Image-space Tensor Field Visualization Using a LIC-like Method

Sebastian Eichelbaum, Mario Hlawitschka, Bernd Hamann, and Gerik Scheuermann

Abstract Tensors are of great interest to many applications in engineering and in medical imaging, but a proper analysis and visualization remains challenging. Physics-based visualization of tensor fields has proven to show the main features of symmetric second-order tensor fields, while still displaying the most important information of the data, namely the main directions in medical diffusion tensor data using texture and additional attributes using color-coding, in a continuous representation. Nevertheless, its application and usability remains limited due to its computational expensive and sensitive nature.

We introduce a novel approach to compute a fabric-like texture pattern from tensor fields motivated by image-space line integral convolution (LIC). Although, our approach can be applied to arbitrary, non-selfintersecting surfaces, we are focusing on special surfaces following neural fibers in the brain. We employ a multi-pass rendering approach whose main focus lies on regaining three-dimensionality of the data under user interaction as well as being able to have a seamless transition between local and global structures including a proper visualization of degenerated points.

1 Motivation and Related Work

Since the introduction of tensor lines and hyperstreamlines [5], there have been many research efforts directed at the continuous representation of tensor fields, including research on tensor field topology [11, 24, 23]. Zheng and Pang introduced HyperLIC [31], which makes it possible to display a single eigendirection of a tensor

Sebastian Eichelbaum · Gerik Scheuermann
Abteilung für Bild- und Signalverarbeitung, Institut für Informatik, Universität Leipzig, Germany, e-mail: {eichelbaum | scheuermann }@informatik.uni-leipzig.de

Mario Hlawitschka and Bernd Hamann
Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis, CA, e-mail: {hlawitschka | hamann}@ucdavis.edu

field in a continuous manner by smoothing a noise texture along integral lines, while neglecting secondary directions. Recent approaches to visualize Lagrangian structures on tensor fields [12] provide information on one chosen tensor direction and are especially useful for diffusion tensor data, where the main tensor direction can be correlated to neural fibers or muscular structures, whereas the secondary direction only plays a minor role. More recently, Dick et al. [6] published an interactive approach to visualize volumetric tensor field for implant planning.

While glyph-based visualization techniques reveal a huge amount of the local information, even with advanced glyph placement techniques [16], an integration of this information into global structures and an interpolation of information in-between glyphs remains the responsibility of the user. On the other side, approaches focusing on global structures, like [17], which is used to calculate a skeleton of white matter tracts, are not able to provide local information or at least a smooth transition to local structures.

Hotz et al. [13] introduced Physically Based Methods (PBM) for tensor field visualization in 2004 as a means to visualize stress and strain tensors arising in geomechanics. A positive-definite metric that has the same topological structure as the tensor field is defined and visualized using a texture-based approach resembling LIC [3]. Besides other information, eigenvalues of the metric can be encoded by free parameters of the texture definition, such as the remaining color space. Whereas the method's implementation for parameterizable surfaces topologically equivalent to discs or spheres is straightforward, implementations for arbitrary surfaces remains computationally challenging. In 2009, Hotz et al. [14] enhanced their approach to isosurfaces in three-dimensional tensor fields. A three-dimensional noise texture is computed in the data set and a convolution is performed along integral lines tangential to the eigenvector field. LIC has been used in vector field visualization methods to imitate *Schlieren* patterns on surfaces experiments where a thin film of oil is applied to surfaces, which show patterns caused by the air flow. In vector field visualization, image-space LIC is a method to compute *Schlieren*-like textures in image space [28, 29, 19, 20, 9], intended for large and non-parameterized geometries. As these methods are based on vector fields, their application to a symmetric tensor field's major eigenvector is possible, with the limitation that the eigenvector field does not provide an orientation. For asymmetric tensor-fields, other approaches exist [30]. Besides the non-trivial application of image-space LIC to tensor data, image-space LIC has certain other drawbacks. Mainly because the noise pattern is defined in image space, it does not follow the movement of the surface and, therefore, during user interaction, the consistent surface impression is lost. A simple method proposed to circumvent this problem is animating the texture pattern by applying randomized trigonometric functions to the input noise. Weiskopf and Ertl [27] solved this problem for vector field visualization by generating a three-dimensional texture that is scaled appropriately in physical space.

We implemented an algorithm similar to the original PBM but for arbitrary non-intersecting surfaces in image space. Our algorithm can perform at interactive frame rates for large data sets on current desktop PCs. We overcome the drawbacks present in image-space LIC implementations by defining a fixed parameterization on the

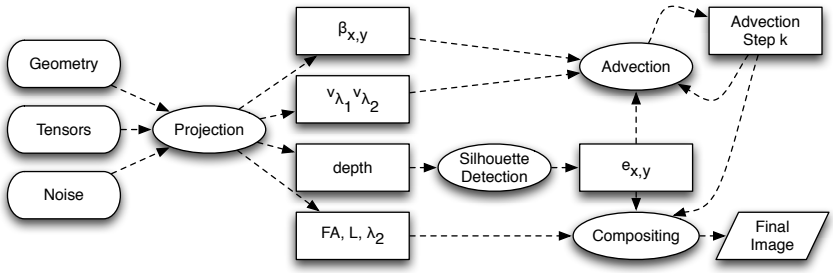

Fig. 1 Flowchart indicating the four major steps of the algorithm: **projection**, which transforms the data set in an image-space representation and produce the initial noise texture $\beta_{x,y}$ on the geometry; **silhouette detection**, required for the advection step and the final rendering; **advection**, which produces the two eigenvector textures; and the final **compositing**, which combines intermediate textures to the final visualization. Between consecutive steps, the data is transferred using textures.

surface. Thus, we do not require a three-dimensional noise texture representation defined at sub-voxel resolution on the data set. Our approach is capable of maintaining local coherence of the texture pattern between frames when (1) transforming, rotating, or scaling the visualization, and (2) changing the surface by, e.g., changing isovalues or sweeping the surface through space. In addition, we implement special application-dependent modes to ensure our method integrates well with existing techniques.

