UNIK-4660/TMA-4235, Summary

Introduction to visualization techniques

Anders Helgeland

oya@ffi.no and ahe@ffi.no

Forsvarets Forskningsinstitutt

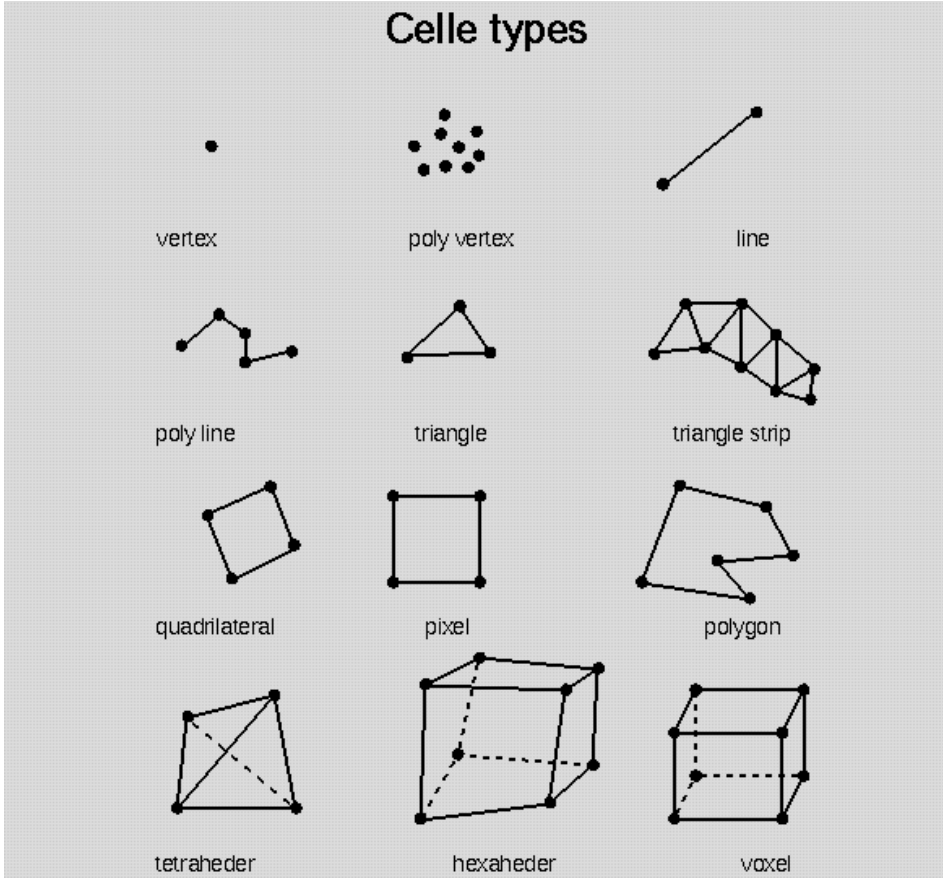
Outline

- Basic data representation
- Techniques for visualizing
  - Scalars
  - Vector
  - Tensors

Basic data representation

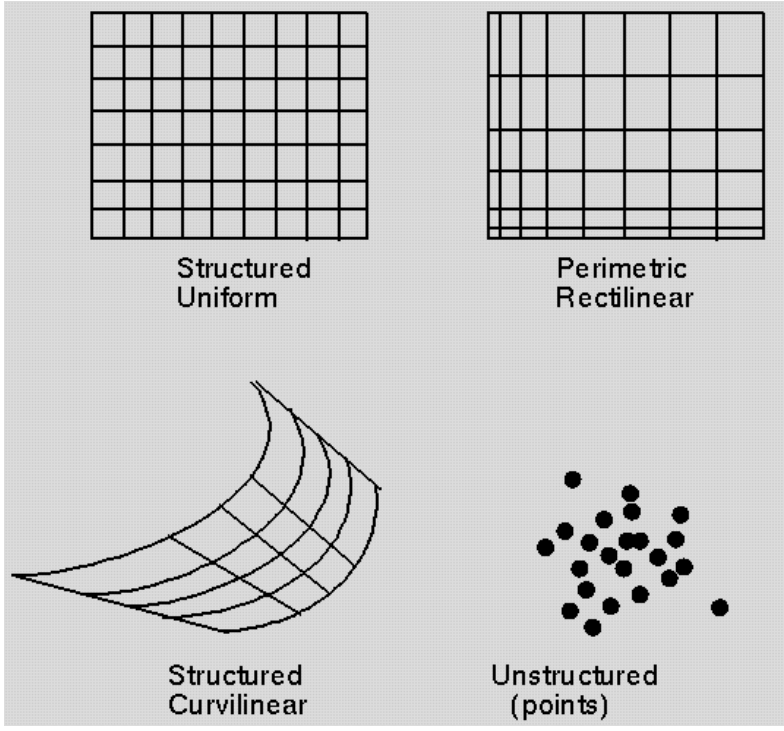
- Data to be visualized on digital computers are on a discrete form
- Data objects can be classified in two parts:
- **Organizing structure**
  - Topology
  - Geometry
  - Consists of cells and points
- **Data attributes**
  - Scalars, vectors, tensors
  - normals, texture coordinates ...

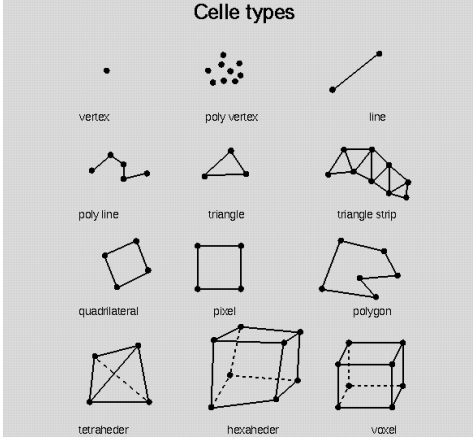
Cell types


Celle types
vertex
poly vertex
line
poly line
triangle
triangle strip
quadrilateral
pixel
polygon
tetraheder
hexaheder
voxel

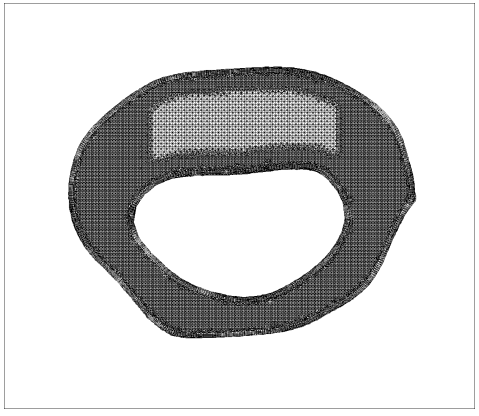
Data (Mesh/Grid) types

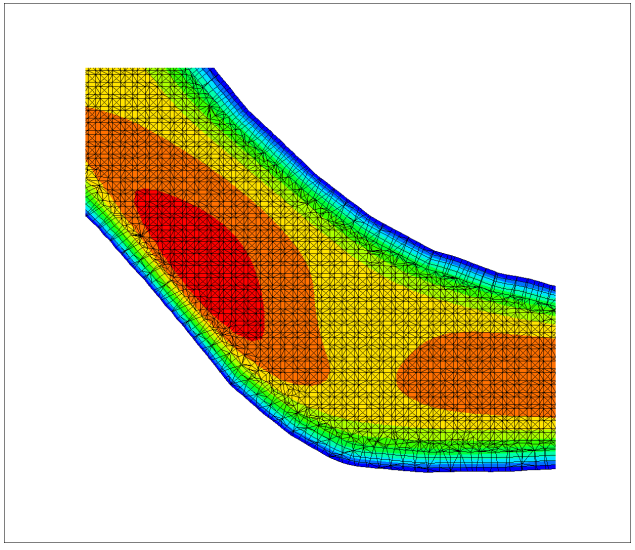
- Structured Meshes
  - Uniform Grid
  - Rectilinear Grid
  - Curved Grid
- Unstructured Meshes
  - Unstructured Points
  - Unstructured Grid

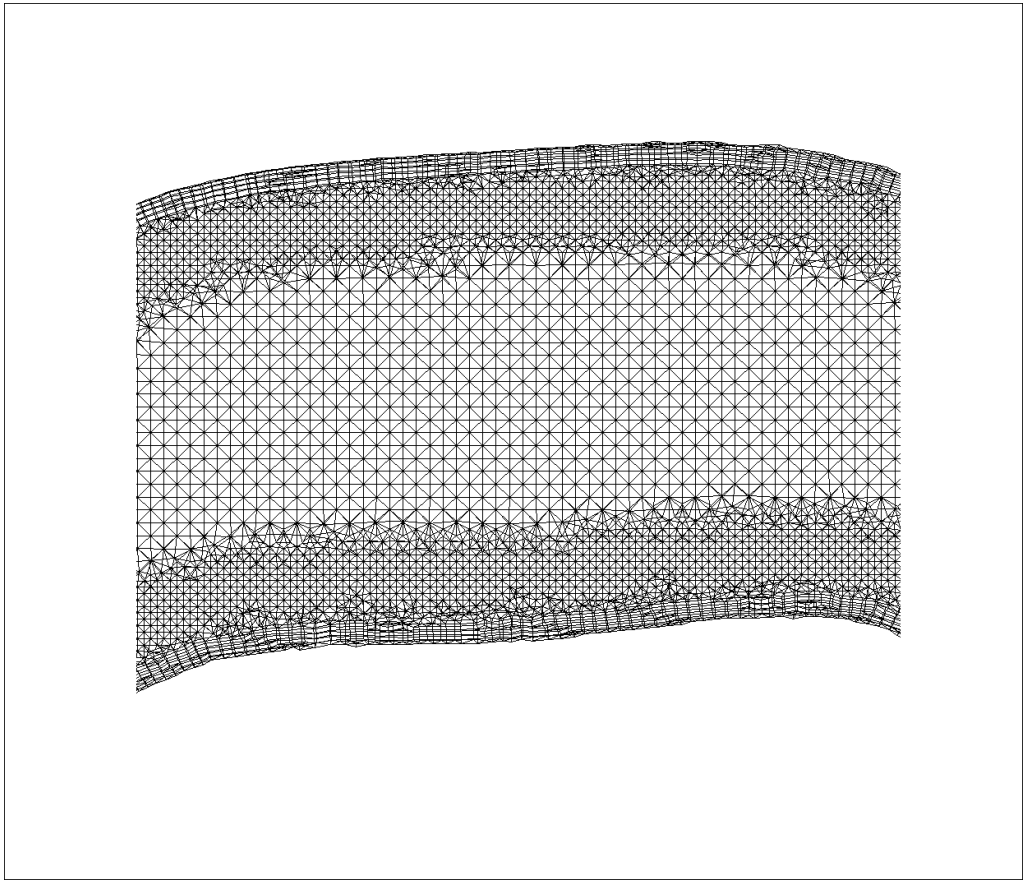




Data (Mesh) types

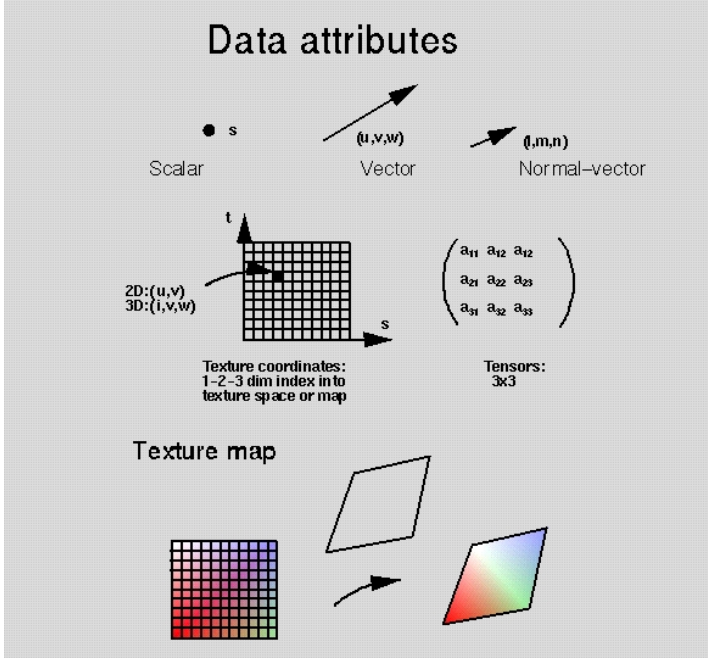




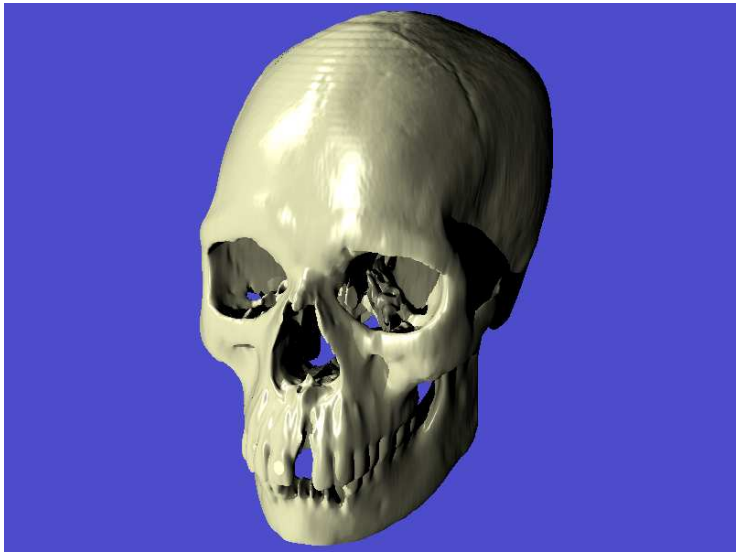
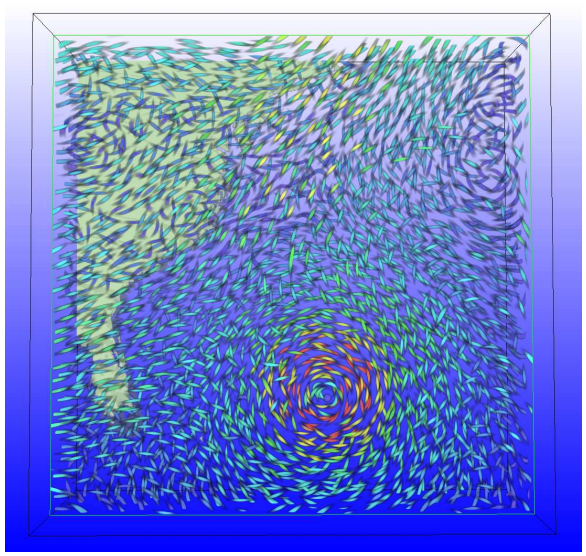
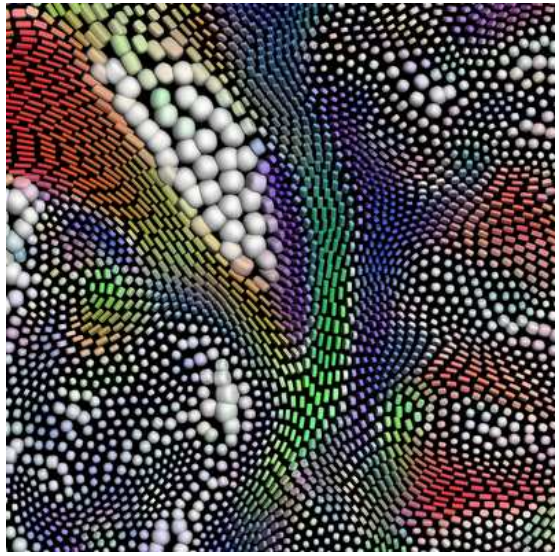


Data attributes

- Quantities that express physical values or fields.
  - Scalars (temperature, pressure)
  - vectors (Velocity, vorticity)
  - tensors (strain, rotation)
  - normals
  - texture coordinates



Visualization techniques

Scalars, Vectors and Tensors

Techniques for visualizing scalars

- 2D data
  - Assigning colors (color mapping)
  - Contouring
  - Carpet plots
- 3D data
  - Cutting planes (with 2D techniques)
  - 3D contouring (iso-surfaces)
  - Volume visualization

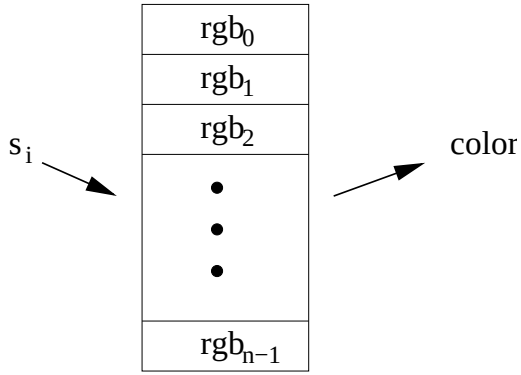
Color mapping

- Given a scalar field S then certain values of S can be given a color through a mapping.
- The scalars are divided into n equal intervals and serve as indices into a lookup table.
- The lookup table holds an array of of colors (represented as RGBA or HSVA).

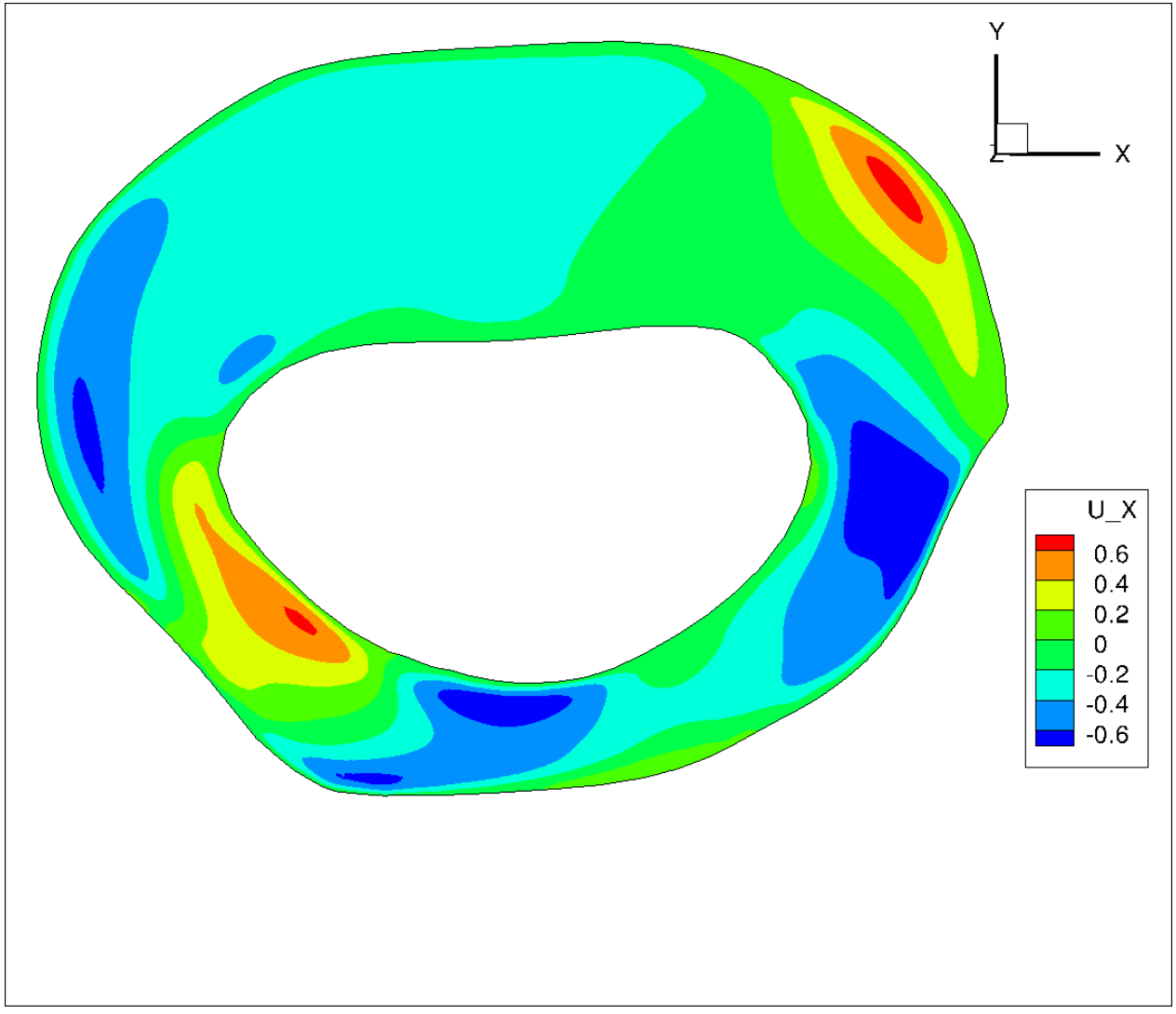
$$s_i < \min,\ i = 0$$

$$s_i > \max,\ i = n-1$$

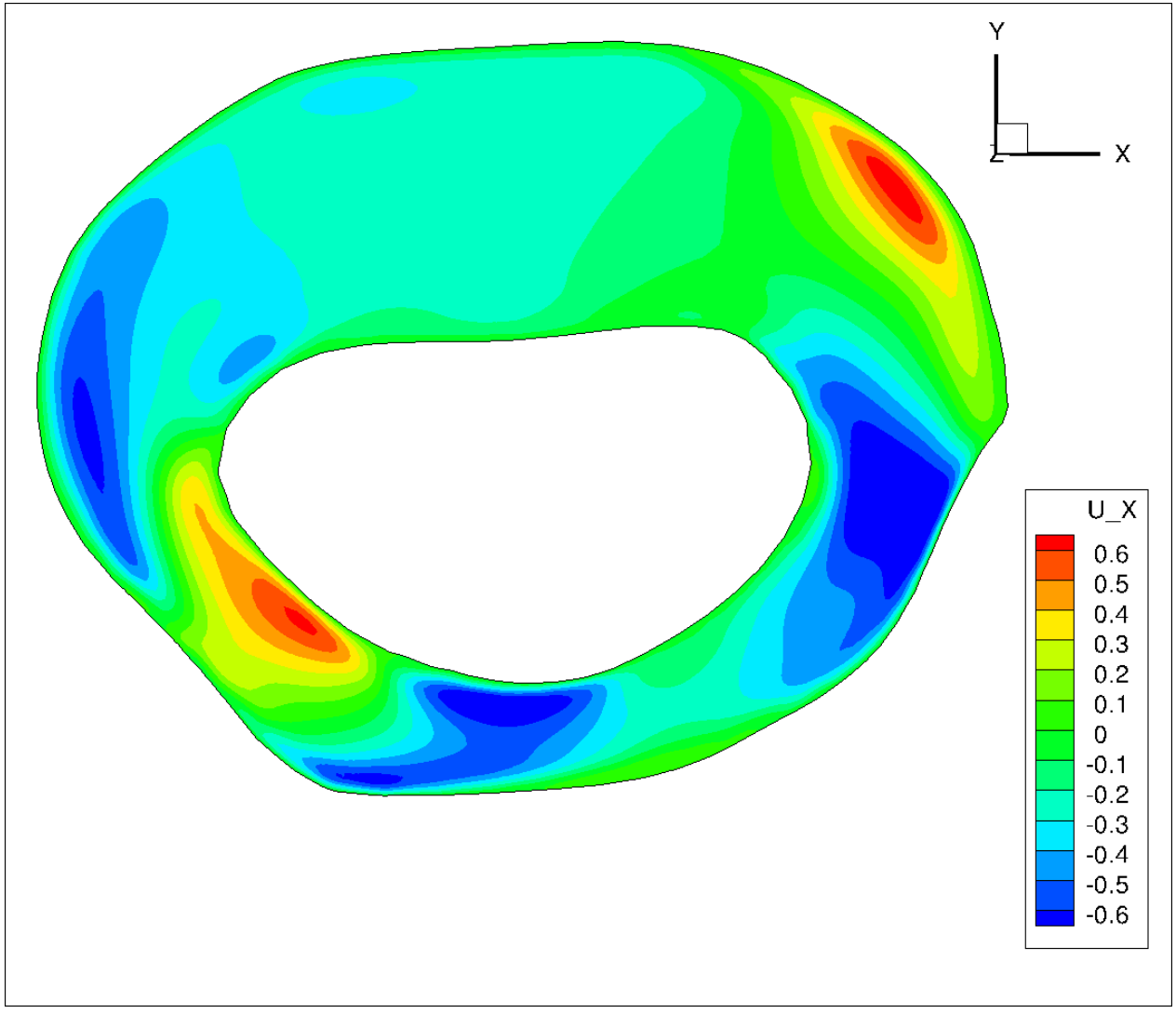
$$i = n\left(\frac{s_i - \min}{\max - \min}\right)$$



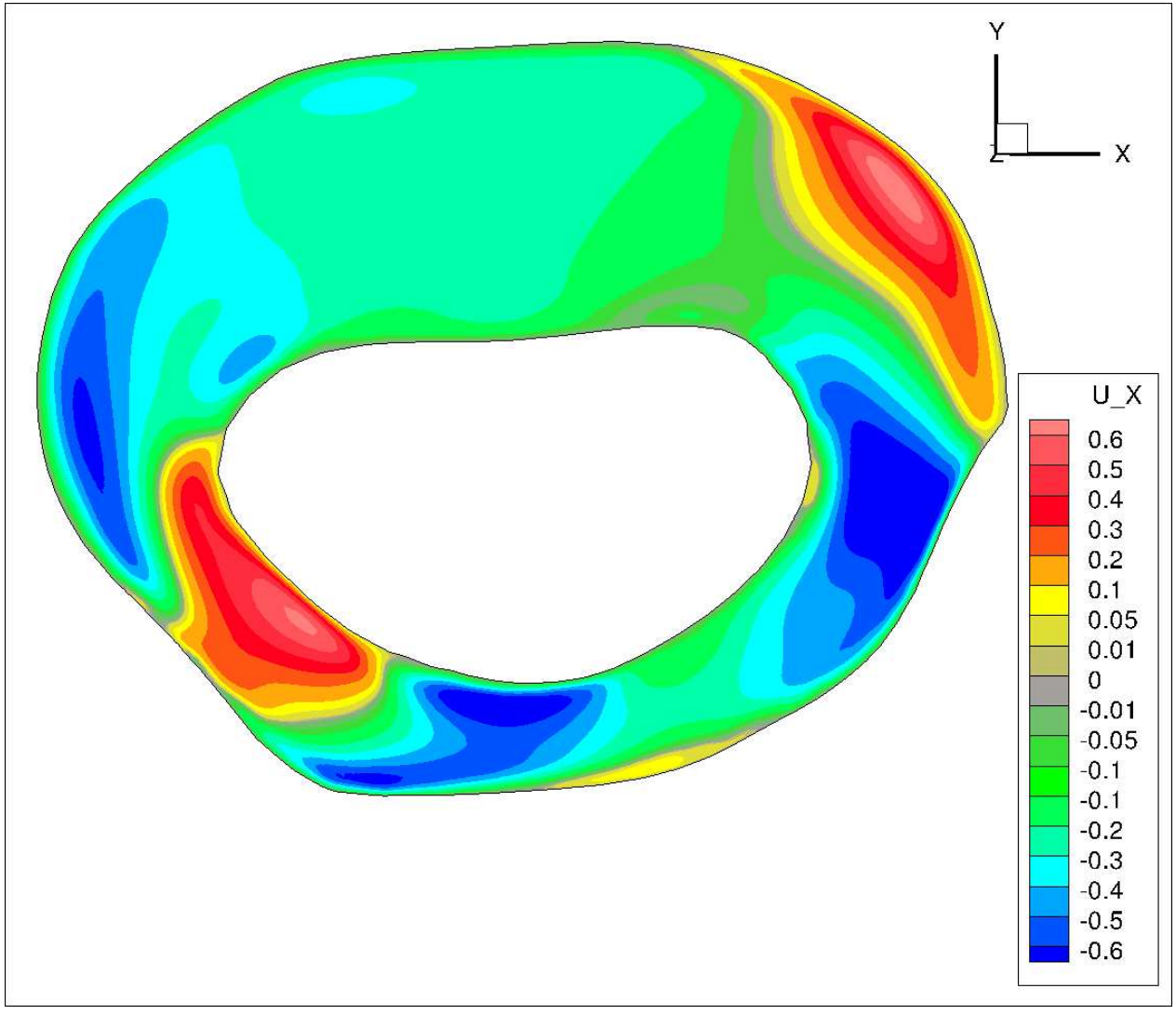
Color mapping



Color mapping



Color mapping



Use of colors

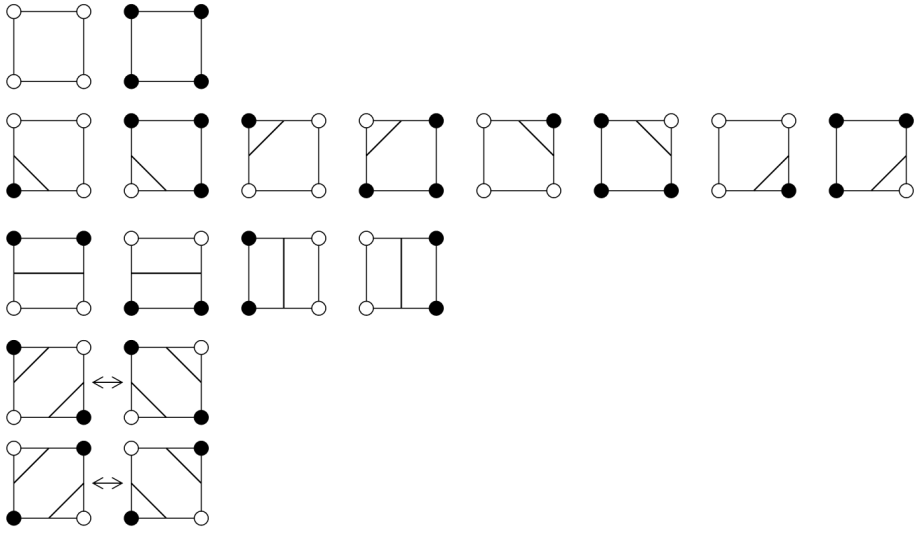
- Use of colors is important in visualization and can be used to emphasize various features in the data.
- Making an appropriate color table that communicate relevant information can be a challenging task.
- "Wrong use" of colors may exaggerate unimportant details.

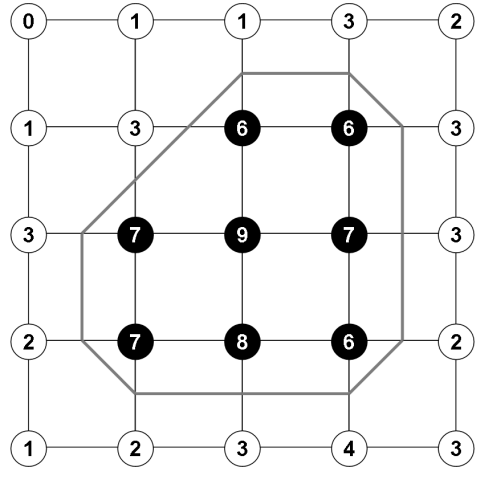
Contouring

- Contouring is a technique often seen in maps.
- Can be used to depict boundaries separating regions with similar data values.
- A single contour level shows a line of constant value (such as constant elevation in topological maps)

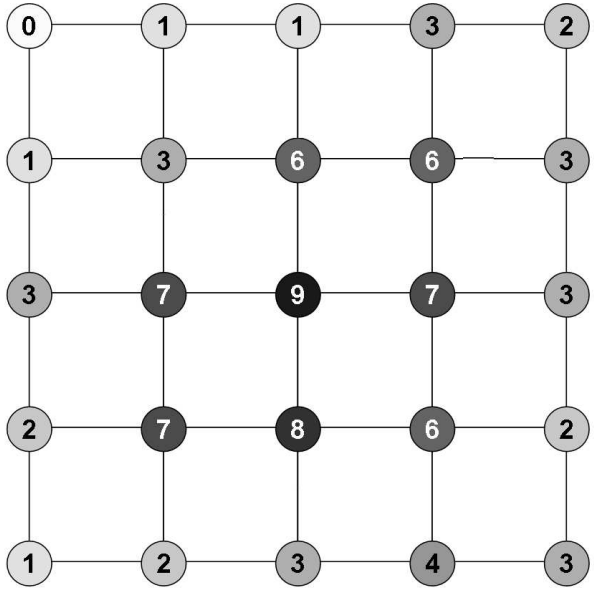
Marching squares

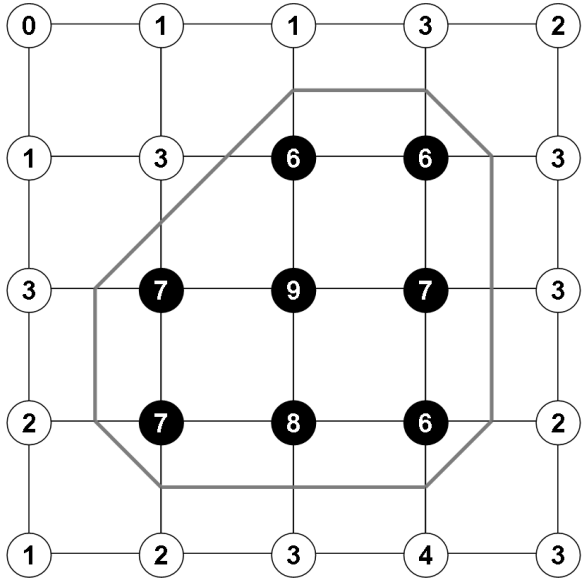
- A contouring techniques that treats each cell separately instead of tracking individual contour lines
  1. Select a cell
  2. Calculate the inside/outside state of each vertex of the cell.
  3. Create an index by storing the binary state of each vertex in a separate bit.
  4. Use the index to look up the topological state of the cell in a case table.
  5. Calculate the contour location (via interpolation) for each edge in the case table.

