EG UK Theory and Practice of Computer Graphics (2006)
M. McDerby, L. Lever (Editors)

HyperStreamball Visualization for Symmetric Second Order Tensor Fields

J. Liu, M. Turner, W.T. Hewitt and J.S. Perrin

liuj@cs.man.ac.uk and {mt, w.t.hewitt, james.s.perrin}@manchester.ac.uk
Manchester Computing
The University of Manchester, United Kingdom

Abstract

This paper proposes a new 3D tensor glyph called a hyperstreamball that extends streamball visualization used within fluid flow fields to applications within second order tensor fields. The hyperstreamball is a hybrid of the ellipsoid, hyperstreamline and hyperstreamsurface. With the proposed system a user can easily interactively change the visualization. First, we define the distance of the influence function which contributes a potential field that can be designed to highlight the three eigenvectors and eigenvalues of a real symmetric tensor at any sample point. Second, we discuss the choice of source position and how the user can control the parameter mapping between the field data and the implicit function. Finally, we test our results using both synthetic and real data that shows the hyperstreamball's two main advantages: one is that hyperstreamballs blend and split with each other automatically depending on the tensor data, and the other advantage is that the user can achieve both discrete and continuous representation of the data based on a single geometrical description.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations

1. Introduction

Tensor fields, especially second-order tensor fields, are useful in many applications such as: fluid dynamics, solid mechanics, geophysics, and many applications within the earth sciences. Engineers and scientists have a great need for effective visualization methods to help them understand and explore their tensor data sets.

The main goal of this work is to develop a simple yet effective representation of any 3D real symmetric tensor field. Motivated by the simple ellipsoid representation of symmetric tensor data and the streamball visualization [BHKD94] used within a vector field, we present a novel glyph called a hyperstreamball for the visualization of symmetric second order tensor fields such as stress, strain and the symmetric part of any general tensor. One advantage of our method is that it provides the user both continuous and discrete representation of the tensor field using metaballs [Bli82] to represent the status of the tensors. For example, if we distributed metaballs along a stress trajectory, they will blend with each other or split depending on the magnitude of the three eigenvalues of the second order tensor. This overcomes the clutter problem when using ellipsoids as they blend together to form a 3D hyperstreamline [DH92]. When compared to hyperstreamline visualization, in which three hyperstreamlines must be integrated along the three eigenvectors respectively, in order to see the three eigenvectors direction, the hyperstreamball can be easily changed back to the discrete case, and then the three eigenvectors directions can be simultaneously seen from the metaball orientation. Furthermore, cluttered hyperstreamlines can form a hyperstreamsurface to give an intuitive representation of the whole tensor field. Another advantage of the hyperstreamball is its ability to blend and split where the tensor field converges or diverges.

The rest of this paper is organized as follows: in Section 2, some essential facts about tensor fields used in the proposed visualization are summarized, in Section 3 the relevant previous work in tensor field visualization using glyphs are reviewed, Section 4 first reviews the generation of a metaball

and streamball visualization within a vector field, and then explains the details of designing a hyperstreamball used to simultaneously visualize the three eigenvectors and the three eigenvalues of the tensor data. Some implementing issues of hyperstreamball are also discussed. Section 5 discusses and evaluates the results based on both synthetic and real data, and final conclusions are drawn in Section 6.

2. Tensor Basics

A tensor is a generalization of the concept of scalar, vector and linear operator in a way that is independent of any chosen coordinate system. In this paper, our focus is applied to the second-order tensor which appears in many physical and engineering applications. A second-order tensor can be represented by a 3×3 matrix relative to a fixed coordinate system, given by nine independent scalars:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \tag{1}$$

The tensor $\mathbf{T}$ is called symmetric if for any coordinate basis, $t_{ij} = t_{ji}$ for $i, j = 1, ..., n$ and it is called antisymmetric if $t_{ij} = -t_{ji}$ for $i, j = 1, ..., n$. For a general tensor $\mathbf{T}$, it can be decomposed to a symmetric part $\mathbf{S}$ and an antisymmetric part $\mathbf{A}$:

$$\mathbf{T} = \mathbf{S} + \mathbf{A} = \frac{1}{2}(\mathbf{T} + \mathbf{T}^t) + \frac{1}{2}(\mathbf{T} - \mathbf{T}^t) \tag{2}$$

The symmetric part of the tensor $\mathbf{S}$ is defined by six independent scalars within a symmetric matrix