2 Method

We employ a multi-pass rendering technique that consists of four major rendering passes as outlined in Figure 1. After generating the basic input textures once, the first pass projects all required data into image space. Pass two performs a silhouette detection that is used to guarantee integrity of the advection step computed by multiple iterations of pass three. Eventually, pass four composes the intermediate textures in a final rendering.

2.1 Projection into Image Space

First, we project the data into image space by rendering the surface using the default OpenGL rendering pipeline. Notably, the surface does not need to be represented by a surface mesh. Any other representation that provides proper depth and surface normal information works just as well (e.g., ray-casting methods for implicit surfaces, cf. Knoll et al. [18]). In the same rendering step, the tensor field is transformed from world space to object space, i.e., each tensor T, which is interpolated at the point on

the surface from the surrounding two- or three-dimensional tensor field is projected onto the surface by

$$T' = P \cdot T \cdot P^T, \tag{1}$$

with a matrix P defined using the surface normal n as

$$P = \begin{pmatrix} 1-n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1-n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1-n_z^2 \end{pmatrix}. \tag{2}$$

The camera viewing system configuration and the available screen resolution imply a super- or sub-sampling of the data. We obtain an interpolated surface tensor in every pixel which is decomposed into the eigenvector/eigenvalue representation using a method derived from the one presented by Hasan et al. [10]. These eigenvectors, which are still defined in object space, are projected into image space using the same projection matrices M_M and M_P used for projecting the geometry to image space, usually the standard *modelview* and *projection* matrices OpenGL offers:

$$v'_{\lambda_i} = M_P \times M_M \times v_{\lambda_i}, \text{ with } (i \in 1,2). \tag{3}$$

Even in the special case of symmetric second-order tensors in $\mathbb{R}^3$, which have three real-valued eigenvalues and three orthogonal eigenvectors in the non-degenerate case, in general, the projected eigenvectors are not orthogonal in two-dimensional space. To simplify further data handling, we scale the eigenvectors as follows:

$$\|v\|_\infty = max\{|v_x|, |v_y|\} \tag{4}$$

$$v''_{\lambda_i} = \frac{v'_{\lambda_i}}{\|v'_{\lambda_i}\|_\infty} \text{ with } i \in \{1,2\}, \quad \text{and} \quad \|v'_{\lambda_i}\|_\infty \neq 0 \tag{5}$$

The special case $\|v'_{\lambda_i}\|_\infty = 0$ only appears when the surface normal is perpendicular to the view direction and, therefore, can be ignored. The maximum norm in Equation 4 ensures that one component is 1 or -1 and, therefore, one avoids numerical instabilities arising when limited storage precision is available, and can use memory-efficient eight-bit textures.

2.2 Initial Noise Texture Generation

In contrast to standard LIC approaches, to achieve a proper visual representation of the data, high-frequency noise textures, such as white noise, are not suitable for the compositing of multiple textures. Therefore, we compute the initial noise texture using the reaction diffusion scheme first introduced by Turing [25] to simulate the mixture of two reacting chemicals, which leads to larger but smooth "spots" that are randomly and almost uniquely distributed (cf. Figure 2, right). For the discrete case,

the governing equations are:

$$\begin{aligned} \Delta a_{i,j} &= F(i,j) + D_a \cdot (a_{i+1,j} + a_{i-1,j} + a_{i,j+1} + a_{i,j-1} - 4 \cdot a_{i,j}), \\ \Delta b_{i,j} &= G(i,j) + D_b \cdot (b_{i+1,j} + b_{i-1,j} + b_{i,j+1} + b_{i,j-1} - 4 \cdot b_{i,j}), \text{where} \\ F(i,j) &= s(16 - a_{i,j} \cdot b_{i,j}) \text{ and } G(i,j) = s(a_{i,j} \cdot b_{i,j} - b_{i,j} - \beta_{i,j}). \end{aligned} \tag{6}$$

Here, we assume continuous boundary conditions to obtain a seamless texture in both directions. The scalar s allows one to control the size of the spots where a smaller value of s leads to larger spots. The constants D_a and D_b are the diffusion constants of each chemical. We use $D_a = 0.125$ and $D_b = 0.031$ to create the input textures.

2.3 Noise Texture Transformation

Mapping the initial texture to the geometry is a difficult and application-dependent task. Even though there exist methods to parameterize a surface, they employ restrictions to the surface (such as being isomorphic to discs or spheres), require additional storage for texture atlases (cf. [21, 15]) and, in general, require additional and often time-consuming pre-processing.

Another solution, proposed by Turk et al. [26], calculates the reaction diffusion texture directly on the surface. A major disadvantage of this method is the computational complexity. Even though these approaches provide almost distortion-free texture representations, isosurfaces, for example, may consist of a large amount of unstructured primitives, which increases the pre-processing time tremendously.

Whereas previous approaches for image space LIC either use parameterized surfaces to apply the initial noise pattern to the surface or use locally or globally defined three-dimensional textures [27], we define an implicit parameterization of the surface that provides an appropriate mapping of the noise texture to the surface.

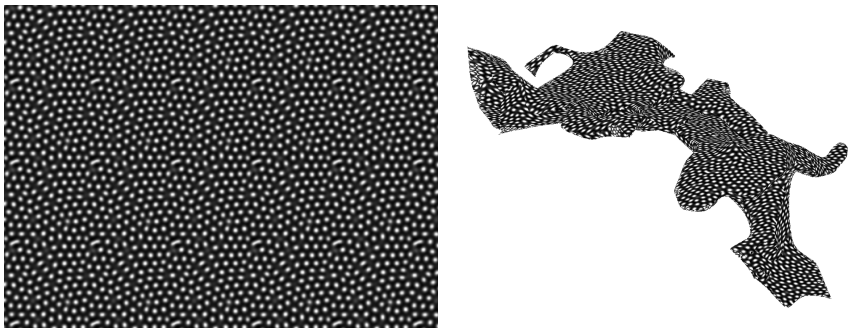

Fig. 2 Illustration of the reaction diffusion texture used (left) and the noise texture mapped to geometry $\beta_{x,y}$ (right).

