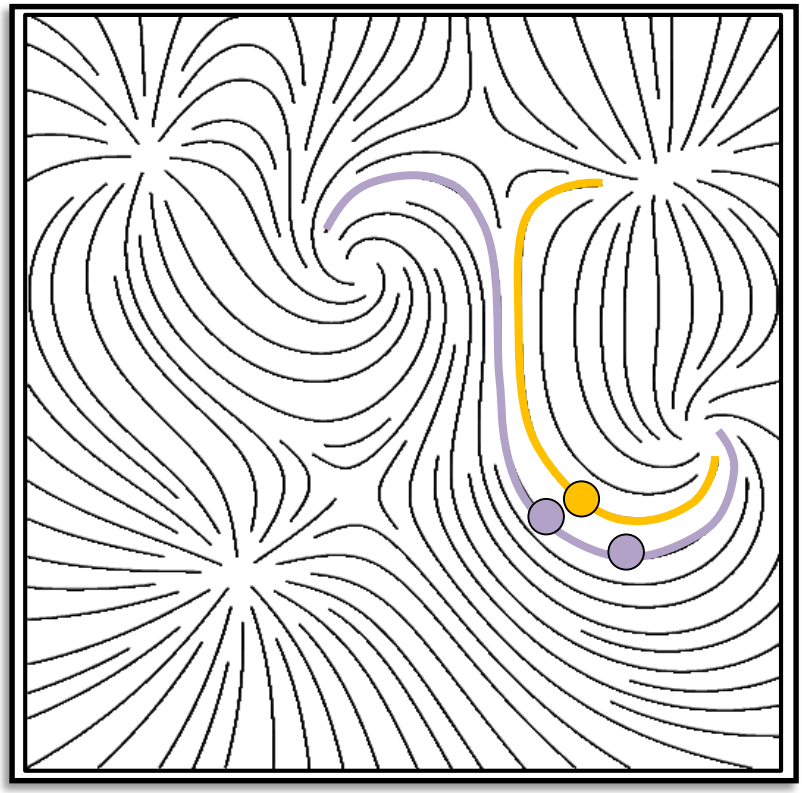
Correlations:

- flow data $\mathbf{v}$: derivative information
  - $d\mathbf{x}/dt = \mathbf{v}(\mathbf{x})$;
    spatial points $\mathbf{x} \in \mathrm{R}^n$, Time $t \in \mathrm{R}$, flow vectors $\mathbf{v} \in \mathrm{R}^n$

- streamline $\mathbf{s}$: integration over time, also called trajectory, solution, curve
  - $\mathbf{s}(t) = \mathbf{s}_0 + \int_{0 \le u \le t} \mathbf{v}(\mathbf{s}(u))\, du$;
    seed point $\mathbf{s}_0$, integration variable u

- Property:
  - uniqueness

- difficulty: result $\mathbf{s}$ also in the integral $\Rightarrow$ analytical solution usually impossible.



Streamlines – Practice

Basic approach:

- theory: $\mathbf{s}(t) = \mathbf{s}_0 + \int_{0 \le u \le t} \mathbf{v}(\mathbf{s}(u))\, \mathrm{d}u$
- practice: numerical integration
- idea:
  (very) locally, the solution is (approx.) linear
- Euler integration:
  follow the current flow vector $\mathbf{v}(\mathbf{s}_i)$ from the current streamline point $\mathbf{s}_i$
  for a very small time ($\mathrm{d}t$) and therefore distance

Euler integration: $\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{v}(\mathbf{s}_i) \cdot dt$,
integration of small steps ($\mathrm{d}t$ very small)

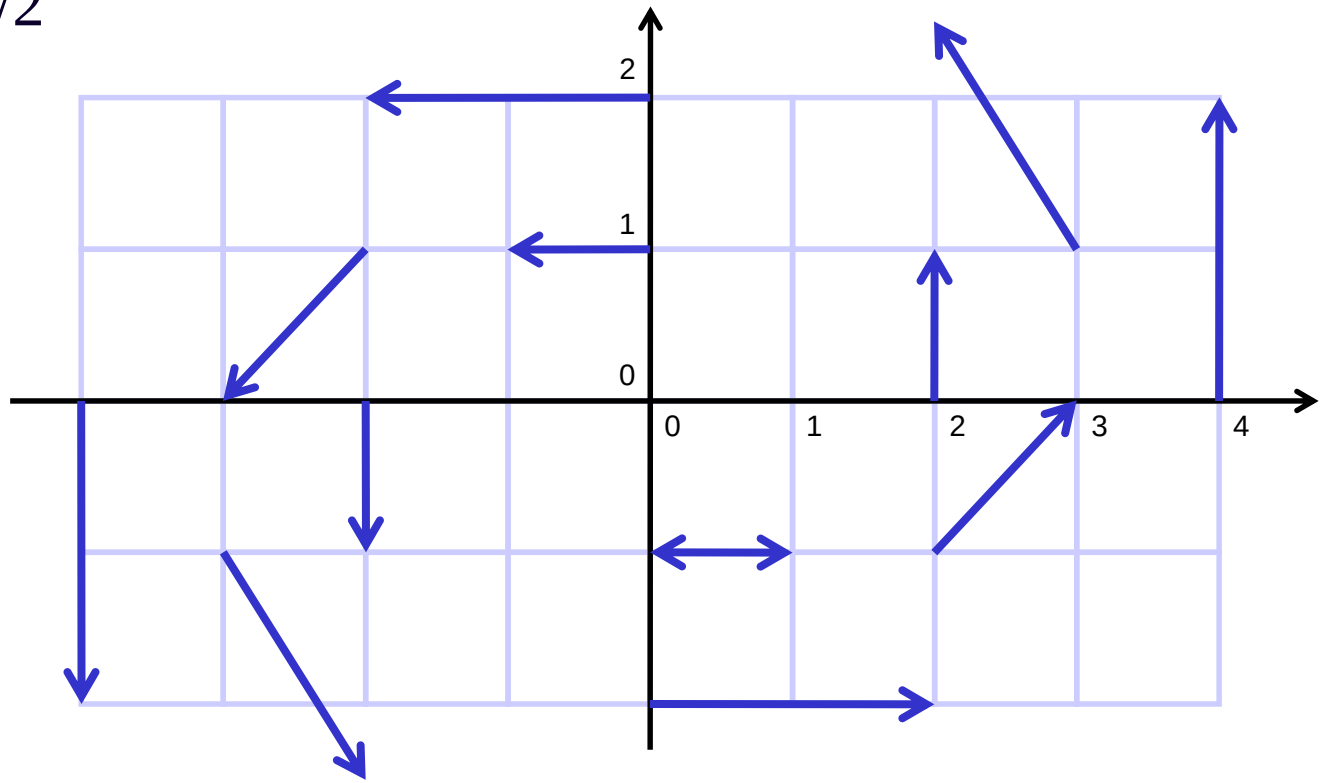
Euler Integration – Example

2D model data:

$v_x = dx/dt = -y$

$v_y = dy/dt = x/2$

Sample arrows:



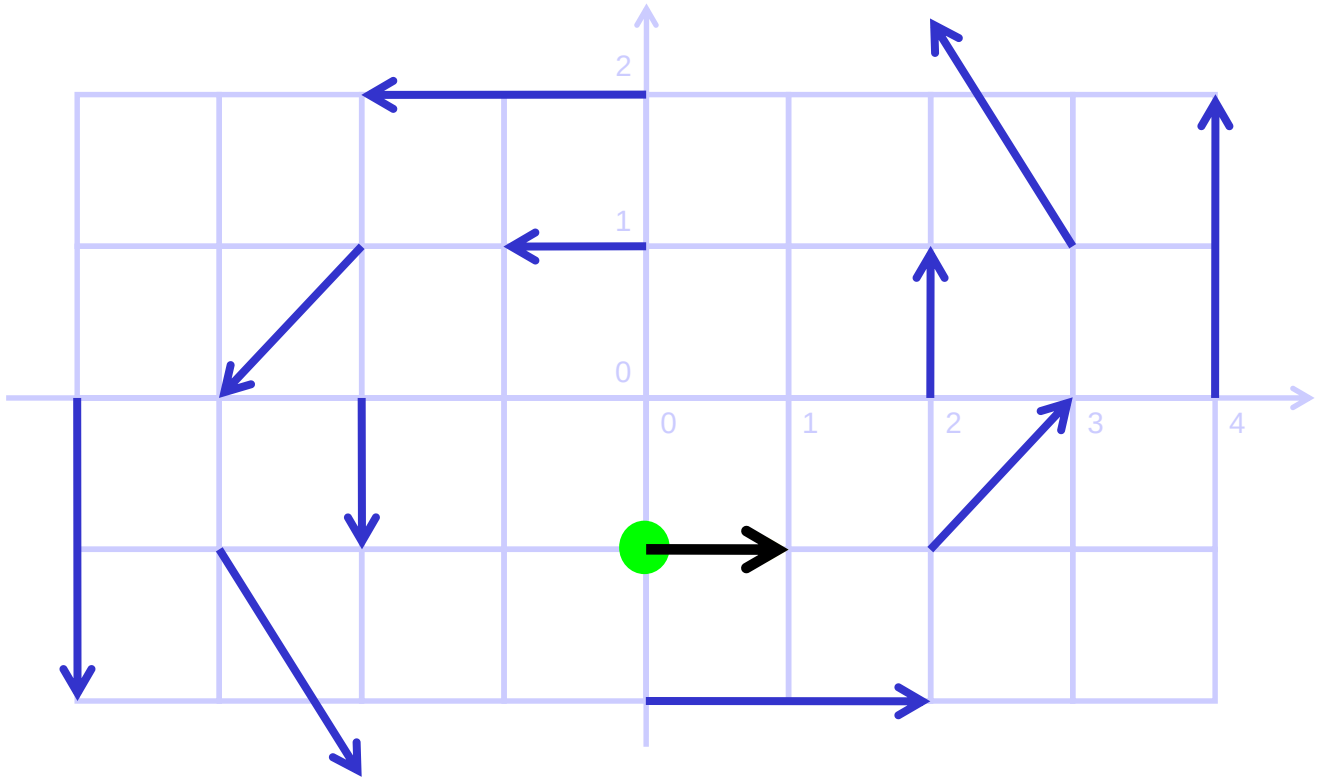
True solution: ellipses.

