Tensor Field Visualization II

Goal: have a basic understanding of the geometric-based and texture-based visualization techniques for second-order tensor fields.
GEOMETRIC-BASED METHOD 
TRACING HYPERSTREAMLINES
Hyperstreamlines

• Let $\mathbf{T}(\mathbf{x})$ 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
  $s_{i+1} = s_i + v(s_i) \cdot dt$

• **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**.
Red – major
Green – minor
Compute One Hyperstreamline

• Choose integrator:
  – Euler
  – Runge-Kutta (2\textsuperscript{nd}, 4\textsuperscript{th}, 45)
  – Others
• Choose step size (can be adaptive)
• Provide seed point position and \textit{determine starting direction}
• Advance the front
• Termination conditions
Compute One Hyperstreamline

- Choose integrator:
  - Euler
  - Runge-Kutta (2\text{nd}, 4\text{th}, 45)
  - Others
- Choose step size (can be adaptive)
- Provide seed point position and determine starting direction
- Advance the front
- Termination conditions

- **Note that the angle ambiguity.** This is because the computation of the eigenvector at each sample point (i.e., vertex 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, 45)
  - Others
- Choose step size (can be adaptive)
- Provide seed point position and determine starting direction
- Advance the front
- Termination conditions

- **Note that the angle ambiguity.** This is because the computation of the eigenvector at each sample point (i.e., vertex 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 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*
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

Hybrid visualization: hyperstreamlines + glyphs

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

[Prckovska et al. 2010]
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 and ambiguous*
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]
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

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

\( \omega_{punct} \) is a user-controlled parameter

<table>
<thead>
<tr>
<th>Anisotropy</th>
<th>Direction In</th>
<th>Desired Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>Any</td>
<td>( e_1 )</td>
</tr>
<tr>
<td>Planar</td>
<td>Tangential to disk</td>
<td>( \mathbf{v}<em>{in} ) or ( \mathbf{v}</em>{out} )</td>
</tr>
<tr>
<td>Planar</td>
<td>Normal to disk plane</td>
<td>( \mathbf{v}_{out} )</td>
</tr>
<tr>
<td>Spherical</td>
<td>Any</td>
<td>( \mathbf{v}<em>{in} ) or ( \mathbf{v}</em>{out} )</td>
</tr>
</tbody>
</table>

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

\[ v_{prop} = c_l e_1 + (1 - c_l) ((1 - \omega_{punct}) v_{in} + \omega_{punct} v_{out}) \]

\( \omega_{punct} \) is a user-controlled parameter

<table>
<thead>
<tr>
<th>Anisotropy</th>
<th>Direction In</th>
<th>Desired Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>Any</td>
<td>( e_1 )</td>
</tr>
<tr>
<td>Planar</td>
<td>Tangential to disk</td>
<td>( v_{in} ) or ( v_{out} )</td>
</tr>
<tr>
<td>Planar</td>
<td>Normal to disk plane</td>
<td>( v_{out} )</td>
</tr>
<tr>
<td>Spherical</td>
<td>Any</td>
<td>( v_{in} ) or ( v_{out} )</td>
</tr>
</tbody>
</table>

[Weinstein et al.1999]
TEXTURE-BASED METHOD
How to extend it to symmetric tensor field?
Instead of using a 1D kernel along the streamline, HyperLIC uses a 2D kernel.
Instead of using a 1D kernel along the streamline, HyperLIC uses a 2D kernel

\[ I_o(P) = \frac{\sum_{i=-N}^{N} \sum_{j=-N}^{N} k(i,j) I_n(P_{i,j})}{\sum_{i=-N}^{N} \sum_{j=-N}^{N} k(i,j)} \]

\[ P_{i,j} = P_{i-1,j} + \lambda_1(P_{i-1,j}) e_1(P_{i-1,j}) \Delta t \]

\[ P_{0,j} = P_{0,j-1} + \lambda_2(P_{0,j-1}) e_2(P_{0,j-1}) \Delta t \]

\[ P_{0,0} = P \]

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

If we define

\[
I_1(P) = \frac{\sum_{j=-N}^{N} k_2(j) I_n(P_j)}{\sum_{j=-N}^{N} k_2(j)}
\]

\[
P_j = P_{j-1} + \lambda_2(P_{j-1}) e_2(P_{j-1}) \Delta t
\]

\[
P_0 = P
\]

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

Then

\[ I_o(P) = \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \]

\[ P_i = P_{i-1} + \lambda_1(P_{i-1}) e_1(P_{i-1}) \Delta t \]

\[ P_0 = P \]

Second, use \( I_1 \) as input texture, then advect along the \textbf{major} eigen vector direction.