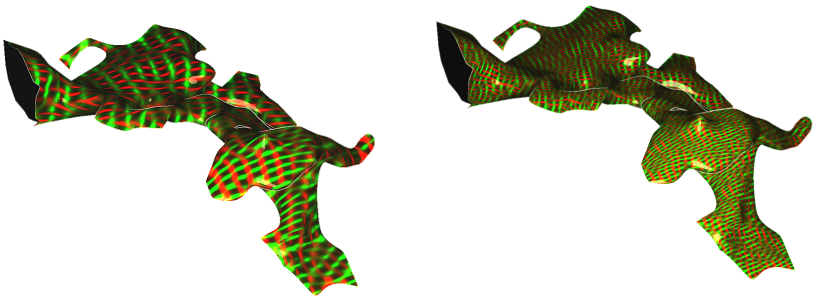

Fig. 3 Comparison of two different values of l to demonstrate the possibility for dynamic refinement of the input noise to achieve different levels of detail.

We start by implicitly splitting world space in voxels of equal size, filling the geometry's bounding box, i.e., we define a regular grid. Each voxel i is described by its base coordinate b_i and a constant edge length l. The seamless reaction diffusion texture is mapped to the surface of each of these voxels. To assign a texture coordinate to each vertex, the object space coordinate is transformed to the voxel space that is described by a minimum and maximum coordinate whose connecting line is the bounding box' diagonal. Points v_g on the geometry are transformed to v_{voxel} using

$$v_{voxel} = v_g \cdot \begin{pmatrix} l & 0 & 0 & -b_{min_x} \\ 0 & l & 0 & -b_{min_y} \\ 0 & 0 & l & -b_{min_z} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

and transformed into the local coordinate system by

$$v_{hit} = v_{voxel} - \lfloor v_{voxel} \rfloor. \tag{8}$$

The two texture coordinates are chosen to be those two components of v_{hit} that form the plane that is closest to the tangential plane of the surface in this point.

$$t = (v_{hit_i}, v_{hit_j}), \text{ with } i \neq j \neq k \wedge (n_k = max\{n_i, n_j, n_k\}). \tag{9}$$

In other words, this method transforms the world coordinate system to a system defined by one voxel, assuring that every component of every point v_{hit} is in $[0,1]$. The texture coordinate is determined by the surface's normal and, in particular, by the voxel side-plane whose normal is most similar to the surface's one (in terms of angle between them). The use of the term "voxel" is for illustration purpose only; those voxels are never created explicitly. As a result, the texture β contains the image space mapped input noise texture as shown in Figure 2. This texture is used as the initial pattern that is advected in the advection step.

Regardless of its simplicity, this method allows a continuous parameterization of the surface space that only introduces irrelevant distortions for mapping the noise

texture (cf. Figure 2). The mapping is continuous but not C^1-continuous, which is not required for mapping the noise texture as discontinuities in the first derivatives automatically vanish in the advection step.

Another positive aspect of this mapping is the possibility of a change of scale that is not available in the approaches of, e.g., Turk et al. [26]. By changing the size of voxels during the calculation, different frequencies of patterns can easily be produced and projected to the geometry. This capability allows one to change the resolution of the texture as required for automatic texture refinement when zooming. A comparison of two different levels of detail is shown in Figure 3.

2.4 Silhouette Detection

To avoid advection over geometric boundaries, a silhouette of the object is required to stop advection in these areas [19]. Otherwise, tensor advection would lead to a constant flow of "particles" across surface boundaries which makes the surface's geometry and topology unrecognizable.

A standard three-by-three Laplacian filter, defined by the convolution mask

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{10}$$

applied to the depth values followed by thresholding has proven to be suitable for our purposes. The silhouette image $e_{x,y}$ for each pixel (x,y) is then in the red color channel of its output texture.

2.5 Advection

We have discussed how to project the geometry and the corresponding tensor field to image space. With the prepared image space eigenvectors and the input noise texture, mapped to geometry, advection can be done. We use a simple Euler integration applied to both vector fields. With Euler's method some particle can be followed along a stream. In our case, we do not calculate streamlines at each position of both vector fields, as normally done in LIC. We directly advect the noise input texture with the given vector fields, which has the same result as locally filtering the data along pre-computed streamlines. This decision was based on the fact that massively parallel architectures like modern GPUs are able to perform this task in parallel for each pixel a hundred times per second. Formally, the advection step can be described as follows: First, we assume an input field P to be a continuous function, defined in a two-dimensional domain:

$$f_P : (x,y) \rightarrow p, \text{ with } x,y,p \in [0,1], \tag{11}$$

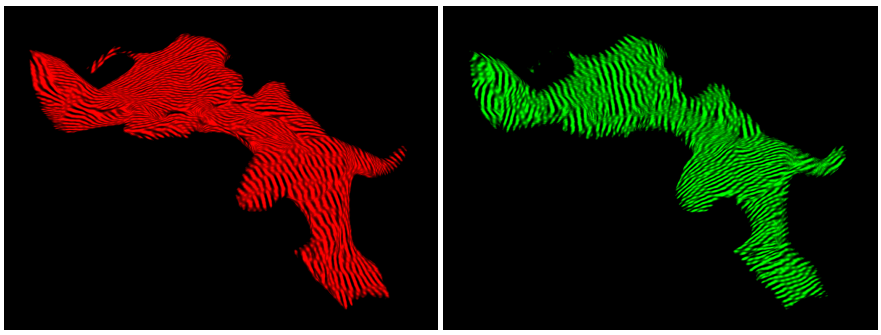

Fig. 4 Advection texture after ten iterations. Left: red channel containing $p_9^{\lambda_1}$, the advection image of eigenvector field 1; right: green channel containing $p_9^{\lambda_2}$, the advection image of eigenvector field 2.

i.e., it is a function returning the input value of the field P at a given position. Continuity is ensured with interpolating values in between. With this in mind, the iterative advection on each point (x,y) on the image plane can now be described by

$$\begin{aligned} &\forall x,y \in [0,1] : \forall \lambda \in \{\lambda_1,\lambda_2\} : \\ &p_0^{\lambda} = \beta_{x,y}, \\ &p_{i+1}^{\lambda} = k \cdot \beta_{x,y} + (1-k) \cdot \frac{f_{p_i^{\lambda}}((x,y)+v'_{\lambda}) + f_{p_i^{\lambda}}((x,y)-v'_{\lambda})}{2}. \end{aligned} \tag{12}$$

The iterative advection process has to be done for each eigenvector field separately with separate input and output fields p_i as can be seen in Figure 4. The value at a given point is a mix of the input noise and the iteratively advected input noise from the prior steps. Since the eigenvectors v'_{λ_j} do not have an orientation, the advection has to be done in both directions. The iteration can be stopped when the value change exceeds a threshold, i.e., if $|p_{i+1}^{\lambda} - p_i^{\lambda}| < \varepsilon$. We have chosen $\varepsilon = 0.05$, which is very conservative but ensures a proper rendering.