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{12} & s_{22} & s_{23} \\ s_{13} & s_{23} & s_{33} \end{bmatrix} \tag{3}$$

and the definition of antisymmetric matrix implies:

$$\mathbf{A} = \begin{bmatrix} 0 & -a_{12} & -a_{13} \\ a_{12} & 0 & -a_{23} \\ a_{13} & a_{23} & 0 \end{bmatrix} \tag{4}$$

which is equivalent to a vector $\mathbf{a} = (a_{12}, a_{13}, a_{23})$. Thus any matrix can be decomposed to a vector $\mathbf{a}$ and a symmetric matrix $\mathbf{S}$. For a symmetric matrix, there always exists an alternative basis, also called eigenvectors, in which $\mathbf{S}$ can be diagonalized and the diagonal components are the eigenvalues. If these basis represented by $\hat{\mathbf{v}}^{(i)}$ are the column vector of matrix $\mathbf{V}$:

$$\begin{bmatrix} \hat{\mathbf{v}}^{(1)} & \hat{\mathbf{v}}^{(2)} & \hat{\mathbf{v}}^{(3)} \end{bmatrix} \tag{5}$$

and the diagonal components λ_i, i.e., eigenvalues, $i = 1, 2, 3$ form the matrix Σ:

$$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{6}$$

then the original symmetric matrix $\mathbf{S}$ can be equally represented by the following:

$$\mathbf{S} = \mathbf{V}\Sigma\mathbf{V}^{\mathbf{T}} \tag{7}$$

The eigenvalues are ordered so that $\lambda_1 \geq \lambda_2 \geq \lambda_3$; the corresponding ortho-normalized eigenvectors $\hat{\mathbf{v}}^{(1)}$ and $\hat{\mathbf{v}}^{(2)}$ and $\hat{\mathbf{v}}^{(3)}$ are respectively called major, medium and minor eigenvectors. Eigenvalues and eigenvectors have profoundly useful physical meanings depending on the symmetric tensor $\mathbf{S}$. For example, if $\mathbf{S}$ is a stress tensor, then the three eigenvectors are perpendicular to three planes where shear stresses are zero, and they are then called principal stresses.

3. Related Work

In this section, we will mainly look at glyph visualizations of tensor fields which are related to our work, but like topology analysis of tensor fields [DH94] [HLL97] [ZP04] [ZPP05] which provides the global structure of a tensor field, and volume renders [KW99] [KWH00] that are mainly for diffusion tensor fields from MRI will not be described here.

The most direct representation of a tensor field is by using glyphs. The simplest one is the Lame stress ellipsoid [F65], which is rotated along the three eigenvectors of the tensor and the three radii are scaled according to the corresponding eigenvalues. The Haber glyph [Hab90] used a modified ellipsoid with a bar drawn along the principal eigenvector and an elliptical disk wrapping around the bar to represent the other two eigenvector directions which helps to clearly show the direction of eigenvectors. The Reynolds glyph [MSM95] is used to visualize the normal stress in any direction from the origin of surface to the any point of surface. A detailed comparison about these glyphs can be found in [YMAHW03]. One common problem with all these glyphs is the clutter problem in 3D space. This can be alleviated by using a glyph interactively in the field, for example, W. de Leeuw et al.'s probe [dLvW93]. This can be interactively moved through the model and changes to reflect physical quantities such as acceleration, curvature, torsion etc. extracted from the gradient tensor. Most recently, the paper-and-pencil graphical method—mohr diagram—is applied to visualize geologic stress interactively [CRB*05]. One main advantage is that it can be applied to negative eigenvalues which provides information like compressive or tensile force.

All of the above glyphs are located at discrete sample points and therefore a second problem is that it is difficult to perceive continuous changes within a tensor field. Delmarcelle and Hesselink's [DH92] hyperstreamline glyph visualizes the tensor data along a line domain, which is based on a tensor line [Dic89]. This is a curve integrated through the tensor field along one of the three eigenvectors, and the cross section of this curve is further divided into ellipses which are rotated and scaled according to transverse eigenvalues and eigenvectors or helix whose arms are proportional to the

transverse eigenvalues. In order to overcome the ambiguity at places where the tensors' two or even all three eigenvalues are nearly equal, tensorlines [WKL99] were introduced by Weinstein et al. Although hyperstreamlines are good at representing how tensors change continuously along the integration curve, they are critically dependent on the start points of integration and many hyperstreamlines can easily again form clutter. Based on the hyperstreamline, Jeremic et al [JSF*02]. proposed to use a hyperstreamsurface, which used polygons to connect hyperstreamlines to generate a surface to visualize the stress tensors in the field of geomechanics.

