EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Vector Field Visualization

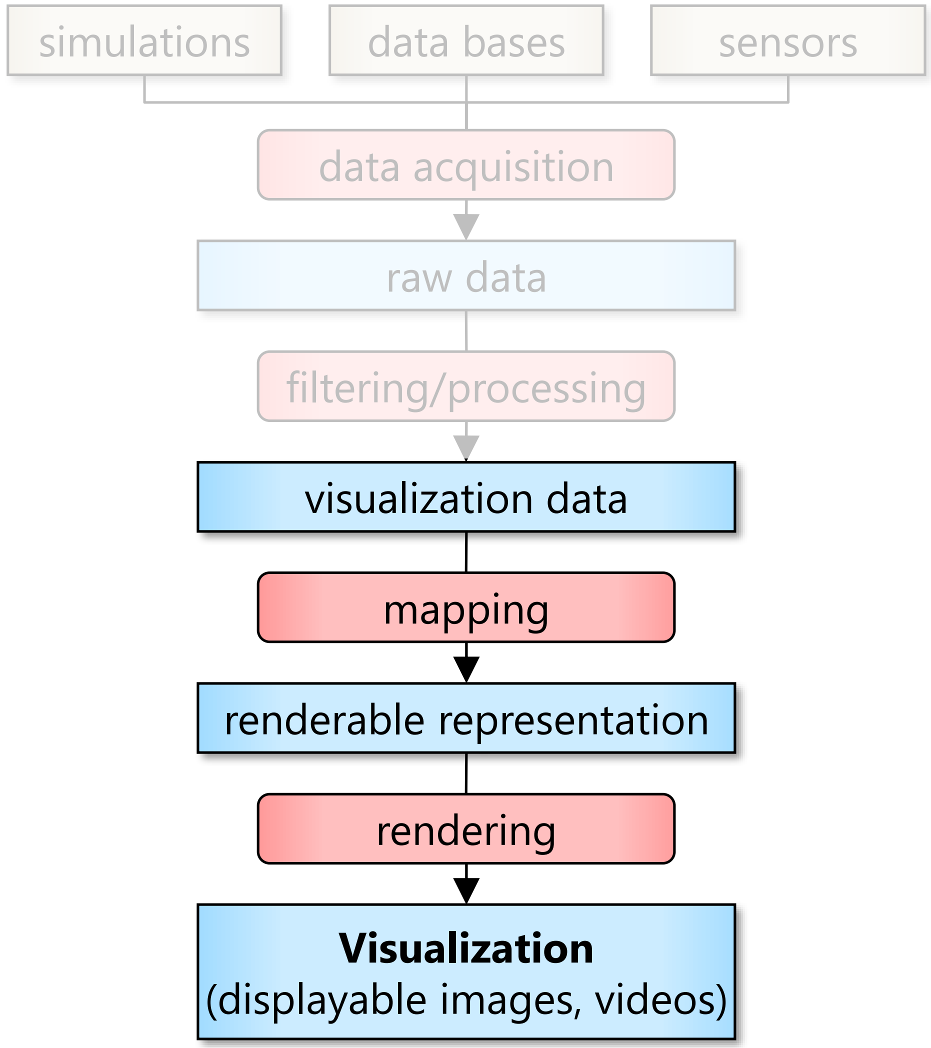
Scientific Visualization – Summer Semester 2021

Jun.-Prof. Dr. **Michael Krone**

Contents

- Vector calculus
- Characteristic lines
- Arrows and glyphs
- Particle tracing and mapping methods
- Numerical integration
- Particle tracing on grids
- Line integral convolution
- Texture advection
- Topology-based visualization
- 3D vector fields

Focus:
Second step of visualization pipeline



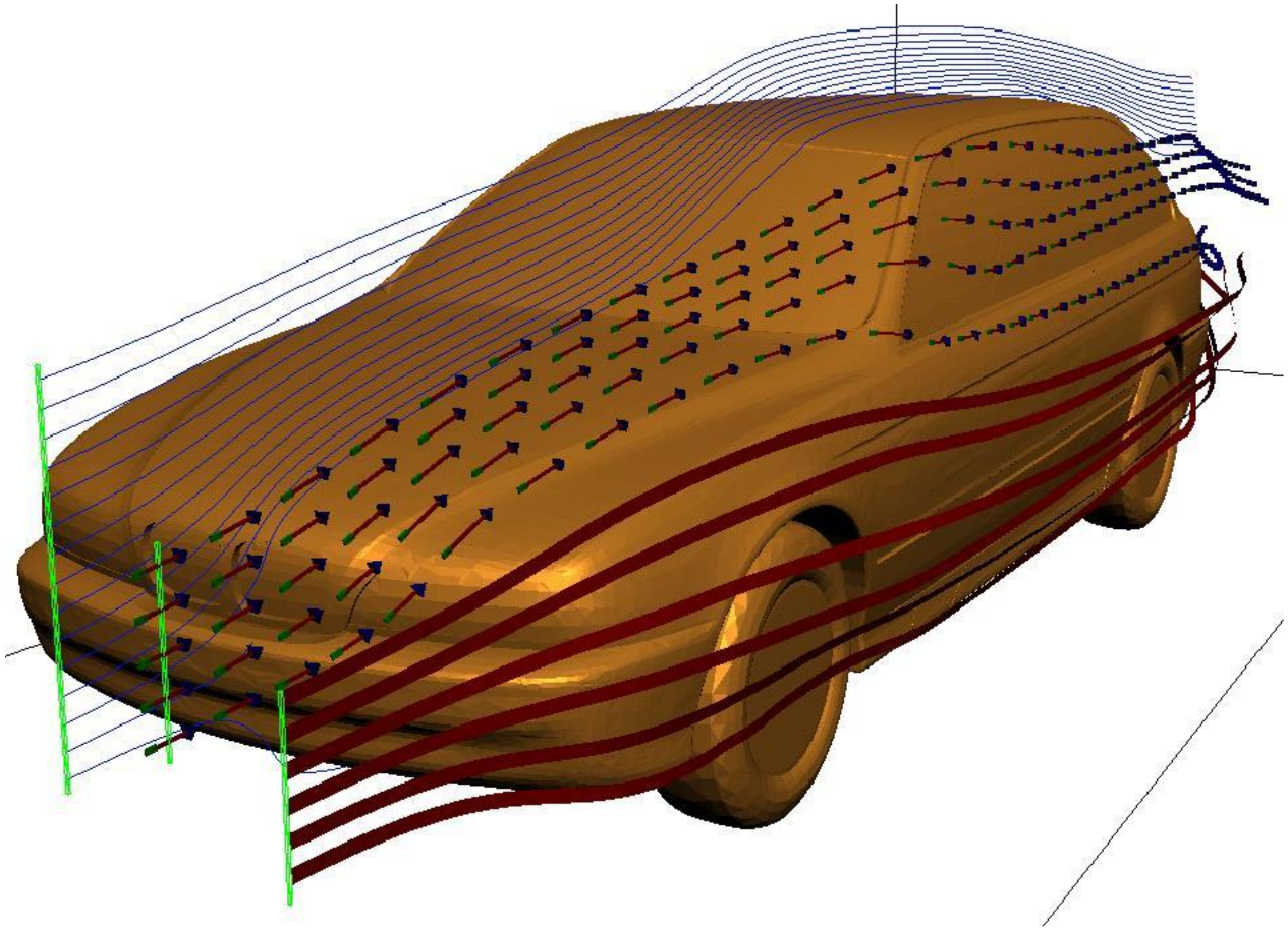
Problem Setting

- Vector field data
  - Represent direction and magnitude
  - Given by a m-tupel $(f_1, \dots, f_m)$ with $f_k = f_k(x_1, \dots, x_n)$, $m \geq 2$ and $1 \leq k \leq m$
  - Typically $m = n$ and $n = 2$ or $n = 3$
- Time-dependence: $f_k = f_k(x_1, \dots, x_n, t)$
- Often denoted as $\boldsymbol{u}(\boldsymbol{x}, t)$ with $\boldsymbol{u} = (u(x,y,z,t), v(x,y,z,t), w(x,y,z,t))$ and $\boldsymbol{x} = (x, y, z)$

Problem Setting

- Main application of vector field visualization is flow visualization
  - Motion of fluids (gas, liquid)
  - Geometric boundary conditions
  - Velocity (flow) field $\boldsymbol{u}(\boldsymbol{x}, t)$
  - Pressure p
  - Temperature T
  - Divergence $\nabla \cdot \boldsymbol{u}$ (or: $div\ \boldsymbol{u}$)
  - Vorticity $\nabla \times \boldsymbol{u}$ (or: $curl\ \boldsymbol{u}$, $rot\ \boldsymbol{u}$)
  - Density ρ
  - Conservation of mass, energy, and momentum
  - Navier-Stokes equations, CFD (Computational Fluid Dynamics)

Problem Setting



Flow visualization
based on CFD data

Problem Setting

- Flow visualization – classification
  - Dimension (2D or 3D)
  - Time-dependency: stationary (steady) vs. instationary (unsteady, transient)
  - Grid type

- In most cases numerical methods required for flow visualization

Vector Calculus

- Review of basics of vector calculus
- Deals with vector fields and various kinds of derivatives
- Flat (Cartesian) manifolds only
- Cartesian coordinates only
- 3D only

Vector Calculus

- Scalar function $f(\boldsymbol{x},t)$
- Gradient $$\nabla f(\mathbf{x},t) = \begin{pmatrix} \frac{\partial}{\partial x} f(\mathbf{x},t) \\ \frac{\partial}{\partial y} f(\mathbf{x},t) \\ \frac{\partial}{\partial z} f(\mathbf{x},t) \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} f(\mathbf{x},t)$$
- Gradient vector points into direction of maximum increase of $f(\boldsymbol{x},t)$
- Laplacian $$\Delta f(\mathbf{x},t) = \nabla \cdot \nabla f(\mathbf{x},t)$$ $$= \frac{\partial^2}{\partial x^2} f(\mathbf{x},t) + \frac{\partial^2}{\partial y^2} f(\mathbf{x},t) + \frac{\partial^2}{\partial z^2} f(\mathbf{x},t)$$
  - Laplacian of a scalar is a scalar (of a vector is a vector)

Vector Calculus

- Vector function $\boldsymbol{u}(\boldsymbol{x}, t)$
- Jacobian matrix ("gradient tensor", "velocity gradient")

$$\mathbf{J} = \nabla \mathbf{u}(\mathbf{x}, t) = \begin{pmatrix} \frac{\partial}{\partial x} u & \frac{\partial}{\partial y} u & \frac{\partial}{\partial z} u \\ \frac{\partial}{\partial x} v & \frac{\partial}{\partial y} v & \frac{\partial}{\partial z} v \\ \frac{\partial}{\partial x} w & \frac{\partial}{\partial y} w & \frac{\partial}{\partial z} w \end{pmatrix}$$

- Divergence

$$\operatorname{div} \mathbf{u}(\mathbf{x}, t) = \nabla \cdot \mathbf{u}(\mathbf{x}, t) = \frac{\partial}{\partial x} u(\mathbf{x}, t) + \frac{\partial}{\partial y} v(\mathbf{x}, t) + \frac{\partial}{\partial z} w(\mathbf{x}, t)$$

$$= \operatorname{tr}(\mathbf{J}) \quad \text{(trace of } \mathbf{J}\text{)}$$

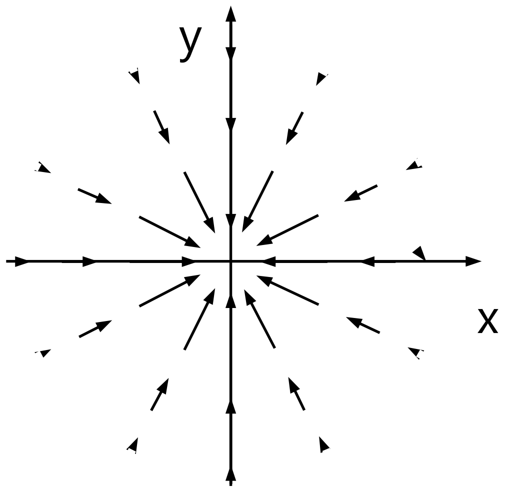
  - Divergence is a scalar

Vector Calculus

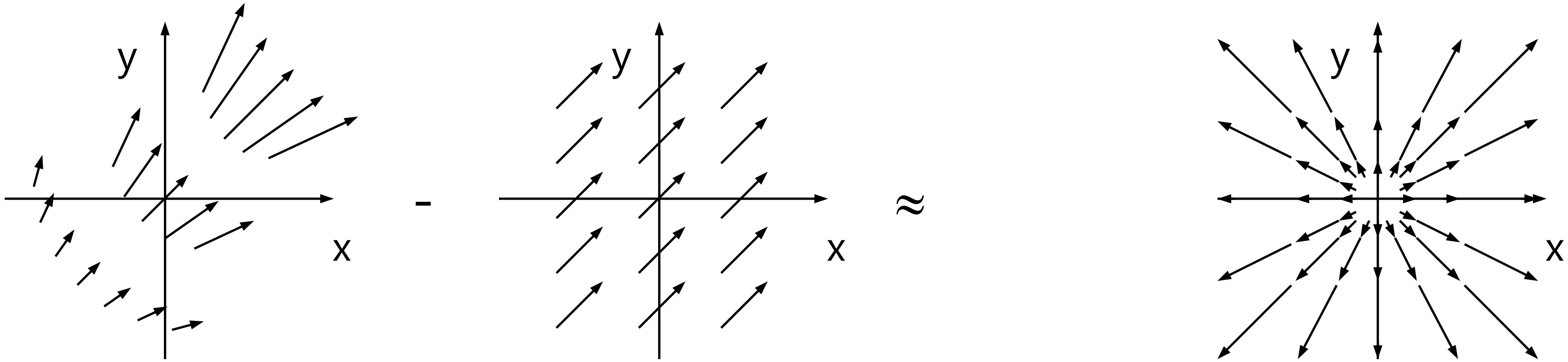
- Properties of divergence:
  - $div\,\boldsymbol{u}$ is a scalar
  - $div\,\boldsymbol{u}(\boldsymbol{x_0}) > 0$: **u** has a "source" in $\mathbf{x}_0$
  - $div\,\boldsymbol{u}(\boldsymbol{x_0}) < 0$: **u** has a "sink" in $\mathbf{x}_0$

  → Describes relative flow into/out of a region
  - $div\,\boldsymbol{u}$ consists of derivatives only

  → $div\,\boldsymbol{u}$ is invariant under addition/subtraction of uniform field:



sink

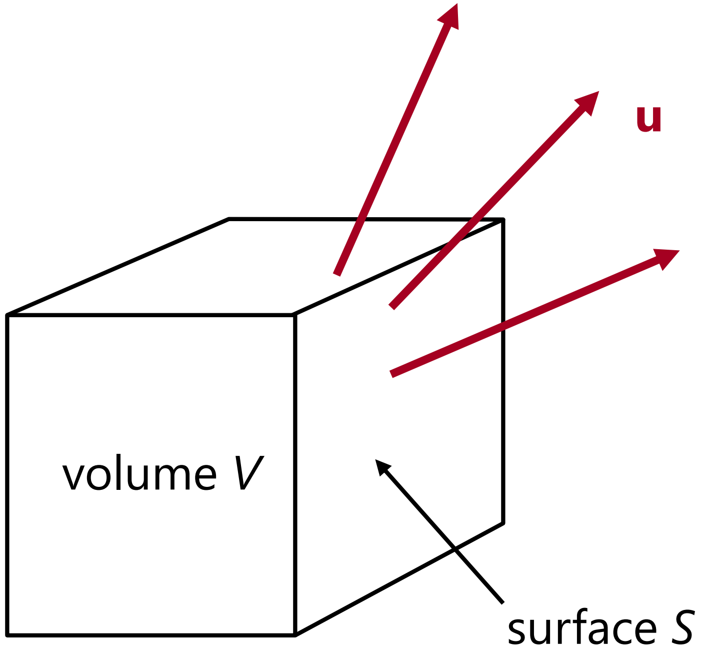


source

Vector Calculus

- Gauss theorem (divergence theorem)

$$\int_V \nabla \cdot \mathbf{u} dV = \oint_S \mathbf{u} \cdot d\mathbf{A}$$ (surface element $d\mathbf{A}$ points outward V)



Vector Calculus

- Continuity equation
  - Flow of mass into a volume *V* with surface *S*

$$-\oint_S \rho \mathbf{u} \cdot d\mathbf{A}$$

$\rho\boldsymbol{u}$: momentum density = mass flux

  - Change of mass inside the volume

$$\frac{\partial}{\partial t}\int_V \rho dV = \int_V \frac{\partial \rho}{\partial t} dV$$