2.6 Compositing

The final rendering pass composes the temporary textures for final visualization. Whereas pixels that are not part of the original rendering of the geometry are discarded using the information from the depth buffer, for all other pixels the color values at a point (x,y) in image space after k iterations is defined by:

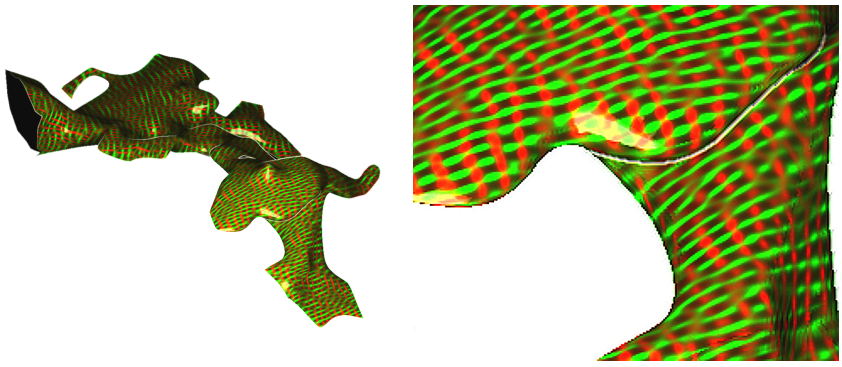

Fig. 5 The final image produced by the output processing shader with lighting. Left: the whole geometry. Right: a zoomed part of the geometry to show the fabric structure on the surface.

$$
\begin{aligned}
R &= \frac{r \cdot f_{p_k^{\lambda_2}}(x,y)}{8 \cdot f^2_{p_k^{\lambda_1}}(x,y)} + e_{x,y} + light(\mathscr{L}_{x,y}),\\
G &= \frac{(1-r) \cdot f_{p_k^{\lambda_1}}(x,y)}{8 \cdot f^2_{p_k^{\lambda_2}}(x,y)} + e_{x,y} + light(\mathscr{L}_{x,y}),\\
B &= e_{x,y} + light(\mathscr{L}_{x,y}),
\end{aligned}
\tag{13}
$$

where $p_k^{\lambda_1}$ and $p_k^{\lambda_2}$ are the fields generated from the eigenvector advection and e is the silhouette image. The scalar factor r is used to blend between the two chosen tensor directions. Equation 13 is a weighting function, which weights This approach creates a mesh resembling the tensor field's structure. To reduce the effect of light sources on the color coding, we use a separate lighting function *light* that, while manipulating the intensity, does not affect the base color of the mesh structure. Even though Blinn-Phong shading [2] has proven to provide the required depth cues, additional emphasis of the third dimension using depth-enhancing color coding has proven to provide a better overall understanding of the data [4]. Finally, further filters can be applied on the composed image, like contrast enhancement or sharpening filters common to vector field LIC [27, 9]. Figure 5 shows the result of Equation 13 combined with Blinn-Phong shading and an applied sharpening filter.

2.7 Implementation

Our implementation is not limited to a special kind of geometry. It is able to handle almost every tensor field defined on a surface. It is, for example, possible to calculate an isosurface on a derived scalar metric, like fractional anisotropy or on a second data set to generate a surface in a three-dimensional data domain. Other methods

include hyper-stream surfaces [5], wrapped streamlines [8], or domain-dependent methods like dissection-like surfaces presented in [1]. The only requirement for the surface is that it is non-selfintersecting and smooth normals are provided as they are required for the projection step and for proper lighting. The noise texture can be pre-calculated in a pre-processing step or stored in a file as it is independent of the data.

Our implementation is basing on an multipath rendering approach, where each of the four processing steps in Figure 1 is implemented as shader program and rendered one after the other utilizing frame-buffer-objects (FBO) and textures for data transfer.

The first step projects the geometry into image space, simply by rendering the geometry and pre-calculating the Phong light intensity $\mathscr{L}_{x,y} \in [0,1]$ at every rendered fragment with the coordinates x and y. In the same step, the tensors are projected as well using the Equations 1 to 5. Tensor projection is done fragment-wise as interpolation of vertex-wise calculated and projected eigensystems causes problems. Please note, that the texture with the projected eigenvectors needs to be initialized in black, as the RGBA-quadruple $(0,0,0,0)$ denotes both eigenvectors to be of length 0 and can be used as abort criterion during advection. The precalculated two-dimensional noise texture (c.f. Section 2.2) is also mapped to each fragment during fragment processing using the Equations 7 to 9. As the projection step is done using an FBO, the resulting values can be written to multiple textures, which then can be used as input textures for the consecutive steps.

Followed by the iterative advection of the mapped noise in image-space, the silhouette detection step applies a Laplacian convolution kernel fragment-wise to the depth buffer to provide the needed border information to the advection step (c.f. Section 2.4). The input of the advection step is the noise, mapped on the geometry. Figure 2, right shows this. As Equation 12 shows, the advection uses the previously advected texture which, in the first iteration, is the geometry-mapped noise and advects it in direction of the eigenvector field. The resulting texture is then combined with the original geometry-mapped noise. During each render loop of the complete scene, we apply advection three times. The resulting texture is then used as input for the advection step during the next render pass. Figure 4 shows these advected images, separate for each of the eigenvector fields.

The advected images are finally composed to the final image shown on the screen, showing the fabric-like structure in combination with color-mapping. Our compositing is mainly the combination of both advected noise textures according to Equation 13.

3 Results

We have introduced a method to create a fabric-like surface tensor LIC in image space, similar to the one introduced in [13]. We used ideas from [19] to transform the algorithm into image space. Our implementation, using this method, is able to

reach frame rates high enough for real-time user interaction. The only bottleneck is the hardware's ability in rendering large and triangle-rich geometry. All further steps can be done in constant time, see Table 1.