In order to overcome the clutter problems resulting from ellipsoids and hyperstreamlines, and also keep the benefits of them, we propose hyperstreamball visualization, which can easily change between the ellipsoid, the hyperstreamline and the hyperstreamsurface representation.

4. The HyperStreamball Glyph

4.1. Streamball in Fluid Flow Visualization

In 1994, Brill et al. [BHKD94] proposed streamball visualization for a vector field, which is distinguished by its ability to split or merge with each other wherever divergence or convergence happens. With discrete streamballs, the positions of particles in the flow are used as the center points for this implicit surface and the magnitude of velocity can be either mapped to the radius or colour of implicit surface, which then are blended with each other to form three-dimensional streamlines, stream surfaces etc. For example, in Figure 1, evenly spaced sample points are chosen along the streamline of a double glazing data set, the velocity magnitude is mapped to the radius of the implicit function. Where streamballs are blended with each other to form a continuous surface this indicates a high velocity area, on the other hand a discrete set of streamballs indicates a low velocity area. The streamballs in this case are colored by the temperature values.

The basic idea behind the streamball visualization is to generate a potential field $F(x,y,z)$ i.e. an implicit surface, and then take the isocontours of this implicit surface $F(x,y,z) = C$, where C is the isovalue. This implicit surface is created by placing a collection of field sources s_i in space and computing a field value at each point of space. The field value F is the sum of the weighted influence function I_i contributed by each source. It is mathematically represented as the following:

$$F(x,y,z,S) = \sum_i w_i I_i \tag{8}$$

where S is the collection of sources s_i, and w_i are the weights.

This was proposed by Blinn [Bli82] in 1982, when he defined the influence function as the electron in an atom which was represented by a density function of the spatial location

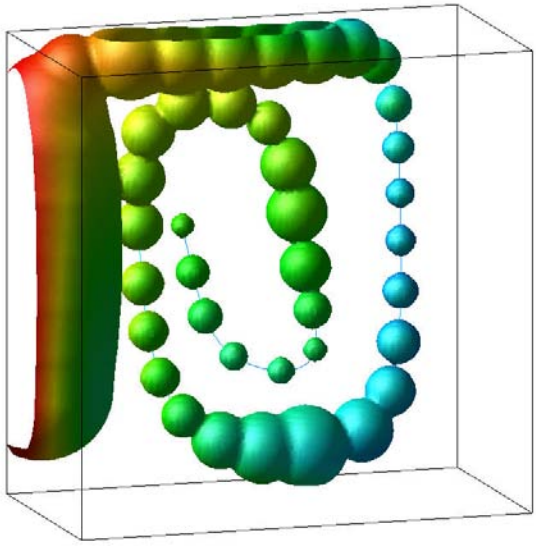

Figure 1: *The streamball visualization of the double glazing data set*

in quantum mechanics.

$$I_i = e^{-a_i f_i(\mathbf{x})} \tag{9}$$

where $f_i(\mathbf{x})$ is the distance of the field point to a source (an atom): $\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2}$, and a_i is the size. If there is just one source, then the resulting isosurface will be a sphere which is commonly known as a metaball or blobby object. In the rest of this paper, these sources refer to metaballs. G. Wyvill et al. [WMW86] modified the influence function to avoid the computation of the exponential function, and applied the following polynomial approximation of the influence function $I_i(x)$ as:

$$I_i(\mathbf{x}) = \begin{cases} a\frac{f_i(\mathbf{x})^6}{R^6} + b\frac{f_i(\mathbf{x})^4}{R^4} + c\frac{f_i(\mathbf{x})^2}{R^2} + 1 & f_i(\mathbf{x}) \le R \\ 0 & f_i(\mathbf{x}) > R \end{cases} \tag{10}$$

$f_i(\mathbf{x})$ here is the same as the one in equation 9, R is the radius of the metaball which any scalar can map to and a,b,c are constants satisfying the following conditions:

$$I_i(0) = 1 \qquad I_i(0.5) = 0.5 \qquad I_i(R) = 0$$
$$I_i'(0) = 0 \qquad I_i'(R) = 0$$

