Tensor Field Visualization II

Goal: have a basic understanding of the geometric-based, texture-based visualization techniques, and simple topology-based method for second-order tensor fields.

GEOMETRIC-BASED METHOD

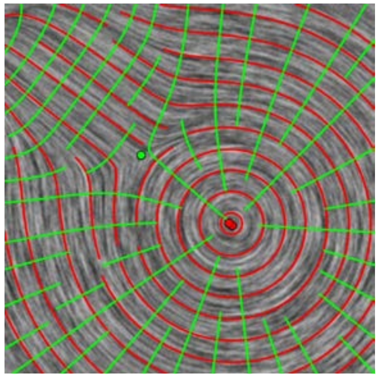
--TRACING HYPERSTREAMLINES

Hyperstreamlines

- Let **T**(**x**) be a (2nd order) **symmetric** tensor field
  - real eigenvalues, orthogonal eigenvectors

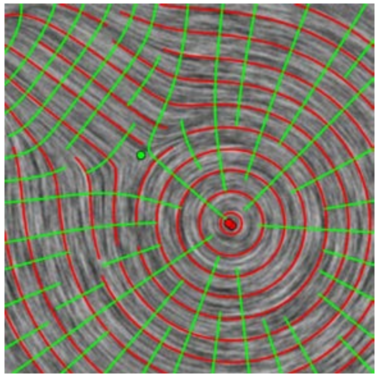
- **Hyperstreamline**: by integrating along one of the eigenvector fields

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{v}(\mathbf{s}_i) \cdot dt$$



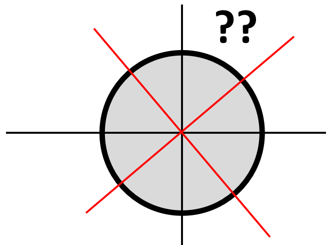
Hyperstreamlines

- Let **T**(**x**) be a (2nd order) **symmetric** tensor field
  - real eigenvalues, orthogonal eigenvectors



- **Hyperstreamline**: by integrating along one of the eigenvector fields $\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{v}(\mathbf{s}_i) \cdot dt$

- **Important: Eigenvector fields are not vector fields!**
  - eigenvectors have ***no magnitude*** and *no direction* (are bidirectional)
  - the choice of the eigenvector can be made consistently *as long as eigenvalues are all different*
  - Hyperstreamlines can intersect only at points where two or more eigenvalues are equal, i.e., **degenerate points**.



Compute One Hyperstreamline

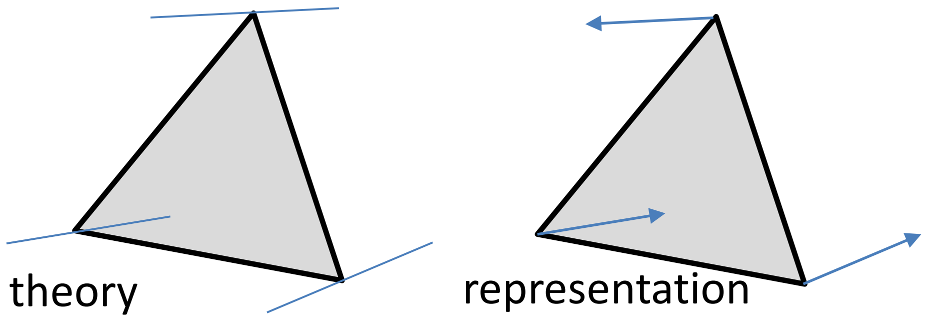
- Choose integrator:
  - Euler
  - Runge-Kutta (2nd, 4th , 4/5)
  - Others
- Choose step size (can be adaptive)
- Provide seed point position and **determine starting direction**
- Advance the front
- Termination conditions

Compute One Hyperstreamline

- Choose integrator:
  - Euler
  - Runge-Kutta (2nd, 4th , 4/5)
  - Others
- Choose step size (can be adaptive)
- Provide seed point position and **determine starting direction**
- <u>**Advance the front**</u>
- Termination conditions

- **Note that the direction/angle ambiguity**. This is because the computation of the eigenvector at each sample point (i.e., a vertex/grid point of the mesh) is independent of each other. Therefore, **inconsistent directions** (a direction of an eigenvector needs to be chosen) may be chosen to store at neighboring vertices.

Any solution?



Compute One Hyperstreamline

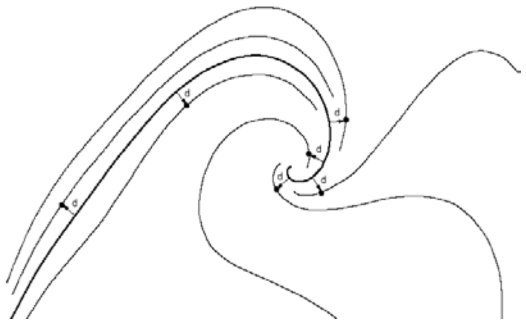
- Choose integrator:
  - Euler
  - Runge-Kutta (2nd, 4th , 4/5)
  - Others
- Choose step size (can be adaptive)
- Provide seed point position and **determine starting direction**
- **Advance the front**
- Termination conditions

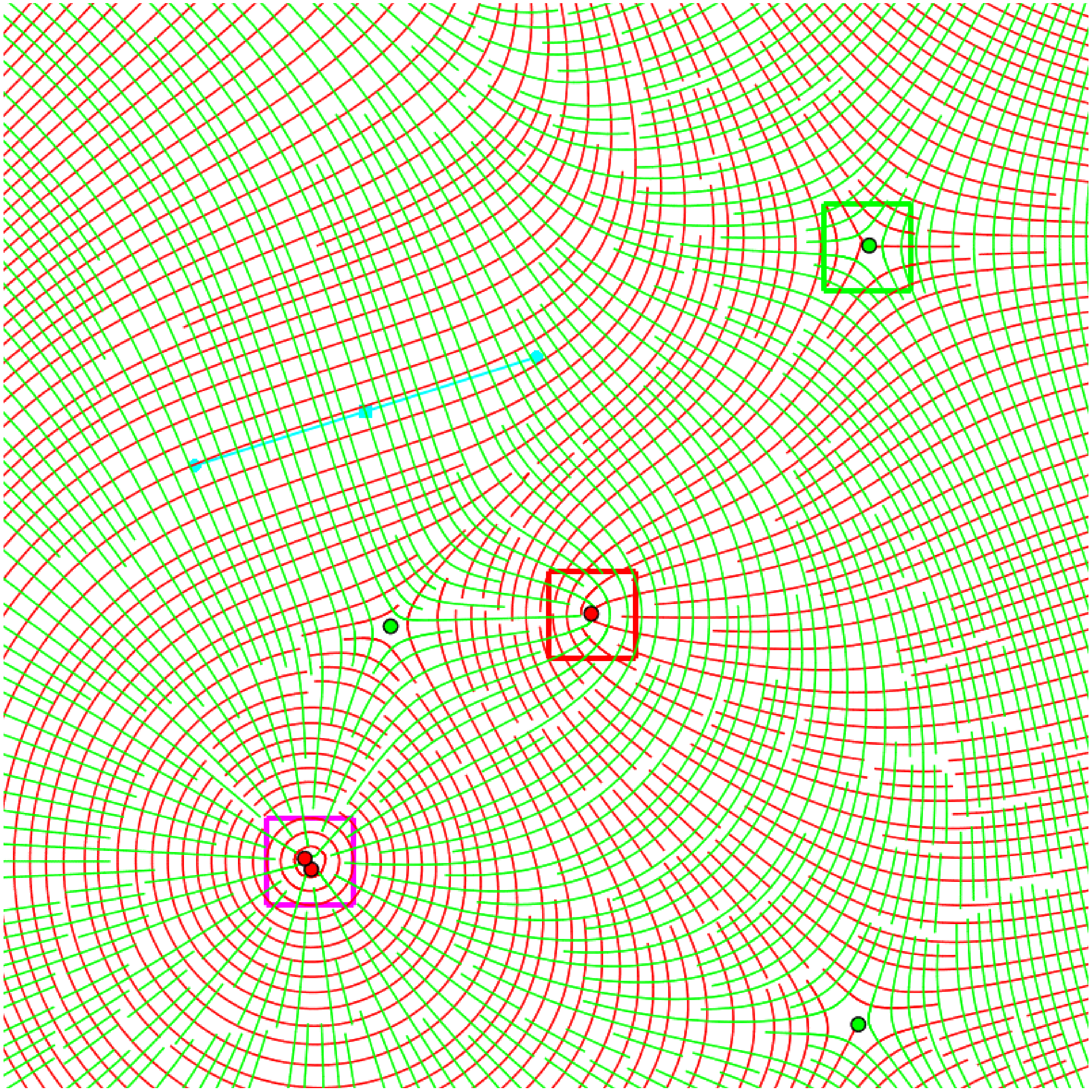
- **Note that the direction/angle ambiguity**. This is because the computation of the eigenvector at each sample point (i.e., a vertex/grid point of the mesh) is independent of each other. Therefore, **inconsistent directions** (a direction of an eigenvector needs to be chosen) may be chosen to store at neighboring vertices.
  - **Additional step to remove direction/angle ambiguity**. A dot product between the current advancing direction and the eigenvector direction at current position is performed. A positive value indicates the consistent direction; otherwise, the inverse direction should be used!

Evenly-Spaced Placement

- Input:
  - d_{sep} … start distance
  - d_{test} … minimum distance

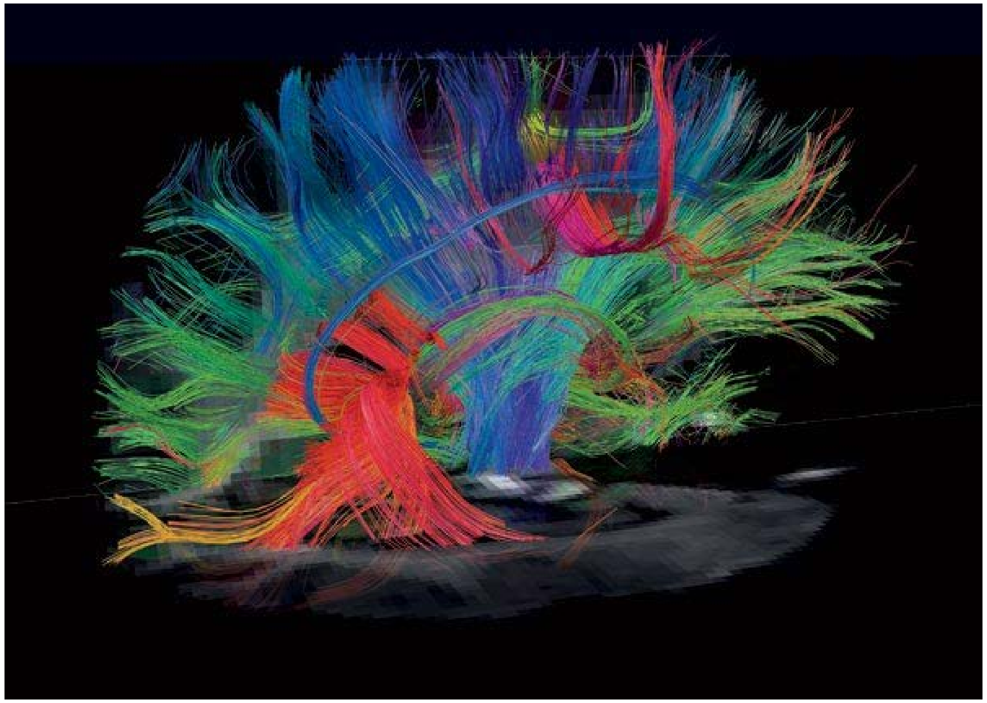
- Compute an initial *hyperstreamline* from a random seed point, put to queue
- current *hyperstreamline* = initial *hyperstreamline*
- *WHILE* not finished DO:
  - TRY: get new seed point which is d_{sep} away from current *hyperstreamline*
  - *IF* successful *THEN*
    - compute new *hyperstreamline* until distance d_{test} is reached (or other…) AND put it to queue
  - *ELSE IF* no more *hyperstreamline* in queue *THEN*
    - exit loop
  - *ELSE* next *hyperstreamline* in queue becomes current *hyperstreamline*

