Volume Deformation For Tensor Visualization

Xiaoqiang Zheng and Alex Pang
Computer Science Department
University of California, Santa Cruz, CA 95064
zhengxq@cse.ucsc.edu, pang@cse.ucsc.edu

ABSTRACT

Visualizing second-order 3D tensor fields continue to be a challenging task. Although there are several algorithms that have been presented, no single algorithm by itself is sufficient for the analysis because of the complex nature of tensor fields. In this paper, we present two new methods, based on volume deformation, to show the effects of the tensor field upon its underlying media. We focus on providing a continuous representation of the nature of the tensor fields. Each of these visualization algorithms is good at displaying some particular properties of the tensor field.

Key Words and Phrases: stress, strain, shear, symmetric tensors, anti-symmetric tensors, anisotropic tensors.

1 INTRODUCTION

The analysis of tensor data with various orders is important in many medical and physical applications. For example, tensors can be found in medical imaging (e.g. diffusion tensor images [9]), fluid flow (e.g. velocity gradient [4]), stress simulation and tectonics. Although scientists often have to deal with tensor data sets, the tools available for visualizing them are quite limited. In this paper, we restrict our discussion to second order 3D tensor fields. Visualizing such tensor fields is difficult because each tensor in the field contains nine unique quantities. To incorporate so much information in a single representation is a challenging task. Current methods either show detailed information about the tensor field but only at a few local, discrete points using glyphs, or show partial information about the tensor field but over a global, continuous domain.

In this paper, we present two new techniques based on deformation of the tensor volume. Deformation is an intuitive way to represent a tensor. We can observe the tensile, compressive and shear effects of a tensor value through the transformation of the volume. The conceptual idea of this algorithm is to consider the tensor field as a force field that deforms an object. The local deformation of the object represents the tensor value at that point in the tensor field.

Both of the two techniques are designed to visualize some particular properties of the tensor field. First, the *normal vector deformation* algorithm is good at showing the directional information of the tensor field. Second, the *anisotropic deformation* is capable of illustrating the compressing and shearing property of the tensor field.

The rest of this paper is organized as follows: Section 2 reviews some previous tensor visualization techniques; Section 3 reviews some tensor decomposition methods to aid with the visualization task; Section 4 presents the two new volume deformation technique; Section 5 discusses some implementation issues; Section 6 illustrates the methods and analyzes the results; Section 7 draws conclusions for the two volume deformation techniques and proposes future works for tensor visualization.

2 PREVIOUS WORK

There are a few methods for visualizing tensors such as pseudo-coloring, tensor glyphs [2, 7, 8], hyperstreamlines [3], and deformation [1, 10, 12].

Pseudo-coloring is the simplest but also least effective method. For a tensor field with 9 independent components, a planar slice is made through the tensor volume and each of the 9 components on that slice are pseudo-colored. The tensor information for that slice is then presented as a 3 by 3 collage. The main drawback of this technique is that users have to mentally integrate the physical interaction of the 9 tensor components. A more popular method involves the use of glyphs. The simplest one is the tensor ellipsoid. Here, the tensor is decomposed into three orthogonal eigenvectors, with their corresponding eigenvalues indicating the magnitude along each eigenvector. An ellipsoid is then constructed oriented according to the 3 eigenvectors and scaled according to the 3 eigenvalues. This works for symmetric tensor fields. More complex glyphs are constructed to show additional features in the tensor fields using flow probes [2]. The general drawback of glyphs is their discrete nature. They do not capture the global or continuous nature of tensor fields. And they also require judicious placement to avoid clutter.

Another well known tensor visualization approach is with hyperstreamlines or tensor field lines [3]. For symmetric tensor fields, the 3 eigenvectors at each point are sorted by their eigenvalues and classified as the major, medium, and minor eigenvectors. Hyperstreamlines are then generated by integrating along one of these eigenvector fields, and letting the two other eigenvector fields to modify the shape of an ellipse that is swept along the principal hyperstreamline. For non-symmetric tensor fields, where the 3 eigenvectors are not necessarily orthogonal to each other, the tensor field is first decomposed into a symmetric and an axial component. Ribbons along the hyperstreamline are then added to show the rotational effects of the axial component. Because one of the eigenvector fields is used for integrating the hyperstreamline, there are two other possible hyperstreamlines that can be generated from each seed location. The understanding of the tensor field is therefore not complete without these two. The drawback of this approach then is that users have to integrate the 3 different views.