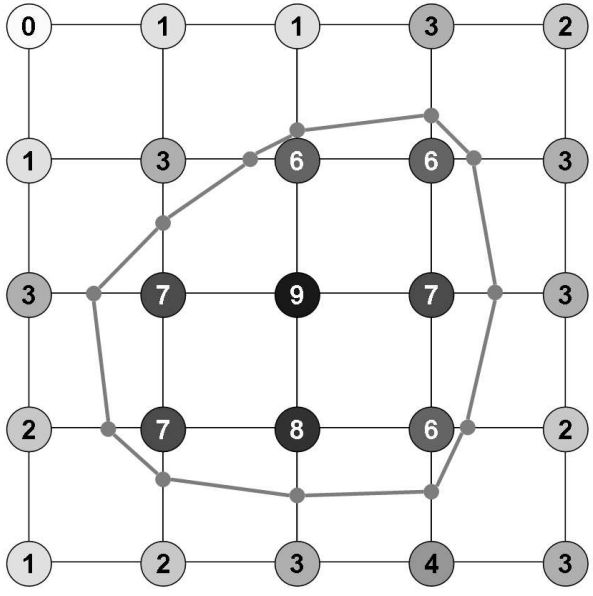




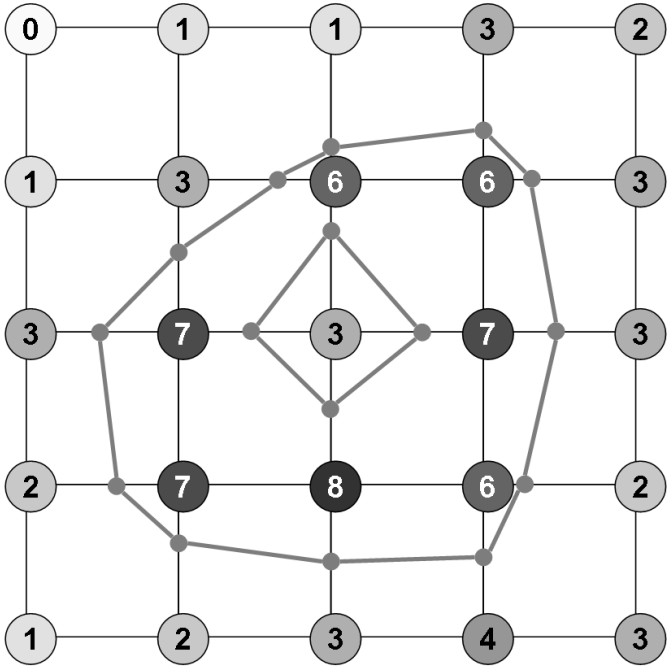
Example: Marching squares

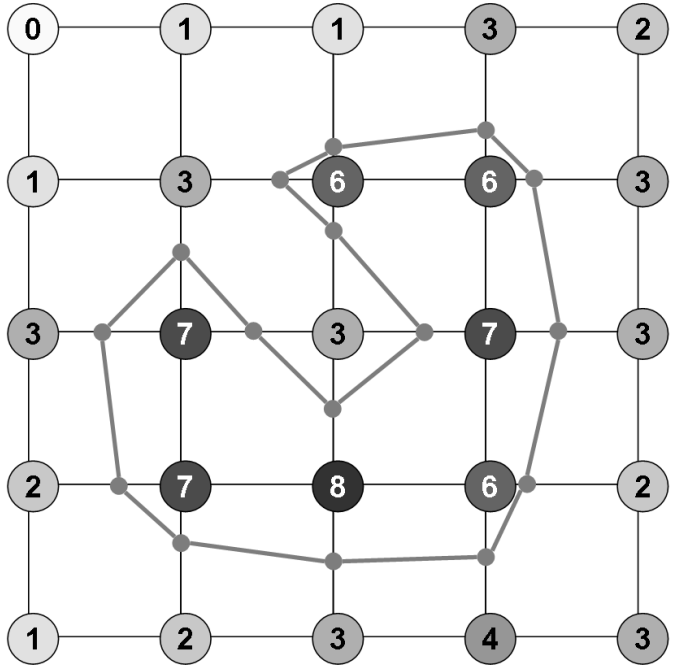


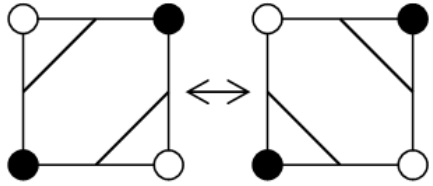




Contouring ambiguity

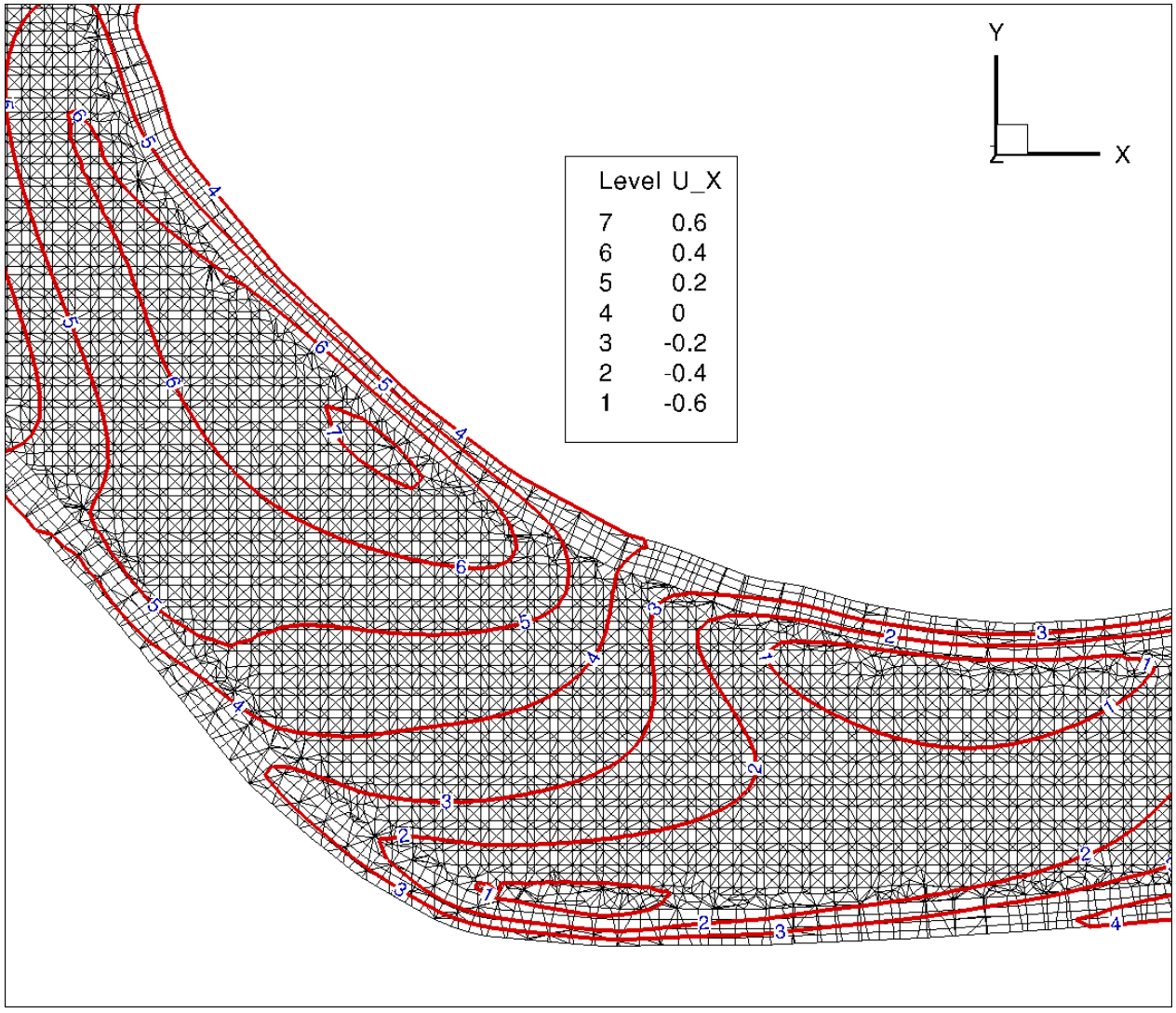




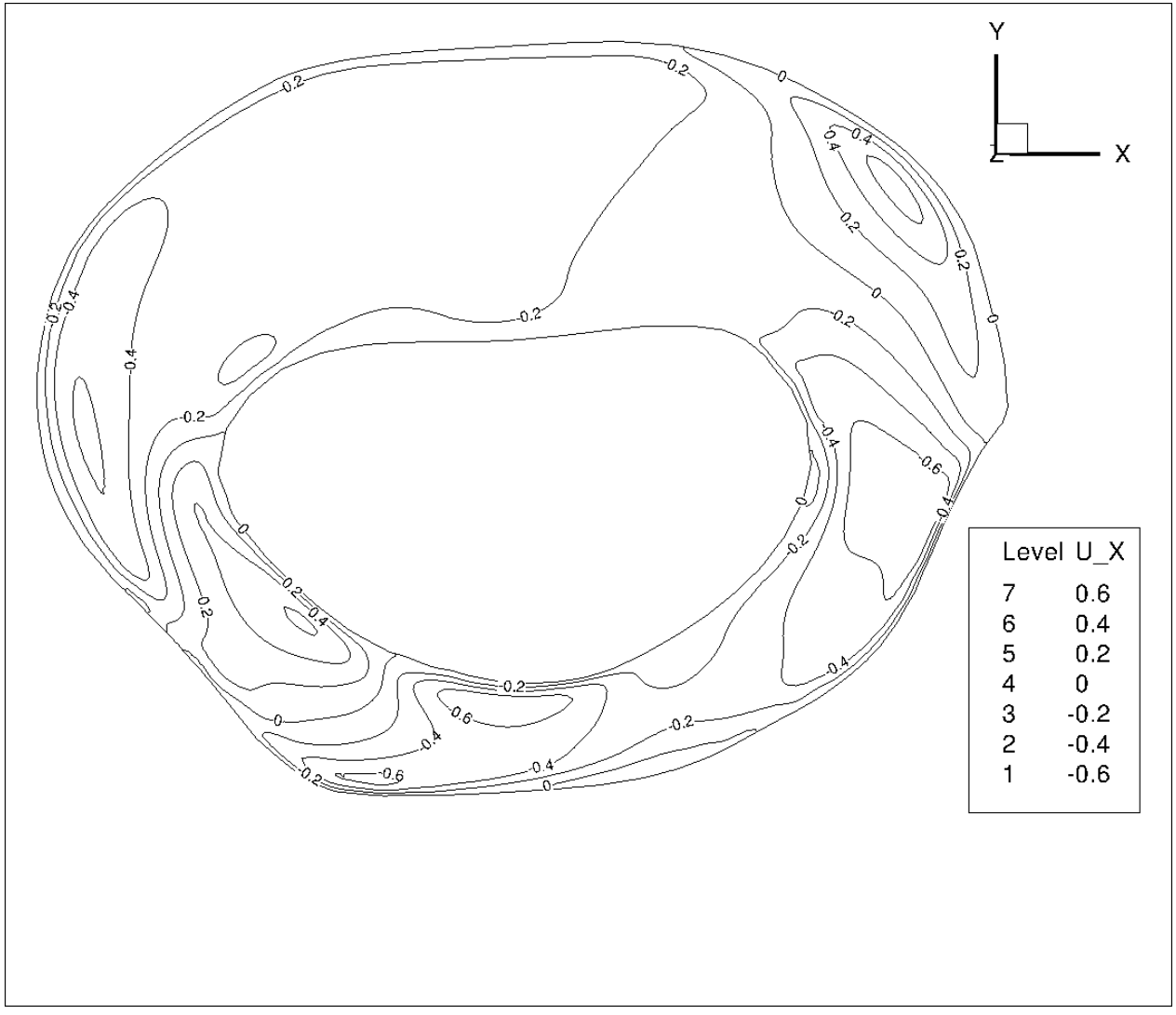


- In 2D, either choice is acceptable and leads to a closed curve.

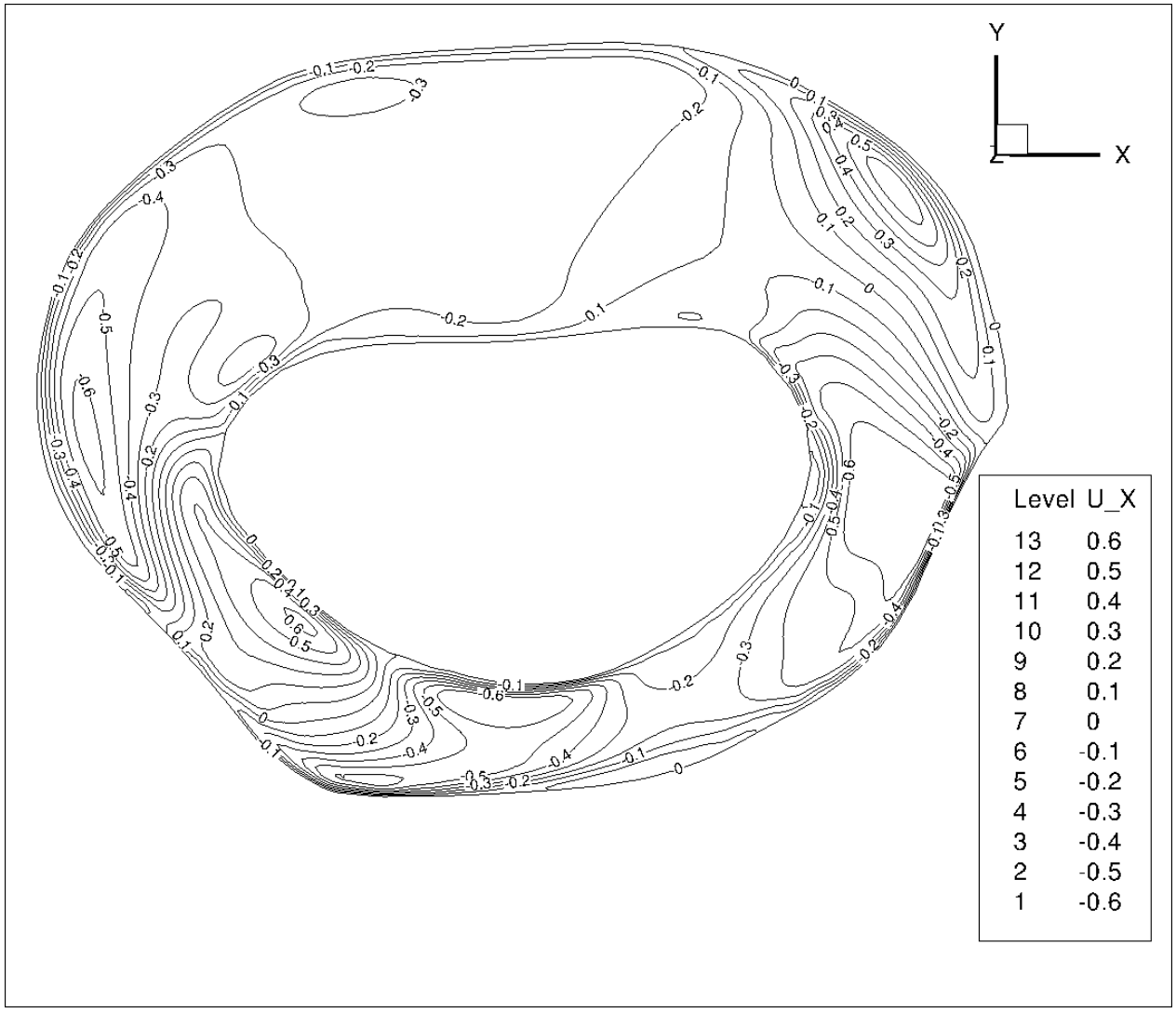
Contouring examples



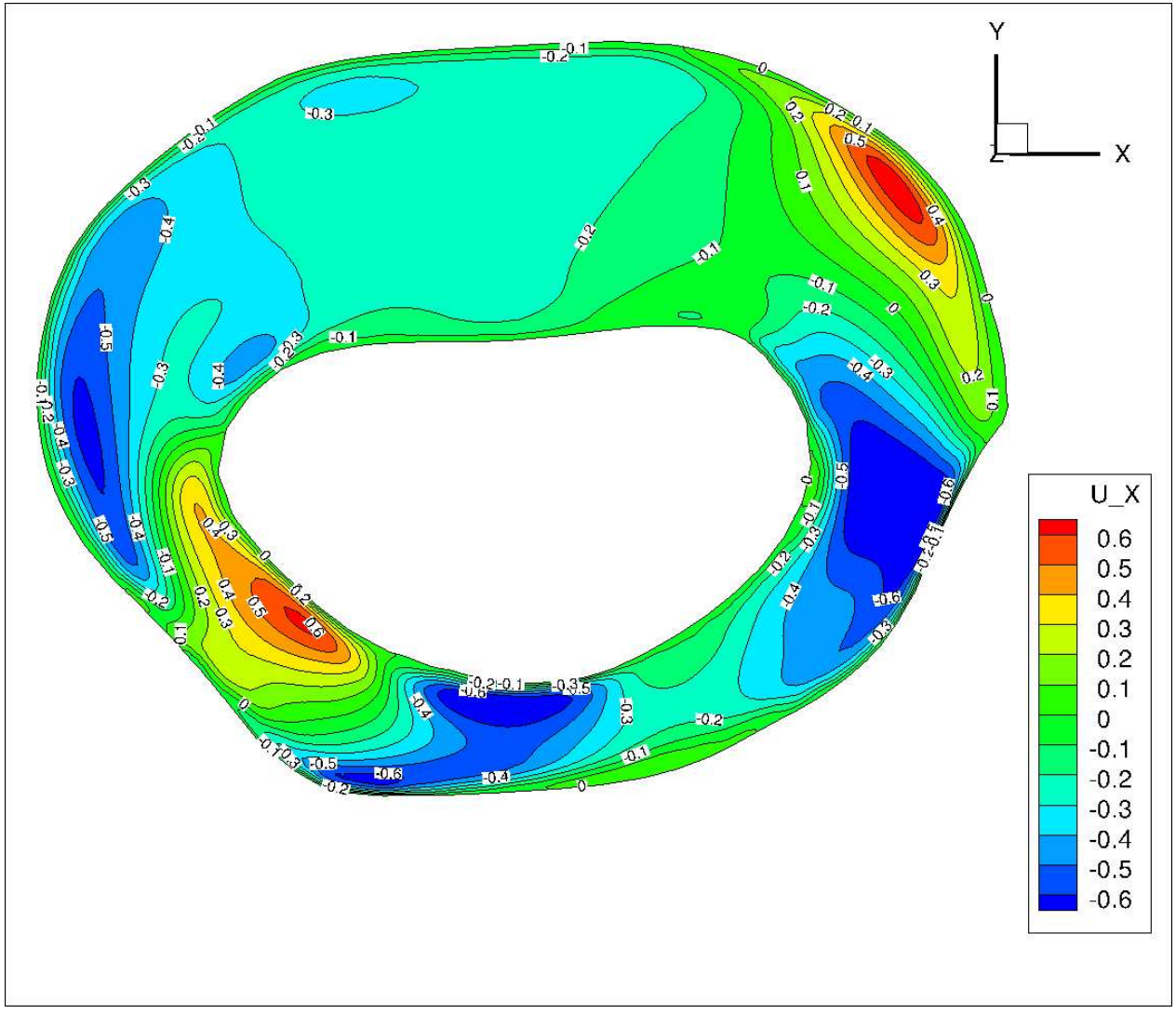
Contouring examples



Contouring examples

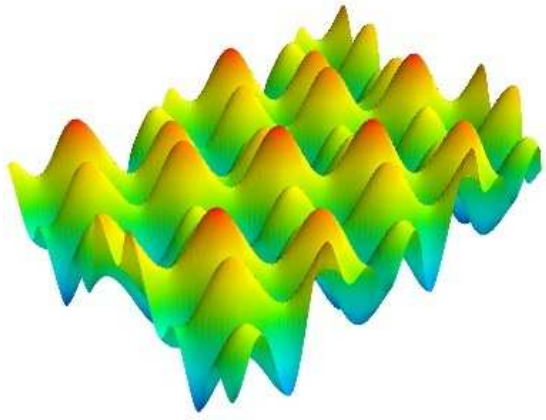


Contouring examples



Carpet plots

- Depicts the 2D scalar field as a surface (in 3D) with the z-axis represented through elevation of the data values
- Can also be used to visualize to variables simultaneously: One through colors and one through elevation.

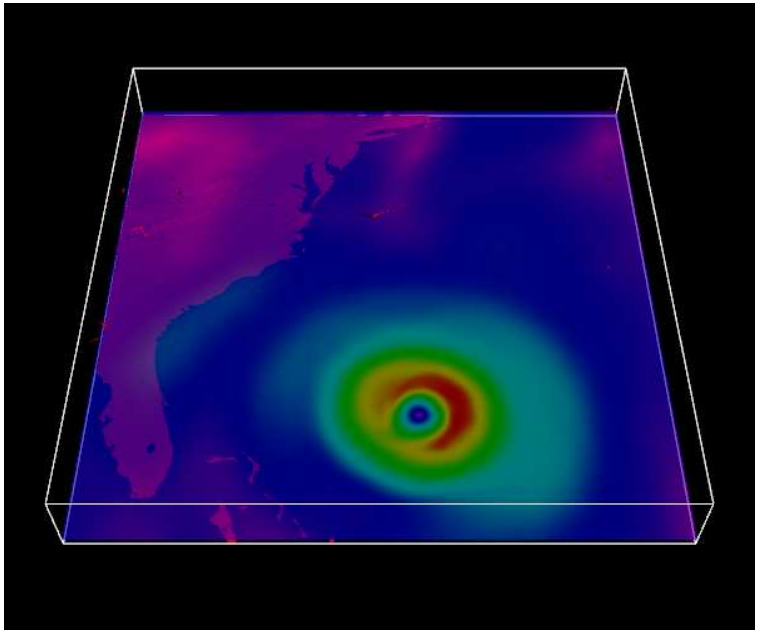


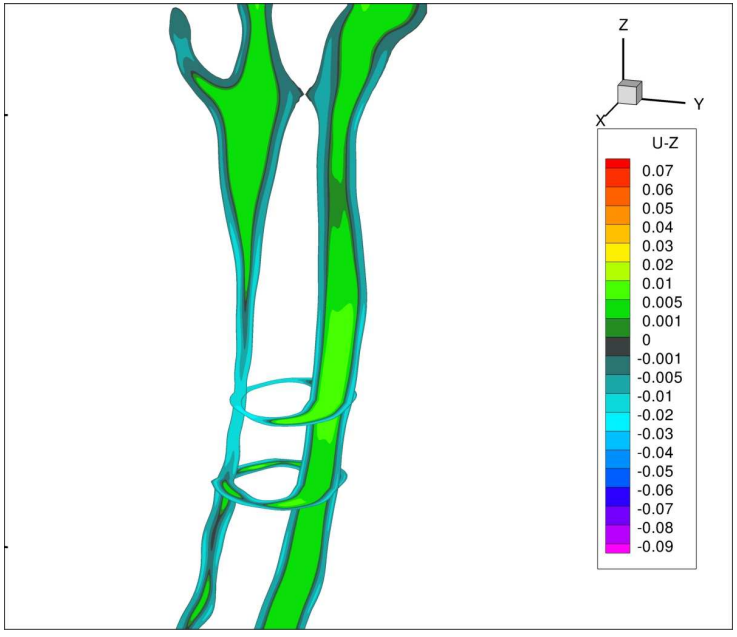
Visualizing scalars in 3D

- Cutting planes (with 2D techniques)
- 3D contouring (iso-surfaces)
- Volume visualization

Cutting planes

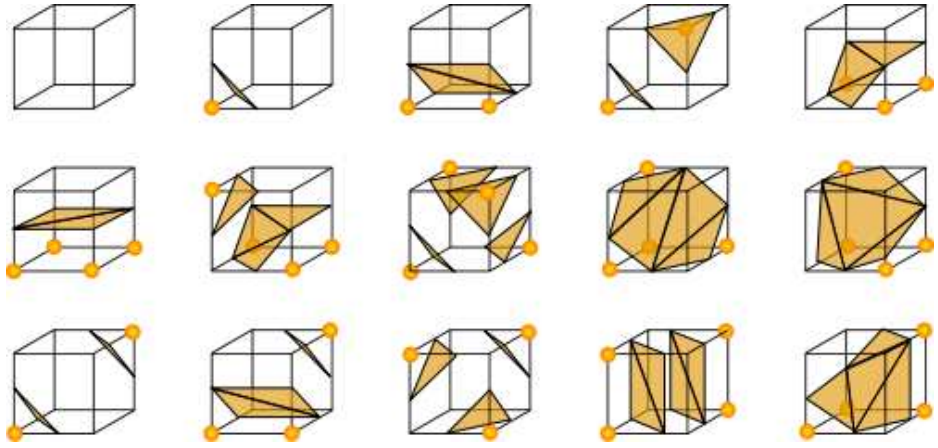
- 3D scalar fields can be visualized by a single or multiple cutting planes intersecting the data volume.
- Standard 2D techniques can then be applied to the cutting surfaces.
- A data volume can be inspected by interactive use of cut planes.
- The technique does not display 3D features. Only shows a small part of the data.



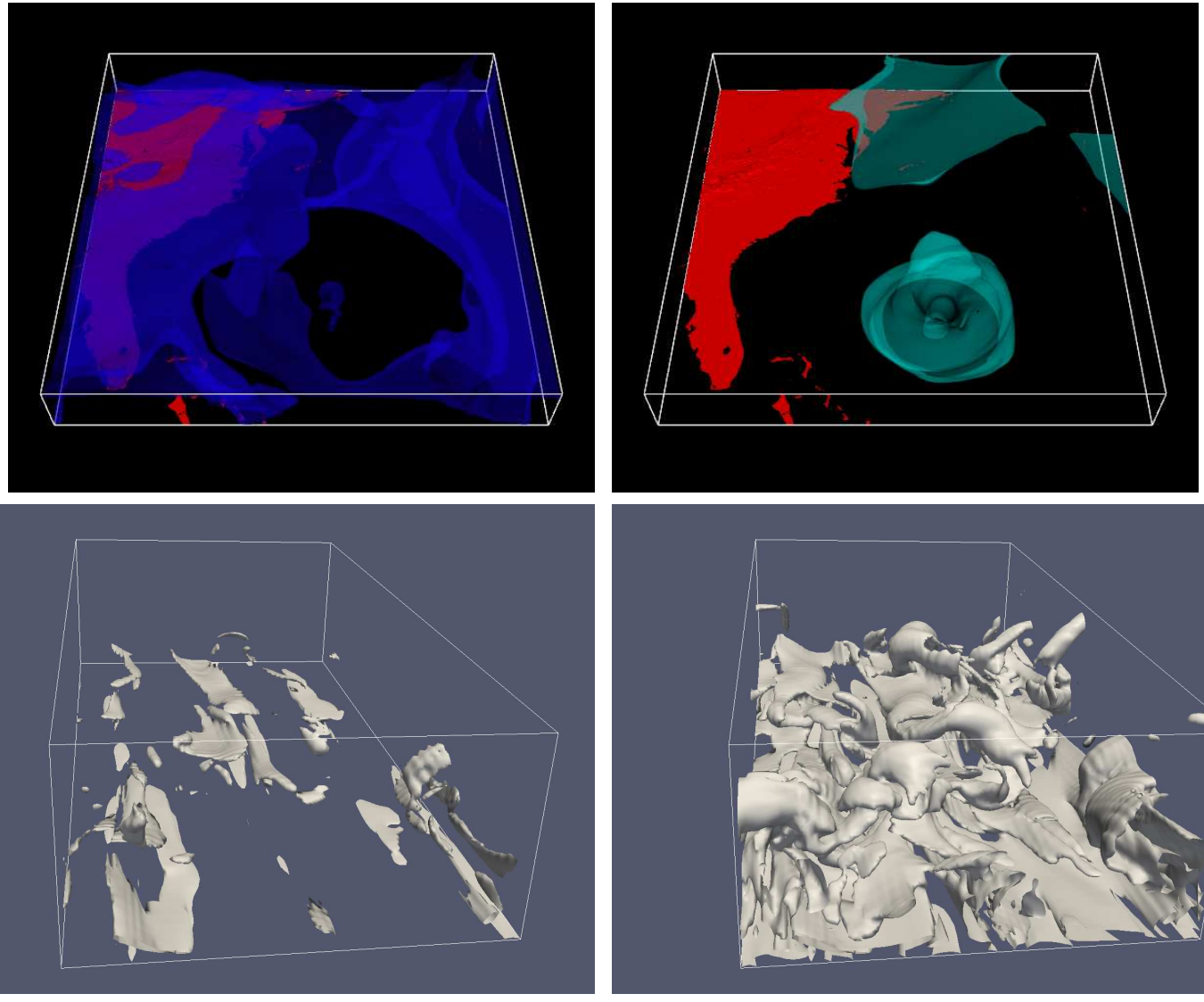


3D contouring

- 3D contouring computes surfaces (iso-surfaces) with constant value instead of curves.
- In 3D, marching squares becomes marching cubes. Can be extended to other cell types such as tetrahedra, pyramids.
- Here, the case table consists of 256 possible combinations. Can be reduced to 15 cases due to symmetry.
- In 3D, the problem of ambiguity is more complex. Arbitrary selection from the case table may lead holes in the surface.

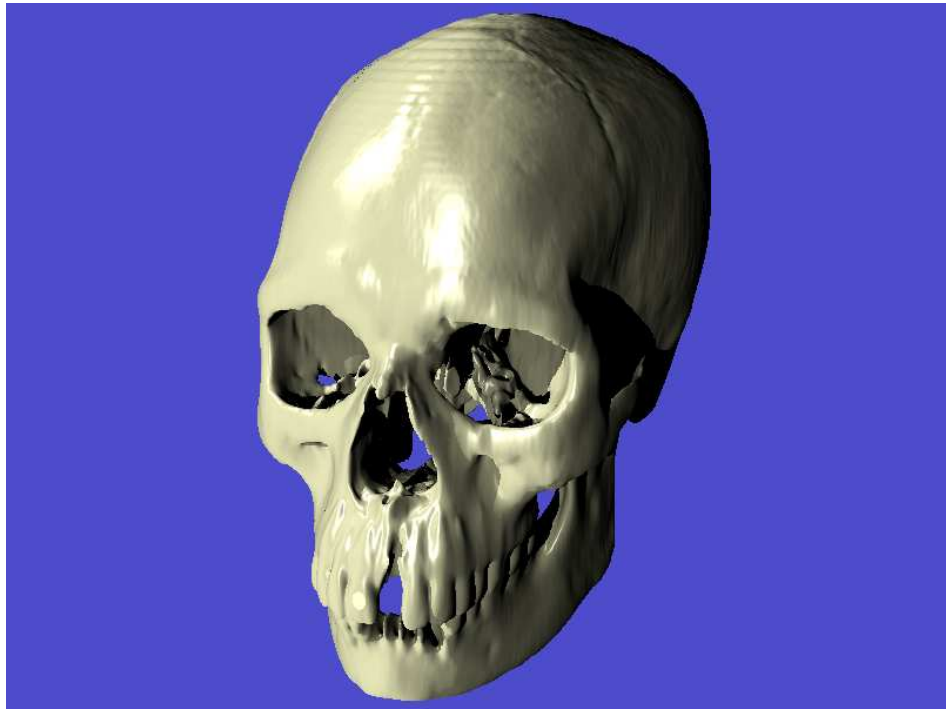


Iso-surfaces

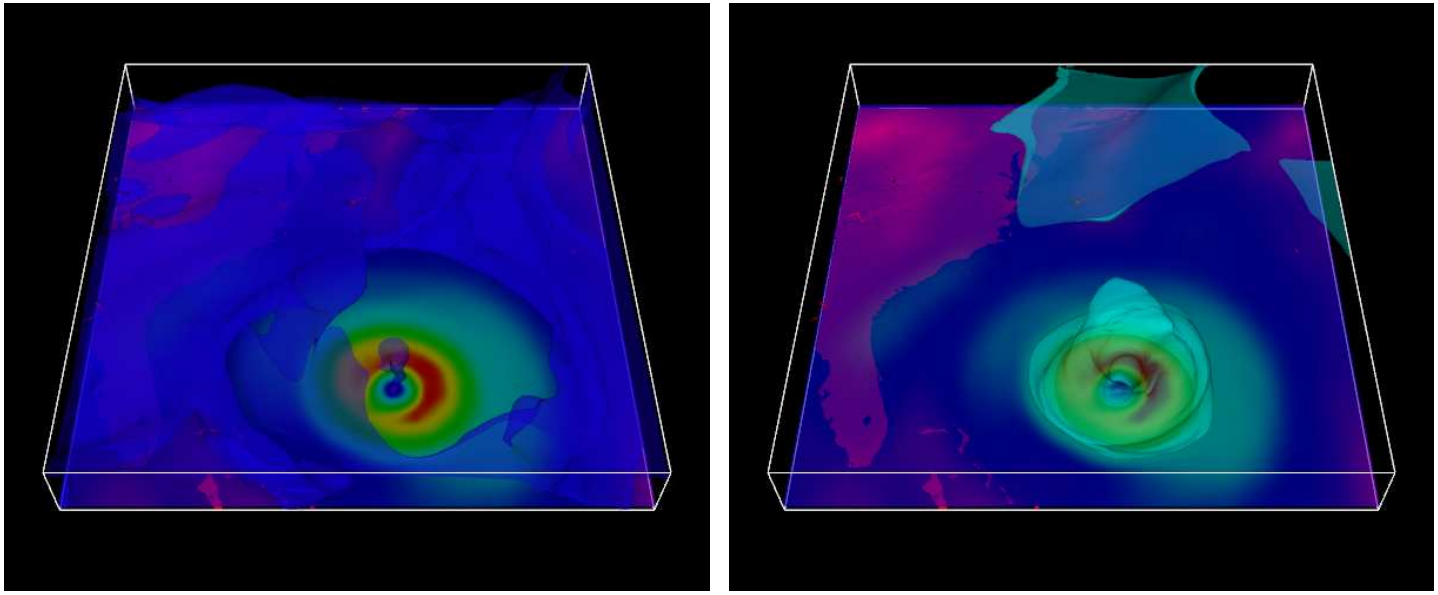


Iso-surfaces

- A good technique if there exists clear boundaries in the data.
- **Challenge:** Can be difficult finding proper iso-values.

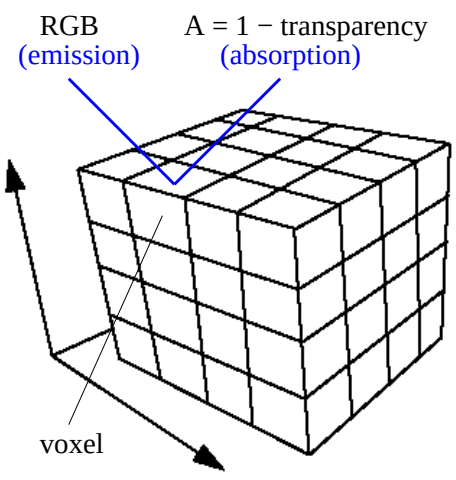


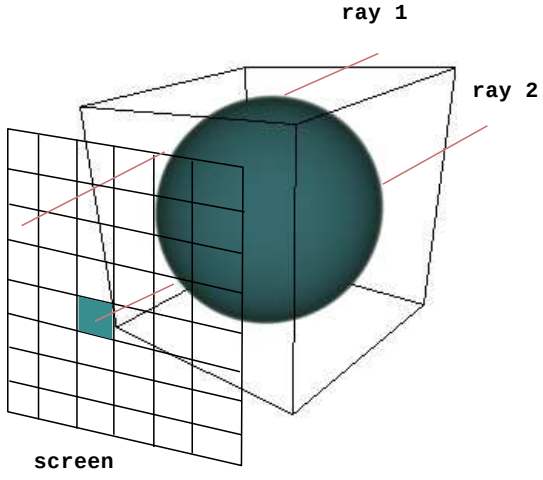
Iso-surfaces and cutting planes

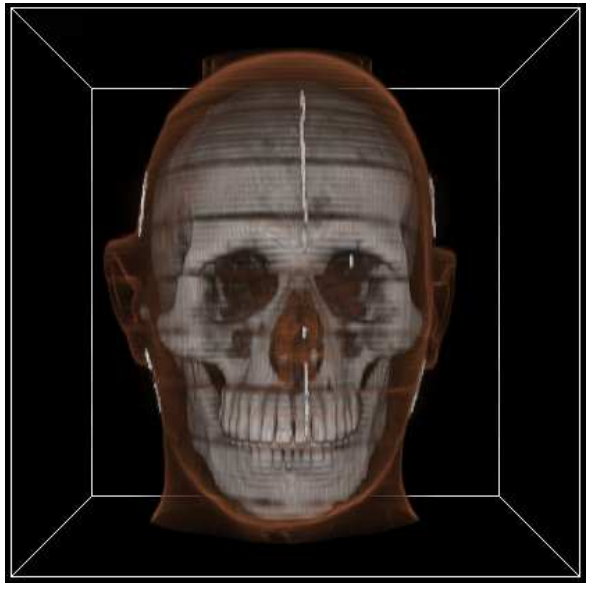


Volume rendering

- Technique for visualizing volume data
- Based on light transport models
- VRI: $I(D) = I(0)T(0, D) + \int_{s=0}^{D} g(s)T(s, D)$
- DVRI: $I(D) = \sum_{i=0}^{n} C_i A_i \prod_{j=i+1}^{n}(1 - A_j)$







Volume rendering

Compositing Schemes

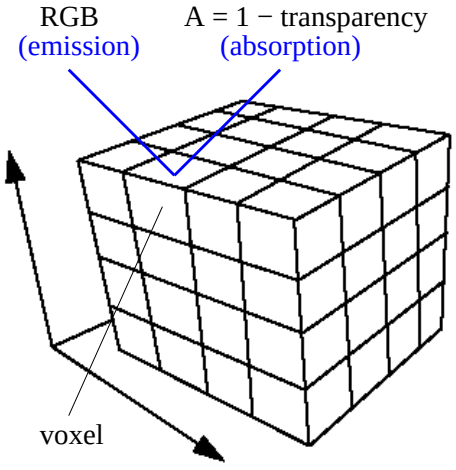
- Back-to-front

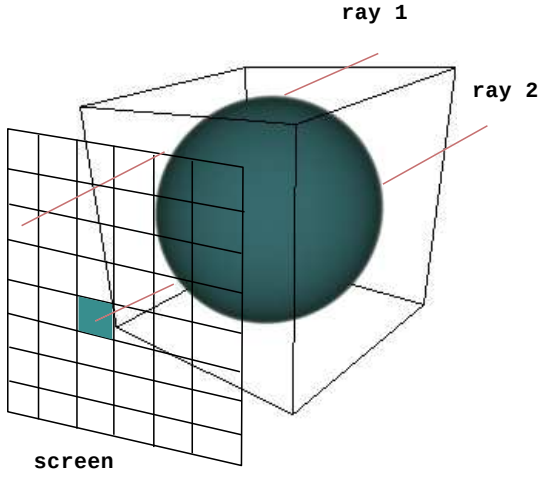
$$\tilde{C}'_i = A_i C_i + (1 - A_i)\tilde{C}'_{i-1}, \quad A'_i = A_i + (1 - A_i)A'_{i-1}$$