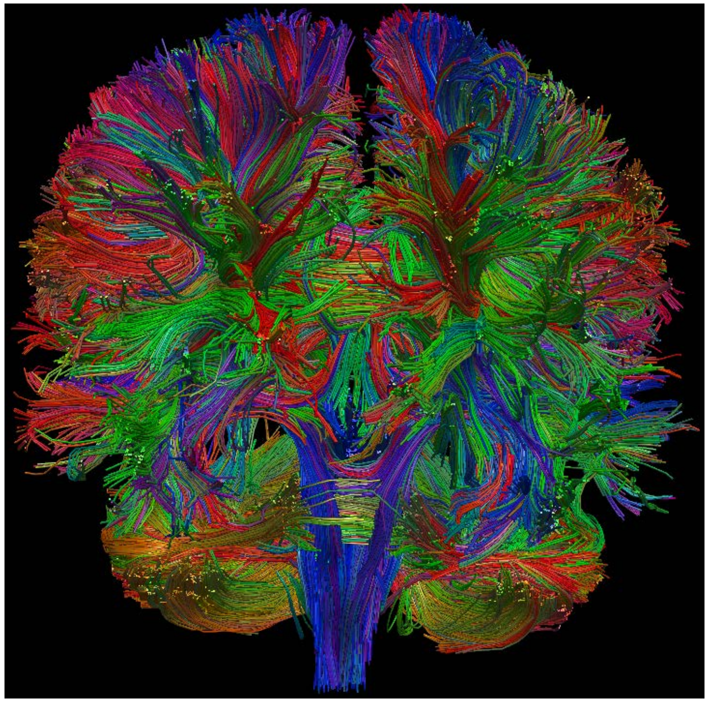




Red – major
Green – minor

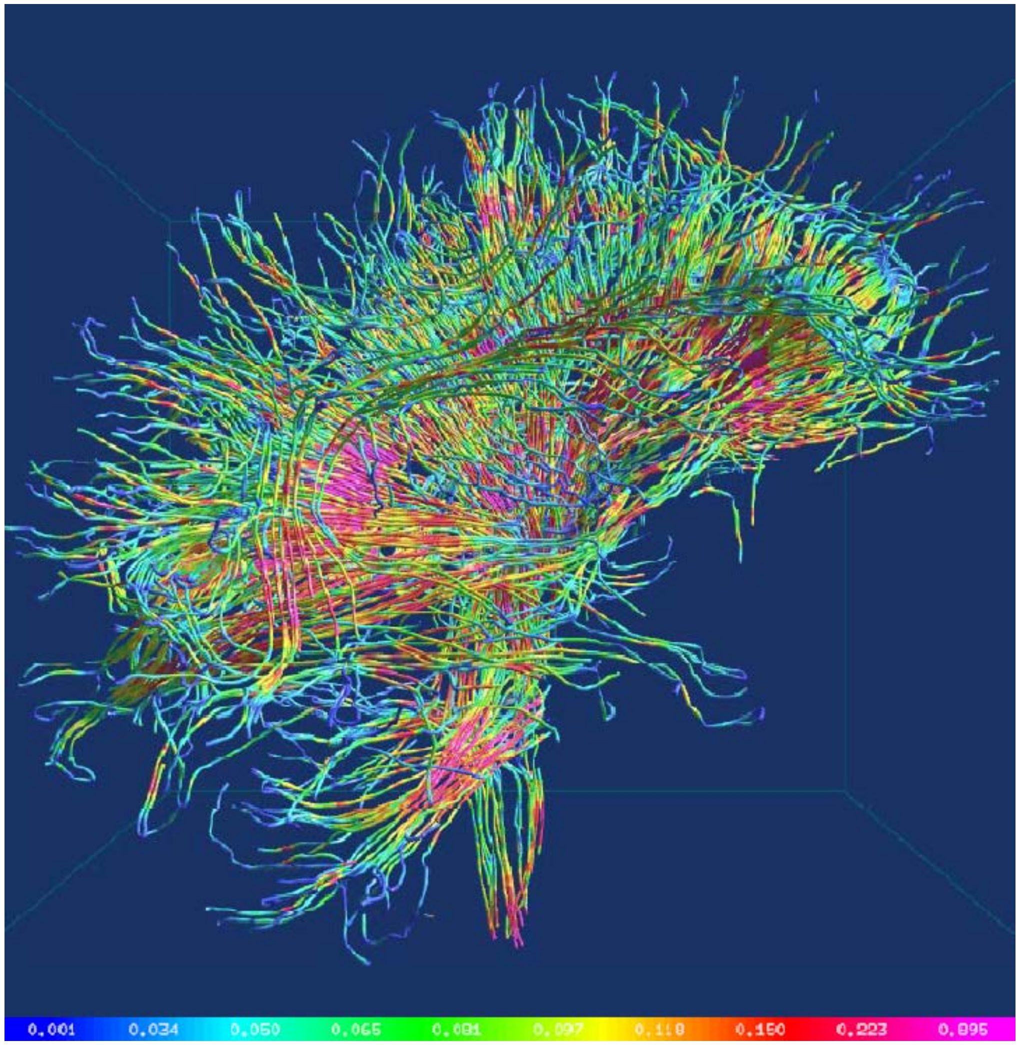
Applications of Hyper-Streamline Placement





Widely used in diffusion tensor imaging tractography

Applications of Hyper-Streamline Placement

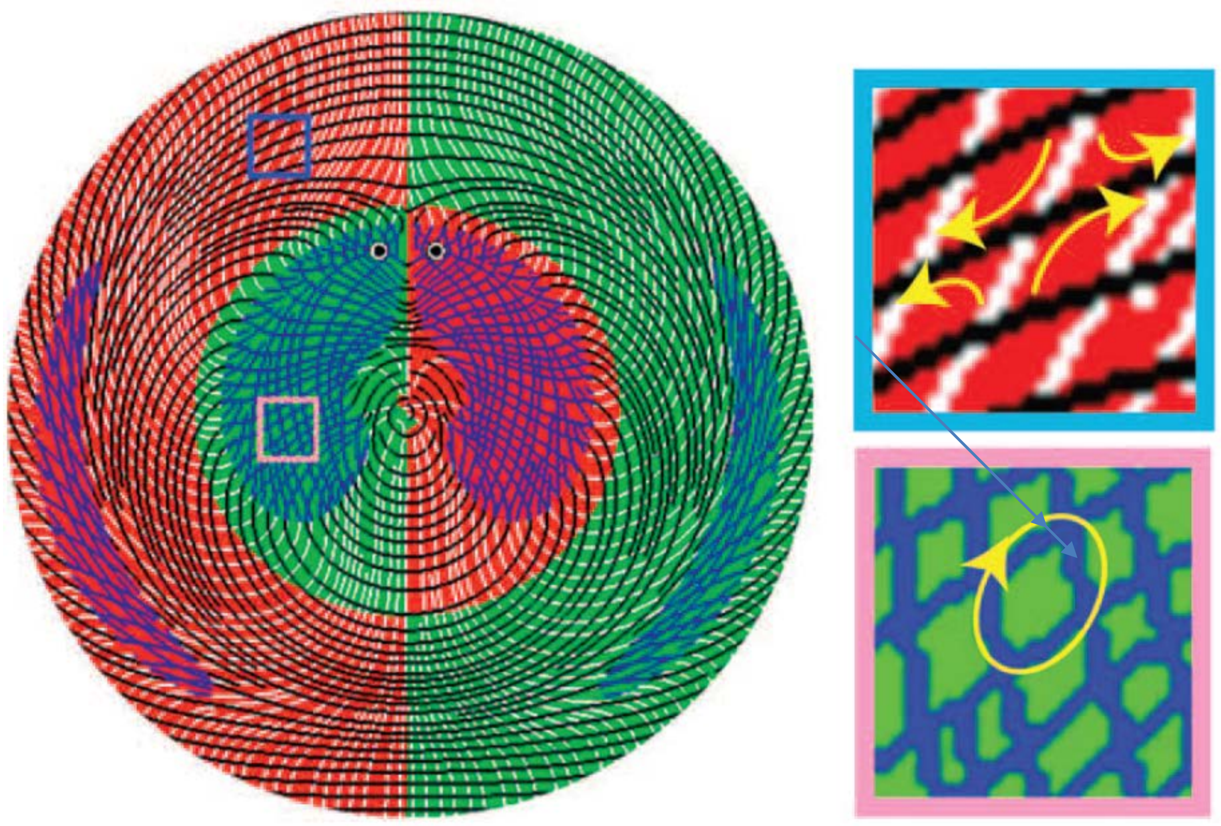


Hyperstreamlines rendered as tubes **with elliptic cross section**, radii proportional to 2nd and 3rd eigenvalue

[Shen and Pang 2004]

Widely used in diffusion tensor imaging tractography

Applications of Hyper-Streamline Placement

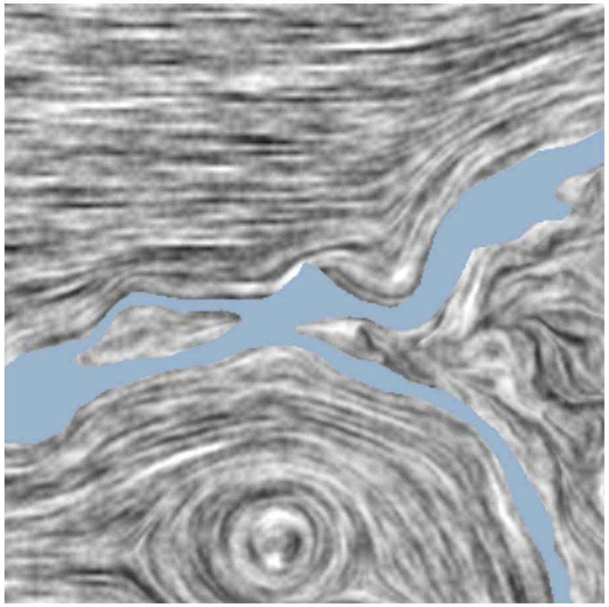
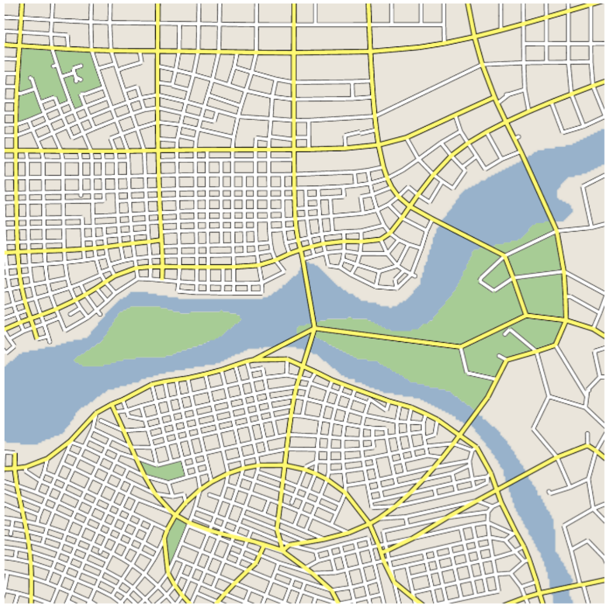
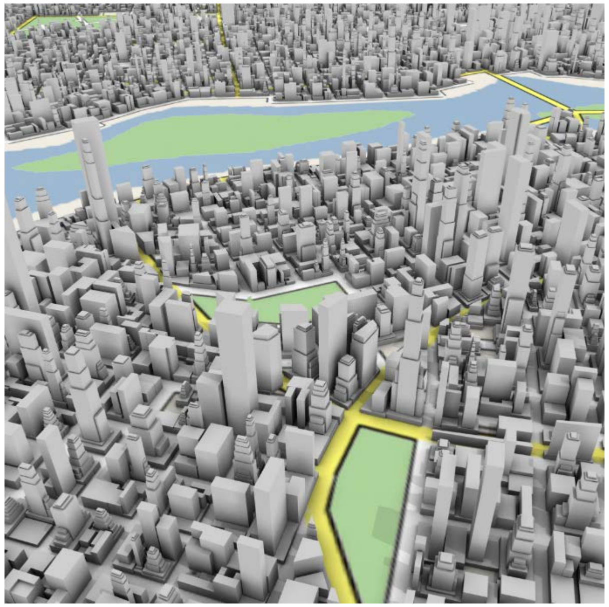


Hyperstreamlines can also be used to convey some physical behaviors in the tensor. For instance, in flow analysis, the hyperstreamlines computed based on the eigen analysis of the Jacobian of the flow field can convey stretching and rotational flow deformation

[Zhang et al. TVCG 2009]

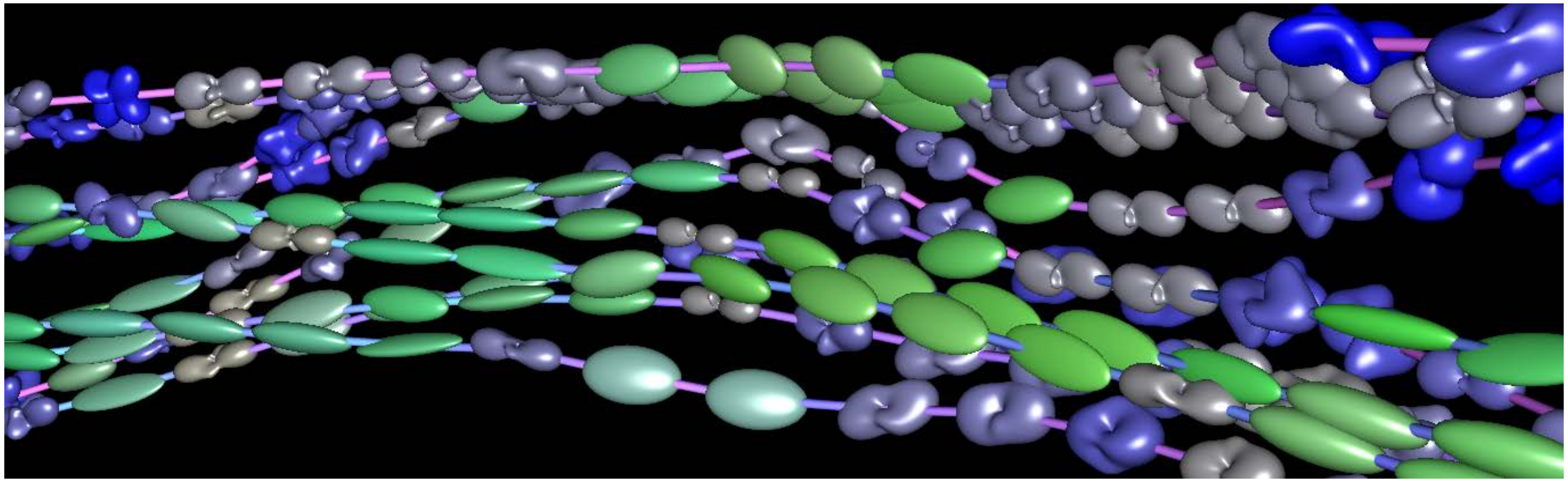
Applications of Hyper-Streamline Placement

- Street network modeling [Chen et al. Siggraph 2008]

According to different applications, the termination conditions may be different

Hyperstreamlines + Glyphs



[Prckovska et al. 2010]

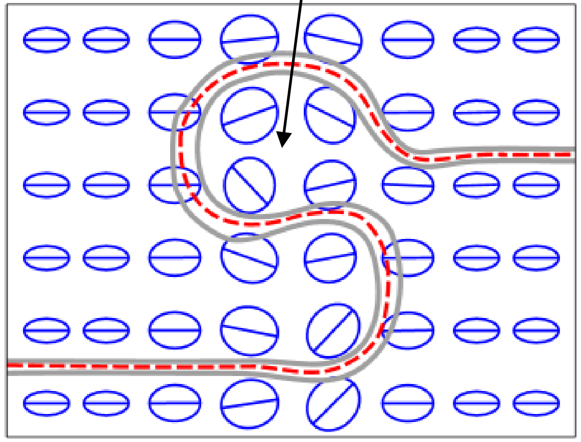
Hybrid visualization: **hyperstreamlines + glyphs**

Good for some non-symmetric tensor visualization where the rotational components can be encoded by the glyphs

Problem of Hyperstreamlines

- Ambiguity in *(nearly) isotropic* regions:
  - *Partial volume effect*, especially in low resolution images (MR images)
  - Noise in data

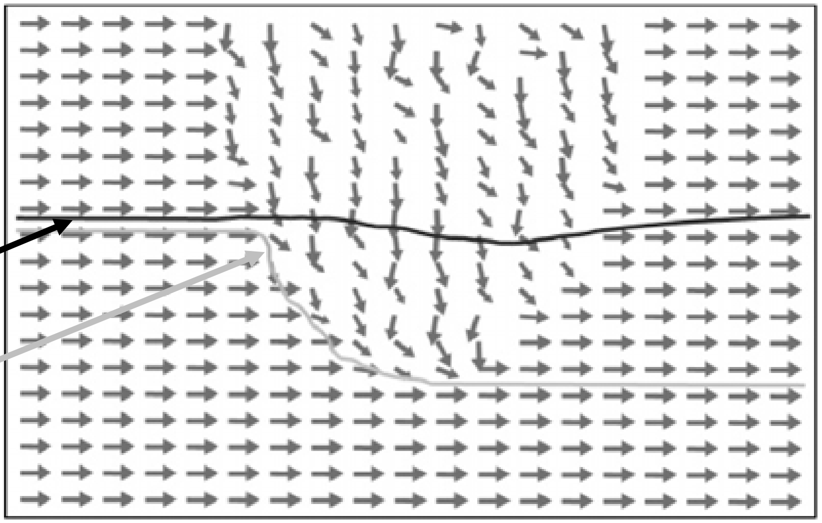
eigen-vectors are not well-defined or perturbed