Euler Integration – Example

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

$v_x = dx/dt = -y$

$v_y = dy/dt = x/2$

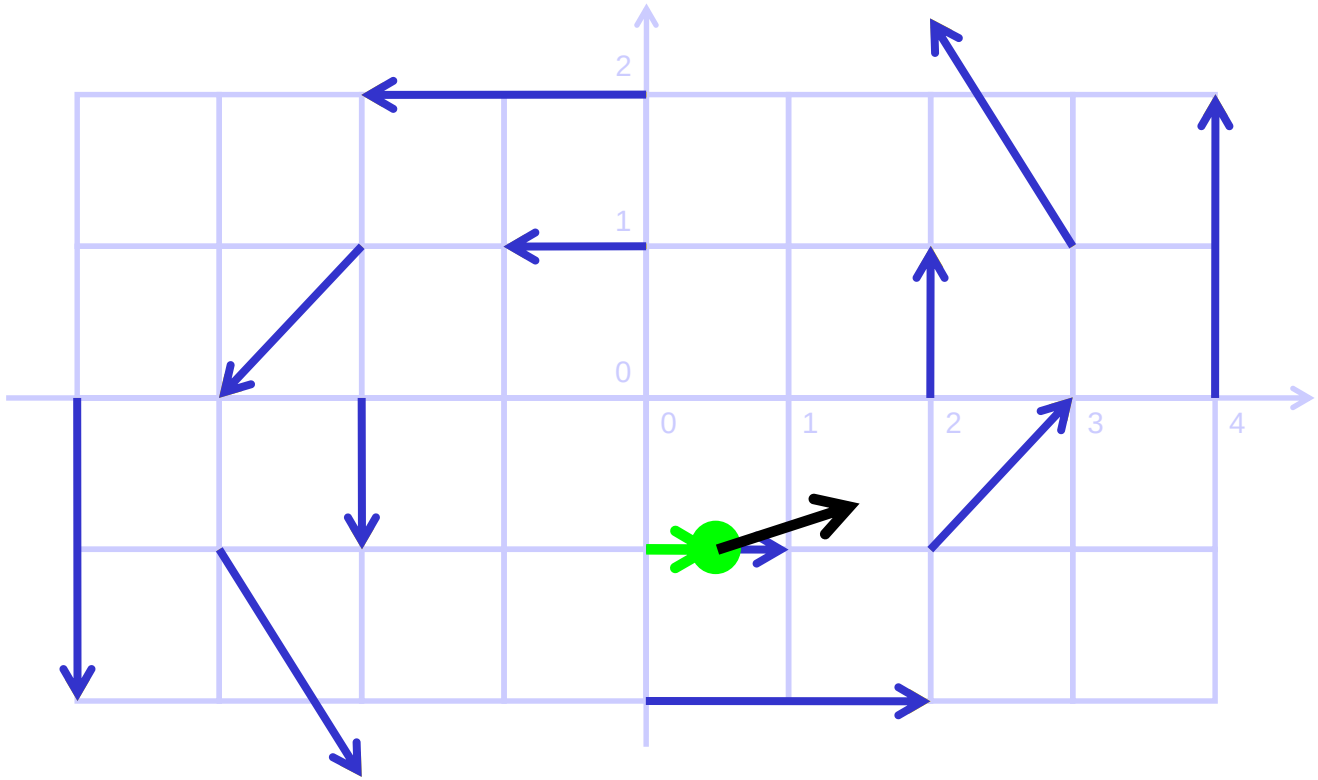


Euler Integration – Example

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

$v_x = dx/dt = -y$

$v_y = dy/dt = x/2$

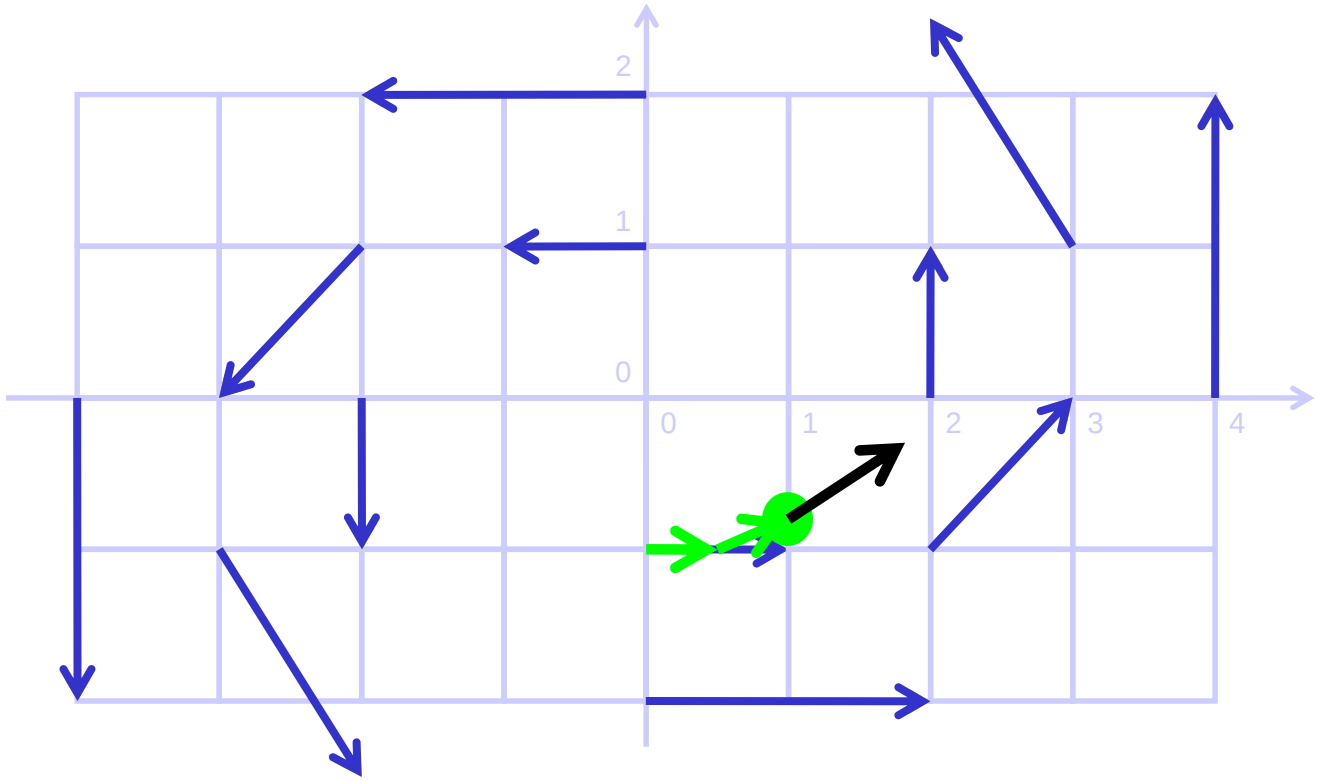


Euler Integration – Example

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

$v_x = dx/dt = -y$

$v_y = dy/dt = x/2$



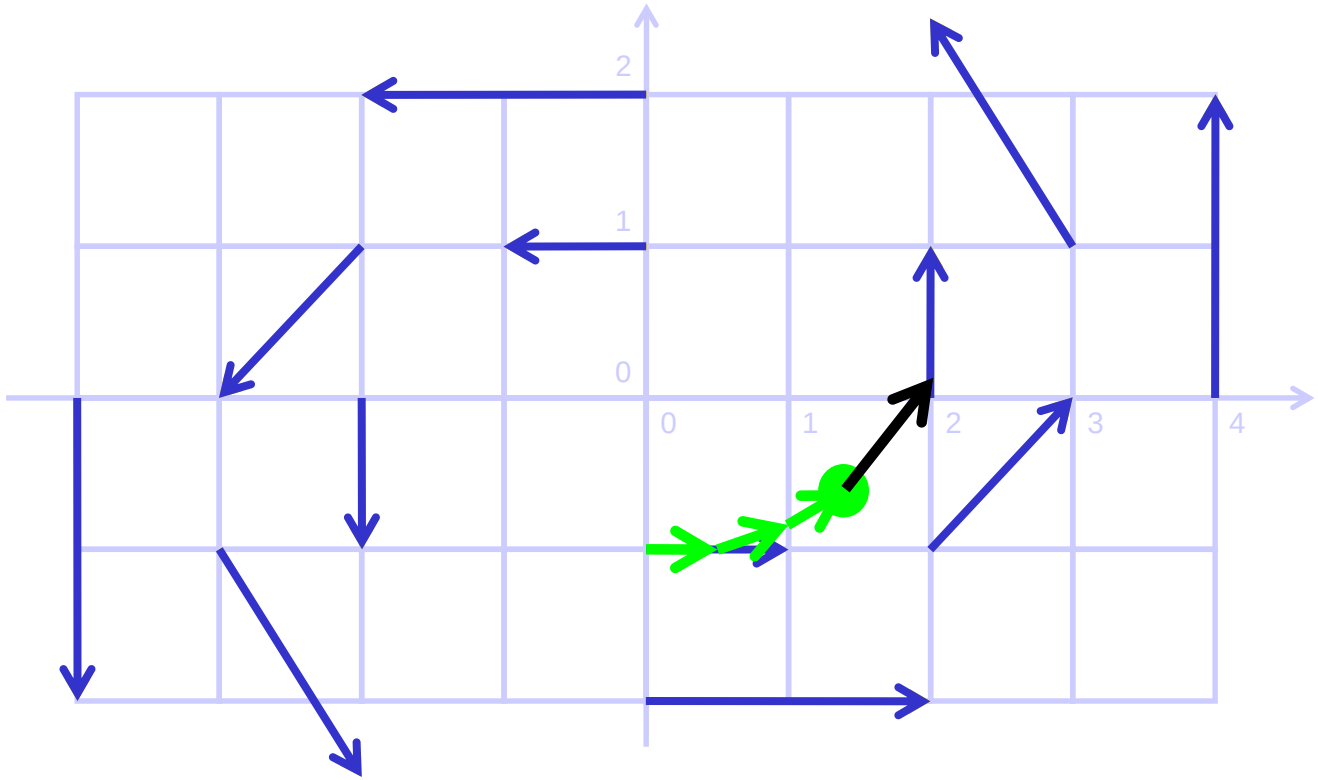
Euler Integration – Example

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

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

$v_x = dx/dt = -y$

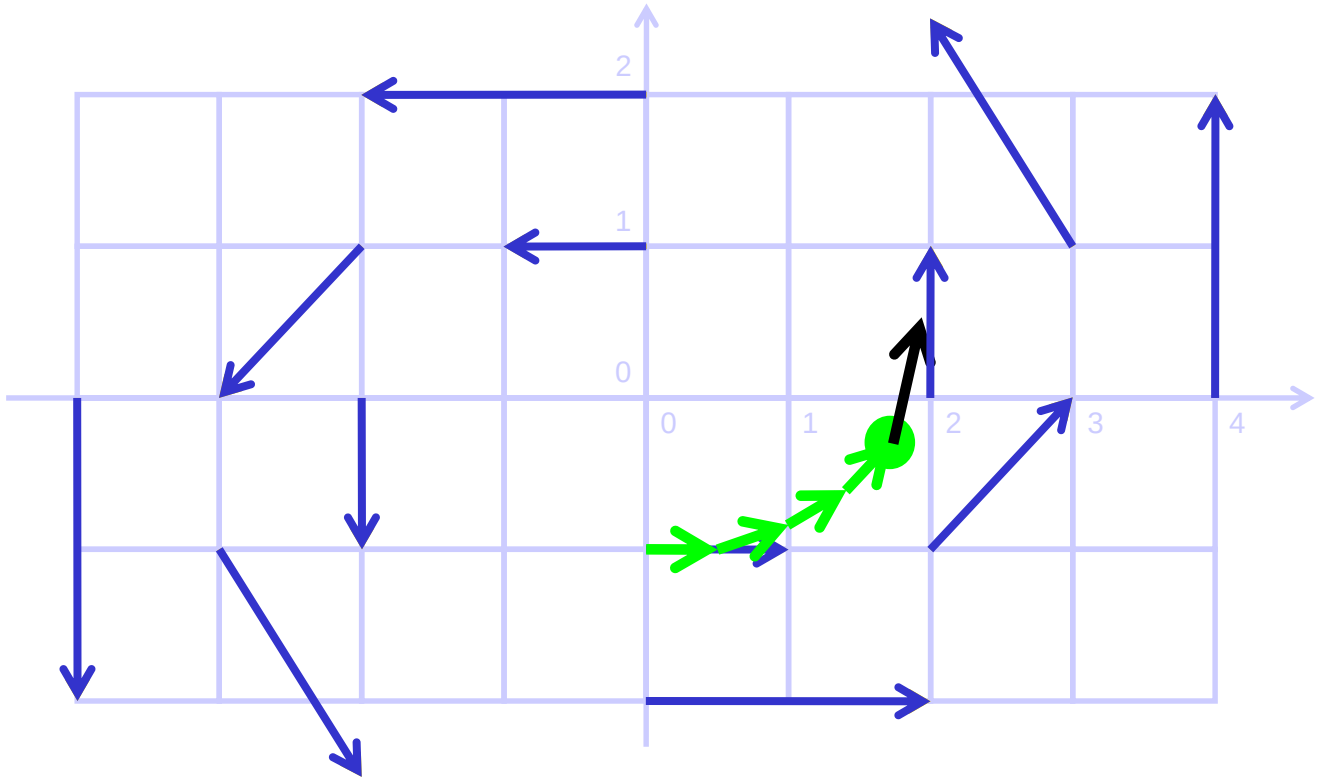
$v_y = dy/dt = x/2$



Euler Integration – Example

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

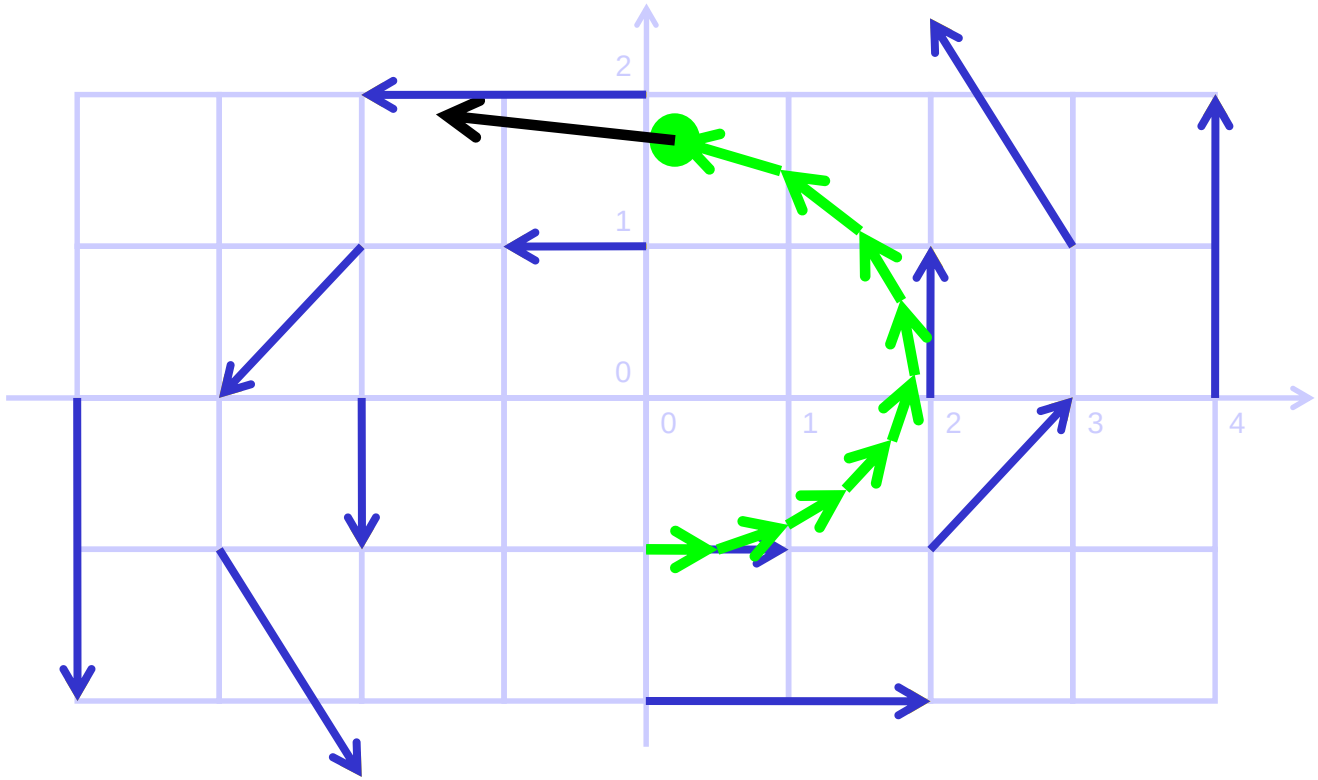
$\mathbf{v}(\mathbf{s}_4) = (17/64 | 7/8)^T \approx (0.27 | 0.88)^T;$



Euler Integration – Example

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

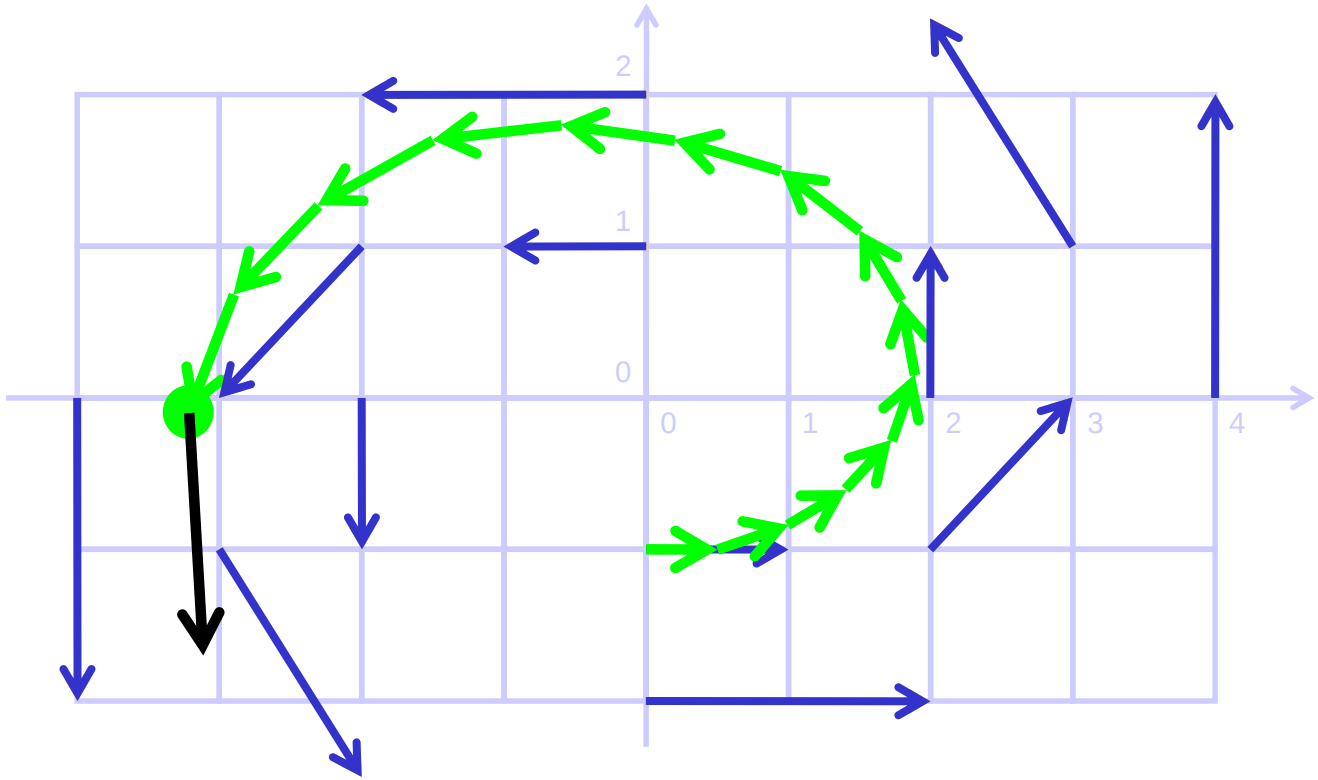
$\mathbf{v}(\mathbf{s}_9) \approx (-1.69|0.10)^T$;



Euler Integration – Example

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

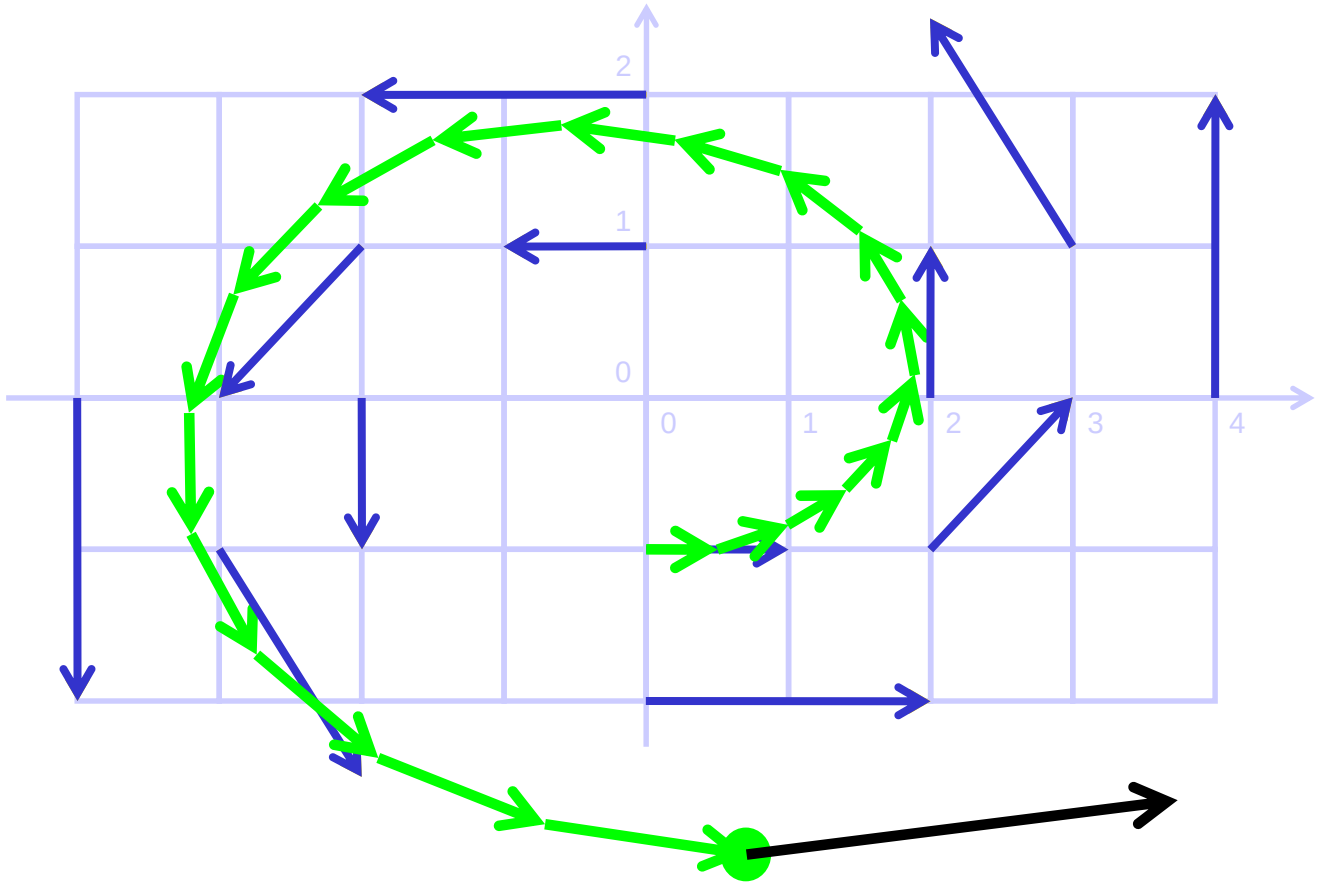
$\mathbf{v}(\mathbf{s}_{14}) \approx (0.10 | -1.61)^T$;



Euler Integration – Example

- $\mathbf{s}_{19} \approx (0.75|\text{-}3.02)^T$; $\mathbf{v}(\mathbf{s}_{19}) \approx (3.02|0.37)^T$;

clearly: large integration error, d*t* too large, 19 steps

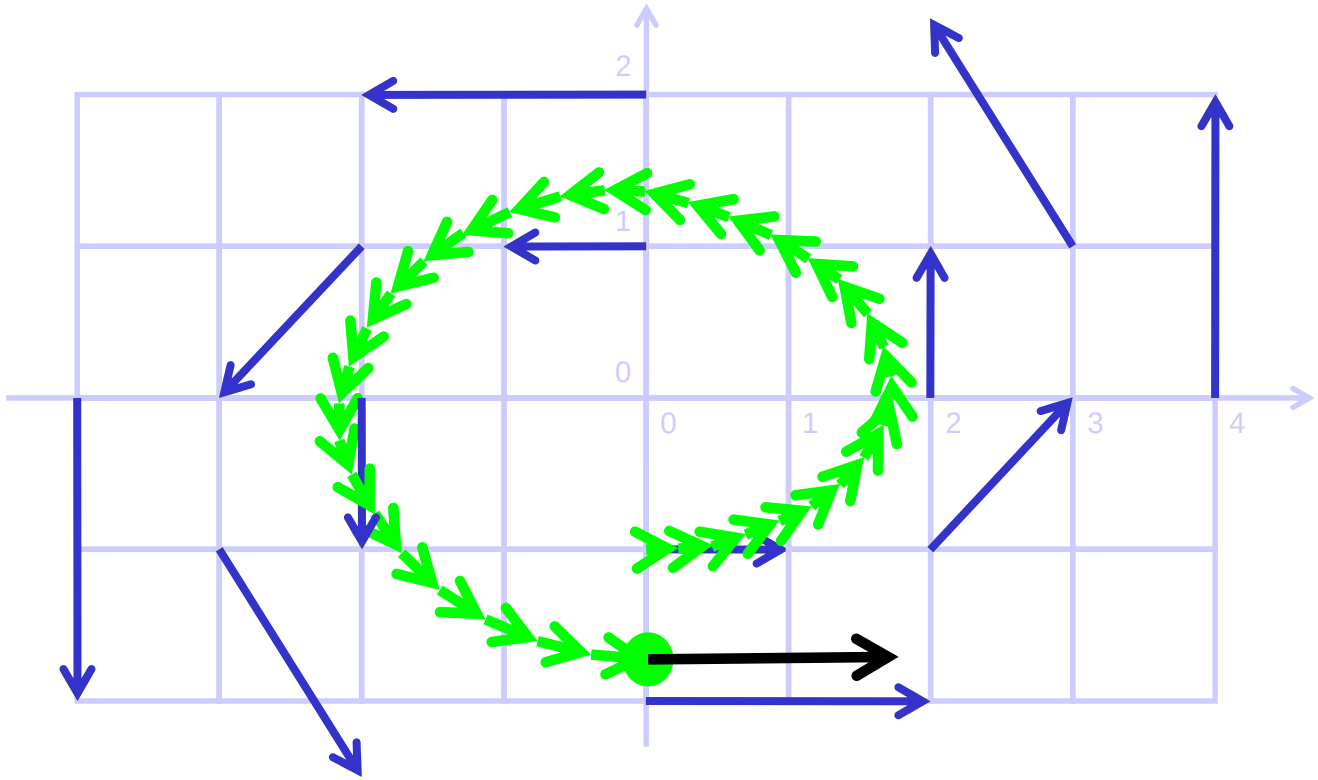


Euler Integration – Example

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

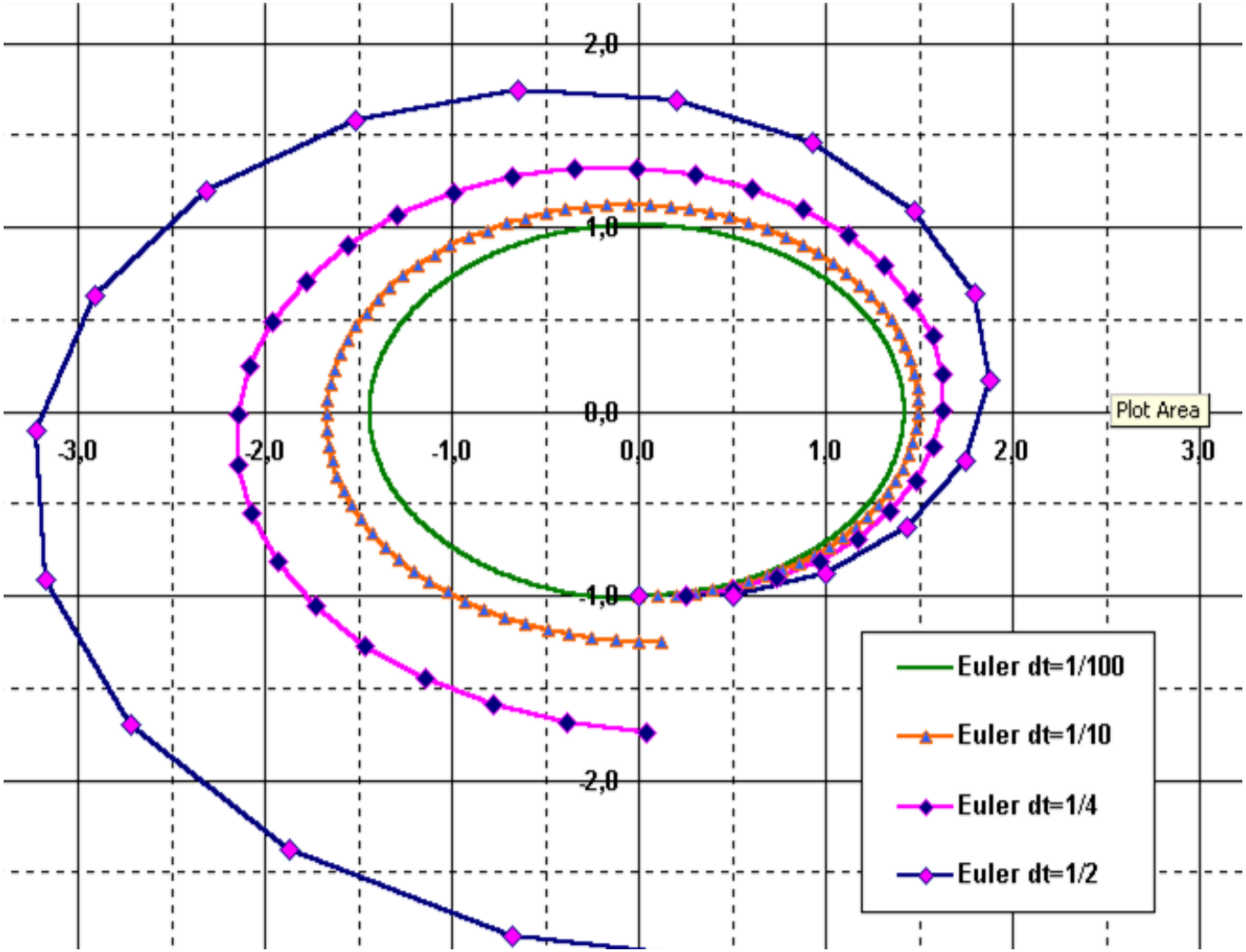
$\mathbf{s}_{36} \approx (0.04 | -1.74)^T$; $\mathbf{v}(\mathbf{s}_{36}) \approx (1.74 | 0.02)^T$;

- 36 steps



Comparison Euler, Step Sizes

Euler
quality is
proportional
to dt

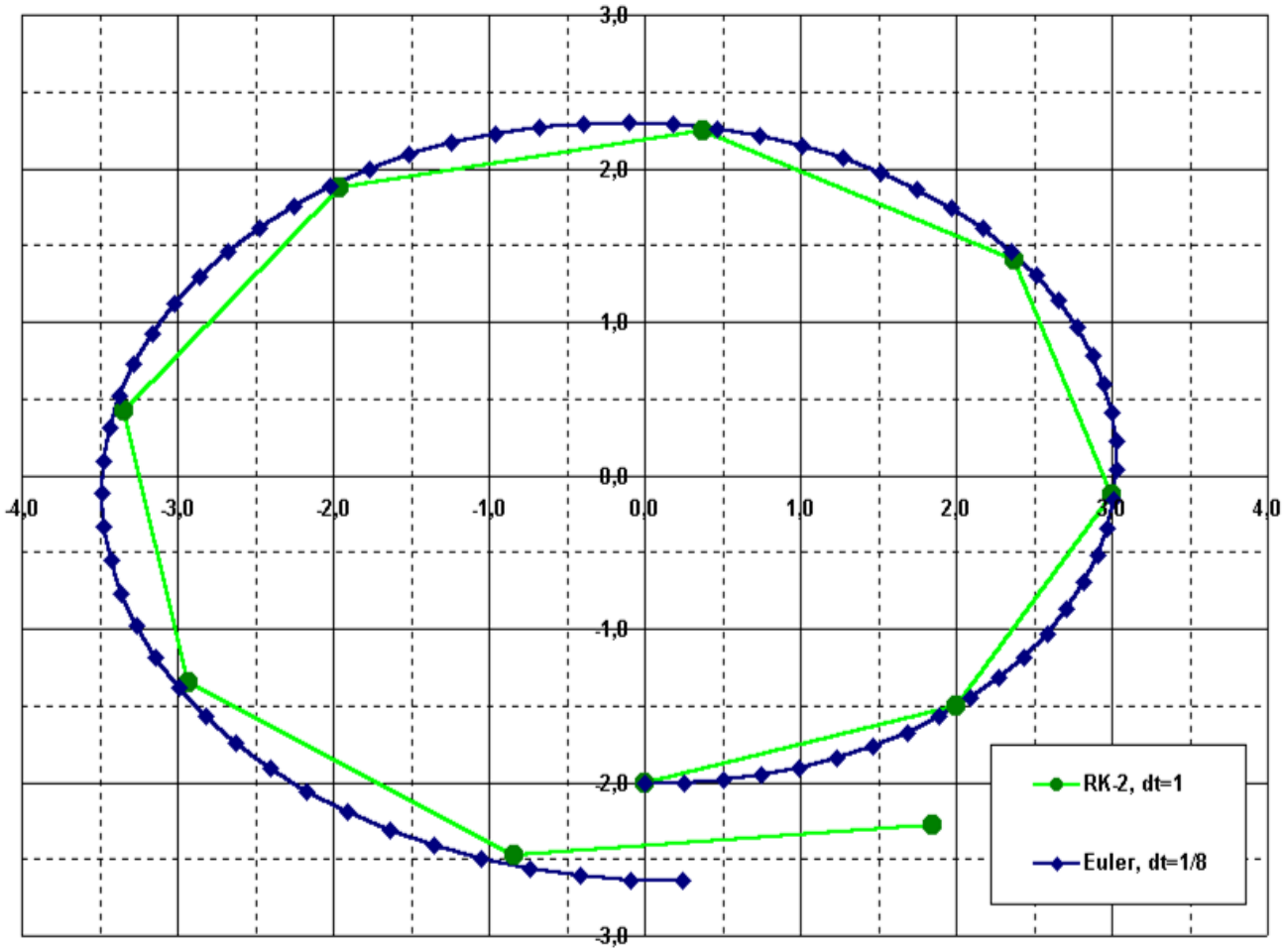


Euler Example – Error Table

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 d*t* = 1 (9 steps)
better
than Euler
with d*t* = 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:
  $\mathbf{s}_{i+1} = \mathbf{s}_i + (\mathbf{a} + 2\,\mathbf{b} + 2\,\mathbf{c} + \mathbf{d})/6$
- vectors:

  $\mathbf{a} = dt\ \mathbf{v}(\mathbf{s}_i)$ … original vector

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

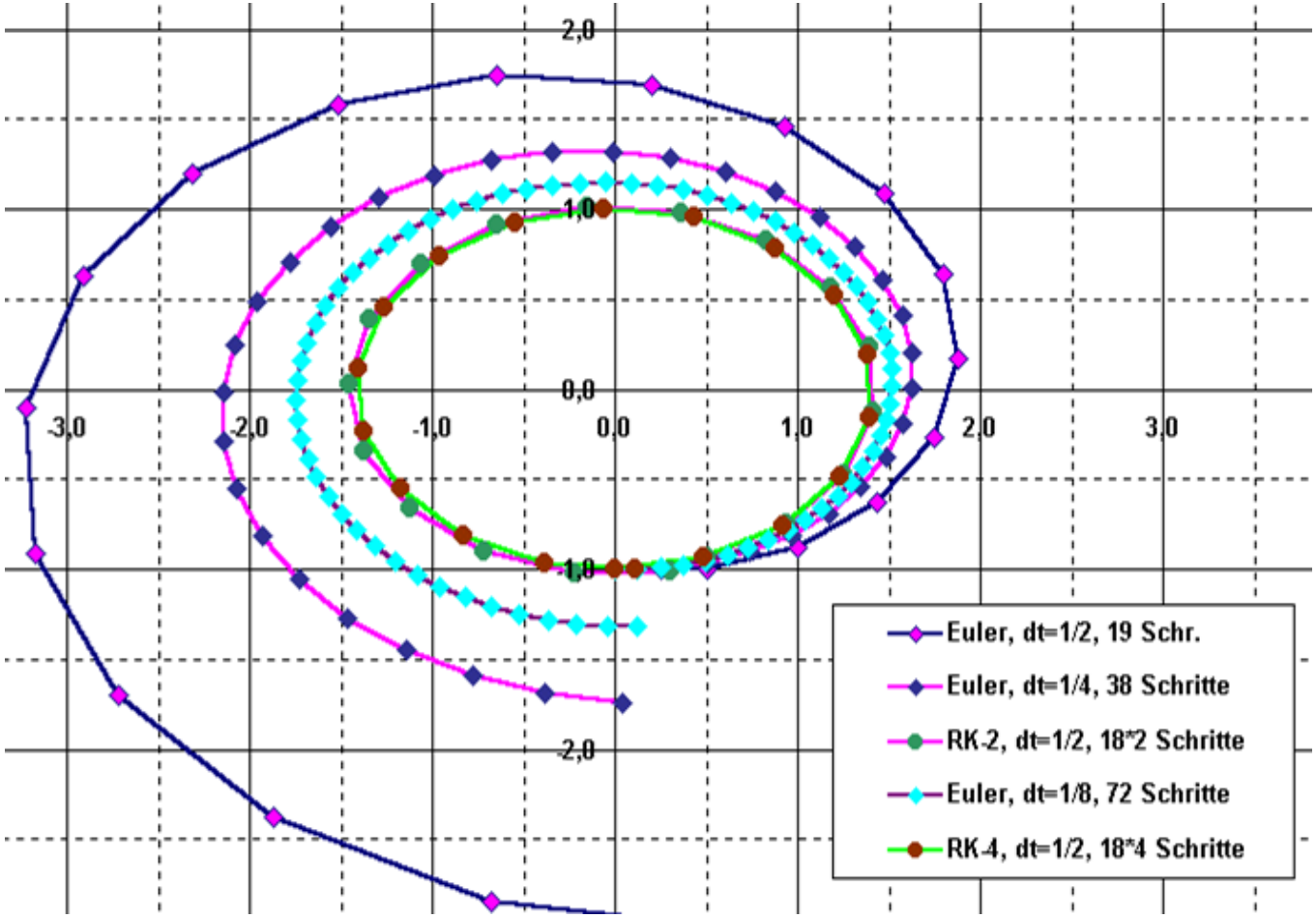
  $\mathbf{c} = dt\ \mathbf{v}(\mathbf{s}_i + \mathbf{b}/2)$ … use RK-2 …

  $\mathbf{d} = dt\ \mathbf{v}(\mathbf{s}_i + \mathbf{c})$ … 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 $\mathrm{d}t$
- RK: more accurate in higher orders
- furthermore: adaptive methods, implicit methods, etc.

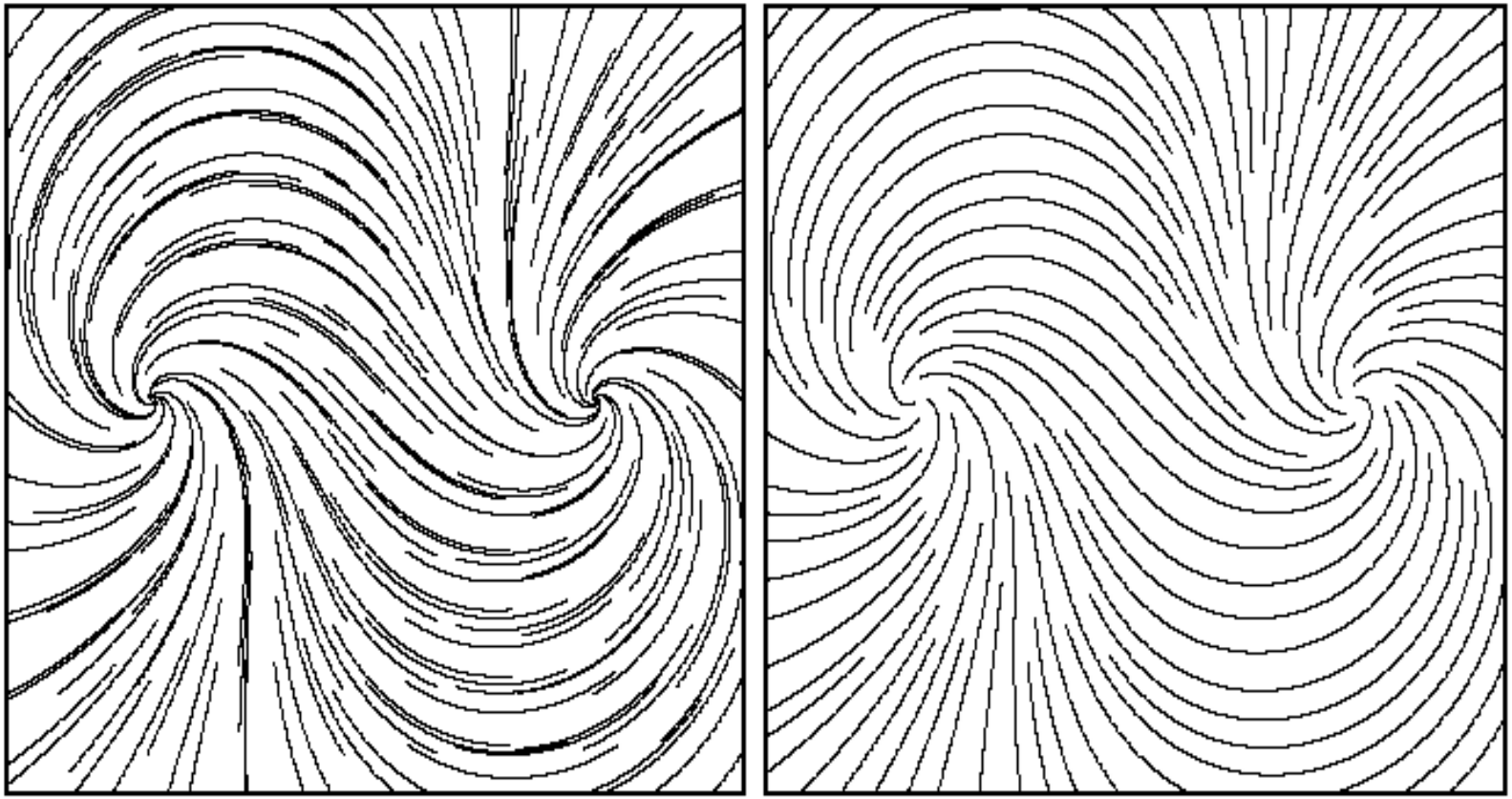
Streamline Placement

in 2D

Problem: Choice of Seed Points

Streamline placement:

- If regular grid used: very irregular result



Overview of Algorithm

Idea: streamlines should not lie too close to one another

Approach:

- choose a seed point with distance d_{sep} from an already existing streamline
- forward- and backward-integration until distance d_{test} is reached (or …).
- two parameters:
- d_{sep} … start distance
- d_{test}… minimum distance

Algorithm – Pseudo-Code

- Compute initial streamline, put it into a queue
- current streamline = initial streamline
- WHILE not finished DO:

  TRY: get new seed point which is d_{sep} away from current streamline

  IF successful THEN

    compute new streamline AND put to queue

  ELSE IF no more streamline in queue THEN

    exit loop

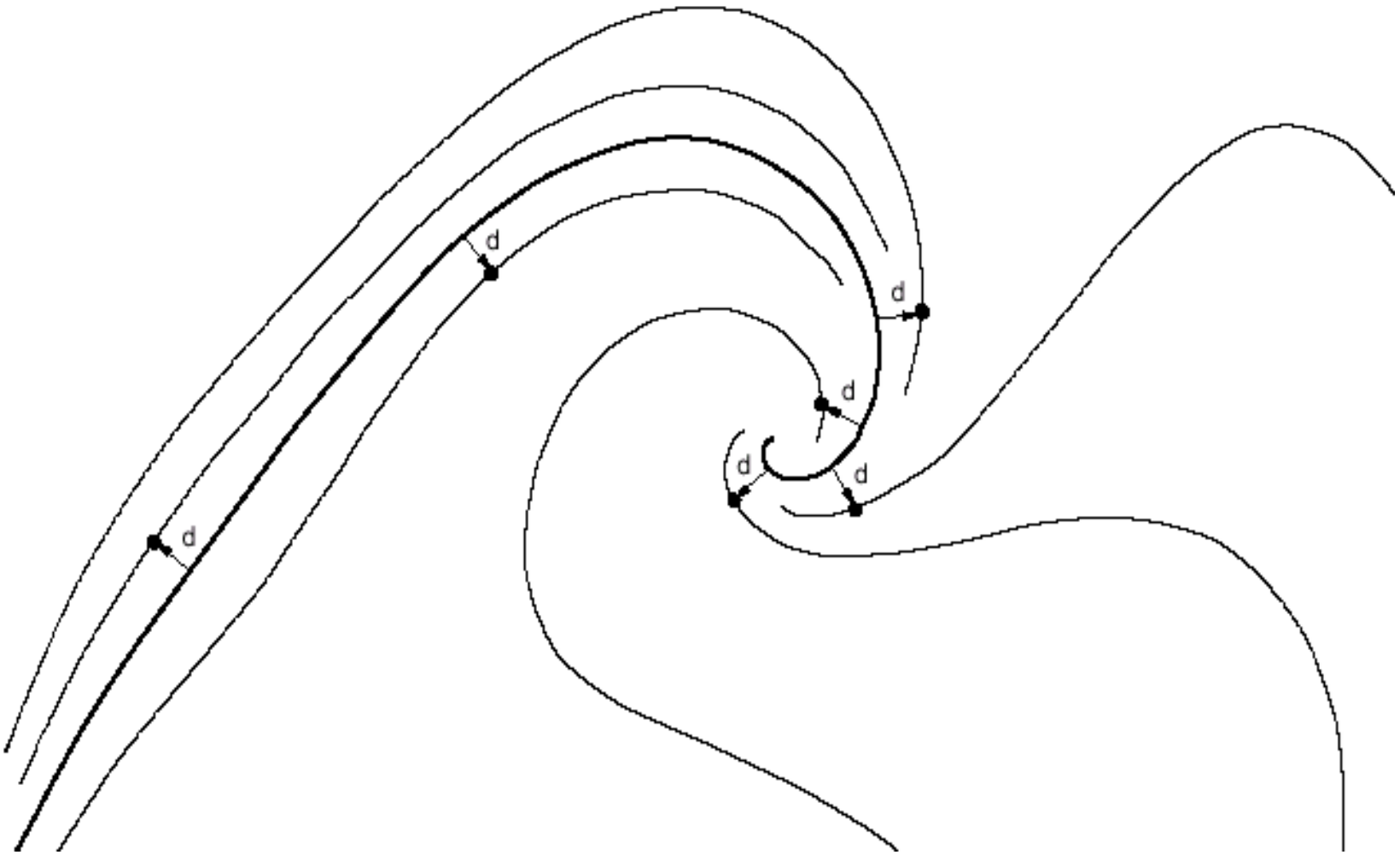
  ELSE next streamline in queue becomes current streamline

Streamline Termination

When to stop streamline integration:

- when distance to neighboring streamline ≤ d_{test}
- when streamline leaves flow domain
- when streamline runs into fixed point ($\mathbf{v} = 0$)
- when streamline gets too near to itself (loop)
- after a certain amount of maximal steps

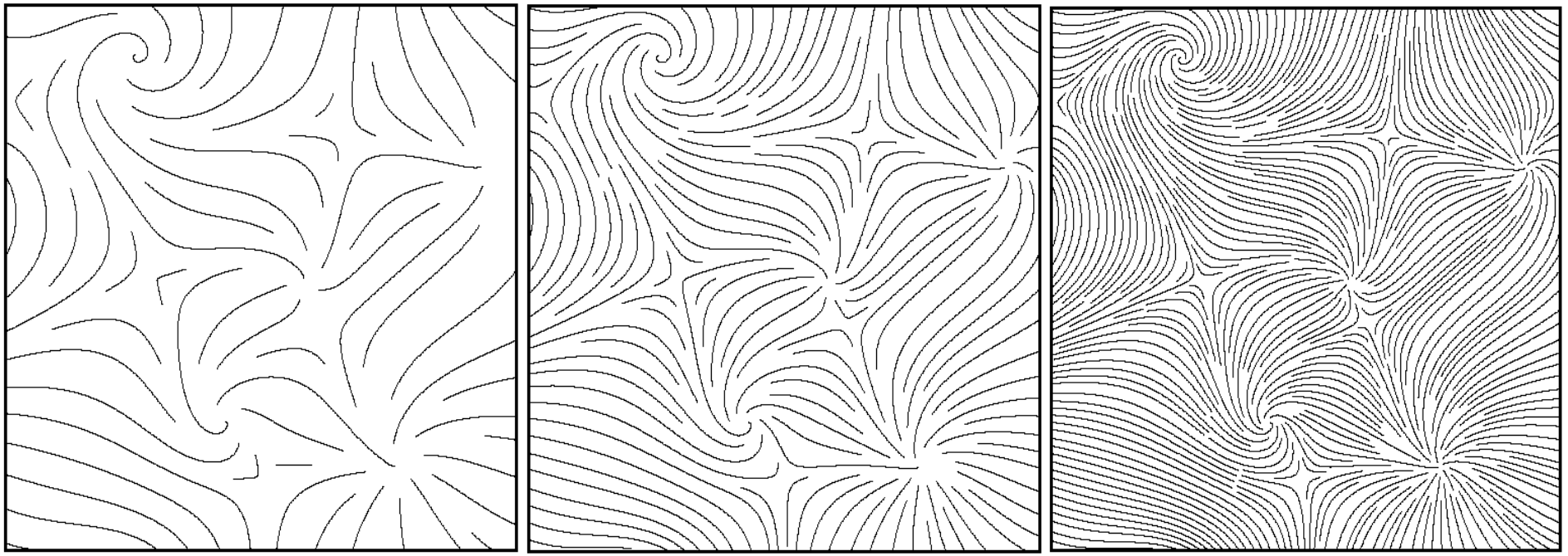
New Streamlines



Different Streamline Densities

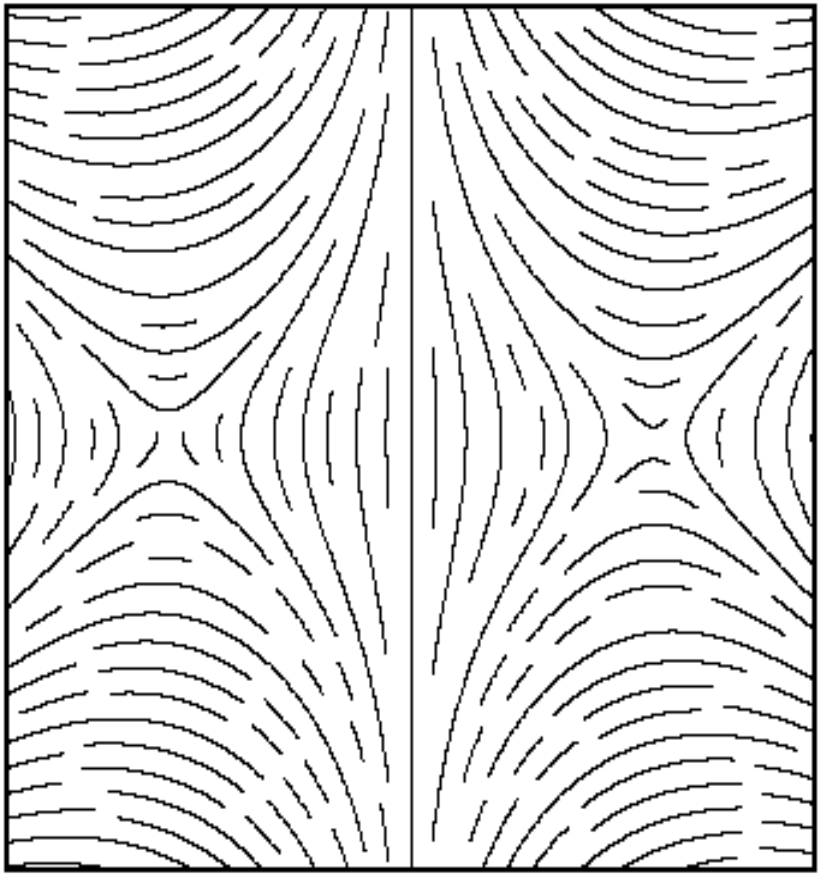
Variations of d_{sep} relative to image width:

6% 3% 1.5%

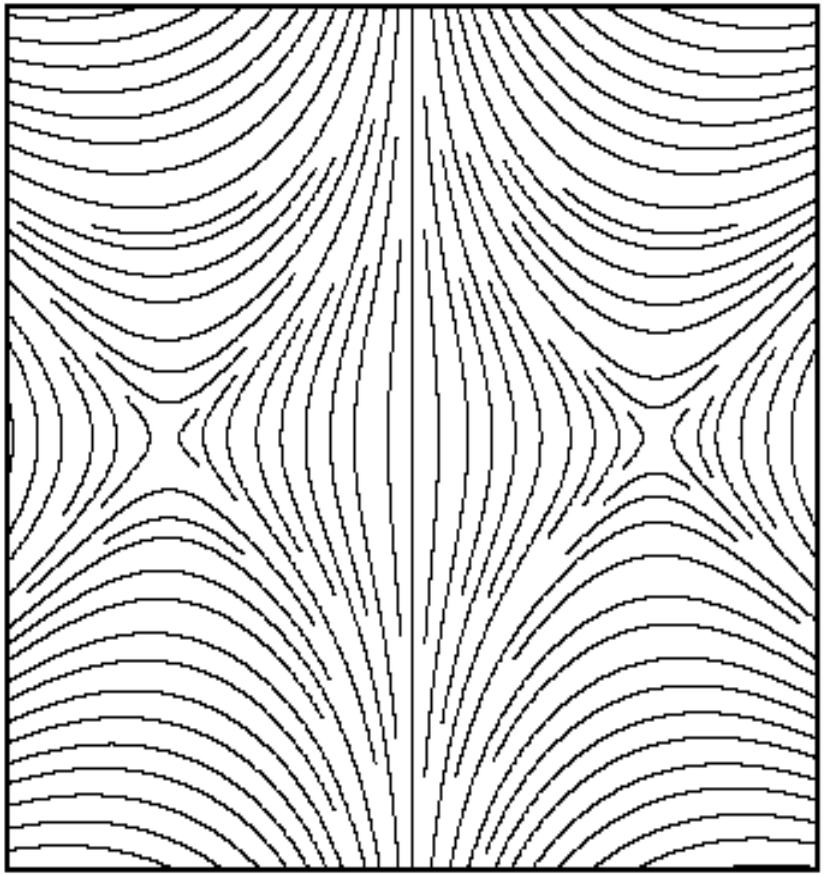


d_{sep} vs. d_{test}

$d_{test} = 0.9 \cdot d_{sep}$



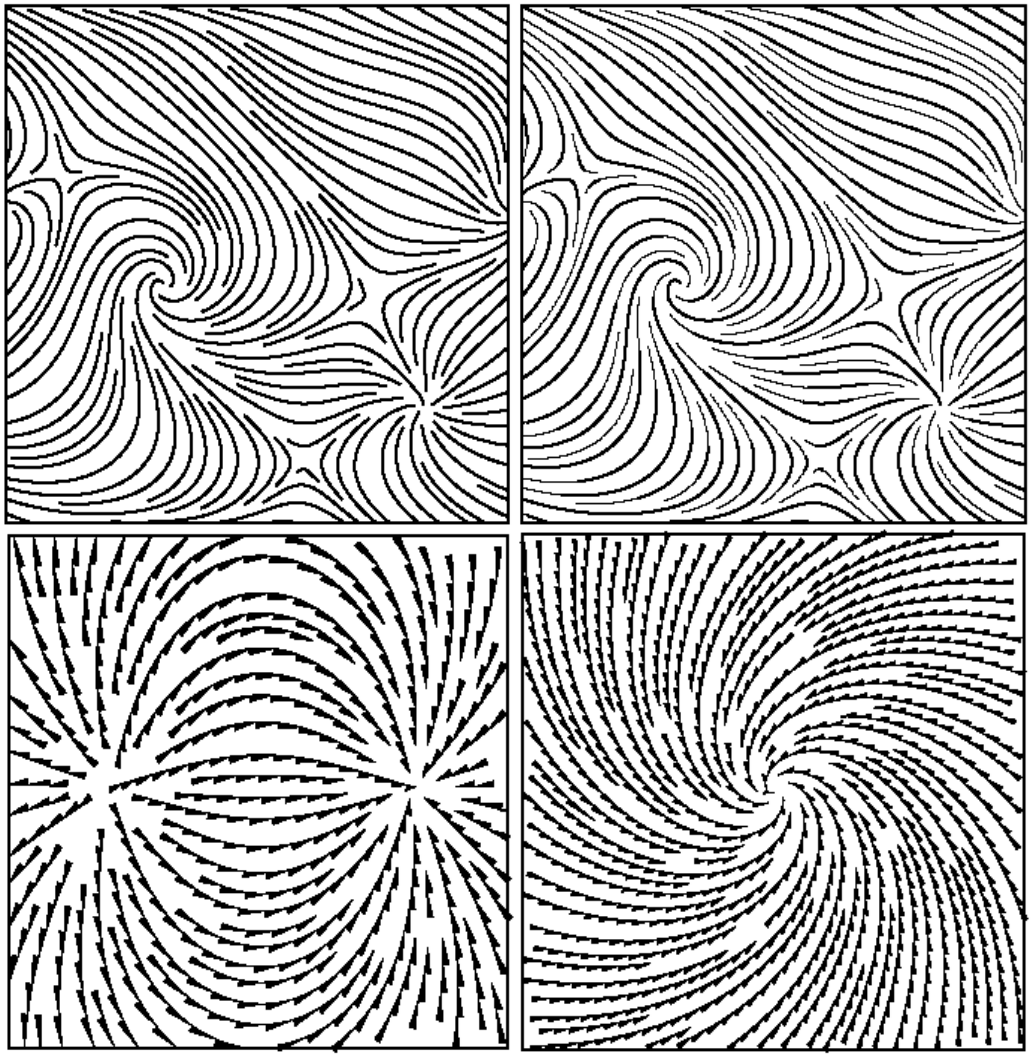
$d_{test} = 0.5 \cdot d_{sep}$



Tapering and Glyphs

Thickness in relation to distance

Directional glyphs:



Literature

For more information, please see:

- B. Jobard & W. Lefer: "**Creating Evenly-Spaced Streamlines of Arbitrary Density**" in *Proceedings of 8th Eurographics Workshop on Visualization in Scientific Computing*, April 1997, pp. 45-55
- **Data Visualization: Principles and Practice, Chapter 6: Vector Visualization** by A. Telea, AK Peters 2008

Acknowledgment

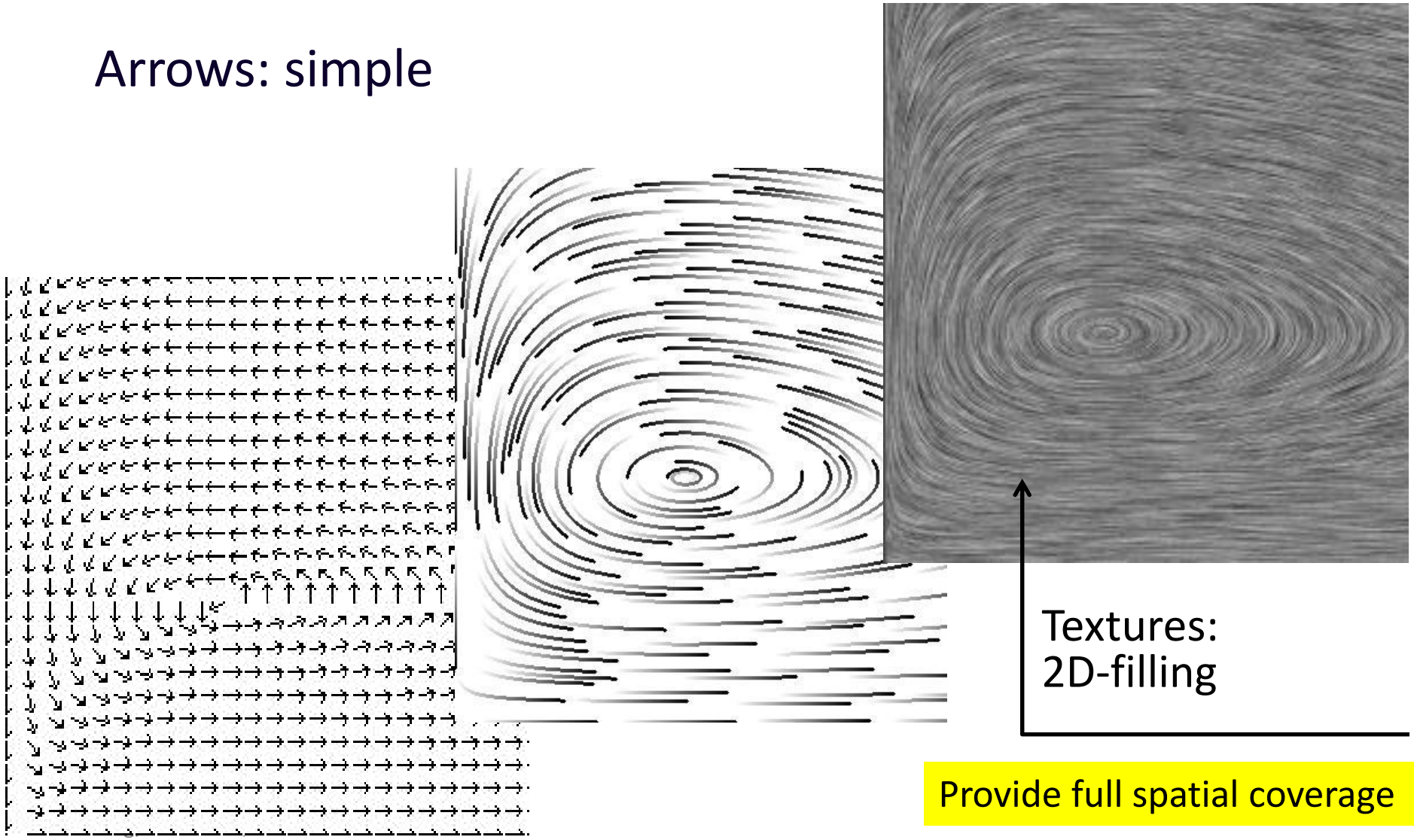
Thanks for the materials

- Prof. Robert S. Laramee, Swansea University, UK

Arrows vs. Streamlines vs. Textures

Streamlines: selective

Arrows: simple



Provide full spatial coverage

Vector Field Visualization: Texture-based Method

A BRIEF OVERVIEW

Overview — Texture-Based Methods

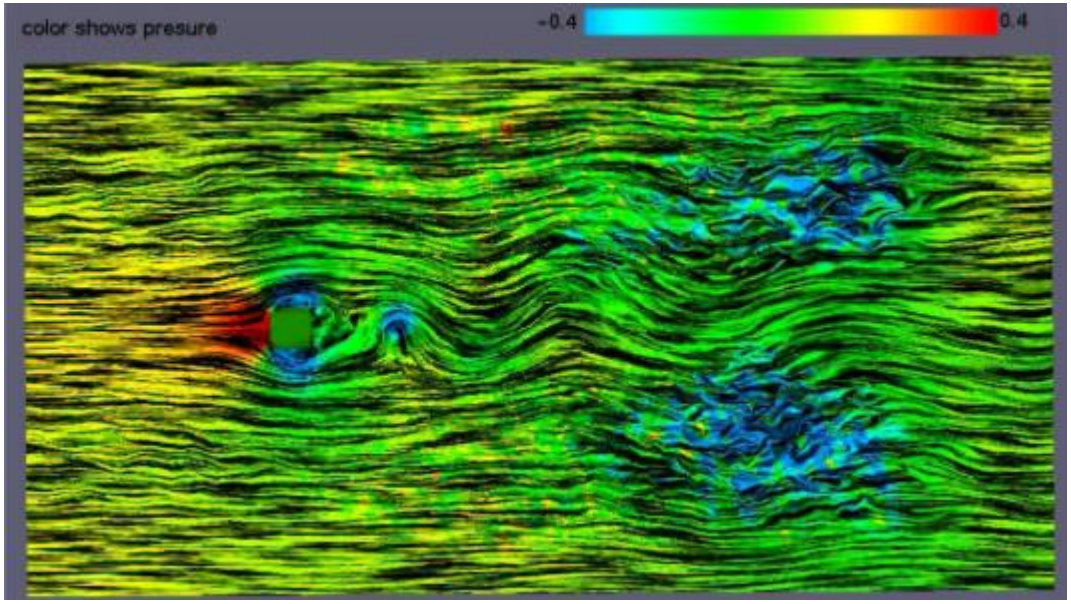
➢ Spot Noise

✧ One of the first texture-based techniques (*Van Wijk, Siggraph1991*).

✧ **Basic idea**: *distribute a set of intensity function, or spot, over the domain, that is wrapped by the flow over a small step*.

✧ **Pro**: mimic the smear effect of oil; encode magnitude; can be applied for both steady and unsteady flow.

✧ **Con**: tricky to implement; low quality; computationally expensive.



[De Leeuw and Van Liere]

Overview — Texture-Based Methods

pixel-based

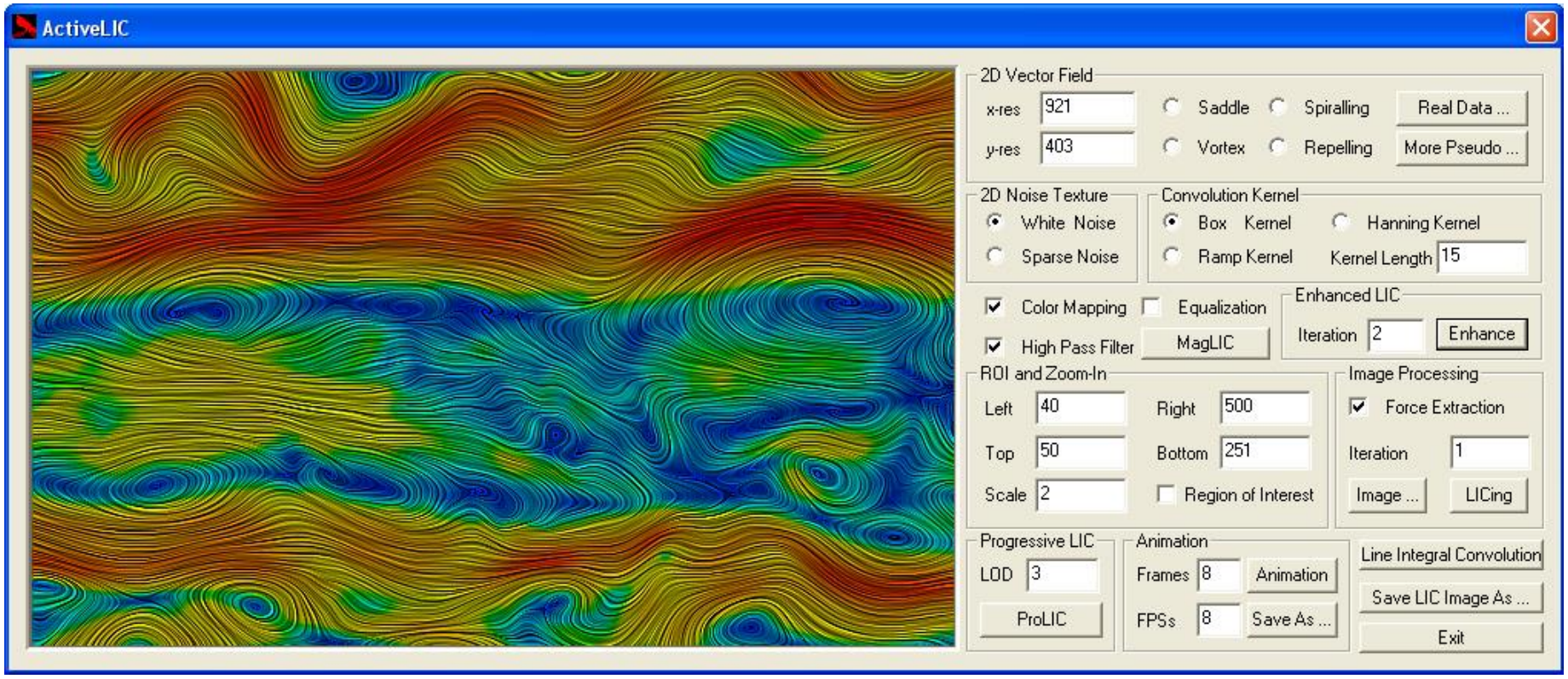
➢ Line Integral Convolution (LIC)

✧ One of the most popular techniques (*Brian Cabral & Leith Leedom, SIGGRAPH93*).

✧ **Basic idea**: *Low-pass filters **white noise** along* pixel-centered symmetrically bi-directional *streamlines* to exploit spatial correlation in the flow direction.

✧ **Pro**: High-quality image with fine features; easy implementation; and many variants.

✧ **Con**: Computationally expensive; limited to steady flow visualization.



Overview — Texture-Based Methods

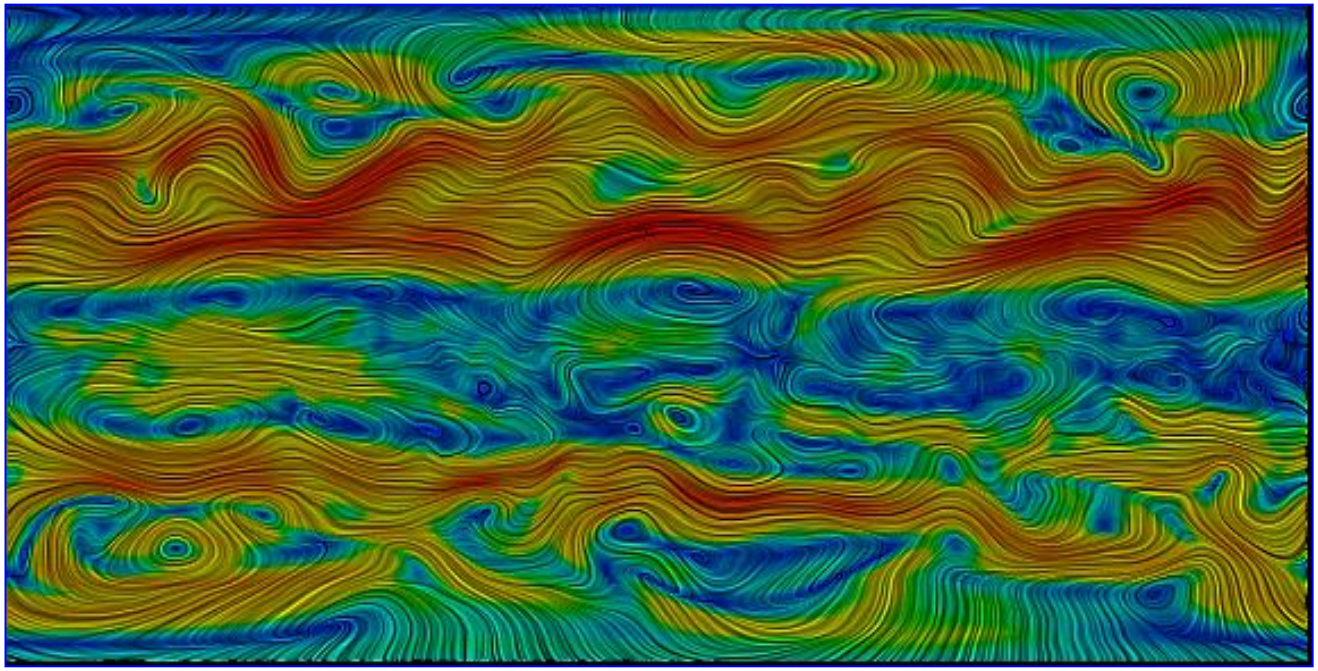
➢ Unsteady Flow LIC (UFLIC)

✧ The first texture-based **unsteady** flow visualization method (by *Han-Wei Shen and David Kao, IEEE Visualization 97 & IEEE TVCG 98*).

✧ **Basic idea:** Time-accurately scatters particle values of *successively fed-forward textures* along **pathlines** *over several time steps* to convey the footprint / contribution that a particle leaves at downstream locations as the flow runs forward.

✧ **Pro**: High temporal coherence & high spatial coherence & hardware-independent.

✧ **Con**: Low computational performance due to *multi-step* (≈ 100) *pathline integration.*



Overview — Texture-Based Methods

➢ Hardware-Accelerated Texture Advection (HATA)

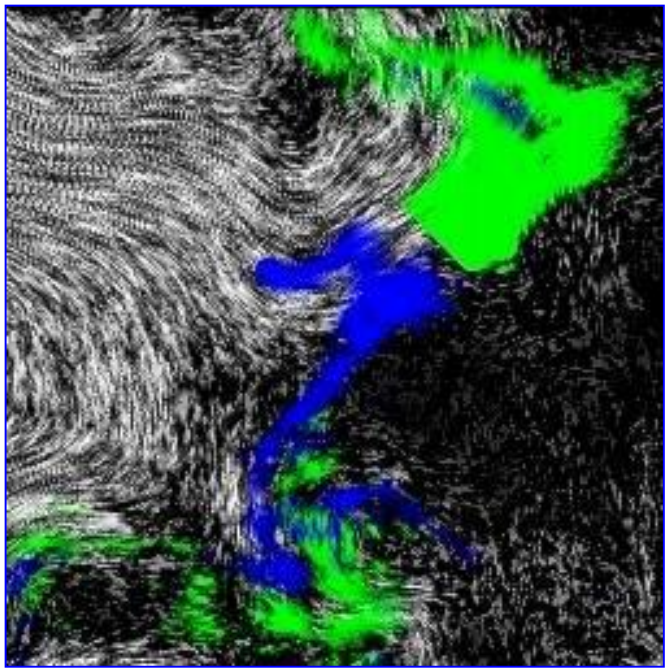
✧ The **first hardware-based** texture synthesis technique for unsteady flow vis (by *Bruno Jobard and et al, IEEE Visualization 00*).

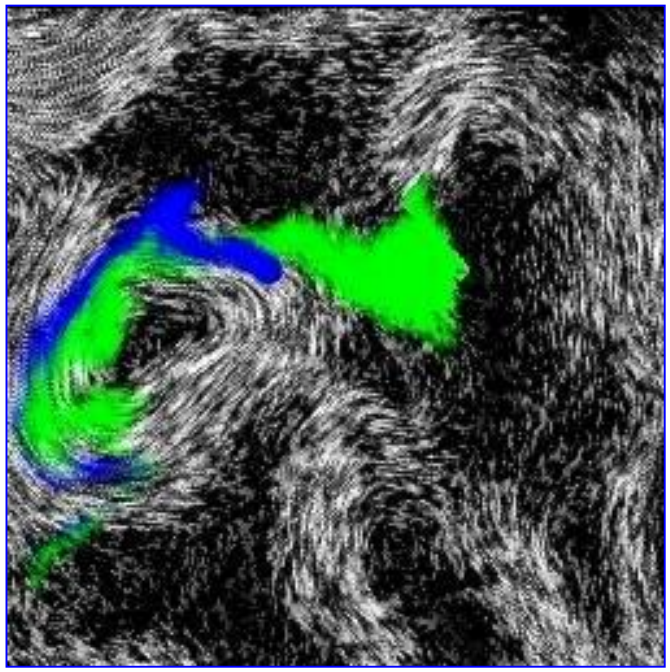
✧ **Basic idea:** Exploits *indirect pixel-texture addressing* for fast flow advection, & additive / subtractive texture blending for fast texture convolution in an efficient pipeline.

✧ **Pro:** Near-interactive frame rates based on special-purpose graphics cards; for both steady and unsteady flow; good temporal coherence .

✧ **Con:** poor spatial coherence (very noisy).

exponential kernel





(Bruno Jobard, Gordon Erlebacher, and M. Yousuff Hussaini)

Overview — Texture-Based Methods

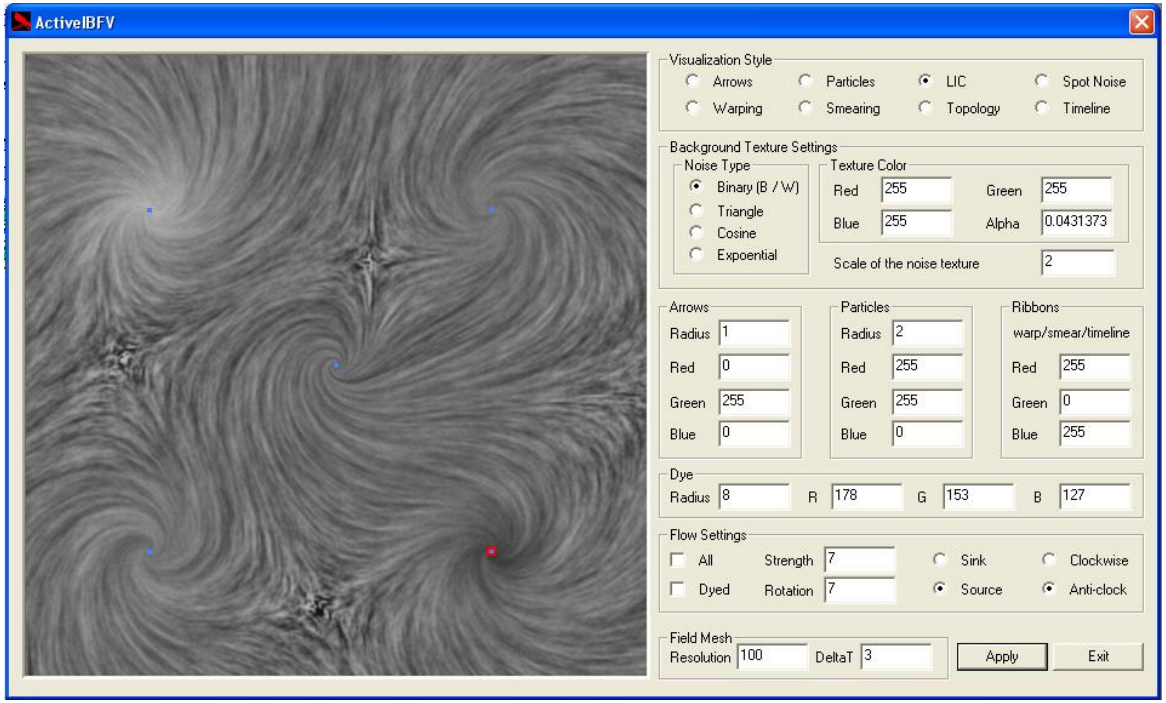
➢ Image-Based Flow Visualization (IBFV)

✧ One of the most versatile and the easiest-to-implement hardware-based methods (by *Jarke J. van Wijk, SIGGRAPH02*).