  - Conservation of mass

$$\int_V \frac{\partial \rho}{\partial t} dV = -\oint_S \rho \mathbf{u} \cdot d\mathbf{A}$$

Vector Calculus

- Continuity equation *(cont.)*
  - Application of Gauss theorem

$$\int_V \frac{\partial \rho}{\partial t} dV + \oint_S \rho \mathbf{u} \cdot d\mathbf{A} = \int_V \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right) dV = 0$$

  - Above equation must be met for any volume element
  - Yields

$$\boxed{\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0}$$

continuity equation
in differential form

current $\mathbf{j} = \rho \mathbf{u}$

Vector Calculus

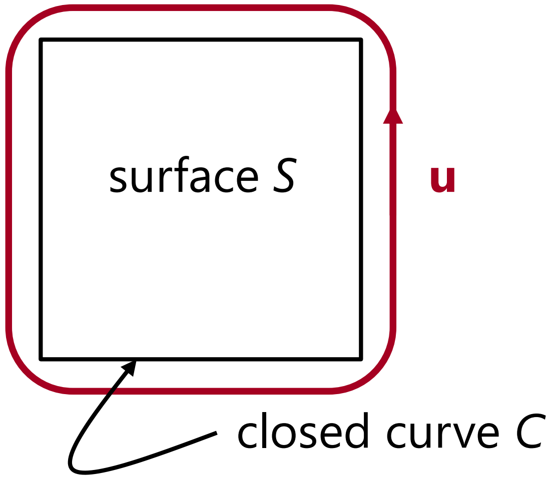
- Curl

$$\boldsymbol{\omega}(\mathbf{x},t) = \operatorname{curl}\mathbf{u}(\mathbf{x},t) = \nabla\times\mathbf{u}(\mathbf{x},t) = \begin{pmatrix} \frac{\partial}{\partial y}w(\mathbf{x},t) - \frac{\partial}{\partial z}v(\mathbf{x},t) \\ \frac{\partial}{\partial z}u(\mathbf{x},t) - \frac{\partial}{\partial x}w(\mathbf{x},t) \\ \frac{\partial}{\partial x}v(\mathbf{x},t) - \frac{\partial}{\partial y}u(\mathbf{x},t) \end{pmatrix}$$

- Stokes theorem

$$\int_S \nabla\times\mathbf{u}\cdot d\mathbf{A} = \oint_C \mathbf{u}\cdot d\mathbf{s} = \Gamma$$

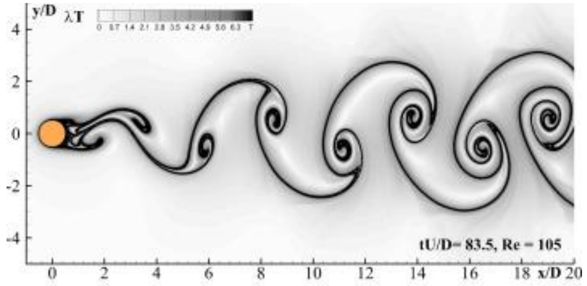
$d\mathbf{s}$: line element along C
Γ: circulation

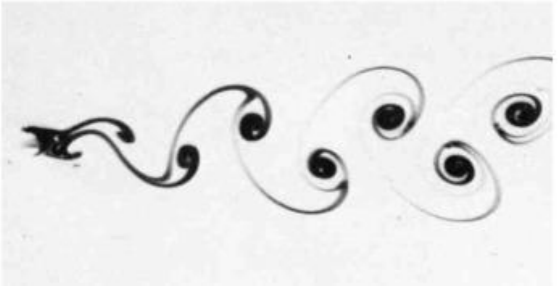


Vector Calculus

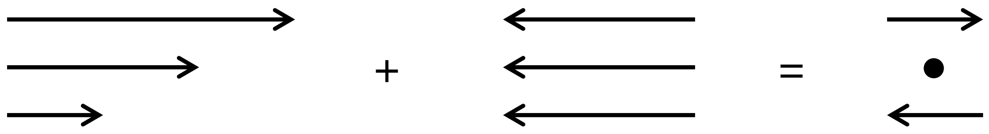
- Properties of curl:
  - In CFD, often called vorticity
  - Orientation of $\boldsymbol{\omega}$ represents right-handed axis of "local rotation"
  - Angular velocity of close particle is $\|\boldsymbol{\omega}\| / 2$
  - $\boldsymbol{\omega}$ consists of derivatives only

  $\rightarrow$ $\boldsymbol{\omega}$ is invariant under addition/subtraction of uniform field
  - $\boldsymbol{\omega} \neq \mathbf{0}$ does not necessarily imply rotational motion!





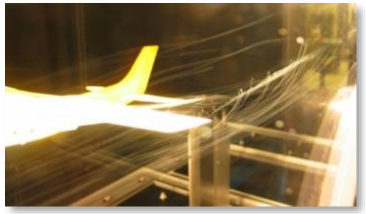
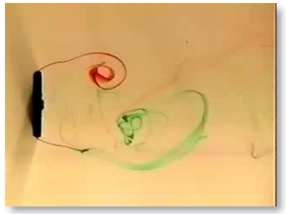
shear flow:



  - Shear flow exhibits straight motion but nonzero vorticity

Characteristic Lines

- Streamlines:
  - Tangential to the vector field (at constant time)
  - “Magnetic field lines”
- Path lines:
  - Trajectories of massless particles in the flow
  - “Long time exposure of particles”
- Streak lines:
  - Set of particles started at same position but different times
  - “Trace of dye (smoke) released at fixed position”
- Time lines:
  - Set of particles started on a seeding curve at same time
  - “Chain of bubbles produced by electrolysis by a voltage pulse on a wire”


Characteristic Lines

- Streamlines
  - Tangential to the vector field
  - Vector field at an arbitrary, yet fixed time *t*
  - Streamline is a solution to the initial value problem of an ordinary differential equation:

$$\underbrace{\boldsymbol{L}(0) = x_0\,,}_{\substack{\text{initial value} \\ \text{(seed point } \boldsymbol{x_0}\text{)}}} \qquad \underbrace{\frac{d\big(\boldsymbol{L}(s)\big)}{ds} \times \boldsymbol{u}(\boldsymbol{L}(s), t) = 0\,,}_{\substack{\text{Streamline } \boldsymbol{L}(s) \\ \text{tangential to } \boldsymbol{u}}} \qquad \underbrace{\frac{d\big(\boldsymbol{L}(s)\big)}{ds} = \boldsymbol{u}(\boldsymbol{L}(s), t)}_{\substack{\text{Ordinary Differential Equation} \\ \text{(ODE)}}}$$

  - Streamline is curve ***L****(s)* with the parameter *s* (arc length of the curve)

Characteristic Lines

- Path lines
  - Trajectories of massless particles in the flow
  - Vector field can be time-dependent
  - Path line is a solution to the initial value problem of an ordinary differential equation:

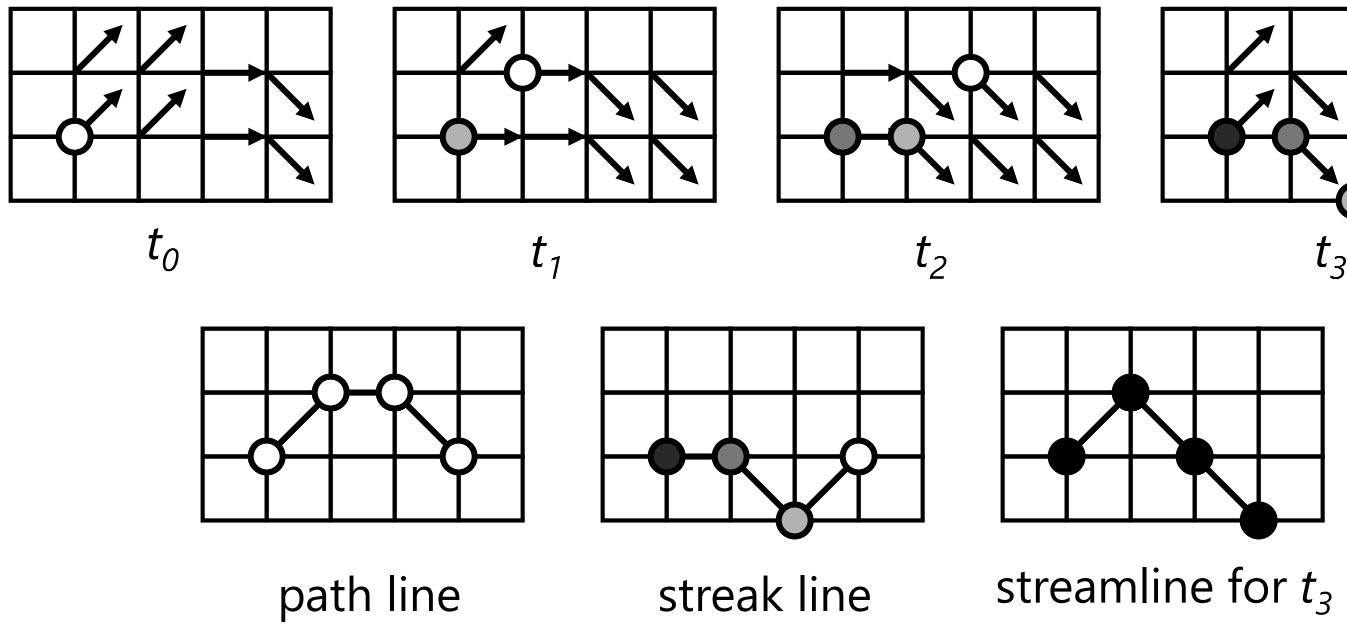
$$\underbrace{\boldsymbol{L}(t_0) = x_0\,,}_{\text{seed point } \boldsymbol{x_0} \text{ at time } t_0} \quad \underbrace{\frac{d\big(\boldsymbol{L}(t)\big)}{dt} \times \boldsymbol{u}(\boldsymbol{L}(t), t) = 0\,,}_{\text{Streamline } \boldsymbol{L}(t) \text{ tangential to } \boldsymbol{u} \text{ at time } t} \quad \frac{d\big(\boldsymbol{L}(t)\big)}{dt} = \boldsymbol{u}(\boldsymbol{L}(t), t)$$

Characteristic Lines

- Streak lines
  - Trace of dye that is released into the flow at a fixed position
  - Connect all particles that passed through a certain position
  - Solve initial value problem for each particle
- Time lines (material lines)
  - Propagation of a line of massless elements in time
  - Idea: “consists” of many point-like particles that are traced
  - Connect particles that were released simultaneously
  - Solve initial value problem for each particle
- Stream-, Path-, Streak-, and Time-Surfaces
  - Sets of characteristic lines started at higher-dimensional seeding structure

Characteristic Lines

- Comparison of path lines, streak lines, and streamlines



- Path lines, streak lines, and streamlines are identical for steady flows

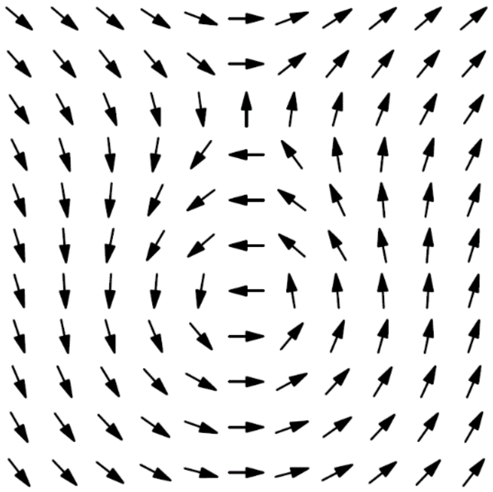
Lagrangian vs. Eulerian

- Difference between Eulerian and Lagrangian point of view
- Lagrangian:
  - Focus on individual particles
  - Can be identified
  - Attached are position, velocity, and other properties
  - Explicit position
  - Standard approach for particle tracing
- Eulerian:
  - Focus on domain
  - No individual particles
  - Properties given on a grid
  - Position of particles is implicit

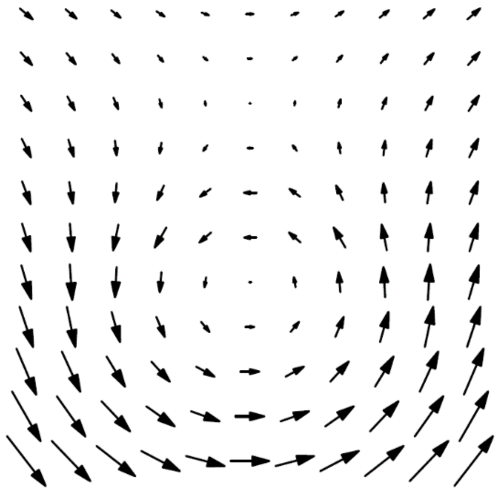
Visualization: Arrows or Glyphs

- Visualize **local** features of the vector field:
  - Vector itself (vorticity, Laplacian)
  - Additional data: temperature, pressure, etc.
- Important elements of a vector:
  - Direction
  - Magnitude
  - Not: components of a vector
- Approaches:
  - Arrow plots
  - Glyphs