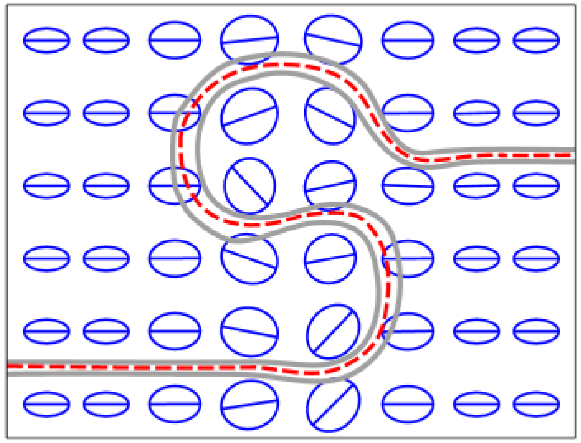
Problem of Hyperstreamlines

- Ambiguity in *(nearly) isotropic* regions:
  - *Partial volume effect*, especially in low resolution images (MR images)
  - Noise in data
- **Solution**: tensorlines

  [Weinstein, Kindlmann 1999]



Tensorline
Hyperstreamline
Arrows: major eigenvector



Tensorlines

- Advection vector
- Stabilization of propagation by considering the following components
  - ✓ Input velocity vector (previous direction)
  - ✓ Output velocity vector (after application of tensor operation, i.e. transformation by tensor)
  - ✓ Vector along major eigenvector

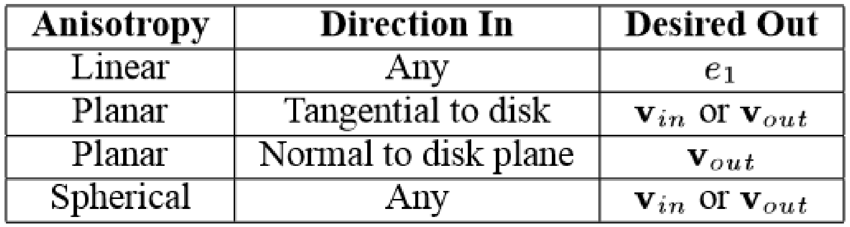
[Weinstein et al.1999]

Tensorlines

- Advection vector
- Stabilization of propagation by considering the following components
  - ✓ Input velocity vector (previous direction)
  - ✓ Output velocity vector (after application of tensor operation, i.e. transformation by tensor)
  - ✓ Vector along major eigenvector
- Weighted sum of the three components depends on anisotropy at specific position:
  - ✓ Linear anisotropy: only along major eigenvector
  - ✓ Other cases: input or output vector

$$\mathbf{v}_{prop} = c_l \mathbf{e}_1 + (1 - c_l)\big((1 - \omega_{punct})\mathbf{v}_{in} + \omega_{punct}\mathbf{v}_{out}\big) \longleftarrow$$ **Two linear interpolation!**

ω_{punct} is a user-controlled parameter



Anisotropy	Direction In	Desired Out
Linear	Any	e_1
Planar	Tangential to disk	$\mathbf{v}_{in}$ or $\mathbf{v}_{out}$
Planar	Normal to disk plane	$\mathbf{v}_{out}$
Spherical	Any	$\mathbf{v}_{in}$ or $\mathbf{v}_{out}$

3D case

[Weinstein et al.1999]

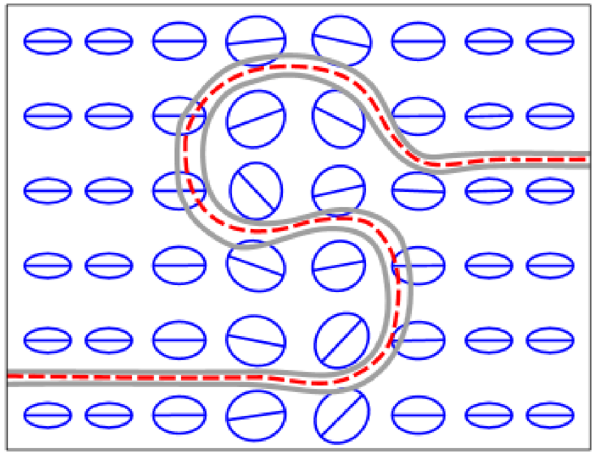
Tensorlines

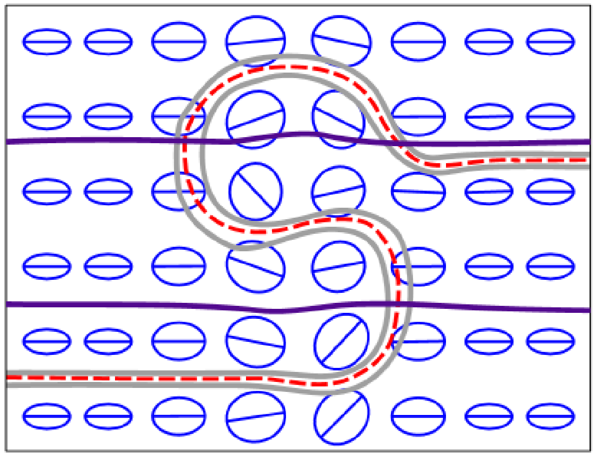
- Advection vector
- Stabilization of propagation by considering the following components
  - ✓ Input velocity vector (previous direction)
  - ✓ Output velocity vector (after application of tensor operation, i.e. transformation by tensor)
  - ✓ Vector along major eigenvector
- Weighted sum of the three components depends on anisotropy at specific position:
  - ✓ Linear anisotropy: only along major eigenvector
  - ✓ Other cases: input or output vector

$$\mathbf{v}_{prop} = c_l \mathbf{e}_1 + (1 - c_l)\big((1 - \omega_{punct})\mathbf{v}_{in} + \omega_{punct}\mathbf{v}_{out}\big)$$

ω_{punct} is a user-controlled parameter

Anisotropy	Direction In	Desired Out
Linear	Any	e_1
Planar	Tangential to disk	$\mathbf{v}_{in}$ or $\mathbf{v}_{out}$
Planar	Normal to disk plane	$\mathbf{v}_{out}$
Spherical	Any	$\mathbf{v}_{in}$ or $\mathbf{v}_{out}$

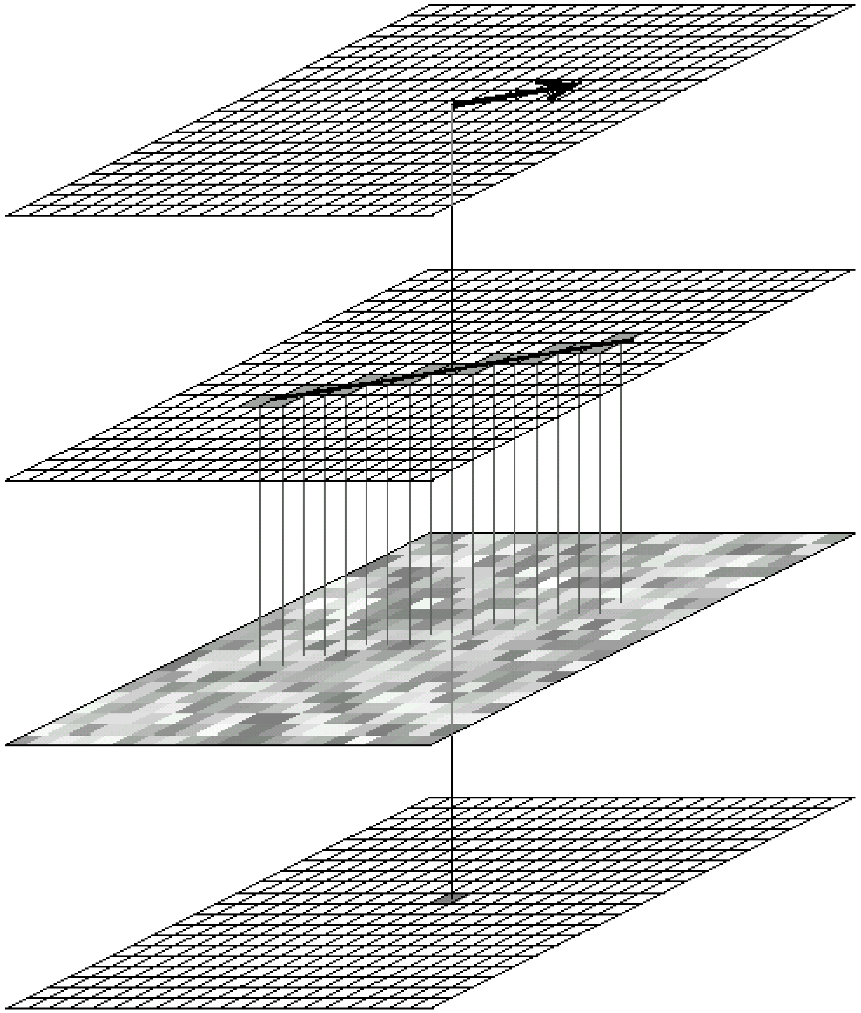




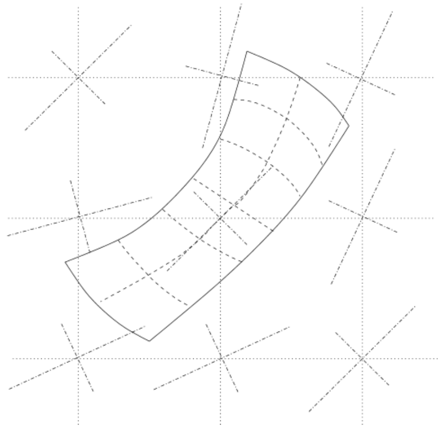
[Weinstein et al.1999]

TEXTURE-BASED METHOD

LIC

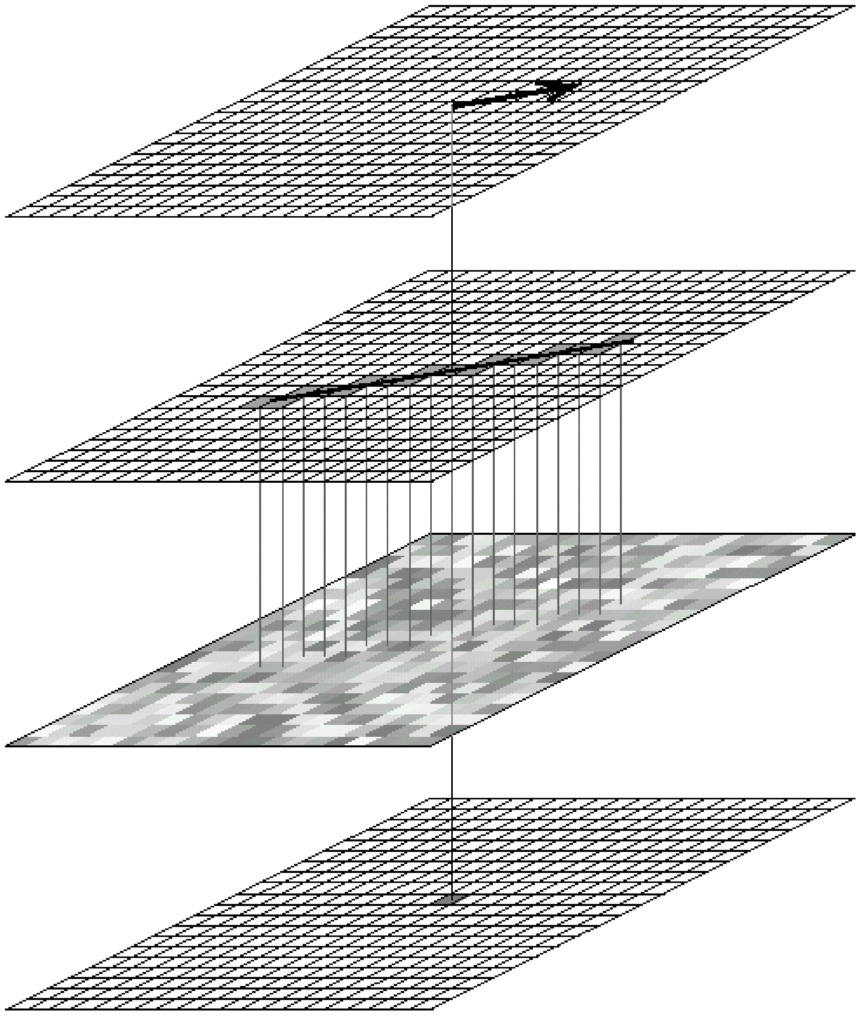

The LIC pipeline

How to extend it to symmetric tensor field?

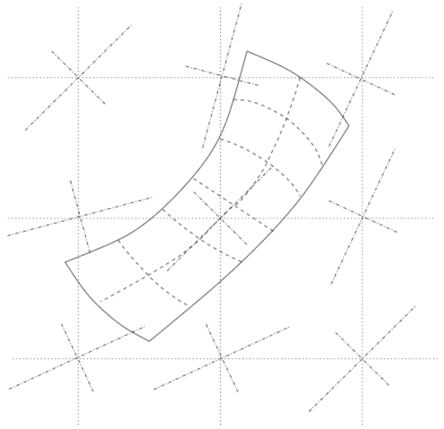


HyperLIC [Zheng and Pang]

Instead of using a 1D kernel along the streamline, HyperLIC uses a 2D kernel



The LIC pipeline

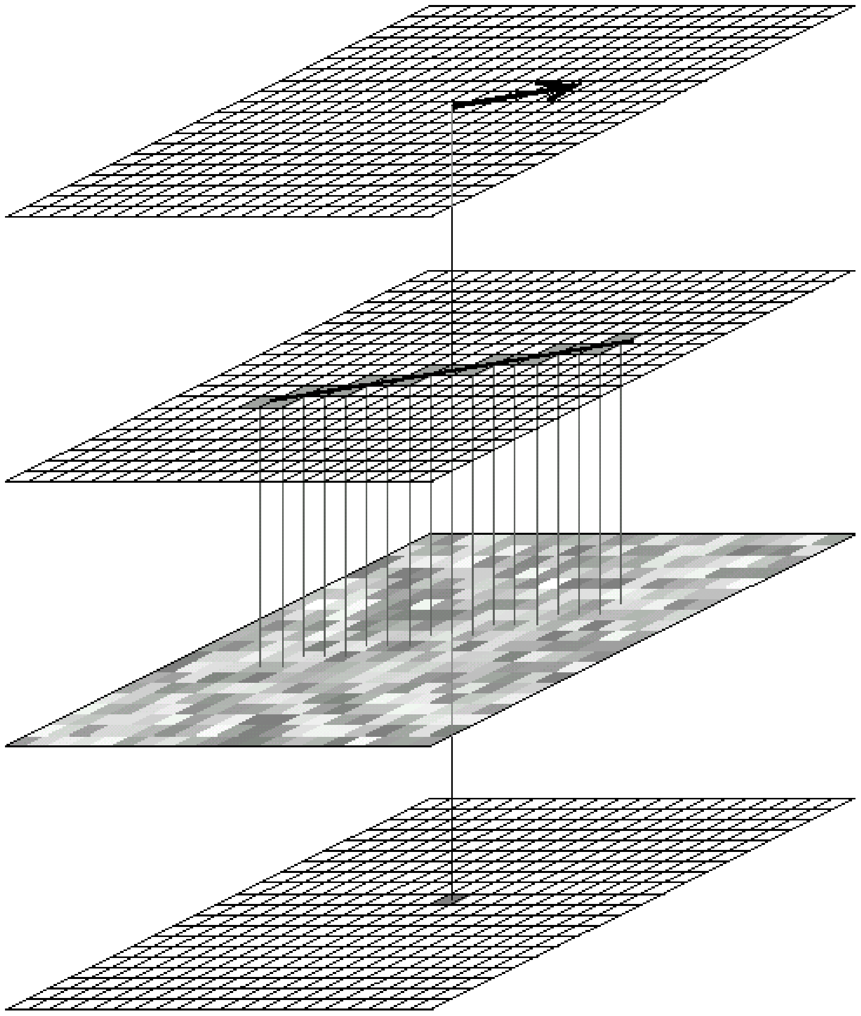


HyperLIC [Zheng and Pang]