An alternative approach is to use deformation. Simple objects such as polygons [12] or cubes [10] are advected by a flow field. The deformations on the objects show the local stretch, shear, and rigid body rotation at a point. This is generalized in [1] to include idealized objects such as lines, planes, sub-volumes. It allows users to interactively modify an interrogation vector to see how the tensor field would deform the object. While the user can now see the continuity over the field, the main drawback of this approach is that users can only see the information in one direction at a time. Our goal then in this paper is to maintain the spatial continuity of the presentation, while at the same time provide a net deformation effect.

3 TENSOR DECOMPOSITION

A general strategy with tensor field visualization is to decompose the field into meaningful components. Visualization is then performed based on the components. We review some basic definitions and two of the common decomposition methods.

A tensor field is a field that contains a tensor at each point. We assume that the data resides in a structured grid and also assume that it is basically continuous. In a general tensor, each component of the tensor is a unique quantity and it can be represented as follows:

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

Velocity Gradient

In many applications, the tensor field is generated from the gradient of a vector field V or the second-order gradient of a scalar field s. In the latter case, the tensor value is symmetric [6]. The tensor fields are represented in the following way:

$$T = \begin{bmatrix} \frac{\partial V_x}{\partial x} & \frac{\partial V_x}{\partial y} & \frac{\partial V_x}{\partial z} \\ \frac{\partial V_y}{\partial x} & \frac{\partial V_y}{\partial y} & \frac{\partial V_y}{\partial z} \\ \frac{\partial V_z}{\partial x} & \frac{\partial V_z}{\partial y} & \frac{\partial V_z}{\partial z} \end{bmatrix} \tag{2}$$

or

$$T = \begin{bmatrix} \frac{\partial^2 s}{\partial x \partial x} & \frac{\partial^2 s}{\partial x \partial y} & \frac{\partial^2 s}{\partial x \partial z} \\ \frac{\partial^2 s}{\partial y \partial x} & \frac{\partial^2 s}{\partial y \partial y} & \frac{\partial^2 s}{\partial y \partial z} \\ \frac{\partial^2 s}{\partial z \partial x} & \frac{\partial^2 s}{\partial z \partial y} & \frac{\partial^2 s}{\partial z \partial z} \end{bmatrix} \tag{3}$$

In the velocity gradient tensor, the local velocity can be expressed as:

$$V = V_0 + \frac{\partial V}{\partial x} \cdot \delta x + \frac{\partial V}{\partial y} \cdot \delta y + \frac{\partial V}{\partial z} \cdot \delta z \tag{4}$$

or

$$V = V_0 + \nabla V \cdot \delta r = V_0 + T \cdot \delta r \tag{5}$$

Given a tensor T and a resolute vector W [1], W will be transformed into another vector U:

$$U = T \cdot W \tag{6}$$

So a tensor can be considered as a linear transformation from a vector to another vector.

Rate of Strain

In fluid flow analysis, the rate of strain tensor is an important feature. It includes information for the local deformation of the fluid elements at any instant. For a small volume of fluid in the field, the rate of strain tensor is defined as its rate of change of shape. We can compute this rate of strain tensor as follows [11]:

$$\begin{aligned} C &= \begin{bmatrix} \frac{\partial V_x}{\partial x} & \frac{1}{2}(\frac{\partial V_x}{\partial y} + \frac{\partial V_y}{\partial x}) & \frac{1}{2}(\frac{\partial V_x}{\partial z} + \frac{\partial V_z}{\partial x}) \\ \frac{1}{2}(\frac{\partial V_y}{\partial x} + \frac{\partial V_x}{\partial y}) & \frac{\partial V_y}{\partial y} & \frac{1}{2}(\frac{\partial V_y}{\partial z} + \frac{\partial V_z}{\partial y}) \\ \frac{1}{2}(\frac{\partial V_z}{\partial x} + \frac{\partial V_x}{\partial z}) & \frac{1}{2}(\frac{\partial V_z}{\partial y} + \frac{\partial V_y}{\partial z}) & \frac{\partial V_z}{\partial z} \end{bmatrix} \\ &= \frac{1}{2}(\nabla V + (\nabla V)^T) \end{aligned} \tag{7}$$

Obviously, the rate of strain tensor is symmetric. And if the fluid is incompressible, this tensor field has the following property [11]:

$$tr(C) = \nabla \cdot V = 0 \tag{8}$$

where C is the rate of strain tensors, and the function tr() is the sum of all diagonal components.