✧ **Basic idea:** Designs a sequence of *temporally-spatially low-pass filtered* noise textures and cyclically blends them with an iteratively advected (using *forward single-step pathline integration*) image (which is initially a BLACK rectangle).

✧ **Pro:** Interactive frame rates and easy simulation of many visualization techniques; good temporal coherence .

✧ **Con:** insufficient spatial coherence (*noisy or blurred*).



Overview — Texture-Based Methods

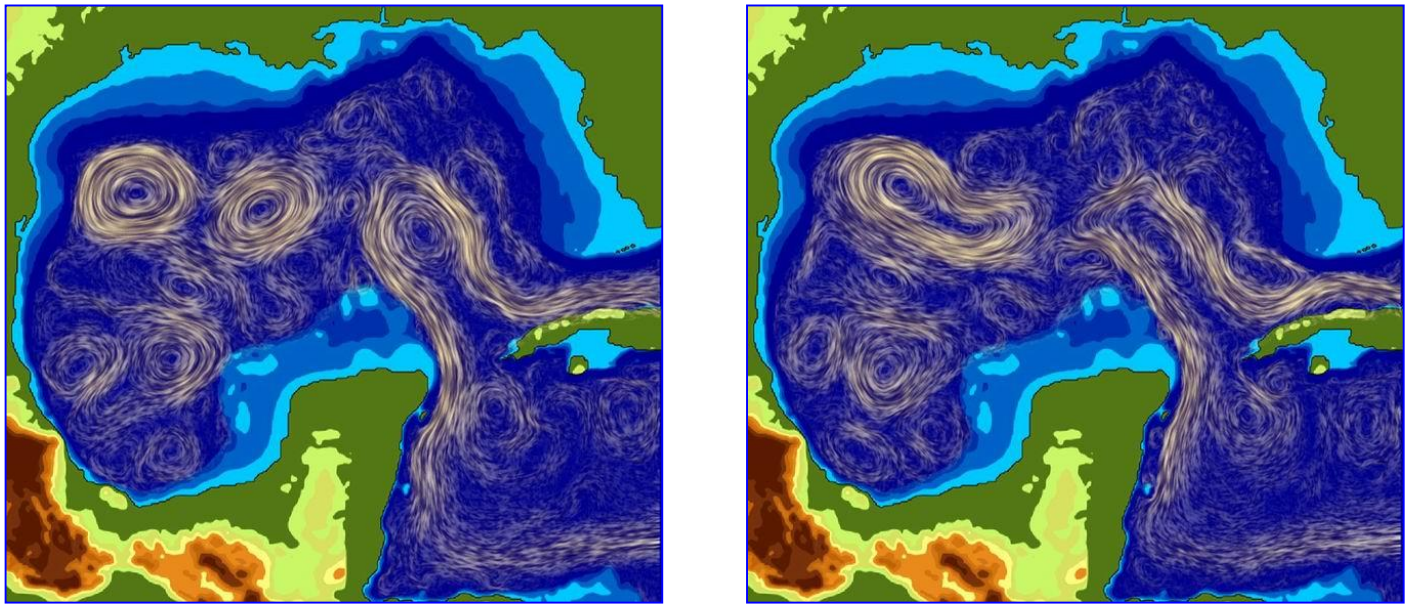
➢ Lagrangian-Eulerian Advection (LEA)

✧ A fast hardware-independent unsteady flow visualization method (by *Bruno Jobard and et al, IEEE TVCG 02*).

✧ **Basic idea:** Employs *backward single-step pathline integration* to search the previous frame for the contributing particle (Eulerian) which scatters the texture value to the target pixel of the current frame (Lagrangian) & blends successive textures.

✧ **Pro:** Interactive frame rates and supportive of arbitrarily-shaped field domains; good temporal coherence .

✧ **Con:** insufficient spatial coherence (obscure direction).

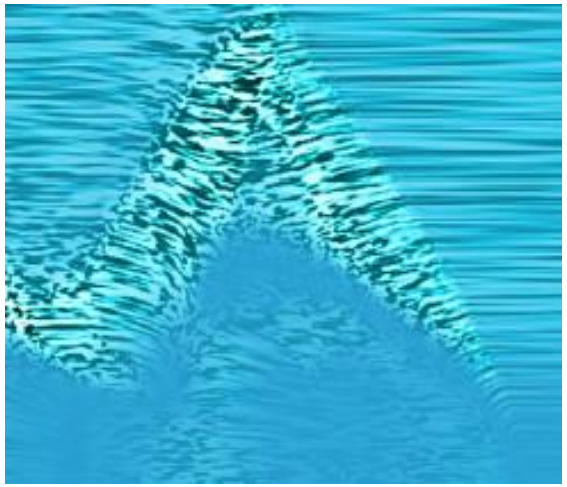


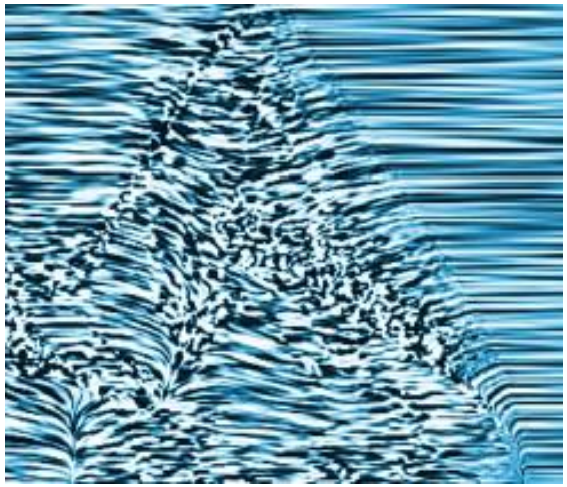
(Bruno Jobard, Gordon Erlebacher, and M. Yousuff Hussaini)

Overview — Texture-Based Methods

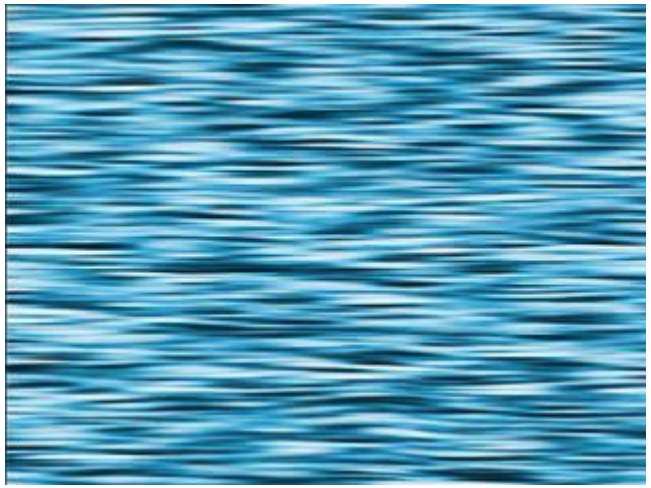
➢ Unsteady Flow Advection-Convolution (UFAC)

- ✧ A separably temporal-spatial texture synthesis method for unsteady flow fields (by *Daniel Weiskopf and et al, IEEE Visualization 03*).
- ✧ **Basic idea:** Establishes temporal coherence by *property advection along pathlines* while building spatial correlation by *texture convolution along streamlines.*
- ✧ With explicit, direct, and separate control over temporal coherence and spatial coherence to balance visualization speed and quality.
- ✧ **Pro:** Interactive rates on graphics cards with fragment (e.g., pixel shader) support.
- ✧ **Con:** Temporal-spatial inconsistency — either flickering animation or noisy image.





Noisy images with (left) / without (right) velocity masking



Good frames in a flickering animation

(Daniel Weiskopf, Gordon Erlebacher, and Thomas Ertl)

Overview — Texture-Based Methods

➢ Steady Flow Visualization Methods

Method	Noise design	Implementation	Image quality	Feature missing	Extensions	Performance
Spot Noise	tricky	tedious	low	yes	few	low
LIC	easy	easy	high	no	many	low

➢ Unsteady Flow Visualization Methods

Method	Temporal coherence	Spatial coherence	Performance	Graphics cards
UFLIC	high	high	low	not required
HATA	good	poor (very noisy)	near-interactive rates	special-purpose
IBFV	good	insufficient (noisy / blurred)	interactive rates	general-purpose
LEA	good	insufficient (obscure direction)	interactive	not required
UFAC	trade-off between noisy image & flickering animation		interactive	special-purpose

Recent Advances

Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf, **The state of the art in flow visualization: dense and texture-Based techniques.** in *Computer Graphics Forum (CGF)*, Vol. 23, No. 2, 2004, pages 203-221.

Guo-Shi Li, Xavier Tricoche, Daniel Weiskopf, and Charles Hansen. Flow Charts: **Visualization of vector fields on arbitrary surface**. IEEE Transactions on Visualization and Computer Graphics, 14(5), pp. 1067-1080.

Jin Huang, Wenjie Pei, Chunfeng Wen, Guoning Chen, Wei Chen, and Hujun Bao. **Output-coherent image-space LIC for surface flow visualization**. IEEE Pacific Visualization Symposium 2012.

Victor Matvienko, Jens Krüger. **Dense flow visualization uisng wave interference**. IEEE Pacific Visualization Symposium 2012.

Jin Huang, Zherong Pan, Guoning Chen, Wei Chen, and Hujun Bao. **Image-Space Texture-Based Output-Coherent Surface Flow Visualization**, IEEE Transactions on Visualization and Computer Graphics, Vol. 19 (9): pp. 1476-1487, 2013.

SOME DETAILS

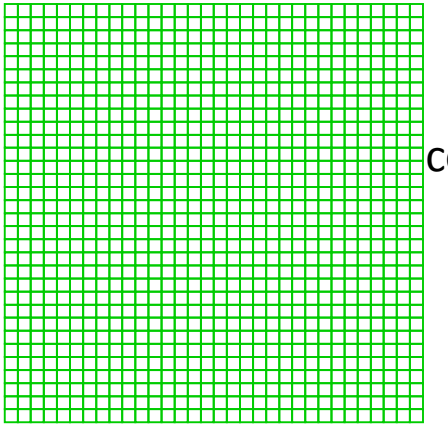
Line Integral Convolution — LIC

Line Integral Convolution (LIC) was presented by *Brian Cabral* and *Casey Leedom* (*ACM SIGGRAPH93*). (cited by 1204 till 2014)

➢ Basic Idea

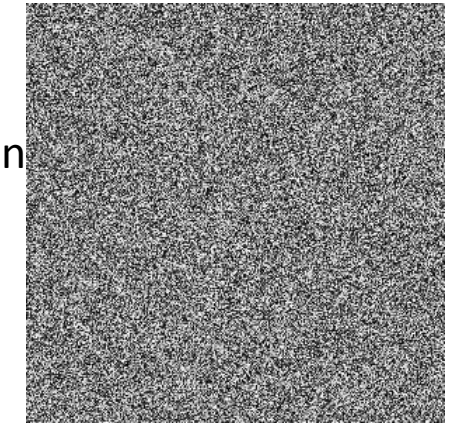
✧ LIC convolves *white noise* using a *low-pass filter* along *pixel-centered symmetrically bi-directional streamlines* to exploit spatial correlation in the flow direction — anisotropic low-pass filtering along flow lines.

✧ LIC synthesizes an image that provides *a global dense representation* of the flow, analogous to *the resulting pattern of wind-blown sand*.

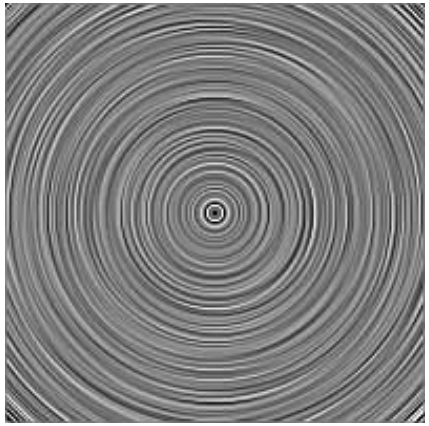


flow field (wind)

convolution ⊕ (blow)



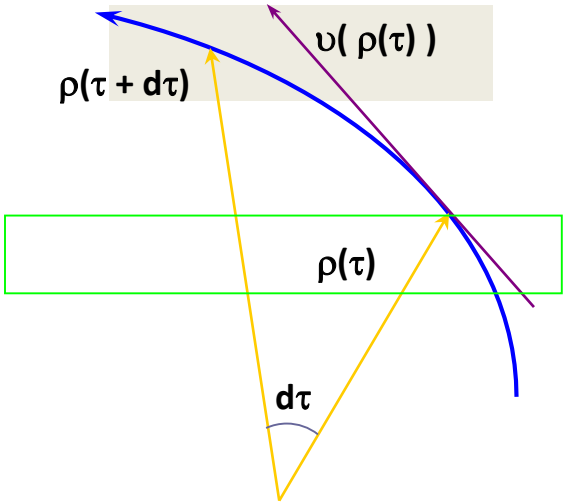
white noise (fine sand)



LIC image (pattern)

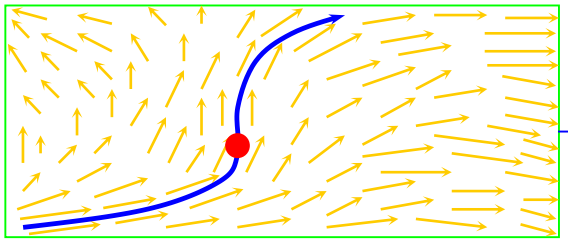
white noise → the texture is freely warped / driven by the flow without any intrinsic resistance

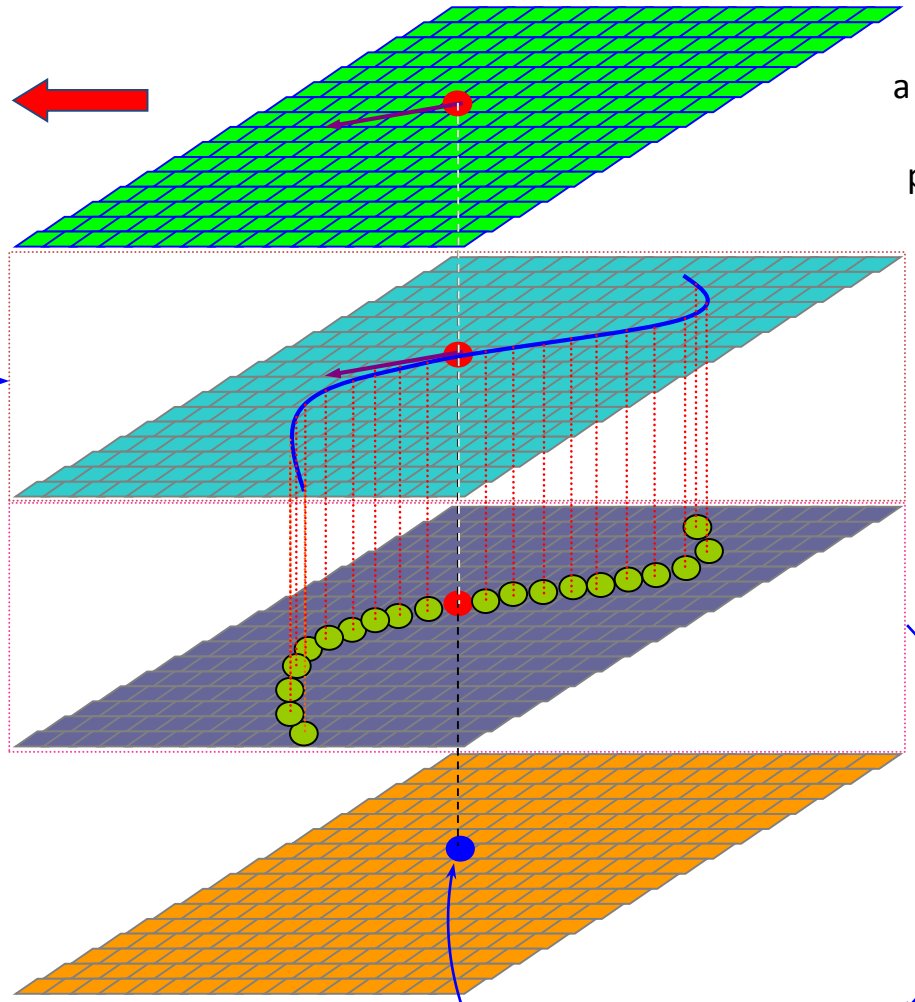
Line Integral Convolution — LIC



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

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





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

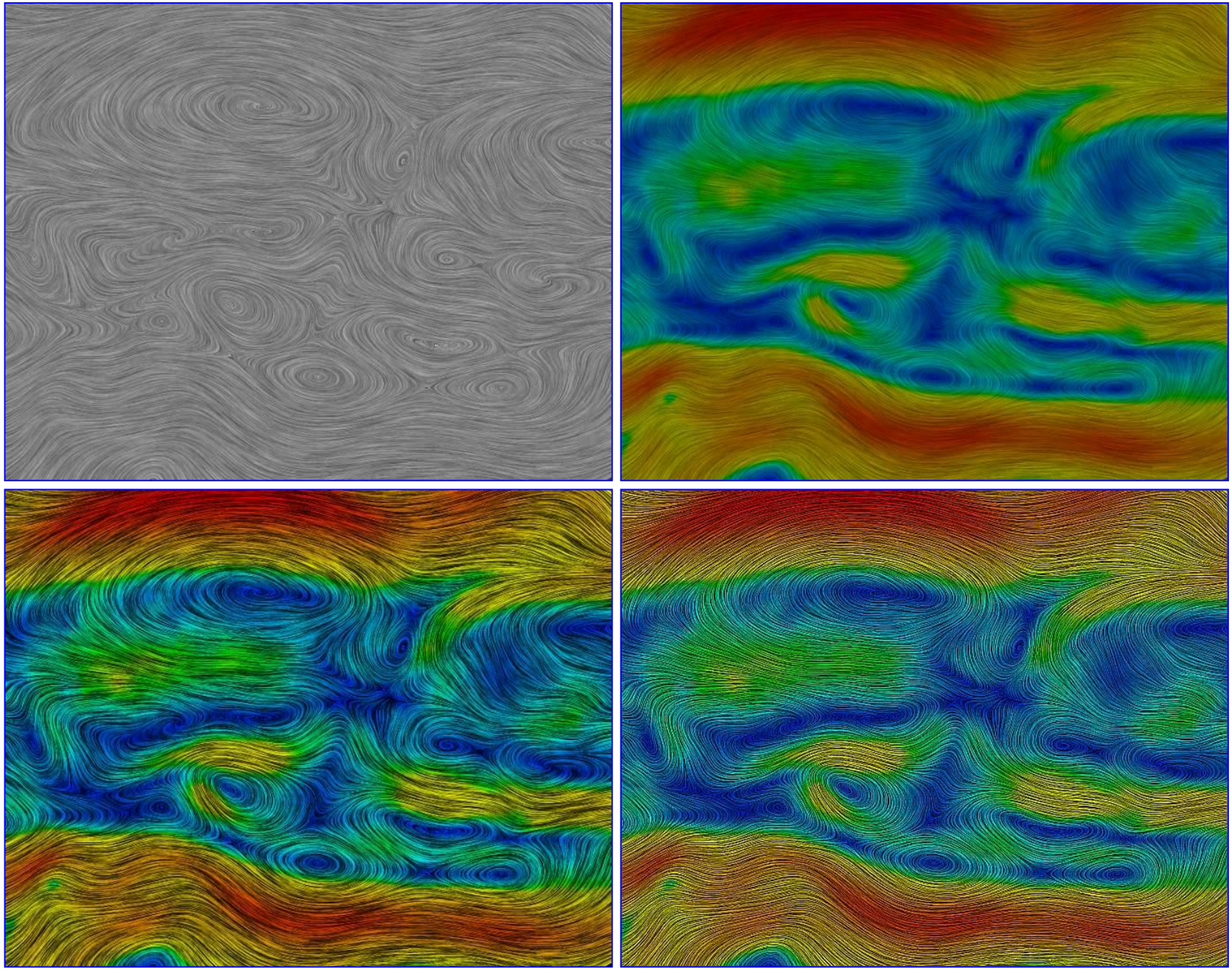
the correlated pixels along the streamline

index the input noise for the texture values

compute the target pixel value in the LIC image by convolution

$$T'(\rho(0)) = \frac{\int_{-L}^{L} K(\tau) T(\rho(\tau)) d\tau}{\int_{-L}^{L} K(\tau) d\tau}$$