3.1 Artificial Test Data Sets

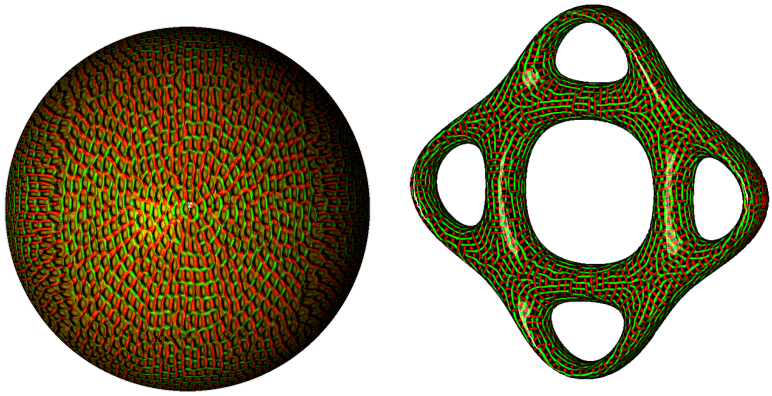

Fig. 6 Analytic test data sets. We applied our method to isosurfaces and the scalar field's Hessian matrix, showing the curvature on the surface, to demonstrate the suitability for topologically more complex surfaces. Shown here are the final images using our method for a sphere and Bretzel5 data set (Equation 14). The image from the sphere data set has been improved by applying a further bump-mapping filter as introduced in [7]. The eigenvalues and eigenvectors of the Hessian matrix denote the change of the normal on the surface, which corresponds to the gradient in the scalar field.

We first applied out method to artificial test data sets with different complex topologies. The Bretzel5 data set shown here is defined implicitly:

$$((x^2 + .25 * y^2 - 1) * (.25 * x^2 + y^2 - 1))^2 + z^2 - 0.1 = 0. \tag{14}$$

We used the Hessian matrix of the corresponding scalar fields on the surfaces as tensor fields which, in fact, describe the curvature. The results displayed in Figure 6 show that neither the topology nor our artificial parameterization of the input noise texture influences the quality of the final rendering. In the center of the sphere in Figure 6, a degenerate point can be seen. On this point, the first and second eigenvalue are both zero. Our method can handle these points properly.

3.2 Modification for Medical Data Processing

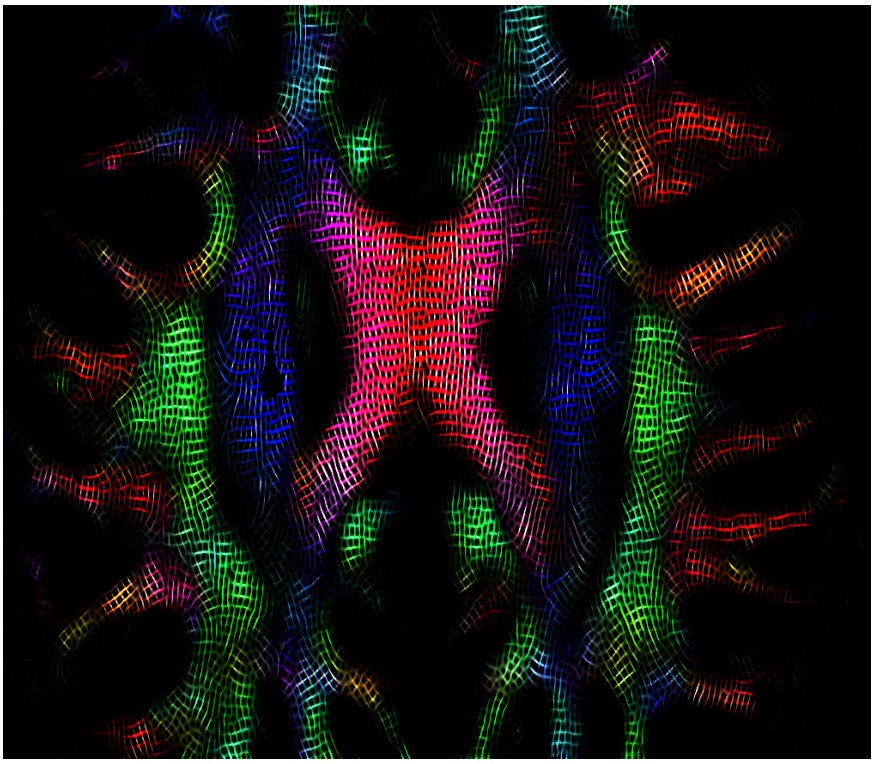

Fig. 7 An axial slice through a human brain: Corpus callosum (CC) (red), pyramidal tract (blue), and parts of the cinguli (green in front and behind the CC) are visible. The main direction in three-dimensional space is indicated by the RGB color map, where red indicates the lateral (left–right), green anterior–posterior, and blue superior–inferior direction. The left–right structure of the CC can clearly be seen in its center, whereas color and pattern indicate uncertainty towards the outer parts. The same is true for the cinguli's anterior–posterior structure. As seen from the blue color, the pyramidal tract is almost perpendicular to the chosen plane and, therefore, secondary and ternary eigenvectors dominate the visualization. Alternatively, we could easily fade out those out-of-plane structures in cases where they distract the user.