4.2. The Design of Hyperstreamball

Brill et al. [BHKD94] in their future work suggested that the streamball could be used to visualize the 3D tensor field, however they did not explore it further. There are two main problems and the first is how to choose the sources' positions. In the vector field visualization, the streamball can be distributed on a streamline, streakline or pathline at different time steps, which can better represent the fluid flow

direction, but the key question is where shall we put the sources' positions for the tensor field? The second problem is in streamball visualization, for a velocity magnitude only one scalar needs to be visualized which can be mapped to R in Equation10 shown in Figure 1, the direction of the velocity is then shown as the 3D streamline formed by the blended metaballs. In the tensor field, we are interested in visualizing all three eigenvectors and all three eigenvalues of the tensor data. So how do we represent three scalars and three eigenvectors using metaballs?

Let us have a look at the second problem first, which can be solved by modifying the distance function $f_i(\mathbf{x})$ in Equation 10. Actually this distance function defines the shape of the metaball, you can also state that it defines the region where a source radiates energy.

For the vector field visualization the distance function is defined as

$$f_i(\mathbf{x})^2 = (x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2$$

where (x_i, y_i, z_i) is the position of the source, which results in one metaball shown as the sphere shape (see Figure 1). The resulting implicit function is a blending of different spheres. If we define the distance function as:

$$f_i(\mathbf{x})^2 = \frac{(x-x_i)^2}{r_1^2} + \frac{(y-y_i)^2}{r_2^2} + \frac{(z-z_i)^2}{r_3^2} \qquad (11)$$

i.e., an ellipsoid where (x_i, y_i, z_i) is still the center, then similarly to the ellipsoid visualization of tensor data, three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ can be mapped to r_1, r_2 and r_3, and the three normalized eigenvectors can form a rotation matrix applied to the distance function so that the generated metaball will align with the three eigenvectors. For example, in Figure 2, a number of ellipsoid shape metaballs along a curve will blend with each other to form a continuous three-dimensional surface as we increase the number of the sources.

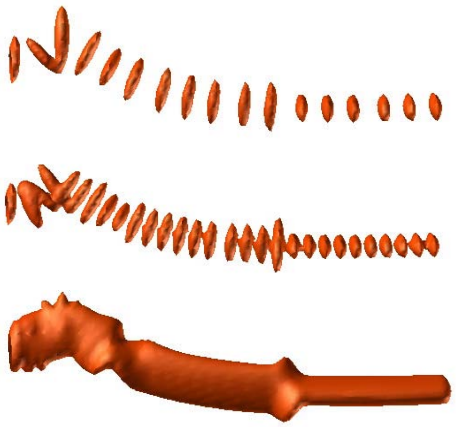

Figure 2: *The blending process of ellipsoid metaballs*

The blending process of the metaball is different from the merging of a set of ellipsoids because the blending surface will be C^∞ continuous, for example, in Figure 3, the left image is the merging of many ellipsoids along a curve, and the right is generated by the blending of a number of metaballs.

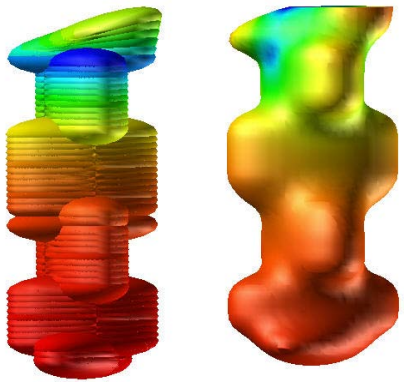

Figure 3: *Comparison of many ellipsoids with the smooth blending of metaballs*

Another issue is in the streamball visualization for the vector field, the velocity magnitude is mapped to R in Equation 10. However for the tensor field visualization we have to make the influence radius fixed, rather than controlled by the scalar. The reason is that the actual size of each metaball is also decided by the influence radius R. If another scalar value is mapped to it, the ellipsoid's three radii can not actually visualize the eigenvalues.

Now let us go back to the first problem about how to choose the position of the sources. In both Figures 2 and 3, we can see that the metaballs form a tube shape like surface if the sources are along a curve. If we select this curve as the stress trajectory in which the tangents are along the direction of one of eigenvectors, then the resulting surface is very similar to the hyperstreamline visualization of tensor data. If we set sources on a number of nearby hyperstreamlines then one large surface may be generated, which is similar to the hyperstreamsurface introduced by [JSF*02], in which a number of hyperstreamlines are connected with polygons.