Instead of using a 1D kernel along the streamline, HyperLIC uses a 2D kernel

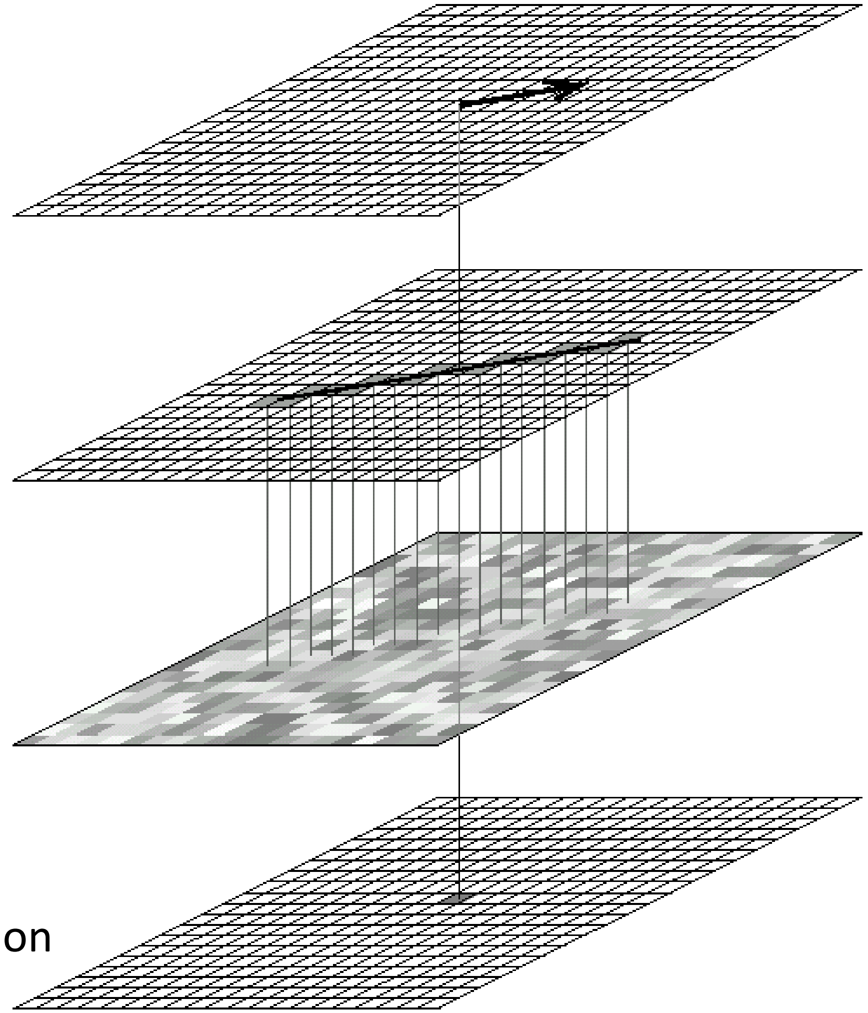
$$
\begin{aligned}
I_o(P) &= \frac{\sum_{i=-N}^{N} \sum_{j=-N}^{N} k(i,j) I_n(P_{i,j})}{\sum_{i=-N}^{N} \sum_{j=-N}^{N} k(i,j)} \\
P_{i,j} &= P_{i-1,j} + \lambda_1(P_{i-1,j}) e_1(P_{i-1,j}) \Delta t \\
P_{0,j} &= P_{0,j-1} + \lambda_2(P_{0,j-1}) e_2(P_{0,j-1}) \Delta t \\
P_{0,0} &= P
\end{aligned}
$$

I_n is the input noise and I_o is the output
$\lambda_n(P), e_n(P), k(i,j), n = 1,2$ are the n^{th} eigenvalues, eigenvectors, and the weight function at point P. Δt is the integration step.



The LIC pipeline

HyperLIC [Zheng and Pang]



The LIC pipeline

Decompose the computation

If we define

$$
\begin{aligned}
I_1(P) &= \frac{\sum_{j=-N}^{N} k_2(j) I_n(P_j)}{\sum_{j=-N}^{N} k_2(j)} \\
P_j &= P_{j-1} + \lambda_2(P_{j-1}) e_2(P_{j-1}) \Delta t \\
P_0 &= P
\end{aligned}
$$

First, advect along the **minor** eigen vector direction corresponding to the minimal eigen value.

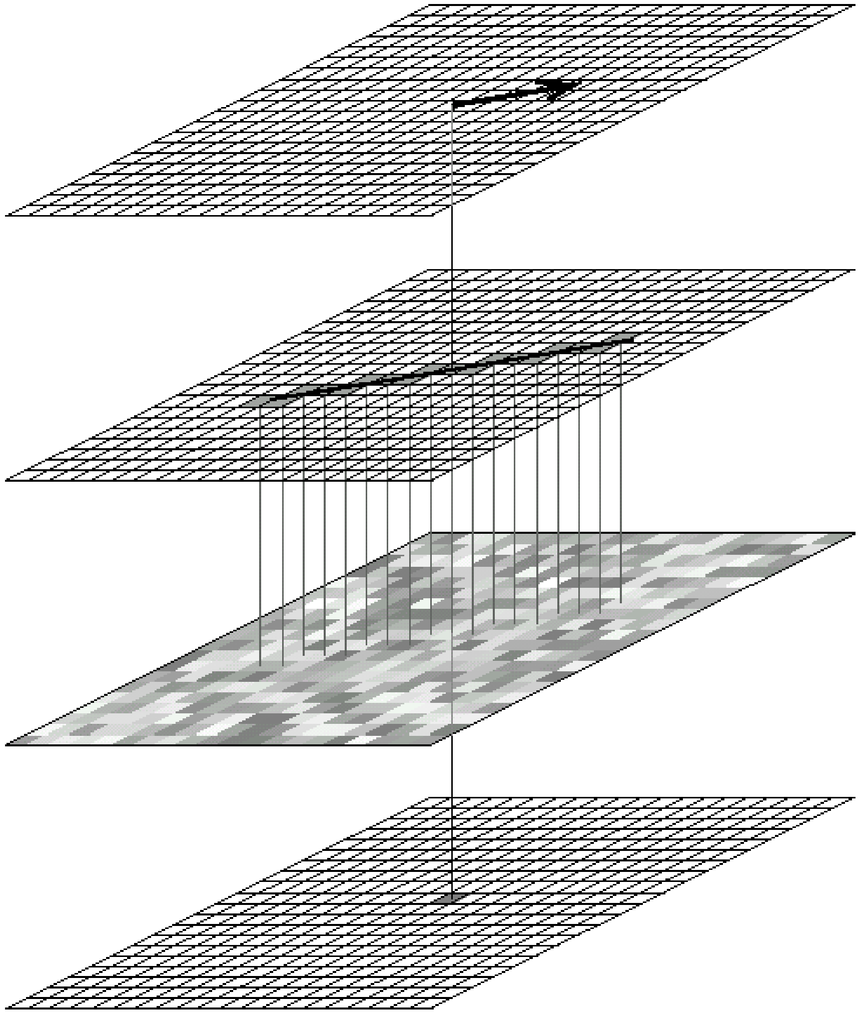
HyperLIC [Zheng and Pang]

Decompose the computation

Then

$$
\begin{aligned}
I_o(P) &= \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \\
P_i &= P_{i-1} + \lambda_1(P_{i-1}) e_1(P_{i-1}) \Delta t \\
P_0 &= P
\end{aligned}
$$

Second, use I_1 as input texture, then advect along the **major** eigen vector direction.


The LIC pipeline

HyperLIC [Zheng and Pang]

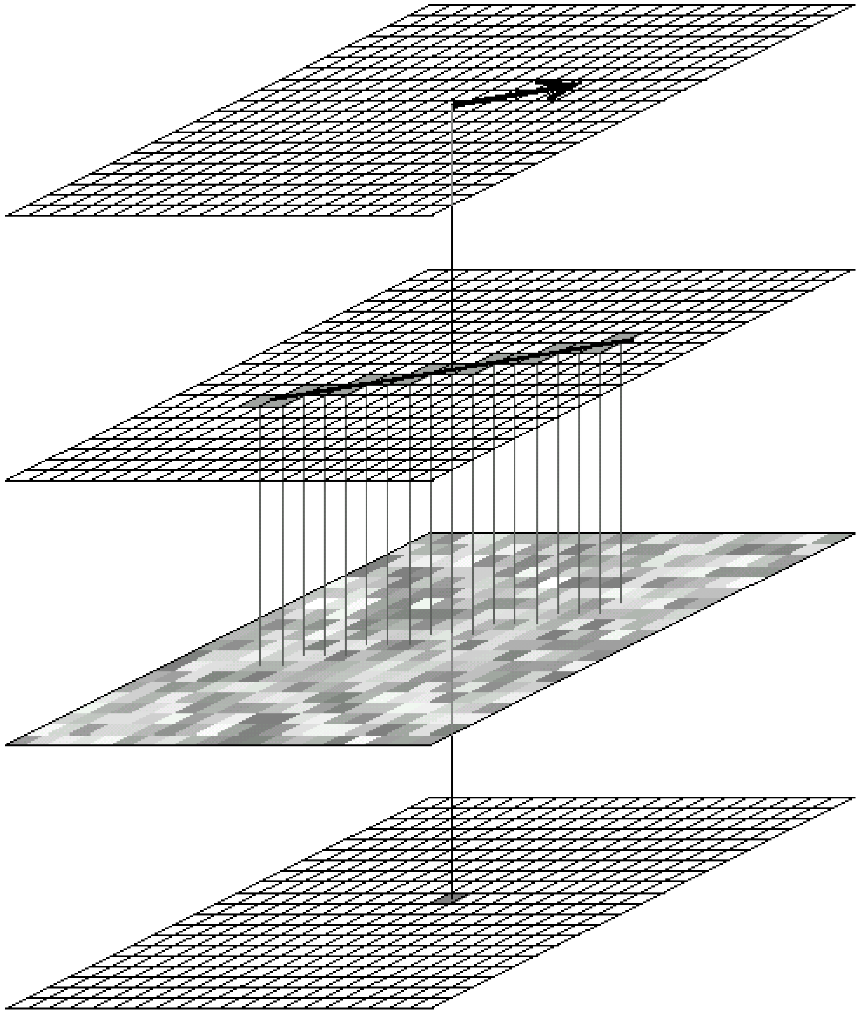
Decompose the computation

Then

$$\begin{aligned} I_o(P) &= \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \\ P_i &= P_{i-1} + \lambda_1(P_{i-1}) e_1(P_{i-1}) \Delta t \\ P_0 &= P \end{aligned}$$

This is a two-pass process

$I_1$and I_0 are the output images of the ***<u>un-normalized</u>*** LIC on $\lambda_2 e_2$ and $\lambda_1 e_1$ vector fields with input images I_n and I_1, respectively.



The LIC pipeline

HyperLIC [Zheng and Pang]

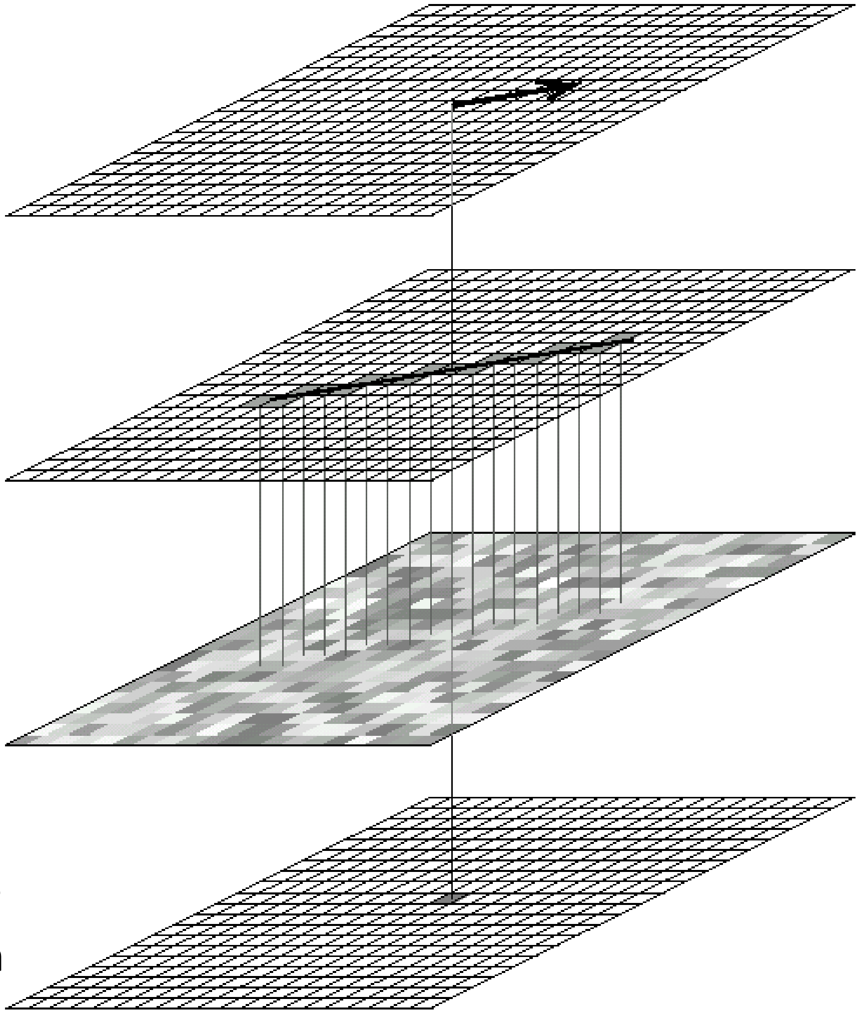
Decompose the computation

Then

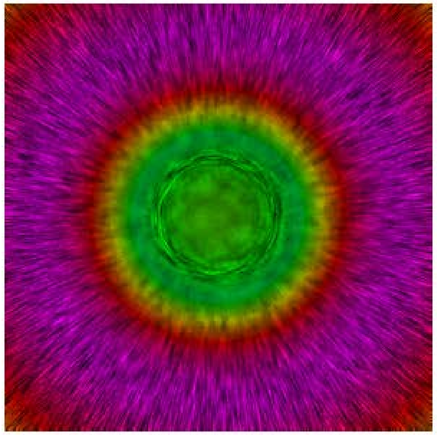
$$\begin{aligned} I_o(P) &= \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \\ P_i &= P_{i-1} + \lambda_1(P_{i-1}) e_1(P_{i-1}) \Delta t \\ P_0 &= P \end{aligned}$$