→Direct mapping

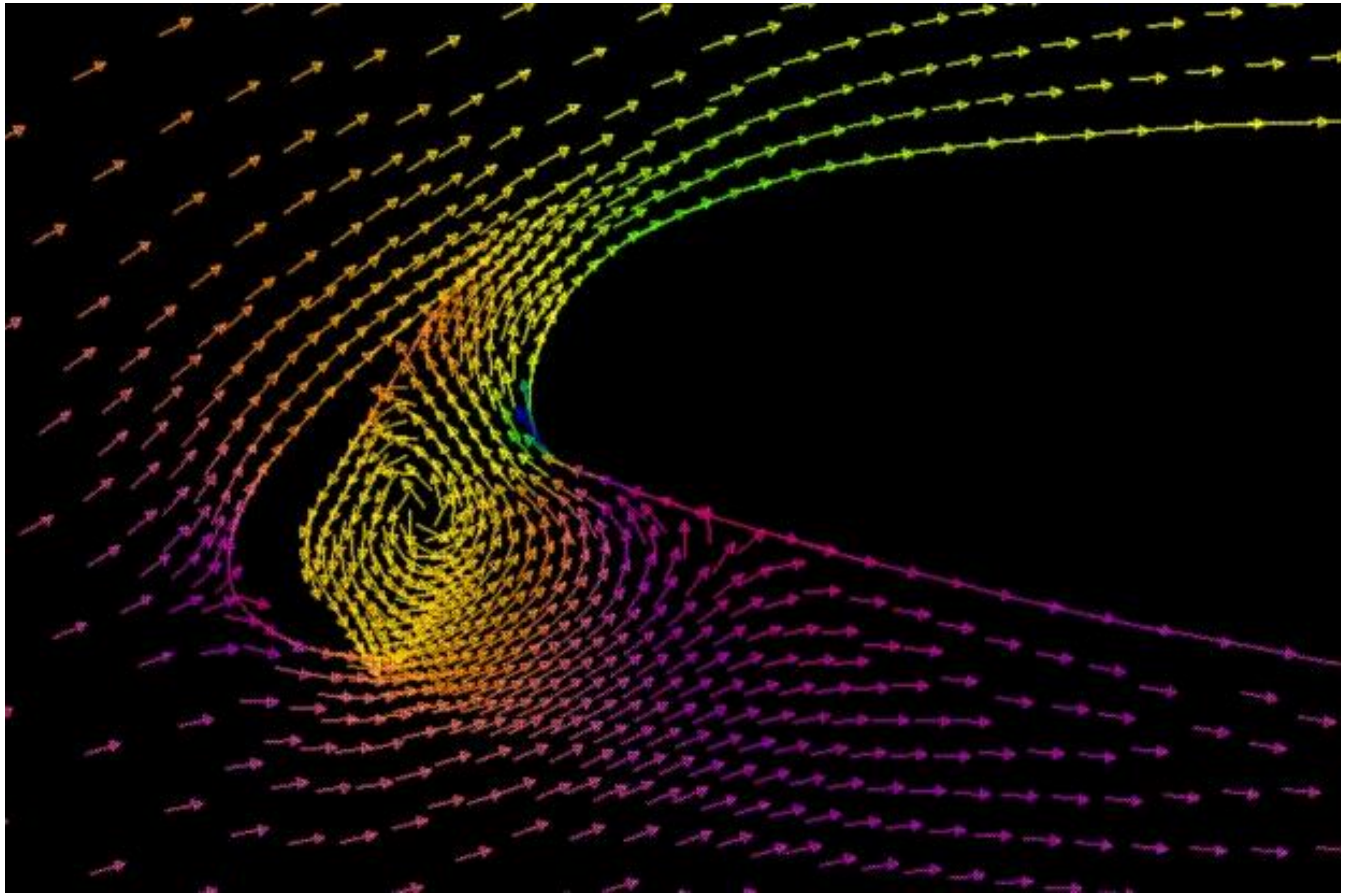


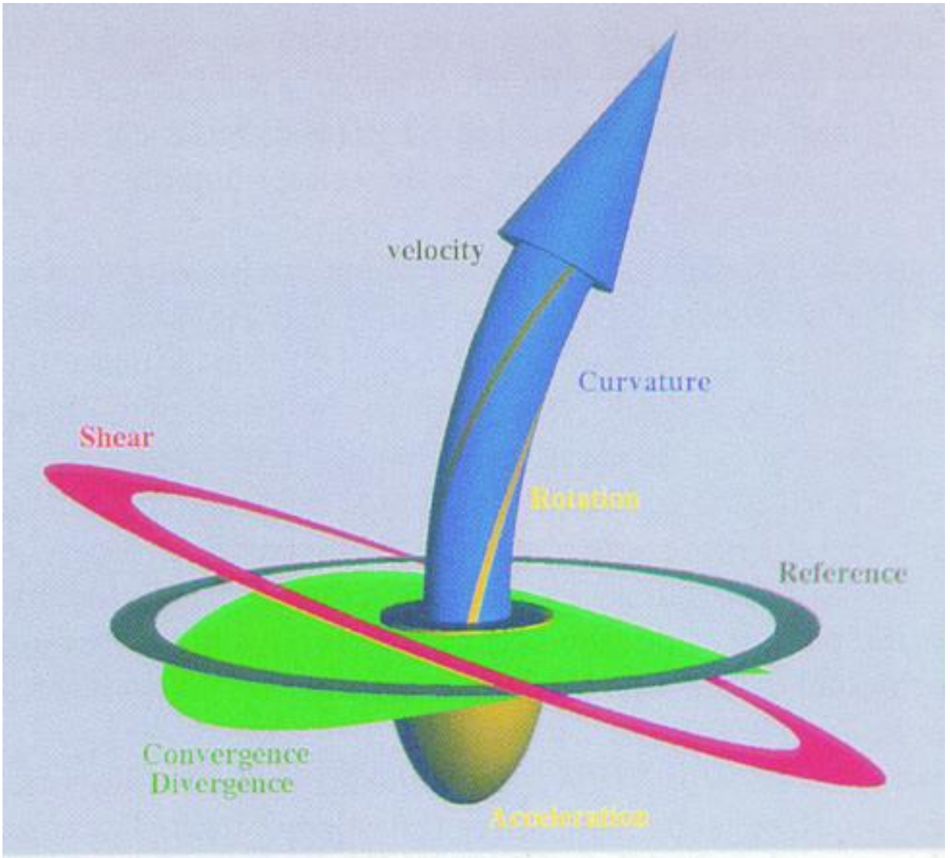
Direction of vector field
(Orientation)



Direction of vector field
+ Magnitude
→ Length (+width) of arrows
Alternative: Color coding

Visualization: Arrows or Glyphs





Glyph that visualizes the Jacobian of a flow field [de Leeuw and van Wijk, 93].

Arrows and Glyphs

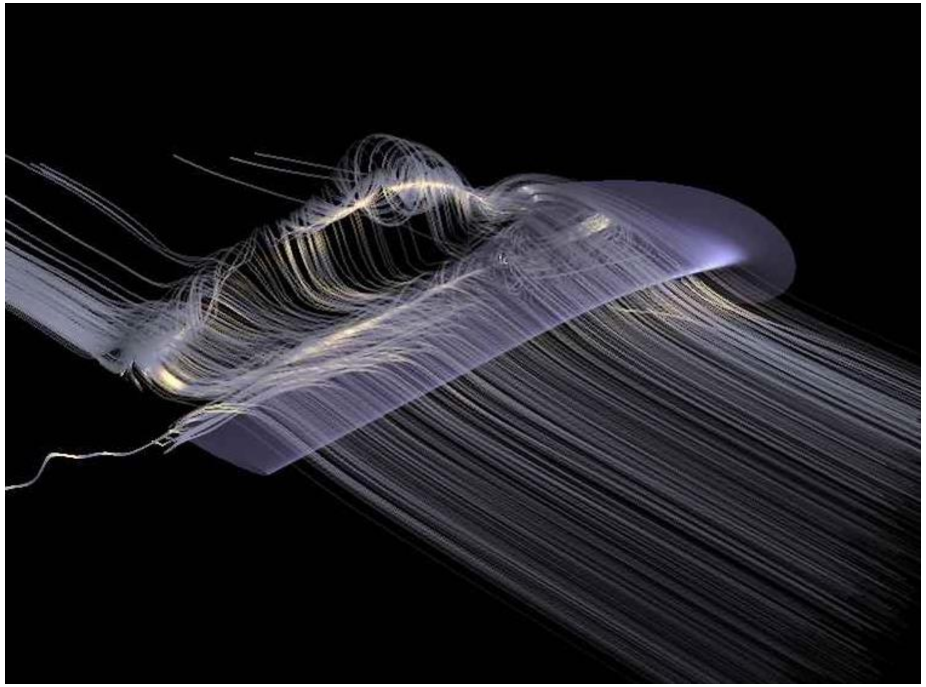
- Advantages and disadvantages of glyphs and arrows:
  - + Simple
  - + 3D effects
  - - Inherent occlusion effects
  - - Poor results if magnitude of velocity changes rapidly
    (Use arrows of constant length and color code magnitude)

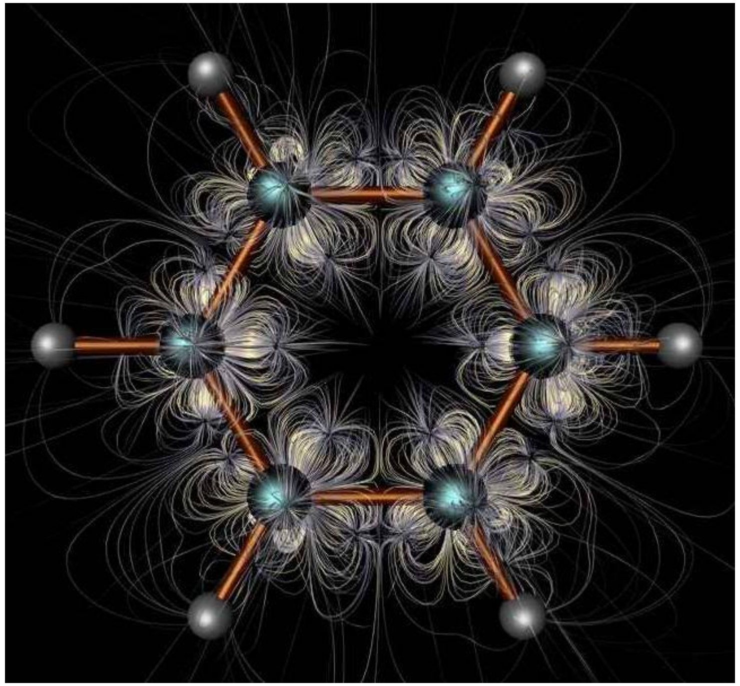
Mapping Methods Based on Particle Tracing

- Basic idea: trace particles
- Characteristic lines
- Mapping approaches:
  - Lines
  - Surfaces
  - Individual particles
  - Texture
  - Sometimes animated
- Density of visual representation
  - Sparse = only a few visual patterns (e.g., only a few streamlines)
  - Dense = complete coverage of the domain by visual structures

Mapping Methods Based on Particle Tracing

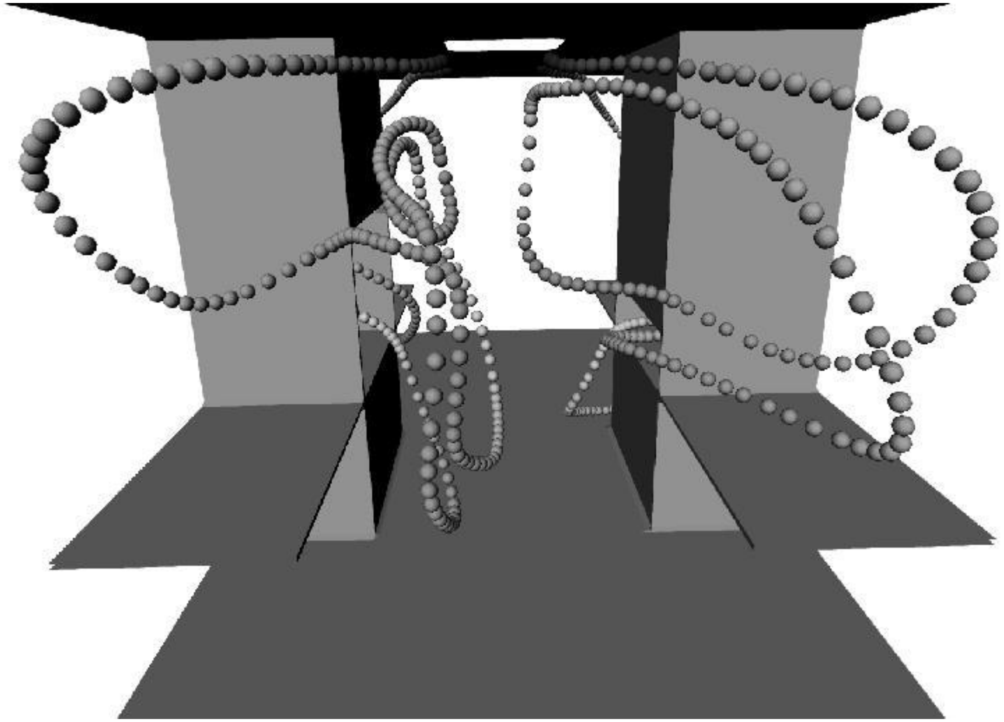
- Path lines
  - Improved perception by illuminated streamlines shading model

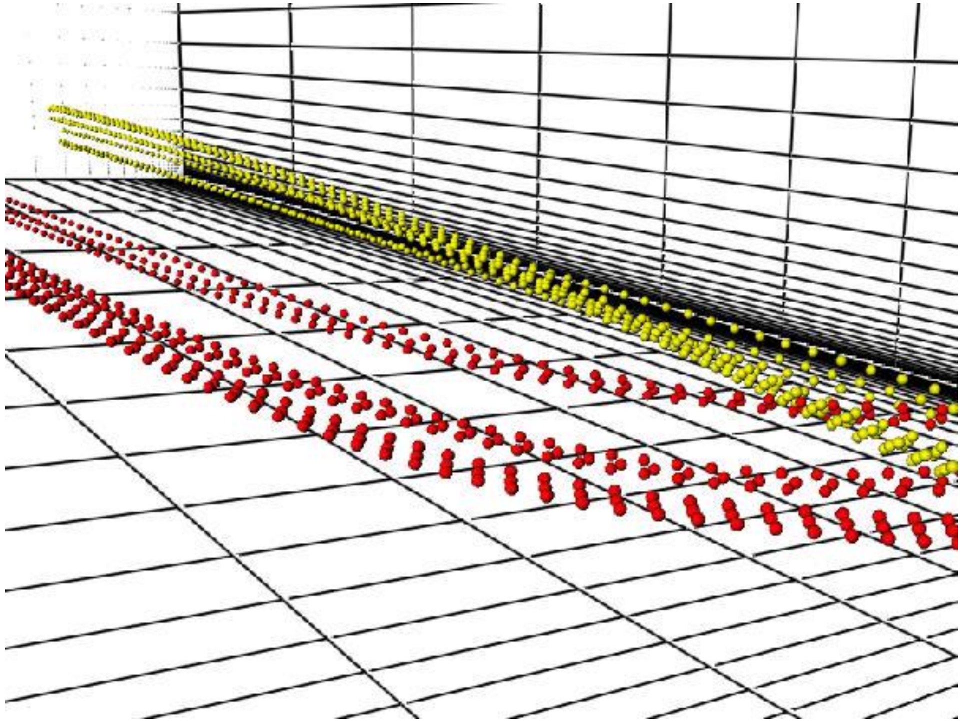




Mapping Methods Based on Particle Tracing

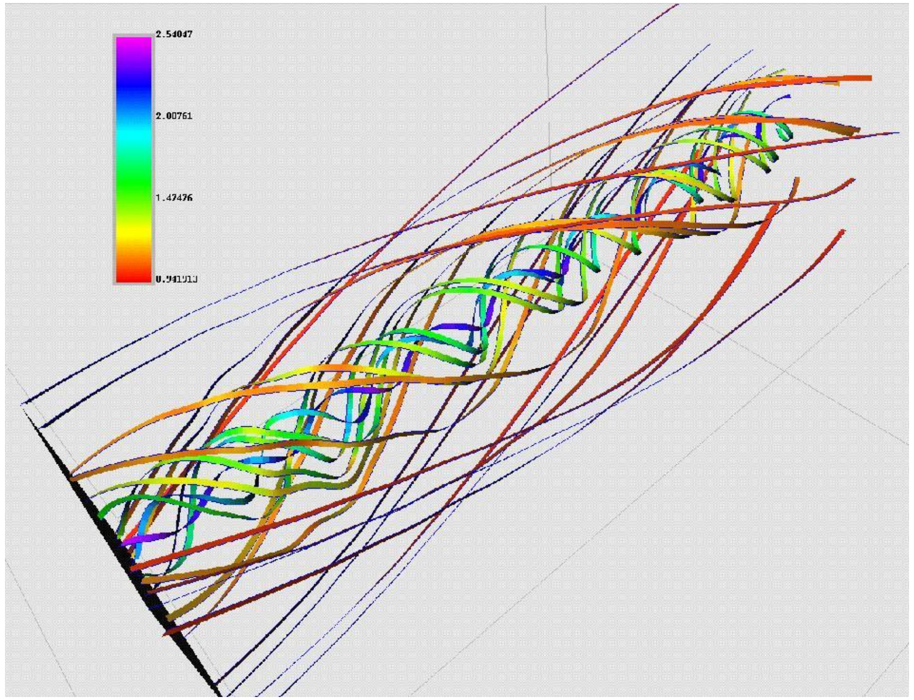
- Stream balls
  - Encode additional scalar value by radius
  - Problems: perspective projection, direction/orientation not visible