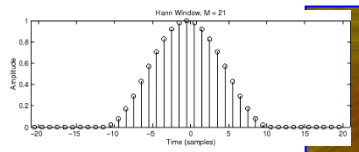
Zhanping Liu @ MSU / HPC / VAIL

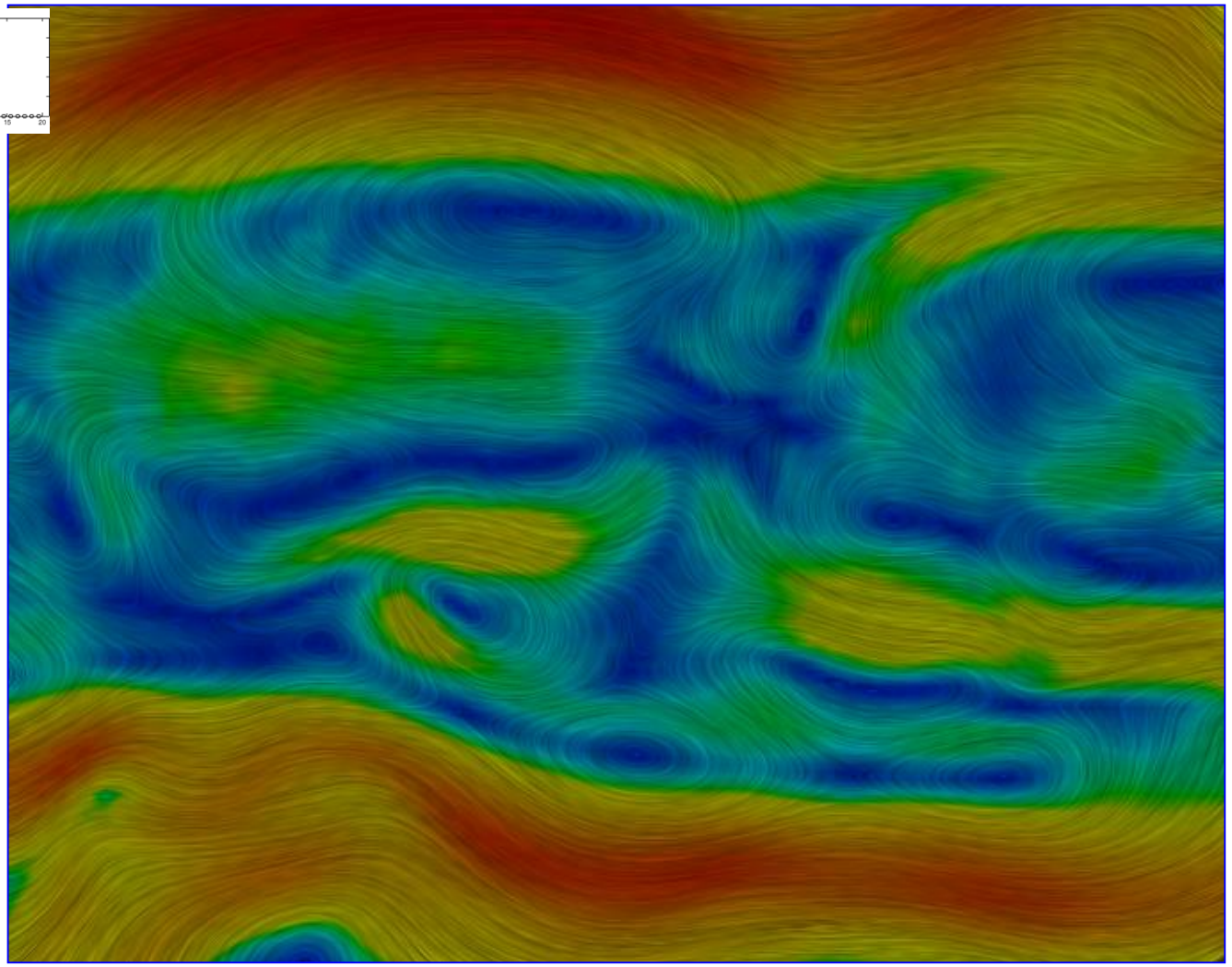


Top-left: gray-scale LIC
Top-right: color-mapped LIC
Bottom-left: contrasted LIC
Bottom-right: high-pass filtered LIC

Line Integral Convolution — LIC

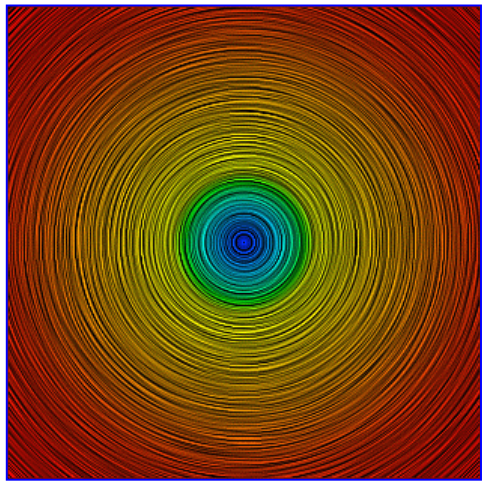
➢ **Animation** successively shifting the phase of a periodic convolution kernel such as Hanning filter ("Motion Without Movement", CG´91)



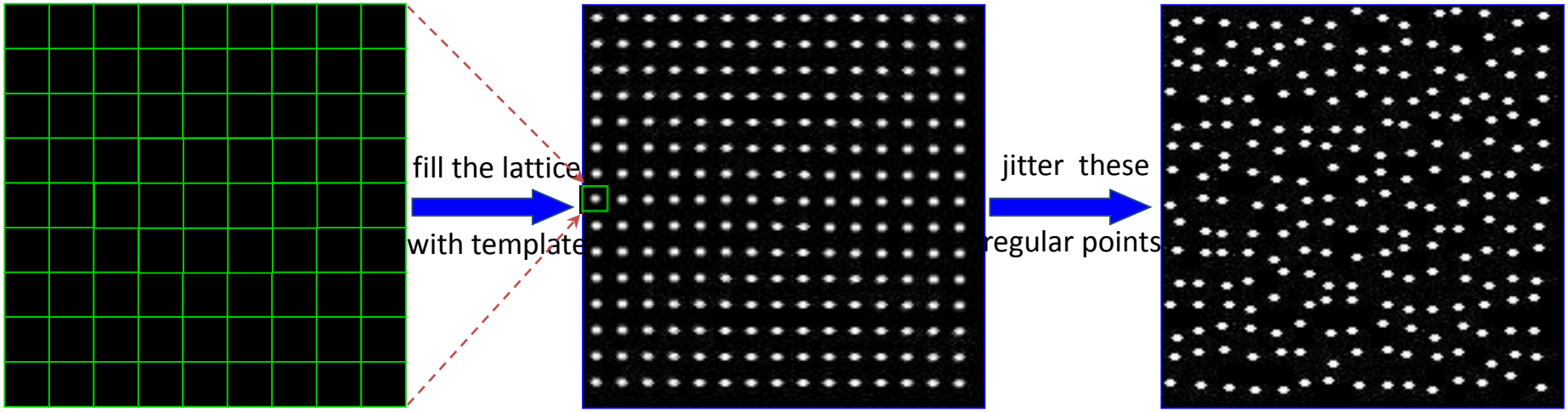


Line Integral Convolution — LIC Variants

➢ OLIC (Oriented LIC)



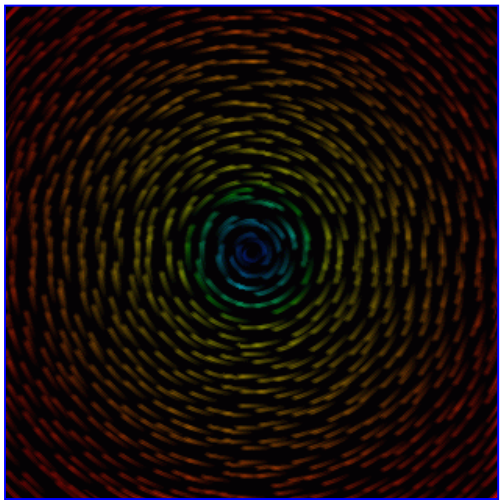
- ✧ R. Wegenkittl and et al. (*Computer Animation 97*).
- ✧ A LIC image shows the flow direction while failing to show the orientation (clockwise or counter-clockwise ?).
- ✧ A ramp filter offers orientation cue by *intensity tapering.*
- ✧ *Sparse noise* offers *enough space for intensity-tapering.*
- ✧ White points of some size are placed at the lattice and then *slightly jittered.*

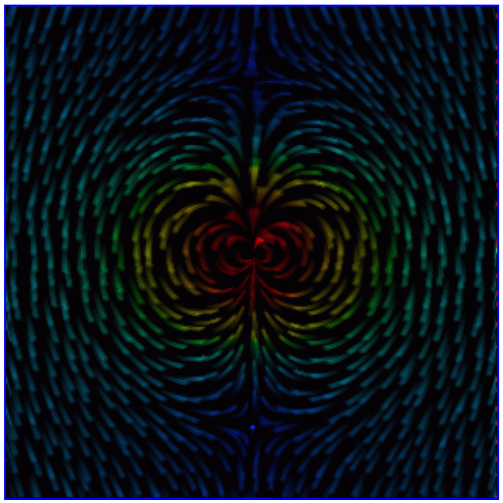


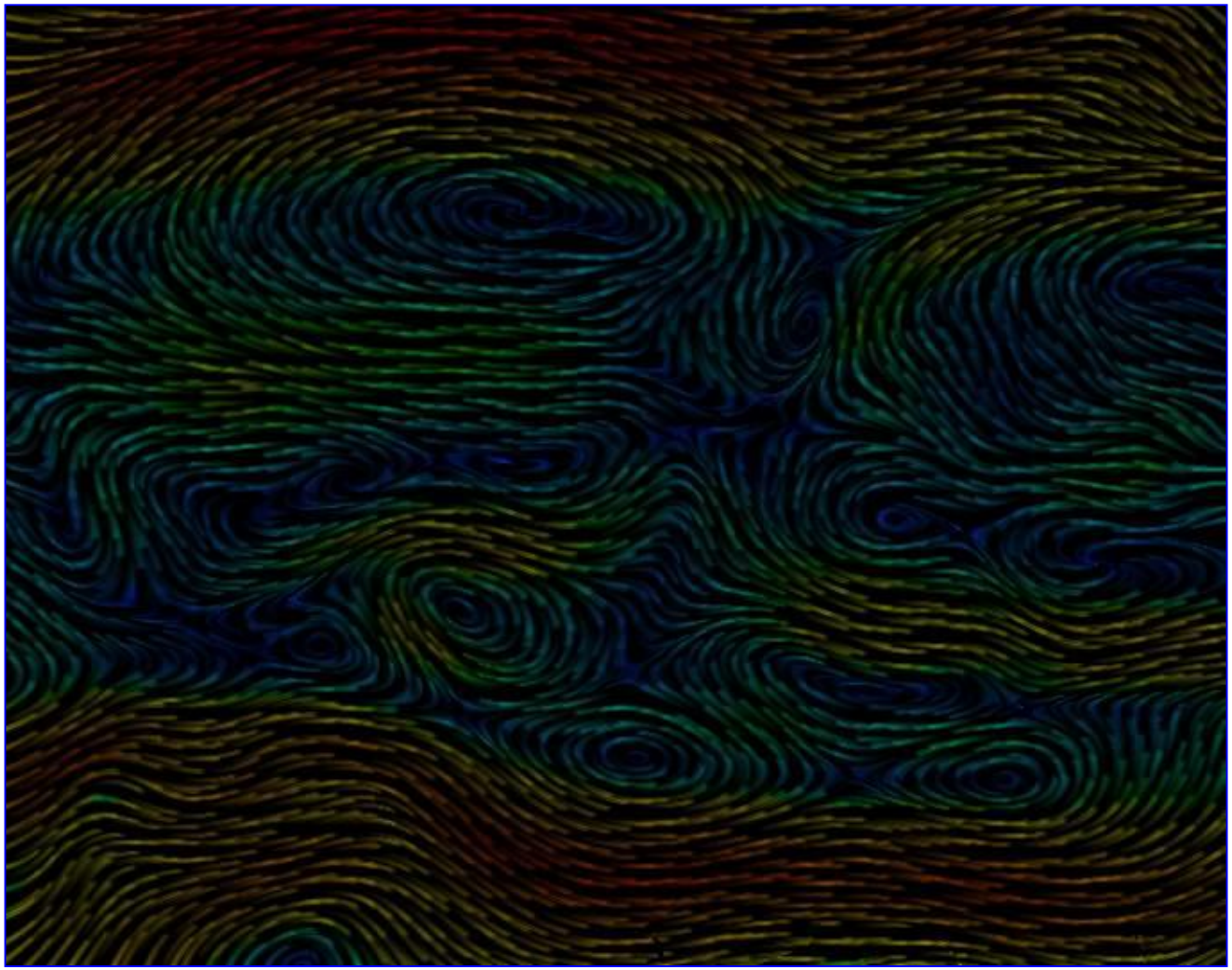
the design of sparse noise

Line Integral Convolution — LIC Variants

Sparse noise ⊕ Ramp convolution kernel ⇨ OLIC (flow orientation in a LIC image)



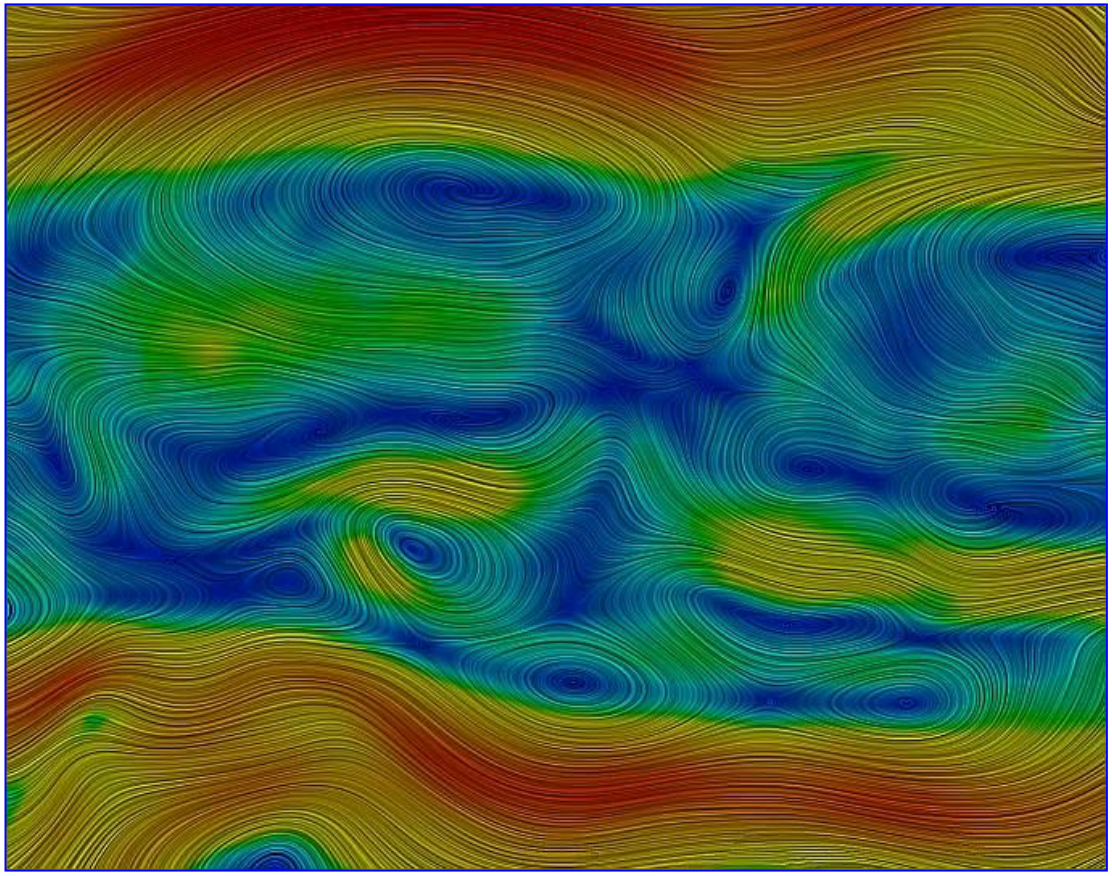




Line Integral Convolution — LIC Variants

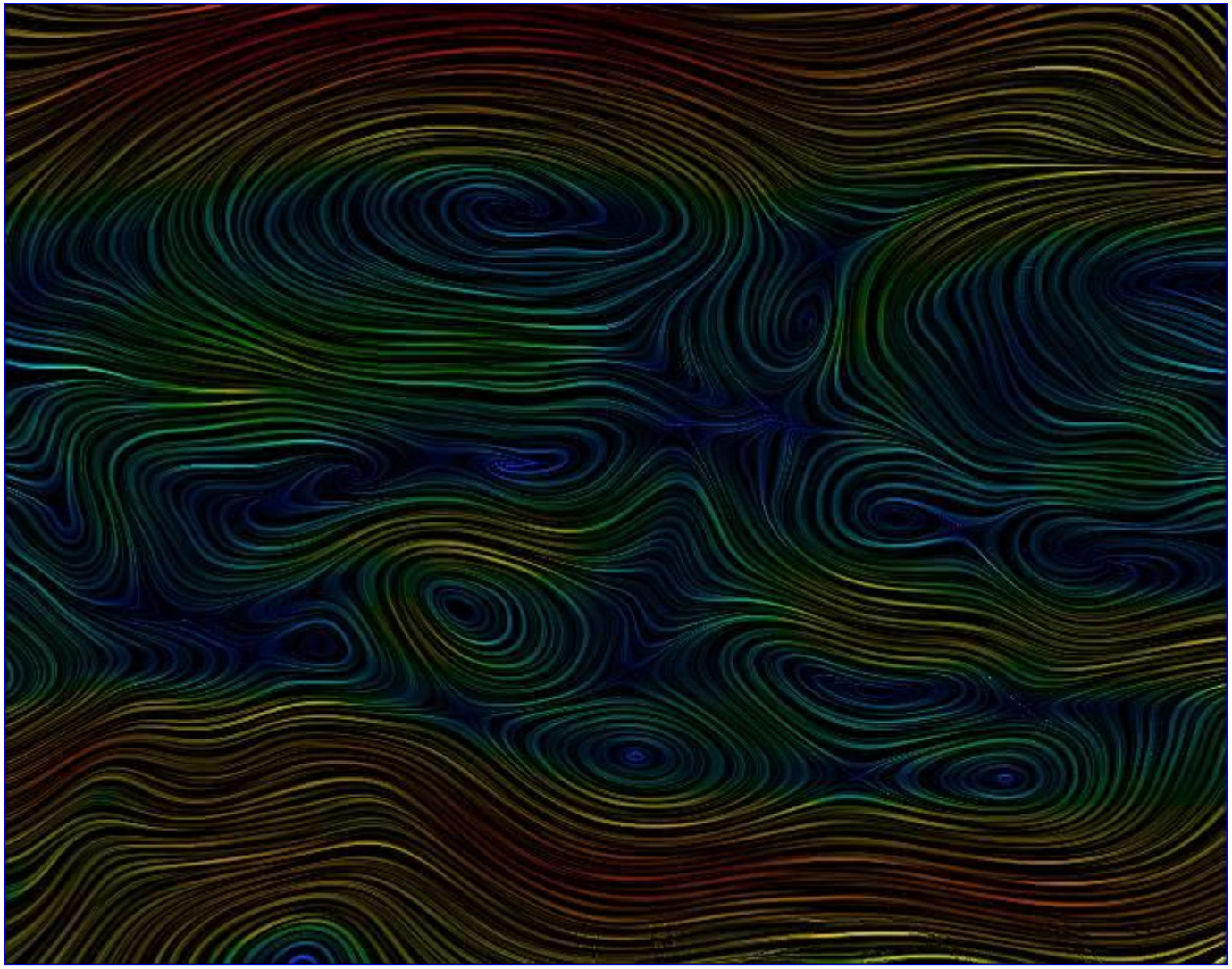
➢ Enhanced LIC

- ✧ A. Okada and D. L. Kao (*IS & T / SPIE Electronics Imaging 97*).
- ✧ *Enhances the appearance of streamlines* — neither noisy nor blurred.
- ✧ Iteratively (iteration times >= 2) *takes an output LIC image as the input* to the next LIC cycle prior to *final high-pass filtering* (e.g., Laplacian filter).