Symmetric and Anti-Symmetric Decomposition

A common decomposition method is to divide the tensor into a symmetric S and an anti-symmetric A component [6].

$$T = S + A = \frac{T + T^t}{2} + \frac{T - T^t}{2} \tag{9}$$

T^t is the transpose of T. The symmetric tensor represents the stress and strain effects and has 6 independent components, while the anti-symmetric tensor has 3 independent components and represents local rigid body rotation.

Polar Decomposition

In this technique, a general tensor is decomposed into a symmetric and an orthonormal rotation component [6]. The symmetric tensors U and V represent the compressive effect of the tensor; the orthonormal one shows the amount of rotation in the tensor transformation.

A symmetric tensor has three orthogonal eigenvectors. This means that as a transformation rule, a symmetric tensor only changes the eigenvalues on the axes in its local orthogonal coordinate. The direction information is not lost after the transformation on these three axes. In contrast, a rotation tensor has only one eigenvector and the length of the vectors are never changed in all directions after the transformation.

We can obtain R, U and V as follows [6]:

$$T = R \cdot U = V \cdot R \tag{10}$$

$$U = \sqrt{T^t \cdot T} \tag{11}$$

$$V = \sqrt{T \cdot T^t} \tag{12}$$

4 APPROACH

We present two techniques whose focus is on providing a continuous and complete representation of the tensor field over the entire data space. Volume deformation is used to visualize the tensor field by letting it deform the space that contains it. The deformation is carried out according to a set of transformation rules that represent some properties of the tensor field. The resulting shape and structure of the deformed space, including subtle details, reveals the nature of the tensor field.

The main difference between the two methods presented here is the definition of the transformation rules. Both of them capture the main features of the tensor field. In both cases, we assume that the tensor field is sufficiently smooth so that there are no sudden changes in neighboring tensors, and tensors vary linearly between neighboring tensors. We also assume that the tensor data resides in a structured grid such as a rectilinear or curvilinear grid.

4.1 Normal Vector Deformation

Boring and Pang [1] described a method where an ideal object, such as a plane, is placed in the tensor field and deformed. The resolute vector to generate the deforming forces is specified by the user as an interrogation vector. This vector is homogeneous throughout the tensor field, and hence only tensor information in one direction is included in the visualization. In addition, only a small number of planes can be included in one picture to avoid visual clutter. So the user needed to interactively move the location of the plane, and change the interrogation vector to see the properties of different parts in the tensor field.

We improve upon this method in two ways. First, the interrogation may be different in different parts of the tensor field, and may change over time as the space is deformed. Secondly, the entire volume is deformed, and can be viewed using wireframe, layered planes, or volume rendered.

We proceed by using the same notion that a tensor field can be considered as a special force field with different properties in all directions. We can explore its features by looking at its impact on the space that contains it. In this method, we multiply the tensors and the normal vectors of the object to get the deforming forces. The object is then twisted or stretched by these forces. While we use the normal vectors in this paper, they can be replaced by a user specified vector.

To obtain the normal vectors, we traverse through the tensor volume along one of the 3 axes in computational space. That is, we are processing the volume one layer or sheet at a time. The normal vectors are then, initially, the normals to the current layer in physical space. As the layer is deformed, the normals will be updated according to the new surface orientation. Therefore, generally speaking, the normal vectors are different everywhere in the tensor field and represent the tensor information in all directions.

With only this adaptation, the simulation is unstable, because the tensors could be heterogeneous in a small area. This produces artifacts on the deformed object because changing normal vectors may fluctuate wildly. To solve this problem and to get a reasonable distribution of normal vectors, we use a physics-based model to constrain the movement of the object. Each layer is treated as a rubber sheet, which is simulated by a lattice of springs connecting the nodes. Using this approach, the deformation proceeds until the spring system reaches equilibrium. An added benefit is that we can vary the spring constants to exaggerate or de-emphasize the amount of deformation while keeping the deformations proportional to the tensors. In Figure 1, we see 3 frames of a volume under a single point load undergoing deformation after different iterations.