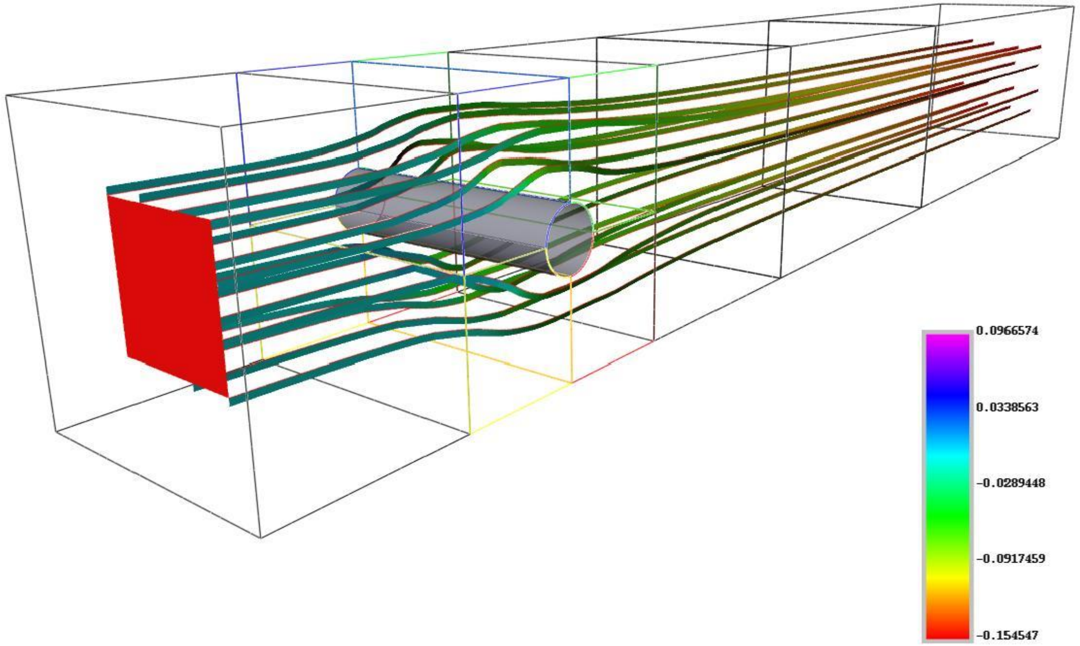




Mapping Methods Based on Particle Tracing

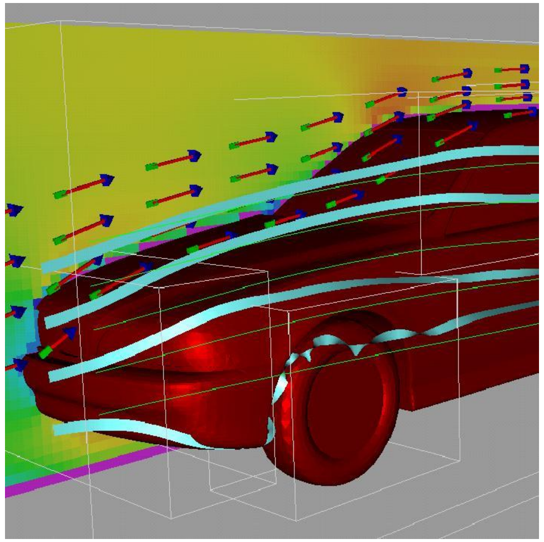
- Streak lines

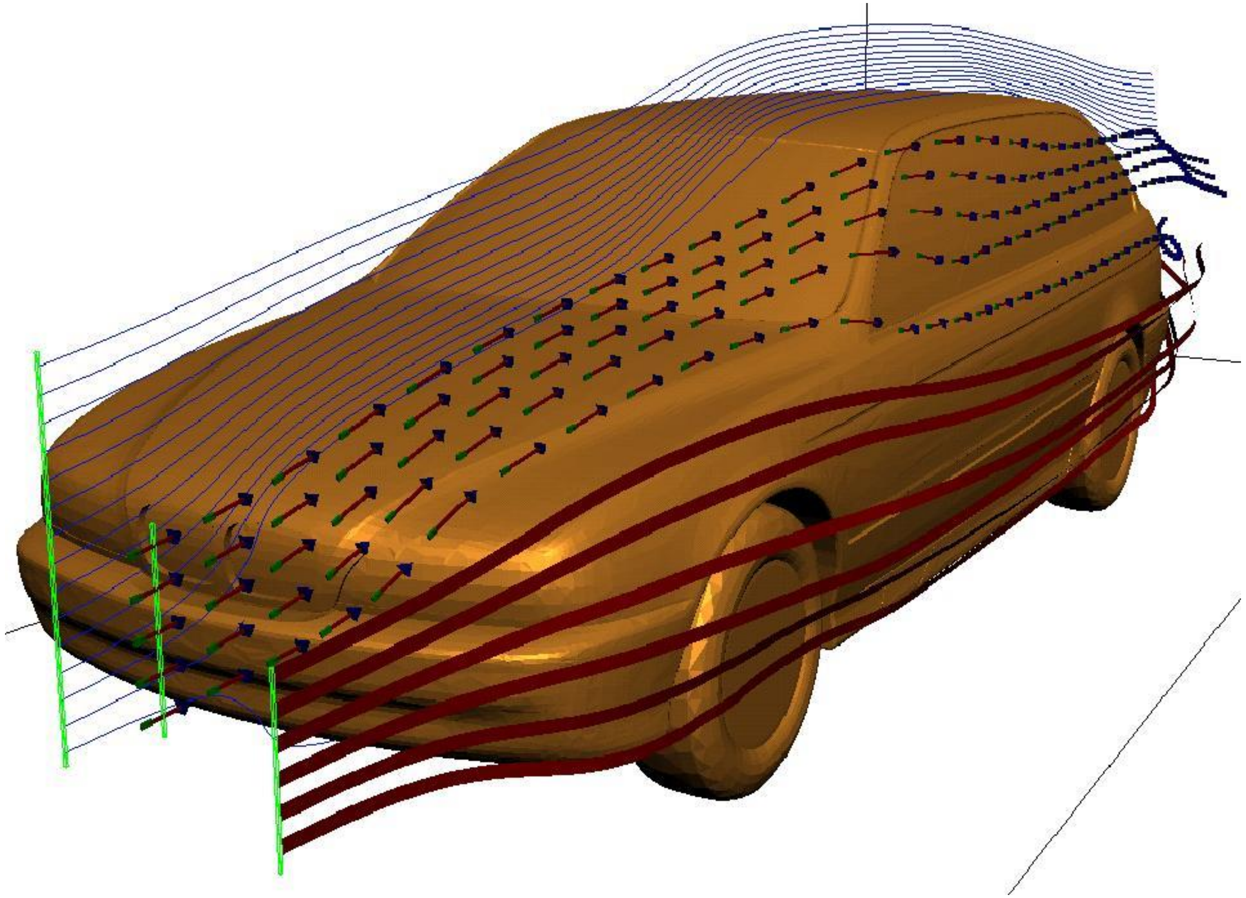




Mapping Methods Based on Particle Tracing

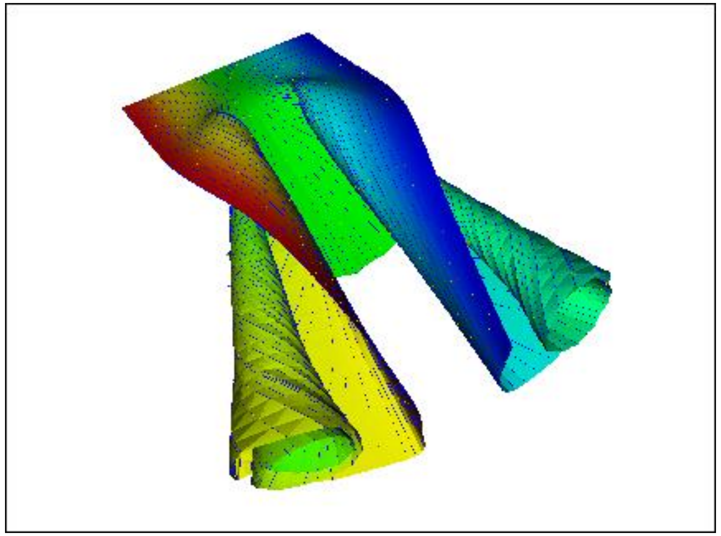
- Stream ribbons
  - Trace two close-by particles
  - Keep distance constant
  - Generate a mesh in between
  - Visualizes twist

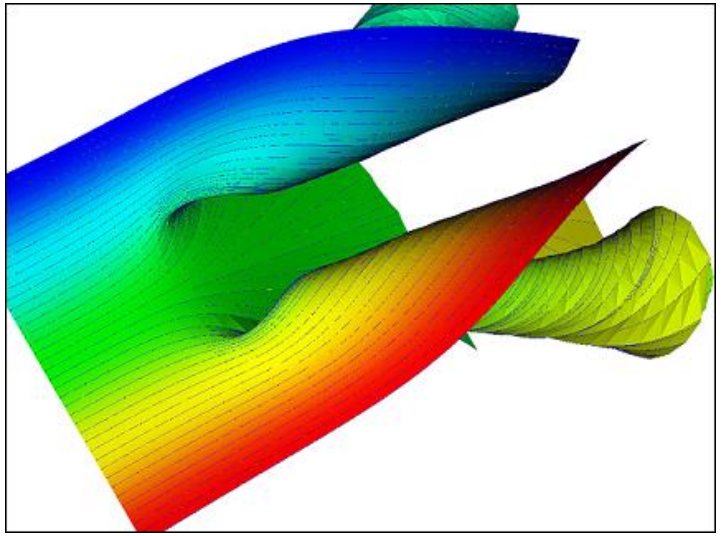




Mapping Methods Based on Particle Tracing

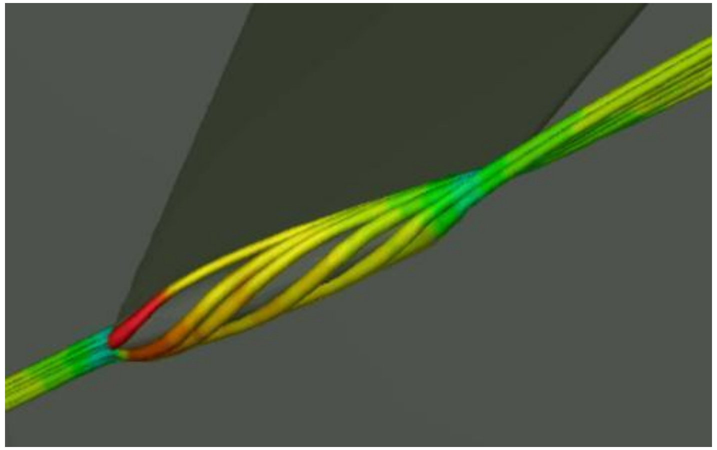
- Stream surfaces
  - Set of streamlines, started on a seeding curve
  - Construct mesh in between
  - Insert/delete streamlines in diverging/converging regions
  - Involved techniques exist for handling, e.g., rotating divergent flow

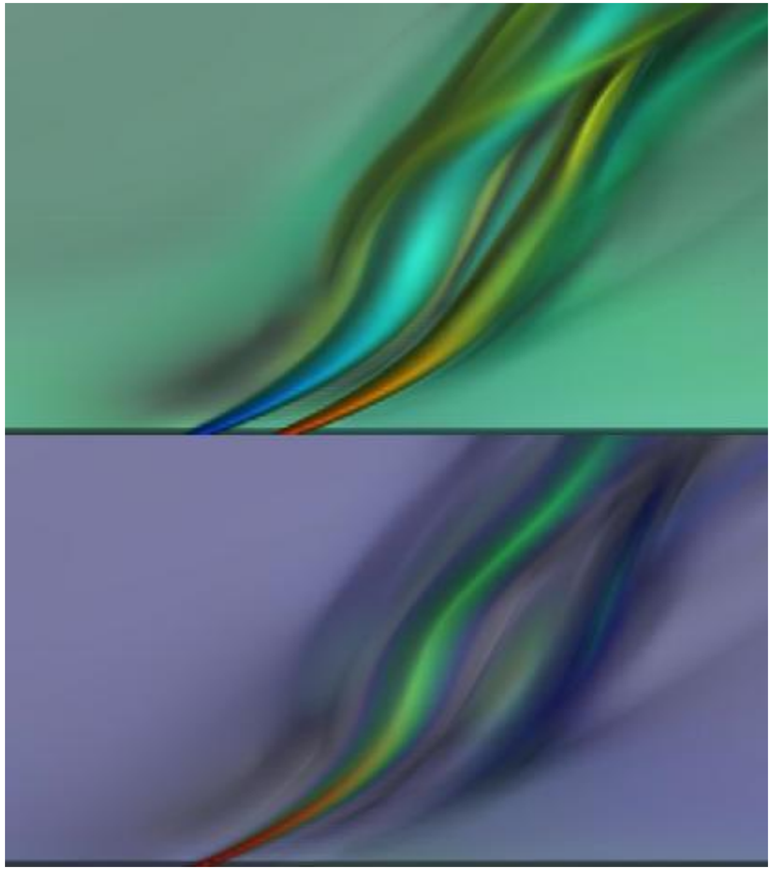




Mapping Methods Based on Particle Tracing

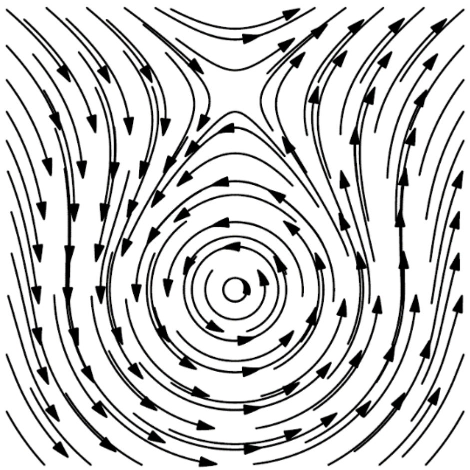
- Stream tubes
  - Closed seeding curve for stream surface, e.g., triangle or circle
  - Relation to conservation laws, e.g., constant flux through cross sections because no flux through tube (Gauss)



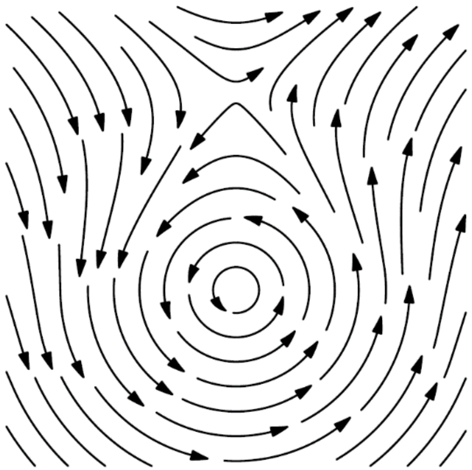


Mapping Methods Based on Particle Tracing

- Streamline placement
  - Arrange streamlines to depict overall flow
  - Even distribution of streamlines
  - Show important features of flow
  - Between sparse and dense representation



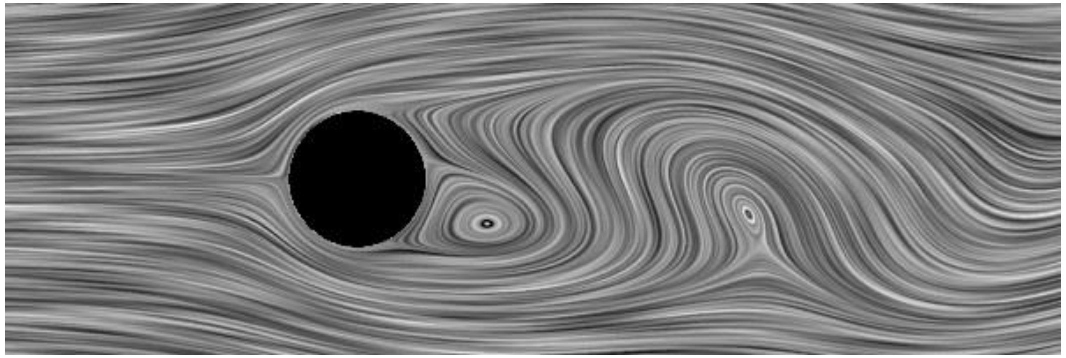
seeded on regular grid

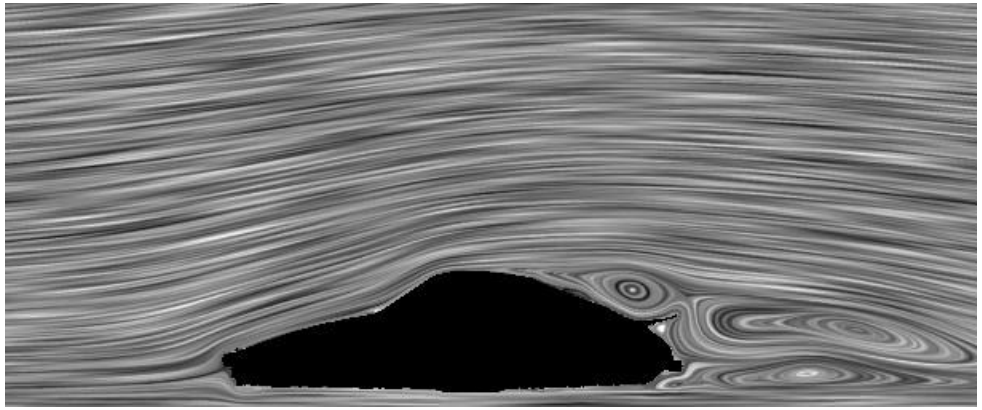


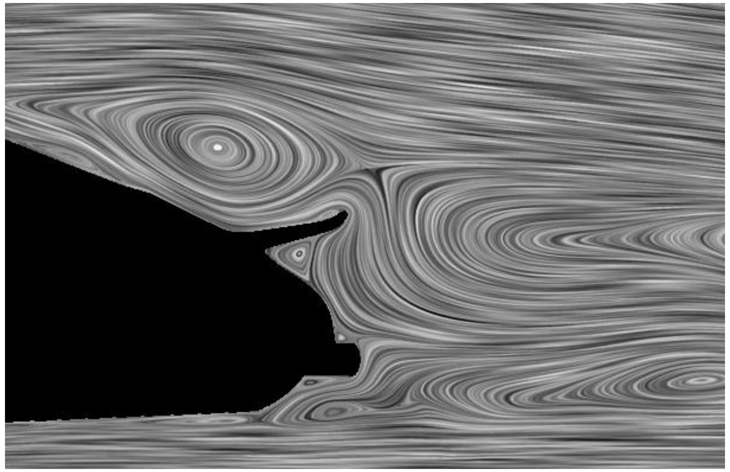
streamline placement

Mapping Methods Based on Particle Tracing

- Line Integral Convolution (LIC)
  - Texture representation
  - Dense







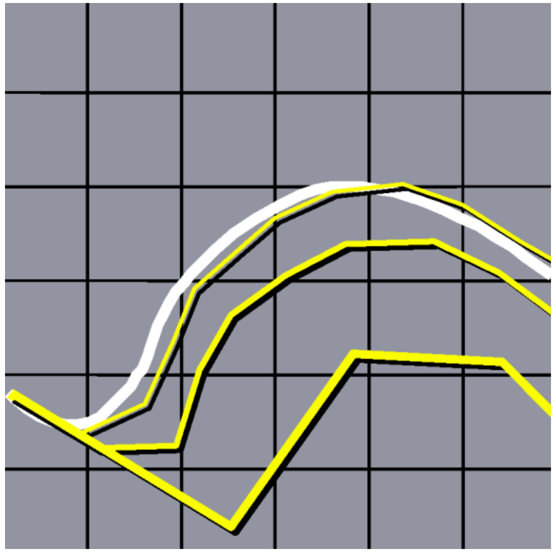
Numerical Integration of ODEs

- Typical example of particle tracing problem (path line):

$$\boldsymbol{L}(t_0) = x_0\,, \qquad \frac{d\big(\boldsymbol{L}(t)\big)}{dt} \times \boldsymbol{u}(\boldsymbol{L}(t), t) = 0\,, \qquad \frac{d\big(\boldsymbol{L}(t)\big)}{dt} = \boldsymbol{u}(\boldsymbol{L}(t), t)$$

- Initial value problem for ordinary differential equations (ODE)
- What kind of numerical solver?