4.3. Implementation

We implemented the hyperstreamball glyph as a module inside AVS/EXPRESS [UTFK*89], which can deal with both uniform and unstructured meshes. The first requirement is to decide the sources' positions, then the implementation of the hyperstreamball becomes one of computing the potential field generated by the sources which reflect the eigenvalues and eigenvectors, and finally this potential field is isocontoured. If there is no tensor matrix at a source, a tri-linear interpolation is applied to each component of the matrix, and then the eigenvalue and eigenvector decomposition is

performed. If the interpolation is not performed on the components of the matrix but its eigenvectors, the results will not be correct, because the sign of the eigenvector is then undetermined. For example, in Figure 4, the direction of the eigenvector at P_0 can either be 1 or 2, and the direction of the eigenvector at P_1 can either be 3 or 4. In order to get a vector at P_x, a linear interpolation can be performed. If we interpolate the vector with directions 1 and 3; for example, there is no problem. However, if we take direction 1 and direction 4, the eigenvector becomes degenerated at P_x, which is not what we want. Furthermore, a linear interpolation between

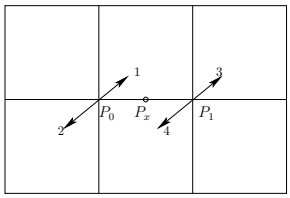

Figure 4: *The sign indeterminacy of eigenvector*

two unit vectors may not be a unit vector anymore. However, any linear combination of symmetric tensors remains a symmetric tensor [ZB02]. Finally, we use an extended marching cube algorithm [LC87] to render the isosurface for both uniform and unstructured meshes. Depending on the size of the metaball relative to the whole field, in order to generate a continuous surface, the mesh has to be of high resolution.

5. Interaction and Exploratory Results

We have evaluated our method using a synthetic but commonly used data set: the one point load stress tensors, and a real data set: the stress in a piece of long 3D railway line track (data courtesy of J. Kelleher, Manchester Materials Science Center). For the synthetic data first, we linearly map the eigenvalues to the radius of the distance function in equation 11 as shown in Figure 5, and the sources of hyperstreamball are located along nine stress trajectories. The large blended surface shows that hyperstreamball converges where the force is loaded on the top of the cube. At the bottom of the cube, there are almost no hyperstreamballs resulting from these being very small eigenvalues. This is because the principal stress decreased rapidly inside of the cube. Due to the large difference of eigenvalues on the top and the bottom of the cube, a visualization solution is to use a square root mapping to scale down the difference so that we can see both parts as shown in Figure 6.

In both Figures 5 and 6 we can see continuous tube shape like surfaces where the eigenvalues are large so that the metaballs blend together, and discrete metaballs where the eigenvalues become very small and separate. Though the square root mapping can show some small eigenvalues near the bottom of the dataset, not all can be seen and they are

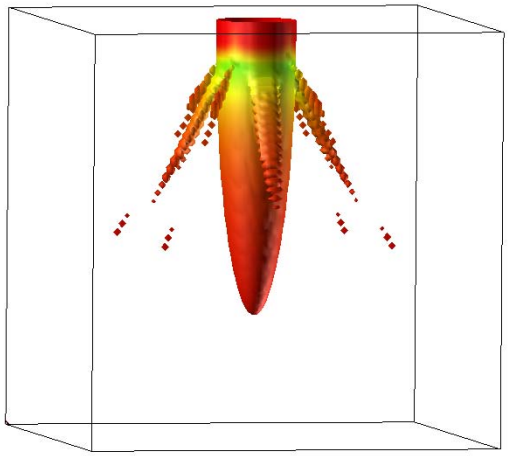

Figure 5: *The hyperstreamball visualization of one point load data by linear mapping*

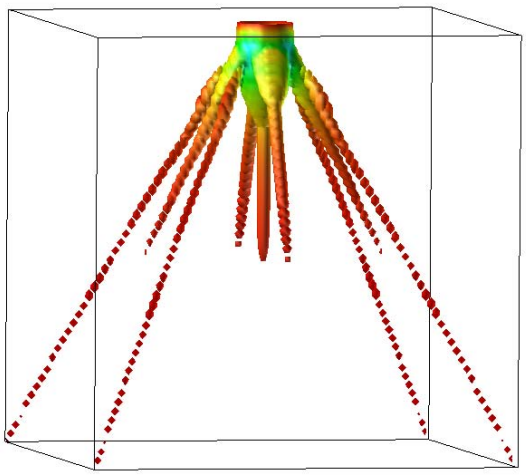

Figure 6: *Hyperstreamball visualization of a one point load tensor using square root mapping*