Another change in this method is that we can add a number of the deformed slices in the visualization to display the properties of different parts of the tensor field. If we simply put these deformed planes together, there will be too much visual clutter and users cannot see anything through it. So, we add very weak spring links between each layer. These springs make the deformation relatively regular through the layers, so the visualization is much cleaner and makes more visual sense. And because we can put an arbitrary number of slices in the visualization without causing too much visual disorders, we can exhibit tensor information everywhere in the the tensor volume in one picture.

In this deformation technique, we define $N(x)$ as the normal vector of the current layer at location x. Then we define the deforming force $F_e(x)$ at that location as:

$$F_e(x) = T(x) \cdot N(x)$$

$F_e(x)$ is the transformed normal vector in the tensor fields. We treat it as an external force, due to the tensor, acting at that point in space. Each point in the tensor field has a corresponding force acting on it, and we deform the object based on the collection. Since the layer is represented as a spring lattice, these external forces will disrupt the balance of forces in the system. To restore this balance, the system will have to displace, compress and stretch, the points around until the internal and external forces reach a new equilibrium state. The new deformed layer represents the state where the forces are at a minimum. This is consistent with our experience with real objects and hence the users can easily identify the force field based on the resulting deformations.

We use numerical integration to carry out the deformation process. In each iteration, we calculate a combined force $F_c(x)$ as a function of the internal forces $F_i(x)$ and external forces $F_e(x)$. We define the deformed layer as the function $D(x)$, and the internal forces $F_i(x)$ is then defined as follows:

$$F_i(x) = -\oint \frac{\partial D}{\partial x} \cdot \left(\frac{\left\| \frac{\partial D}{\partial x} \Delta x \right\|}{\|\Delta x\|} - 1 \right) \cdot \Delta x$$

In each iteration, the combined force $F_c(x)$ is applied to the object to compute the actual deformation, and to obtain the new position $P_{k+1}(x)$ of the deformed layer. When the values in the combined force fields are zero at all points, the object has reached equilibrium and the integration terminates. The resulting deformation is the equilibrium state where the active forces are minimal throughout.

$$F_c(x) = F_e(x) + F_i(x) = T \cdot N + F_i(x)$$

$$P_{k+1}(x) = P_k(x) + F_c(x) \cdot TimeStep$$

where $TimeStep$ is the integration time step. In our experiments, it must be less that 0.1 to be stable. For better results, it should be chosen at between 0.005 and 0.05. We set it as 0.01 for most of the simulations. Each node in the lattice is connected to its immediate neighbor requiring 26 springs for each 3D point. The rest length of a spring is the distance between the initial positions of the two nodes straddling it. Except for the four corner points of each layer, all lattice nodes are free to move around.

The main drawback of this algorithm is that users have to specify one of 3 directions to traverse the volume. It provides a visualization of the entire tensor field domain, but suffers the same drawback as hyperstreamlines in that users will need to mentally integrate 3 different views. Ideally, we want the interrogation vector to be generated automatically, be data dependent and represent the maximal impact of the tensor at that point. A possible candidate is to use the average of the 3 eigenvectors as the interrogation vector, but it is not clear how this average vector can be updated in subsequent iterations. We continue to investigate this alternative. In the meantime, we also have another method, presented in the next section, that addresses the problem of having to depend on an interrogation vector.

4.2 Anisotropic Deformation

This is another deformation based algorithm that addresses the limitation in the previous algorithm, and is good at showing the compression and shearing properties of the entire tensor fields.

We consider tensors as a transforming force such that if one has a tensor field where T is the same at every location, then a spherical ball placed in such a medium would be deformed into an ellipsoid with the appropriate orientation and scaling. Likewise, in 2D, a tensor will transform a square into a parallelogram. Simply placing these parallelograms in 2D or ellipsoids in 3D will produce too much clutter. The conceptual idea of the *Anisotropic Deformation* algorithm is to assemble these deformed shapes and volumes in a natural way. Each deformed shape, a deformed hexahedron in 3D, represents the local tensor information. The feature presented by this algorithm is the anisotropy and magnitude of the tensors, which are intuitively encoded by the deformation. This algorithm is a natural realization to the idea of the strain tensors, because the tensor data are supposed to estimate the local deformation. We call this algorithm *Anisotropic Deformation* because tensor values are different in each direction.