Numerical Integration of ODEs



- Rewrite ODE in generic form
- Initial value problem for:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, t)$$

- Most simple approach: explicit Euler method

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{f}(\mathbf{x}, t)$$

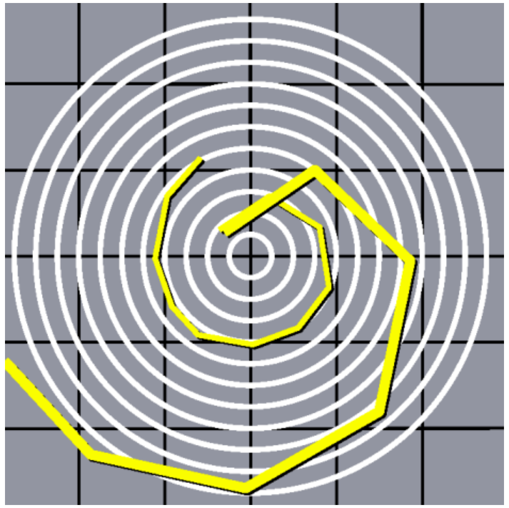
- Based on Taylor expansion

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \dot{\mathbf{x}}(t) + O(\Delta t^2)$$

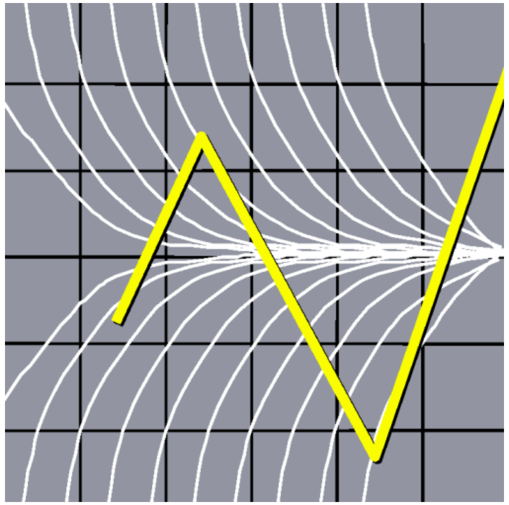
- First-order method (global error proportional to Δt)
- Higher accuracy with smaller step size Δt

Numerical Integration of ODEs

- Problem of Euler method
  - Inaccurate



  - Unstable



- Example: $f = -kx$

  $x = e^{-kt}$ divergence for $\Delta t > 2/k$

Numerical Integration of ODEs

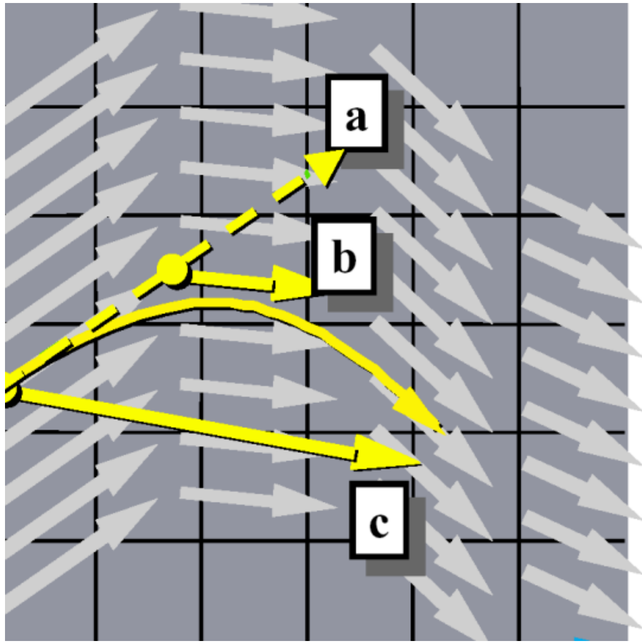
- **Midpoint method:**
  a. Euler step $\Delta \mathbf{x} = \Delta t\, \mathbf{f}(\mathbf{x}, t)$

  b. Evaluation of **f** at midpoint

$$\mathbf{f}_{\text{mid}} = \mathbf{f}\left(\mathbf{x} + \frac{\Delta \mathbf{x}}{2}, t + \frac{\Delta t}{2}\right)$$

  c. Complete step with value at midpoint

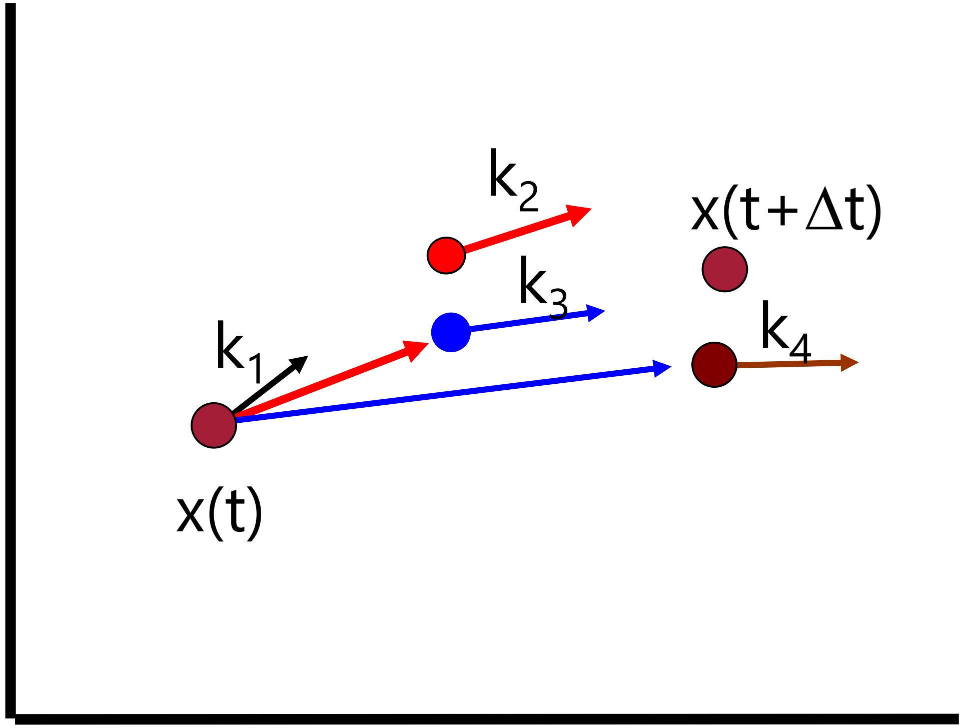
$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{f}_{\text{mid}}$$



  - Method of second order (global error reduced by factor $1/2^2$ if step $\Delta t/2$)

Numerical Integration of ODEs

- Fourth-order Runge-Kutta method



$$\mathbf{k}_1 = \Delta t\,\mathbf{f}(\mathbf{x}, t)$$

$$\mathbf{k}_2 = \Delta t\,\mathbf{f}\left(\mathbf{x} + \frac{\mathbf{k}_1}{2}, t + \frac{\Delta t}{2}\right)$$

$$\mathbf{k}_3 = \Delta t\,\mathbf{f}\left(\mathbf{x} + \frac{\mathbf{k}_2}{2}, t + \frac{\Delta t}{2}\right)$$

$$\mathbf{k}_4 = \Delta t\,\mathbf{f}\left(\mathbf{x} + \mathbf{k}_3, t + \Delta t\right)$$

$$\mathbf{x}(t + \Delta t) = \mathbf{x} + \frac{\mathbf{k}_1}{6} + \frac{\mathbf{k}_2}{3} + \frac{\mathbf{k}_3}{3} + \frac{\mathbf{k}_4}{6} + O\left(\Delta t^5\right)$$

Numerical Integration of ODEs

- Adaptive step size control
  - Change step size according to the error
  - Decrease/increase step size depending on whether actual local error is high/low
  - Higher integration speed in “simple” regions
  - Good error control
- Approaches:
  - Step size doubling
  - Embedded Runge-Kutta schemes
- Further reading:
  - SA Teukolsky, WT Vetterling, BP Flannery: *Numerical Recipes*, WH Press

Numerical Integration of ODEs

- So far only explicit methods
- Stability problem can be solved by implicit methods
- Implicit Euler method

$$\mathbf{x}(t+\Delta t)-\mathbf{x}(t)=\Delta t\ \mathbf{f}\big(\mathbf{x}(t+\Delta t),t+\Delta t\big)$$

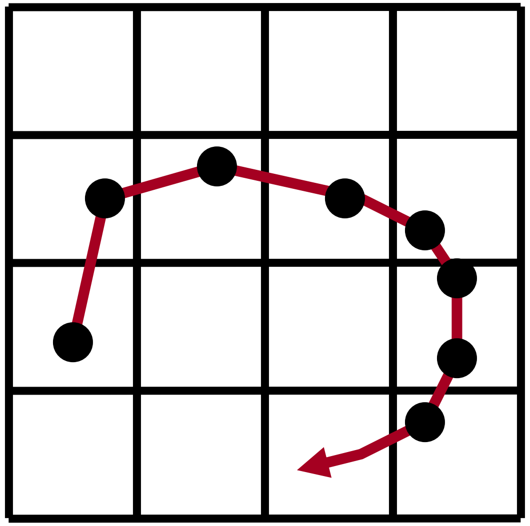
- "Reversing" the explicit Euler integration step → $\mathbf{x}(t+\Delta t)=\mathbf{x}(t)+\Delta t\,\mathbf{f}(\mathbf{x},t)$
- Taylor expansion around $t + \Delta t$ instead of t
- Solving the system of non-linear equations to determine $\mathbf{x}(t + \Delta t)$
- Using implicit methods allows larger time steps

Particle Tracing on Grids

- Vector field given on a grid
- Solve

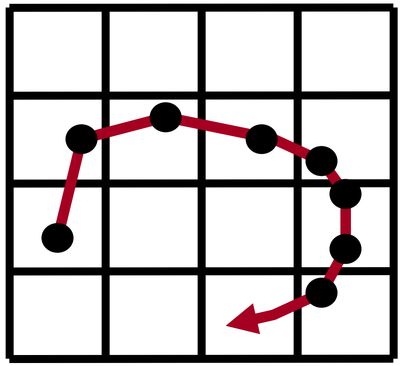
$$\boldsymbol{L}(t_0) = x_0\,, \qquad \frac{d\big(\boldsymbol{L}(t)\big)}{dt} = \boldsymbol{u}(\boldsymbol{L}(t), t)$$

  for the path line
- Incremental integration
- Discretized path of the particle



Particle Tracing on Grids

- Most simple case: Cartesian grid for the path line



- Basic algorithm:

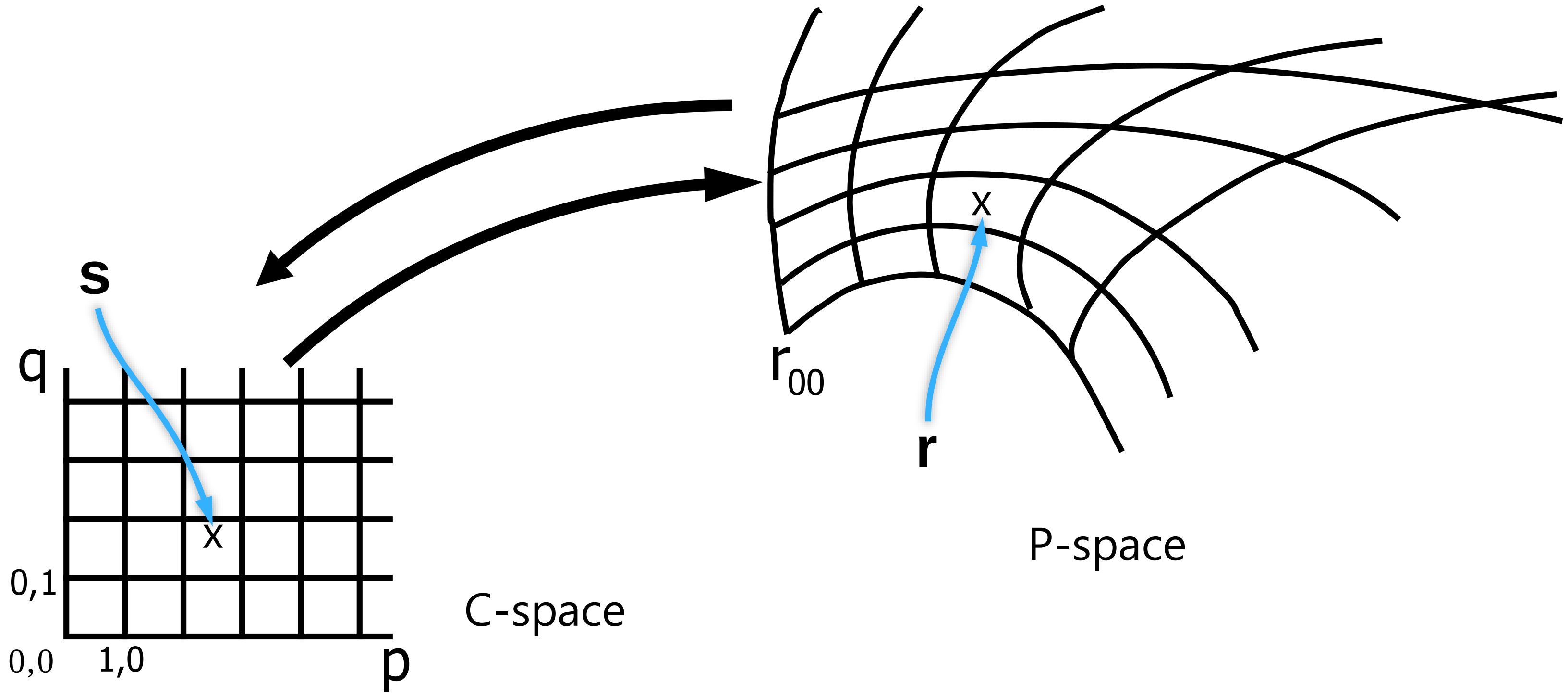
```
Select start point (seed point)
Find cell that contains start point                → point location
While (particle in domain) do
    Interpolate vector field at current position   → interpolation
    Integrate to new position                      → integration
    Find new cell                                  → point location
    Draw line segment between latest particle positions
Endwhile
```

Particle Tracing on Grids

- Point location (cell search) on Cartesian grids:
  - Indices of cell directly from position (x, y, z)
  - For example: $i_x = (x - x_0) / \Delta x$
  - Simple and fast
- Interpolation on Cartesian grids:
  - Bilinear (in 2D) or trilinear (in 3D) interpolation
  - Required to compute the vector field (= velocity) inside a cell
  - Component-wise interpolation
  - Based on offsets (local coordinates within cell)