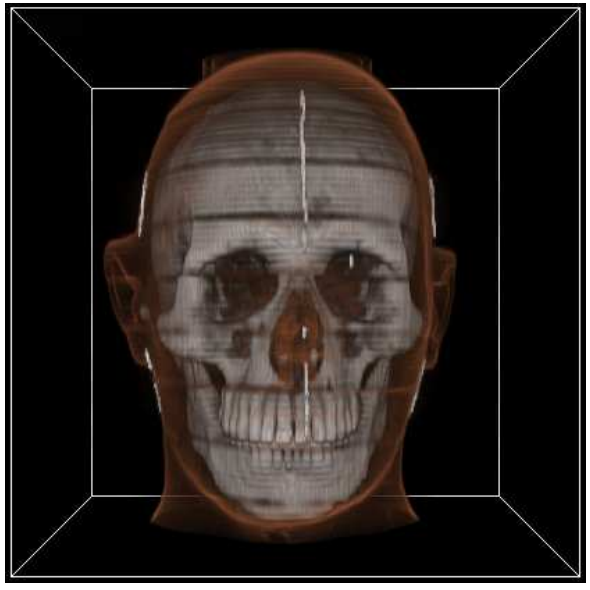
- Front-to-back

$$\tilde{C}'_i = \tilde{C}'_{i+1} + (1 - A'_{i+1})A_i C_i, \quad A'_i = A'_{i+1} + (1 - A'_{i+1})A_i$$

$$(\tilde{C}_i = A_i C_i)$$

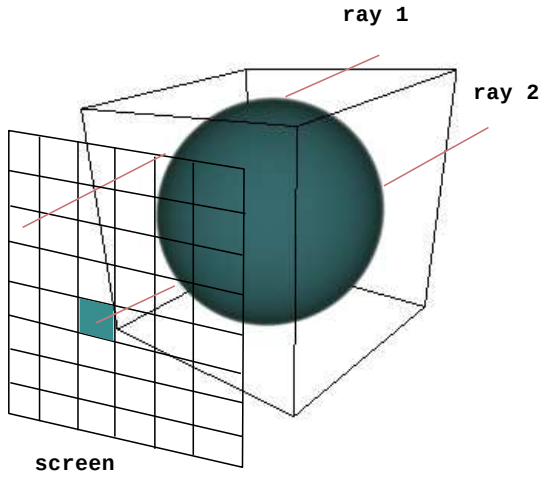




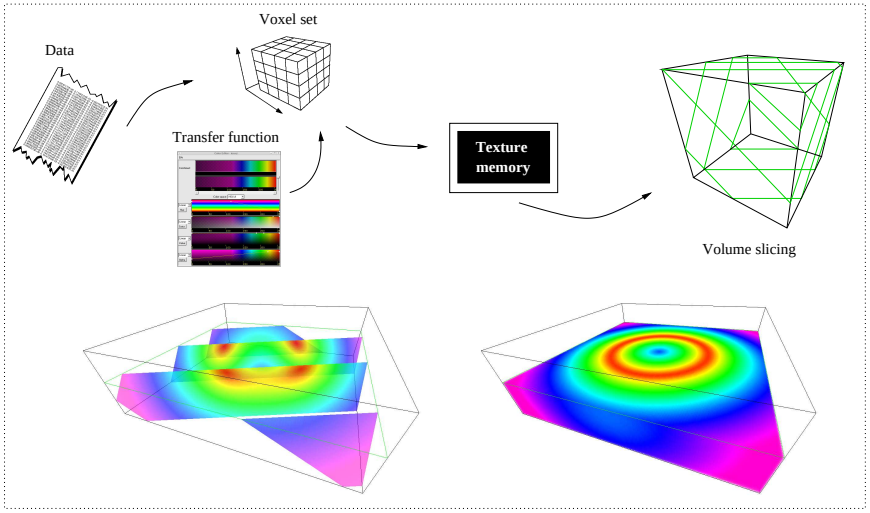


Volume rendering

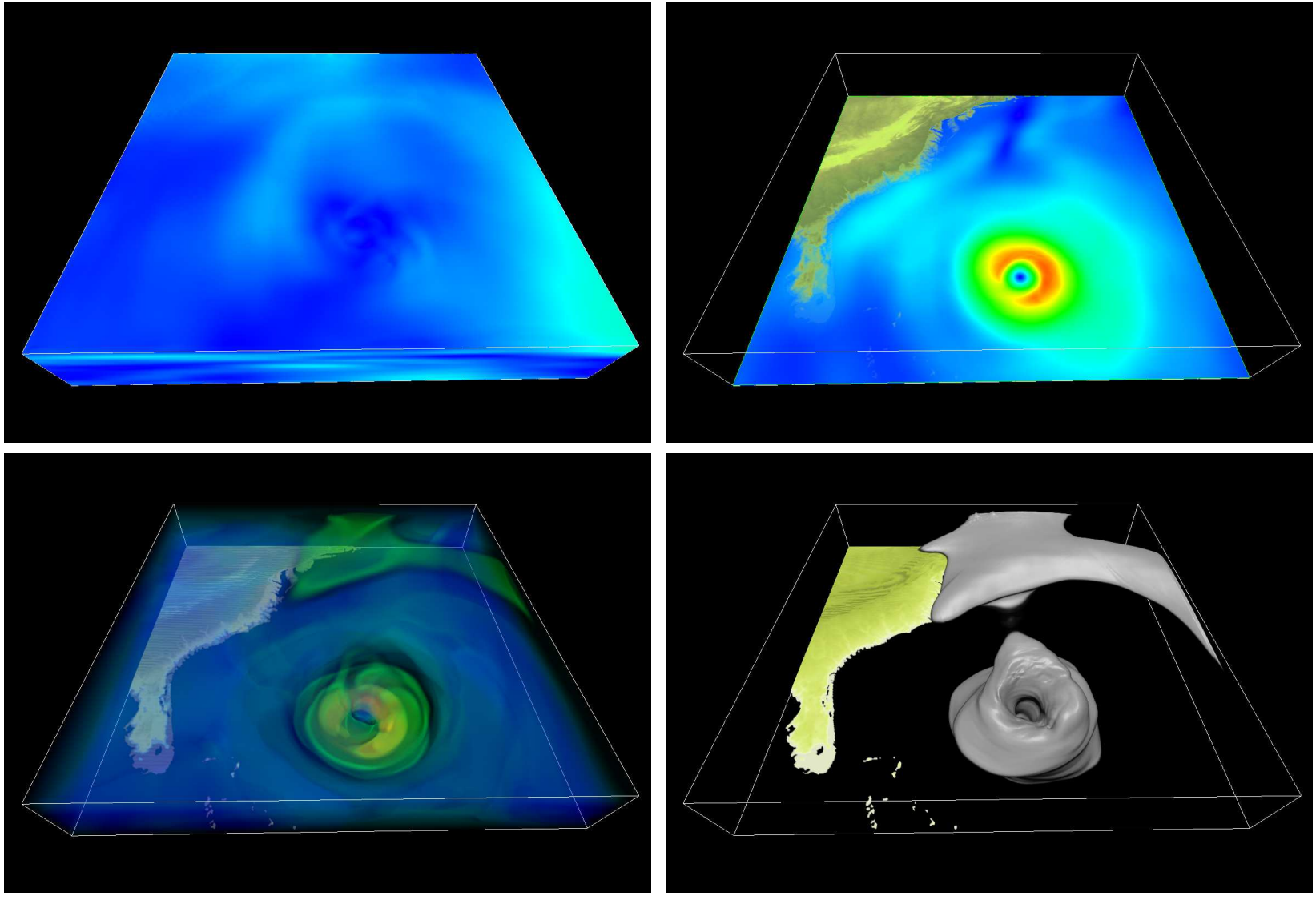
Ray Casting



Texture-Based Volume Rendering



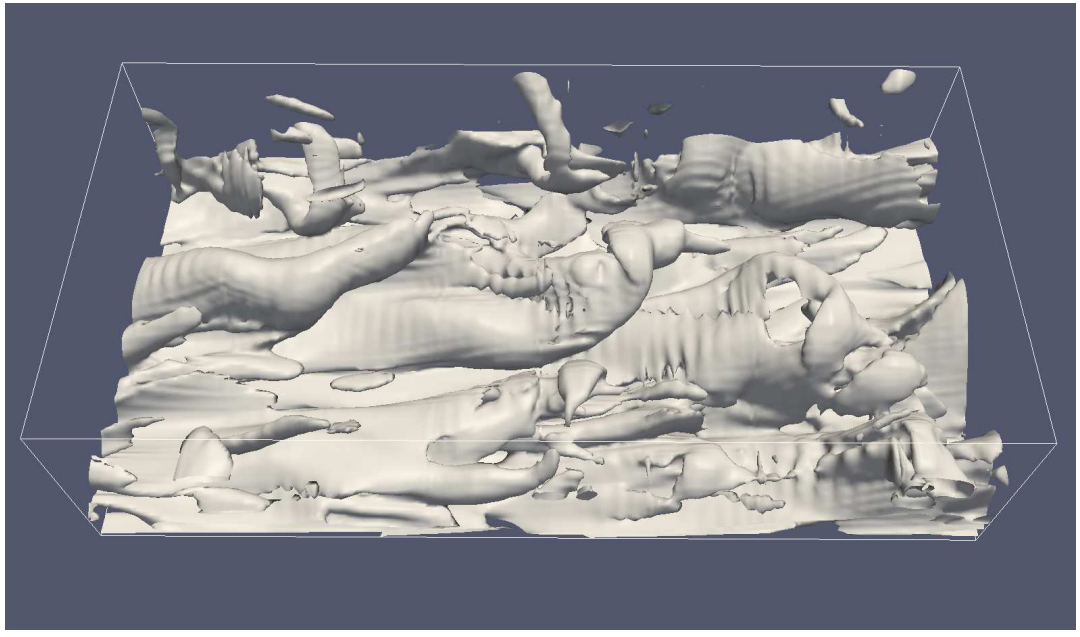
Volume rendering

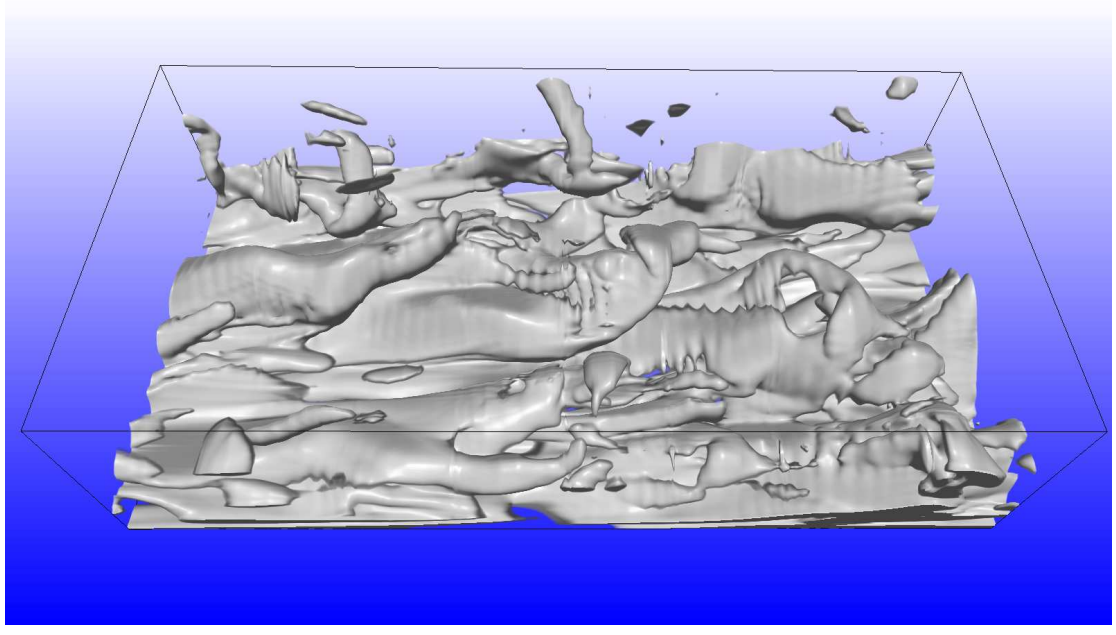


Volume rendering

- Well-suited for interactive investigation of 3D data
- Can emulate other techniques such as iso-surfaces and cutting planes by manipulation of opacity values and by using clip planes
- Excellent for displaying 3D features in the data volume
- Can be used both to display clear boundaries in the data as well as visualizing more "fuzzy" objects such as gasses or features in the data without a clear boundary.

Comparison: Volume rendering v.s. Iso-surface





Outline

- Basic data representation
- Techniques for visualizing
  - Scalars
  - Vector
  - Tensors

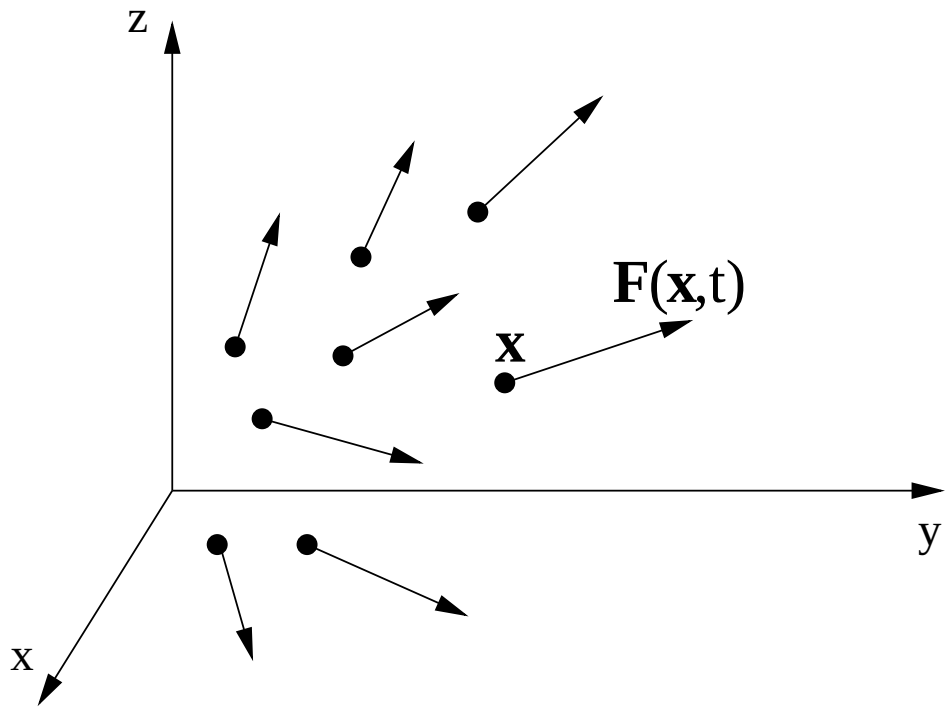
Techniques for visualizing vector fields

- Direct visualiztion
  - Glyph-based techniques
- Curve-based techniques
- Texture-based techniques
- Feature-based techniques
  - Topology-based techniques
  - Feature-detection techniques
  - Clustering techniques

Vector field

A *vector field* is defined by a map

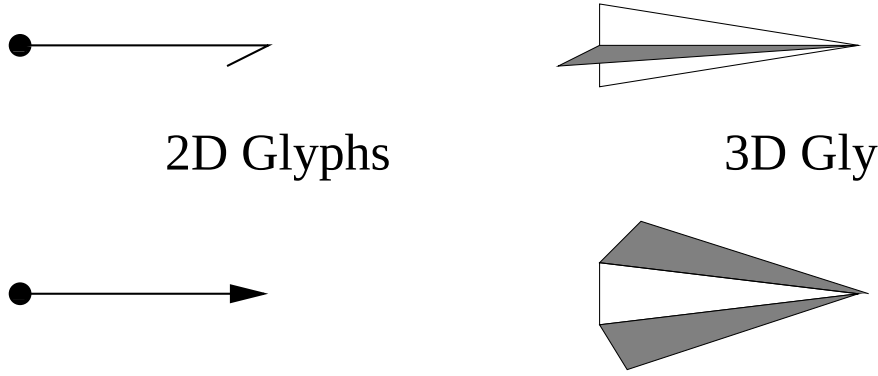
$$\boldsymbol{F} : \Omega \times T \to \mathbb{R}^n, \quad (\mathbf{x}, t) \mapsto \boldsymbol{F}(\mathbf{x}, t).$$



A vector field $\boldsymbol{F}$ assigns a vector $\boldsymbol{F}(\mathbf{x}, t)$ to each point $\mathbf{x}$ of its domain at time t.

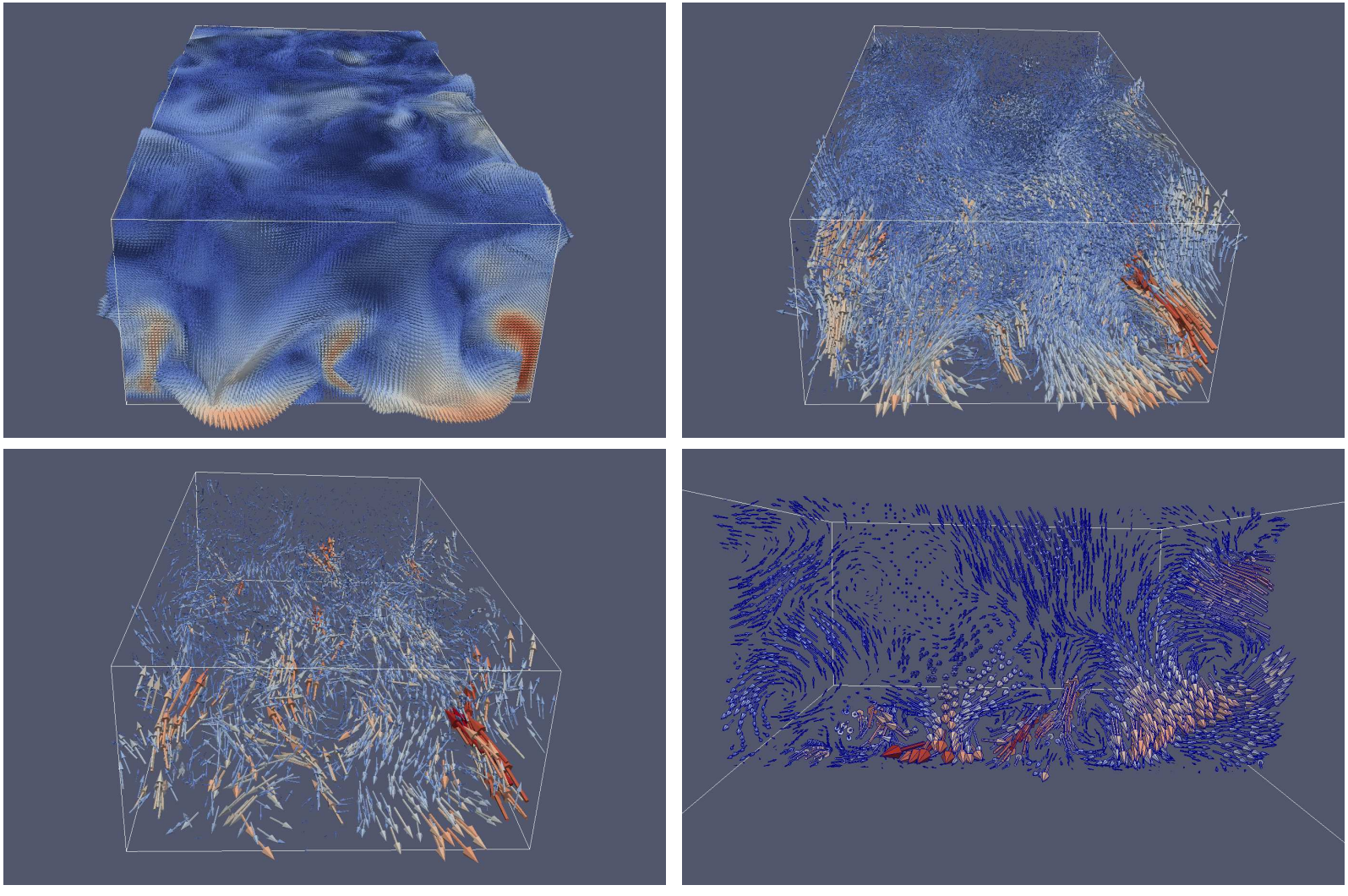
Glyph-based techniques

- Vector fields can be visualized directly by drawing the vector as arrows (or glyphs) directly at given points in the domain.

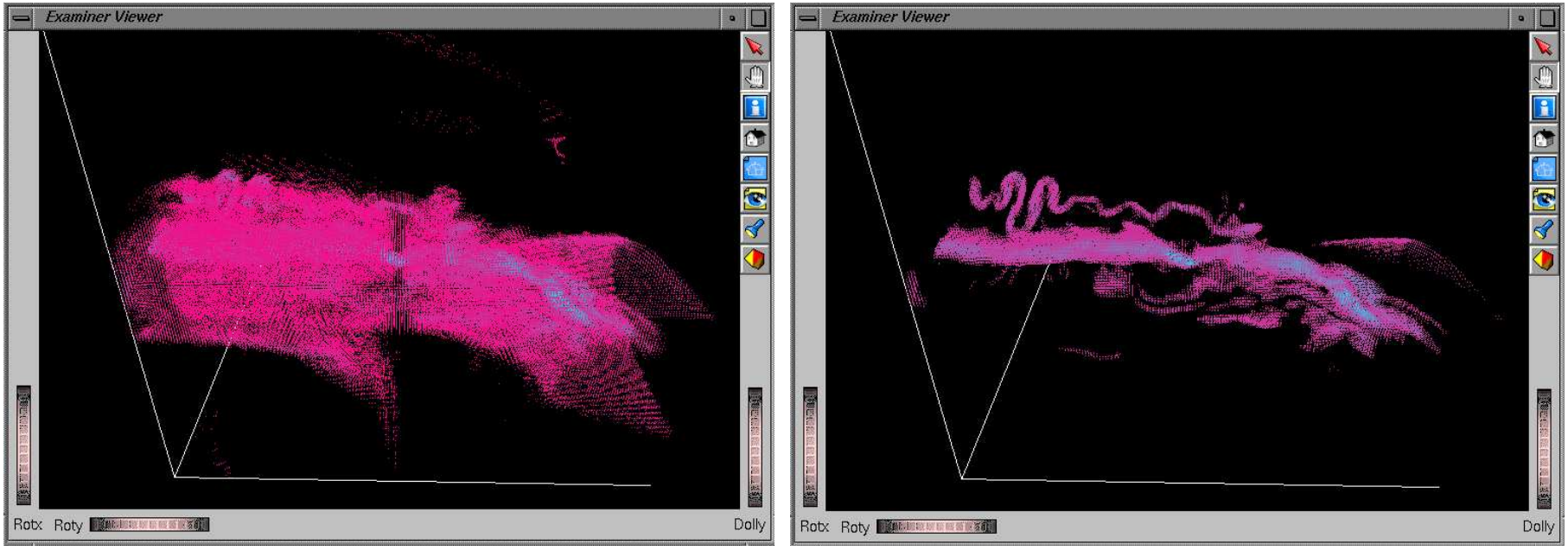


- Best suited for small datasets.
- If the placements of the glyphs are too dense or the variations in magnitude is to big, we get “cluttered” images.
- The result can be improved by using some form of thresholding (neglect the drawing of glyphs where the length of the vector is below a certain value, $\|v\| < c$).

Glyphs and cluttering



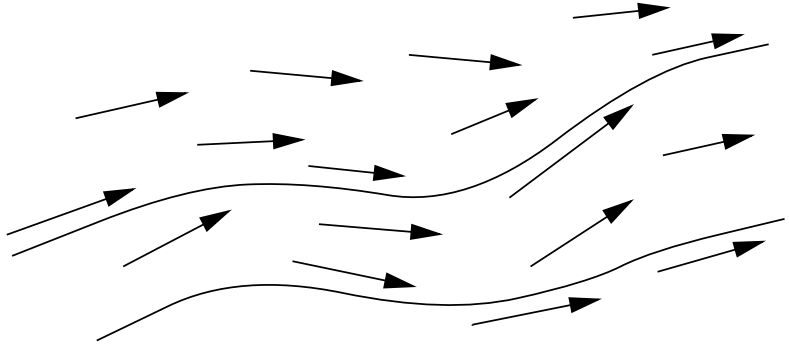
Threshold



Visualization of a vector field using glyphs with thresholding

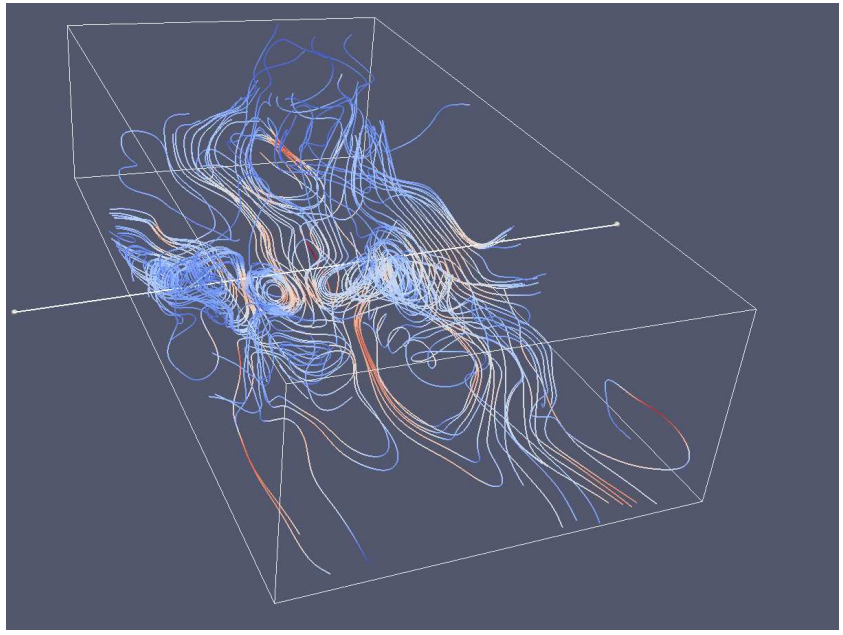
Curve-based techniques

- An improved representation of the vector fields can be obtained by drawing curves that reveal the orientation of the vector field.
- **Field lines:** $\frac{d}{ds}\mathbf{x}(s) = \boldsymbol{F}\big(\mathbf{x}(s), \vec{t}\big)$
- **Path lines:** $\frac{d\mathbf{x}}{dt} = \boldsymbol{u}\big(\boldsymbol{x}, t\big)$
- Both are curves $\mathbf{x}(s)$ where the tangent vectors of the curve coincide with the vector field.



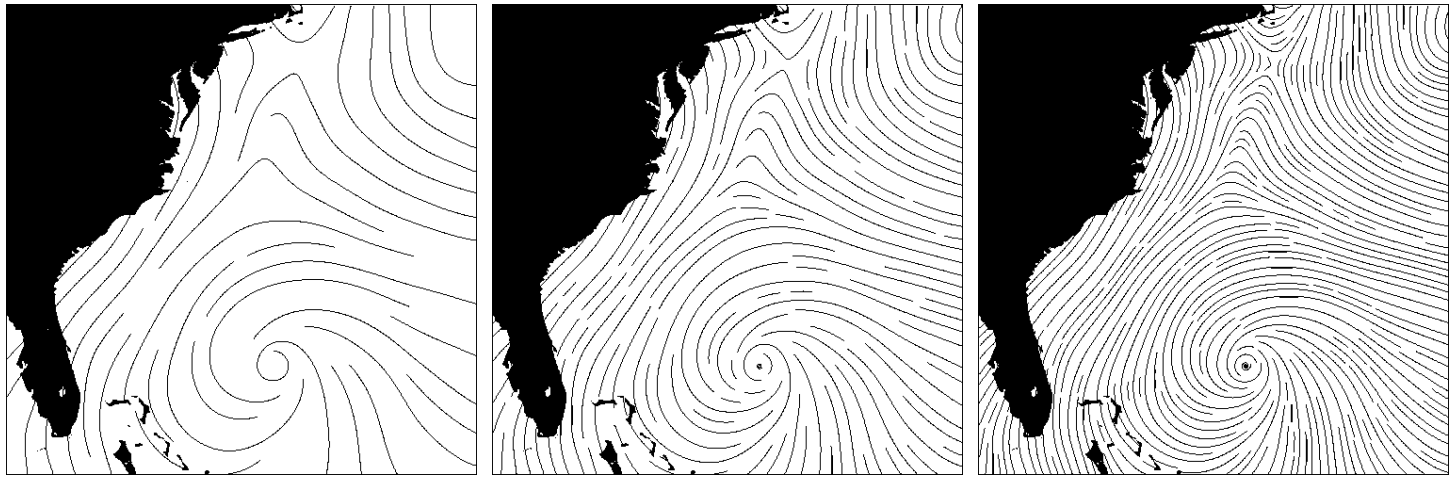
Curve-based techniques

- Limitation on the number of curves that can be displayed in the scene (without cluttering the image).
- This makes the visualization dependent on the choice of seed points.
- May loose important details in the field.



Seeding strategies

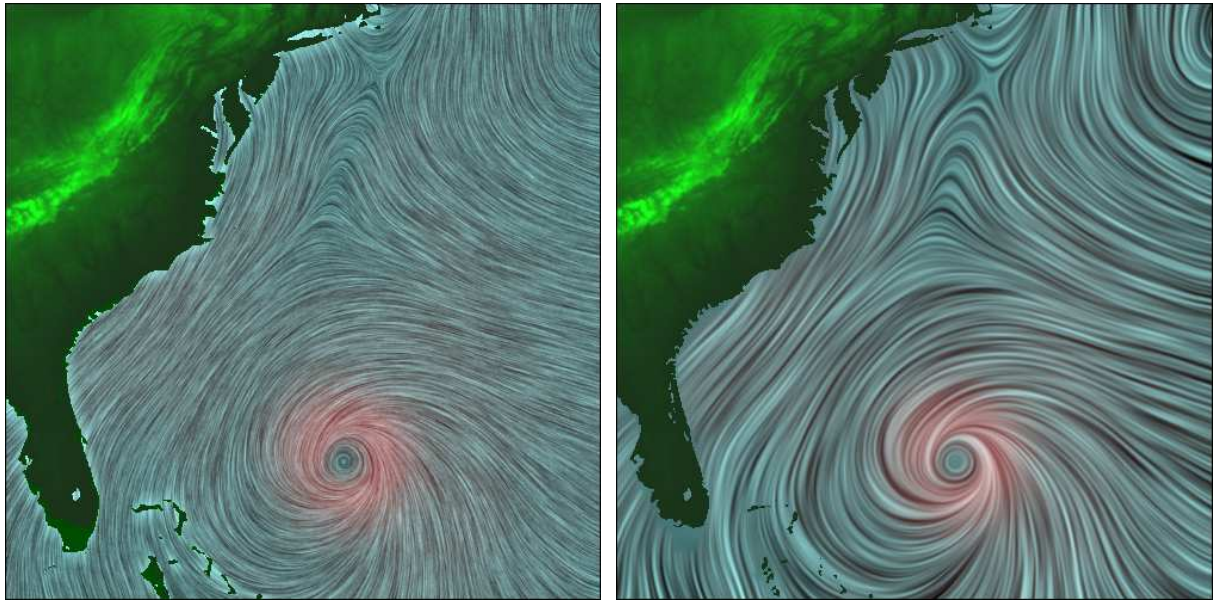
- It is not obvious how to distribute the field lines in space without either cluttering the image or missing important details in the field.
- To address the issue a number of seed point strategies have been proposed.



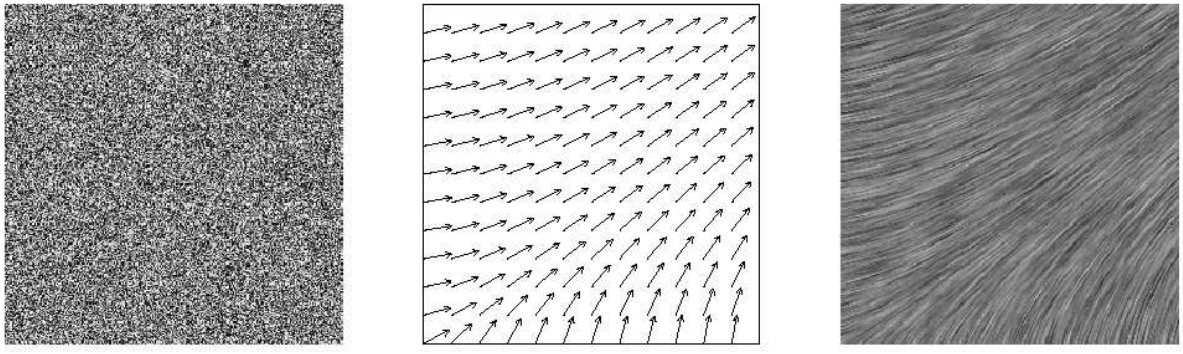
Evenly spaced field lines

Texture-based techniques

- Geometry-based techniques visualizes local flow features (using for instance glyphs and curves).
- Texture-based techniques attempt to create a continuous visual representation to illustrate the global behavior of the flow.
- In 2D, they depict the entire vector field with a dense representation, hence they are not dependent on seed points.

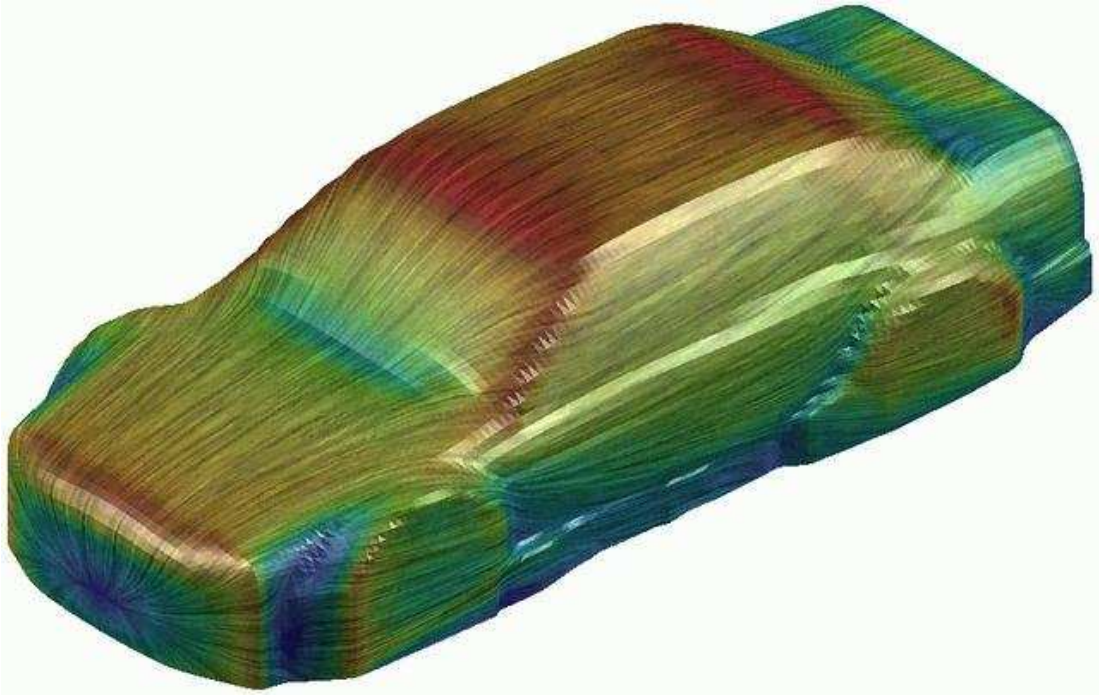


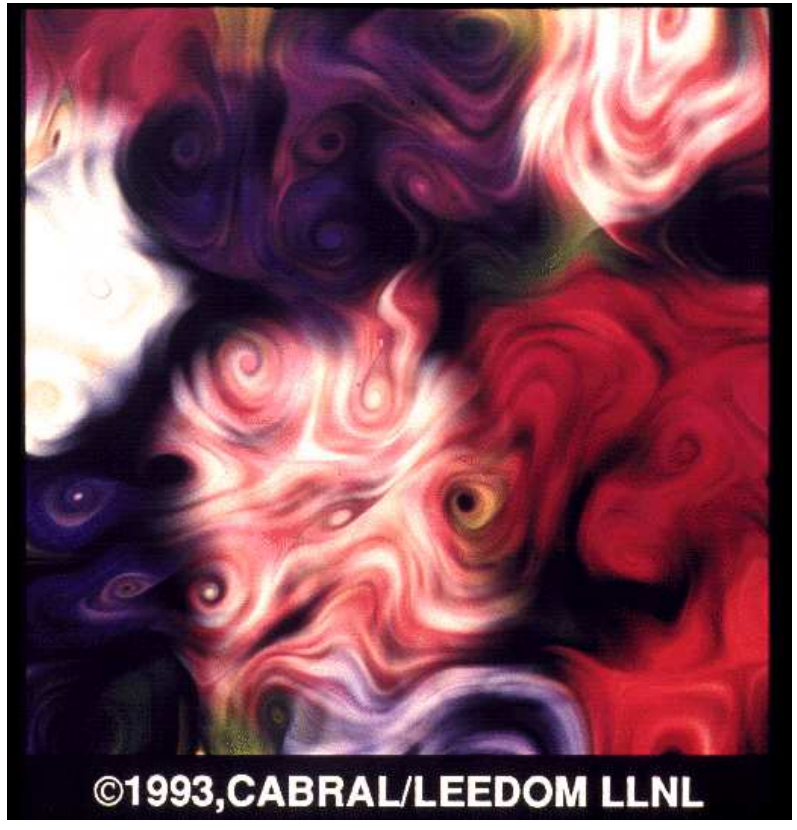
LIC (Line Integral Convolution)



A 2D example of line integral convolution

LIC images

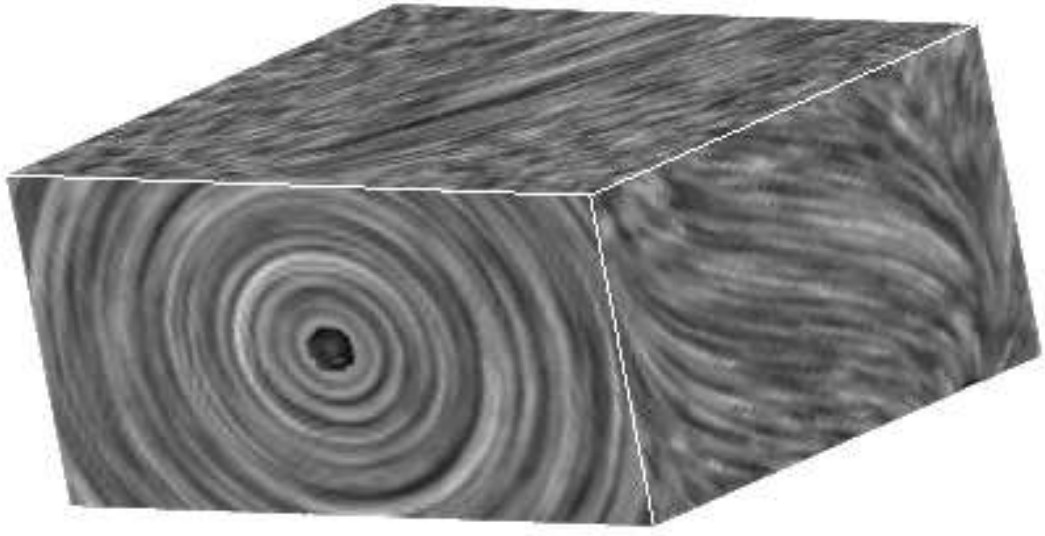




Examples of LIC images.

Texture-based techniques

- In 3D, it can be challenging to find a good visual representation of the flow due to perceptional difficulties encountered with dense 3D displays such as occlusion and cluttering.