When assembling the deformed hexahedral volumes, we need to rotate and move them, and even change their shape a little so that adjacent volumes can be put together. When applied to each element in the tensor volume, the entire volume is deformed and each sub-volume represents the local tensor information. So both globally and locally consistent deformations are obtained.

To carry out this type of deformation, we proceed as follows. Assume there are two neighboring points x_0 and x_1, and a vector v formed by the difference between these 2 points. Then, the distance between these 2 points after deformation is equal to the product of the tensor and vector v. Specifically, if we let the deformation function be denoted by $D(x)$ to indicate the position of point x after deformation, then we want to find $D(x)$ such that:

$$\|D(x_0) - D(x_1)\| = \|T(x_0) \cdot (x_0 - x_1)\| \tag{13}$$

If this equation holds for any neighboring points x_0 and x_1, then it follows that:

$$\left.\frac{\partial D}{\partial x}\right|_{x=x_0} = R(x_0) \cdot T(x_0) \tag{14}$$

where R(x) is a rotation tensor field. This is because if all v satisfy this condition, then $\frac{\partial D}{\partial x} \cdot T^{-1}$ should be a rotation tensor. Because a tensor can be decomposed into a symmetric tensor and a rotation tensor, as provided by polar decomposition, the visualization of $D(x)$ is equivalent to that of the original symmetric real tensor field $T(x)$.

However, one can prove that for a general tensor field, there is no transformation function $D(x)$ satisfying this critical condition. In other words, there is too much information to be crammed into a single deformed object. So we employ the physics-based model again to resolve the conflicts and try to find a deformation $D(x)$ that is closest to the critical condition defined by Equation 14.

In this method, part of the tensor information will be lost, but the surviving features will be the dominant ones, which are important for understanding the data sets.

Again, we establish a spring model for the object. Each node is connected to its neighbors in 3D as in the previous method. However, because this method does not require a resolute vector and hence does not require us to process the volume in layers, we do not need to have weaker spring constants between layers. Instead the spring constants in this model are homogeneous. The default length of each spring is the original distance between neighboring nodes. Therefore, before deformation, all springs are at a rest. When the deformation starts, the tensor field “consumes” the springs forces by changing their rest lengths.

In order to make the deformation approach specified by Equation 13 work, the rest length of the spring between x_0 and x_1 is recalculated and expressed as $L(x_0, x_1)$ below:

$$L(x_0, x_1) = \lambda \|T(x_0) \cdot (x_0 - x_1)\| + (1 - \lambda) \|x_0 - x_1\| \tag{15}$$

where λ is a user-specified parameter that controls the amount of deformation incorporated in the object. Its value should be in the [0, 1] range. In real simulations, it should be set between 0.1 and 0.9. If it is too small, no deformation can be detected by users. If it is too large, i.e. 1, then the length of some springs may degenerate to 0 because of some singular tensors. Physics-based rules are used to simulate the deformation process of the object. At equilibrium, the length between the points is an approximate measurement to $L(x_0, x_1)$. The internal force fields are computed in the following way:

$$F(x_0) = \sum_{x_1} K \cdot (x_1 - x_0) \cdot \left(\frac{\|x_0 - x_1\|}{L(x_0, x + x_1)} - 1 \right)$$

where K is the spring constant and there is no external force field. So $F(x)$ is used to compute the new positions of all points i.e. the transformation of the object in each iteration. This deformation process stops when it comes into equilibrium, where the magnitude of $F(x)$ is below a threshold everywhere.

This algorithm shows the compressive and shearing properties of the tensor data. Although it is an approximation of the ideal situation, the important and dominant features do survive and are presented in the deformed object. An error detection factor $e(x_0)$ is defined to keep track of the accuracy.

$$e(x_0) = \max_{x_1} \left(\left| \frac{\|P(x_0) - P(x_1)\| - L(x_0, x_1)}{L(x_0, x_1)} \right| \right)$$

This error detection factor is an accuracy measurement of the local transformation of the tensor value at that point. Its maximum over the entire space gives a lower bound of the accuracy of the visualization. We find that for most simulations, this error factor is below 10%. This means the algorithms work quite effectively. However, it is still possible that a smooth-looking deformation is a visualization of a not-so-smooth tensor field smoothed by the spring model. In that case, the error rate will be very high. This serves as a reminder for users that many subtle details may be lost. They can then specify a smaller volume to conduct the deformation and to take a closer look at the tensor field.