still very small being shown as points. One option to solve this is fixing the three radii of metaball to normalize them, so that we can see the direction in the small eigenvalues and the three eigenvalues can be mapped to the colour of metaball respectively. For example, in Figure 7, the sources of the hyperstreamball are still located along the nine stress trajectories, and discrete hyperstreamballs are shown as ellipsoids where the directions of an ellipsoid show the eigenvectors direction. This can be better observed by interactively rotating the whole model. If we decrease the distance of the sources

along the stress trajectories then nine hyperstreamlines are generated as shown in Figure 8, the distortion of the hyperstreamlines show the direction changes of the eigenvectors.

This also illustrates the second advantage of using hyperstreamball as the user can easily change from ellipsoid visualization to a continuous hyperstreamline visualization so that the clutter problem can be selectively reduced. However, too many hyperstreamlines can still clutter the visualization, and then the blended hyperstreamlines can form a surface as shown in Figure 9. In this figure hyperstreamballs are located on 49 hyperstreamlines to give an intuitive representation of how the stress propagates inside the cube. Figure 5 to 9, are all colour coded by the minor eigenvalues. Other colour mapping can be used as well.

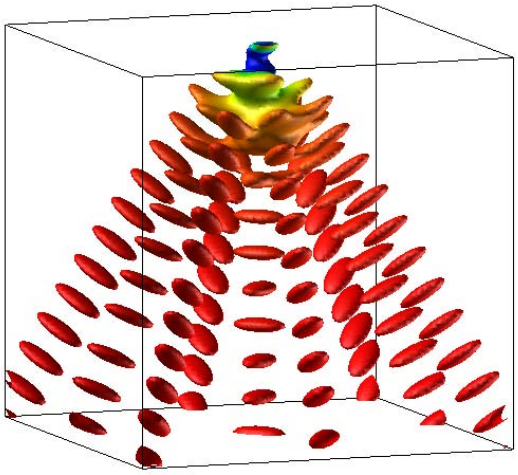

Figure 7: *The discrete hyperstreamball*

After exploring the hyperstreamball results of synthetic data, we can consider some real data. In the railway line track, the potential field was computed on an unstructured mesh, which consists of 609,200 elements, while the tensor data is located on a coarser mesh consisting of 39,520 elements to reduce computation of the eigenvalue-eigenvector decomposition and the integration for stress trajectories. The stress data here is the changing stress in the rail line track rather than the residual stress, which is simulated using ABAQUS. Along the rail direction (see Figure 10), the left is near the center of the rail while the right is the cut end of the rail. The stress gradually changes between these two ends. The engineer is interested in how it changes. In both Figures 10 and 12, the sources of hyperstreamball are located along five stress trajectories and the eigenvalues are normalized. In Figure 13, the hyperstreamballs are along 20 stress trajectories, and a hyperstreamsurface is formed. It can be seen that a sharp change of the principal stress direction happens near the cut end of the piece of rail (see Figure 11), while around the center it is almost invariant shown as straight cylinder except with ellipsoids rather than spheres at each end (see Figure 12) that are parallel to the direction of the rail. Figures 10, 12 and 13 are all colour encoded by the minor eigenvalues.