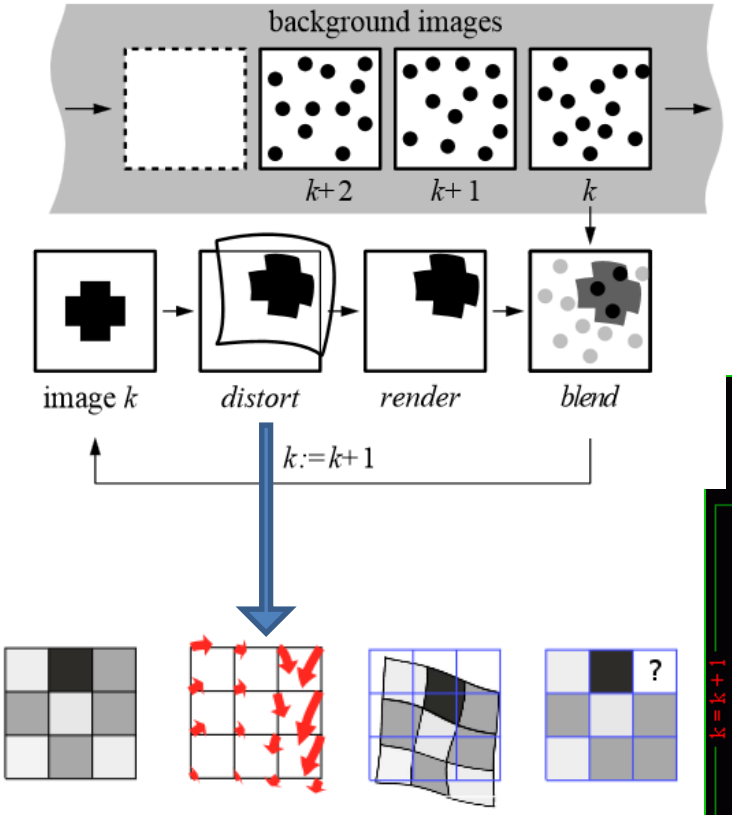
Line Integral Convolution — LIC Variants

A quite fancy LIC image results from using *sparse noise in enhanced LIC.*

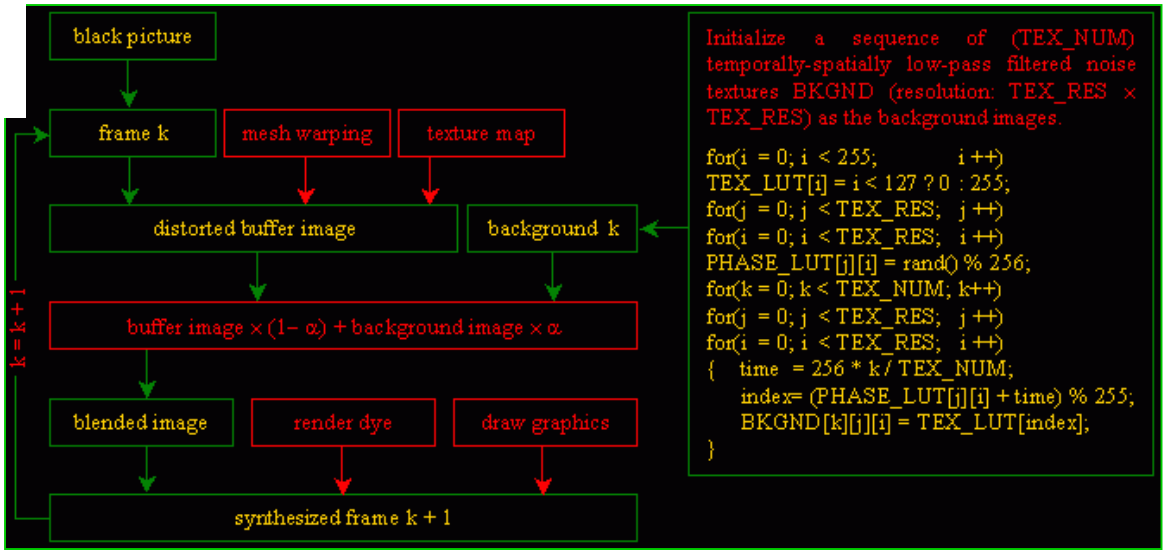


Details — Texture-Based Methods

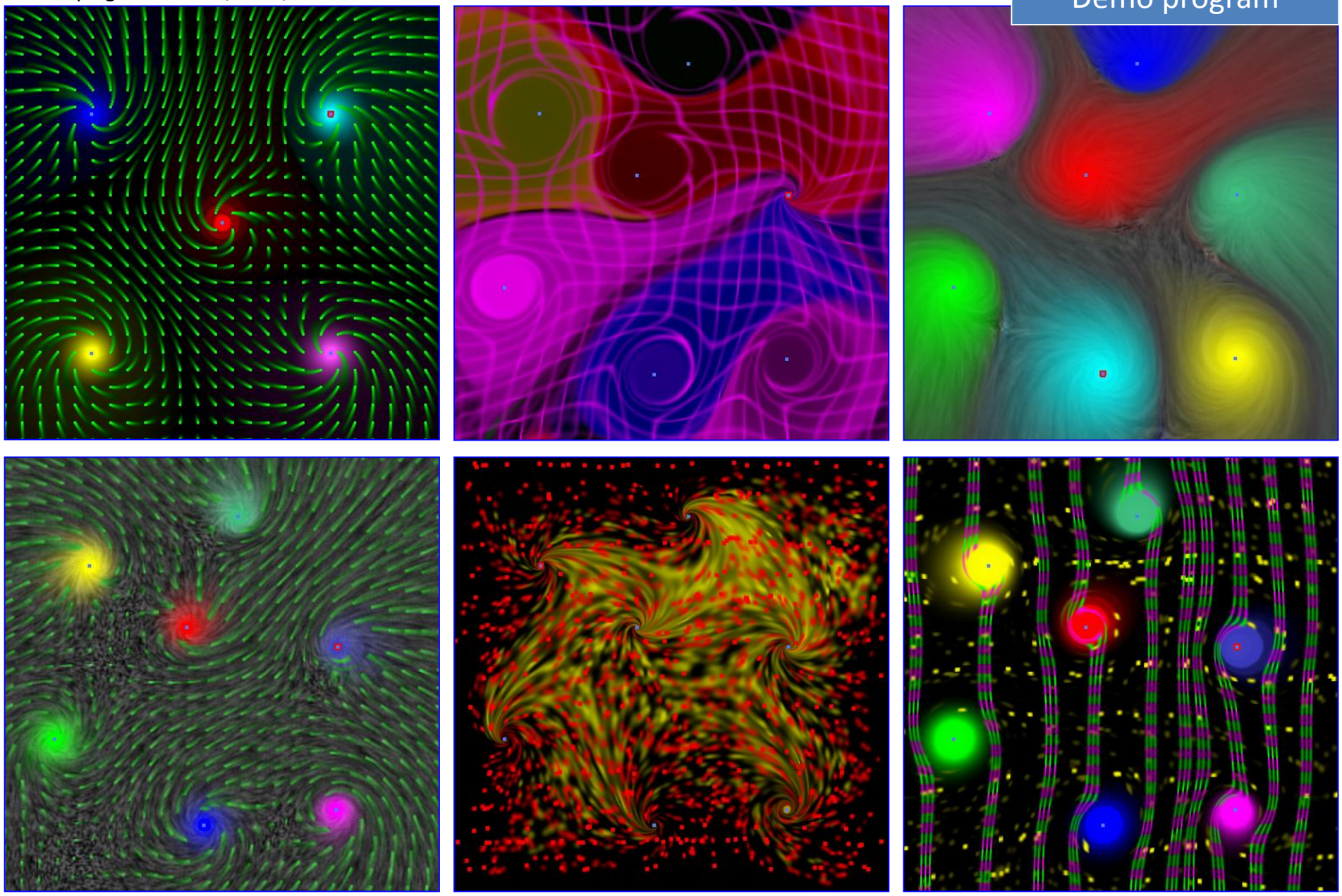
➢ Image-Based Flow Visualization (IBFV)



$$F(\mathbf{p}_k; k) = (1-\alpha)F(\mathbf{p}_{k-1}; k-1) + \alpha G(\mathbf{p}_k; k)$$



Demo program



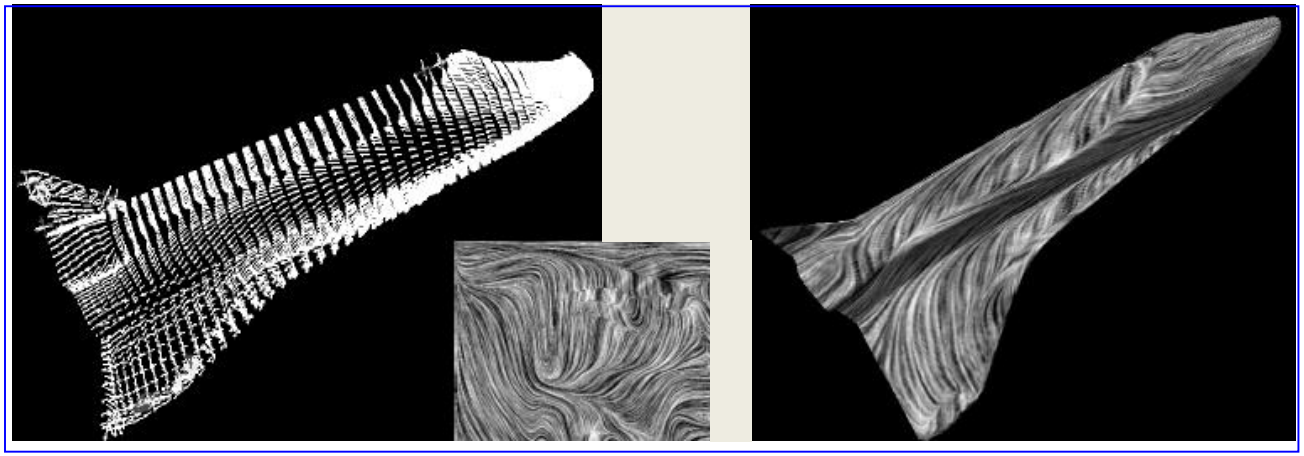
A variety of visualization techniques such as particles, arrow plots, streamlines, timelines, spot noise, LIC, and flow topology can be easily simulated by tuning IBFV parameters

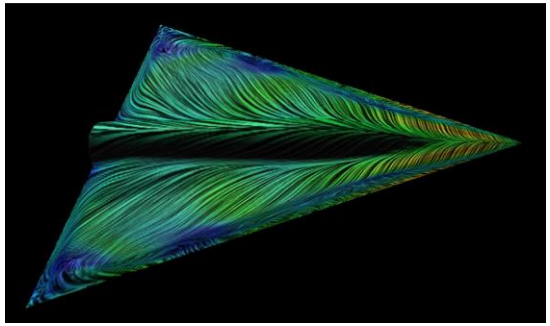
TEXTURE-BASED VISUALIZATION FOR SURFACE FLOW

Line Integral Convolution — LIC Variants

➢ Surface LIC

- ✧ Dense visualization of flows on curved surfaces
- ✧ **Parametric surface LIC — on well-defined surfaces**
  - ◆ On a parameterized CFD surface (model).
  - ◆ On a parameterized stream surface extracted by *Advancing Front* from 3D flows.
  - ◆ *Maps vectors from physical space to parametric space by nonlinear transform.*
  - ◆ *Generates a 2D LIC texture in parametric space.*
  - ◆ *Maps the 2D LIC texture back onto the curved surface (physical space).*
  - ◆ Compensates texture distortions from non-isometric physical-parametric space mapping by using carefully-designed input noise and adaptive kernel length.



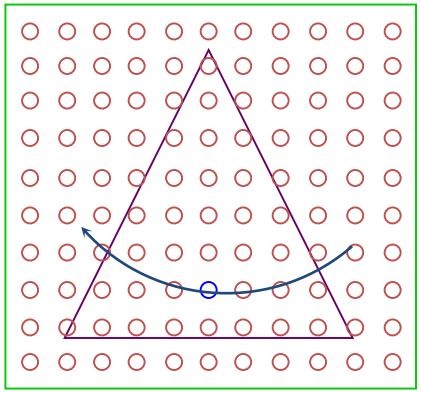


(Lisa Forssell et al., IEEE TVCG 95)

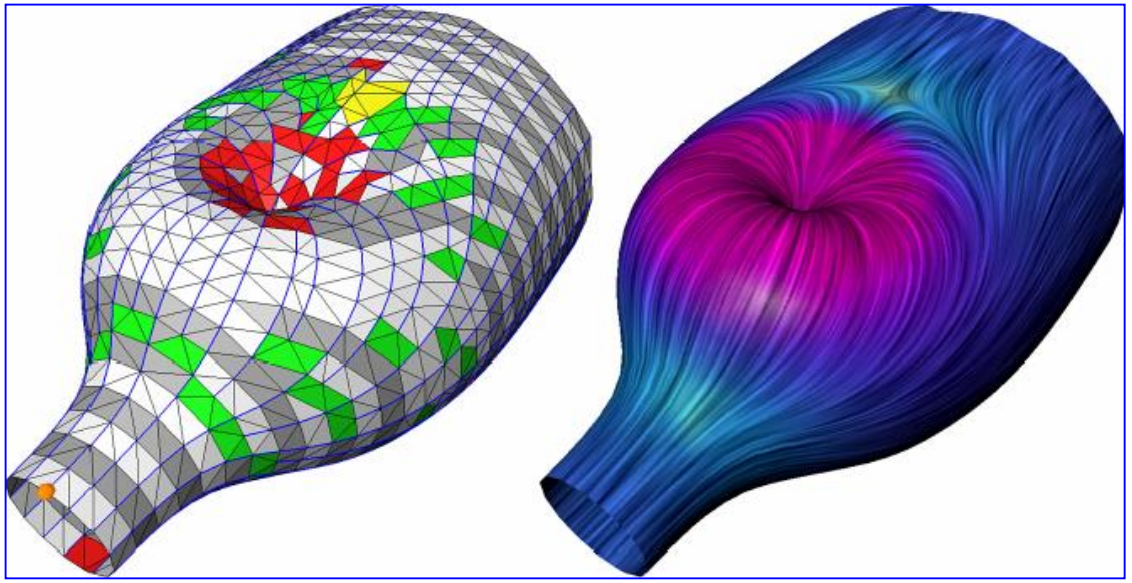
Line Integral Convolution — LIC Variants

✧ Triangulated surface LIC — on arbitrarily complex surfaces

- On extracted iso-surfaces or other implicit surfaces through a volume flow.
- Adopts fast and robust streamline integration directly on a triangular domain.
- Obviates non-isometric space mapping to avoid texture distortions.
- Uses *solid noise* (usually by a procedural noise function).
- Obtains the value of each texel (texture element) sampled in a triangle via LIC.
- Efficiently *packs numerous triangular-textures* into a few rectangular-texture blocks stored in memory for fast texture retrieval at low memory cost.
- Maps each triangular texture onto the target triangle in rendering.



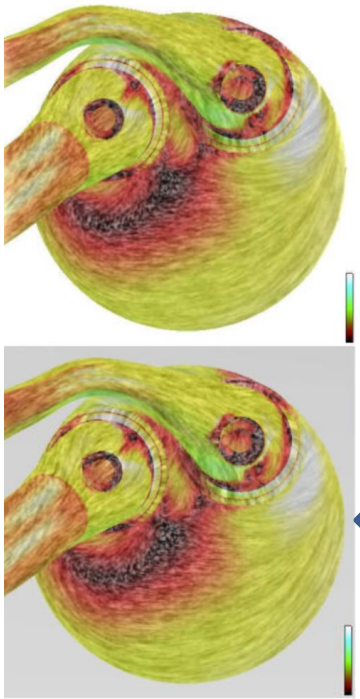
compute each texel value

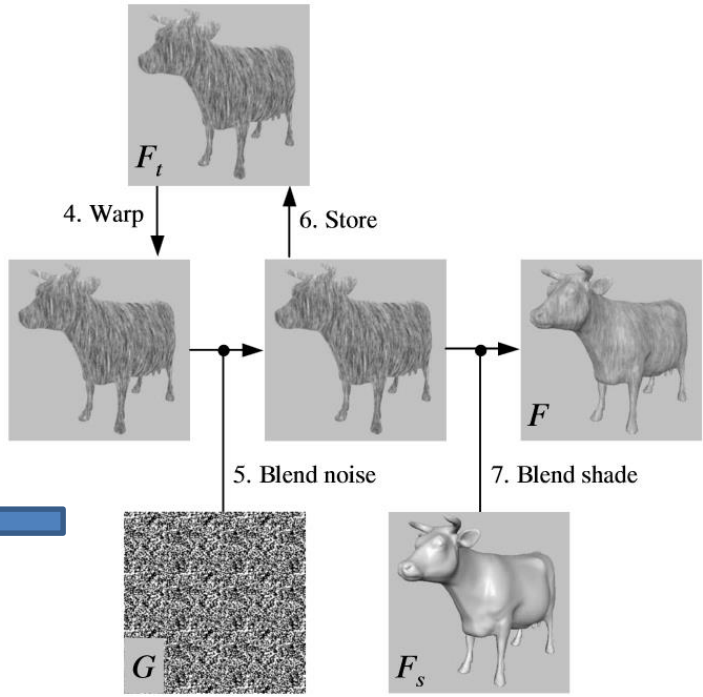


(Detlev Stalling, ZIB, Germany)

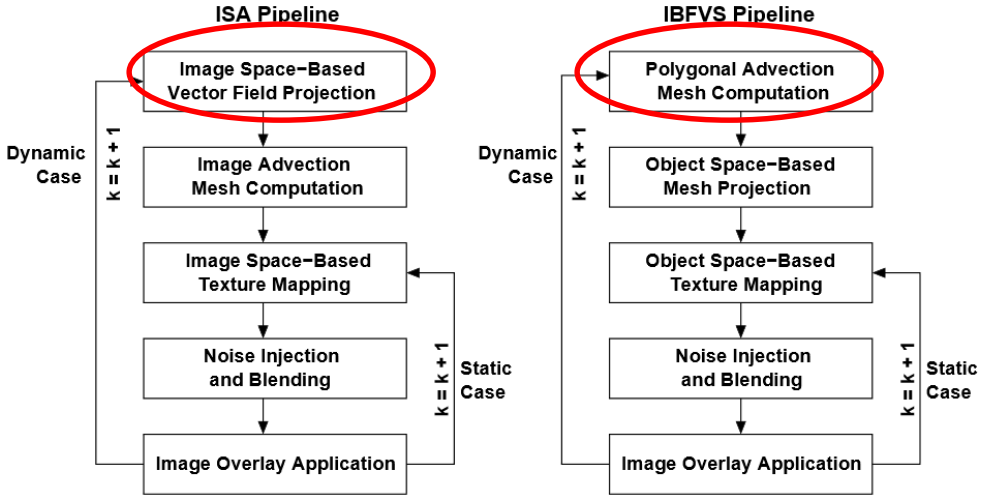
ISA vs. IBFVS

ISA



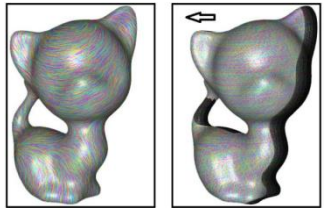


IBFVS

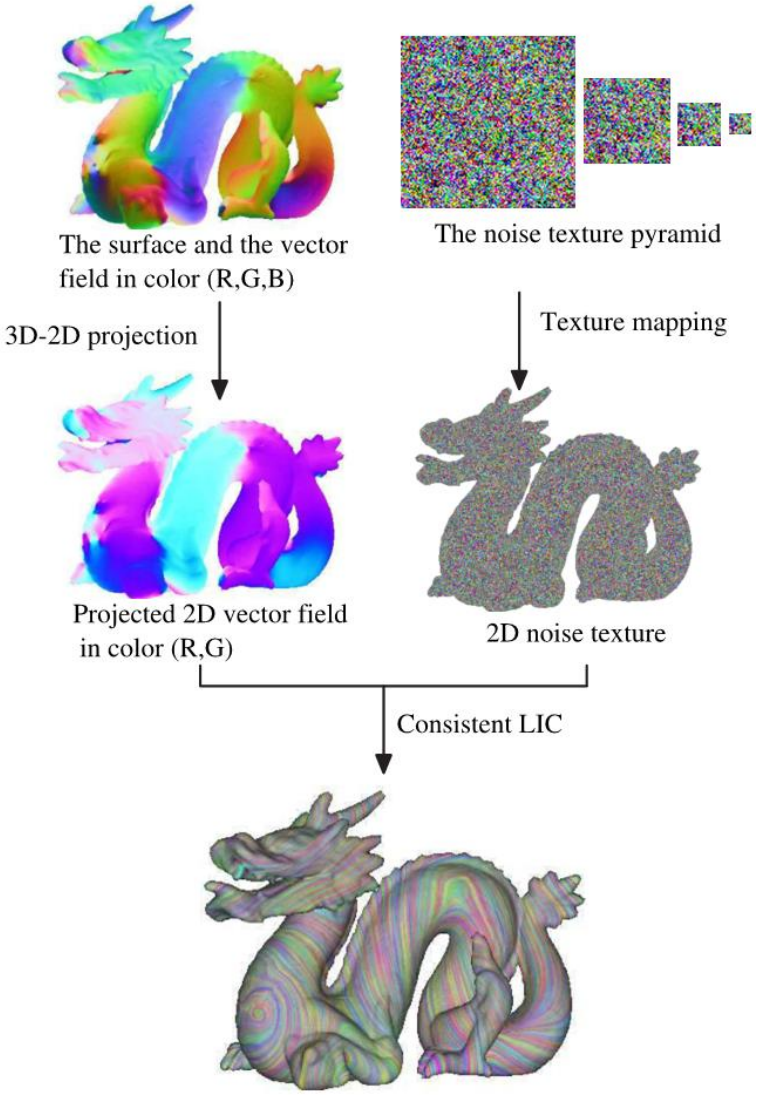


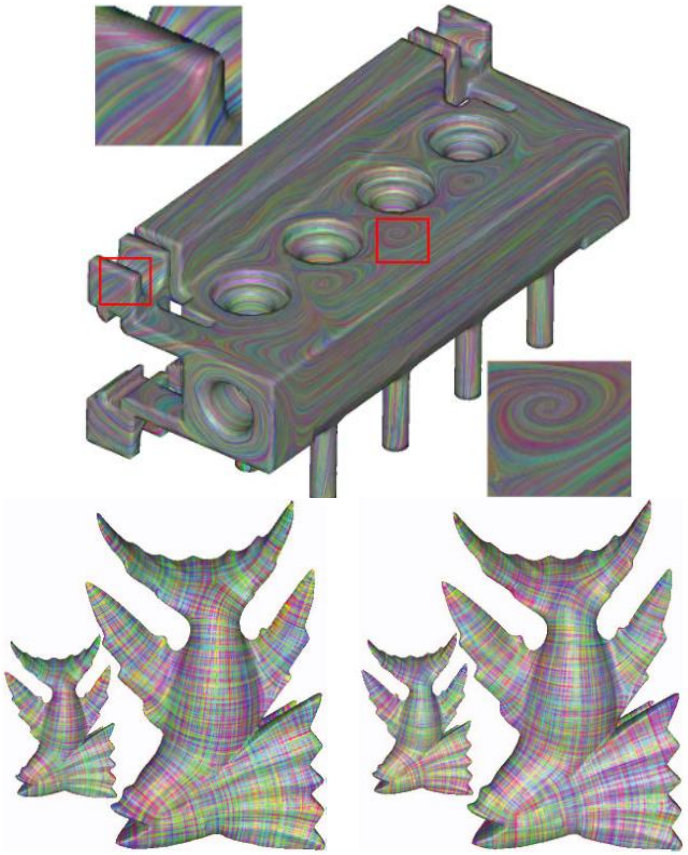
[Laramee et al. TVCG03]

Coherent Texture on Surfaces



Address the inconsistency of flow image when the view point is changed.