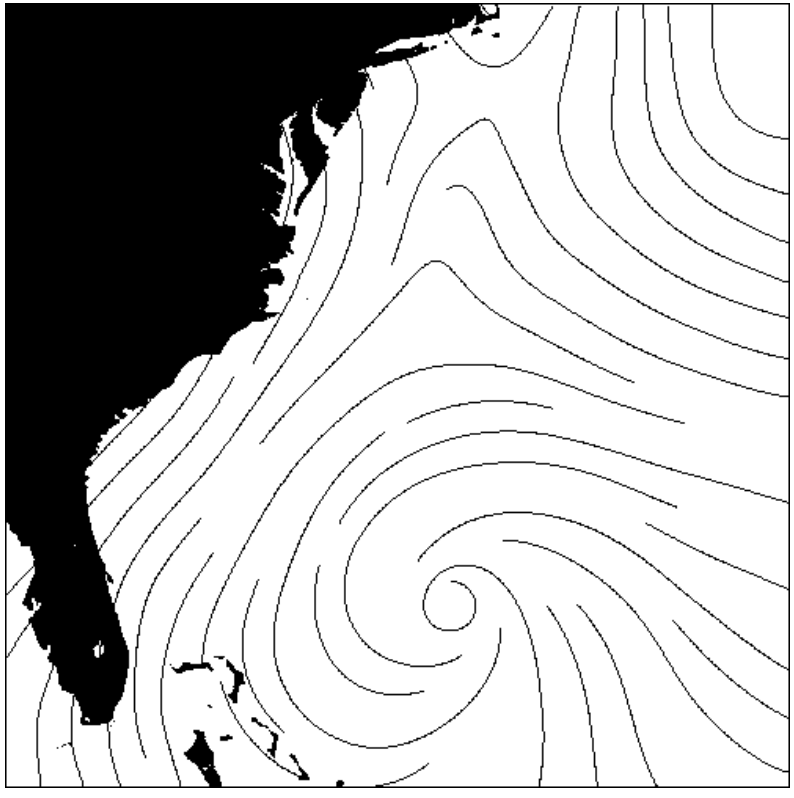
Geometry-based v.s. Texture-based

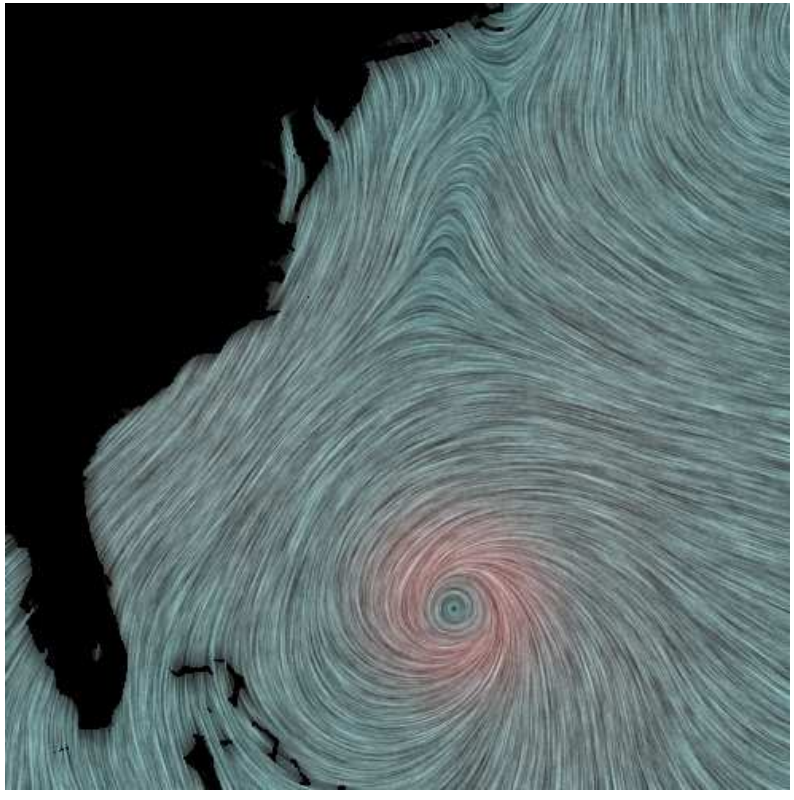
Geometry-based

- particle description
- local description

Texture-based

- field description
- global description





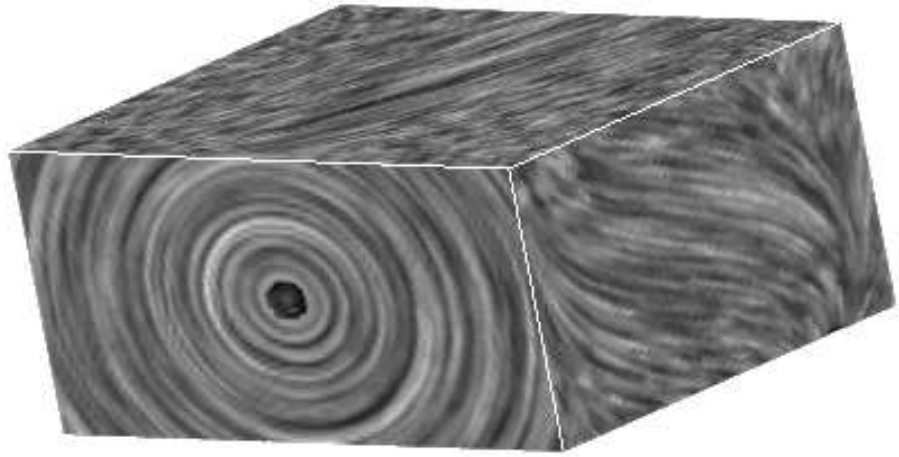
Local and global descriptions

Local description

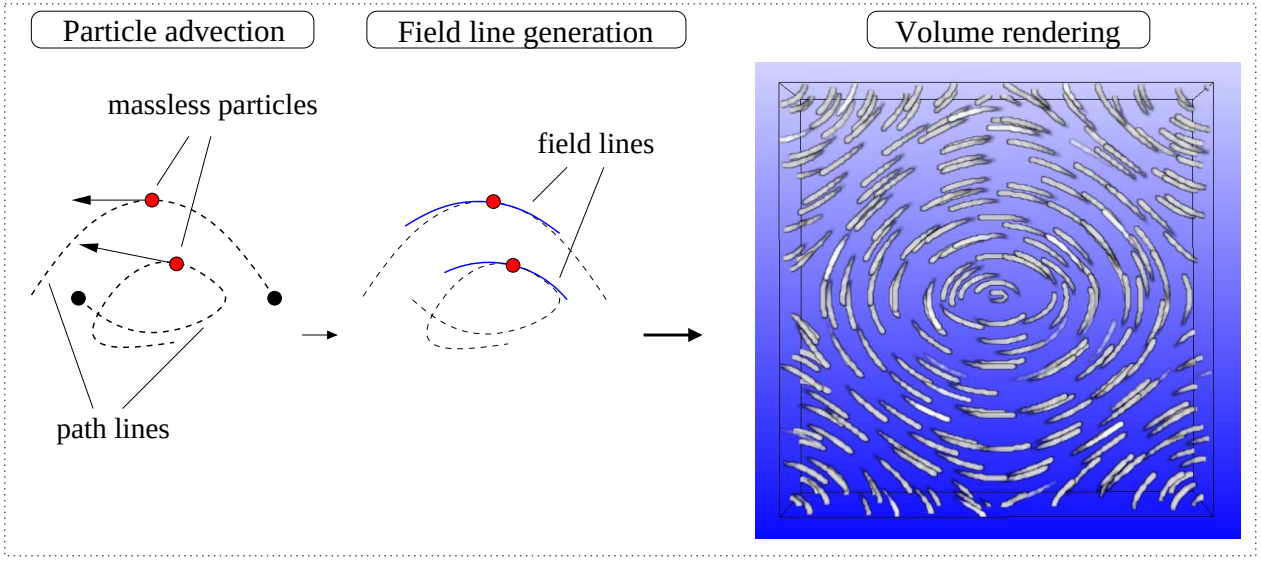
- Dependent on the seed point placement

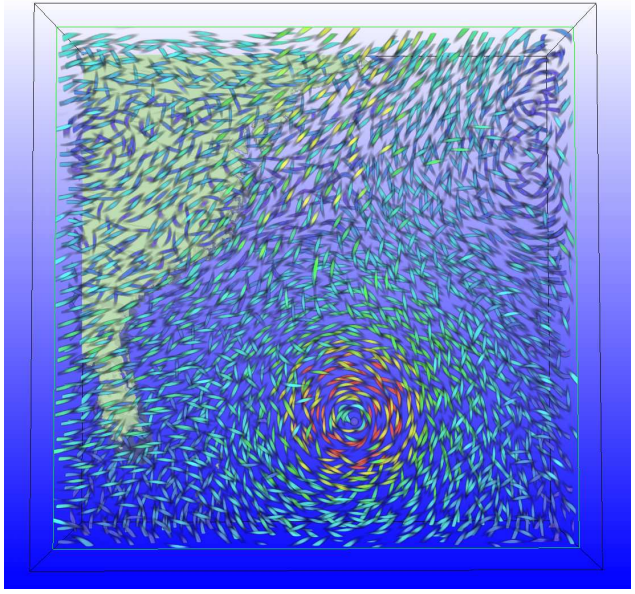
Global and dense description

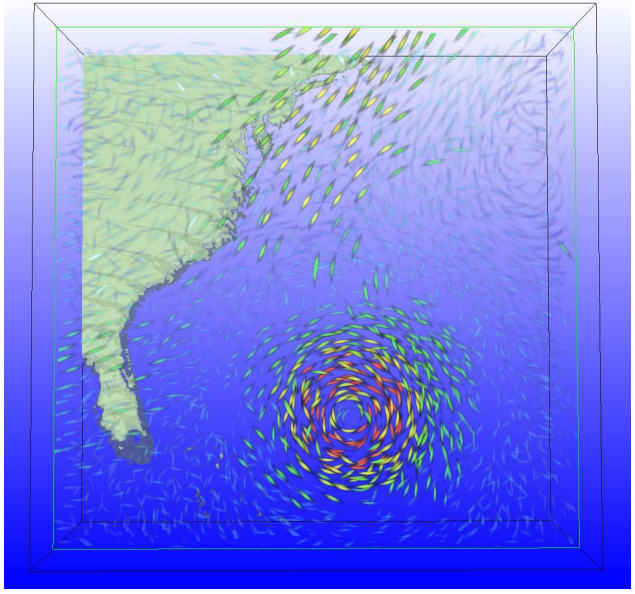
- Ideal for 2D, but not for 3D



Hybrid (texture-based and geometry-based) solution





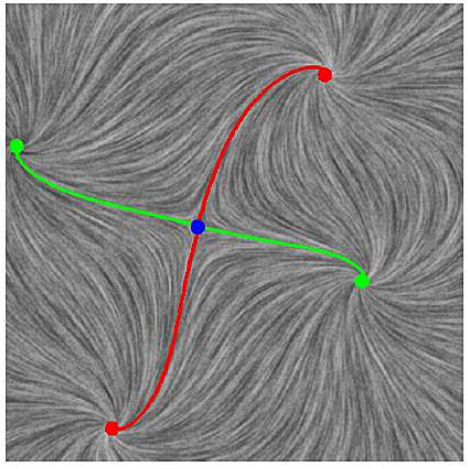


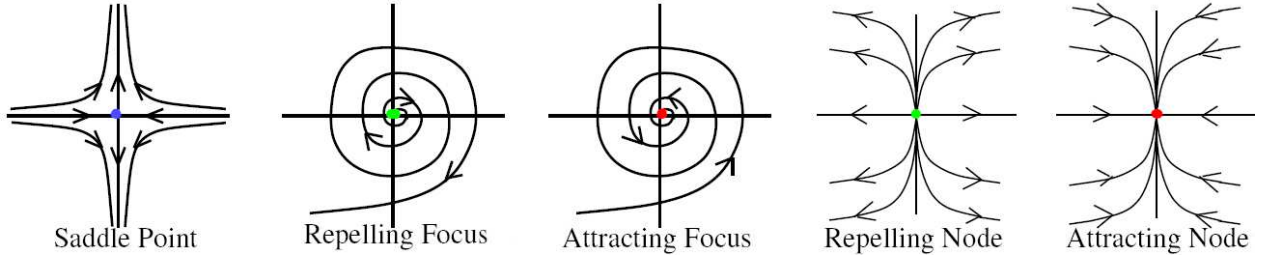
Feature-based techniques

- Goal:
  - Identify the important features of the flow.
  - Reduce the amount of data by only showing the essential information of the field.
- Topology-based
- Flow-feature based (vortices, shock waves)
- Clustering techniques

Topology-based techniques

- Based on identifying critical points (zero points) in the vector field.
- Critical points can be regarded as traffical intersections, where flow from different directions meet, crash and deflect (in new directions).
- A topological analysis leads to a separation of the vector field domain into subdomains of similar topological structure.
- Appealing for large data sets because it reduced the amount of data needed to be represented.





Vector fields as ODEs

- A vector field $\boldsymbol{V}\big(\boldsymbol{x}(t), t\big)$ describes the connection between location and velocity of a (massless) particle
- This relation can be expressed as an ordinary differential equation.

$$\mathbf{x}'(t) = \boldsymbol{V}\big(\boldsymbol{x}(t), t\big), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \text{ (time-dependent)}$$
$$\mathbf{x}'(t) = \boldsymbol{V}\big(\boldsymbol{x}(t)\big), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \text{ (stationary)}$$

- With an integral curve as a solution

$$\mathbf{x}(t) = \mathbf{x_0} + \int_{t_0}^{t} \boldsymbol{V}\big(\boldsymbol{x}(\tau), \tau\big) d\tau.$$
$$\mathbf{x}(t) = \mathbf{x_0} + \int_{t_0}^{t} \boldsymbol{V}\big(\boldsymbol{x}, \tau\big) d\tau.$$

- The curve $\mathbf{x}(t)$ degenerates into a single point if $\boldsymbol{V}\big(\boldsymbol{x}(t_0)\big) = 0$. These points are known as critical points.

Vector fields as ODEs

- Using Taylor expansion we can express the velocity field as:

$$\boldsymbol{V}(\boldsymbol{x}+\mathbf{h}) = \boldsymbol{V}(\boldsymbol{x_0}) + \nabla \boldsymbol{V}(\boldsymbol{x_0})\mathbf{h} + \ldots \quad , \quad \mathbf{h} = \mathbf{x} - \mathbf{x_0}$$

- Around a critical point the velocity field can be approximated linearly by the velocity gradient matrix (also known as the Jacobien):

$$\boldsymbol{V}(\boldsymbol{x_0}+\mathbf{h}) \approx \nabla \boldsymbol{V}(\boldsymbol{x_0})(\mathbf{x} - \mathbf{x_0})$$

$$\nabla \boldsymbol{V}(\boldsymbol{x_0}) = \begin{pmatrix} \frac{\partial V_1}{\partial x}(\boldsymbol{x_0}) & \frac{\partial V_1}{\partial y}(\boldsymbol{x_0}) \\ \frac{\partial V_2}{\partial x}(\boldsymbol{x_0}) & \frac{\partial V_2}{\partial y}(\boldsymbol{x_0}) \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

- Hence, around a critical point the ODE can be rewritten as:

$$\mathbf{x}'(t) = \boldsymbol{V}(\boldsymbol{x}(t))$$

$$\mathbf{x}' \approx \nabla \boldsymbol{V} \mathbf{x}$$

$$\mathbf{x}' = \mathbf{A}\mathbf{x}$$

Vector fields as ODEs

- In a small region around a critical point the vector field can be approximated by the Jacobian evaluated at the critical point:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} \tag{1}$$

- If we insert the solution $\mathbf{x}(t) = \mathbf{v}e^{\lambda t}$ into eq(1) we get the eigenvector equation of the matrix A

$$\lambda\mathbf{v} = \mathbf{A}\mathbf{v} \tag{2}$$

- Hence, finding eigenvalues and eigenvectors of $\mathbf{A}$ gives us a solution of eq (1).
- Equation (2) can be rewritten as

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0$$

with the eigenvalues of $\mathbf{A}$ given as roots to the polynomial

$$p(\lambda) = det(\mathbf{A} - \lambda\mathbf{I})$$

Vector fields as ODEs

- Assume a 2D vector field given as

$$\mathbf{x}'(t) = \boldsymbol{V}(\boldsymbol{x}) = \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

- Its Jacobian is

$$\nabla \boldsymbol{V} = \begin{pmatrix} \frac{\partial V_1}{\partial x} & \frac{\partial V_1}{\partial y} \\ \frac{\partial V_2}{\partial x} & \frac{\partial V_2}{\partial y} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

- With eigenvalues given by the equation

$$\lambda^2 - p\lambda + q = 0,$$

$$p = (a + d) = Trace(\mathbf{A}) \quad \text{and} \quad q = (ad - bc) = det(\boldsymbol{A})$$

- With solution

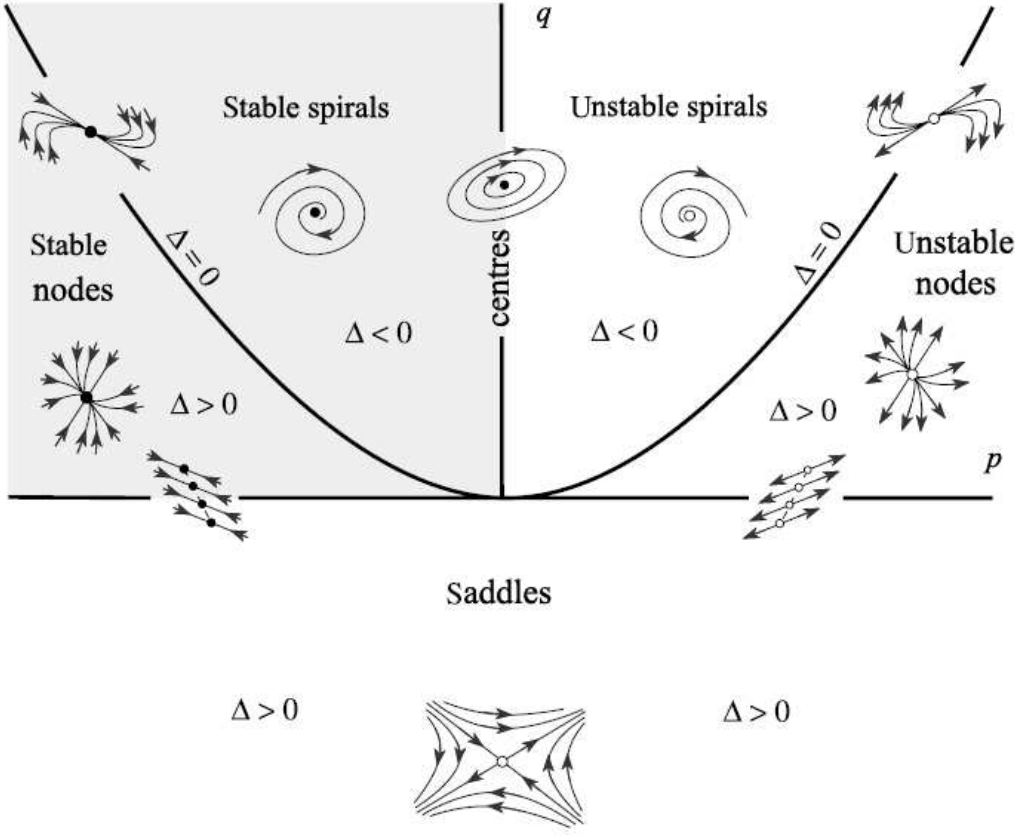
$$\lambda_1, \lambda_2 = \frac{1}{2}(p \pm \sqrt{p^2 - 4q})$$

Classification of critical points

- Classification of critical points can be determined by the eigenvalues
- Real eigenvalues
  - $\lambda_1 > \lambda_2 > 0$ (or $\lambda_1 = \lambda_2 > 0$): Repelling node (source)
  - $\lambda_1 < \lambda_2 < 0$ (or $\lambda_1 = \lambda_2 < 0$): attracting node (sink)
  - $\lambda_1 < 0 < \lambda_2$: Saddle
- Both real and imaginary parts
  - $\lambda_1, \lambda_2 = p \pm qi, p > 0$: Repelling spiral (source)
  - $\lambda_1, \lambda_2 = p \pm qi, p < 0$: attracting spiral (sink)
- Only imaginary parts
  - $\lambda_1, \lambda_2 = \pm qi$: Center

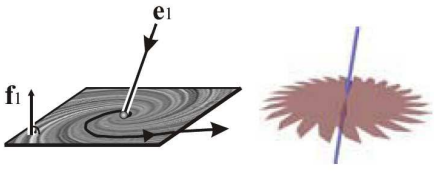
Trace-determinant plane

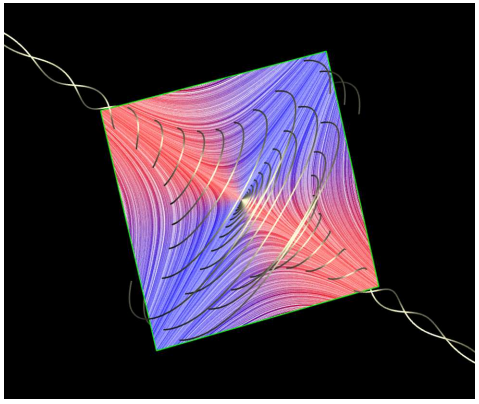
- Classification for the linear system $\mathbf{x}'(t) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mathbf{x}$ in the (p, q) plane with $p = (a + d) = Trace(\mathbf{A})$, $q = (ad - bc) = det(\boldsymbol{A})$ and $\Delta = p^2 - 4q$.

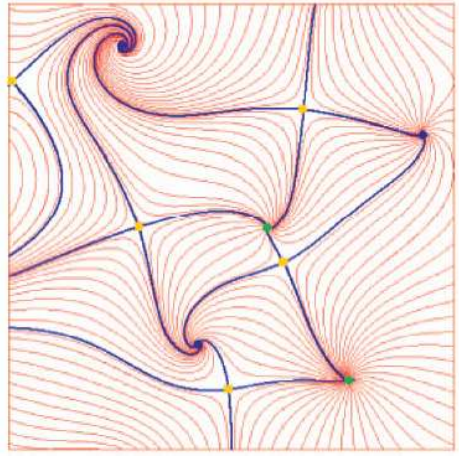


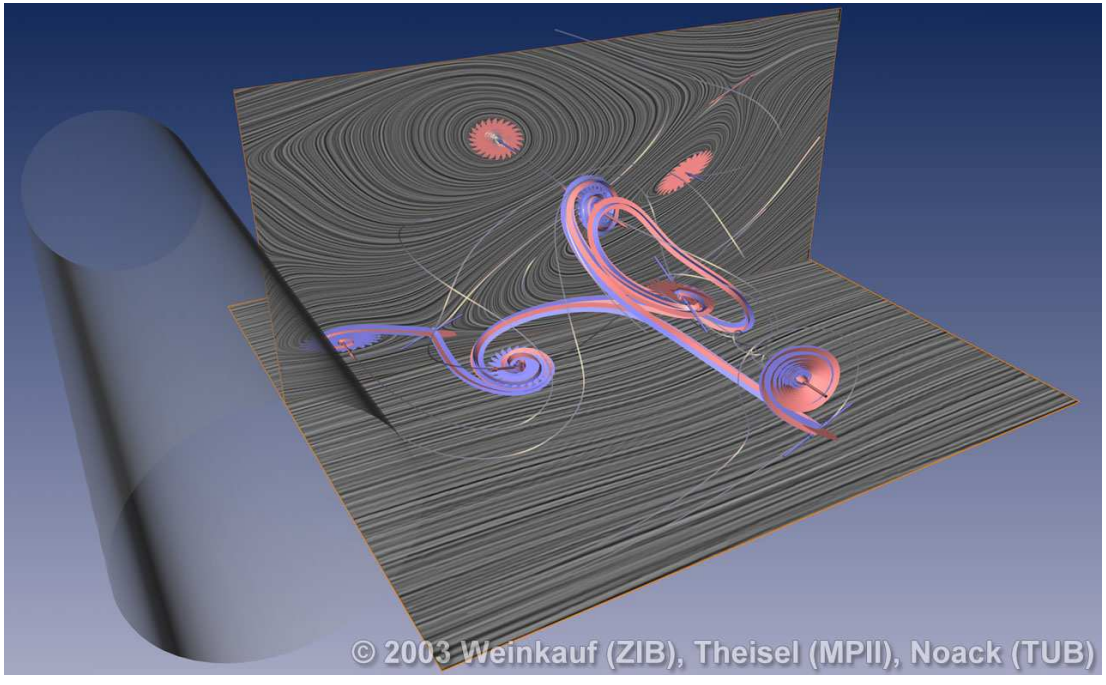
Topology-based techniques

Repelling saddle focus



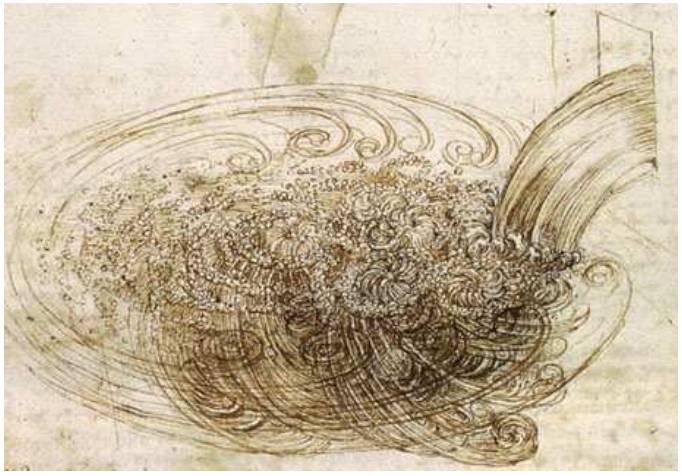


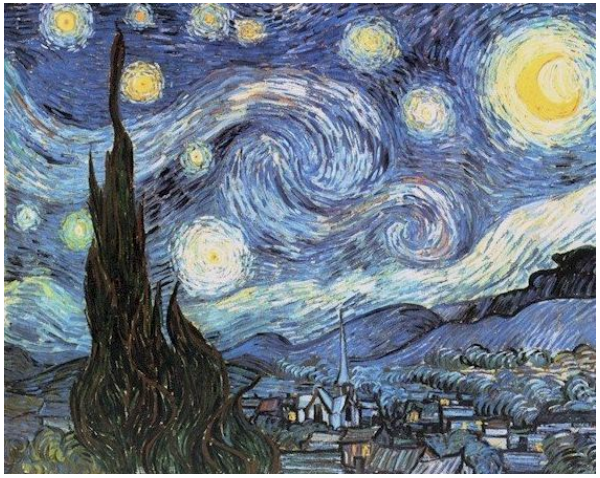




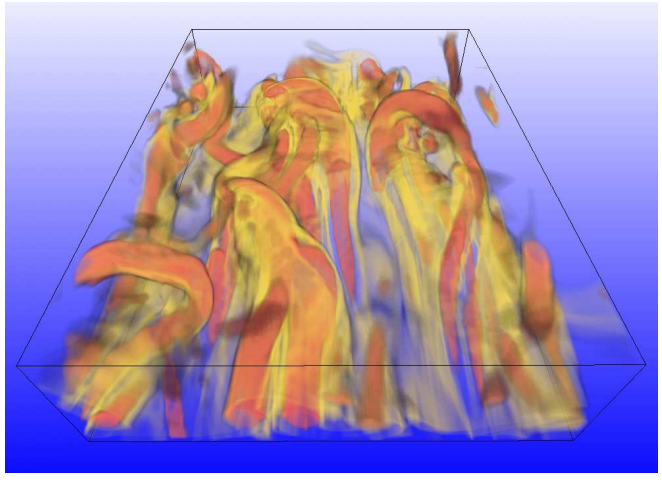
Flow feature-based techniques

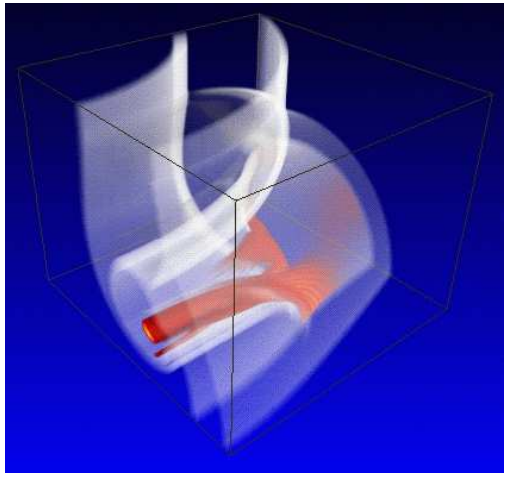
- Identifies important flow features in the data, such as as vortices and shocks.
- **Vortex-detection methods:** enstrophy ($\eta = \|\nabla \times \mathbf{u}\|^2$), the Q criterion, the λ_2 criterion, and the Predictor-Corrector, Eigenvector, Parallel Vectors, and Combinatorial methods.

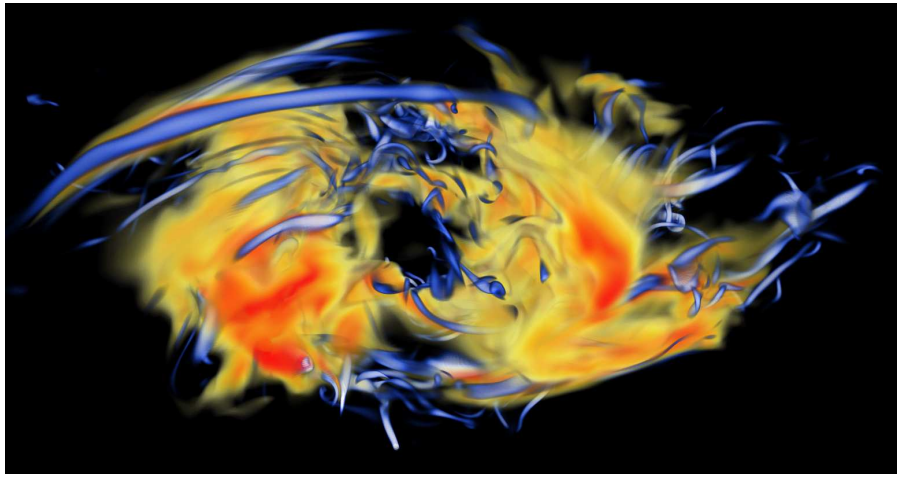




Flow feature-based techniques

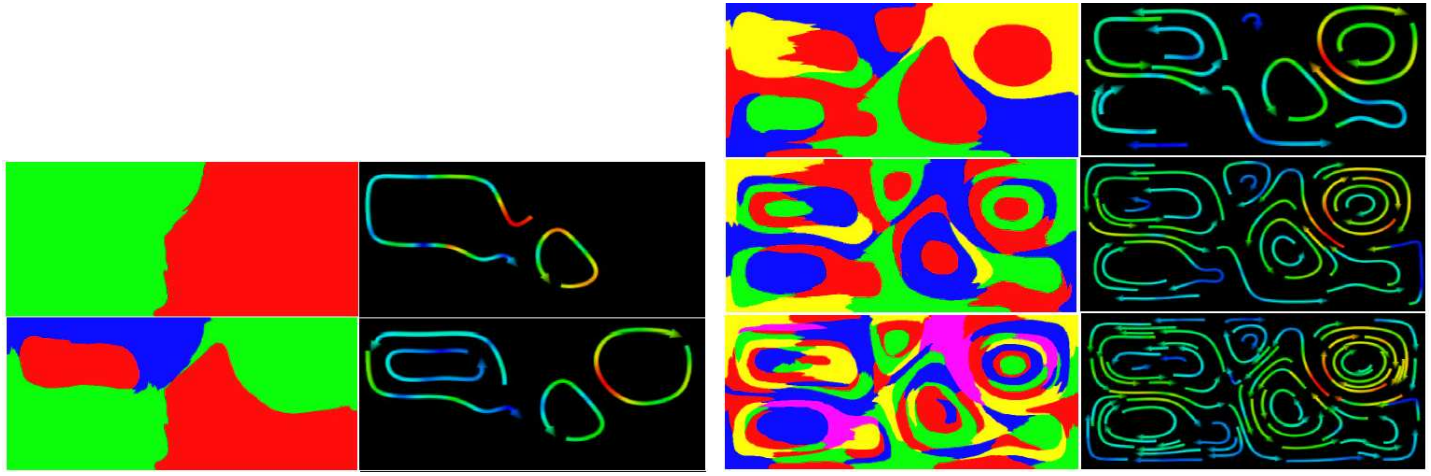






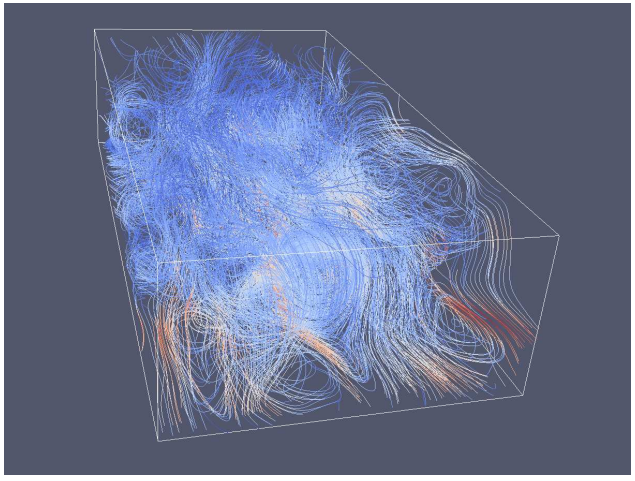
Clustering techniques

- Generates a coarser representation of the vector field by clustering/segmentation methods.
- **Goal:** Create a simpler model that is easier to visualize without loosing important features
- The vector field is typically divided into a number of distinct regions with similar vector field direction or topology.



Seeding strategies

- The placement of the seed points directly determines the visualization quality
  - Too many leads to scene cluttering
  - Too few results in an incomplete rendering with important patterns missing
- The seeds has to be properly placed with the right amount.



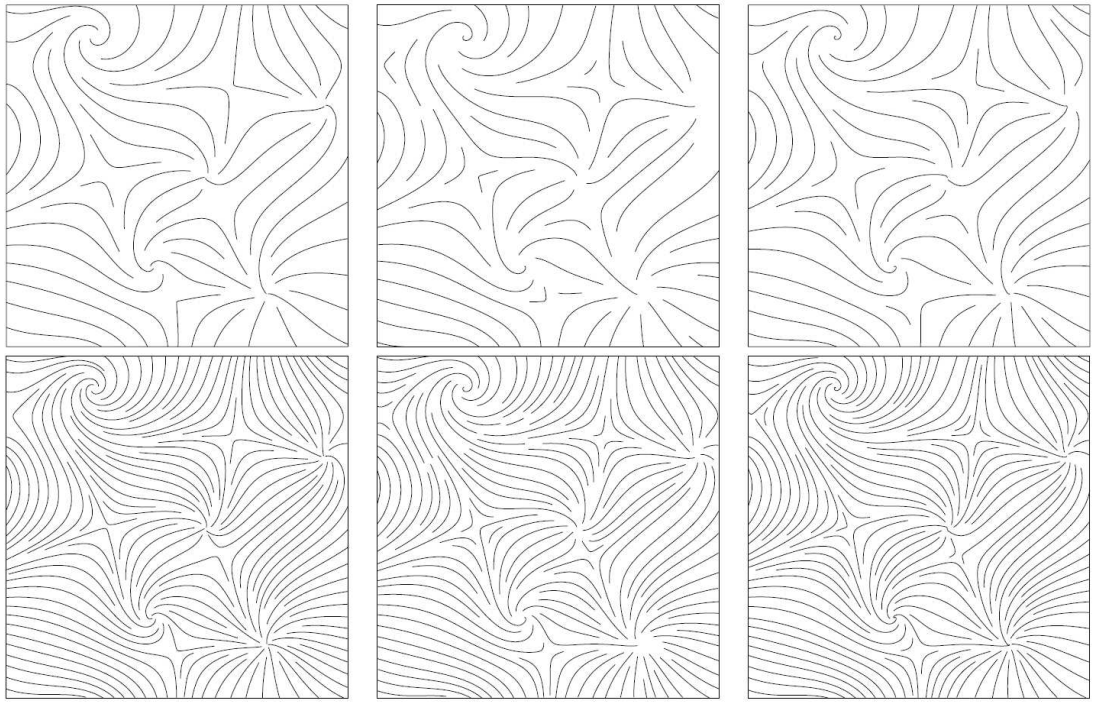
A bad seeding example

Seeding strategies

- Density based
- Flow feature based
  - Flow field based (curvature, vorticity, ..)
  - Flow topology based
  - Clustering based
- View dependent based
- image based
- Opacity based

Density based seeding

- Focuses on finding seeds that generate curves that cover the data domain with an even distribution of field lines.
- Both sparse and dense representations.

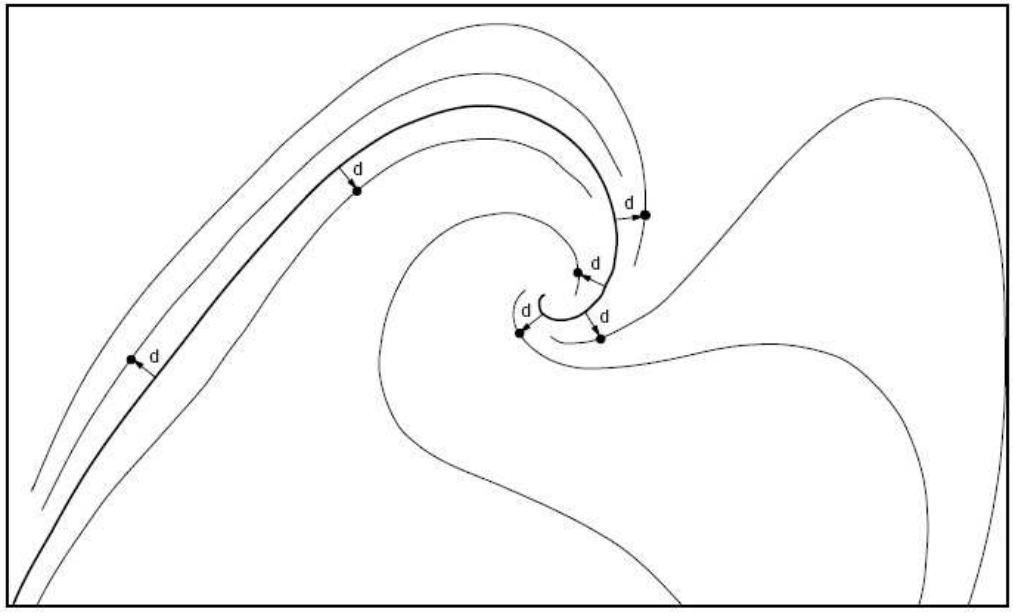


[Turk-Banks 1996], [Jobard-Lefer 1997], [Mebarki et. al. 2005]

[Minimizing energy function], [Evenly spaced], [Farthest point seeding]

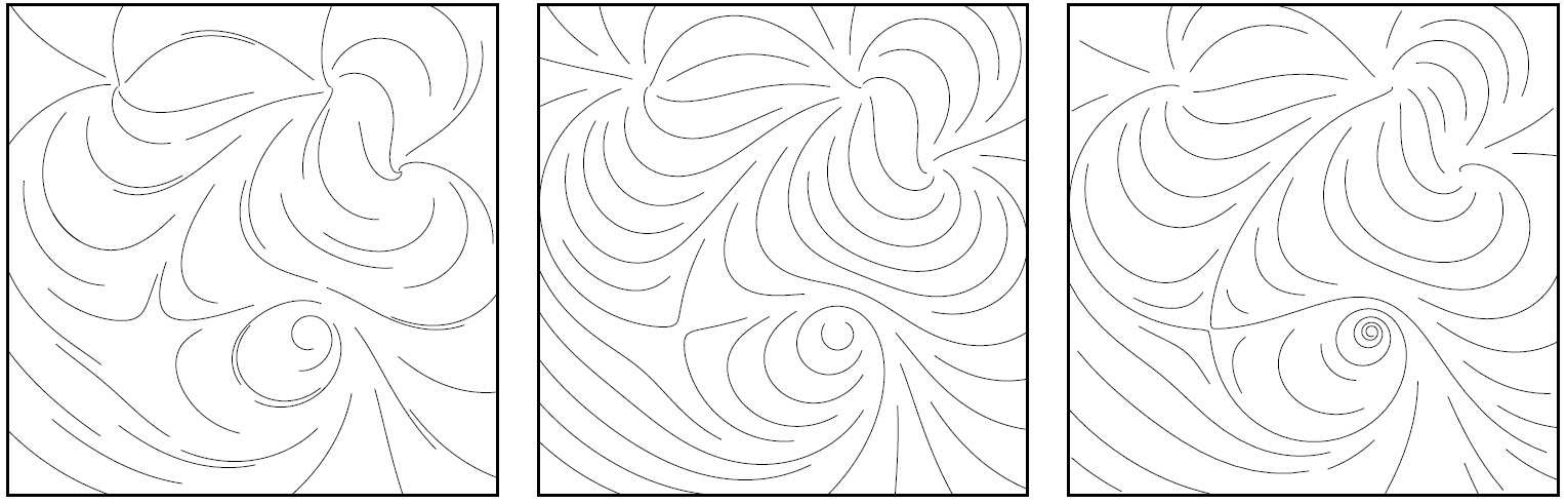
Evenly spaced [Jobard-Lefer 1997]

```
Compute an initial streamline and put it into the queue
Let this initial streamline be the current streamline
Finished := False
Repeat
    Repeat
        Select a candidate seedpoint at d = d_sep apart from the current streamline
    Until the candidate is valid or there is no more available candidate
    If a valid candidate has been selected Then
      Compute a new streamline and put it into the queue
    Else
      If there is no more available streamline in the queue Then
        Finished := True
      Else
        Let the next streamline in the queue be the current streamline
      EndIf
    EndIf
Until Finished=True
```



Topology based seeding

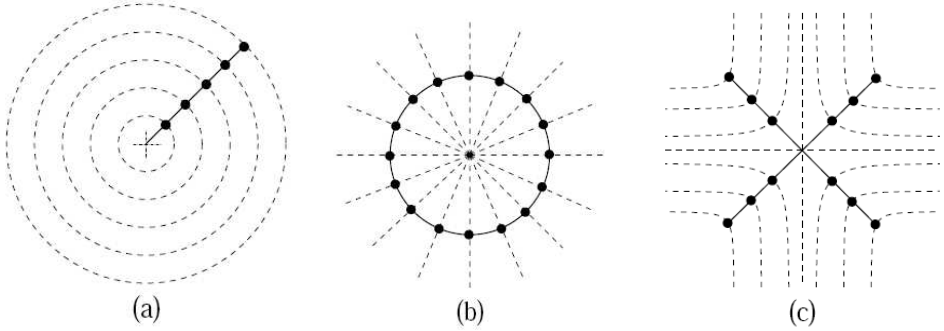
- Does a topological analysis of the vector field (finds critical points and seperatrices) prior to the seeding.
- Uses topology based information to guide the seeding of field lines.
- Ensures that the topological features are well represented by the field lines.