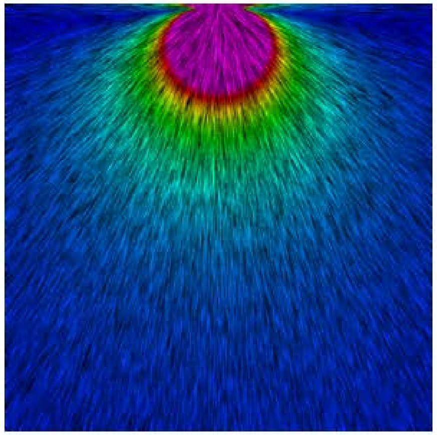
This is a two-pass process

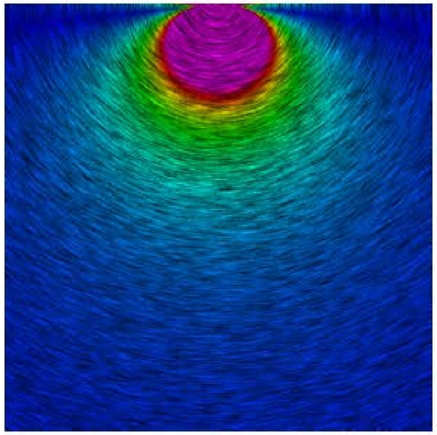
Theoretically, the order in which the eigenvector fields are processed will affect the final image. In practice, the differences are not noticeable

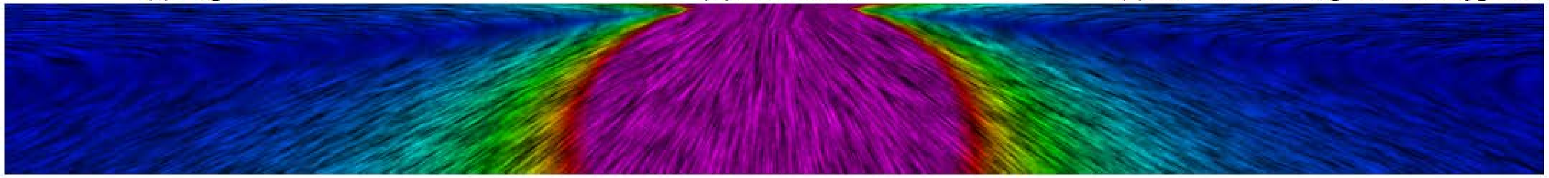

The LIC pipeline

Applications


(a) Top view

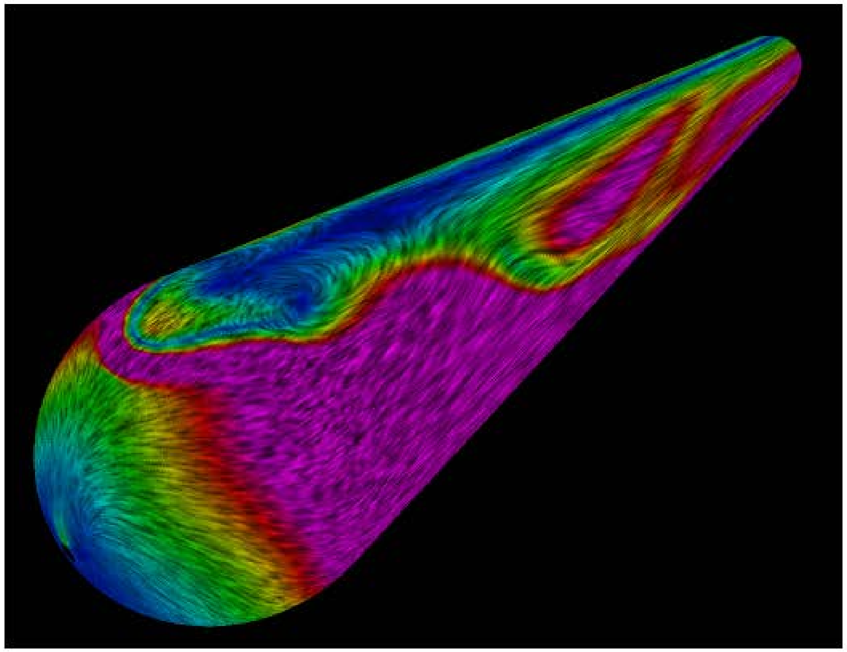

(b) Side view

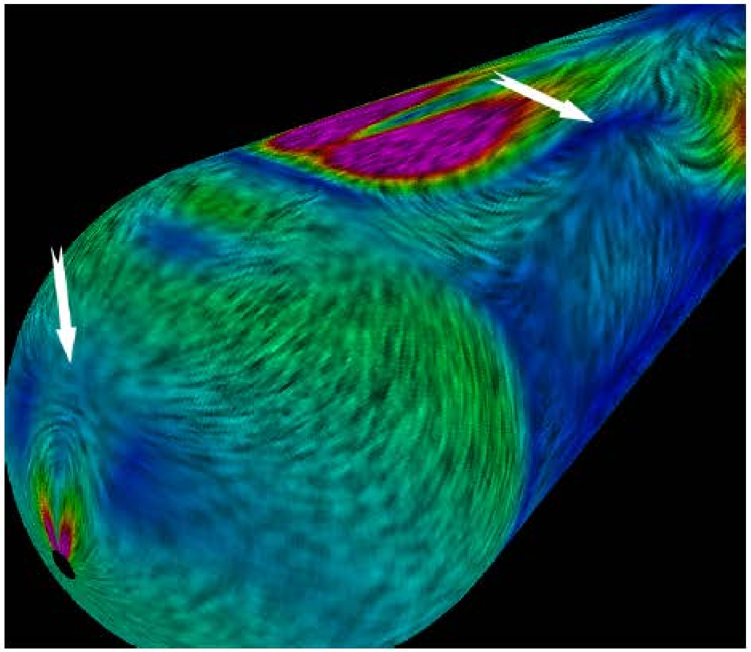

(c) Side view using Inverse HyperLIC


(d) Zoomed view of the top portion of (b)

A 2D slice from single point load stress tensors. It is taken from the middle of the volume and viewed from the point load direction. It is mostly composed of components from medium or minor eigenvectors. We see that the center of this slice is quite isotropic. Around the center is a ring formed by lines, which means tensors are highly anisotropic. It is the boundary where the minor eigenvalues are zero.

Applications

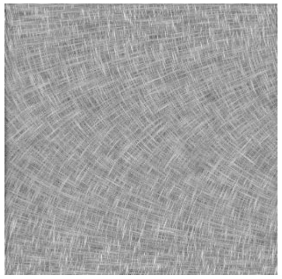

(a) Inner layer


(b) Middle layer

Flow past a cylinder with hemispherical cap. HyperLIC of two different computational layers of the strain rate tensor. Arrows point to locations of degenerate wedge points

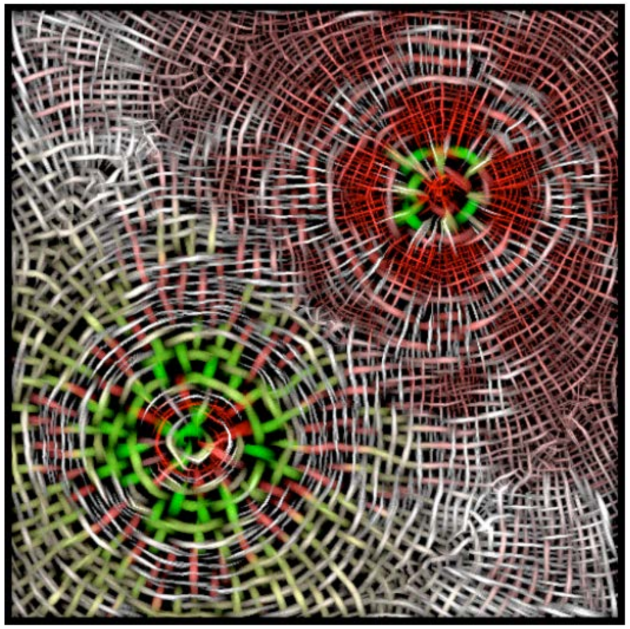
A Simplified HyperLIC

- Compute **two LIC images** along the major and minor eigen-vector fields, **separately**.
- These two computations are independent of each other, and thus, can be parallelized.
- Note, the direction/angle ambiguity needs to be properly handled as in the hyperstreamline tracing.
- NOTE, this is only meaningful for symmetric positive definite tensors

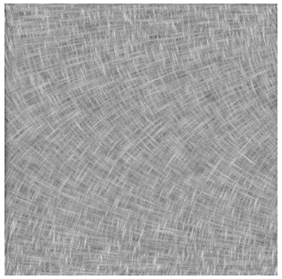


A Simplified HyperLIC

- Compute **two LIC images** along the major and minor eigen-vector fields, **separately**.
- These two computations are independent of each other, and thus, can be parallelized.
- Note, the direction/angle ambiguity needs to be properly handled as in the hyperstreamline tracing.
- NOTE, this is only meaningful for symmetric positive definite tensors



The image represents a xz-plane slice of a two-force dataset. The left circle corresponds to the pushing and the right to the pulling force. The fluctuation of the color is a result of the low resolution of the simulation [Hotz et al. 2004]

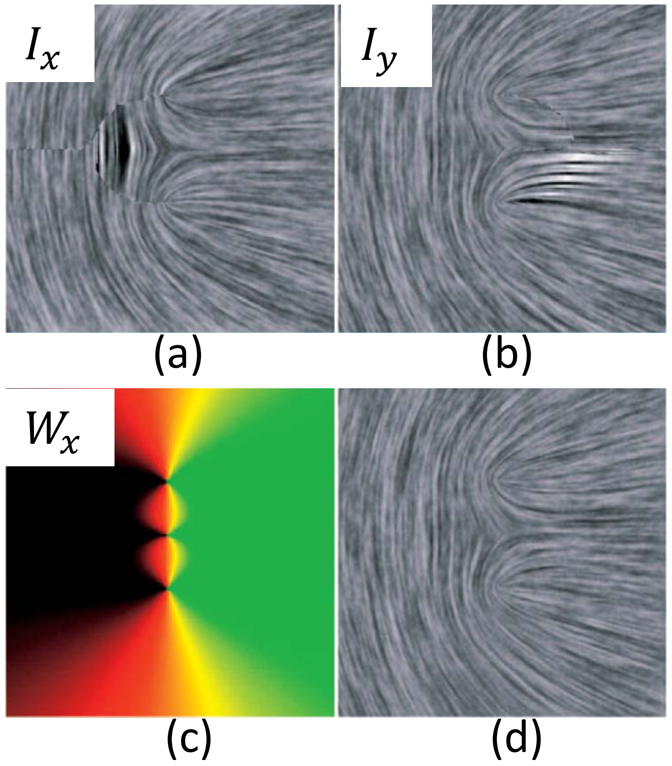


Extended IBFV

IBFV does not trace out hyperstreamlines. So it cannot address the angle ambiguity explicitly!

$$V_1 = V_x = \begin{cases} \rho\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix} & \text{if } \cos\theta \geq 0 \\ \rho\begin{pmatrix}-\cos\theta \\ -\sin\theta\end{pmatrix} & \text{otherwise,}\end{cases}$$

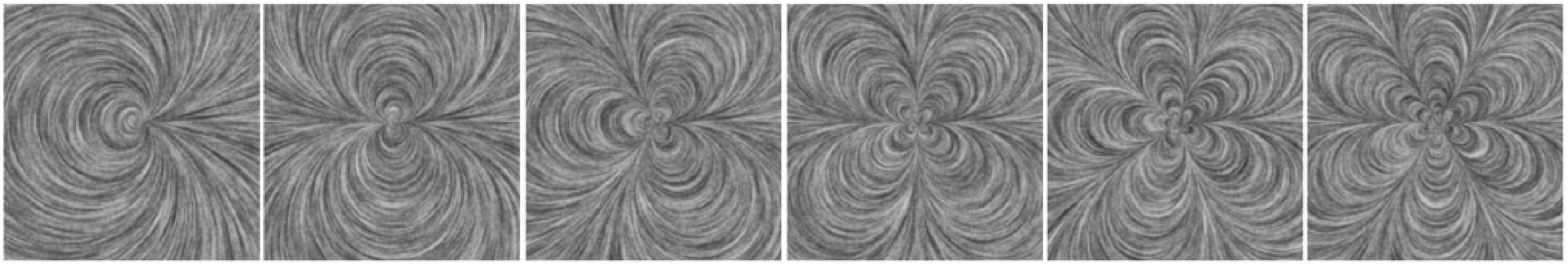
$$V_2 = V_y = \begin{cases} \rho\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix} & \text{if } \sin\theta \geq 0 \\ \rho\begin{pmatrix}-\cos\theta \\ -\sin\theta\end{pmatrix} & \text{otherwise.}\end{cases}$$

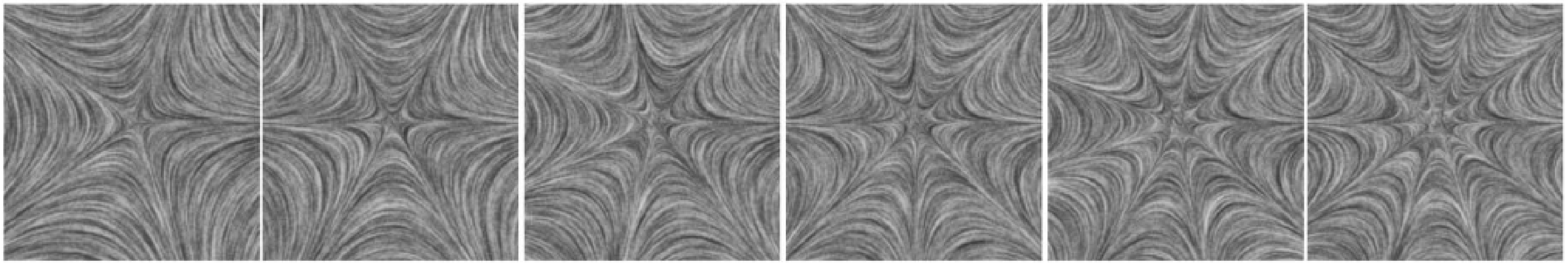


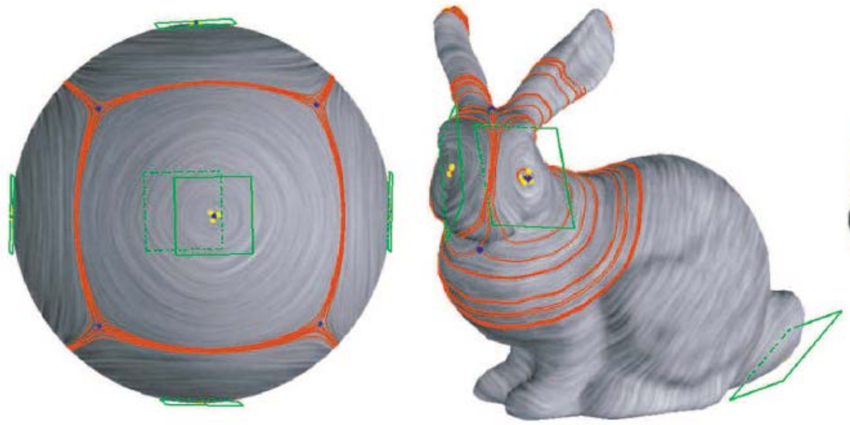
The system first produces images according to two direction assignments: ((a), in the positive x-direction) V_x and ((b), in the positive y-direction) V_y . The images are then blended according to weight functions W_x (a color coding shown in (c)) and $W_y = 1 - W_x$. (d) The resulting image no longer contains the visual artifacts from V_x and V_y .