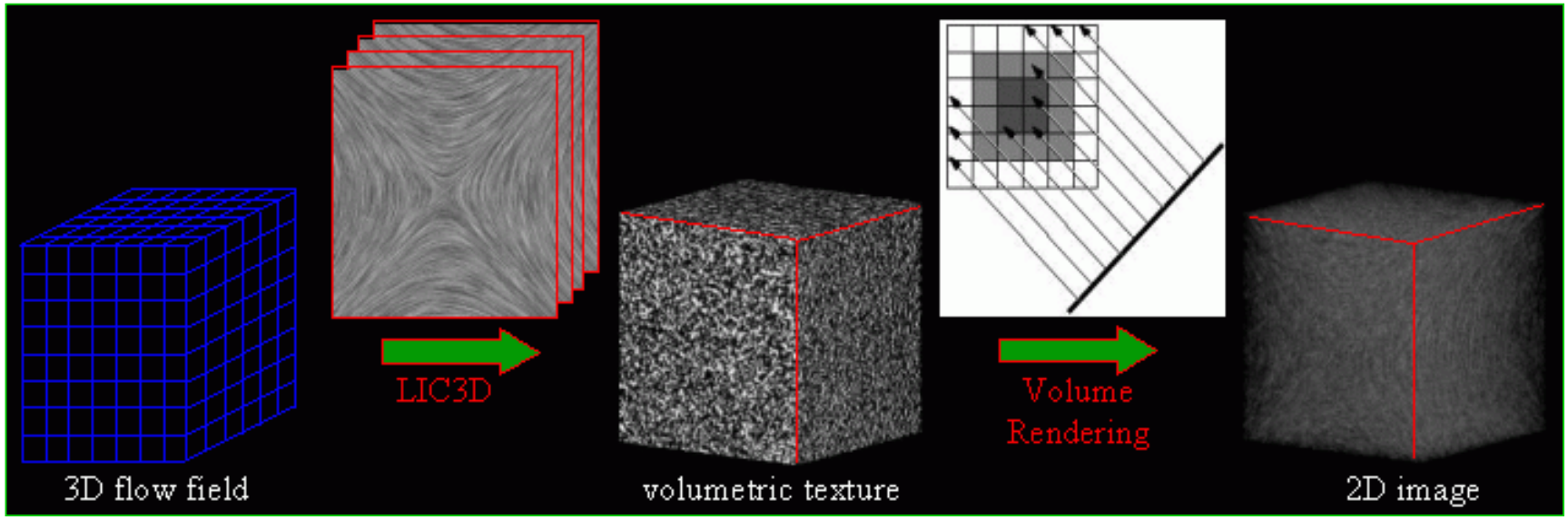
[Huang et al. TVCG13]

TEXTURE-BASED VISUALIZATION FOR 3D FLOW

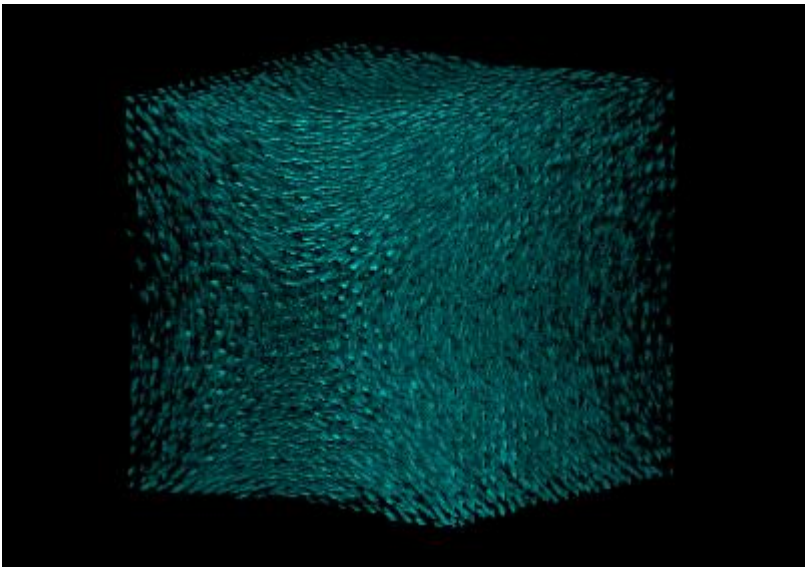
Line Integral Convolution — LIC Variants

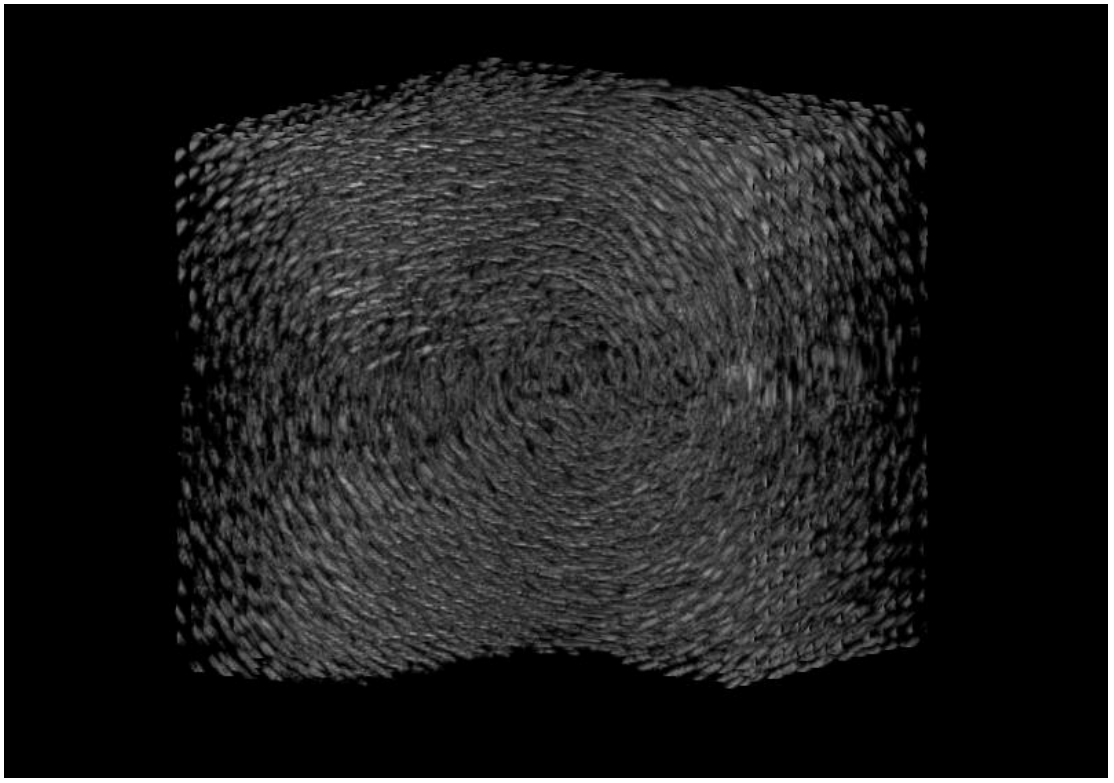
➢ Volume LIC

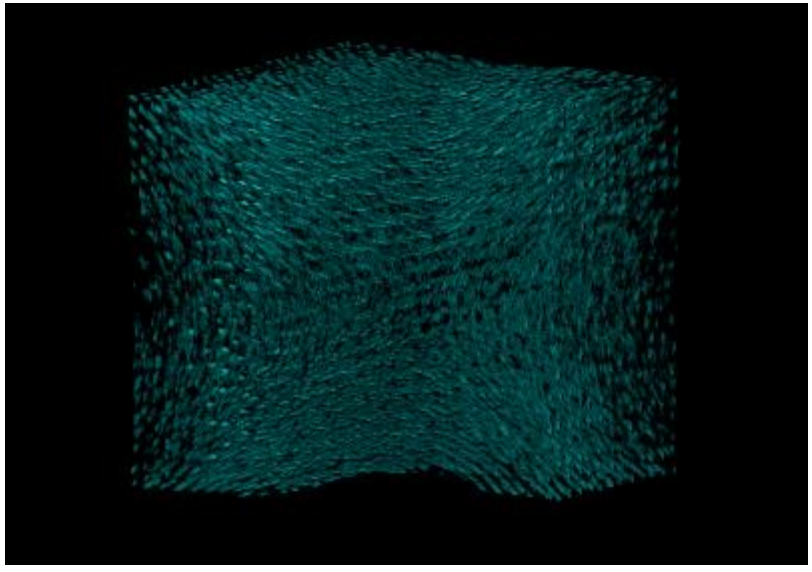
- ✧ Victoria Interrante and Chester Grosch (*IEEE Visualization 97*).
- ✧ A straightforward extension of LIC to 3D flow fields.
- ✧ Low-pass filters *volumetric noise* along 3D streamlines.
- ✧ Uses *volume rendering* to display resulting 3D LIC textures.
- ✧ Very time-consuming to generate 3D LIC textures.
- ✧ Texture values offer no useful guidance for transfer function design due to *lack of intrinsic physical info* that can be exploited to distinguish components.
- ⇨ Very challenging to clearly show *flow directions and interior structures through a dense texture volume.*

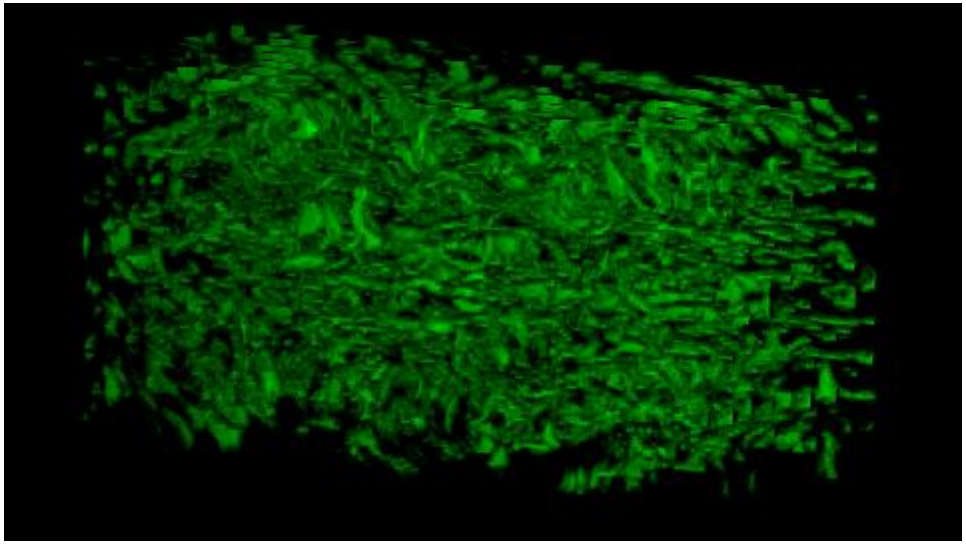


Sparse noise + Hybrid Hanning-Ramp kernel (Zhanping Liu and et al., *Journal of Image and Graphics 2001*)



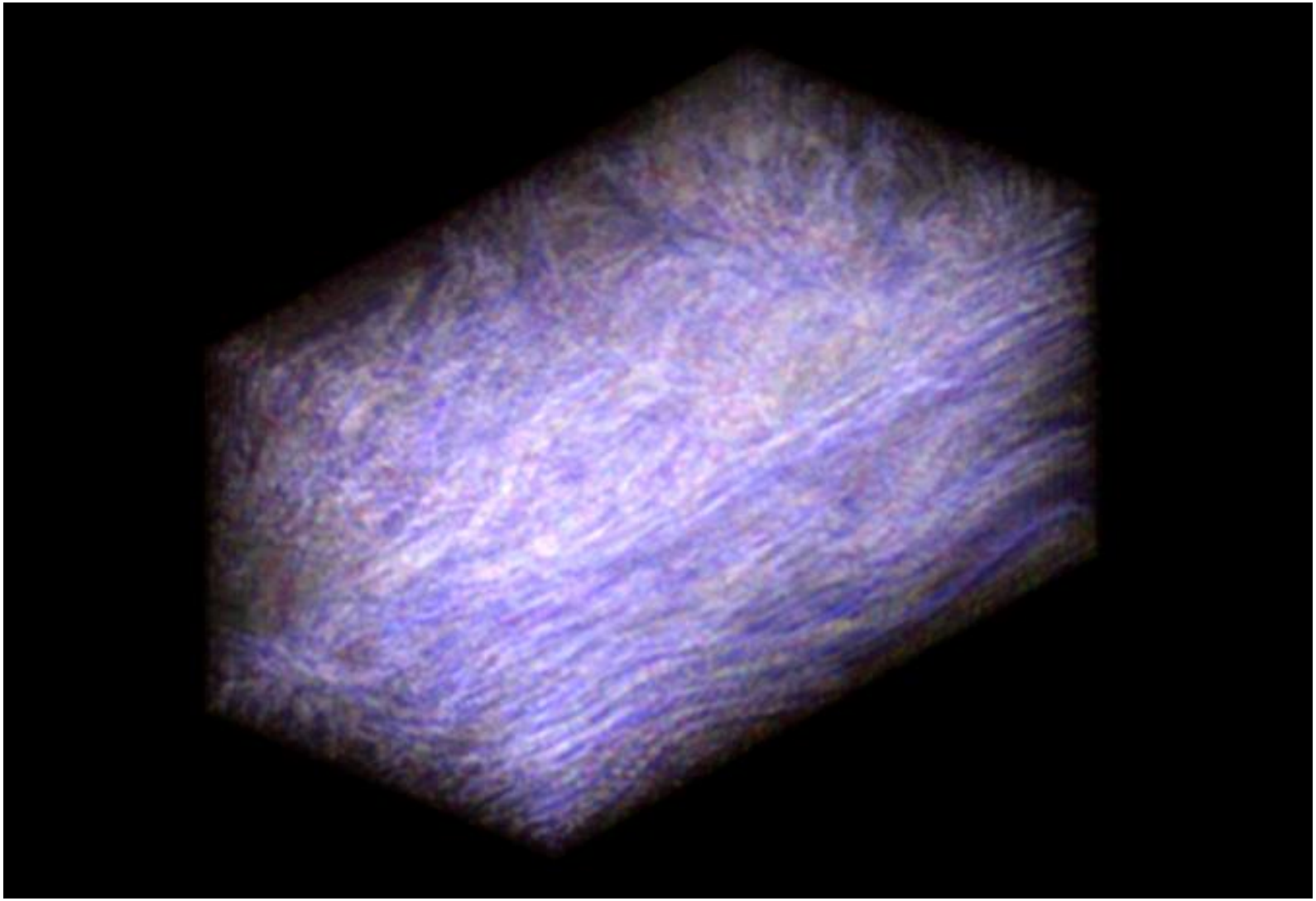


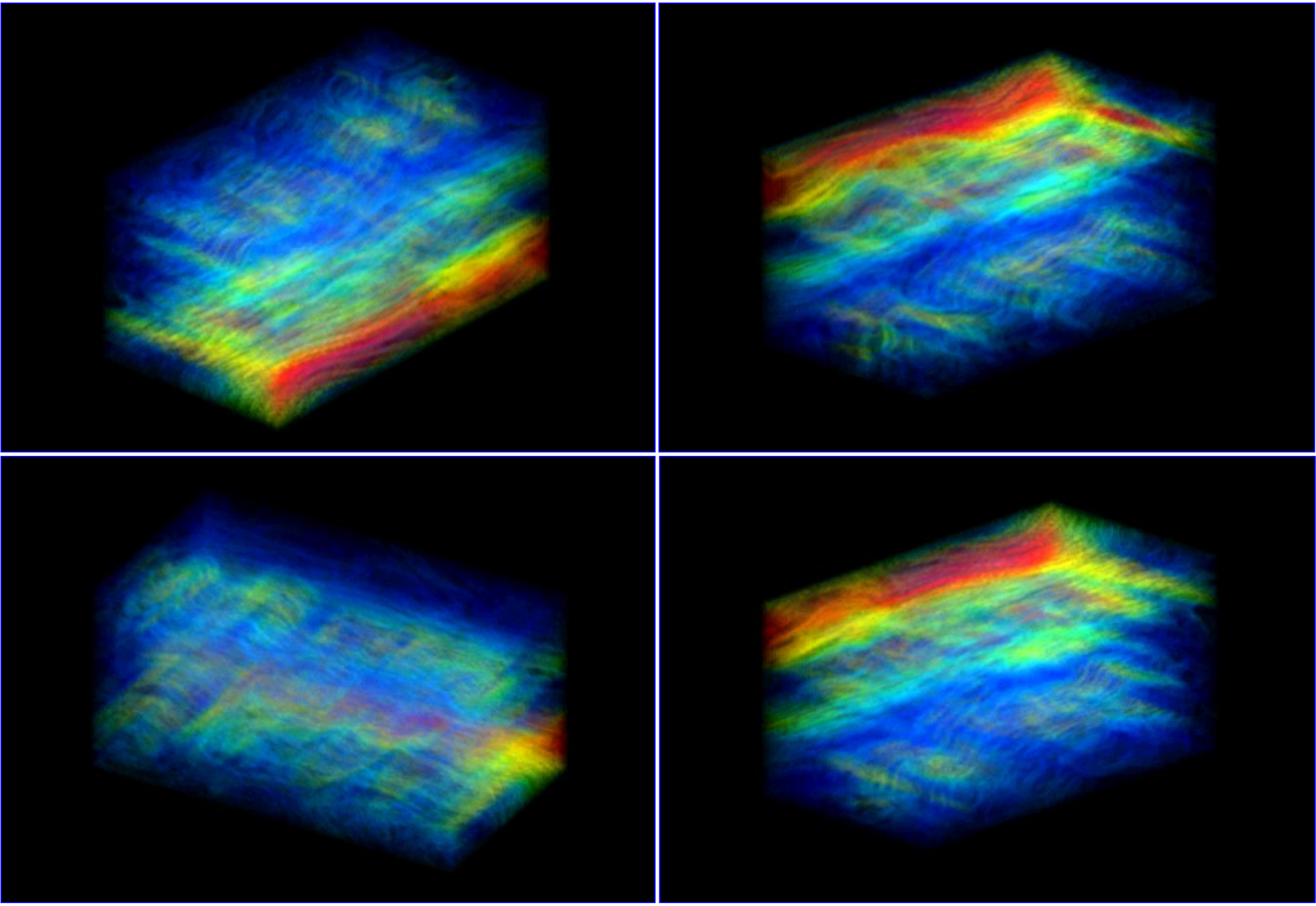




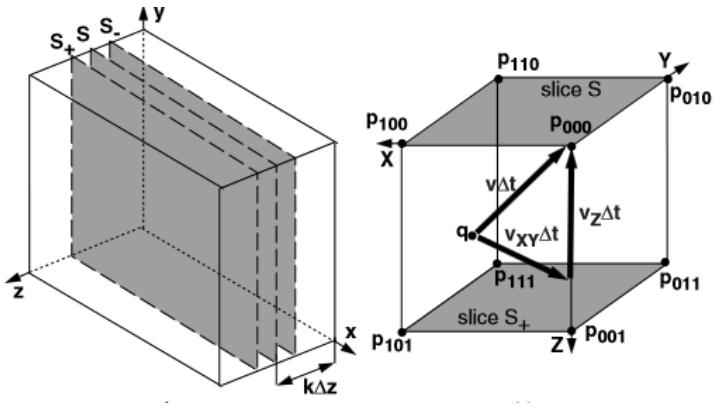
Unsteady Flow LIC — VAUFLIC

Image generated by using a texture-based transfer function

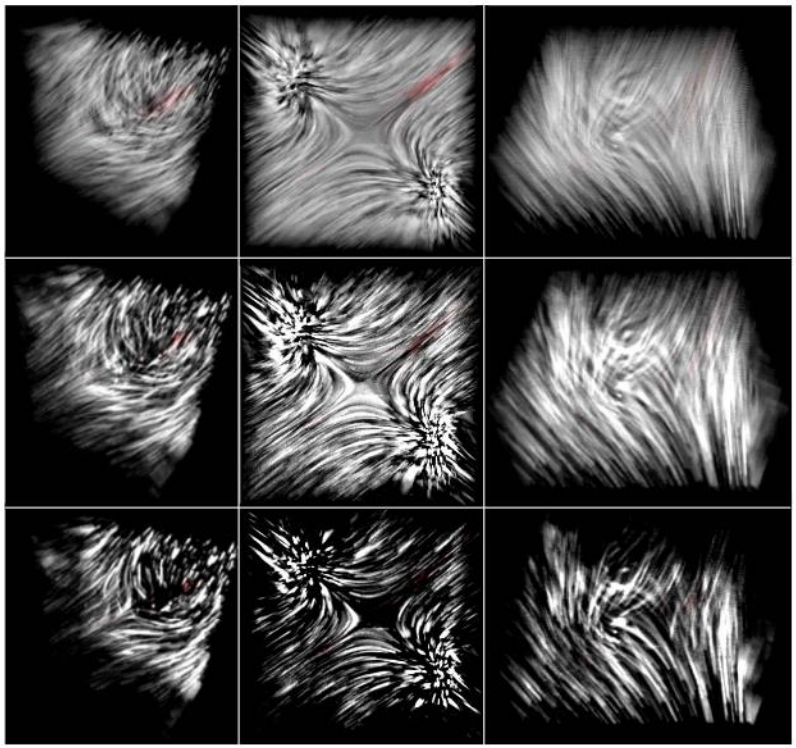


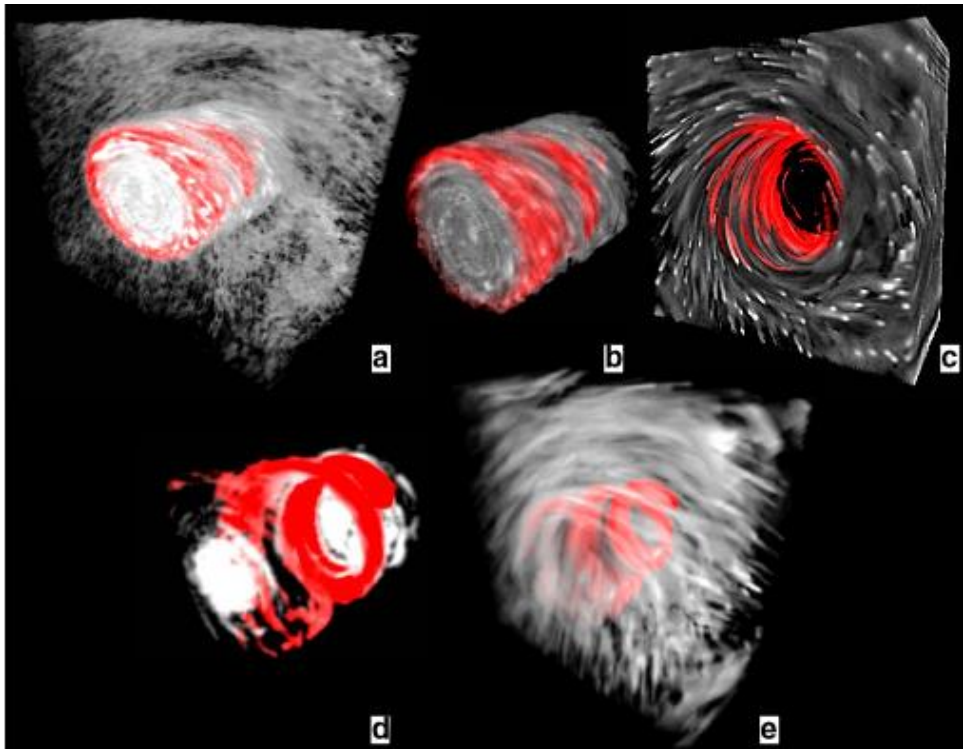


3D IBFV




```
for i = 0 to N-1
{
  if (i>0)                                                    (1)
    do 1D Z-axis advection from S_{i-1} to S_i
  if (i<N-1)                                                  (2)
    do 1D Z-axis advection from S_{i+1} to S_i
  do 2D IBFV-based advection in the slice S_i                 (3)
}
```





[Telea and van Wijk Vis03]

Acknowledgment

- Thanks for the materials from
  - Dr. Zhanping Liu, Kentucky State University
  - Dr. Robert Laramee, Swansea University, UK