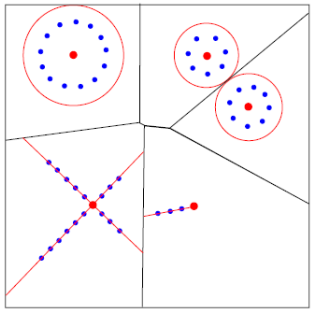
[Regular seeding], [Density based seeding], [topology based seeding]

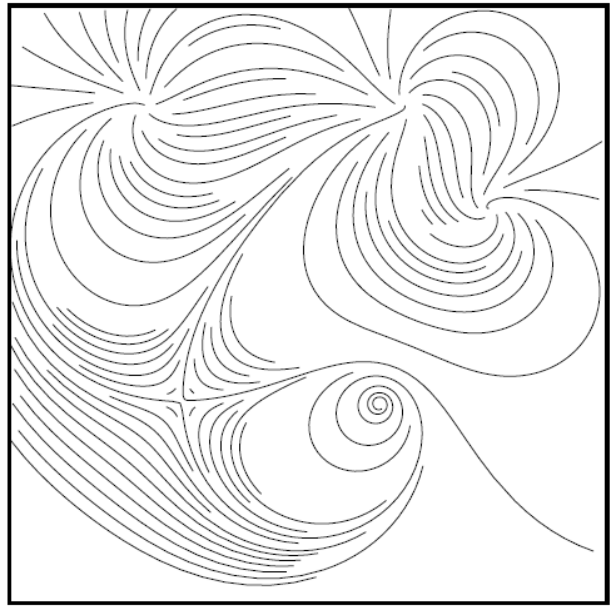
Topology based seeding [Verma et. al. 2000]

- Seeds placed according to templates for different critical points.
- "Blank" spaces filled with an even distrubition of field lines using a random poisson disk distribution.



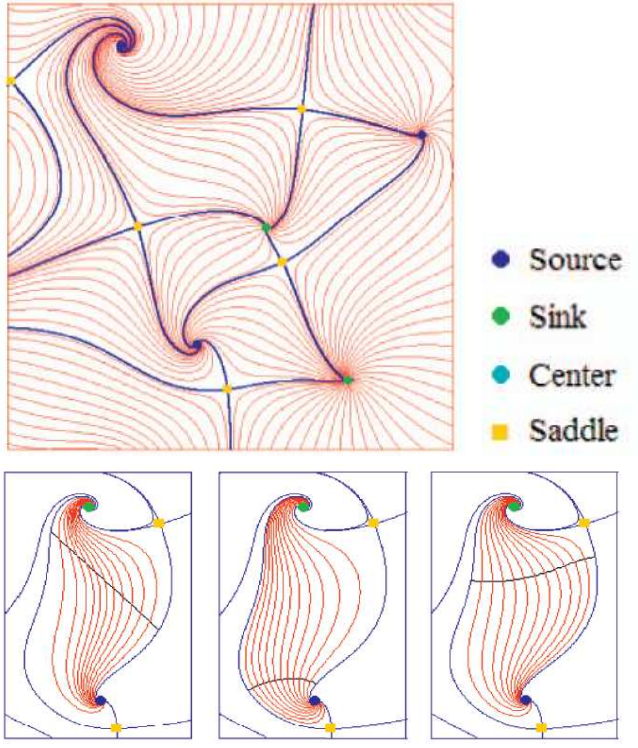
[Center, spiral], [Source, sink], [Saddle]

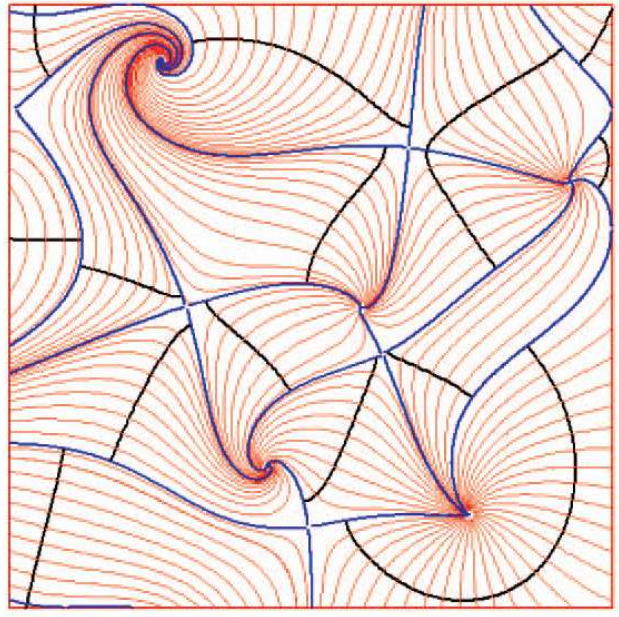




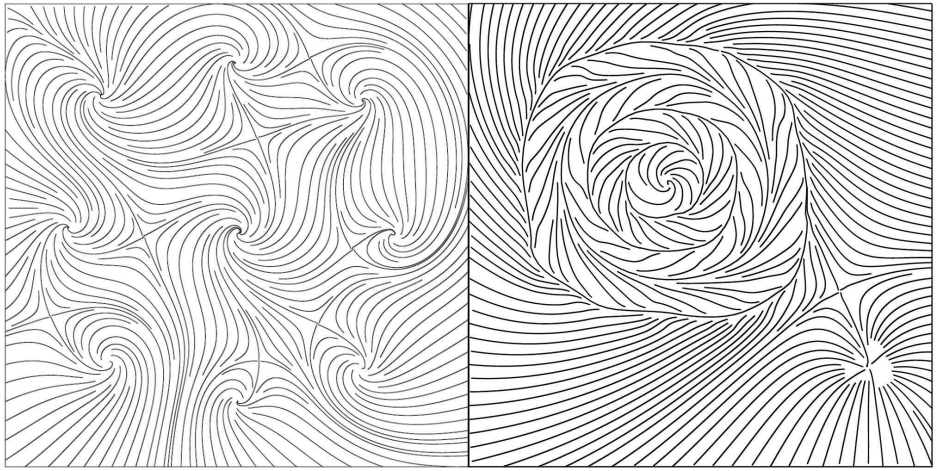
Topology based seeding [Wu et. al. 2010]

- Generates a topological similar regions by finding seperatrices connecting critical points.
- Creates evenly spaced field lines in each region by seeding along (the longest) orthogonal curves.

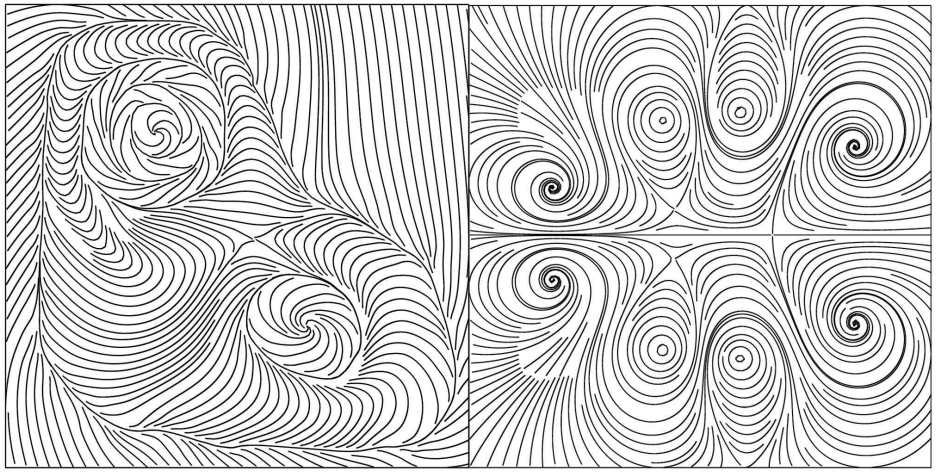




Examples [Wu et. al. 2010]

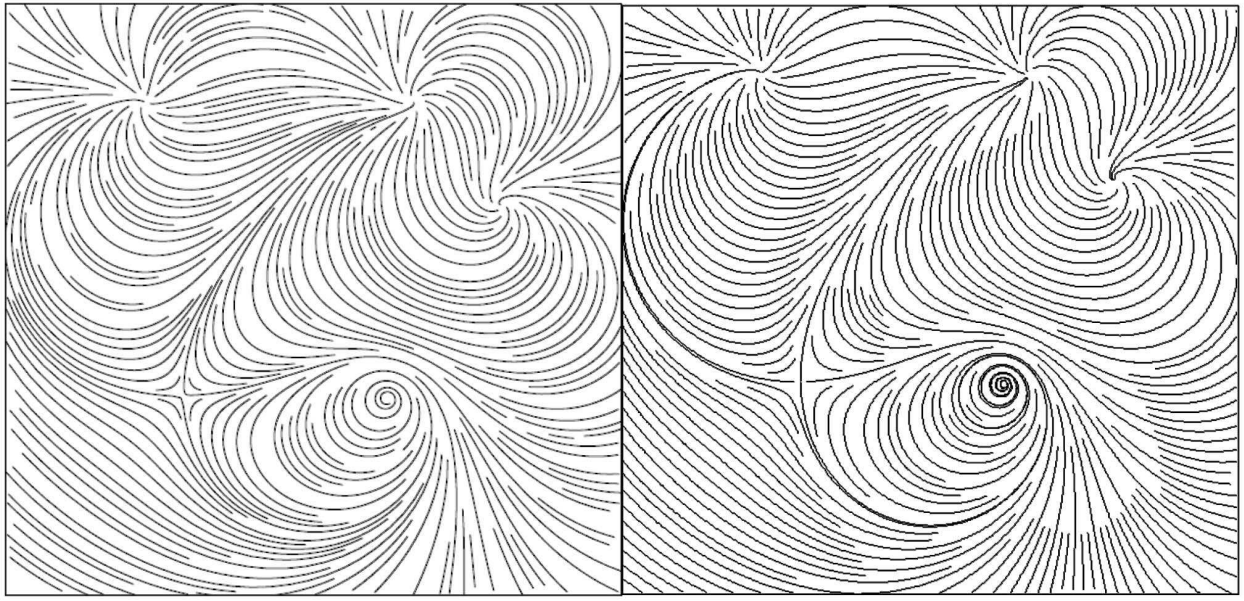


(a) (b)



(c) (d)

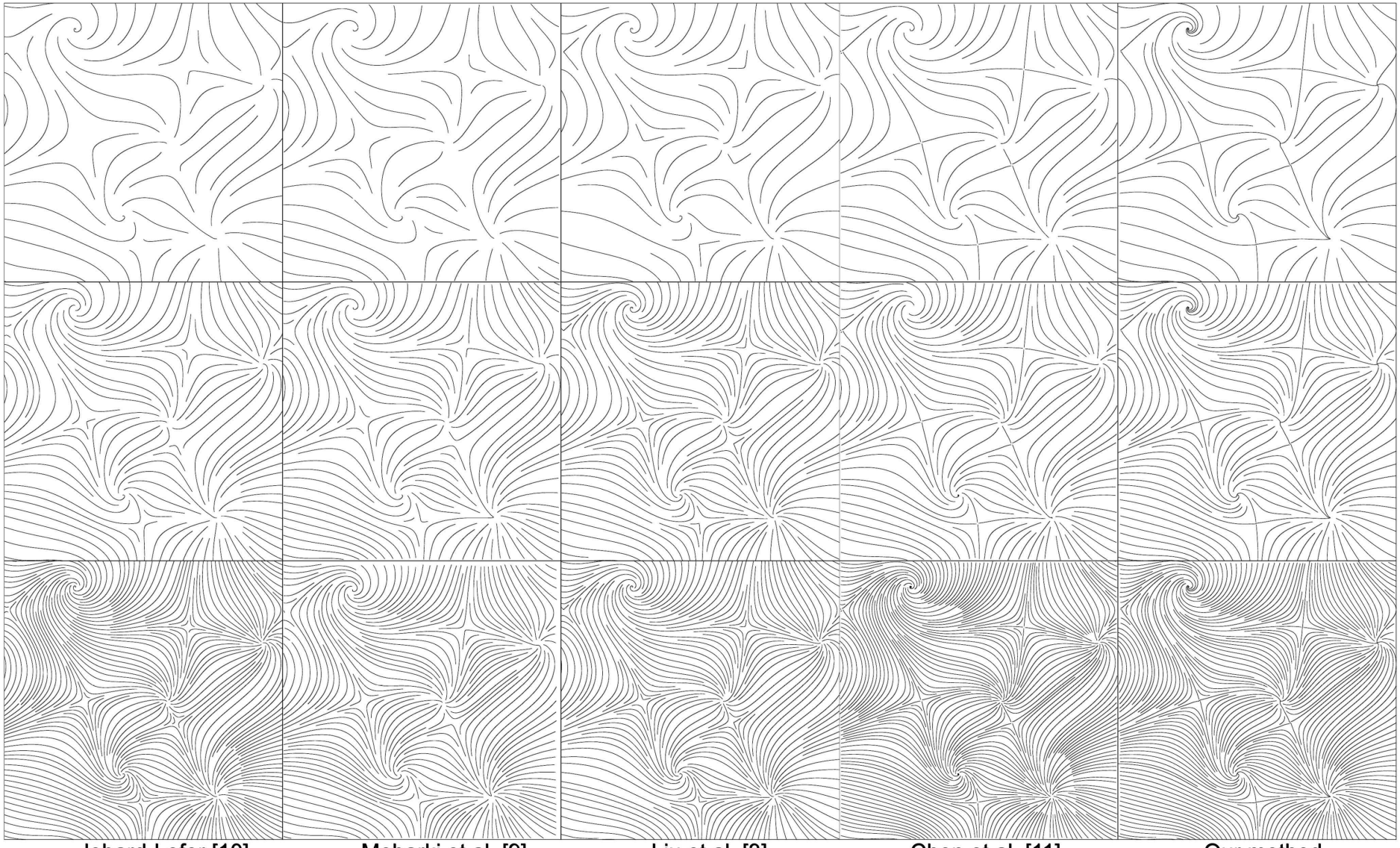
Comparison: [Wu et. al. 2010] & [Verma et. al. 2000]



(a) (b)

[Verma et. al. 2000], [Wu et. al. 2010]

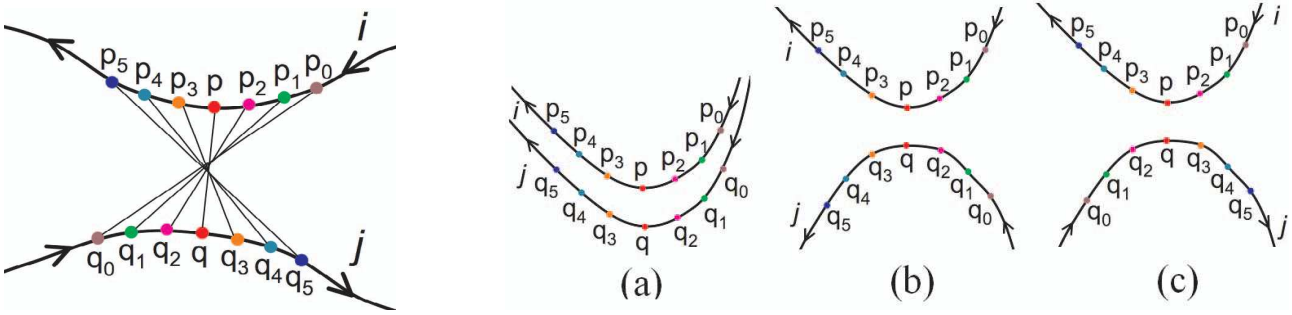
Comparison: [Wu et. al. 2010] & [Others]



Similarity guided seeding [Chen et. al. 2007]

- A density based approach which uses a similarity distance instead of a Euclidean distance metric.
- The similarity distance measures the difference in shape and orientation between field lines.

$$d_{sim} = \|\mathbf{p} - \mathbf{q}\| + \alpha \frac{\sum_{k=0}^{m-1} \left| \|\mathbf{p_k} - \mathbf{q_k}\| - \|\mathbf{p} - \mathbf{q}\| \right|}{m}$$



- More field lines are placed in regions of the flow where the field line similarity distance are high compared to low.
- Thus it naturally favors interesting features such as the vicinity of critical points.

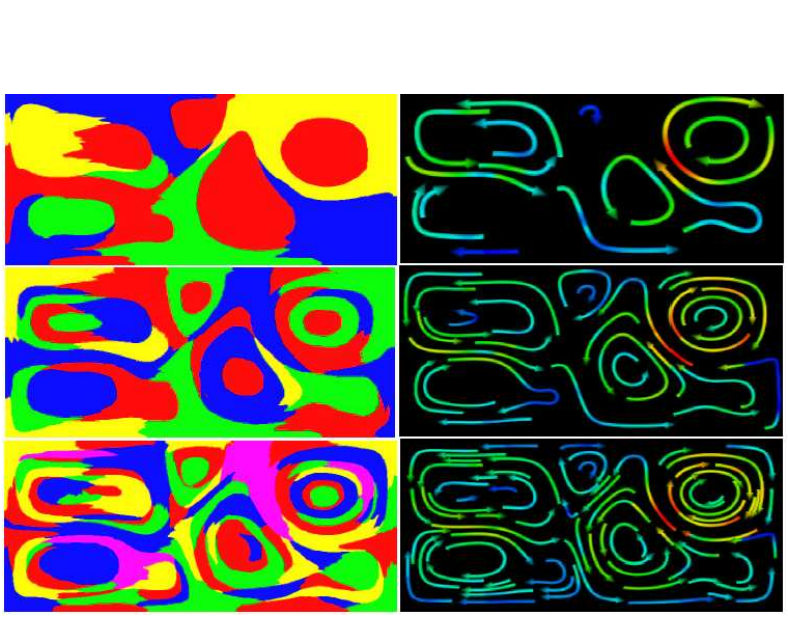
Similarity guided seeding [Chen et. al. 2007]

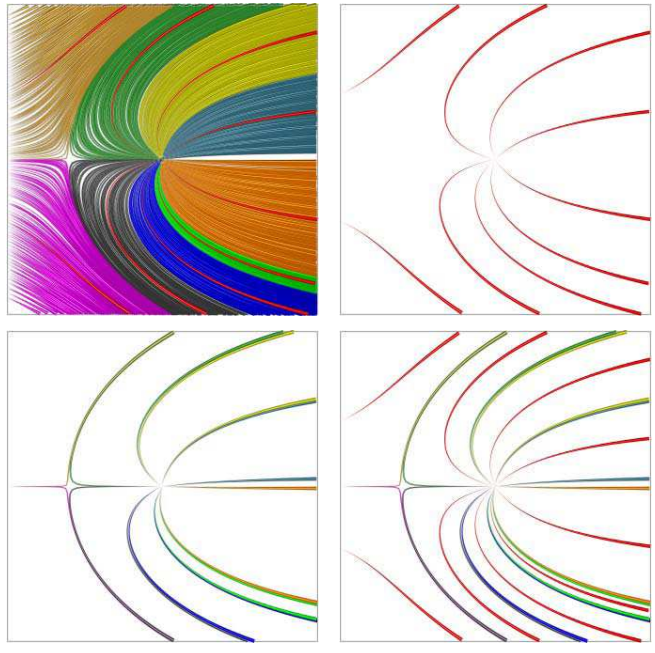
```
Placement(vecField, numSeeds, dSep, dSelfSep, minLen)
    Seeds = randomSeeds(vecField, numSeeds)
    while(Seeds is not empty)
        seed = Seeds.dequeue()
        Integrate(forward, vecField, seed, Lines, line)
        Integrate(backward, vecField, seed, Lines, line)
        if (Length(line) > minLen)
            Lines.insert(line)

Integrate(direction, vecField, seed, Lines, line)
    while(RK_OneStep(direction, line, p) succeeds)
        if dSim(p, line, line) < dSelfSep
            Close(p, line)  // optional
            return
        forall prevLine in Lines
            if d(p, line, prevLine) < dSep
                return
```

[Euclidian distance metric], [Similarity distance metric]

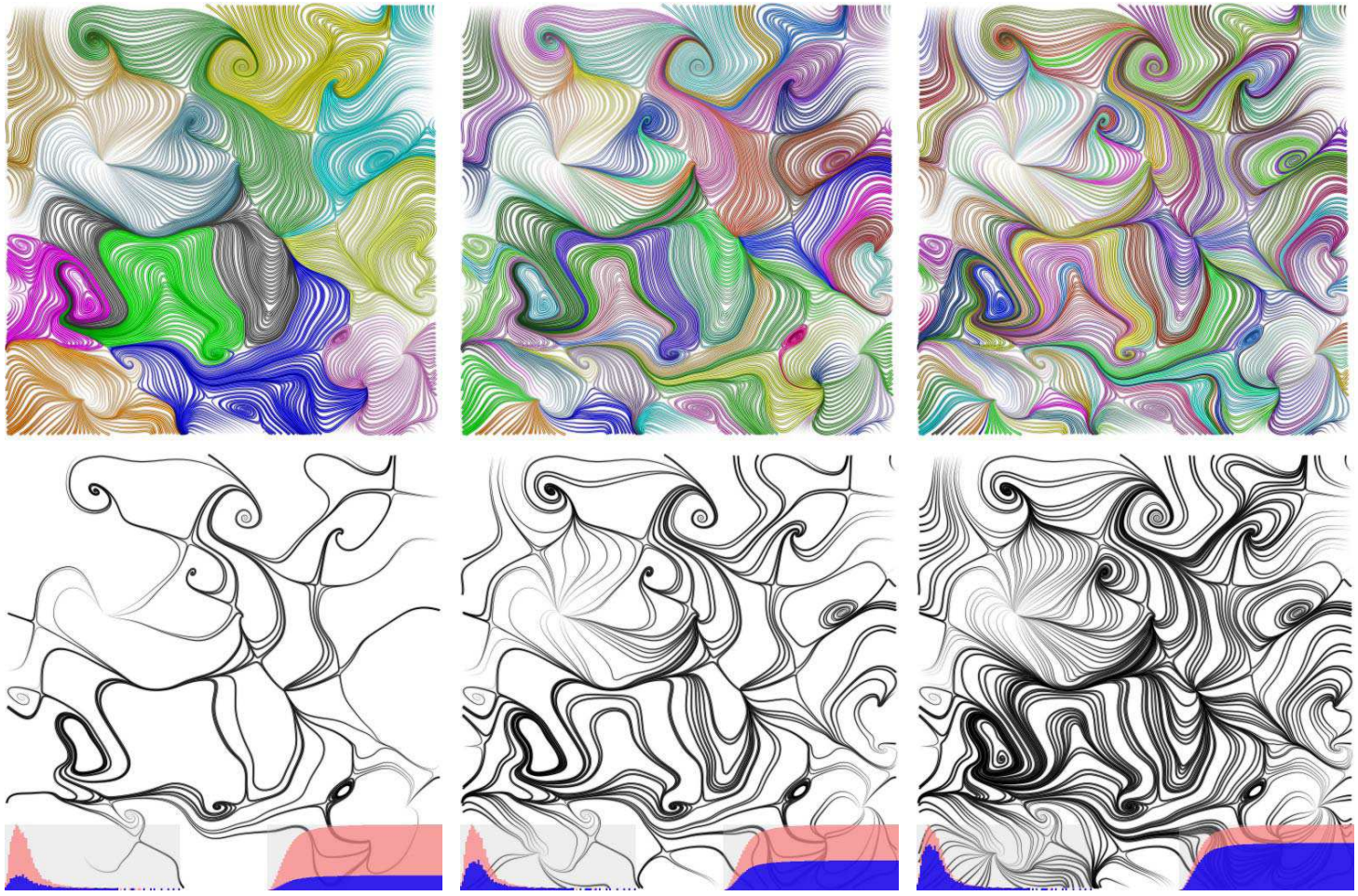
Clustering based seeding





[Flow decomposition, Griebel2004], [Streamline bundles, Yu2012]

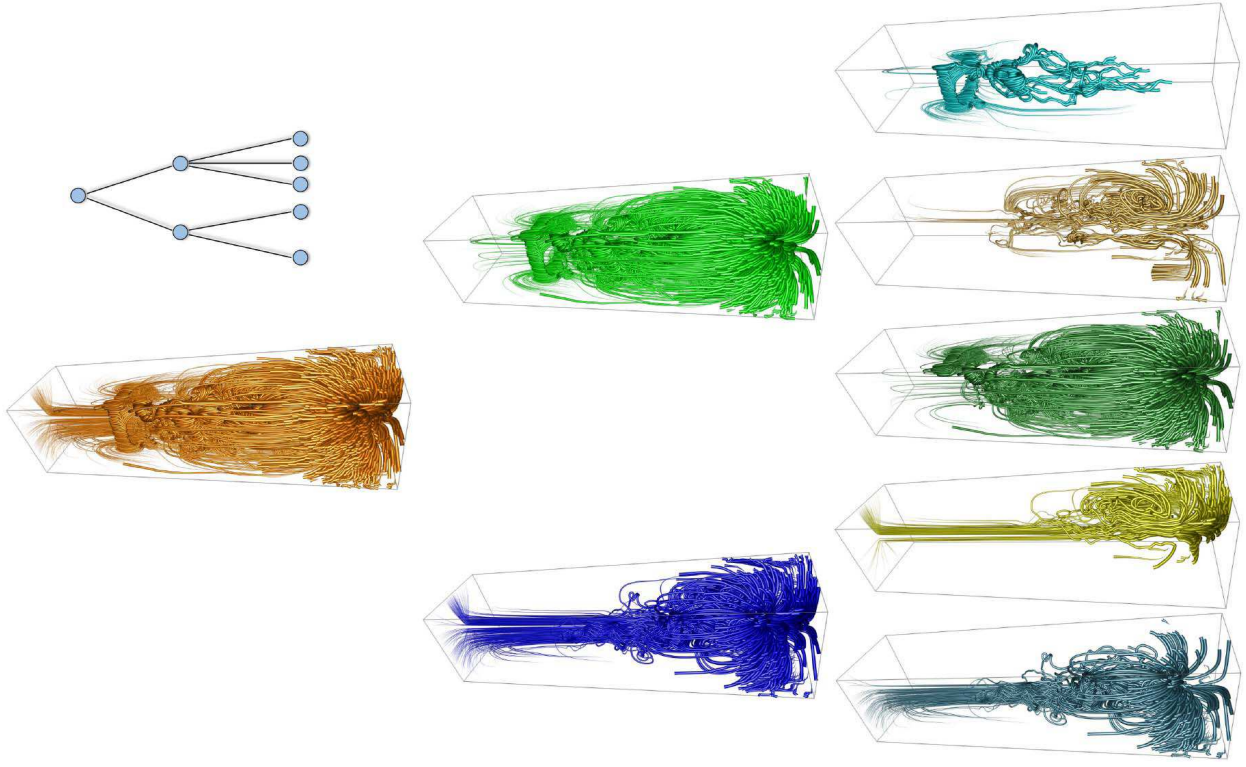
Streamline bundling [Yu et. al. 2012]



[12 clusters], [70 clusters], [166 clusters]

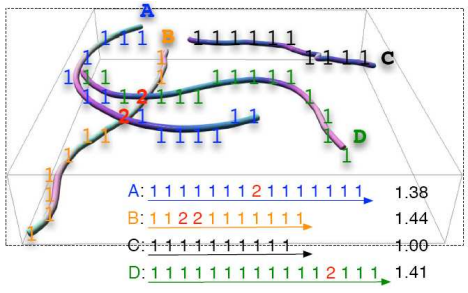
Streamline bundling [Yu et. al. 2012]

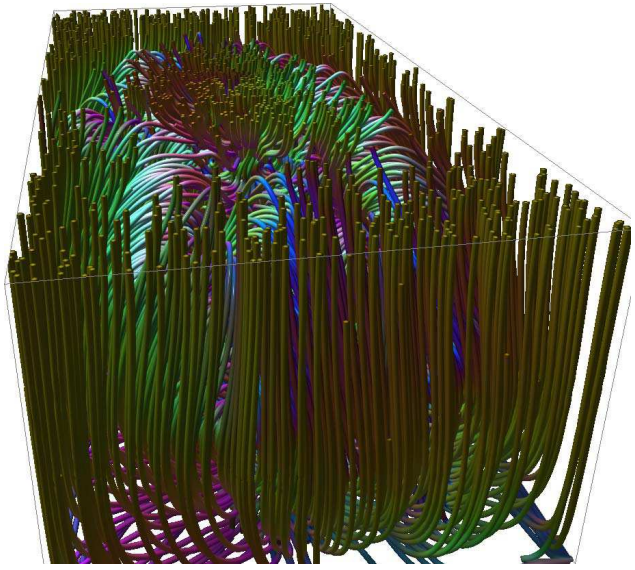
- Builds a hierarchy of clusters at different resolutions.
- Can be used to partition the flow field and enables the investigation of individual parts.

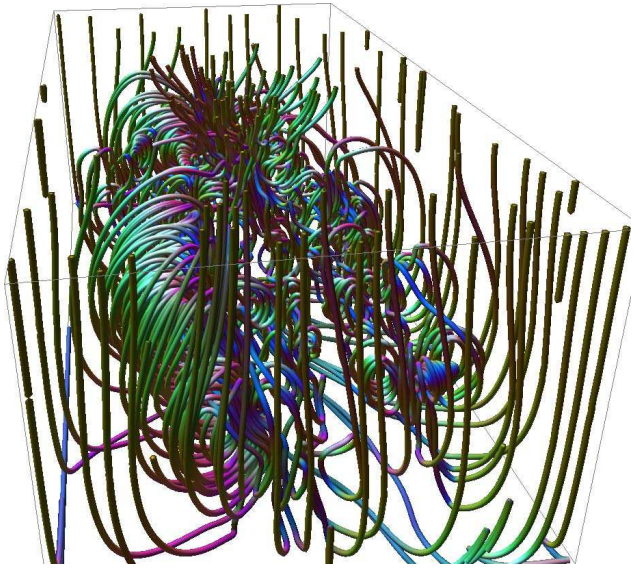


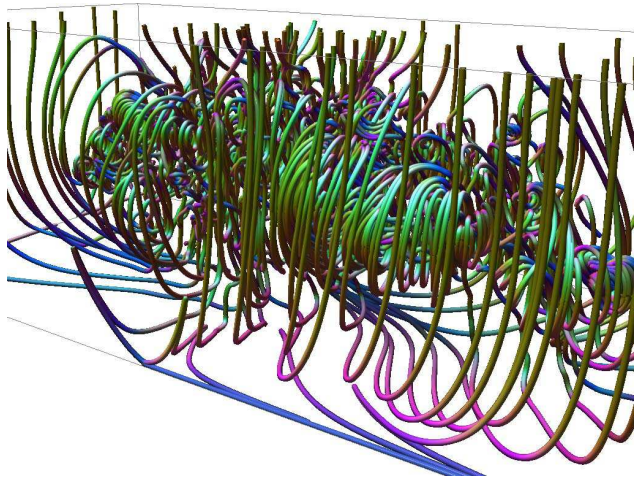
View-dependent selection [Marchesin et. al. 2010]

- Tries to adress the occlusion issue by removing field lines that occludes important flow features.
- Streamline metric $C = S_{relevance}/S_{Overlap}$



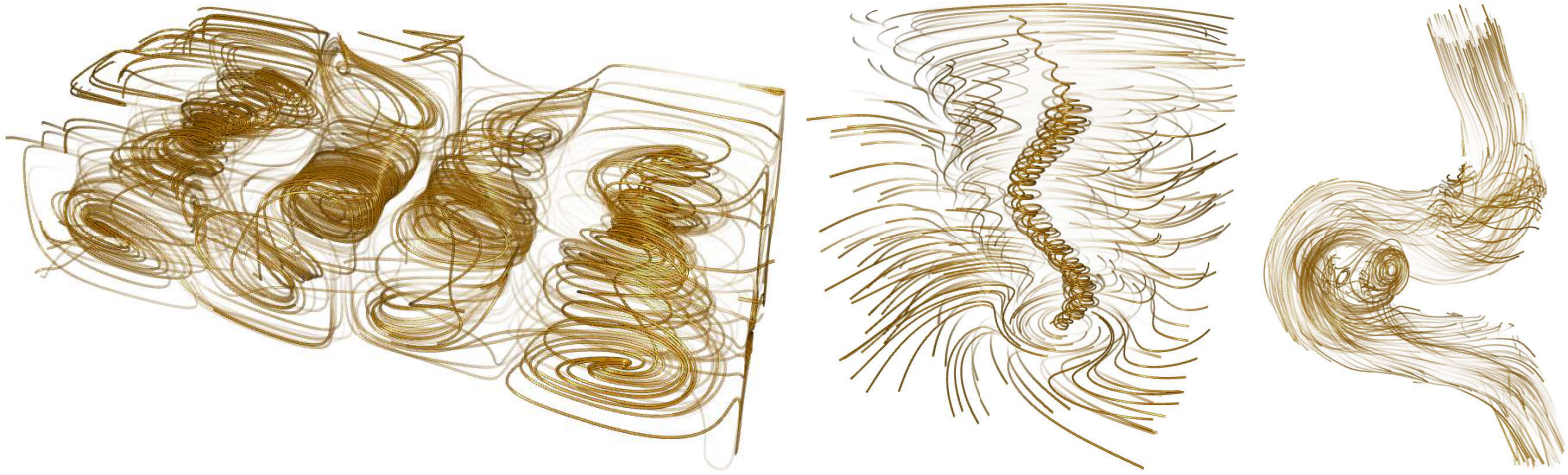






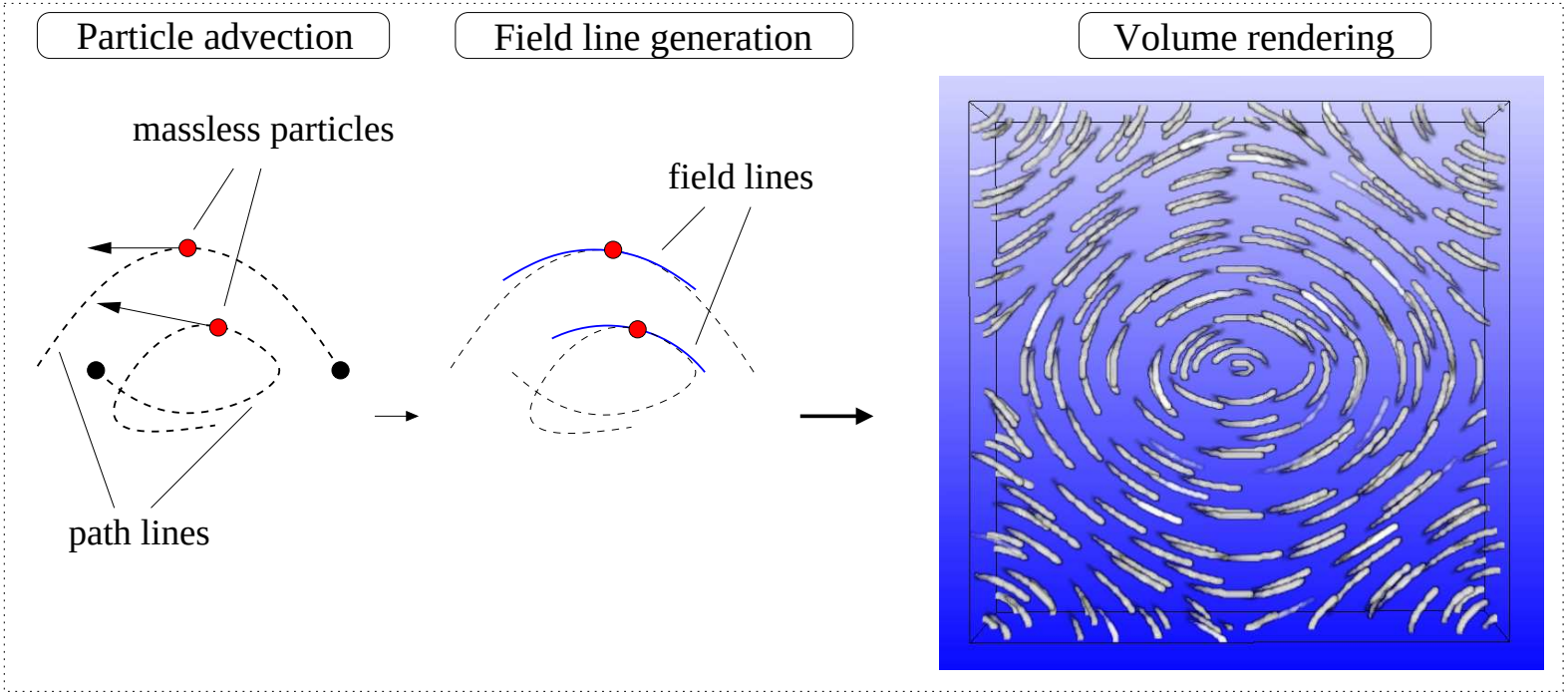
Field line opacity optimization [Gunther et. al. 2013]

- Computes local opacity values for each line segment by minimizing an error function.
- The error function takes into account both the relevance or importance of the field line as well as whether they occlude other important field lines or not.



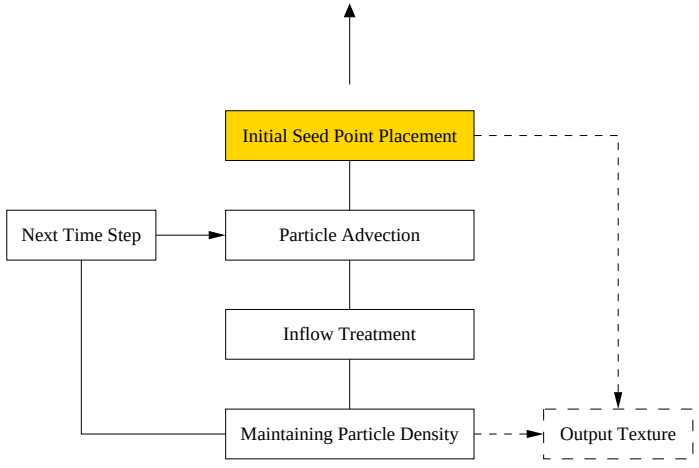
(a) Given is a set of polylines. (b) Discretize polylines into n segments (here: $n = 6$). (c) Compute per-segment opacity α_i by energy minimization. (d) Interpolate opacities between adjacent segments for final rendering.