5 IMPLEMENTATION ISSUES

In the previous section, we proposed two algorithms to visualize the tensor fields. In the implementation, we need to address some issues in order to get a good result and a high processing speed.

Numerical Stability

The work horse method for numerical integration is the 4th order Runge-Kutta method. We use a modified version as follows. Suppose $P(x, t)$ is the position of point x at time t, $F(x, P(t))$ is the force of point x . The force function F takes the position P as a parameter to compute the force field. The equation is as follows:

$$\frac{\partial P(x,t)}{\partial t} = F(x, P(t)) \tag{16}$$

This means the adjustment of the positions is equal to the force field at the same time step. Then the problem is adjusted so that we can use fourth-order Runge-Kutta method to do the integration. We need to keep in mind that this is not exactly the original problem. The original problem only needs the resting state of the spring models and it does not require the accurate intermediate stages. So we can change the definition of F to speed up the processing as long as F has the same resting states.

Adaptive Time Step

A wisely chosen time step is important for the spring models in the integration process. If it is too small, the integration will take more iterations to finish, making the program slow. On the other hand, if it is too big, we may never reach a stable solution of the resting states. So we employ an adaptive time step algorithm in the integration process to get a reasonable time step for each round. The time step is determined as follows:

$$T = \gamma \cdot \min_x \left(\frac{\max_{x_0} \|x - x_0\|}{\|F(x)\|} \right)$$

where γ is a user-specified parameter to control the time step. In our implementation, it is set to 0.1. This algorithm ensures that no single movement in an iteration can destroy the connection structure by constraining the maximum amount of deformation.

6 RESULTS

Data sets

We test our algorithms on three data sets: single point load, delta wing, and cylinder with hemispherical cap data sets. The single point load data is a standard tensor data set. It is simple, well known and thoroughly studied, so it can verify the correctness and the usefulness of a new algorithm. Both the delta wing and the cylinder with hemispherical cap data sets are vector data. The velocity gradients in delta wing data set and the rate of strain tensors in cylinder with hemispherical cap data set are extracted as the tensor data sets.

The delta wing data contains information about “flow past a simplified geometry representing a delta wing aircraft, at a moderately high angle of attack” [5] and is available from www.nas.nasa.gov/Research/Datasets/datasets.html. The cylinder with hemispherical cap data contains information about flow past that cylinder at an angle.

In this section, we demonstrate the two deformation algorithms using these data sets and make observations about the tensor fields.

Single point load

Because this data set is a simple tensor data set and it is often used for the validation tests, we conduct experiments of both methods on it and present the results.

Figure 1 shows the results of the *normal vector deformation* algorithm. The images in the sequence are taken at different integration times to show the dynamic deformation process. The color is mapped into the magnitude of forces exerted on the object by the tensors. Red means high magnitude, while blue means low magnitudes. We see that the central part of the layers are pushed out and while the layers try to stay smooth to minimize the stress applied on it.

The primary sweeping axis in this figure is in the same as direction of the point load. It is obvious that the tensors in the middle of the field have larger values than the tensors on the sides along the point load direction.

The maximal error detection factor for this method in the final stage is 0.0311. This means that the distance between the transformed points is within the $\pm 3.11\%$ range of the tensor value in that direction at the point.

Figure 2 illustrates the tensor field using anisotropic deformation. Each of the images is rendered and colored in different ways to display various properties of the tensor field. We can see that all of their central parts are pushed out. This means that the tensor values are large in these directions. Figure 2(a) is rendered using opaque surfaces and color coded by the remaining tensor magnitude at each point. We note that there is still room for some additional deformation because of the green regions. Figure 2(b) is rendered using transparent volumes and color coded with the initial tensor magnitude. It clearly shows the distribution of the deformation. We use a frontal view in Figure 2(c) to show that the perimeter are slightly pushed out as part of the deformation. Figure 2(d) is taken from the back side of the cube where the point load is actually applied. Both the cube structure and the color distribution display the spread-out properties of the tensor field. In this visualization, we observe that the central tensors have larger values along the main direction and it is larger than those of the other two directions too.

Both figures show some particular properties of the point load data set. *Normal vector deformation* shows us the tensor information with emphasis in one primary direction. *Anisotropic deformation* is good at exhibiting enlarging, compressing, and shearing properties in all directions of the tensor field.