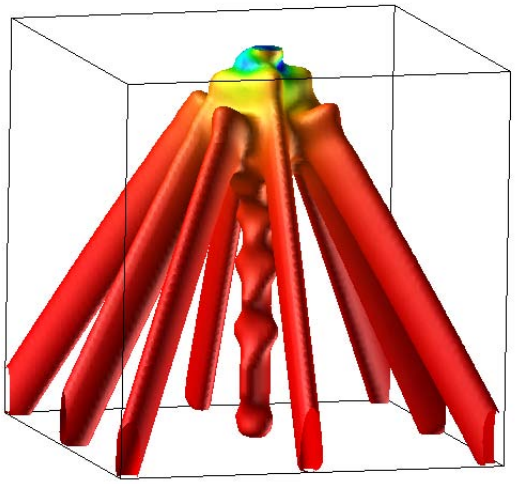

Figure 8: *The hyperstreamballs shown as hyperstreamlines*

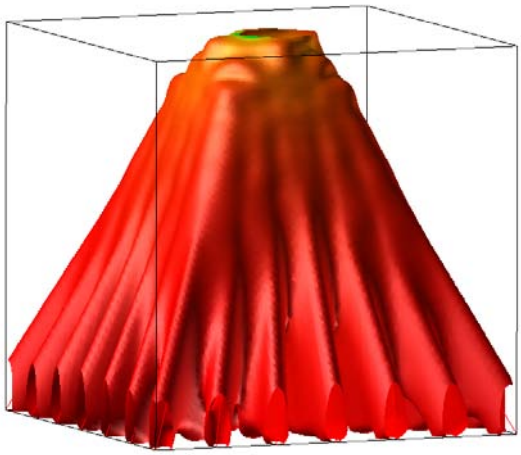

Figure 9: *The hyperstreamball visualization shown as hyperstreamsurface*

In summary, the hyperstreamballs have the following properties

- Under the same sample ratio, with the decrease of eigenvalues of stress tensor along the stress trajectory, the hyperstreamball changes from hyperstreamline to discrete ellipsoids, so that the continuous parts reflect the large

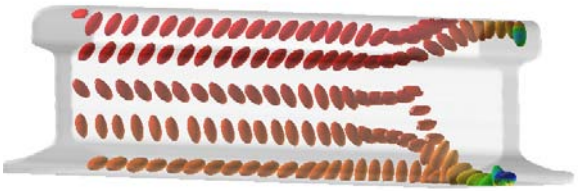

Figure 10: *The hyperstreamball visualization of stress shown as ellipsoids*

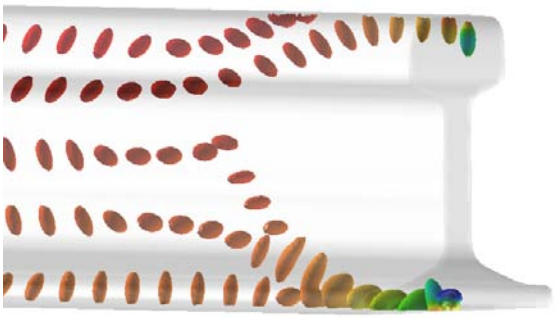

Figure 11: *The scaled view of discrete hyperstreamballs near the cut end of the rail*

principal stress areas and discrete parts reflect the small principal stress areas.

- When the eigenvalues are fixed along the stress trajectory, under a low sample ratio, hyperstreamballs show as discrete ellipsoids which present simultaneously the three eigenvector directions and eigenvalues. As the sample ratio increases, the ellipsoids will blend with each other smoothly to generate a surface, which is similar to the hyperstreamline. Furthermore cluttered hyperstreamlines can then form a hyperstreamsurface.

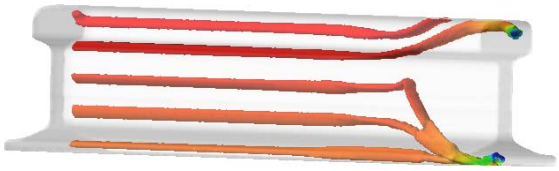

Figure 12: *The hyperstreamball visualization of stress shown as hyperstreamlines*

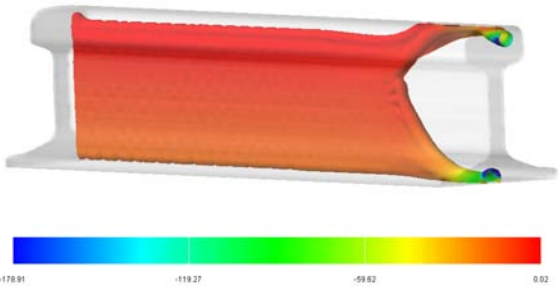

Figure 13: *The hyperstreamball visualization of stress shown as hyperstreamsurface*

6. Conclusions and Future Work

In this paper we propose a new glyph called hyperstreamball to visualize 3D symmetric tensor fields. The biggest advantage of the hyperstreamball is that it bridges the difference between ellipsoid, hyperstreamline and hyperstreamsurface. The user can easily switch between these modes using just one glyph.

One of future directions of our work is to improve the speed of computation of the potential field. Currently every time the user switches from ellipsoid to the hyperstreamline or hyperstreamsurface, the whole potential field is recomputed. It is possible to only compute the potential field near the sources. Another direction of the future work is to apply the hyperstreamball to more tensor data such as diffusion tensor data obtained from MRI. In this case, the seeding of hyperstreamball will be of interest.

7. Acknowledgment

This work is funded by the School of Computer Science at the University of Manchester.