[Zhang et al. TVCG07]

Some Results







[Zhang et al. TVCG07]

Extension to N-Symmetric Field Visualization

[Palacios and Zhang TVCG 2011]

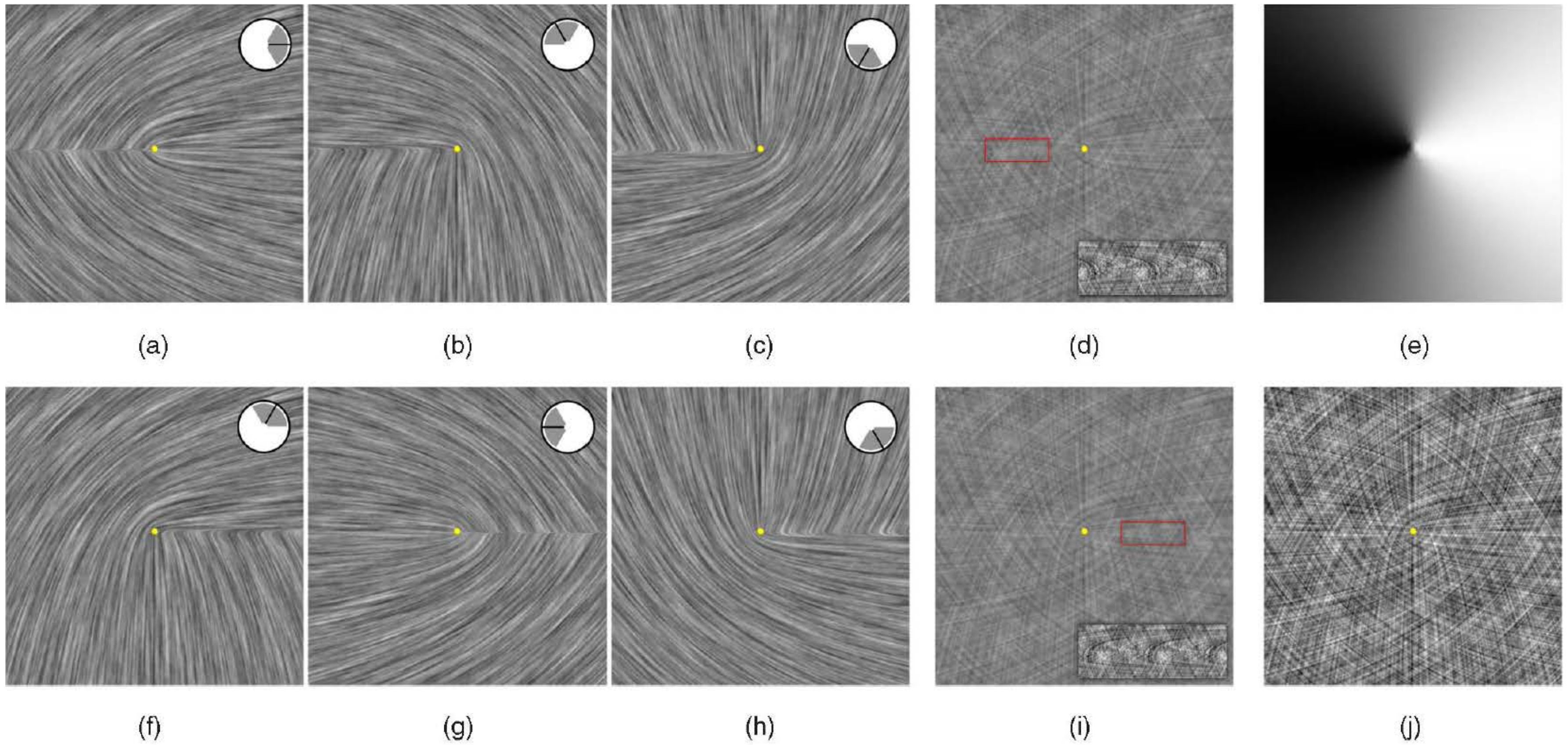


Fig. 5. Our visualization algorithm is demonstrated with an example 3-RoSy field S. In (a), (b), and (c), we applied the LIC algorithm to V_0, V_1, and V_2 (the guiding angle for each is shown in the upper-right corner) to obtain I_0, I_1, and I_2, respectively. Notice that while (a), (b), and (c) provide a complete coverage of the streamlines passing through any regular point in the domain, they have the same regions of breaking points (left X-axis). By blending them uniformly, we obtain I (d), a visualization of S with visual artifacts in the same place (a closeup of the artifact, highlighted in red, is seen as an inset with the contrast enhanced; note the curving patterns in a region that should be regular). To remedy the problem, we also apply the LIC algorithm to V_0', V_1', and V_2', generating the images I_0' (f), I_1' (g), and I_2' (h), and blend them uniformly to obtain I' (i). The visual artifacts in I' appear on the right side (again, a closeup is inset) of the X-axis. By blending I and I' using the weight map w (e), we obtain the final image in (j) in which the artifacts due to field discontinuities are no longer visible. Note that the image in (j) has had its contrast corrected via the transformation described in Section 5.

BRIEF REVIEW OF 2D TENSOR FIELD TOPOLOGY

Topological Skeleton in 2D

We only consider the topology for 2nd symmetric tensor fields!

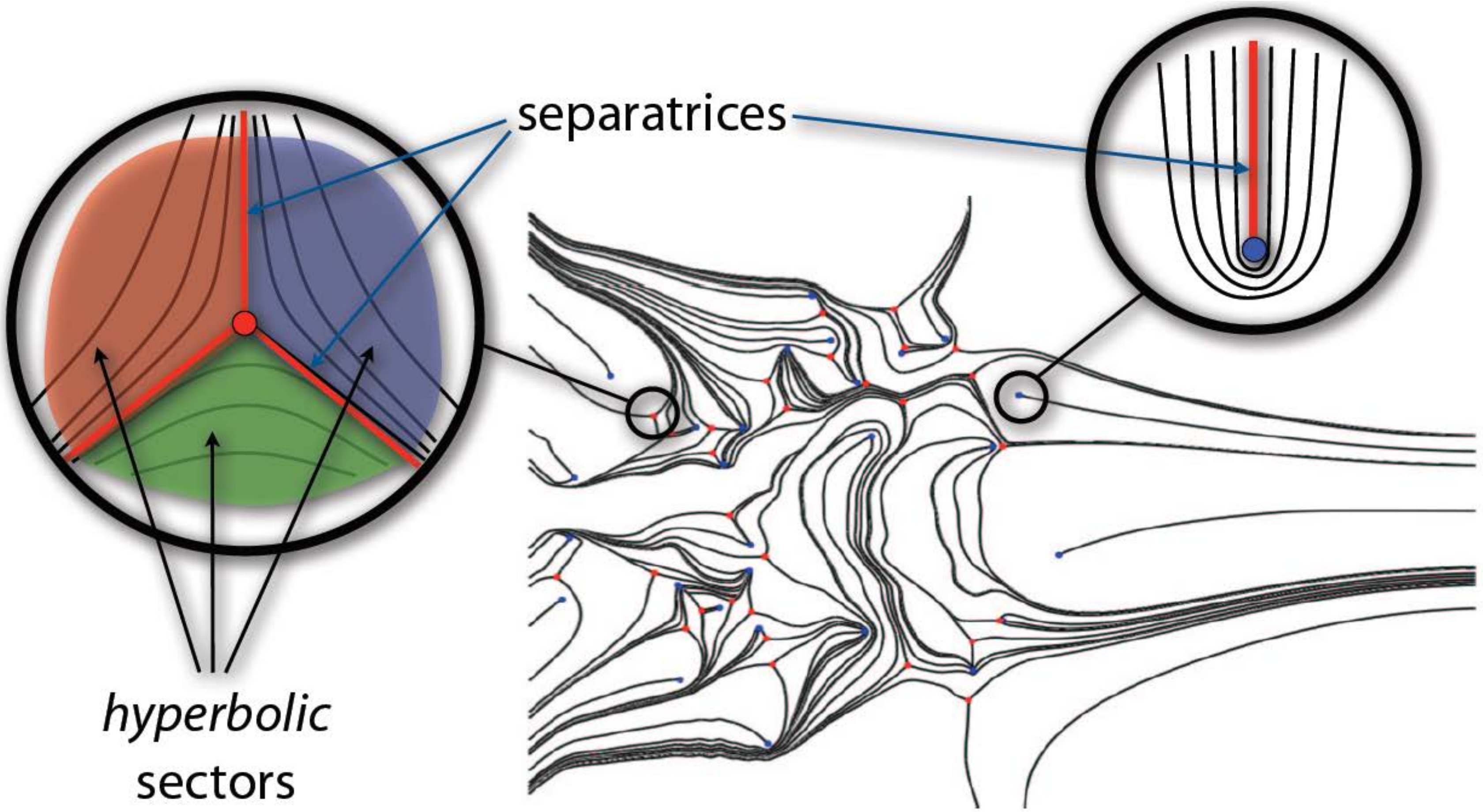
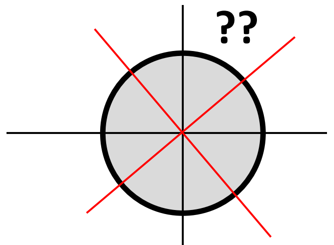


Image by Xavier Tricoche

Degenerate Points

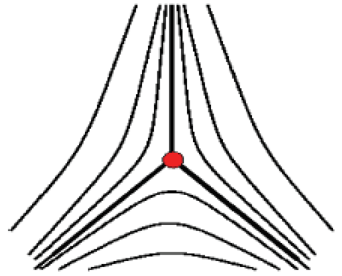
- The topology for 2nd symmetric tensor fields is extracted by identifying their **degenerate points** and their connectivity.

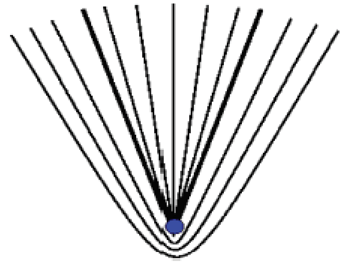
- A point $\boldsymbol{p}$ is a degenerate point of the tensor field T iff the two eigenvalues of $T(\mathbf{p})$ are equal to each other.
  - There are infinite many eigenvectors at $\boldsymbol{p}$.
  - Hyperstreamlines cross each other at only degenerate points (similar to streamlines)

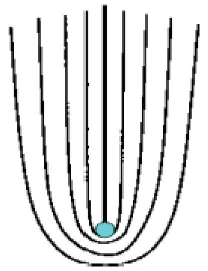


Degenerate Points in 2D

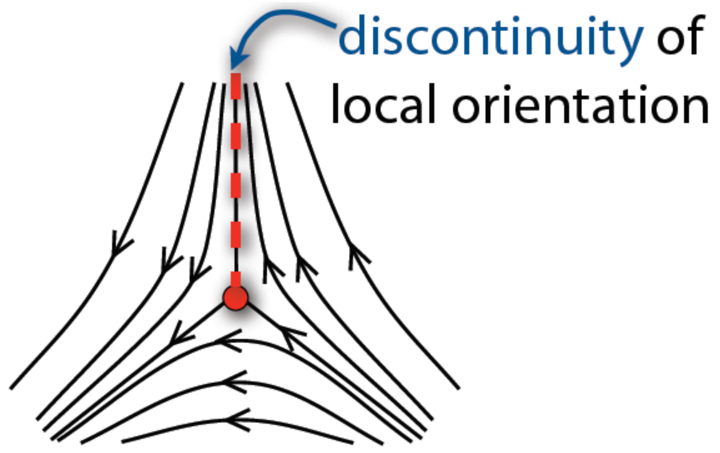
Three linear (first-order) types exist







None of these patterns would be possible in vector fields!