Even though many higher-order methods have been proposed, due to scanner, time, and cost limitations, second-order tensor data is still dominant in clinical application. Medical second-order diffusion tensor data sets differ from engineering data sets because they indicate one major direction whereas the secondary and ternary directions only provide information in areas where the major direction is not well-defined, i.e., the fractional anisotropy—a measure for the tensor shape—is low. Almost spherical tensors, which indicate isotropic diffusion, occur in areas where multiple fiber bundles traverse a single voxel of the measurement or when

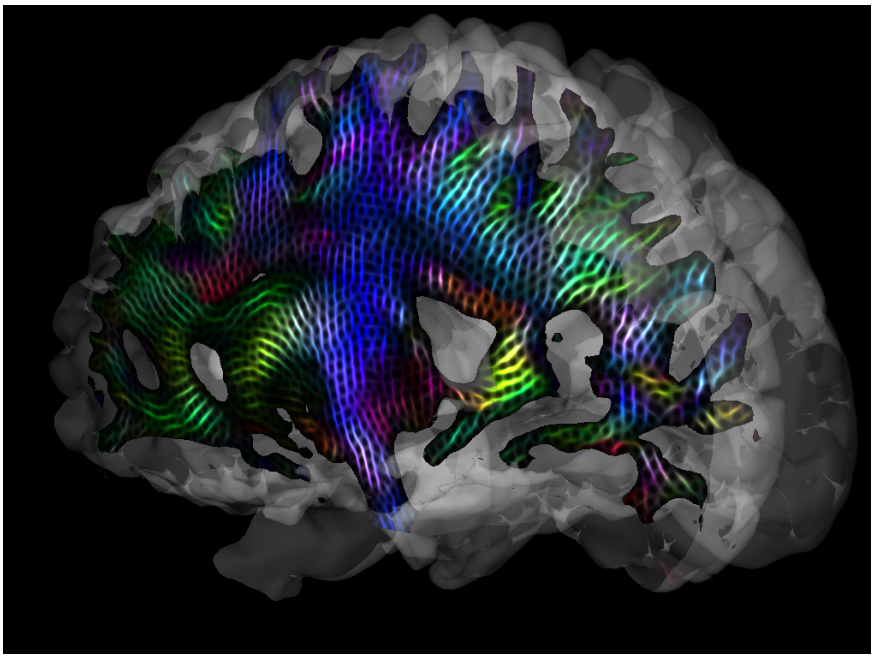

Fig. 8 Diffusion tensor data set of a human brain. We employed the method by Anwander et al. [1] to extract a surface following neural fibers and applied our method with an alternative color coding that is more suitable and can be incorporated more easily into medical visualization tools.

no directional structures are present. Therefore, we modulate the color coding using additional information: In areas where one fiber direction dominates, we only display this major direction using the standard color coding for medical data sets, where x, y, and z alignment are displayed in red, green, and blue, respectively. In areas where a secondary direction in the plane exists, we display this information as well but omit the secondary color coding and display the secondary direction in gray-scale rendering mode and always below the primary direction (cf. Fig. 8). We use the method of Anwander et al. [1] to extract surfaces that are, where possible, tangential to the fiber directions. Hence, we can guarantee that the projection error introduced by using our method in the surface's domain remains sufficiently small, which can be seen in Figure 9. Even in areas where the fractional anisotropy is low and the color coding does no longer provide directional information, such as in some parts of the pyramidal tract in Fig. 8, the texture pattern still provides this information.

The applicability of our method, especially to medical second-order tensor data, is mainly due to its real-time ability and its ability to provide a smooth transition between local and global structures. Neuroscientist researcher can explore tensor data interactively, by using slices inside the volume or by calculating surfaces in interesting regions.

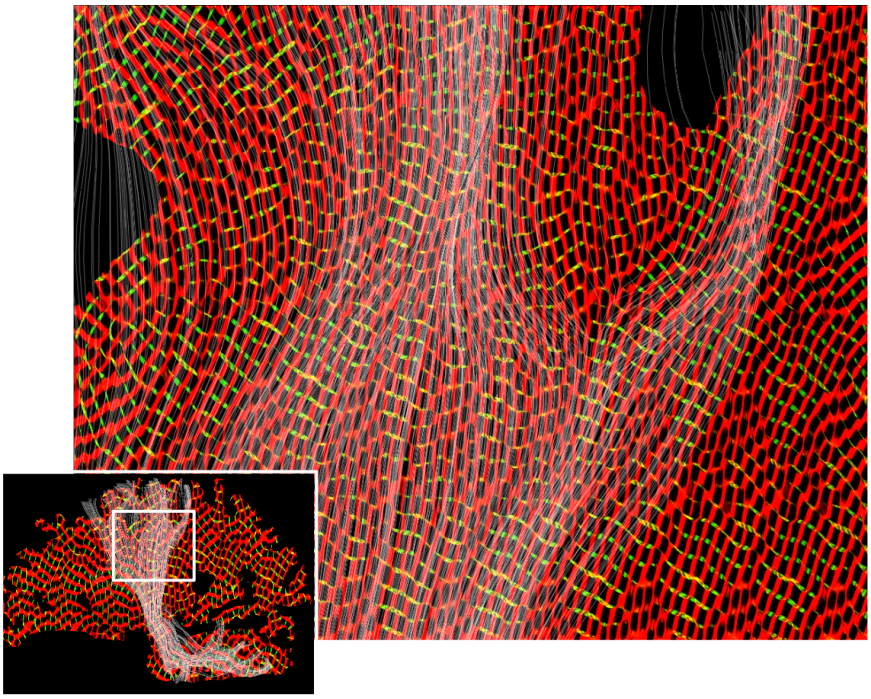

Fig. 9 For validation, we calculated the fiber-tracts near the used surface. The directions can now be compared. It shows that the major eigenvector direction correlates with the direction of the fiber-tracts. As the used slice is near to the pyramidal tract, there is a strong diffusion in axial direction. But not all the shown fiber-tracts match the direction of the shown surface tensor, which is caused by not having all fiber-tracts directly on the surface. Most of them are slightly in front of it and therefore can have other local directions.

3.3 Performance

As indicated before, the only "bottleneck" in the visualization pipeline that is strongly geometry-dependent is the projection step. Since the surface needs to be rendered repeatedly in case of user interaction, the performance measures of our method consider repeated rendering of the geometry. The frame rate with geometry not being moved and, therefore, making the projection step and the edge detection step unnecessary, is considerably higher. Our implementation requires only few advection iterations per frame, which ensures high frame rates and smooth interaction. To make the frame rates comparable, in the following tables, user interaction is assumed and, therefore, rendering a single frame always consists of

- one projection step, including geometry rendering;
- one edge detection pass;
- three advection iterations; and
- one output processing pass.

As seen in the previous sections, fragments not belonging to the geometry are discarded as soon as possible without using deferred shading. This also generates performance gain in advection and output processing. In Table 1, a selection of data sets with their corresponding number of triangles and tensors are listed. These measurements where taken on a 1024×768 viewport with about $\frac{3}{4}$ of the pixels covered by the surface.