The authors greatly appreciate J. Kelleher who provided the rail tensor data and a lot of insightful comments on stress tensors. Thanks to Tobias Schiebeck, Yien Kwok, Mary McDerby, and all the other Manchester Visualization Center students and staff for their help and support during this research.

References

[BHKD94] BRILL M., HAGEN H., KLIMENKO S., DJATSCHIN W.: Streamball techniques for flow visualization. In *IEEE Visualization '94 Washington DC* (1994), IEEE Computer Society Press, pp. 225–331.

[Bli82] BLINN J.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[CRB*05] CROSSNO P., ROGERS D. H., BRANNON R. M., COBLENTZ D., FREDRICH J. T.: Visualization of

geologic stress perturbations using mohr diagrams. *IEEE Trans. Vis. Comput. Graph. 11*, 5 (2005), 508–518.

[DH92] DELMARCELLE T., HESSELINK L.: Visualization of second order tensor fields and matrix data. In *Proceedings Visualization '92* (1992), IEEE Computer Society Press.

[DH94] DELMARCELLE T., HESSELINK L.: The topology of symmetric, second-order tensor fields. In *Proceedings of the conference on Visualization '94* (1994), IEEE Computer Society Press, pp. 140–147.

[Dic89] DICKINSON R.: A unified approach to the design of visualization software for the analysis of field problems. *SPIE Proceedings 1083* (January 1989), 173–180.

[dLvW93] DE LEEUW W., VAN WIJK J.: A probe for local flow field visualization. In *Proceedings Visualization '93* (1993), IEEE Computer Society Press, Los Alamitos, CA, pp. 39–45.

[F65] F Y.: *Foundations of solid mechanics*. Prentice Hall international series in dynamics, 1965.

[Hab90] HABER R.: Visualization techniques for engineering mechanics. *Computing Systems in Engineering 1* (1990), 37–50.

[HLL97] HESSELINK L., LEVY Y., LAVIN Y.: The topology of symmetric second-order 3D tensor fields. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (January-March 1997), 1–11.

[JSF*02] JEREMIC B., SCHEUERMANN G., FREY J., YANG Z., HAMANN B., JOY K. I., HAGEN H.: Tensor visualization in computational geomechanics. *International Journal for Numerical and Analytical Methods in Geomechanics 26* (2002), 925–944.

[KW99] KINDLMANN G., WEINSTEIN D.: Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *Proceedings of the conference on Visualization '99* (1999), IEEE Computer Society Press, pp. 183–189.

[KWH00] KINDLMANN G., WEINSTEIN D., HART D.: Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics 6*, 2 (April-June 2000), 124–138.

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169.

[MSM95] MOORE J., SCHORN S., MOORE J.: Methods of classical mechanics applied to turbulence stresses in a tip leakage vortex. In *International Gas Turbine and Aeroengine Congress & Exposition. AMSE* (1995).

[UTFK*89] UPSON C., THOMAS FAULHABER J., KAMINS D., LAIDLAW D. H., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A.: The application visualization system: A computational environment for scientific visualization. *IEEE Comput. Graph. Appl. 9*, 4 (1989), 30–42.

[WKL99] WEINSTEIN D., KINDLMANN G., LUNDBERG E.: Tensorlines: advection-diffusion based propagation through diffusion tensor fields. In *VIS '99: Proceedings of the conference on Visualization '99* (Los Alamitos, CA, USA, 1999), IEEE Computer Society Press, pp. 249–253.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2*, 4 (April 1986), 227–234.

[YMAHW03] YOUSSEF M. A. HASHASH1 J. I.-C. Y., WOTRING D. C.: Glyph and hyperstreamline representation of stress and strain tensors and material constitutive response. *International Journal For Numerical and Analytical Methods in Geomechanics* (2003).

[ZB02] ZHUKOV L., BARR A. H.: Oriented tensor reconstruction: Tracing neural pathways from diffusion tensor mri. In *Proceedings of the conference on Visualization '02* (2002), IEEE Computer Society.

[ZP04] ZHENG X., PANG A.: Topological lines in 3D tensor fields. In *IEEE Visualization 2004, Austin Texas* (2004), IEEE Computer Society Press, pp. 313–320.

[ZPP05] ZHENG X., PARLETT B. N., PANG A.: Topological lines in 3D tensor fields and discriminant hessian factorization. *IEEE Trans. Vis. Comput. Graph. 11*, 4 (2005), 395–407.