Degenerate Points in 2D

- Find degenerate points
  - *Form* $\tilde{\mathbf{D}} := \mathbf{D} - \frac{1}{2}\mathrm{trace}(\mathbf{D})\mathbf{I}_2 \Rightarrow \tilde{\mathbf{D}} = \begin{pmatrix} \alpha & \beta \\ \beta & -\alpha \end{pmatrix}$ deviatoric
  - *Solve in each cell for* $\tilde{\mathbf{D}}(x, y) = \mathbf{0}$

$$D = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix}$$

$$D - \frac{1}{2} trace(D) I_2 = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} - \begin{pmatrix} \frac{T_{11} + T_{22}}{2} & 0 \\ 0 & \frac{T_{11} + T_{22}}{2} \end{pmatrix} = \cdots$$

Degenerate Points in 2D

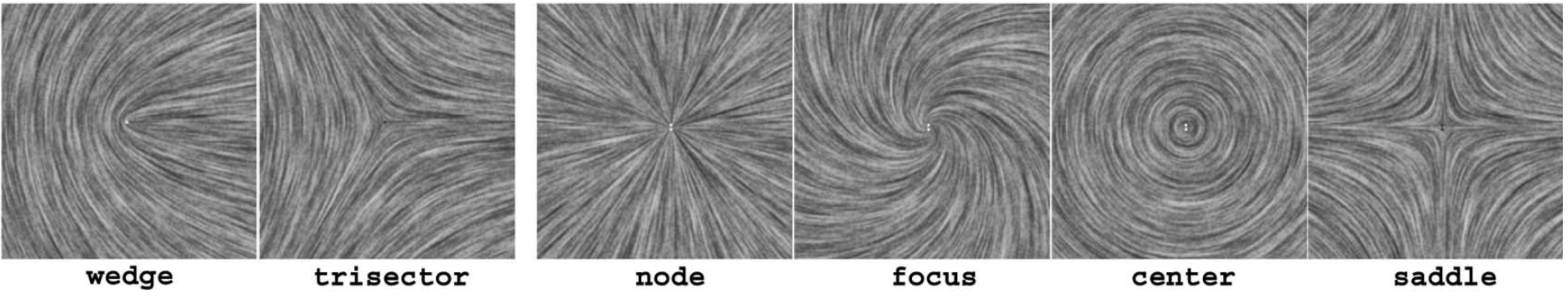
- Classifying tensor degenerate points

$$\begin{pmatrix} \alpha(x,y) & \beta(x,y) \\ \beta(x,y) & -\alpha(x,y) \end{pmatrix} = \begin{pmatrix} \alpha_1 x + \alpha_2 y & \beta_1 x + \beta_2 y \\ \beta_1 x + \beta_2 y & -(\alpha_1 x + \alpha_2 y) \end{pmatrix}$$

  - Depending on the determinant of $\begin{pmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{pmatrix}$
    - >0 wedge
    - <0 trisector
    - =0 higher-order degenerate points

A few degenerate points used in tensor field design

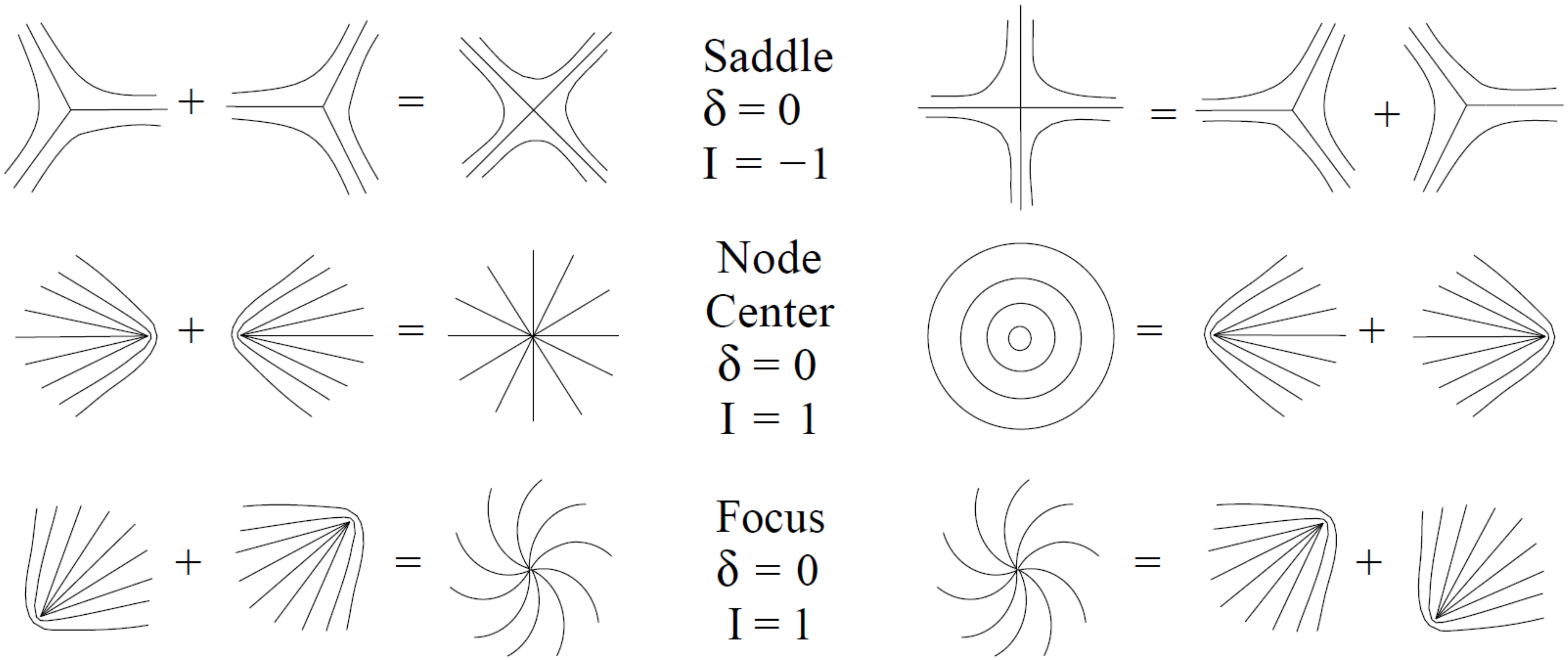
$$\begin{pmatrix} x^2-y^2 & -2xy \\ -2xy & -(x^2-y^2) \end{pmatrix}$$



$$\begin{pmatrix} x & y \\ y & -x \end{pmatrix} \quad \begin{pmatrix} x & -y \\ -y & -x \end{pmatrix} \quad \begin{pmatrix} x^2-y^2 & 2xy \\ 2xy & -(x^2-y^2) \end{pmatrix} \quad \begin{pmatrix} y^2-x^2 & -2xy \\ -2xy & -(y^2-x^2) \end{pmatrix}$$

$$\begin{pmatrix} \alpha_1 x+\alpha_2 y & \beta_1 x+\beta_2 y \\ \beta_1 x+\beta_2 y & -(\alpha_1 x+\alpha_2 y) \end{pmatrix} \Rightarrow \begin{pmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{pmatrix}$$

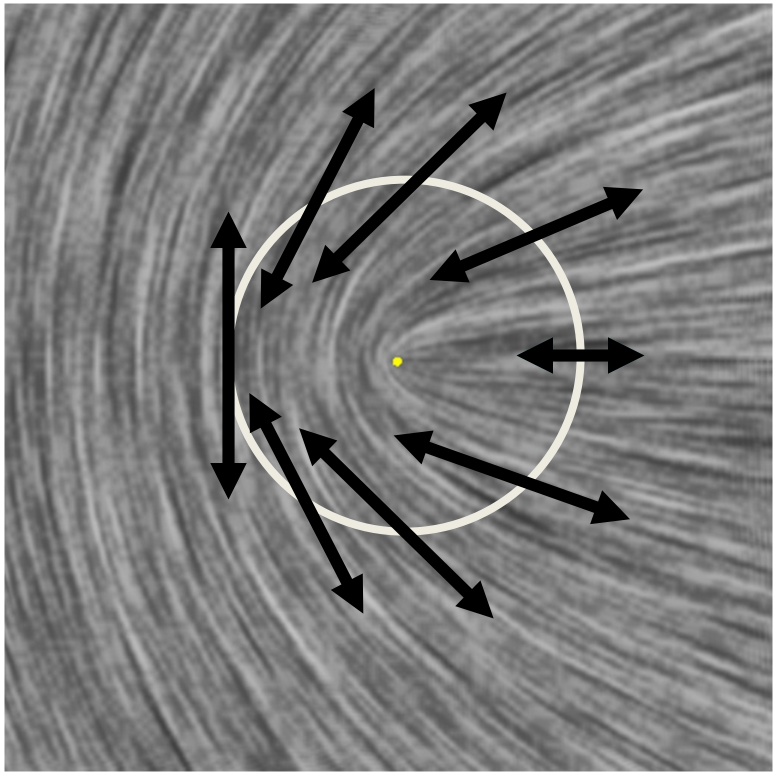
>0 wedge
<0 trisector
=0 higher-order degenerate points

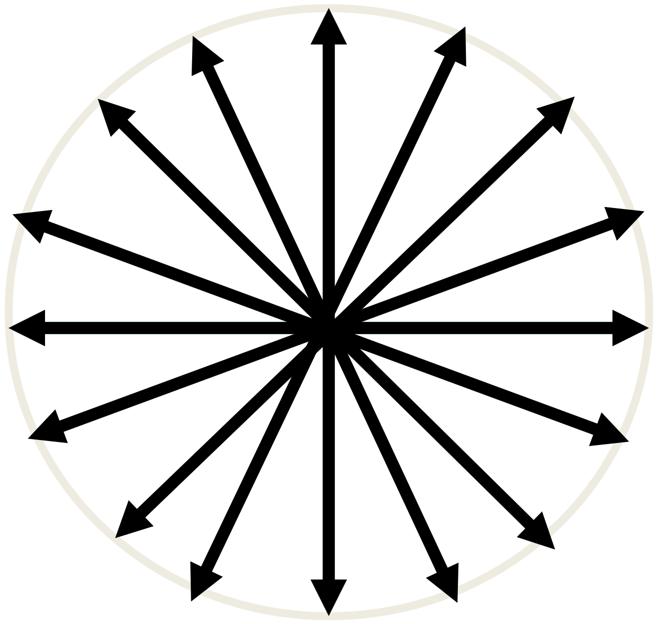


[Delmarcelle and Hesselink, 1994]

Degenerate Points in 2D

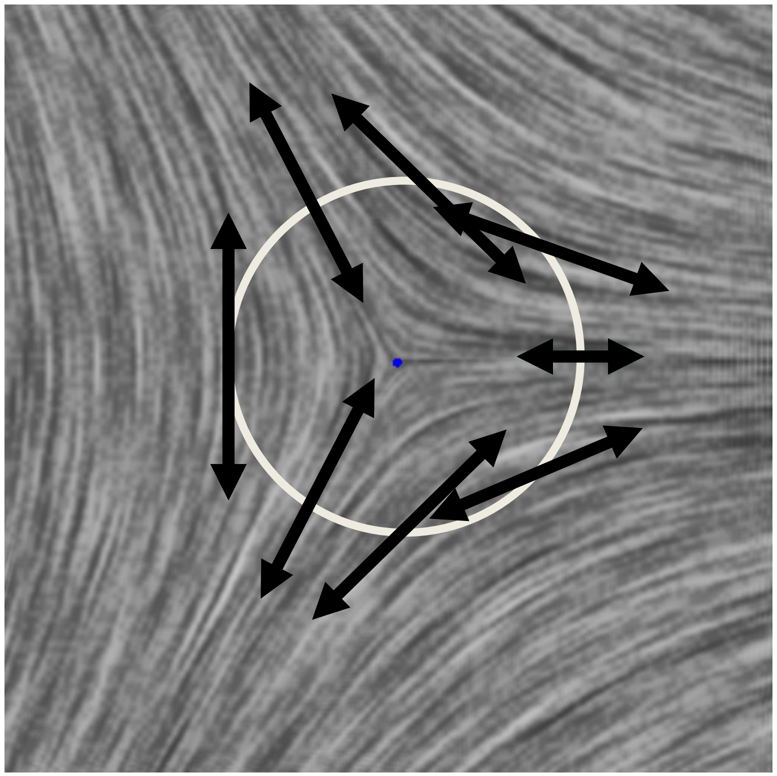
- Tensor index: wedges

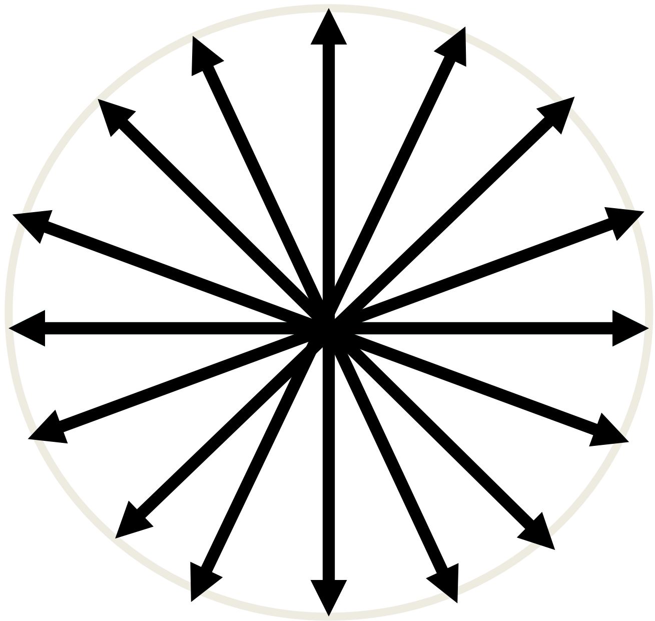




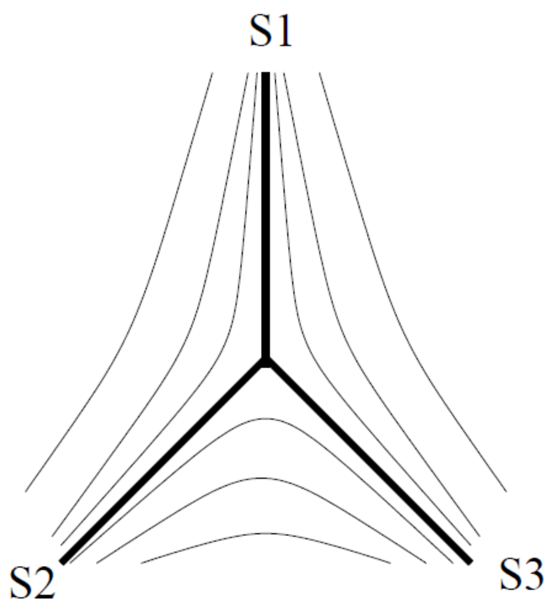
Degenerate Points in 2D

- Tensor index: trisectors

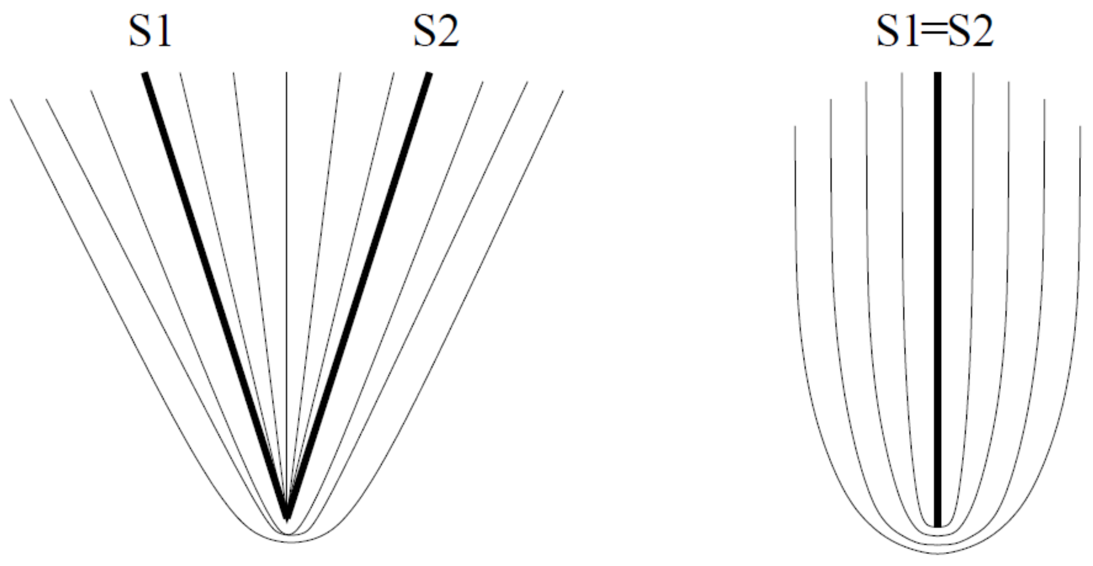




Separatrices



TRISECTOR



WEDGE POINTS

Hyperbolic sectors n_h
Parabolic sectors n_p

Index $I = 1 - \frac{n_h}{2}$

Separatrices

- Find degenerate points
  - *Form* $\underbrace{\tilde{\mathbf{D}} := \mathbf{D} - \frac{1}{2}\mathrm{trace}(\mathbf{D})\mathbf{I}_2}_{\text{deviatoric}} \Rightarrow \tilde{\mathbf{D}} = \begin{pmatrix} \alpha & \beta \\ \beta & -\alpha \end{pmatrix}$
  - *Solve in each cell for* $\tilde{\mathbf{D}}(x, y) = \mathbf{0}$
- Compute separatrices
  - *Linear analysis at each singularity* $\left\{ \begin{array}{lcl} \alpha(x,y) & \approx & \alpha_1\, x + \alpha_2\, y \\ \beta(x,y) & \approx & \beta_1\, x + \beta_2\, y \end{array} \right.$
  - *Determine angular coordinate of separatrices*

$$\beta_2\, u^3 + (\beta_1 + 2\alpha_2)\, u^2 + (2\alpha_1 - \beta_2)\, u - \beta_1 = 0$$
$$u := \tan\theta$$