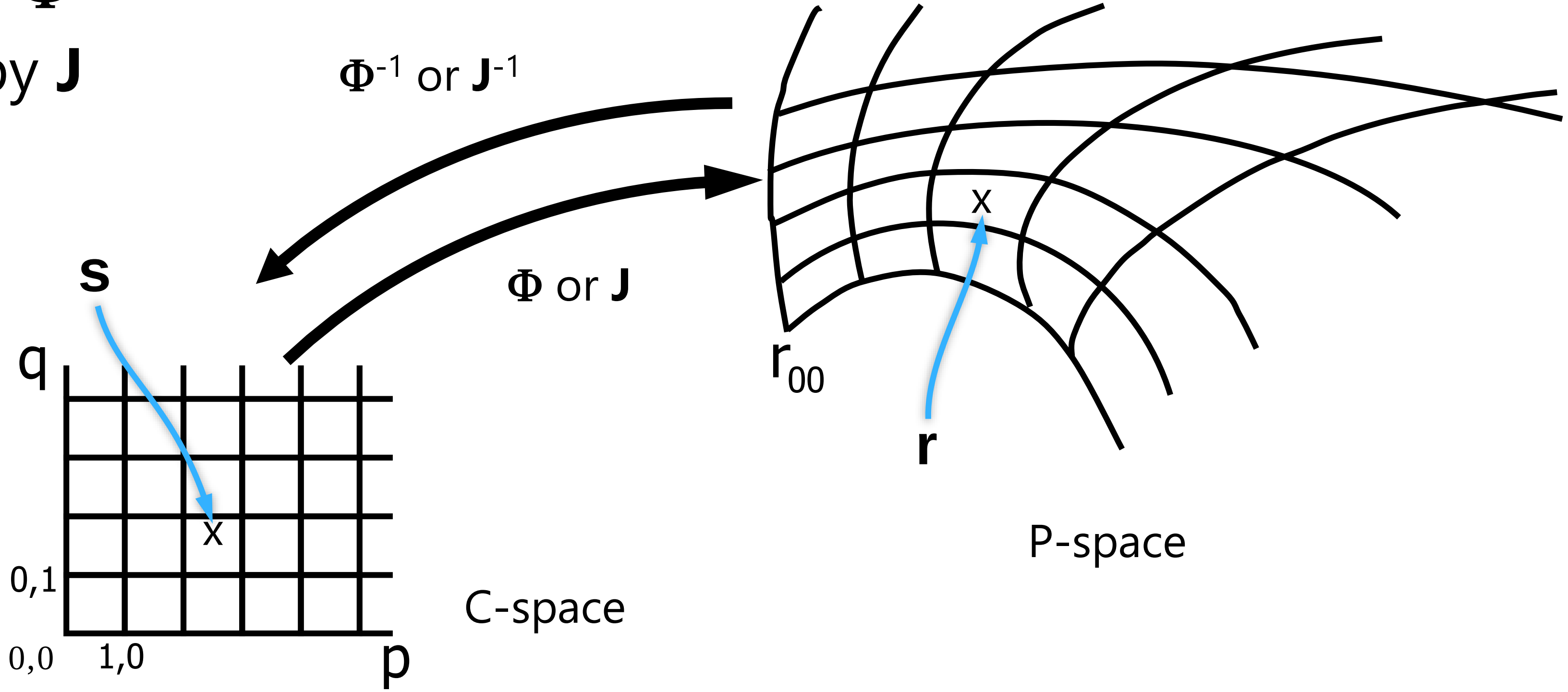
Particle Tracing on Grids

- How are curvilinear grids handled?
- C-space (computational space) vs. P-space (physical space)



Particle Tracing on Grids

- Particle tracing can either be done in C-space or P-space
- Transformation of
  - Points by **Φ**
  - Vectors by **J**



Particle Tracing on Grids

- Transformation of points:
  - From C-space to P-space: $\mathbf{r} = \mathbf{\Phi}(\mathbf{s})$
  - From P-space to C-space: $\mathbf{s} = \mathbf{\Phi}^{-1}(\mathbf{r})$
- Transformation of vectors:
  - From C-space to P-space: $\mathbf{u} = \mathbf{J} \cdot \mathbf{v}$
  - From P-space to C-space: $\mathbf{v} = \mathbf{J}^{-1} \cdot \mathbf{u}$
  - $\mathbf{J}$ is Jacobian of $\mathbf{\Phi}$:

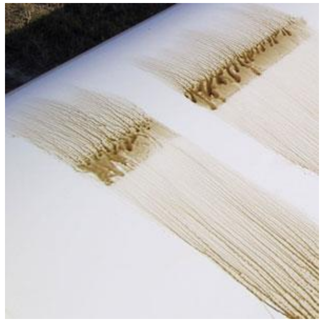
$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathbf{\Phi}_x}{\partial p} & \frac{\partial \mathbf{\Phi}_x}{\partial q} \\ \frac{\partial \mathbf{\Phi}_y}{\partial p} & \frac{\partial \mathbf{\Phi}_y}{\partial q} \end{pmatrix} \quad \text{(2D case)}$$

Particle Tracing on Grids

- Important properties of C-space integration:
  + Simple incremental cell search
  + Simple interpolation
  - Complicated transformation of velocities / vectors

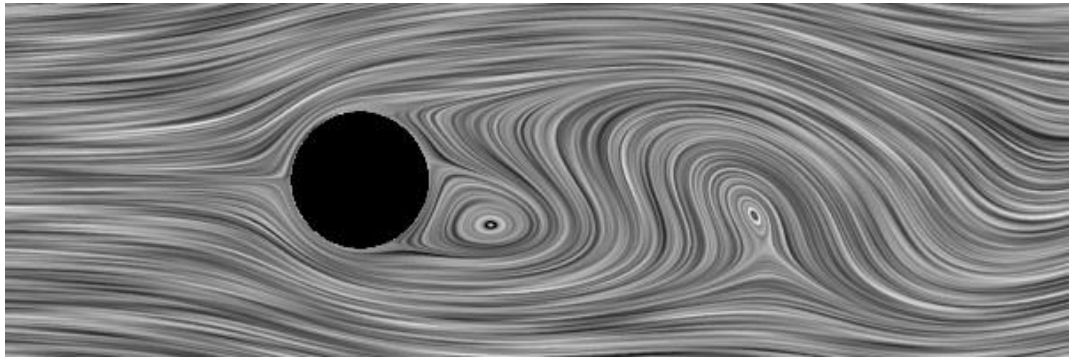
- Important properties of P-space integration:
  + No transformation of velocities / vectors
  - Complicated point location for bi- / trilinear interpolation

Line Integral Convolution (LIC)



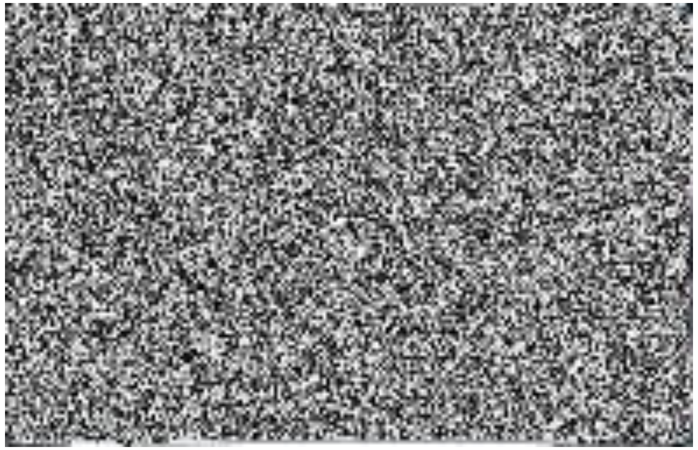
- Mimic physical experiment
  - Place oil drops on surface, apply flow (wind)
- Cover domain with a random texture
  - So-called ‚input texture', usually stationary white noise
- Blur (convolve) texture along streamlines using specified filter kernel

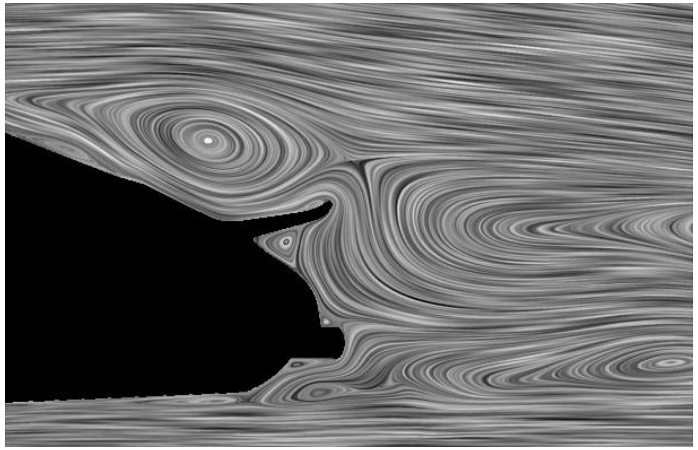
- Look of 2D LIC images
  - Intensity distribution along streamlines shows high correlation
  - No correlation between neighboring streamlines



Line Integral Convolution (LIC)

- Global visualization technique
- Dense representation
- Start with random texture
- Smear out along streamlines





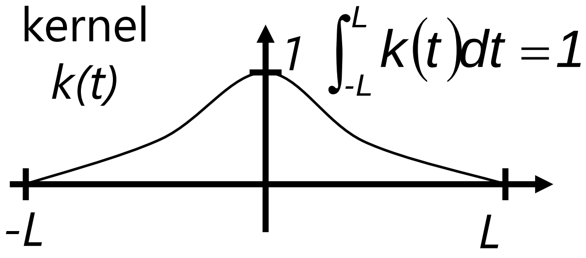
Line Integral Convolution (LIC)

- Algorithm for 2D LIC
  - Let $t \rightarrow \Phi_0(t)$ be the streamline containing the point (x_0, y_0) at $t = 0$
  - $T(x, y)$ is the randomly generated input texture (noise)
  - Compute the pixel intensity as:

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

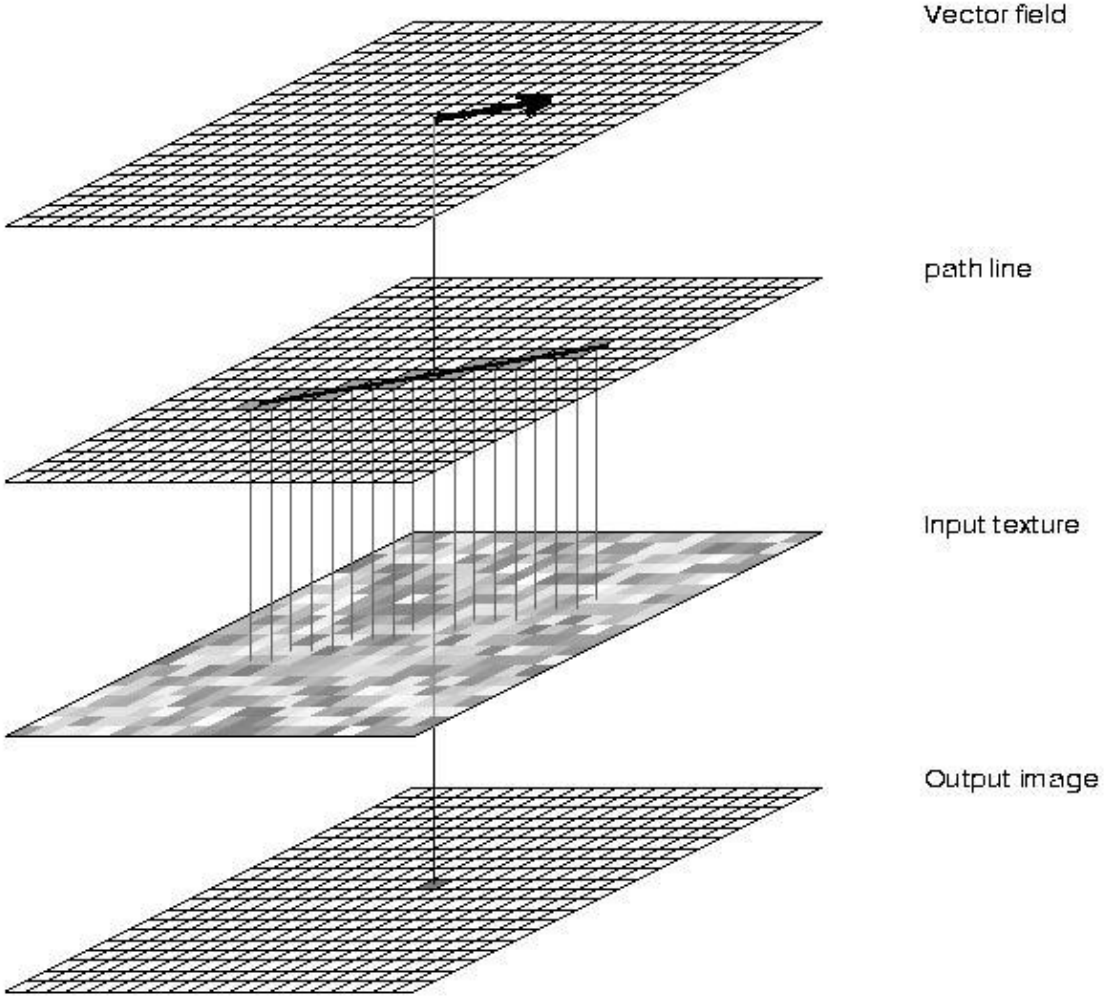
convolution with kernel k

- Kernel:
  - Finite support $[-L, L]$
  - Normalized
  - Often simple box filter used for *k(t)*
  - Often symmetric (isotropic)

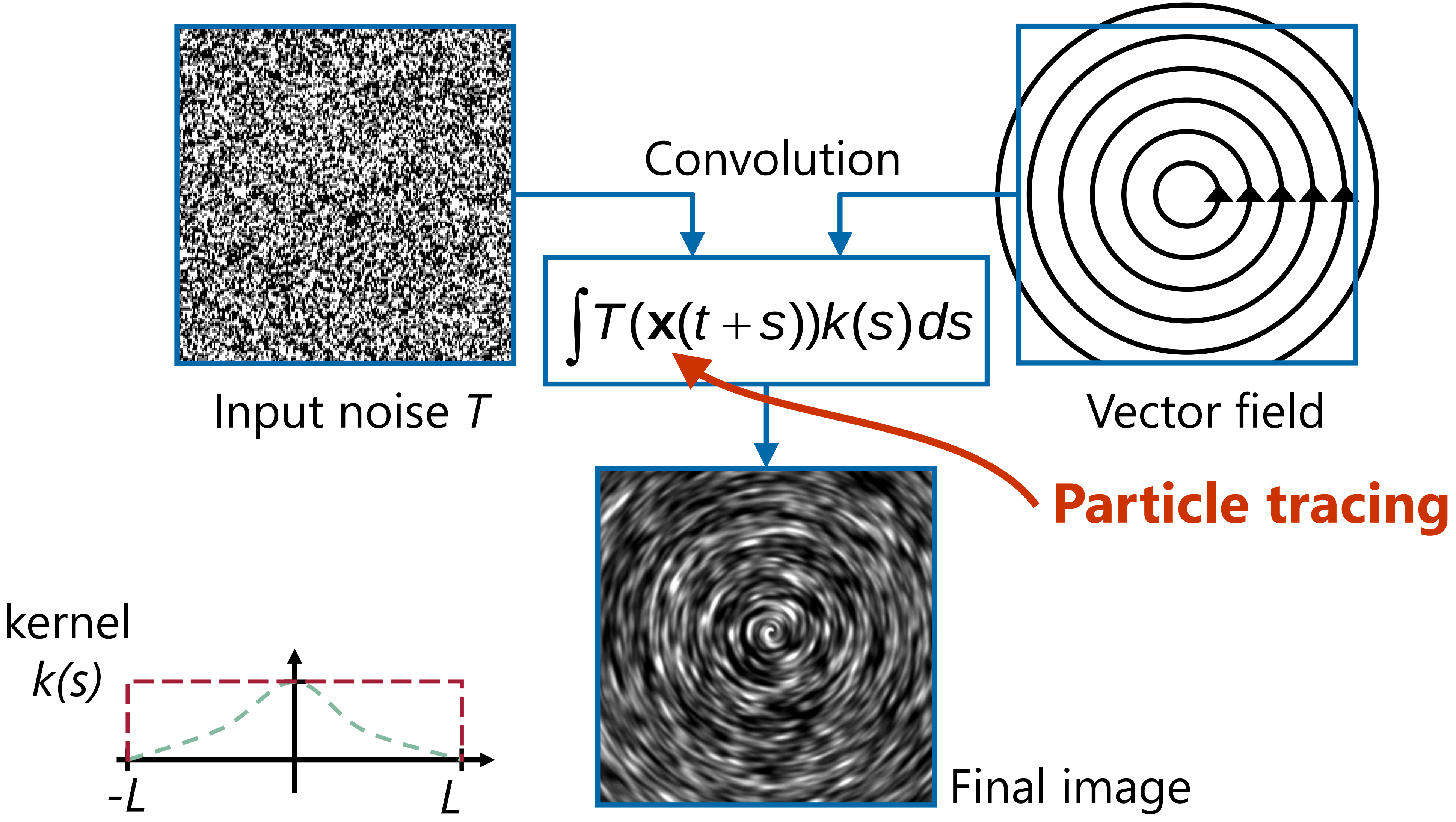


Line Integral Convolution (LIC)