Figure	Nb Triangles	Nb Tensors	fps	fps (Phong only)	∅ Geometry Share
8	41472	63075	32	61	72%
5	58624	88803	30	60	69%
7	571776	861981	14	16	90%

Table 1 Frames per second (fps) for different data sets with given number of triangles and numbers of tensors. The frame rates are compared to simple rendering of the geometry using Phong shading. The frame rates were obtained for an AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (512K L2 Cache) with an NVIDIA G80 GPU (GeForce 8800 GTS) and 640MB of graphics memory at a resolution of 1024×768 pixels. The geometry share relates the time used by the GPU to rasterize the geometry to the overall rendering time, which contains all steps of the pipeline. The time used to render the geometry clearly dominates the rendering times and reaches up to 90 percent of the overall rendering time even for medium-sized geometries.

The assumption that geometry rendering with projection is the weakest component in this pipeline and that edge detection, advection, and output processing perform at a data-independent frame rate is confirmed by the frame rates shown in Table 1. It confirms that for large geometries, rendering the geometry alone is the dominating component. Since the vertex-wise calculations during projection are limited to tensor projection (Equation 1) and vertex projection (Equation 7), the most expensive calculations during projection are executed per fragment. This means that the expensive eigenvalue decomposition and eigenvector calculations are only required for fragments (pixels) actually visible on the screen. Independent of the geometry's complexity, the number of fragments that require the costly processing steps reaches a saturation value and therefore does not require further processing time even if the geometry complexity increases. Table 1 states that this saturation point has been reached in Figure 7. The computation for per-fragment Phong lighting consumes much more time as it is calculated before the invisible fragments have been discarded, which explains the nearly equal framerates. To further decouple the calculation effort from the geometry's size, the depth test should be performed before performing the eigendecomposition. This goal can be achieved by first rendering the projected tensors to a texture, and computing the decomposition on visible fragments only. Nevertheless, this is not necessary for our current data set and screen sizes where the time required to render the geometry itself clearly dominates the time required to compute the texture pattern in image space. This can be seen in the increasing values in Table 1 with increasing size of vertices rendered.

4 Conclusions and Possible Directions for Future Research

We have presented a novel method for rendering fabric-like structures to visualize tensor fields on almost arbitrary surfaces without generating three-dimensional textures that span the whole data set at sub-voxel resolution. Therefore, our method can be applied to complex data sets without introducing texture memory problems common to methods relying on tree-dimensional noise textures. As major parts of the calculation are performed in image space, the performance of our algorithm is almost independent of data set size, provided that surfaces can be drawn efficiently, e.g., by using acceleration structures to draw only those parts of the geometry that intersect the view frustum or using ray tracing methods.

Whether the surface itself is the domain of the data, a surface defined on the tensor information (e.g., hyper stream surfaces), or a surface defined by other unrelated quantities (e.g., given by material boundaries in engineering data or anatomical structures in medical data) is independent from our approach. Nevertheless, the surface has to be chosen appropriately because only in-plane information is visualized. To circumvent this limitation, information perpendicular to the plane could be incorporated in the color coding, but due to a proper selection of the plane that is aligned with our features of interest, this has not been necessary for our purposes.

Especially in medical visualization, higher-order tensor information is becoming increasingly important and different methods exist to visualize these tensors, including local color coding, glyphs, and integral lines. Nevertheless, an extension of our approach is one of our major aims. In brain imaging, experts agree that the maximum number of possible fiber directions is limited. Typically, a maximum of three or four directions in a single voxel are assumed (cf., [22]). Whereas the number of output textures can easily be adapted, the major remaining problem is a lack of suitable decomposition algorithms on the GPU. Image-space techniques, by their very nature, resample the data and, therefore, require one to use such proper interpolation schemes. In addition, maintaining orientations and assigning same fibers in higher-order data to the same texture globally is not possible today and, therefore, is a potential topic for further investigation.

Acknowledgements

We thank Alfred Anwander and Thomas R. Knösche from Max Planck Insitute for Human Cognitive and Brain Sciences, Leipzig, Germany, for providing the human brain image data sets, and for fruitful discussions and comments. We thank the members of the Visualization and Computer Graphics Research Group of the Institute for Data Analysis and Visualization, Department of Computer Science, UC Davis, and the members of the Abteilung für Bild- und Signalverarbeitung des Instituts für Informatik der Universität Leipzig, Germany.
Mario Hlawitschka was supported by NSF grant CCF-0702817.