Density based particle advection and field line rendering

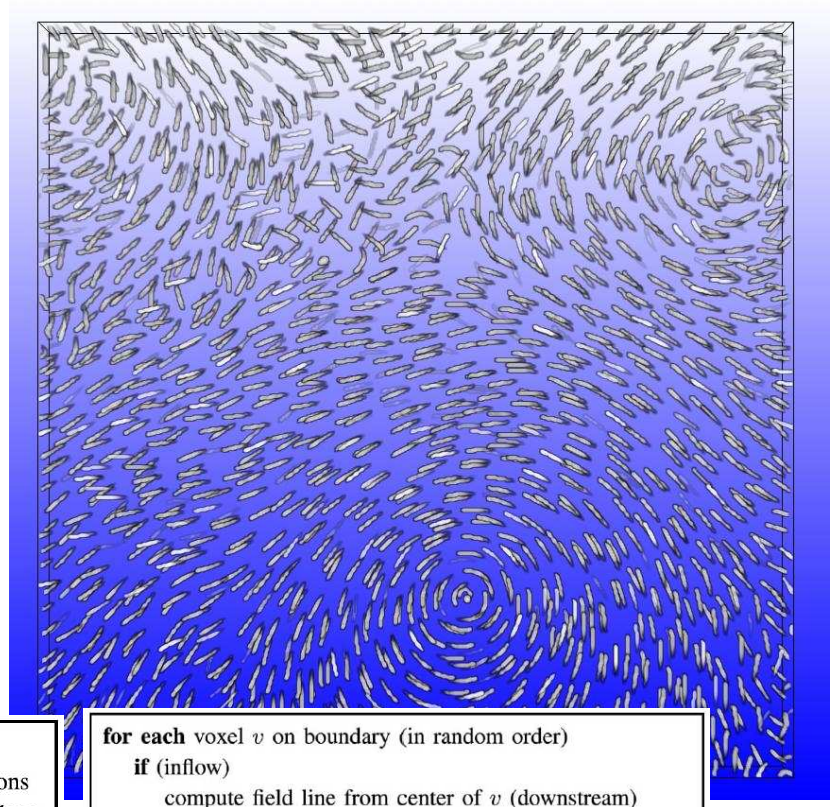


[Helgeland et. al 2006]

Density based particle advection and field line rendering



t = 0

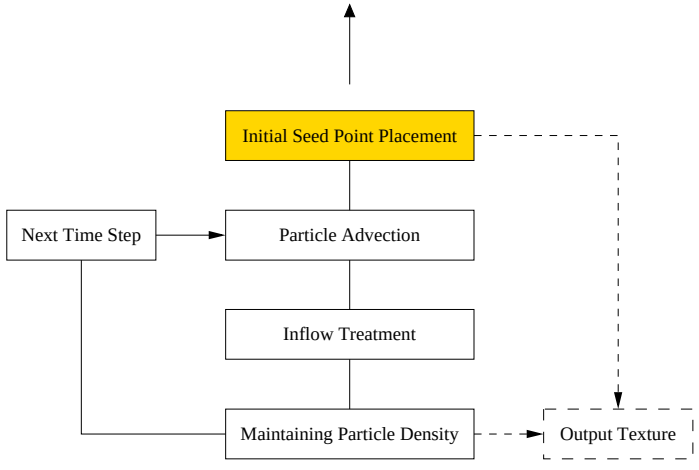


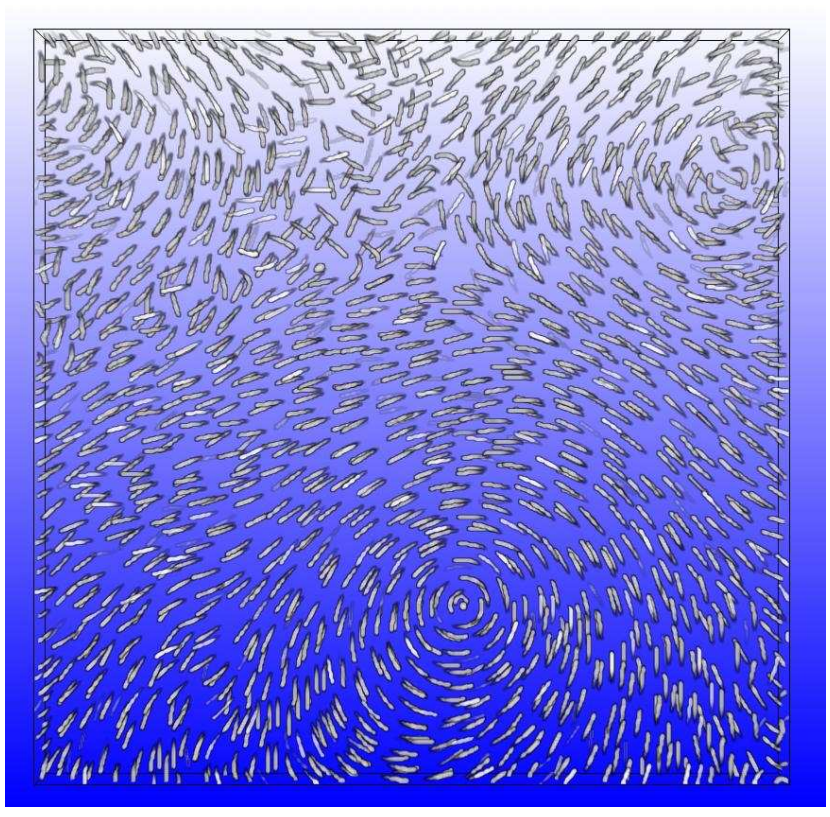
```
for each voxel v (in random order)
    compute field line from center of v in both directions
    if (some point segment on the field line falls too close
        to an already inserted field line) then
            reject seed point
    else
            mark all voxels covered by the field line
            insert seed point
```

```
for each voxel v on boundary (in random order)
    if (inflow)
        compute field line from center of v (downstream)
        if (some point segment on the field line falls too close
            to an already inserted field line) then
                reject seed point
        else
                mark all voxels covered by the field line
                insert seed point
```

[Pseudocode for initial seed point placement], [Pseudocode for inflow treatment]

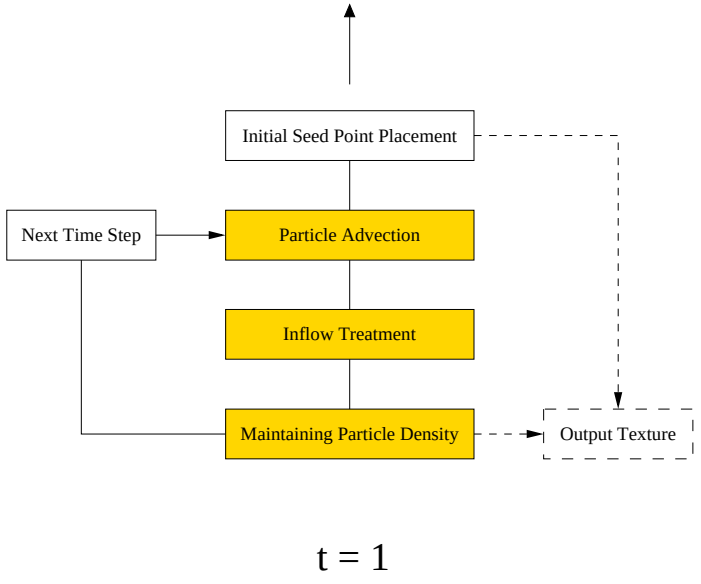
Density based particle advection and field line rendering

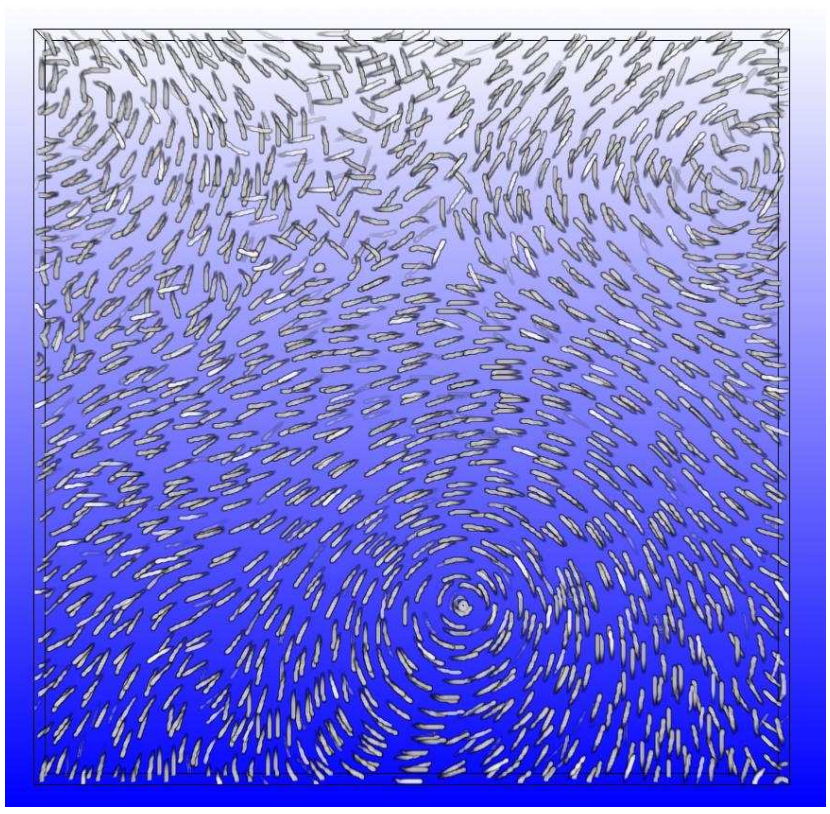




[Helgeland et. al 2006]

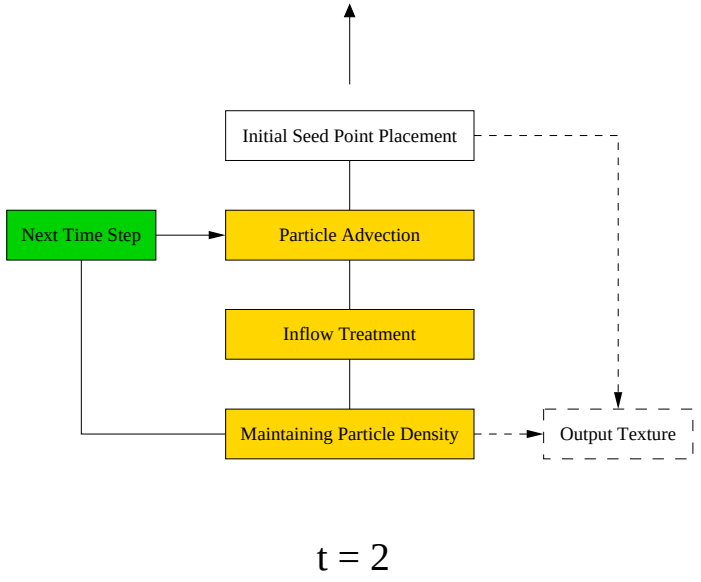
Density based particle advection and field line rendering

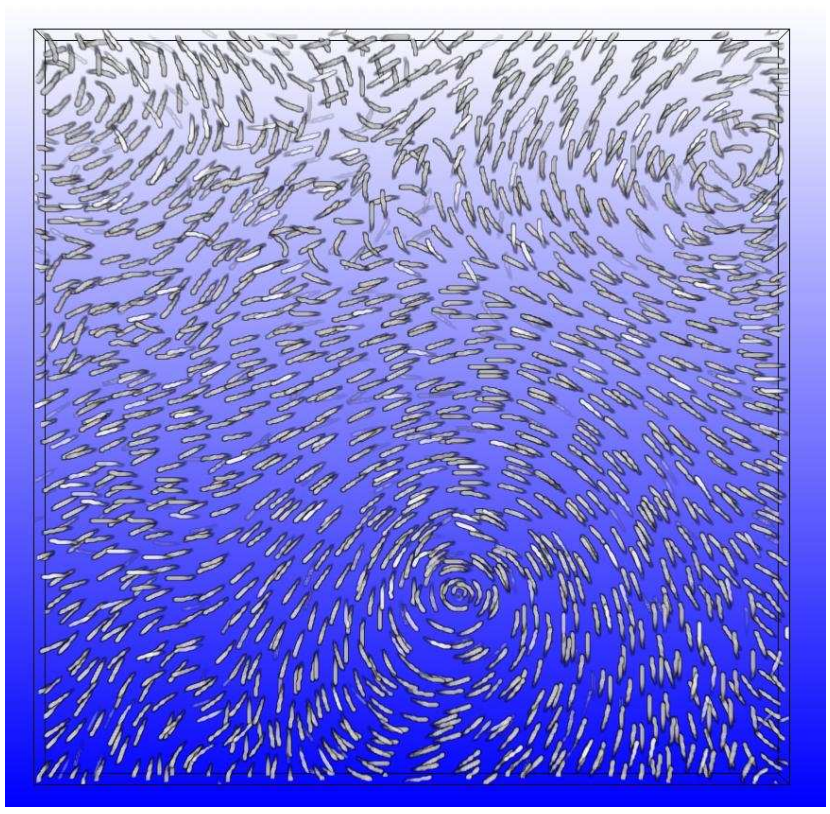




[Helgeland et. al 2006]

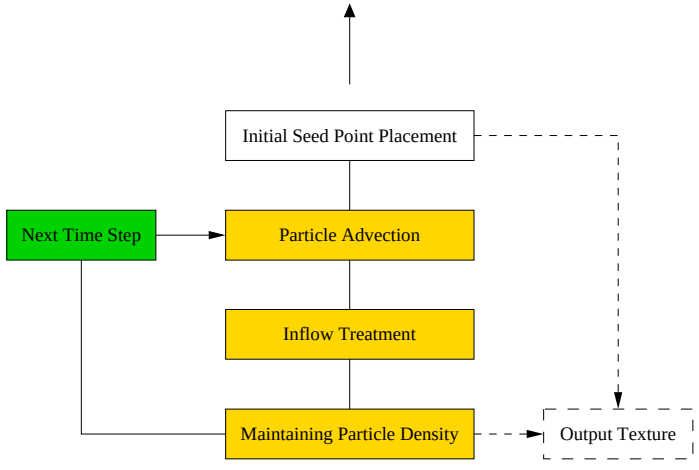
Density based particle advection and field line rendering

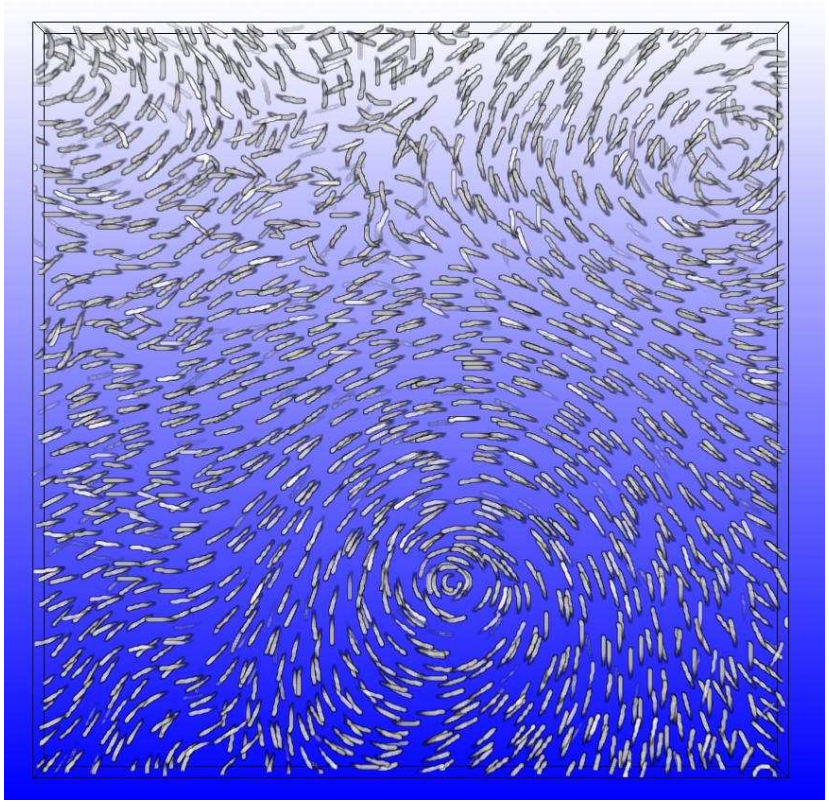




[Helgeland et. al 2006]

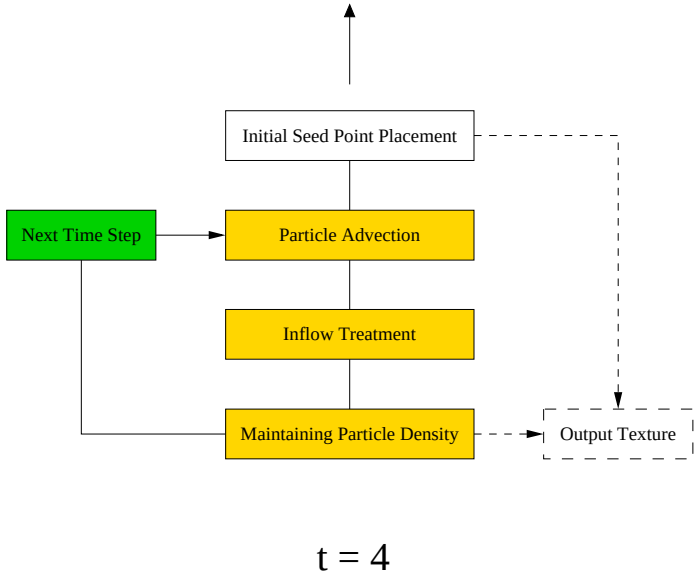
Density based particle advection and field line rendering

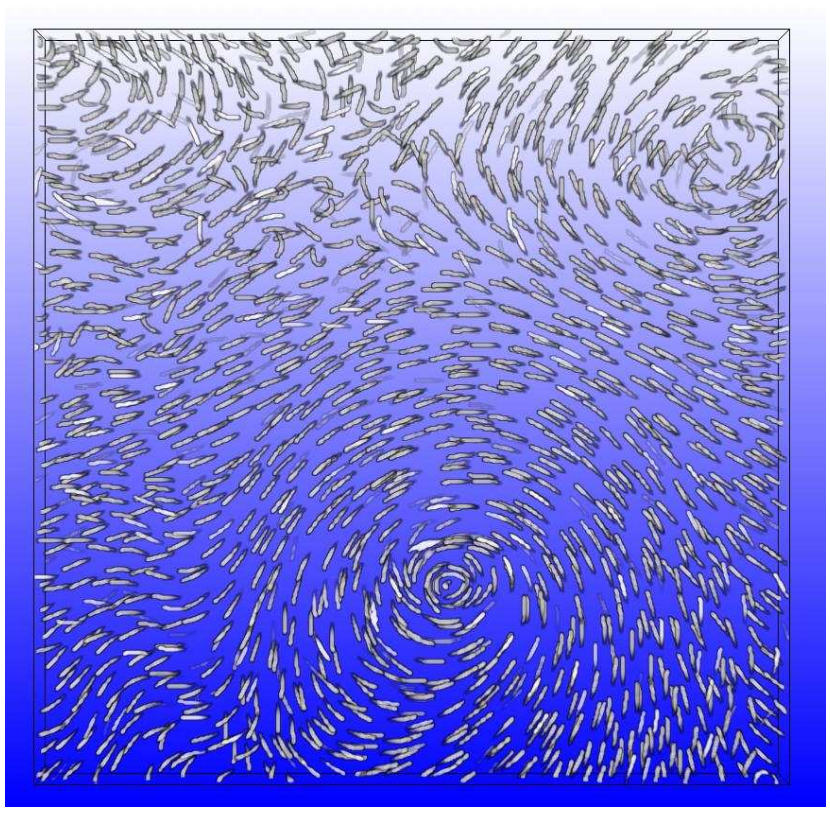




[Helgeland et. al 2006]

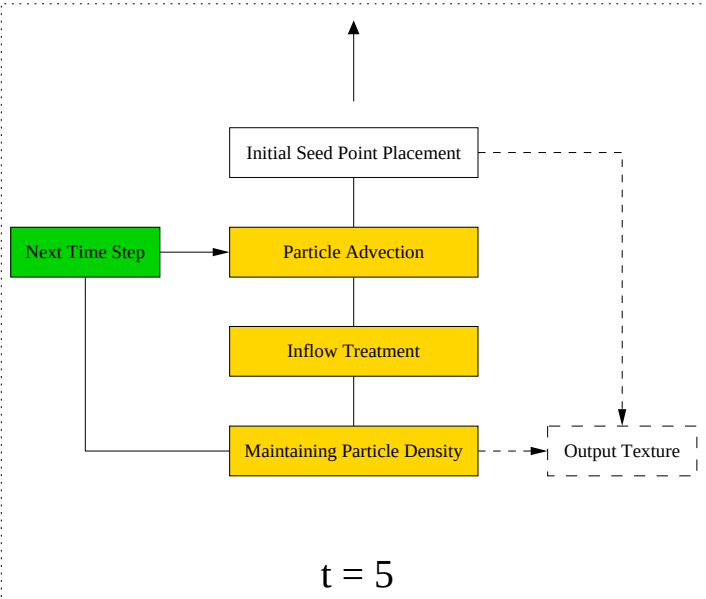
Density based particle advection and field line rendering

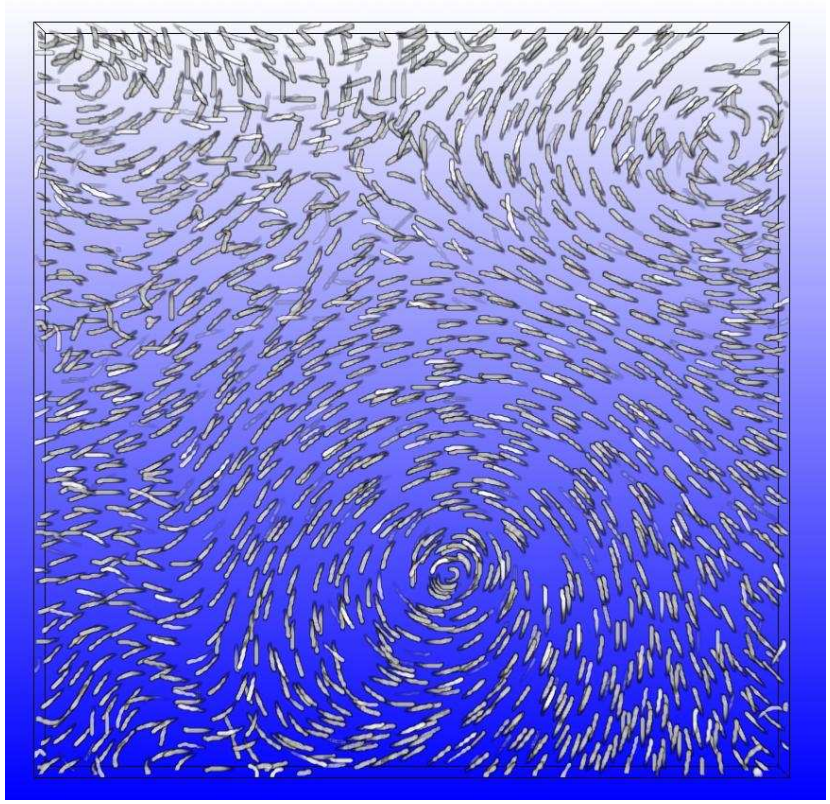




[Helgeland et. al 2006]

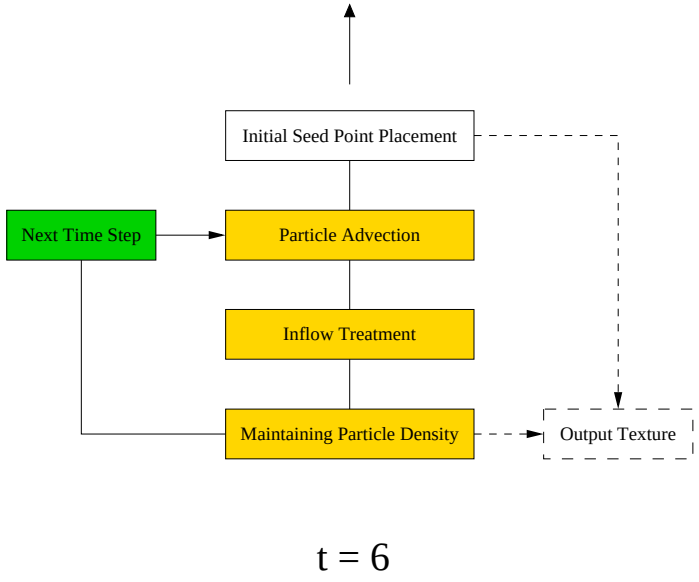
Density based particle advection and field line rendering

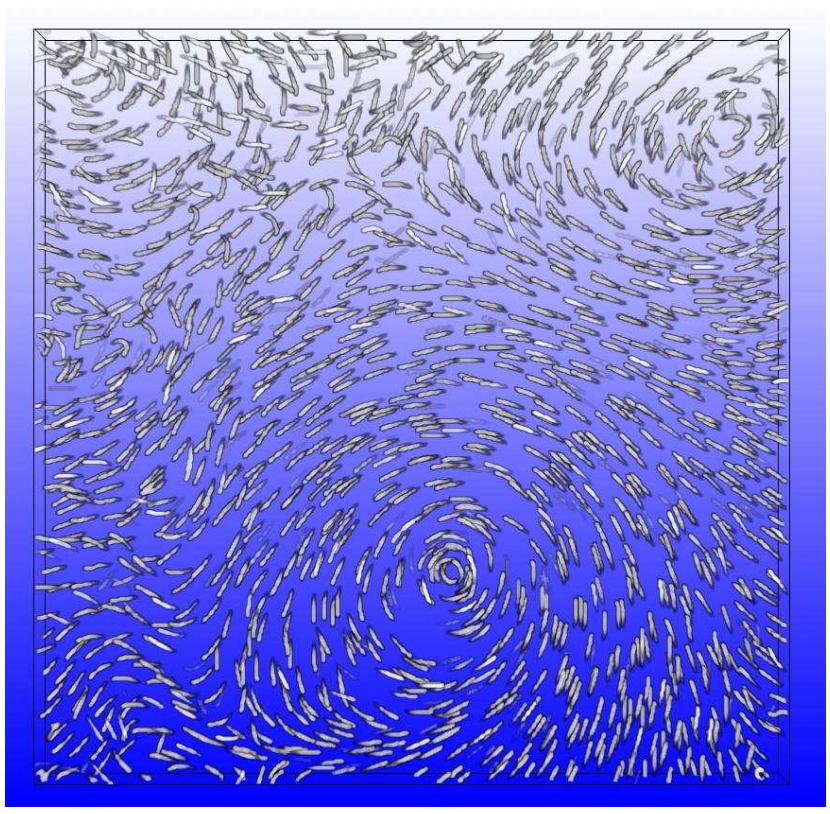




[Helgeland et. al 2006]

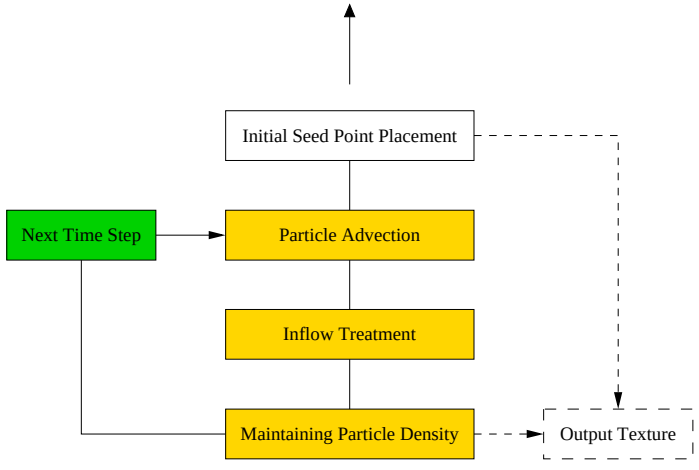
Density based particle advection and field line rendering

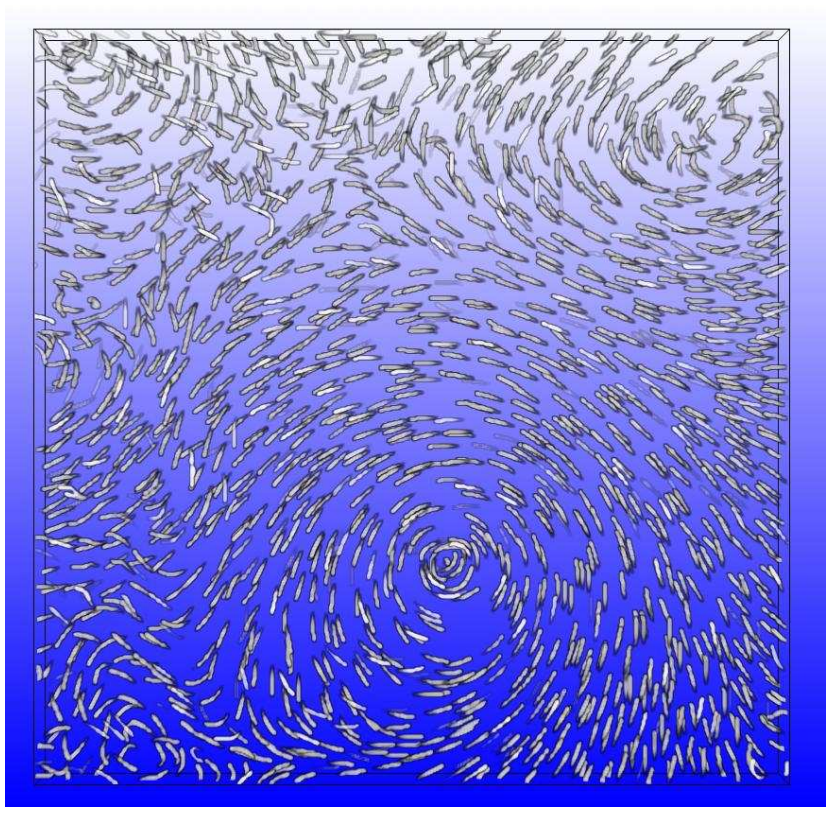




[Helgeland et. al 2006]

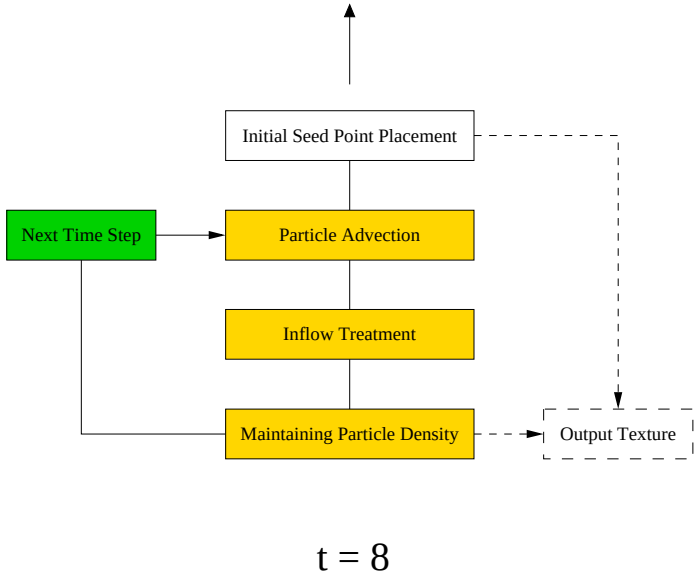
Density based particle advection and field line rendering

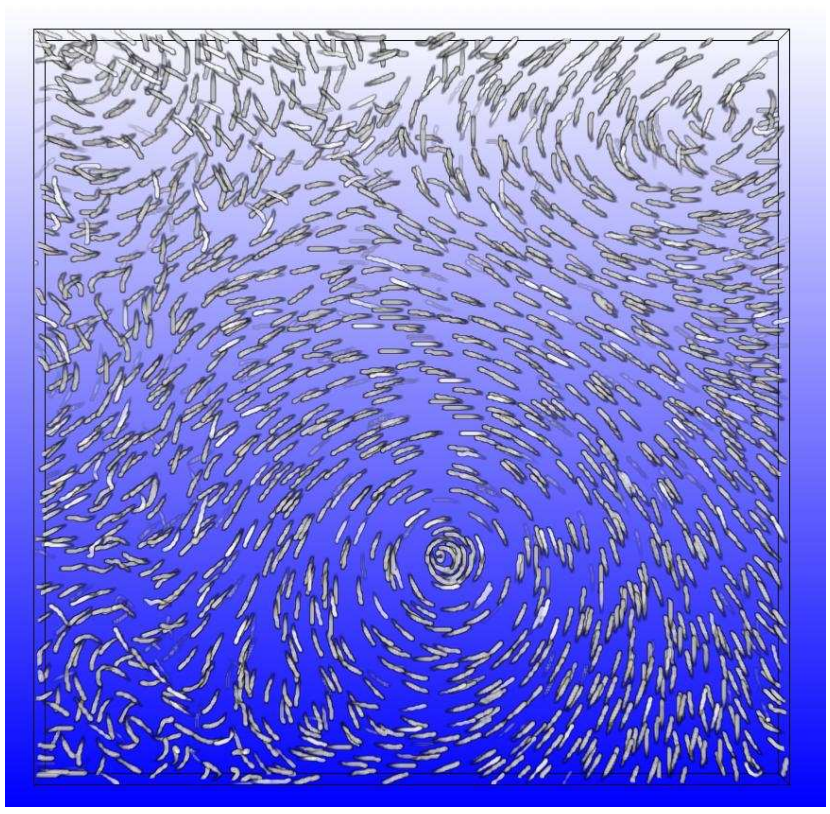




[Helgeland et. al 2006]

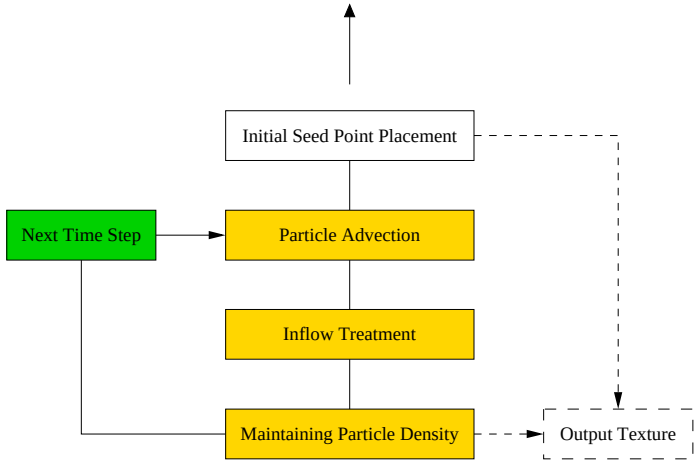
Density based particle advection and field line rendering



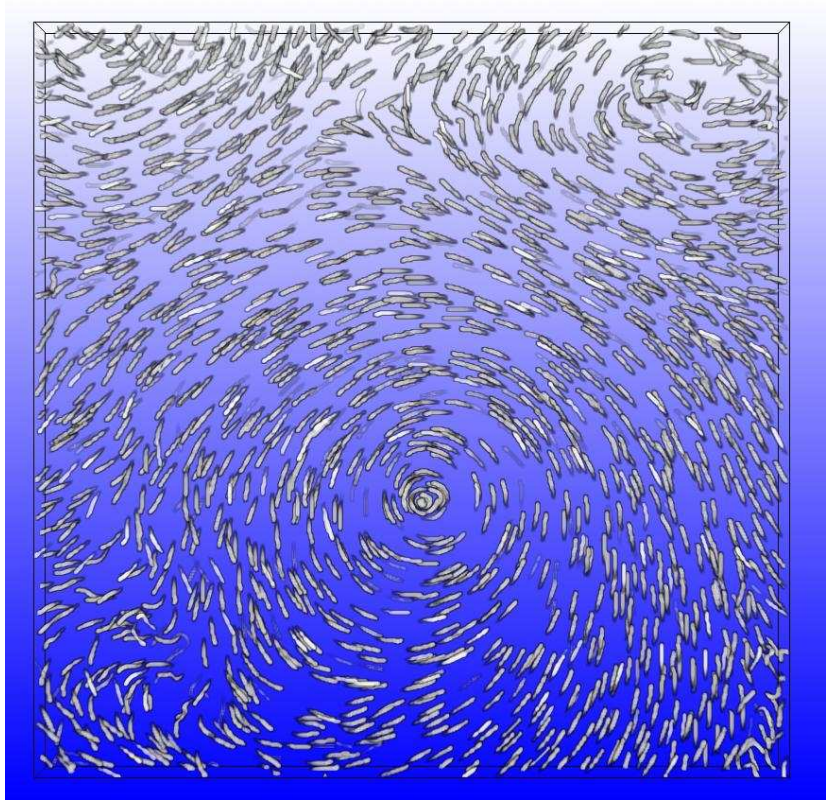


[Helgeland et. al 2006]

Density based particle advection and field line rendering

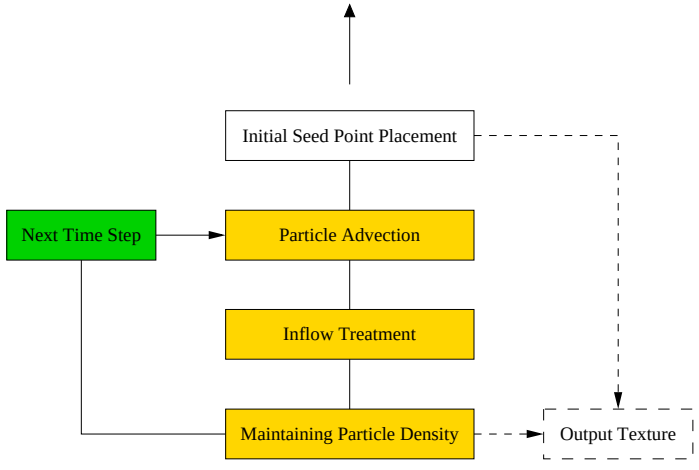


t = 18

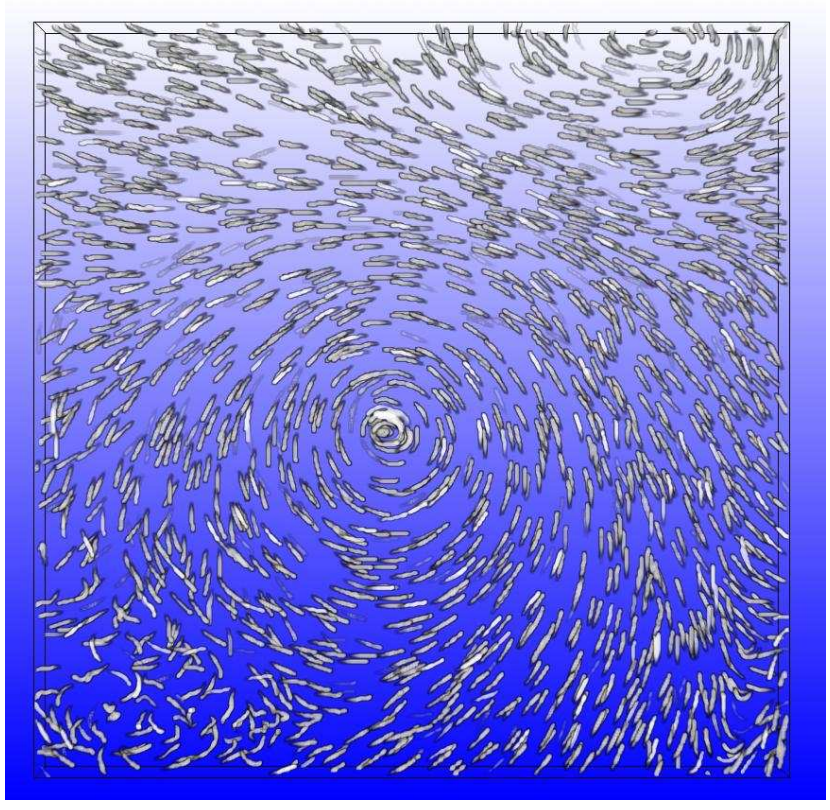


[Helgeland et. al 2006]