- Algorithm for 2D LIC
  - Convolve a random texture along the streamlines



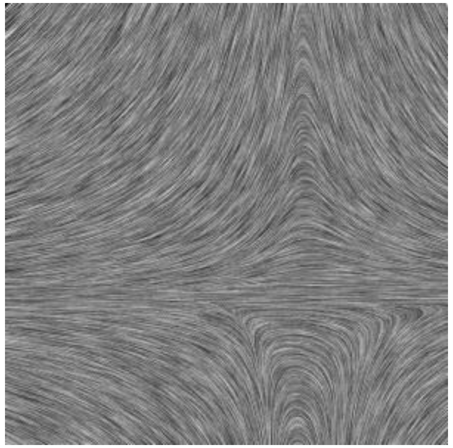
Line Integral Convolution (LIC)

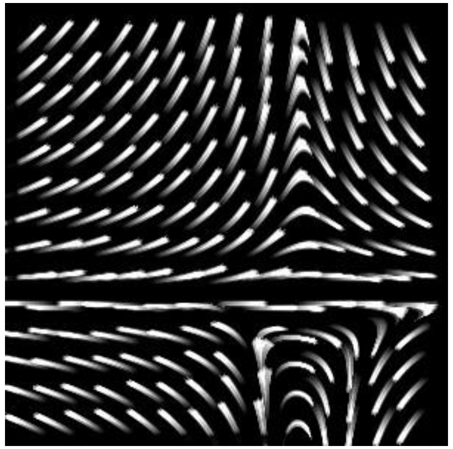


Line Integral Convolution (LIC)

Extensions

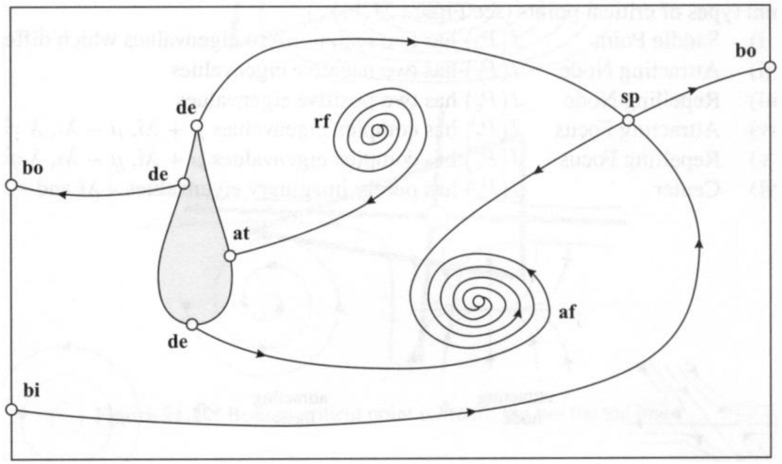
- Fast LIC
  - Problem:
    - New streamline & convolution (integral) is computed at each pixel → Slow
  - Idea:
    - Compute very long streamlines & reuse them for many different pixels
    - Incremental computation of the convolution integral
- Oriented LIC (OLIC):
  - Visualizes orientation (in addition to direction)
  - Use sparse texture & anisotropic convolution kernel


LIC


OLIC

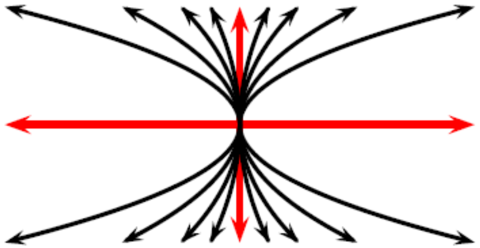
Vector Field Topology

- **Idea:** Do not draw "all" streamlines, but only the "important" ones
  - Show only *topological skeleton*
- Important points in the vector field: *critical points*
  - Points where the vector field vanishes $\boldsymbol{u} = 0$ (vector direction is undefined)
  - Sources, sinks, saddles, …
- Critical points are connected by *separatrices* (streamlines), divide the flow into regions with similar qualitatively similar behavior (in 3D: also 2D-manifolds of streamlines)
- Structure of particle behavior for $t \rightarrow$ +/- ∞

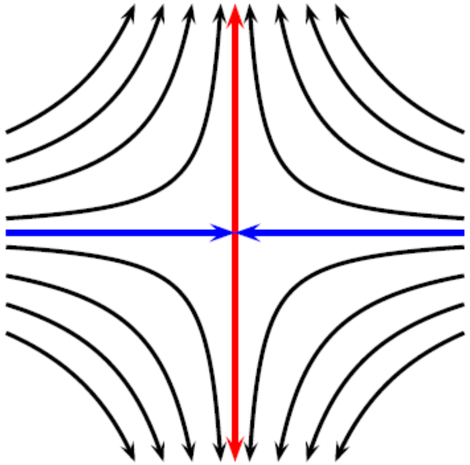


Vector Field Topology

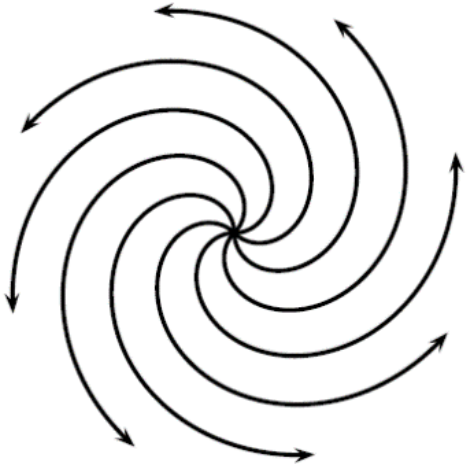
- Examples of structures in 2D



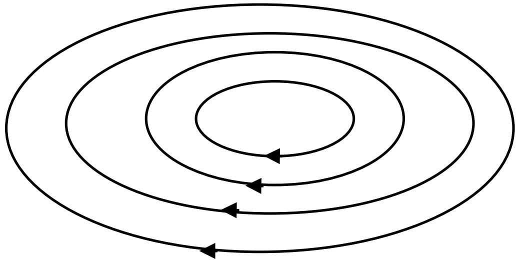
Repelling node



Saddle point



Repelling focus

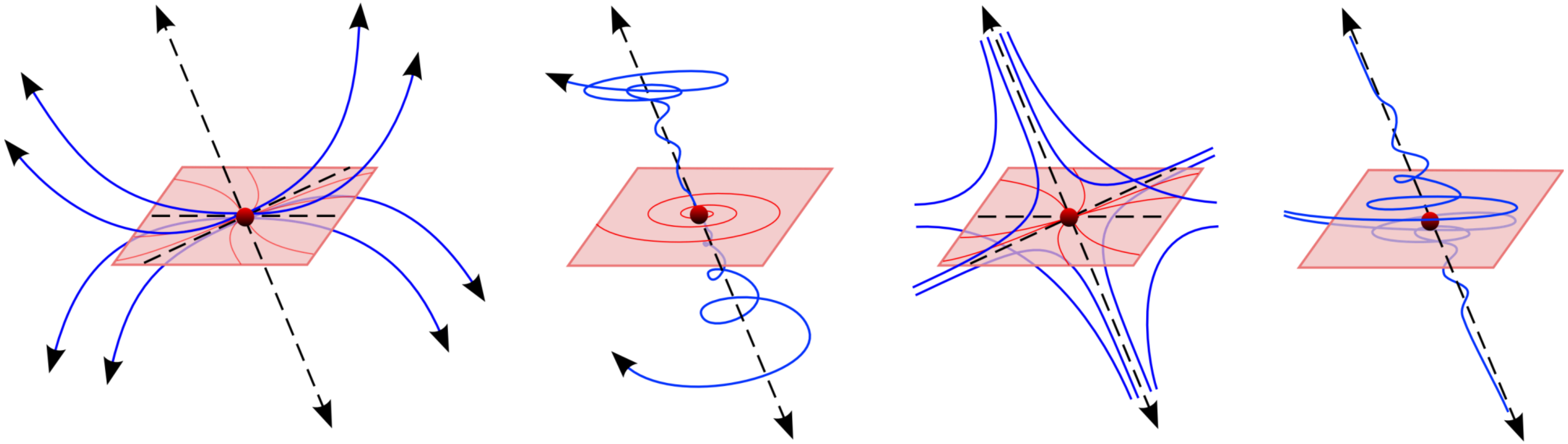


Center

Opposite cases (attracting node, attracting focus) by reversing arrows

Vector Field Topology

- Example: Types of hyperbolic critical points in 3D

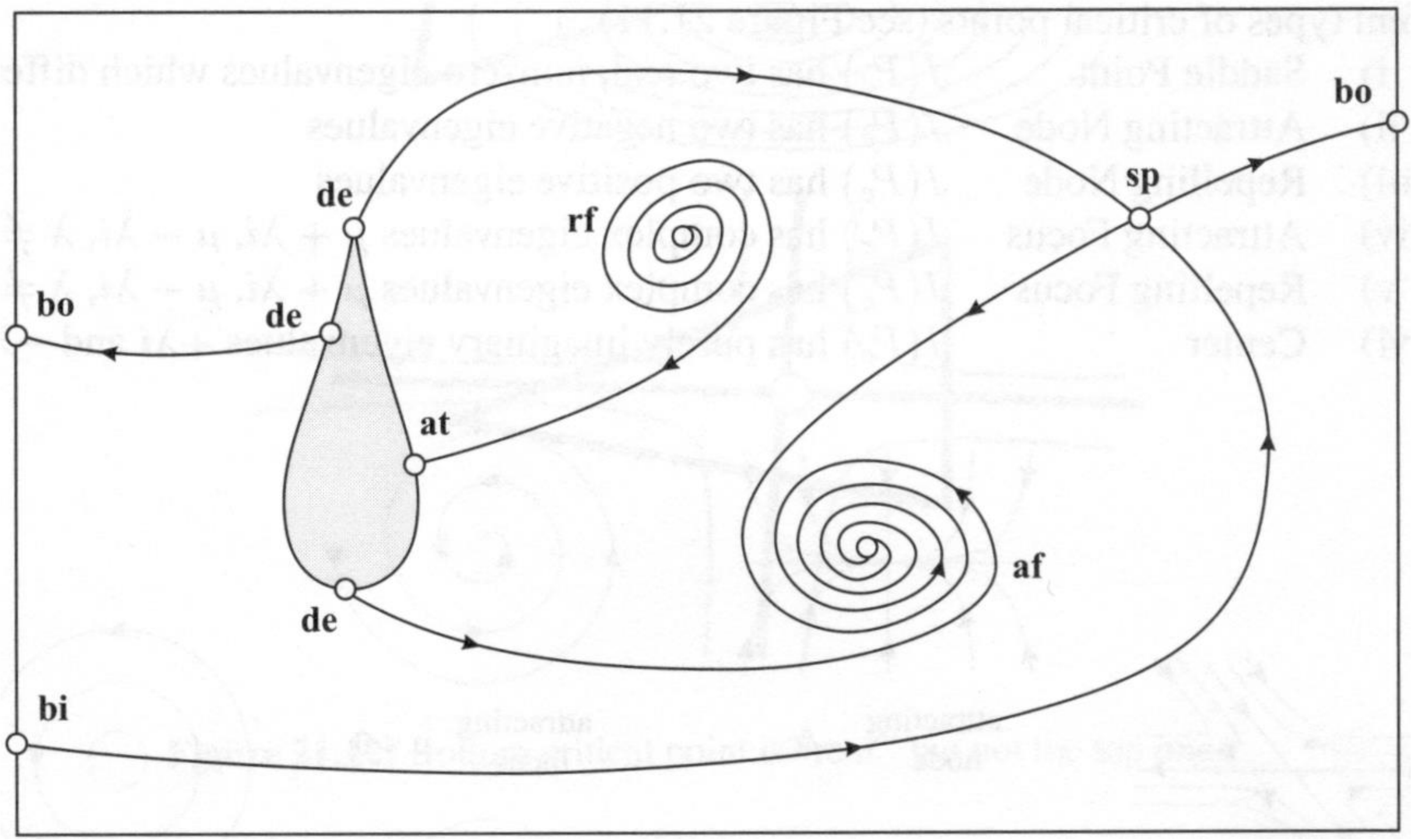


source | spiral source | 2:1 saddle | 2:1 spiral saddle

The other four types are obtained by reversing arrows

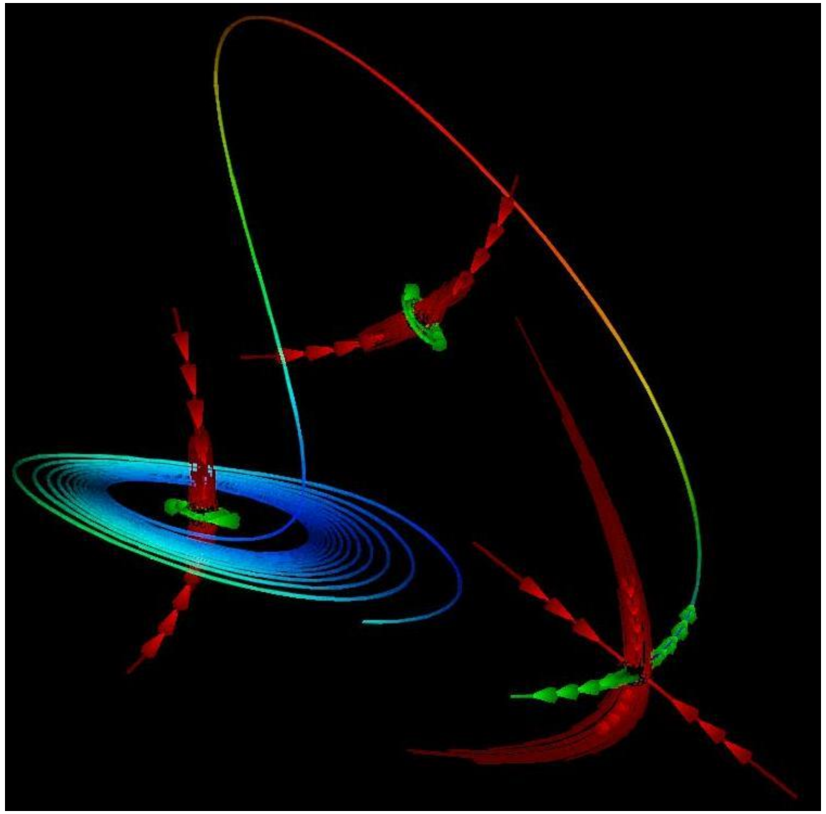
Vector Field Topology

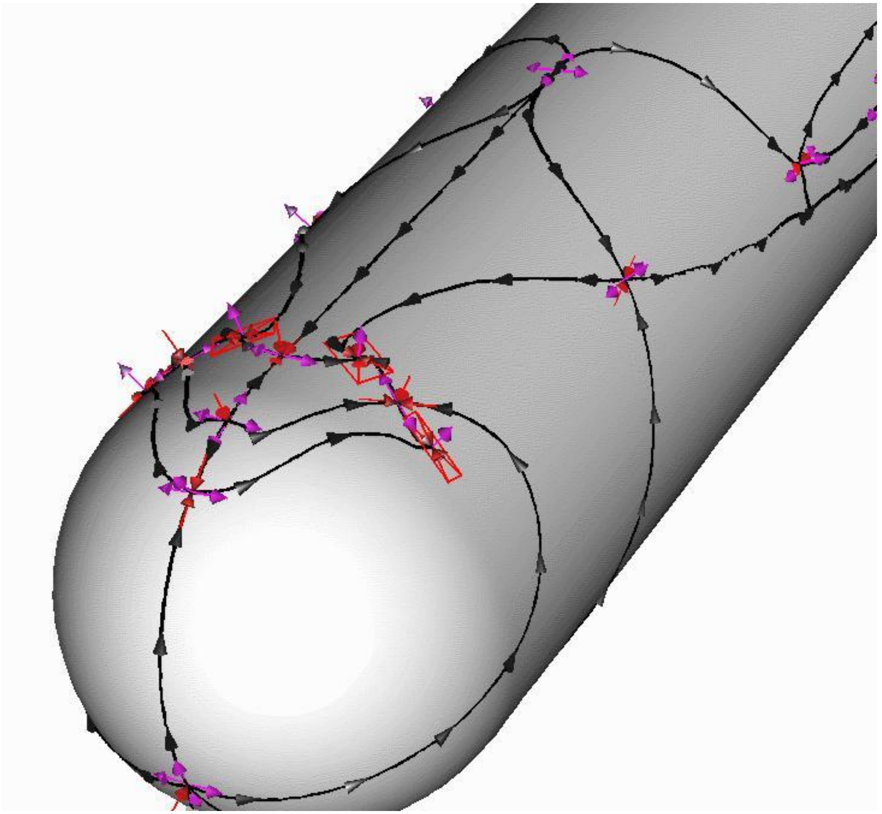
- Example of a topological graph of 2D flow field