Delta Wing

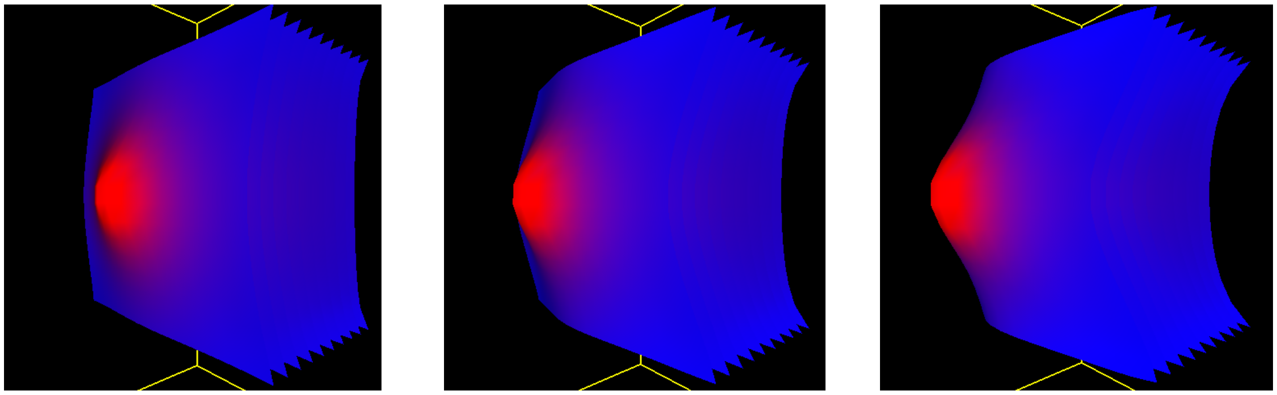
We apply the *normal vector decomposition* on the velocity gradient tensor field derived from this data set and display the results in Figure 3. Figure 3(a) is taken from the front of the delta wing and colored with tensor magnitude. From the twisted contour of the volume, we see there is a high shear region in the top part of the volume. Figure 3(b) is basically the same picture but rendered with opaque surfaces. The shading gives users better clues about the subtle structure of the deformed space around the delta wing. Figure 3(c) and Figure 3(d) are both colored with tensor forces. One is taken from the front and the other is from the back. From these images, we see how the major compressive direction and anisotropy changes along the profile.

The maximal error rate is 7.61%. In places where with a high error rate in the deformation, the anisotropy information is better preserved than the magnitude. This means that although the spring is stretched or compressed and cannot well represent their rest length, the scale of change is approximately the same in all directions. This property is especially useful in applications where anisotropy is very important.

Cylinder With Hemispherical Cap

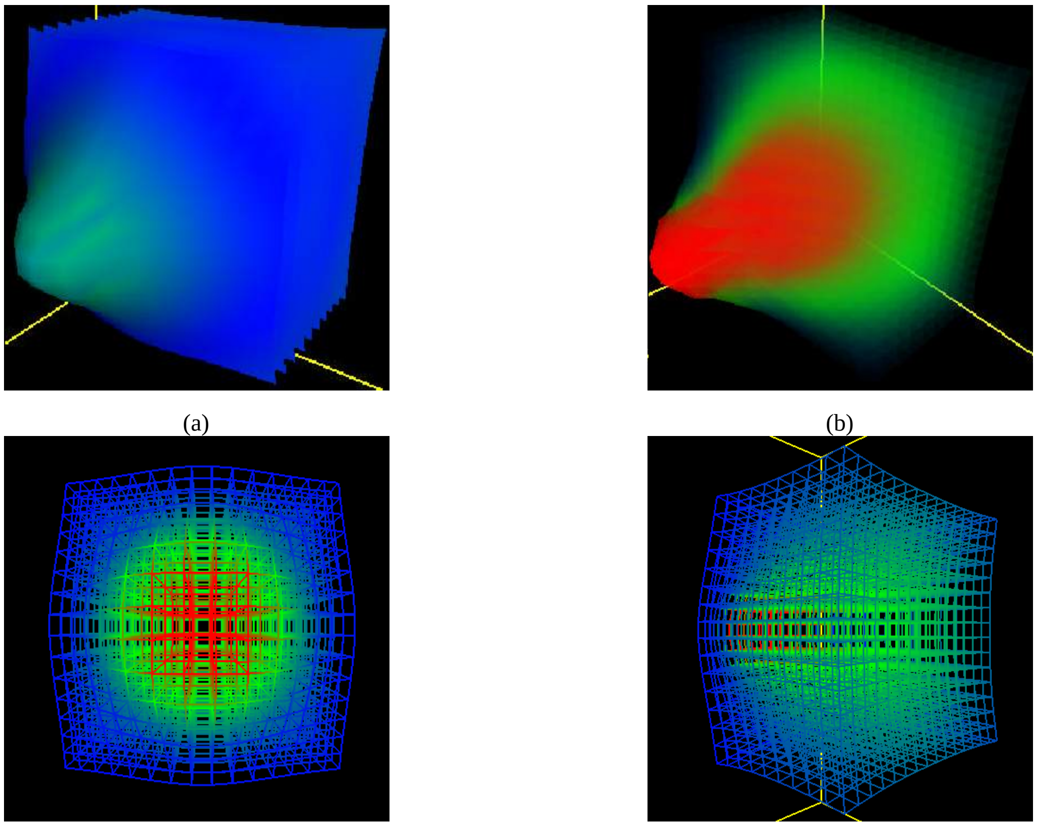
We use the *anisotropic deformation* on the rate of strain tensor field derived from this data set and the results are shown in Figure 4. Images from different perspectives and different rendering techniques are grouped together to show the properties of different parts of the tensor field.