Density based particle advection and field line rendering

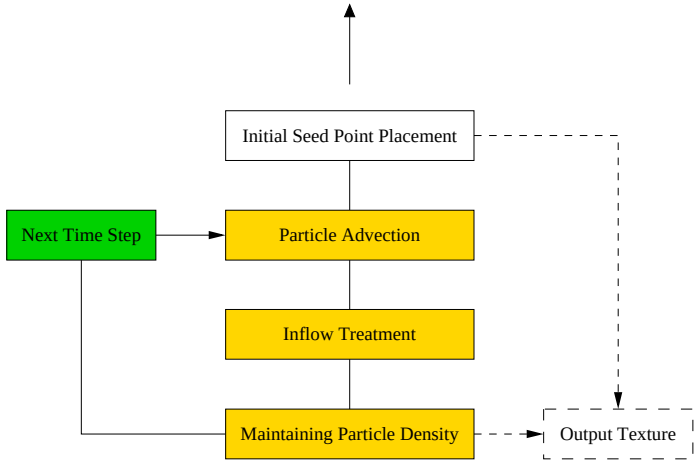


t = 28

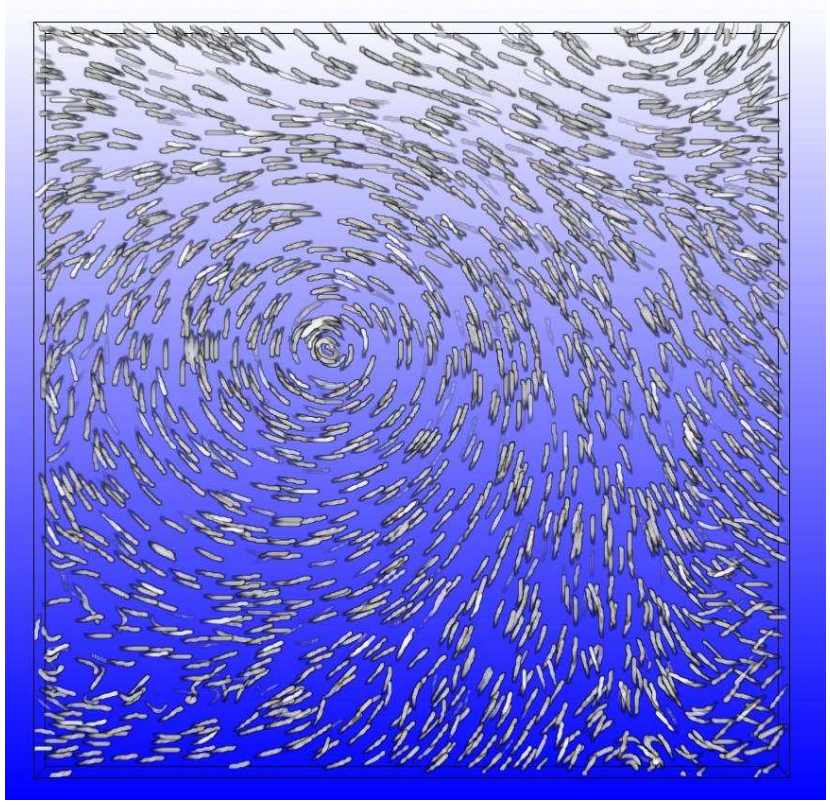


[Helgeland et. al 2006]

Density based particle advection and field line rendering

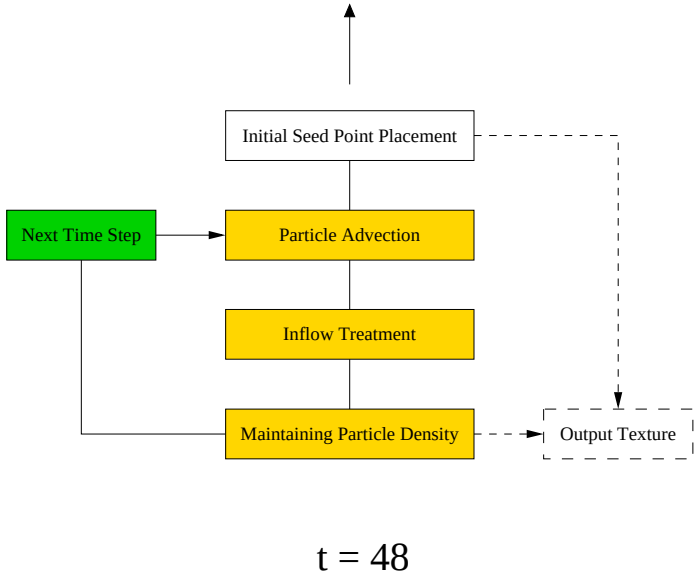


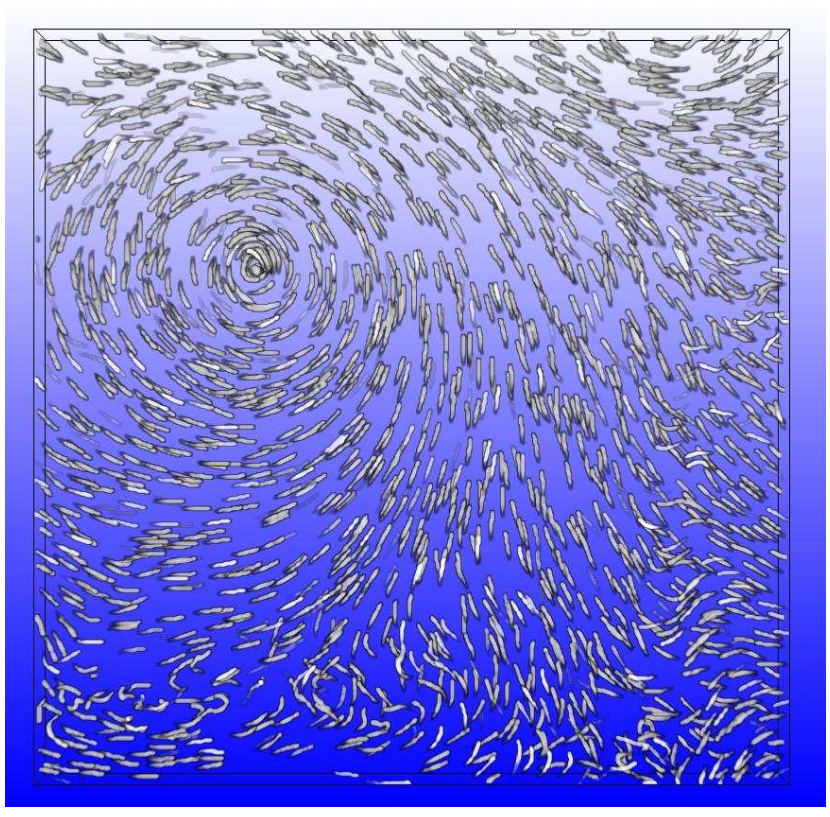
t = 38



[Helgeland et. al 2006]

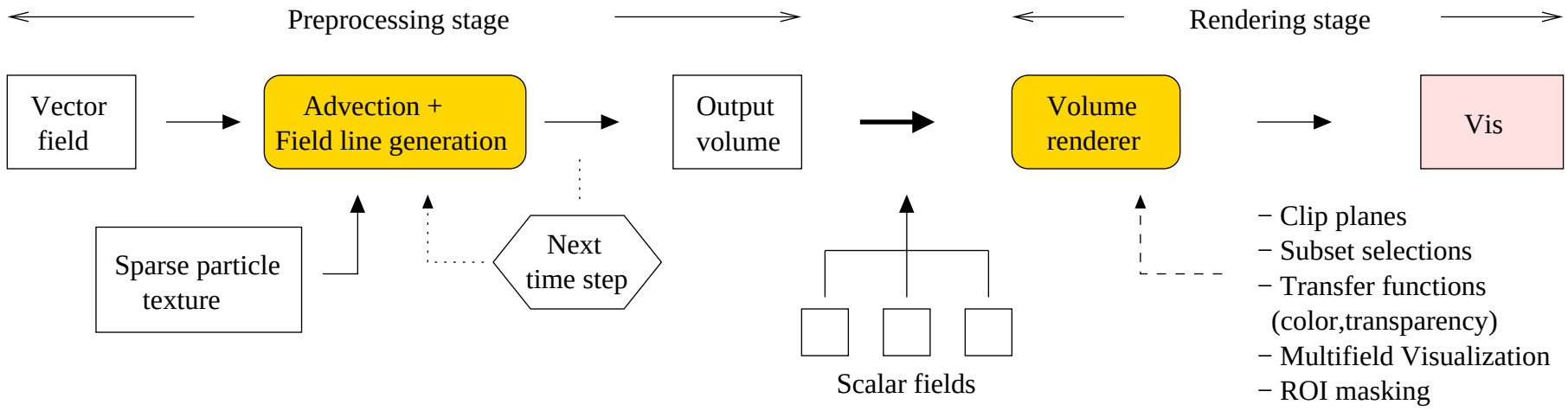
Density based particle advection and field line rendering

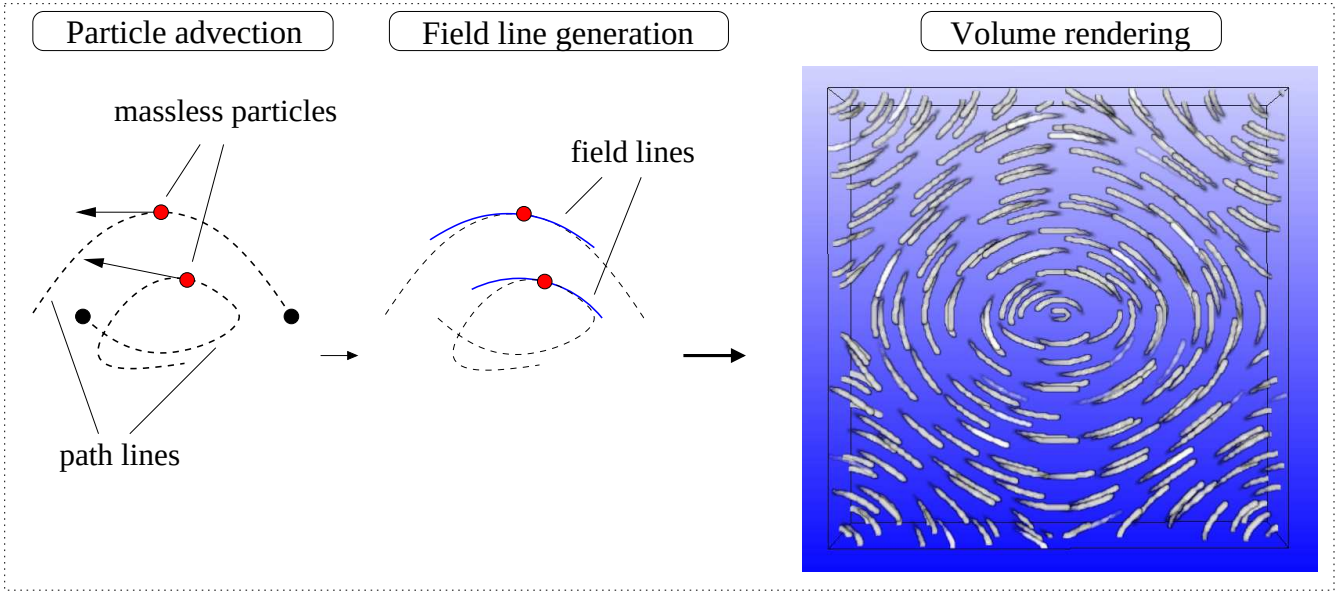




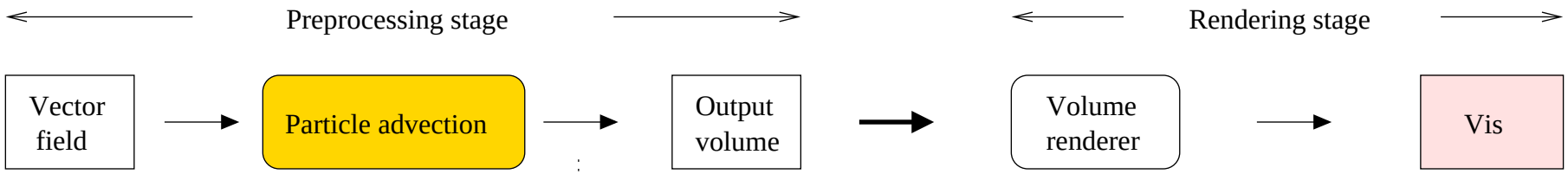
[Helgeland et. al 2006]

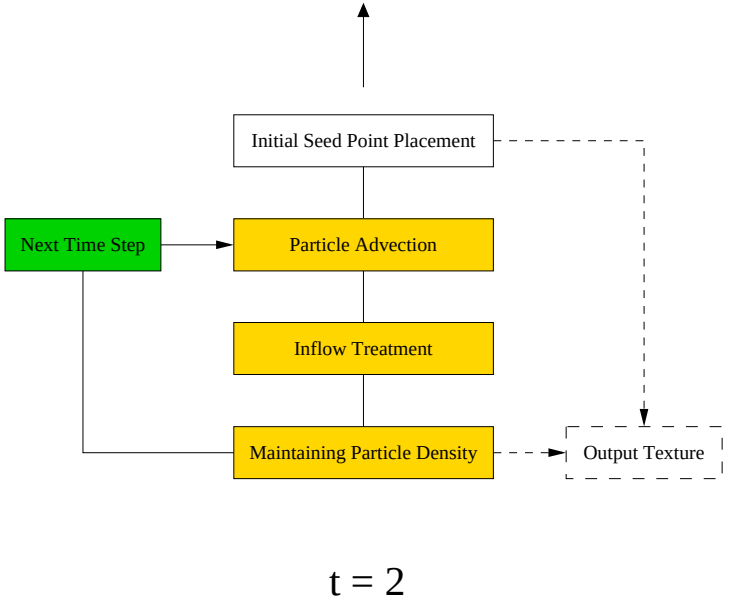
Sparse and global texture-based flow visualization



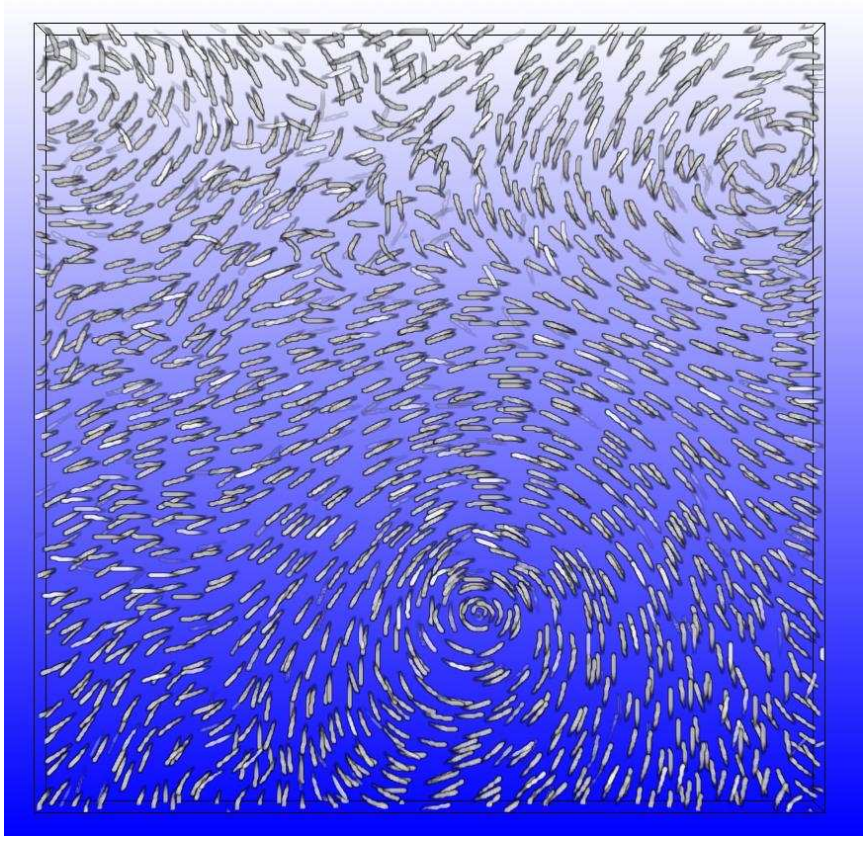


Particle advection



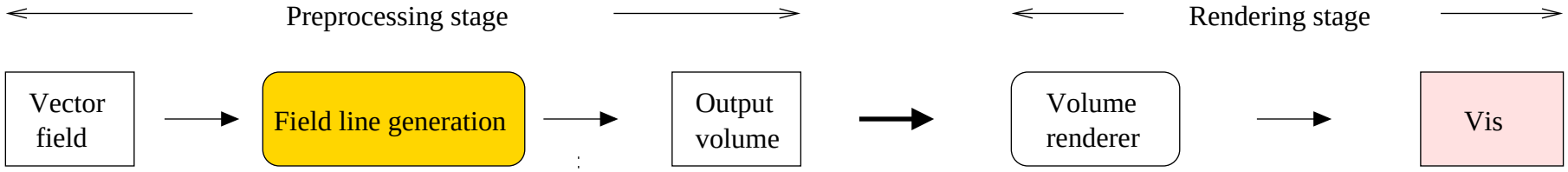


t = 2



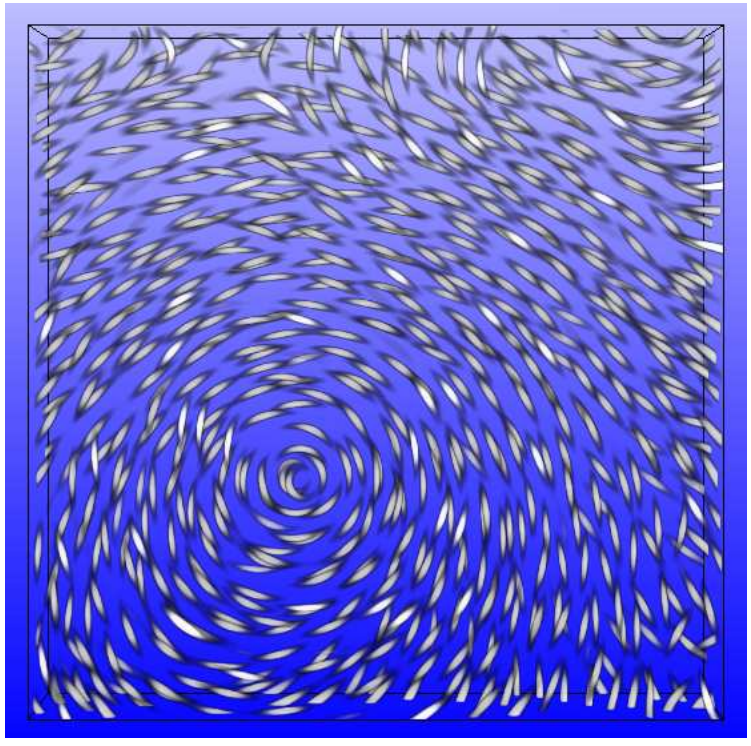
on

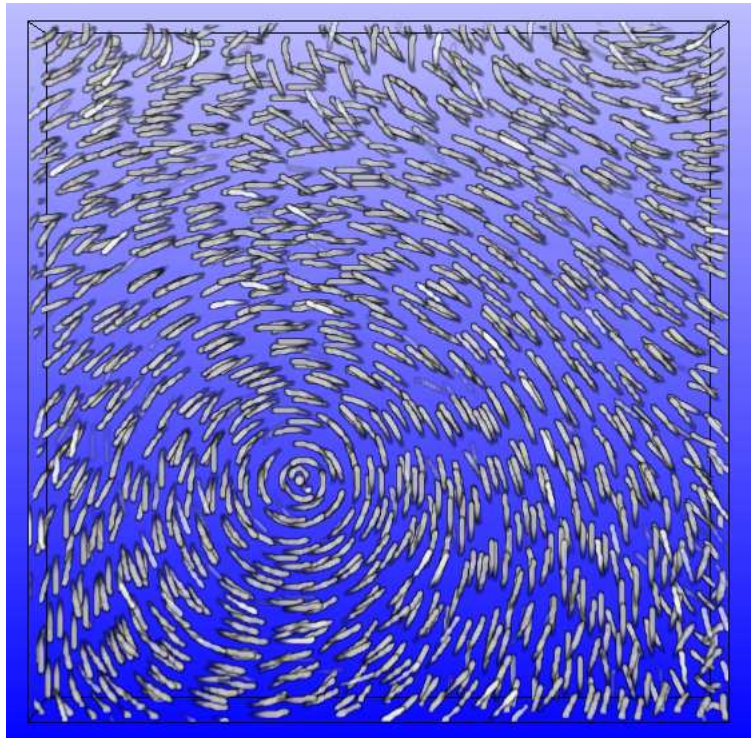
Field line generation



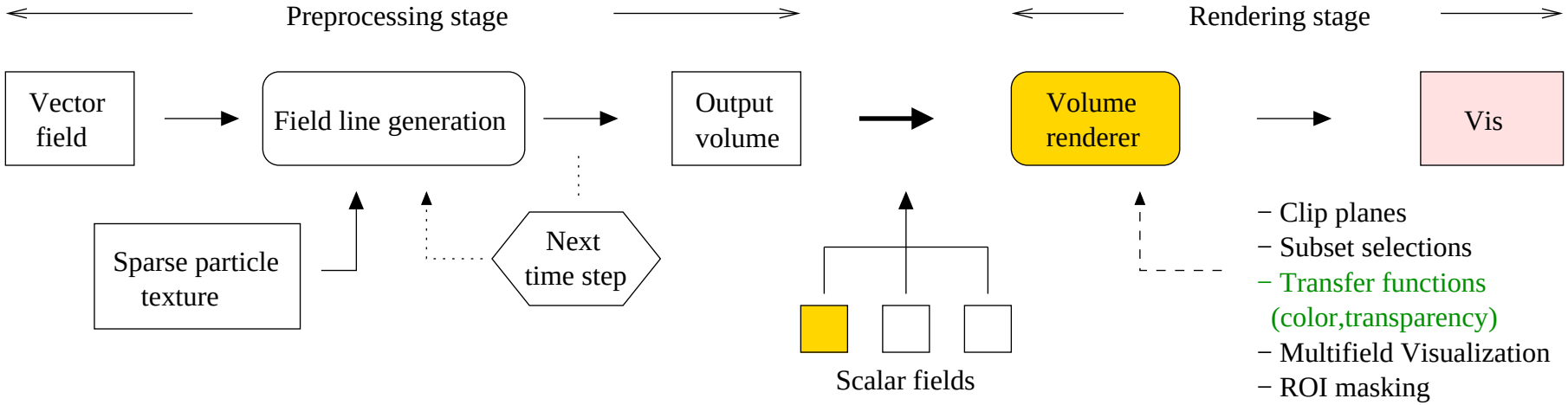
Cigar−shaped field lines:

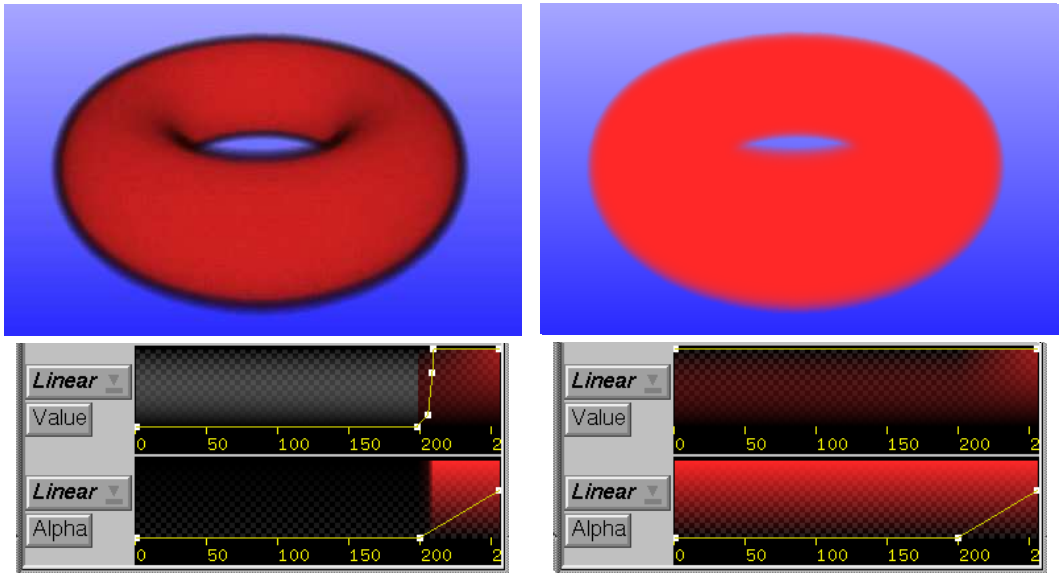
Comet−shaped field lines:

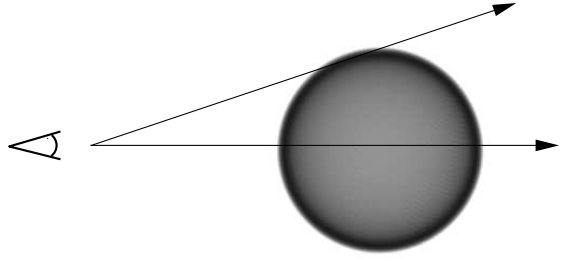




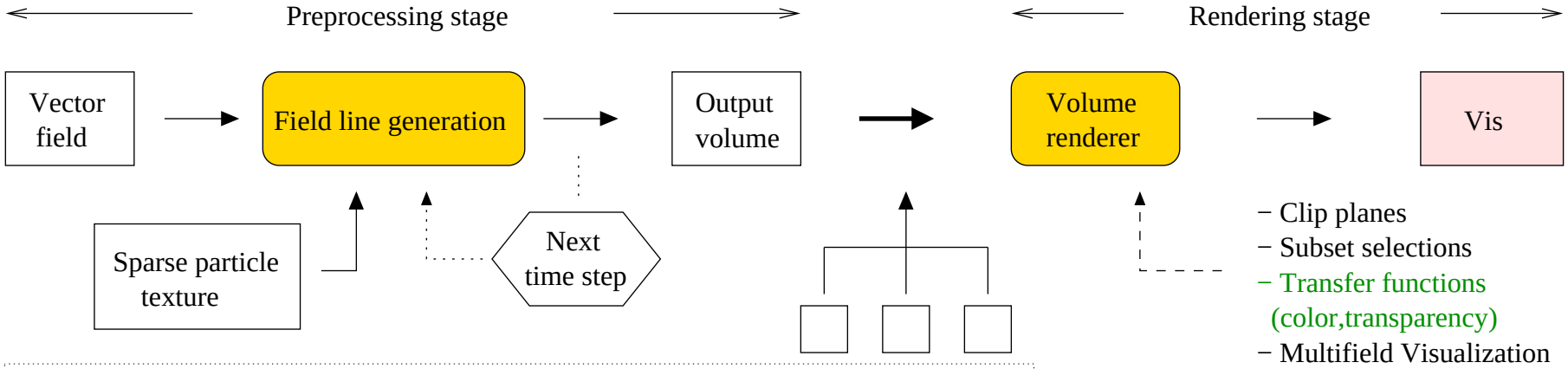
Limb darkening





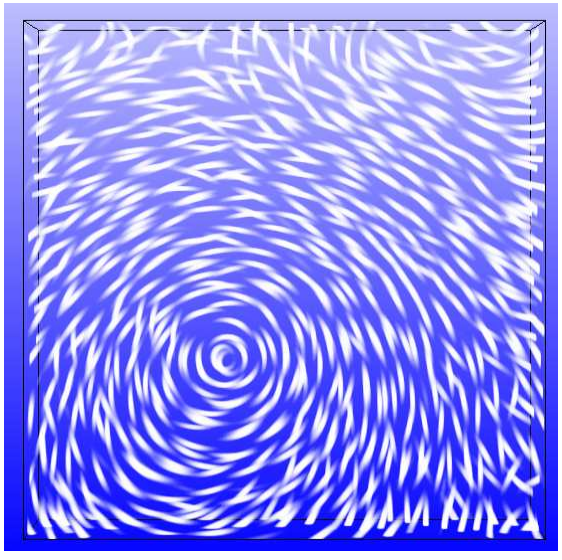


Field line generation

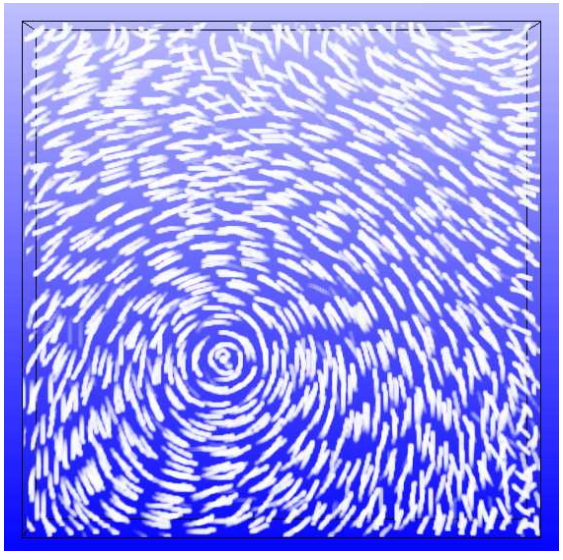


- Clip planes
- Subset selections
- Transfer functions (color,transparency)
- Multifield Visualization
- ROI masking

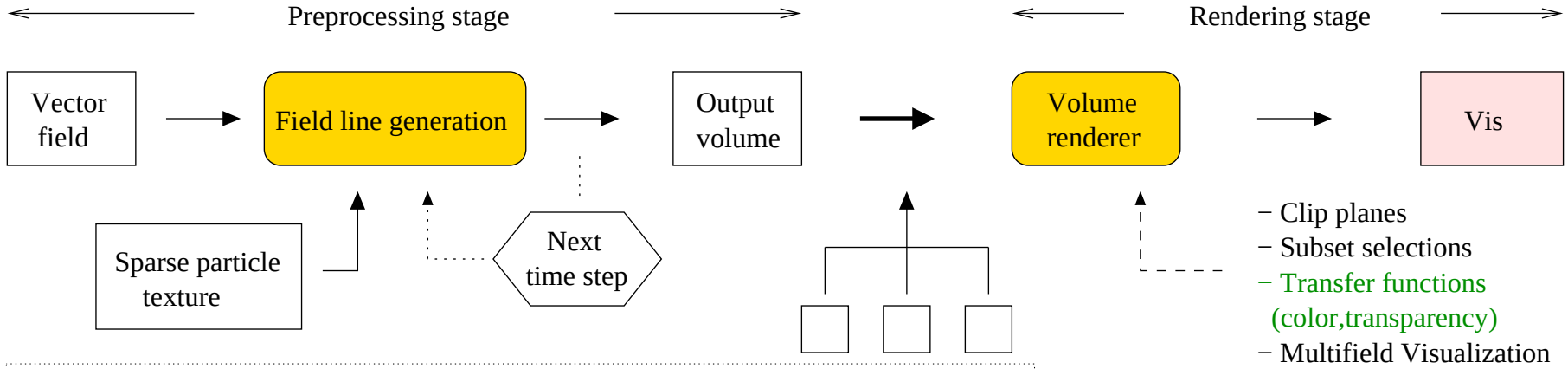
Cigar−shaped field lines



Comet−shaped field lines

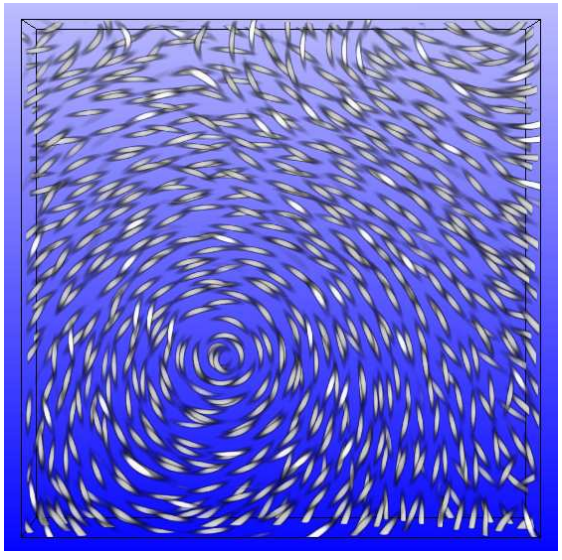


Field line generation

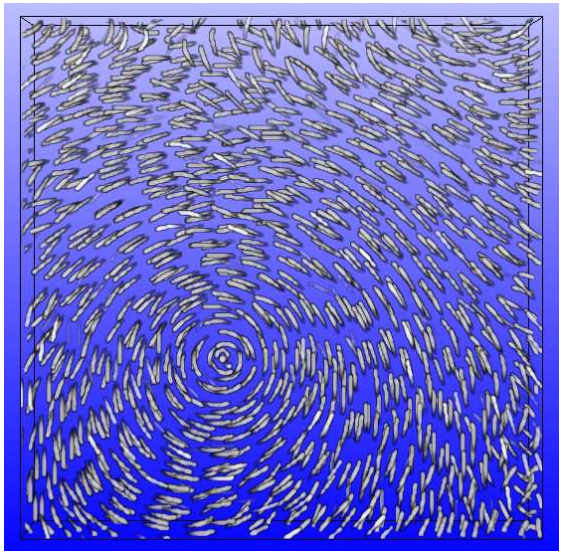


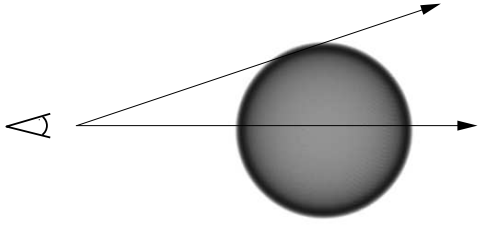
- Clip planes
- Subset selections
- Transfer functions (color,transparency)
- Multifield Visualization
- ROI masking