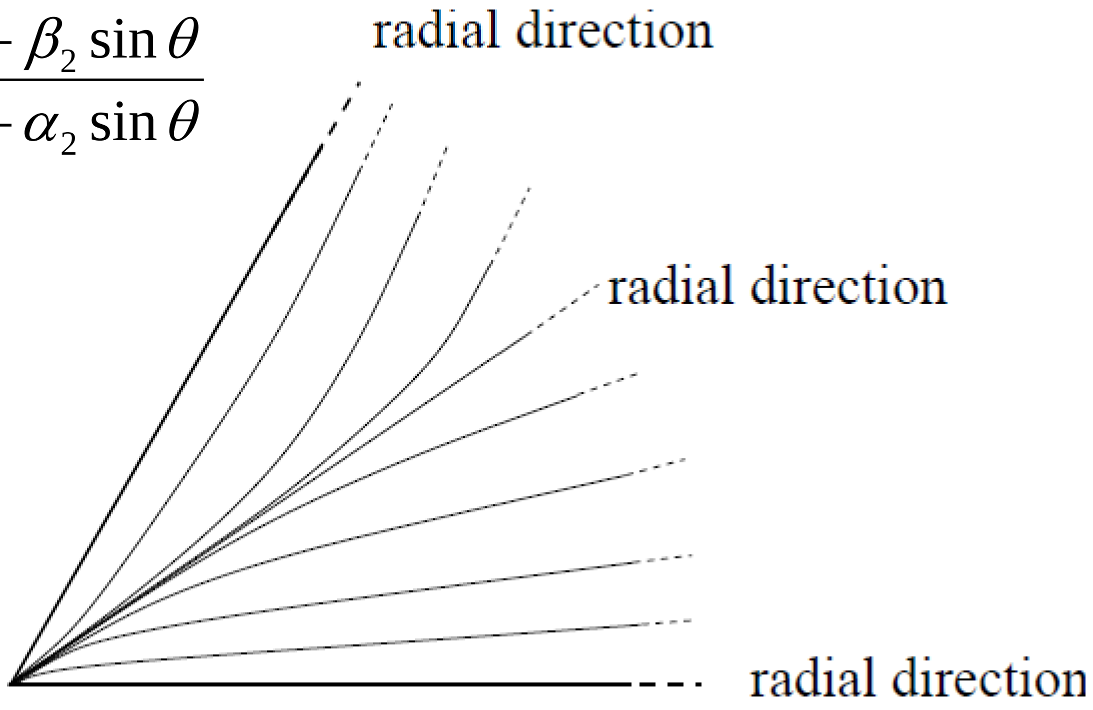
(Delmarcelle and Hesselink, 1994)

  - *Integrate separatrices (standard ODE solver with embedded orientation consistency check)*

Separatrices

- Separatrices

$$\frac{\sin 2\theta}{\cos 2\theta} = \frac{\beta_1 \cos\theta + \beta_2 \sin\theta}{\alpha_1 \cos\theta + \alpha_2 \sin\theta}$$



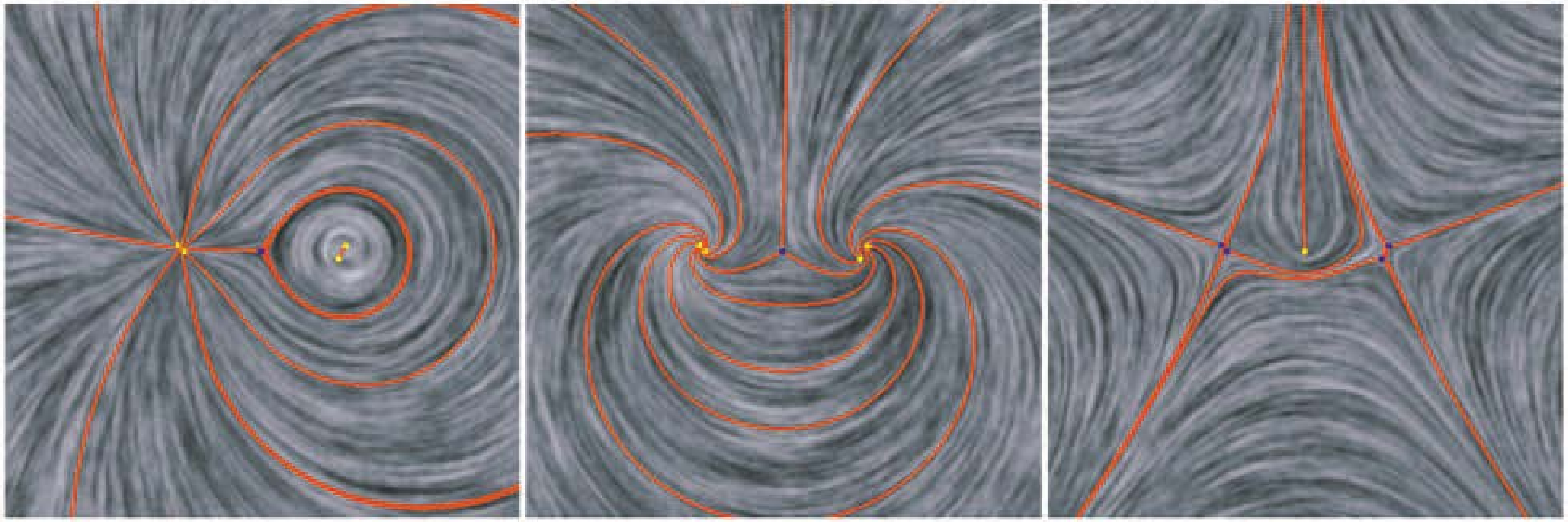
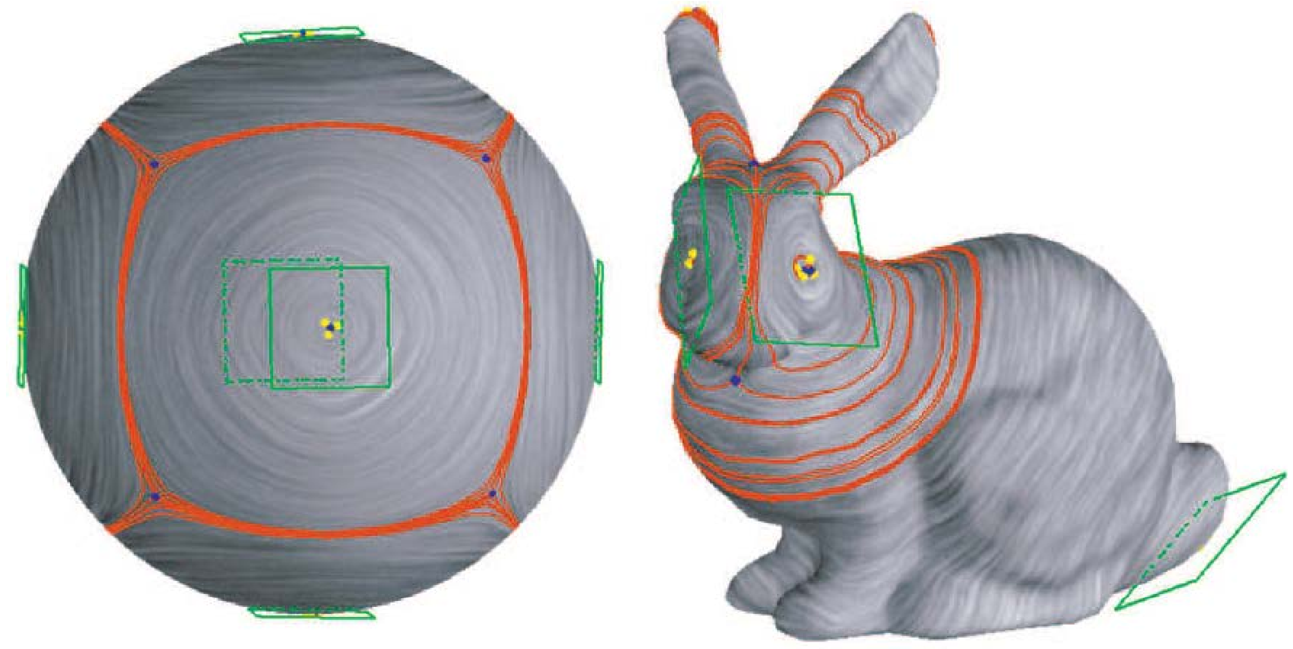



Image by Eugene Zhang

Additional Readings

- David Weinstein, Gordon Kindlmann, and Eric Lundberg. "Tensorlines: Advection-diffusion based propagation through diffusion tensor fields." *Proceedings of the conference on Visualization'99: celebrating ten years*. IEEE Computer Society Press, 1999.

- Guoning Chen, Gregory Esch, Peter Wonka, Pascal Mueller and Eugene Zhang. “Interactive Procedural Street Modeling”. ACM Transaction on Graphics 27 (3) (SIGGRAPH 2008).

- Xiaoqiang Zheng and Alex Pang. “HyperLIC”. IEEE Visualization 2003.

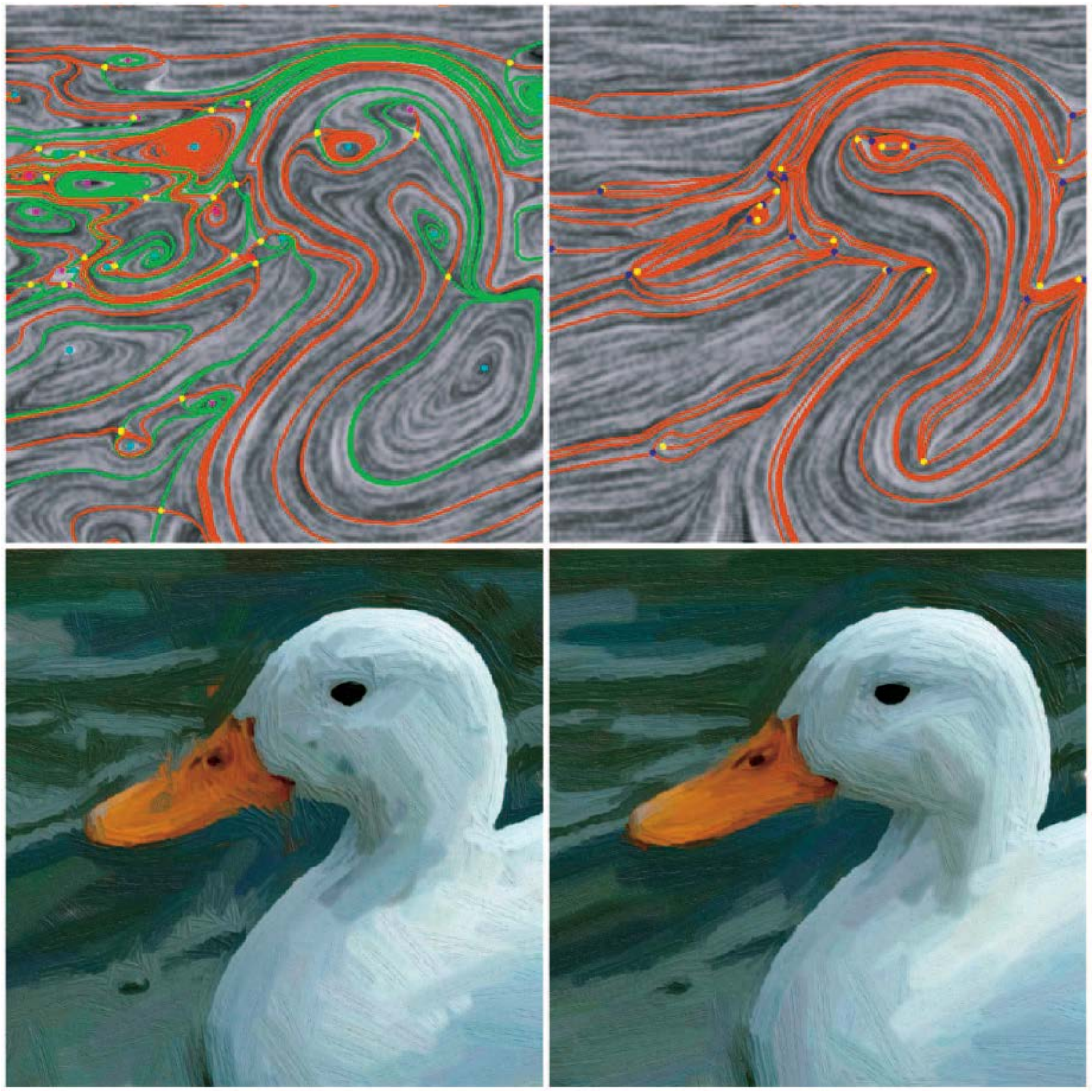
- Hotz, I., Feng, L., Hagen, H., Hamann, B., Jeremic, B., Joy, K. “Physically Based Methods for Tensor Field Visualization”*.* IEEE Visualization 2004, pp. 123-130.

- Eugene Zhang, James Hays, and Greg Turk, "Interactive Tensor Field Design and Visualization on Surfaces", IEEE Transactions on Visualization and Computer Graphics, Vol 13(1), 2007, pp 94-107.

- Jonathan Palacios and Eugene Zhang, "Interactive Visualization of Rotational Symmetry Fields on Surfaces", IEEE Transactions on Visualization and Computer Graphics, Vol. 17(7), 2011, pp. 947-955.

- Eugene Zhang, Harry Yeh, Zhongzang Lin, and Robert S. Laramee, "Asymmetric Tensor Analysis for Flow Visualization", IEEE Transactions on Visualization and Computer Graphics, Vol. 15(1), 2009, pp. 106-122.

Compared With Vector Field Topology



Comparison between the vector-based image edge field (VIEF, left) and the tensor-based image edge field (TIEF, right) for painterly rendering of an image of a duck.

Notice that TIEF is much smoother than VIEF (top row), and their impact on the painterly results are clearly visible near the beak of the duck.

Image by Eugene Zhang

Acknowledgment

- Thanks for materials from
  - Dr. Eugene Zhang
  - Dr. Xavier Tricoche