Vector Field Topology

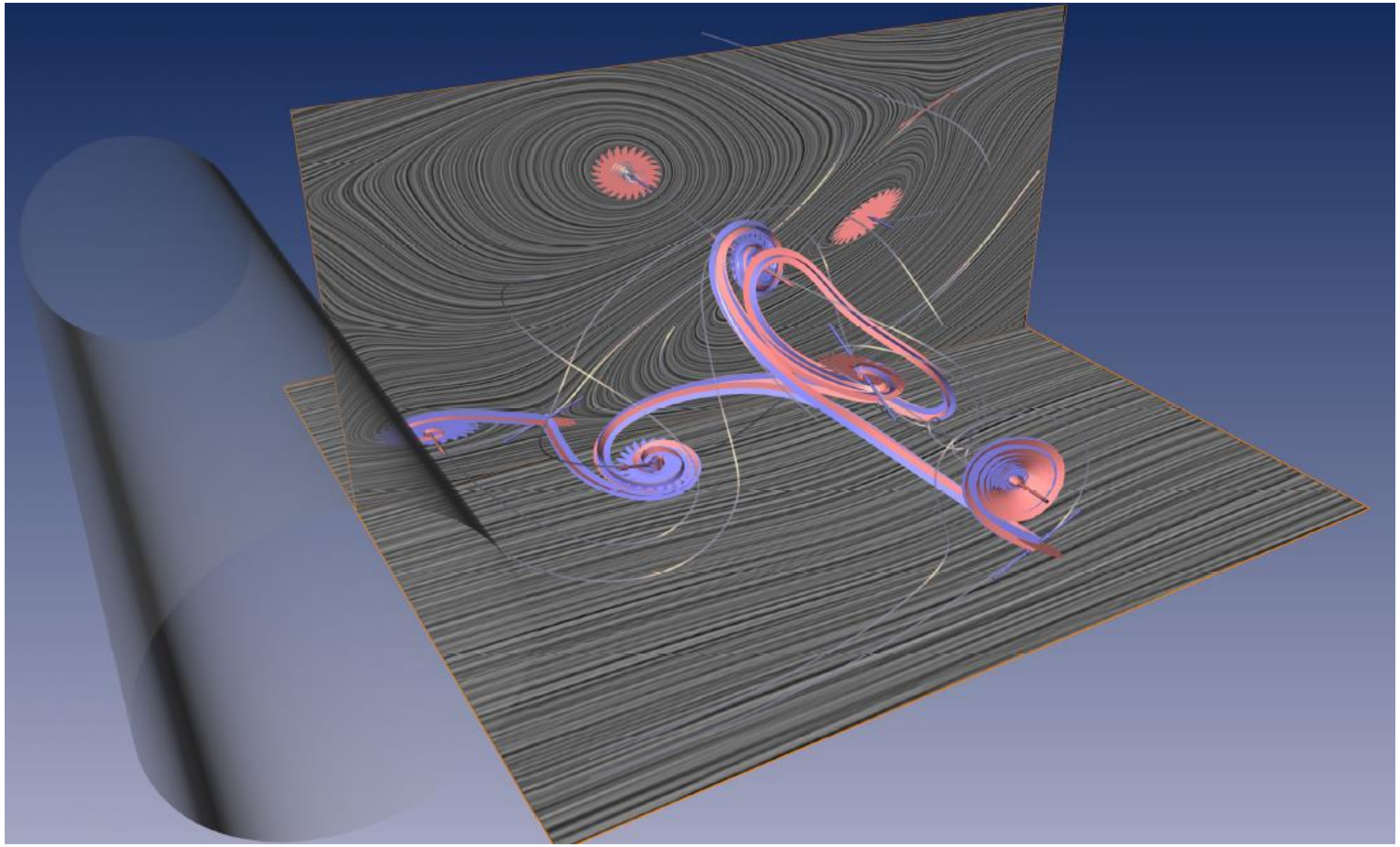
- Examples of topology-guided streamline positioning



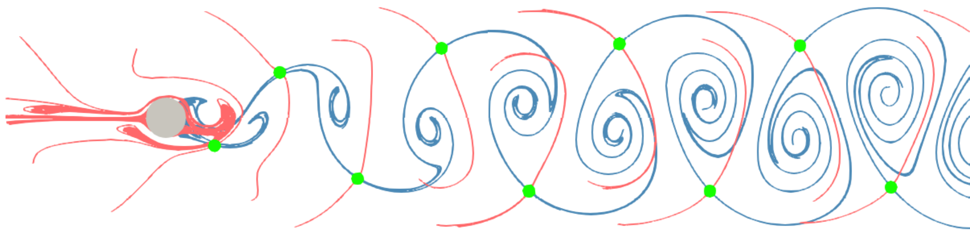


Vector Field Topology

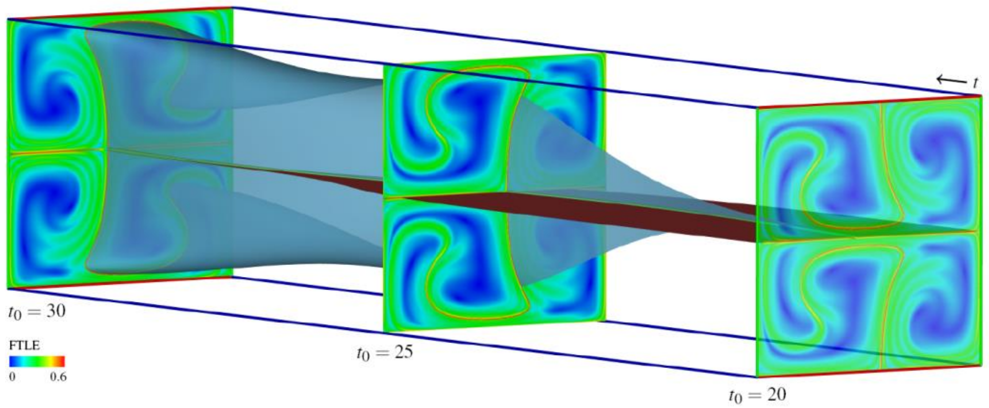
- Saddle connectors in 3D



Vector Field Topology

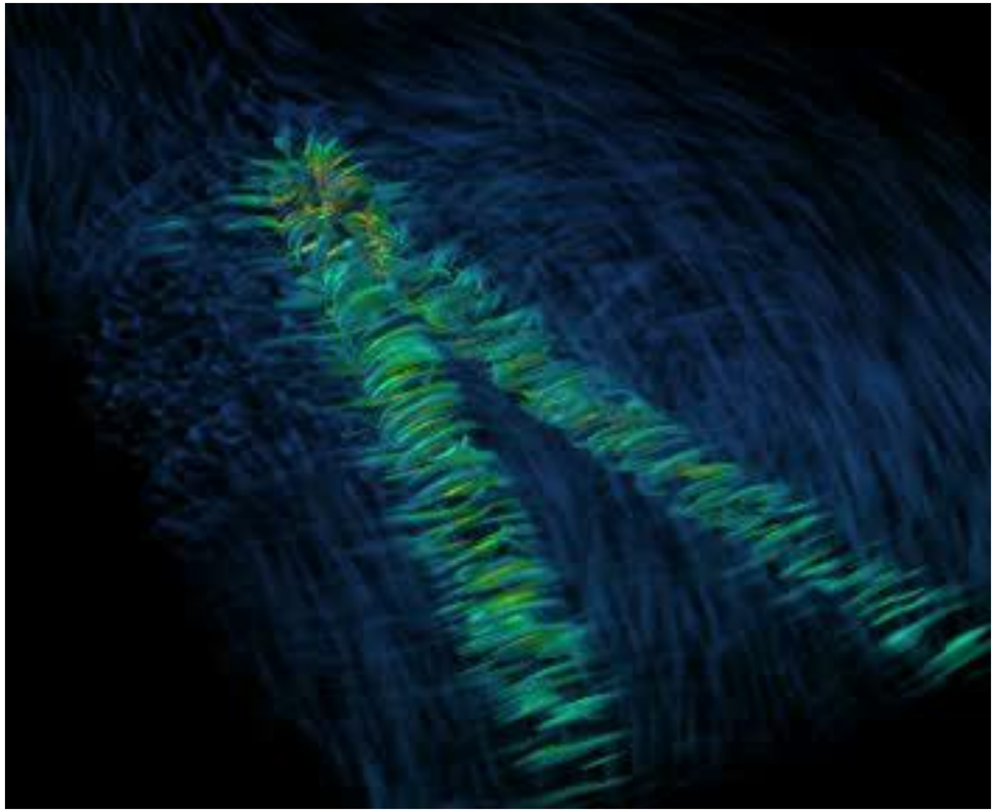


- Summary:
  - Draw only relevant streamlines/streamsurfaces (topological skeleton)
  - Partition domain into regions with similar flow behavior
  - Based on critical points
  - Strictly correct only for stationary flows (because streamlines are instantaneous)
  - Unsteady flows → Lagrangian coherent structures (finite-time Lyapunov exponent, FTLE)

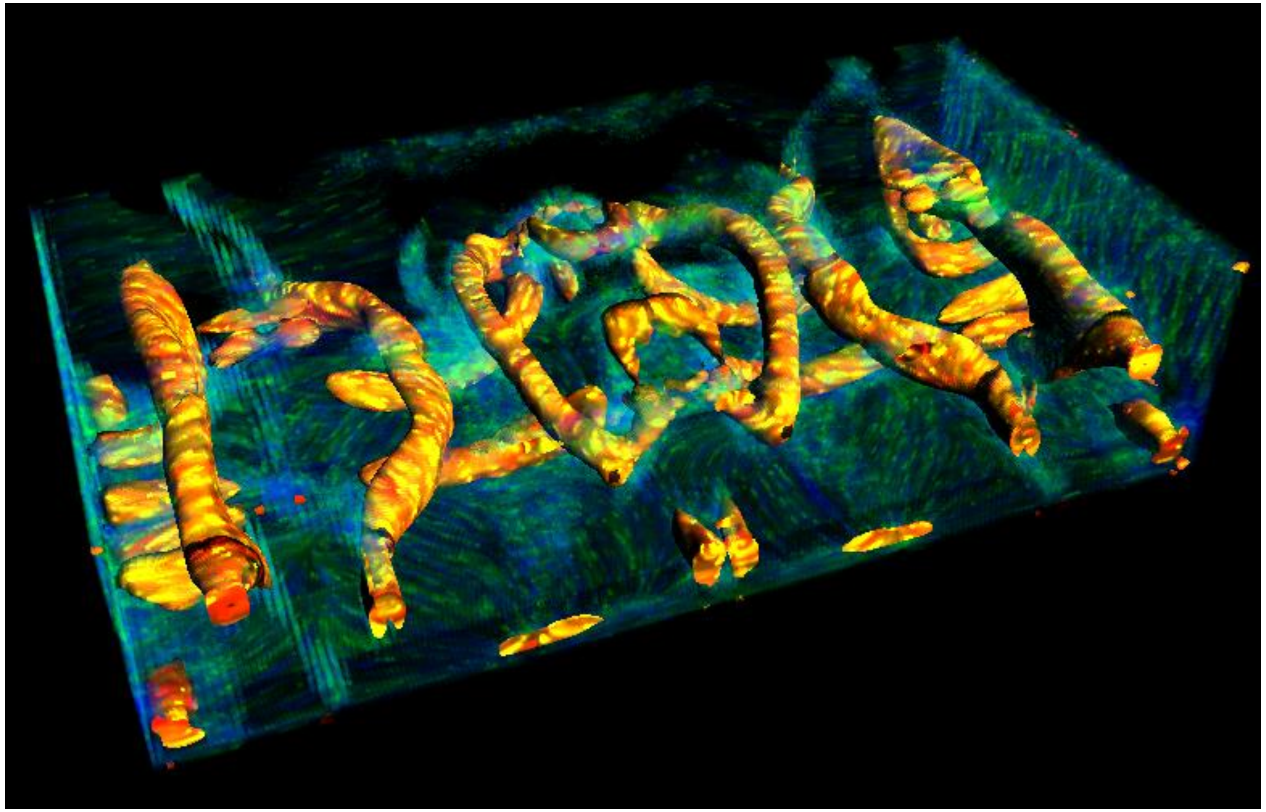


Feature-Based Visualization: Vortex Extraction

- Vortex extraction (vortices are important in fluid flow)



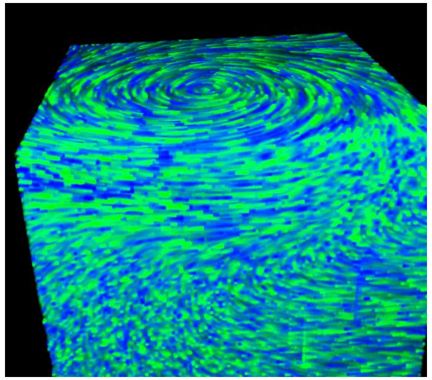
[Falk, Weiskopf, IEEE TVCG, 2008]

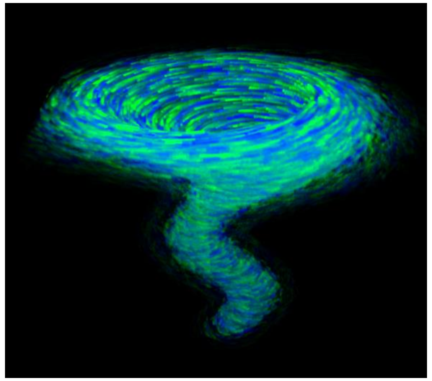


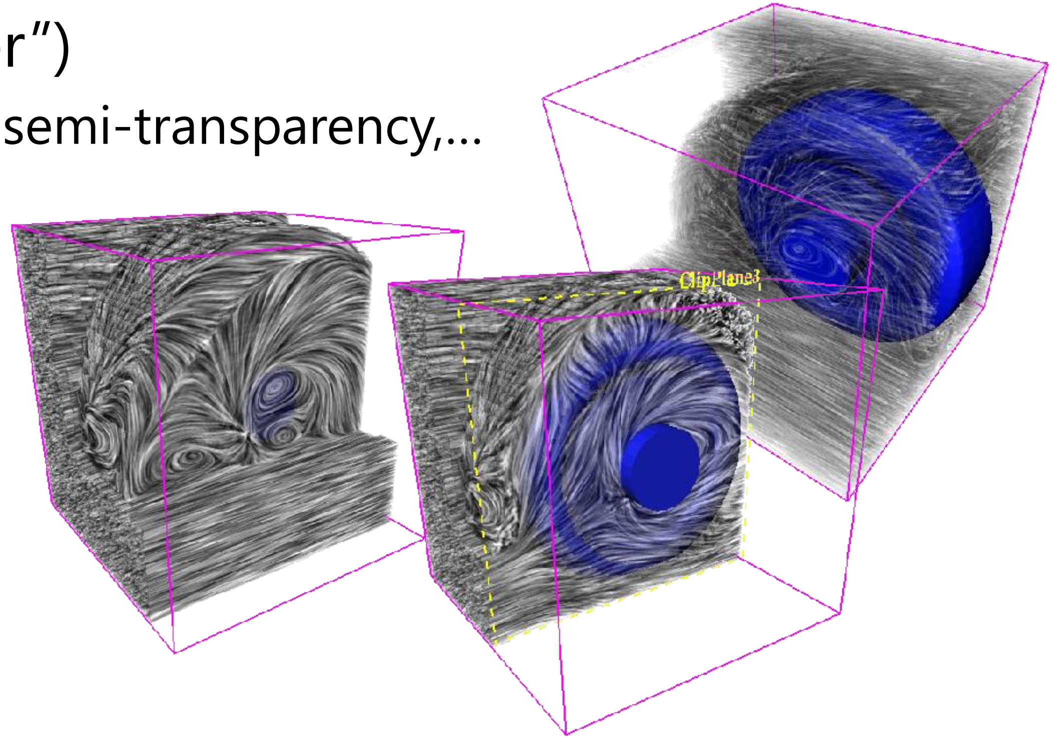
[Weiskopf et al, IEEE TVCG, 2006]

3D Vector Fields

- Most algorithms can be applied to 2D and 3D vector fields
- Main problem in 3D: effective mapping to graphical primitives
- Main aspects:
  - Occlusion & amount of data (“visual clutter”)
    - e.g., sparse representations, clipping/masking, semi-transparency,…
  - Depth perception
    - e.g., shading, occlusion, stereo disparity,…







3D Vector Fields

- **Flow visualization:** Combination of vector and scalar visualization techniques