References

1. Anwander, A., Schurade, R., Hlawitschka, M., Scheuermann, G., Knösche, T.: White matter imaging with virtual klingler dissection. NeuroImage **47**(Supplement 1), S105 – S105 (2009). DOI 10.1016/S1053-8119(09)70916-4. Organization for Human Brain Mapping 2009 Annual Meeting
2. Blinn, J.F.: Models of light reflection for computer synthesized pictures. In: SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques, pp. 192–198. ACM, New York, NY, USA (1977). DOI http://doi.acm.org/10.1145/563858.563893
3. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 263–270. ACM, New York, NY, USA (1993). DOI http://doi.acm.org/10.1145/166117.166151
4. Chu, A., Chan, W.Y., Guo, J., Pang, W.M., Heng, P.A.: Perception-aware depth cueing for illustrative vascular visualization. In: BMEI '08: Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics, pp. 341–346. IEEE Computer Society, Washington, DC, USA (2008). DOI http://dx.doi.org/10.1109/BMEI.2008.347
5. Delmarcelle, T., Hesselink, L.: Visualization of second order tensor fields and matrix data. In: VIS '92: Proceedings of the 3rd conference on Visualization '92, pp. 316–323. IEEE Computer Society Press, Los Alamitos, CA, USA (1992)
6. Dick, C., Georgii, J., Burgkart, R., Westermann, R.: Stress tensor field visualization for implant planning in orthopedics. IEEE Transactions on Visualization and Computer Graphics **15**(6), 1399–1406 (2009). DOI http://doi.ieeecomputersociety.org/10.1109/TVCG.2009.184
7. Eichelbaum, S., Hlawitschka, M., Hamann, B., Scheuermann, G.: Fabric-like visualization of tensor field data on arbitrary surfaces in image space. Submitted to Dagstuhl Seminar 09302
8. Enders, F., Sauber, N., Merhof, D., Hastreiter, P., Nimsky, C., Stamminger, M.: Visualization of white matter tracts with wrapped streamlines. In: C.T. Silva, E. Gröller, H. Rushmeier (eds.) Proceedings of IEEE Visualization 2005, pp. 51–58. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA (2005)
9. Grabner, M., Laramee, R.S.: Image space advection on graphics hardware. In: SCCG '05: Proceedings of the 21st spring conference on Computer graphics, pp. 77–84. ACM, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1090122.1090136
10. Hasan, K.M., Basser, P.J., Parker, D.L., Alexander, A.L.: Analytical computation of the eigenvalues and eigenvectors in DT-MRI. Journal of Magnetic Resonance **152**(1), 41 – 47 (2001). DOI 10.1006/jmre.2001.2400
11. Hesselink, L., Levy, Y., Lavin, Y.: The topology of symmetric, second-order 3d tensor fields. IEEE Transactions on Visualization and Computer Graphics **3**(1), 1–11 (1997). DOI http://dx.doi.org/10.1109/2945.582332
12. Hlawitschka, M., Garth, C., Tricoche, X., Kindlmann, G., Scheuermann, G., Joy, K.I., Hamann, B.: Direct visualization of fiber information by coherence. International Journal of Computer Assisted Radiology and Surgery, CARS, CUARC.08 Special Issue (2009)
13. Hotz, I., Feng, L., Hagen, H., Hamann, B., Joy, K., Jeremic, B.: Physically based methods for tensor field visualization. In: VIS '04: Proceedings of the conference on Visualization '04, pp. 123–130. IEEE Computer Society, Washington, DC, USA (2004). DOI http://dx.doi.org/10.1109/VIS.2004.80
14. Hotz, I., Feng, Z.X., Hamann, B., Joy, K.I.: Tensor field visualization using a fabric-like texture on arbitrary two-dimensional surfaces. In: T. Möller, B. Hamann, R.D. Russel (eds.) Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration. Springer-Verlag Heidelberg, Germany (2009)
15. Iwakiri, Y., Omori, Y., Kanko, T.: Practical texture mapping on free-form surfaces. In: PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications, p. 97. IEEE Computer Society, Washington, DC, USA (2000)

16. Kindlmann, G.: Superquadric tensor glyphs. In: Proceedings of IEEE TCVG/EG Symposium on Visualization 2004, pp. 147–154 (2004)
17. Kindlmann, G., Tricoche, X., Westin, C.F.: Anisotropy creases delineate white matter structure in diffusion tensor MRI. In: Ninth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'06), Lecture Notes in Computer Science 4190, pp. 126–133. Copenhagen, Denmark (2006)
18. Knoll, A., Hijazi, Y., Hansen, C., Wald, I., Hagen, H.: Interactive ray tracing of arbitrary implicits with simd interval arithmetic. In: RT '07: Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing, pp. 11–18. IEEE Computer Society, Washington, DC, USA (2007). DOI http://dx.doi.org/10.1109/RT.2007.4342585
19. Laramee, R.S., Jobard, B., Hauser, H.: Image space based visualization of unsteady flow on surfaces. In: VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), p. 18. IEEE Computer Society, Washington, DC, USA (2003). DOI http://dx.doi.org/10.1109/VISUAL.2003.1250364
20. Laramee, R.S., van Wijk, J.J., Jobard, B., Hauser, H.: Isa and ibfvs: Image space-based visualization of flow on surfaces. IEEE Transactions on Visualization and Computer Graphics **10**, 637–648 (2004). DOI http://doi.ieeecomputersociety.org/10.1109/TVCG.2004.47
21. Purnomo, B., Cohen, J.D., Kumar, S.: Seamless texture atlases. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 65–74. ACM, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1057432.1057441
22. Schultz, T., Seidel, H.P.: Estimating crossing fibers: A tensor decomposition approach. IEEE Transactions on Visualization and Computer Graphics **14**(6), 1635–1642 (2008). DOI http://doi.ieeecomputersociety.org/10.1109/TVCG.2008.128
23. Tricoche, X.: Vector and tensor field topology simplification, tracking, and visualization. Ph.D. thesis, University of Kaiserslautern, Germany (2002)
24. Tricoche, X., Scheuermann, G., Hagen, H.: Tensor topology tracking: A visualization method for time-dependent 2D symmetric tensor fields. In: Eurographics 2001 Proceedings, Computer Graphics Forum 20(3), pp. 461–470. The Eurographics Association, Saarbrücken, Germany (2001). DOI http://dx.doi.org/10.1111/1467-8659.00539
25. Turing, A.: The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society of London **237**(641), 37 – 72 (1952)
26. Turk, G.: Generating textures on arbitrary surfaces using reaction-diffusion. In: SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques, pp. 289–298. ACM, New York, NY, USA (1991). DOI http://doi.acm.org/10.1145/122718.122749
27. Weiskopf, D., Ertl, T.: A hybrid physical/device-space approach for spatio-temporally coherent interactive texture advection on curved surfaces. In: GI '04: Proceedings of Graphics Interface 2004, pp. 263–270. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (2004)
28. van Wijk, J.J.: Image based flow visualization. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 745–754. ACM, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/566570.566646
29. van Wijk, J.J.: Image based flow visualization for curved surfaces. In: VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), p. 17. IEEE Computer Society, Washington, DC, USA (2003). DOI http://dx.doi.org/10.1109/VISUAL.2003.1250363
30. Zhang, E., Yeh, H., Lin, Z., Laramee, R.S.: Asymmetric tensor analysis for flow visualization. IEEE Transactions on Visualization and Computer Graphics **15**, 106–122 (2009). DOI http://doi.ieeecomputersociety.org/10.1109/TVCG.2008.68
31. Zheng, X., Pang, A.: Hyperlic. In: VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), p. 33. IEEE Computer Society, Washington, DC, USA (2003). DOI http://dx.doi.org/10.1109/VISUAL.2003.1250379