The LIC pipeline
Decompose the computation

Then

\[ I_o(P) = \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \]

\[ P_i = P_{i-1} + \lambda_1 (P_{i-1}) e_1 (P_{i-1}) \Delta t \]

\[ P_0 = P \]

This is a two-pass process

\( I_1 \) and \( I_0 \) are the output images of the **un-normalized** LIC on \( \lambda_2 e_2 \) and \( \lambda_1 e_1 \) vector fields with input images \( I_n \) and \( I_1 \), respectively.
Decompose the computation

Then

\[ I_o(P) = \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \]

\[ P_i = P_{i-1} + \lambda_1(P_{i-1}) e_1(P_{i-1}) \Delta t \]

\[ P_0 = P \]

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
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

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 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 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!

\[ V_1 = V_x = \begin{cases} 
\rho \left( \frac{\cos \theta}{\sin \theta} \right) & \text{if } \cos \theta \geq 0 \\
\rho \left( \frac{-\cos \theta}{-\sin \theta} \right) & \text{otherwise,}
\end{cases} \]

\[ V_2 = V_y = \begin{cases} 
\rho \left( \frac{\cos \theta}{\sin \theta} \right) & \text{if } \sin \theta \geq 0 \\
\rho \left( \frac{-\cos \theta}{-\sin \theta} \right) & \text{otherwise.}
\end{cases} \]

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]
Extension to N-Symmetric Field Visualization

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 2\textsuperscript{nd} symmetric tensor fields!

Image by Xavier Tricoche
Degenerate Points

• The topology for 2\textsuperscript{nd} symmetric tensor fields is extracted by identifying their degenerate points and their connectivity.

• A point $p$ is a degenerate point of the tensor field $T$ iff the two eigenvalues of $T(p)$ are equal to each other.
  – There are infinite many eigenvectors at $p$.
  – Hyperstreamlines cross each other at only degenerate points (similar to streamlines)
Degenerate Points in 2D

Three linear (first-order) types exist:

- Trisector
- Wedge I
- Wedge II

None of these patterns would be possible in vector fields!
Degenerate Points in 2D

- Find degenerate points
- Form
  \[ \tilde{D} := D - \frac{1}{2} \text{trace}(D) I_2 \]
- Solve in each cell for \( \tilde{D}(x, y) = 0 \)

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

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

\[
\begin{pmatrix}
\alpha_1 x + \alpha_2 y \\
\beta_1 x + \beta_2 y
\end{pmatrix}
\begin{pmatrix}
\alpha_1 \\
\beta_1
\end{pmatrix}
= 0 \text{ higher-order degenerate points}
\]

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

>0 wedge

<0 trisector

=0 higher-order degenerate points
Saddle
$\delta = 0$
$I = -1$

Node
$\delta = 0$
$I = 1$

Focus
$\delta = 0$
$I = 1$

[Delmarcelle and Hesselink, 1994]
Degenerate Points in 2D

- Tensor index: wedges
Degenerate Points in 2D

• Tensor index: trisectors
Separatrices

Hyperbolic sectors \( n_h \)
Parabolic sectors \( n_p \)

Index \( I = 1 - \frac{n_h}{2} \)
Separatrices

- Find degenerate points

\[ \tilde{D} := D - \frac{1}{2} \text{trace}(D) I_2 \Rightarrow \tilde{D} = \begin{pmatrix} \alpha & \beta \\ \beta & -\alpha \end{pmatrix} \]

- Solve in each cell for \( \tilde{D}(x, y) = 0 \)

- Compute separatrices
  - Linear analysis at each singularity
    \[ \begin{align*}
    \alpha(x,y) & \approx \alpha_1 x + \alpha_2 y \\
    \beta(x,y) & \approx \beta_1 x + \beta_2 y
    \end{align*} \]
  
- 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 Hesselinck, 1994)

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

\[
\frac{\sin 2\theta}{\cos 2\theta} = \frac{\beta_1 \cos \theta + \beta_2 \sin \theta}{\alpha_1 \cos \theta + \alpha_2 \sin \theta}
\]
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
Additional Readings


Acknowledgment

• Thanks for materials from
  – Dr. Eugene Zhang
  – Dr. Xavier Tricoche