Cigar−shaped field lines

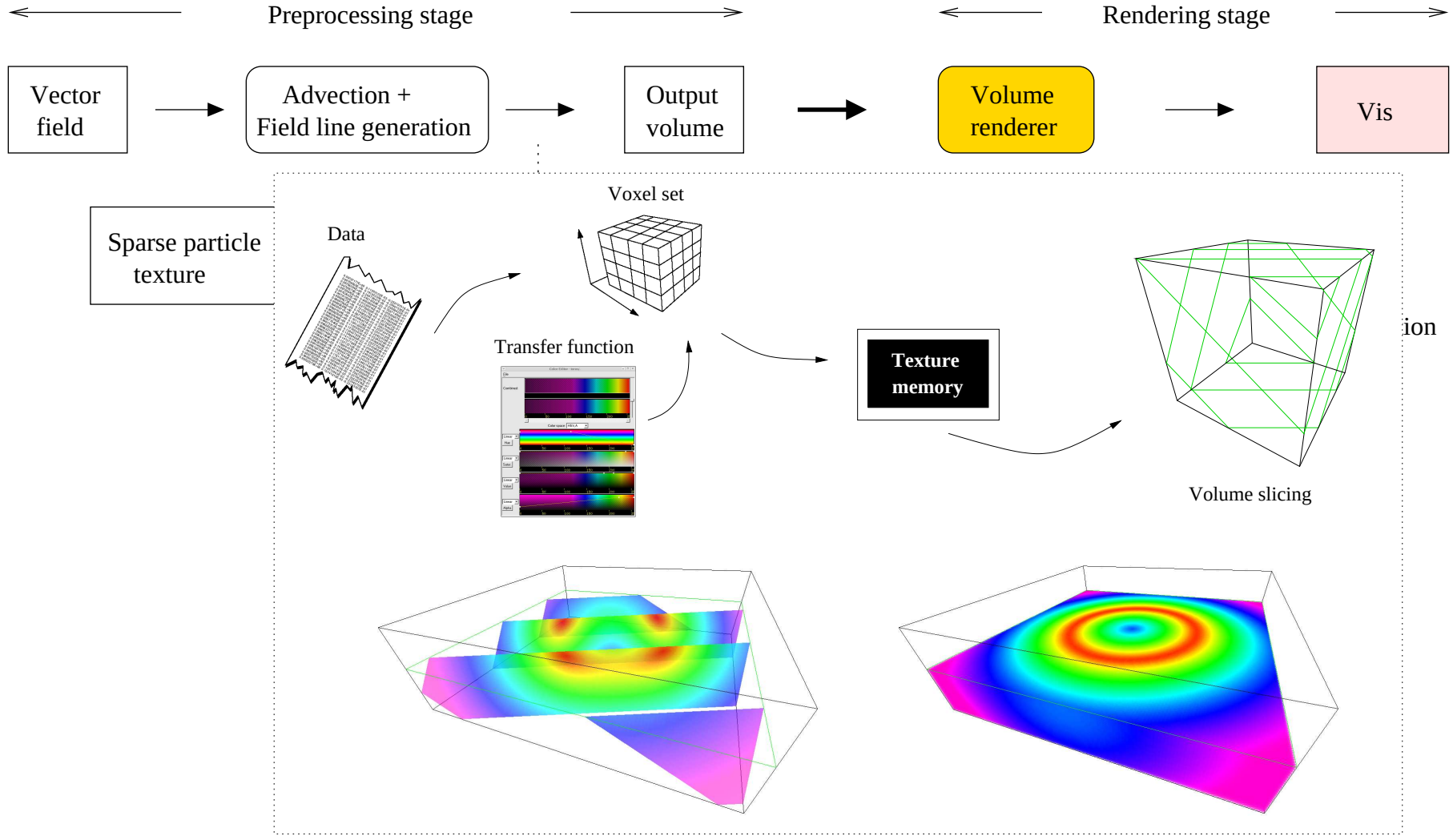


Comet−shaped field lines

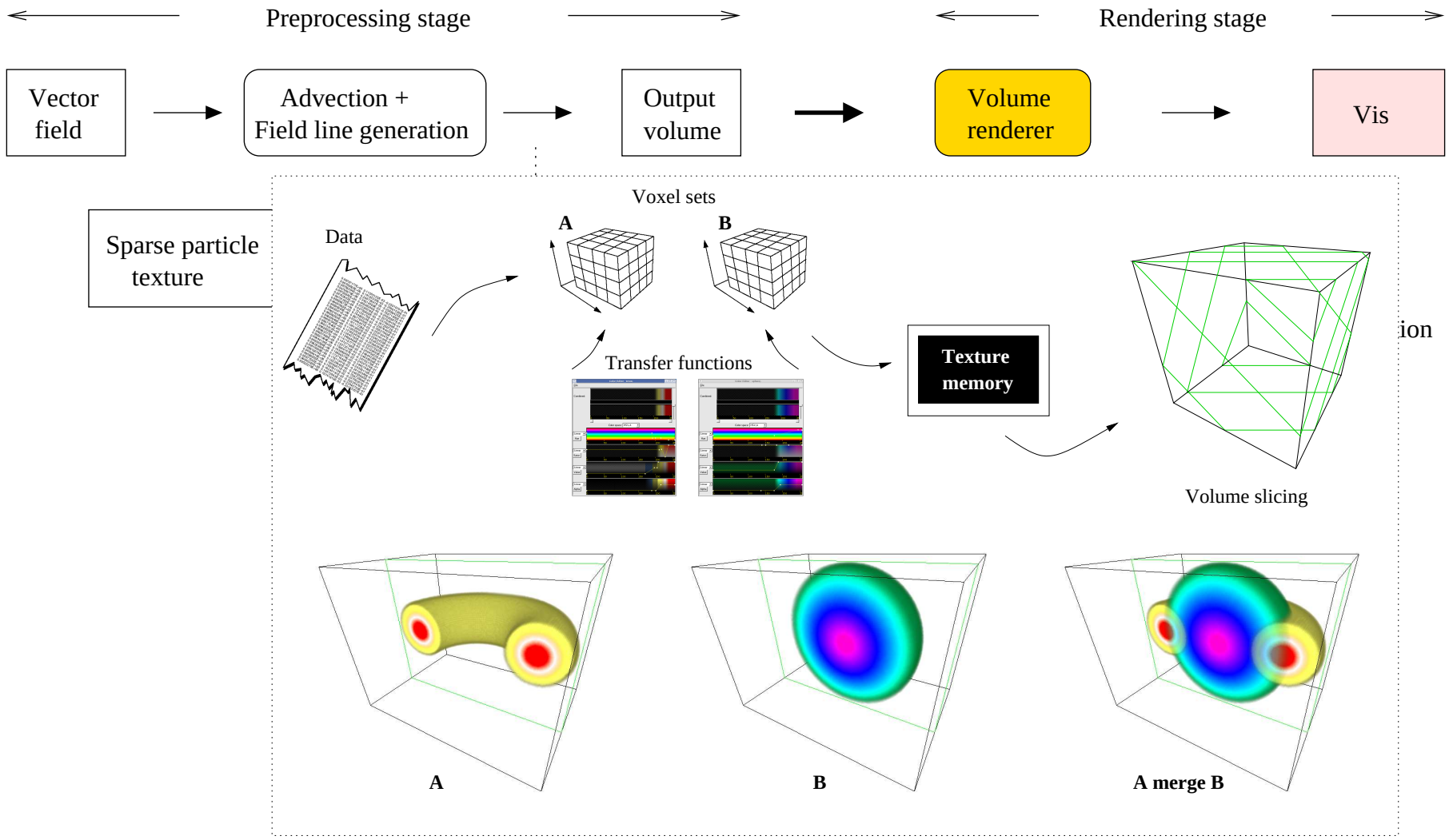




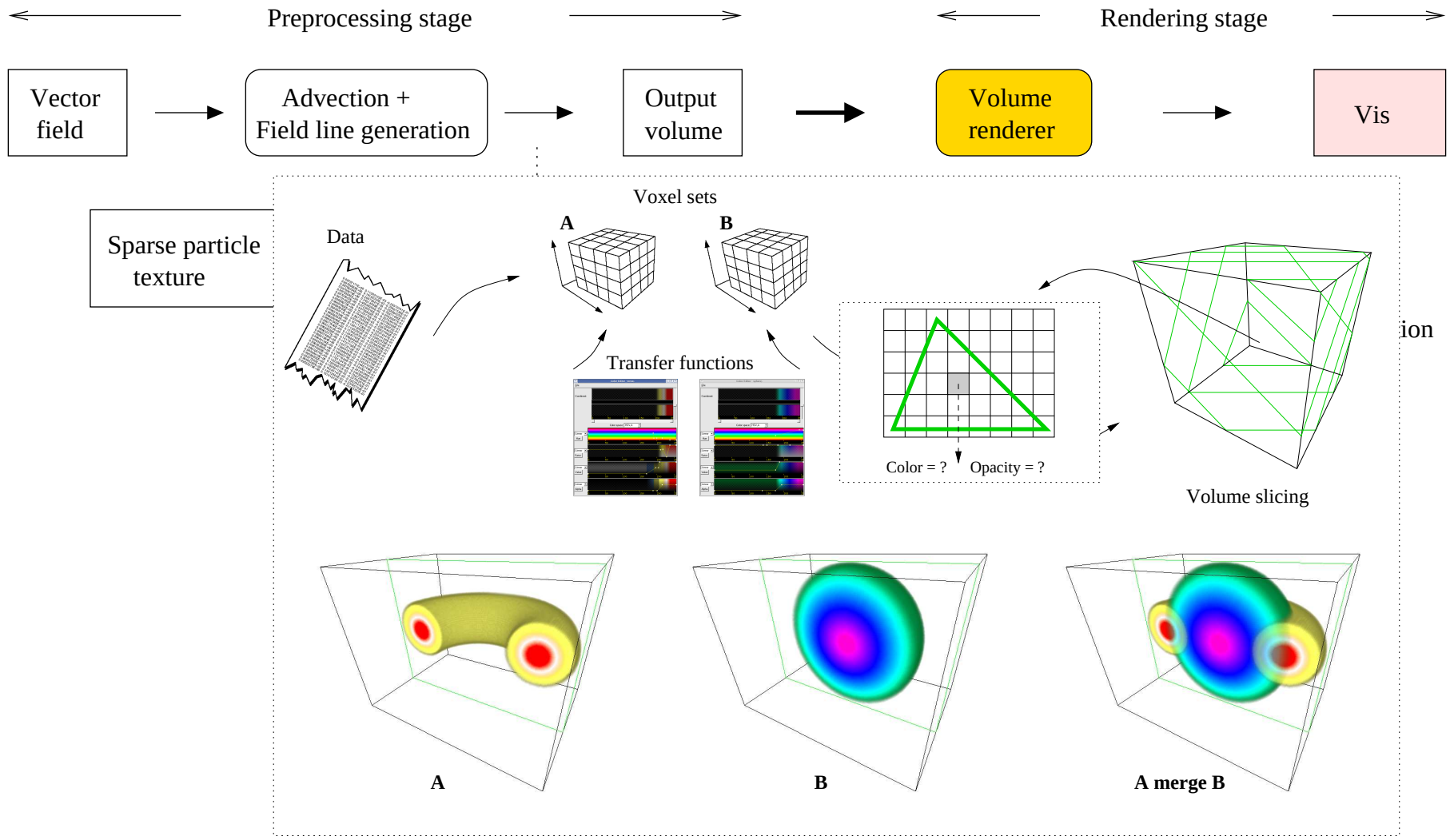
Texture-based direct volume rendering



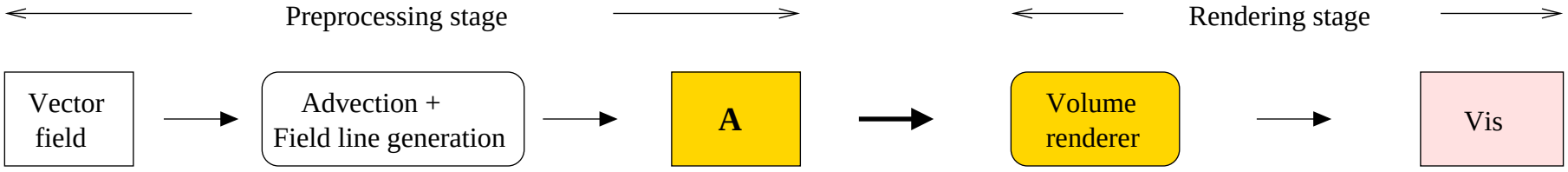
Multifield volume rendering

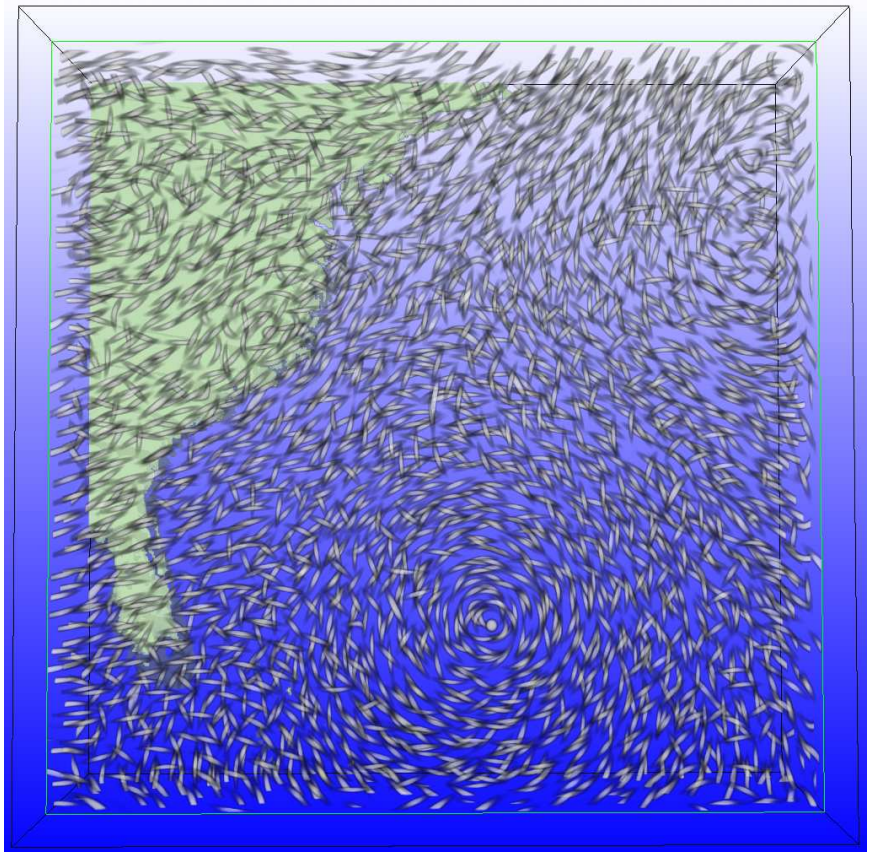


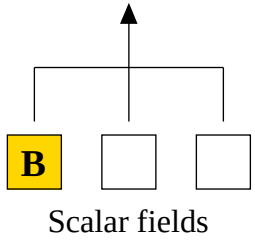
Multifield volume rendering



Multifield volume rendering





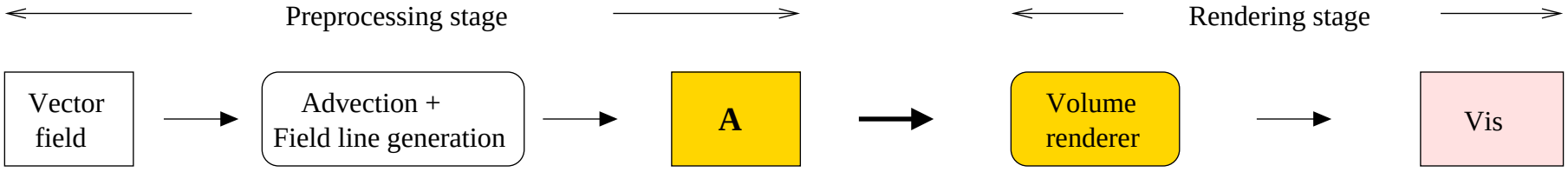


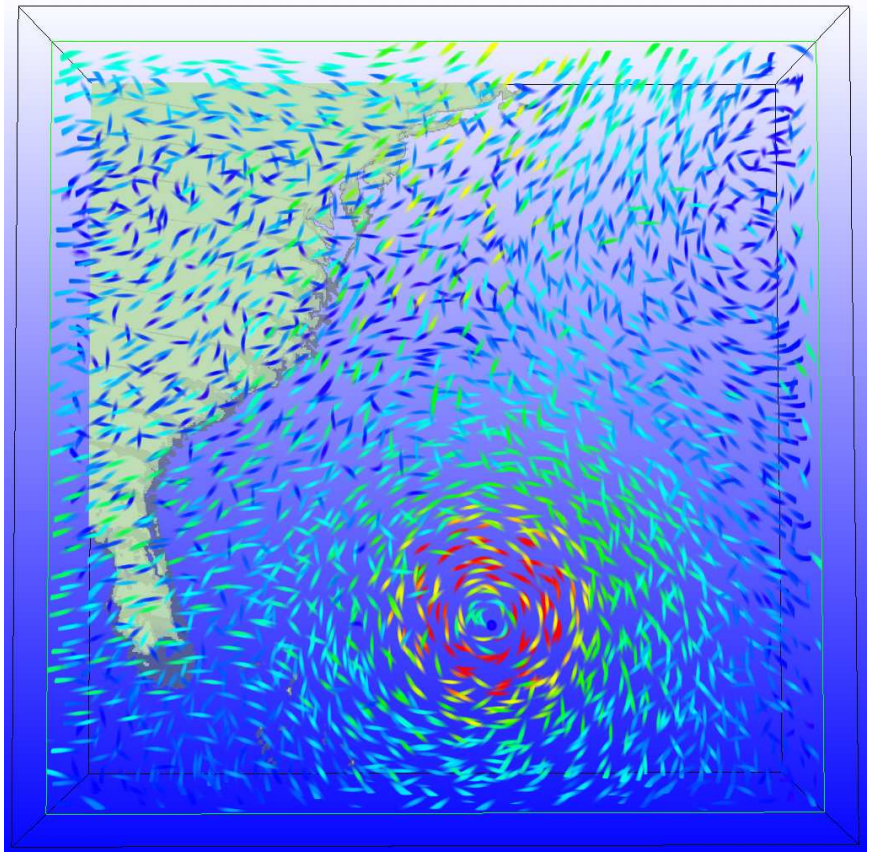
- Clip planes
- Subset selections
- Transfer functions (color,transparency)
- Multifield Visualization
- ROI masking

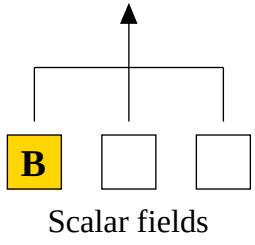
A = Velocity
B = Velocity magnitude

[color,opacity] = A

Multifield volume rendering





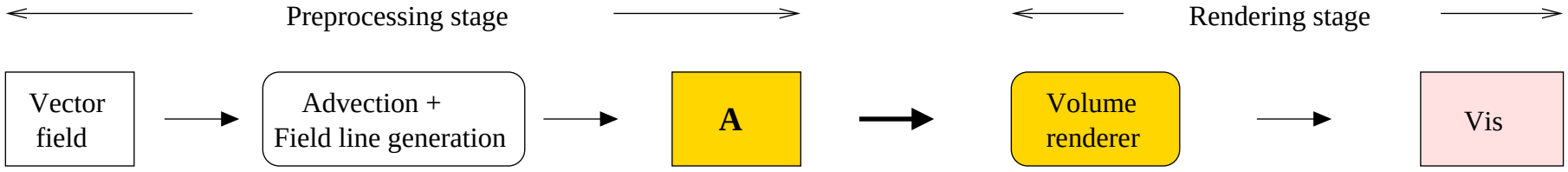


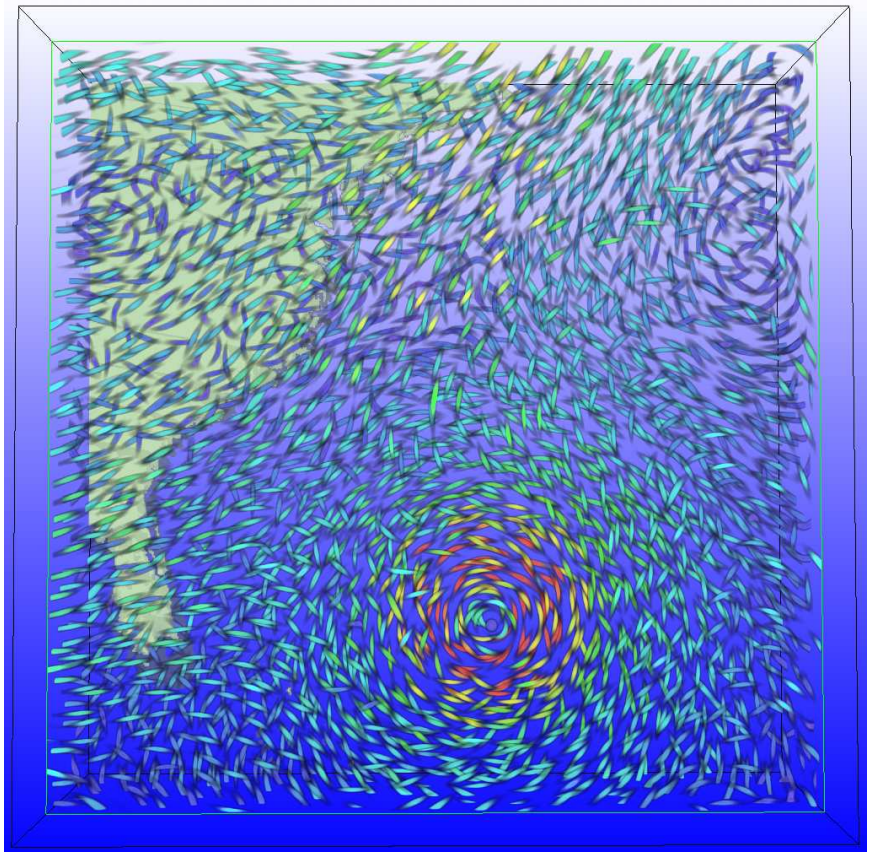
- Clip planes
- Subset selections
- Transfer functions (color,transparency)
- Multifield Visualization
- ROI masking

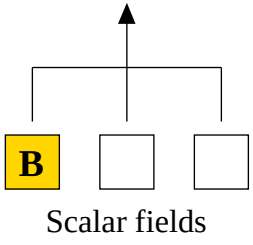
A = Velocity
B = Velocity magnitude

[color,opacity] $= [B, A]$

Multifield volume rendering





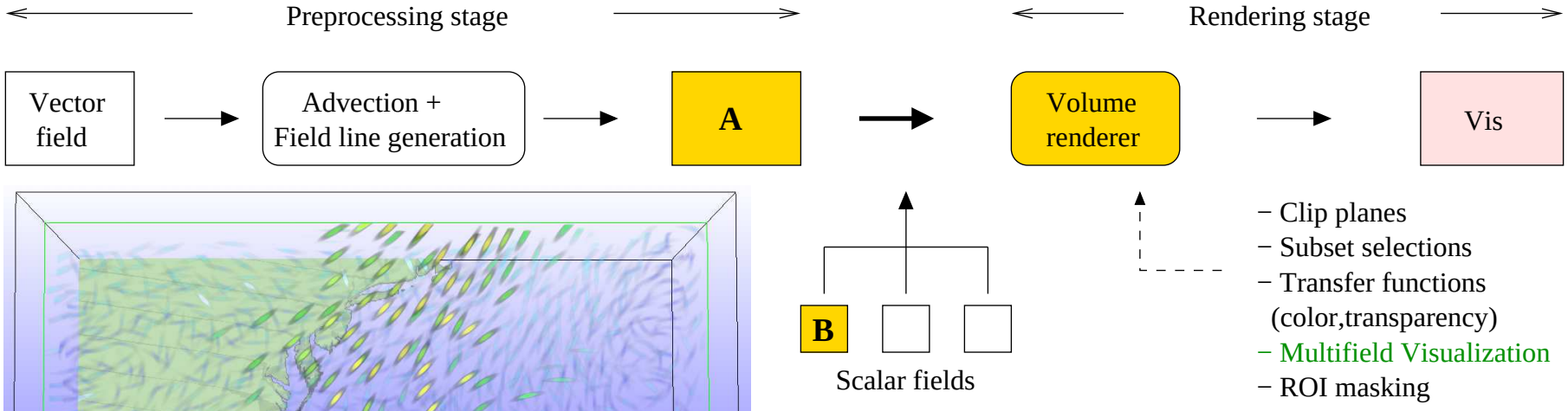


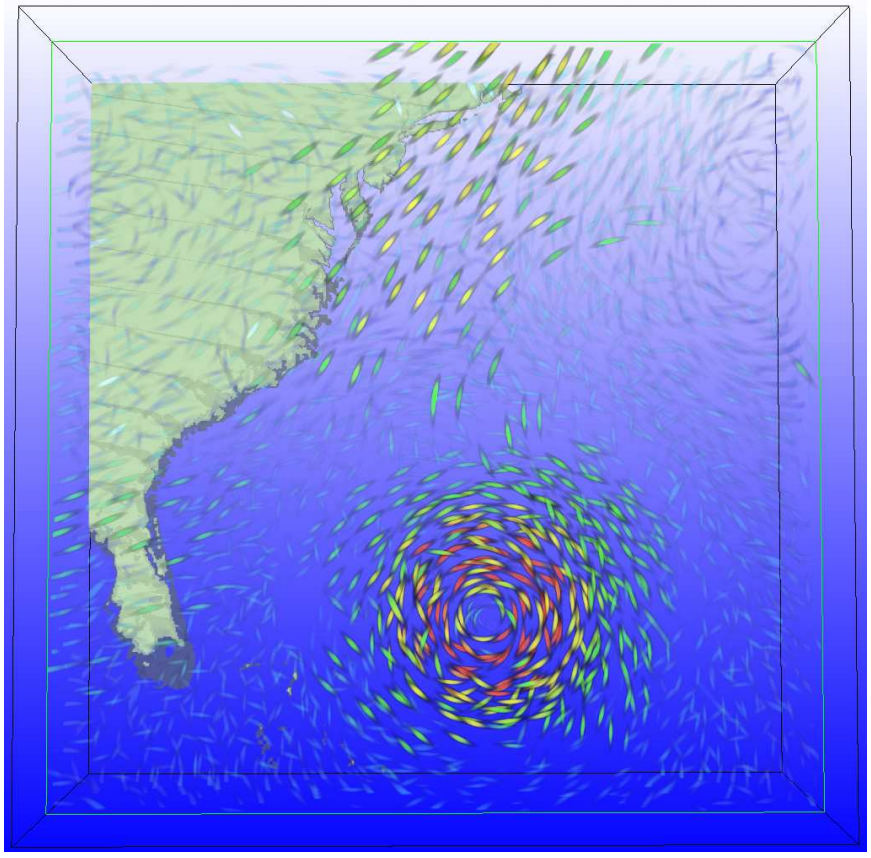
- Clip planes
- Subset selections
- Transfer functions (color,transparency)
- Multifield Visualization
- ROI masking

A = Velocity
B = Velocity magnitude

[color,opacity] = *merge*$(A, [B, A])$

Multifield volume rendering

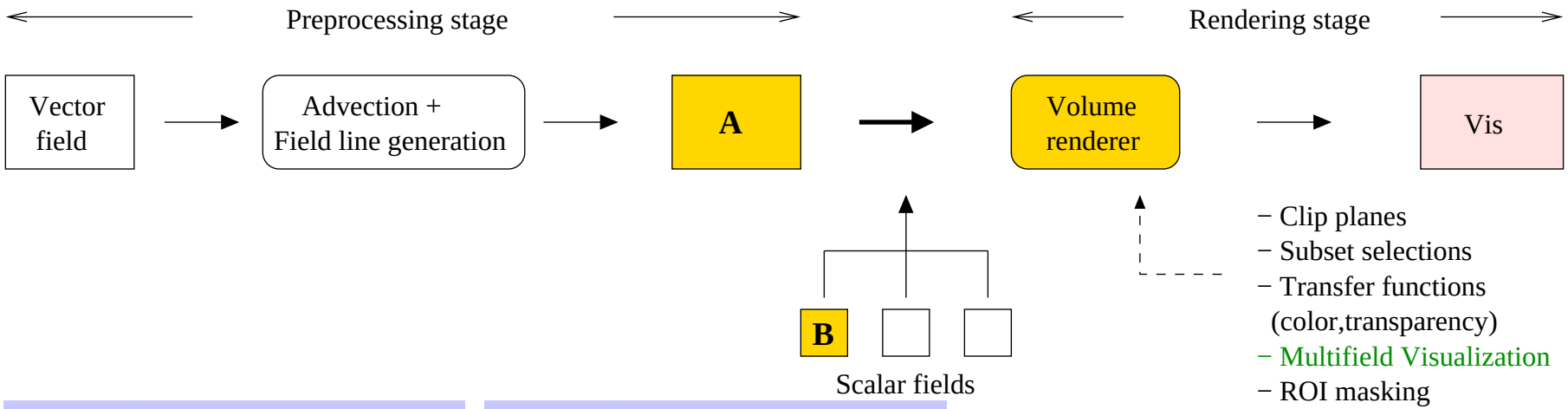


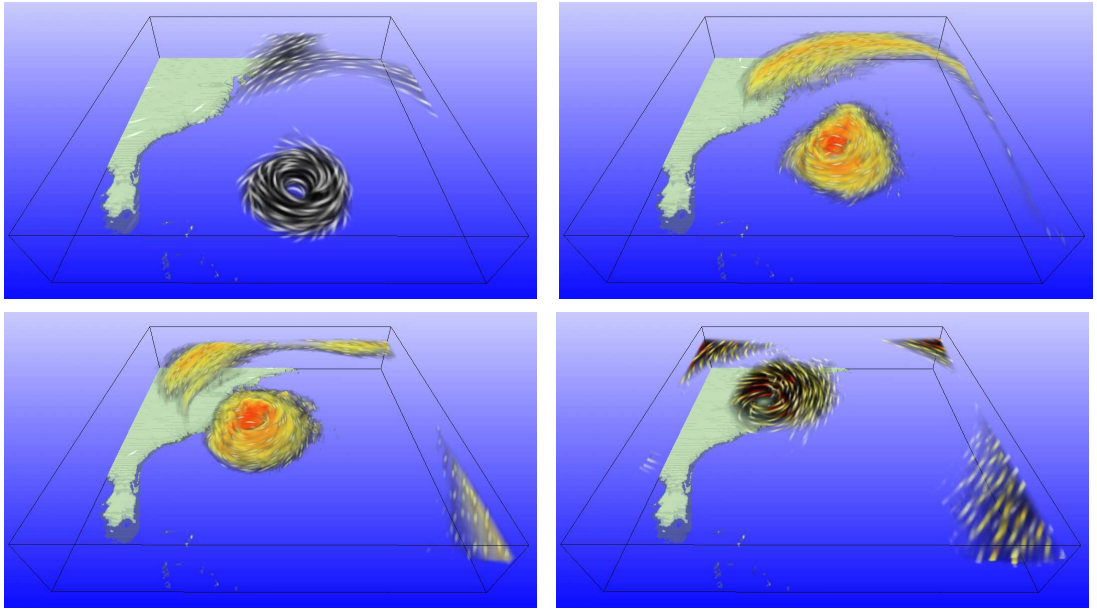


A = Velocity
B = Velocity magnitude

$$[\text{color,opacity}] = \mathit{merge}\big((A\ \mathit{mask}\ \tilde{B}),\ ([B, A]\ \mathit{mask}\ \tilde{B})\big)$$

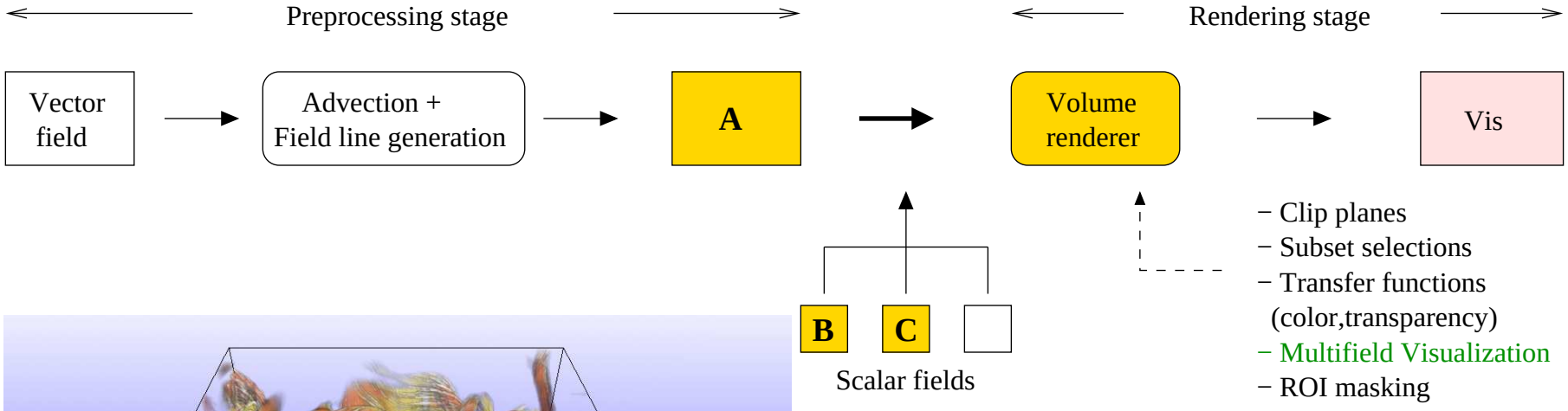
Multifield volume rendering

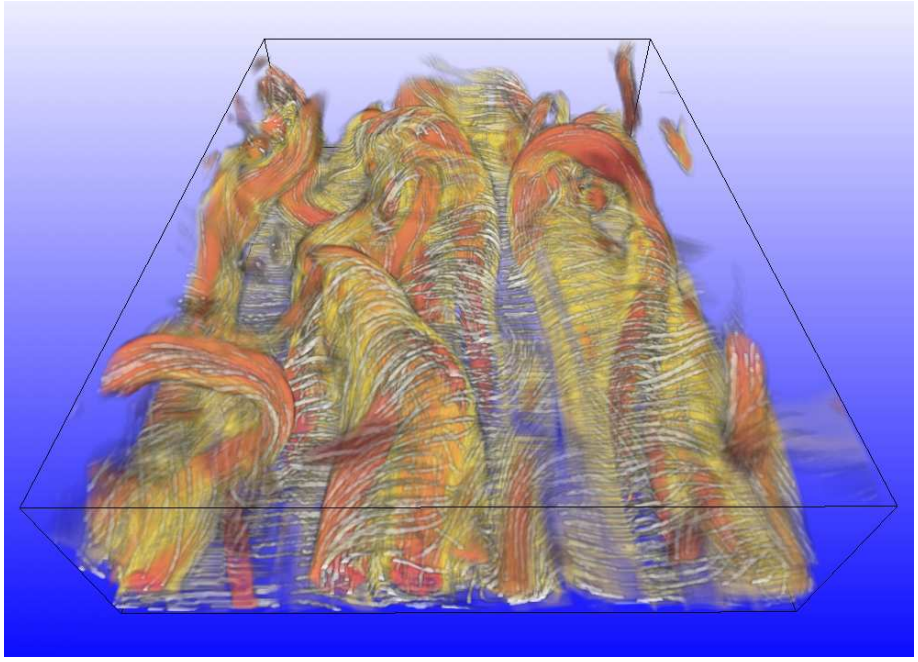




A = Velocity
B = Velocity magnitude

Multifield volume rendering

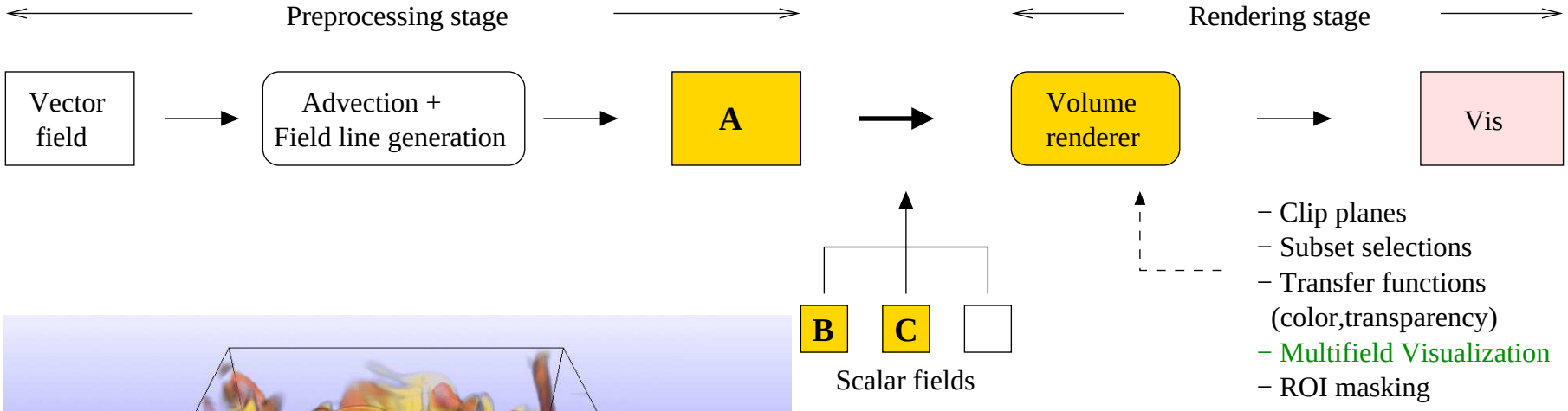


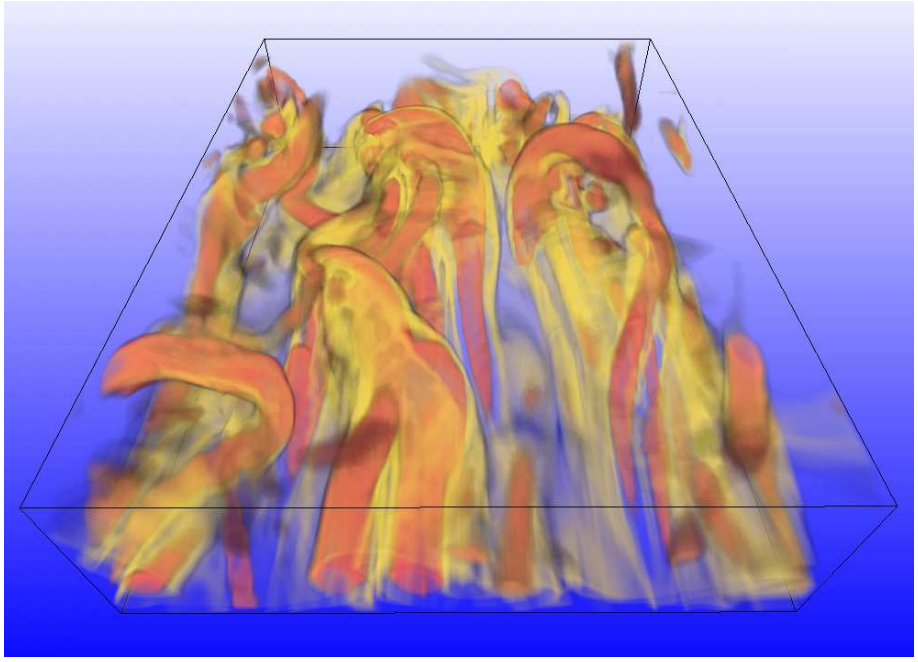


A = Vorticity (gray)
B = Enstrophy (yellow)
$C = \lambda_2$ (red)

$$\text{[color,opacity]} = \mathit{merge}\big((A\ \mathit{mask}\ \tilde{B}), B, C\big)$$

Multifield volume rendering





$A =$ Vorticity (gray)
$B =$ Enstrophy (yellow)
$C = \lambda_2$ (red)

$$[\text{color,opacity}] = \mathit{merge}(B, C)$$

Outline

- Basic data representation
- Techniques for visualizing
  - Scalars
  - Vector
  - Tensors

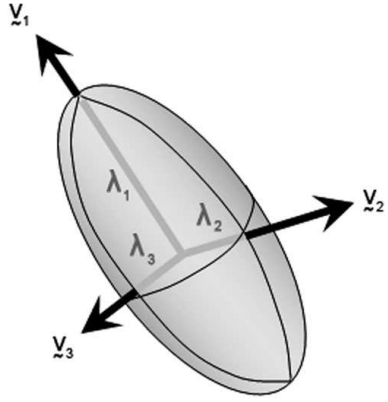
Tensors

- A tensor (in 3D) has 9 components and can be witten as

$$T = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix}.$$

- In general, eigenvalues λ and eigenvectors $\mathbf{x}$ can be found as a solution to the eigen-equation $T\mathbf{x_i} = \lambda_i \mathbf{x_i}, \quad i = 1, 2, 3.$
- For symmetric tensors its eigenvalues are always real number and the eigenvector are orthogonal. Then T can be rewritten as

$$T = (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix} (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})^T.$$



Some tensors

- Strain and rotation tensors

  $$J_{ij} = \frac{\partial u_i}{\partial x_j} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) = S_{ij} + \Omega_{ij},$$

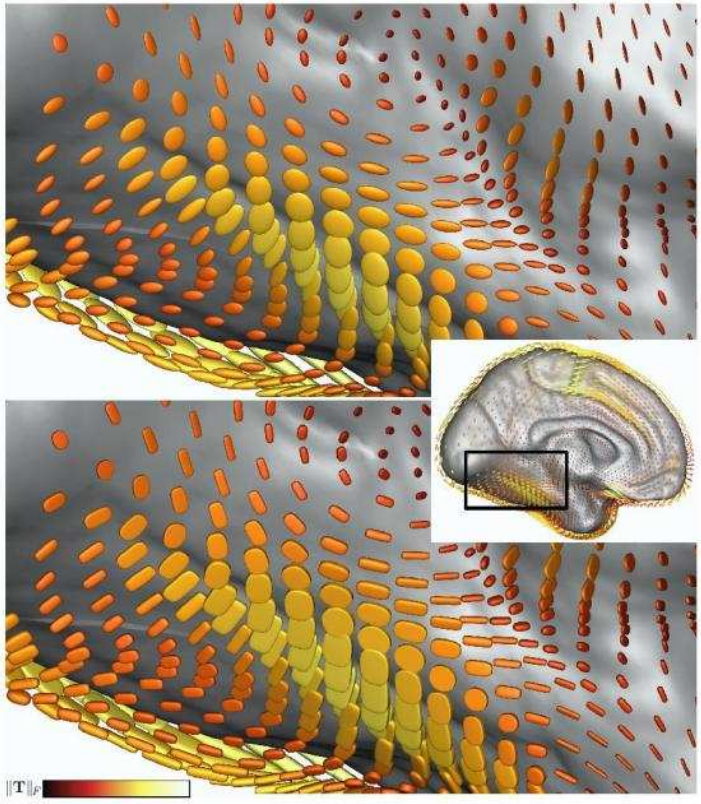
- Diffusion tensors (Used for studying the structure in the brain such as nerve root pathways).

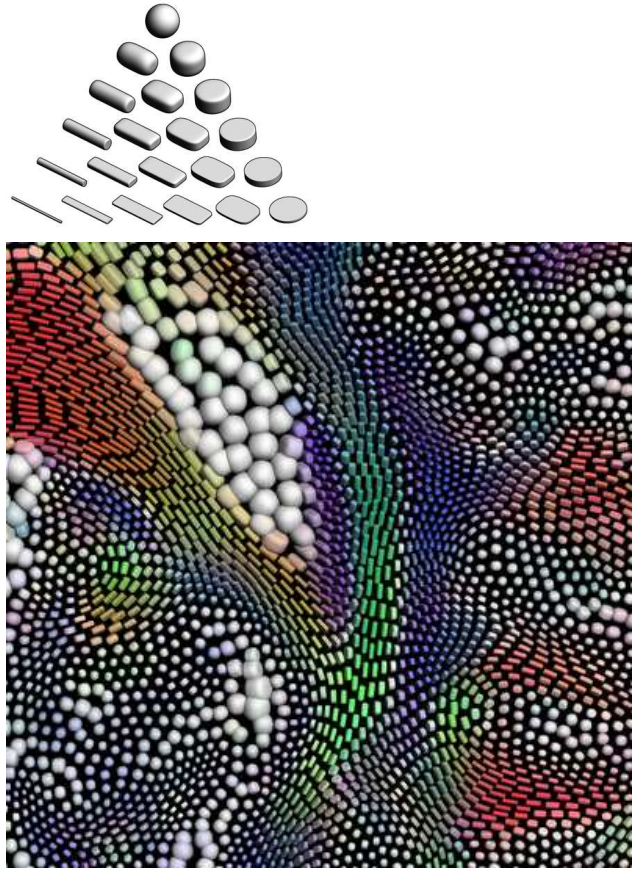
Techniques for visualizing tensors

- Tensor glyphs
- Hyper field lines
- HyperLIC
- Topological methods.

Tensor glyphs

- Insert glyphs locally in the data domain.
- Exhibit the same limitations as for glyph-based vector field visualization.





Hyper field lines

- Integration of curves along the most dominant eigenvector direction.
- Hyper field lines are rendered as tubes with elliptic cross-section with radius proportional to the 2^{nd} and 3^{nd} eigenvalue.