Figure 4(a) is an image from the back of the cylinder with hemispherical cap. We can clearly see that the tensor values in the inner layers are much larger than those of the outer layers. The spacing of the layers at different points along the cylinder also convey information about the tensor values. Figure 4(b) is taken from the right side of the tensor field and is rendered in wire frame mode. The bulge on the open mouth of the rendered volume shows a high compressive and stretching tensor transformation in that area. That means the major eigenvectors, which is the main stretching direction, merge in that spot. This can also be observed by the hyperstreamlines result presented by Hesselink [3]. Also from the wireframe, we can



(a) t=100 (b) t=200 (c) t=300

Figure 1: Normal vector deformation on the single point load data colored with tensor forces.



(a) (b)

(c) (d)

Figure 2: Anisotropic deformation on the single point load data. (a) is rendered with opaque surfaces and colored with the remaining tensor magnitude; (b) is rendered with transparent volumes and colored with tensor forces; (c) and (d) is rendered with wireframes and colored with tensor forces.

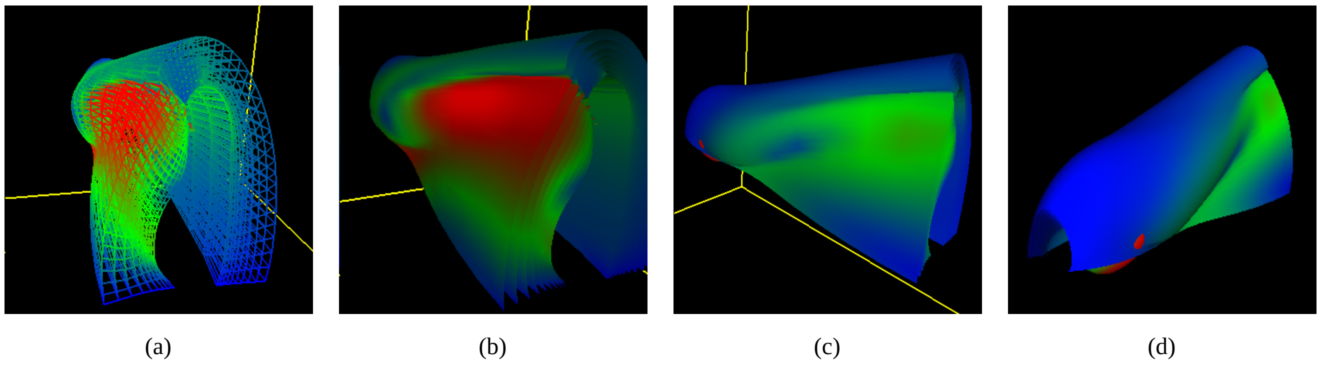


Figure 3: Normal vector deformation on the delta wing data. (a) is rendered in wireframe and colored with tensor magnitude; (b) is rendered with opaque surfaces and colored with tensor magnitude; (c) and (d) is rendered with opaque surfaces and colored with tensor forces.

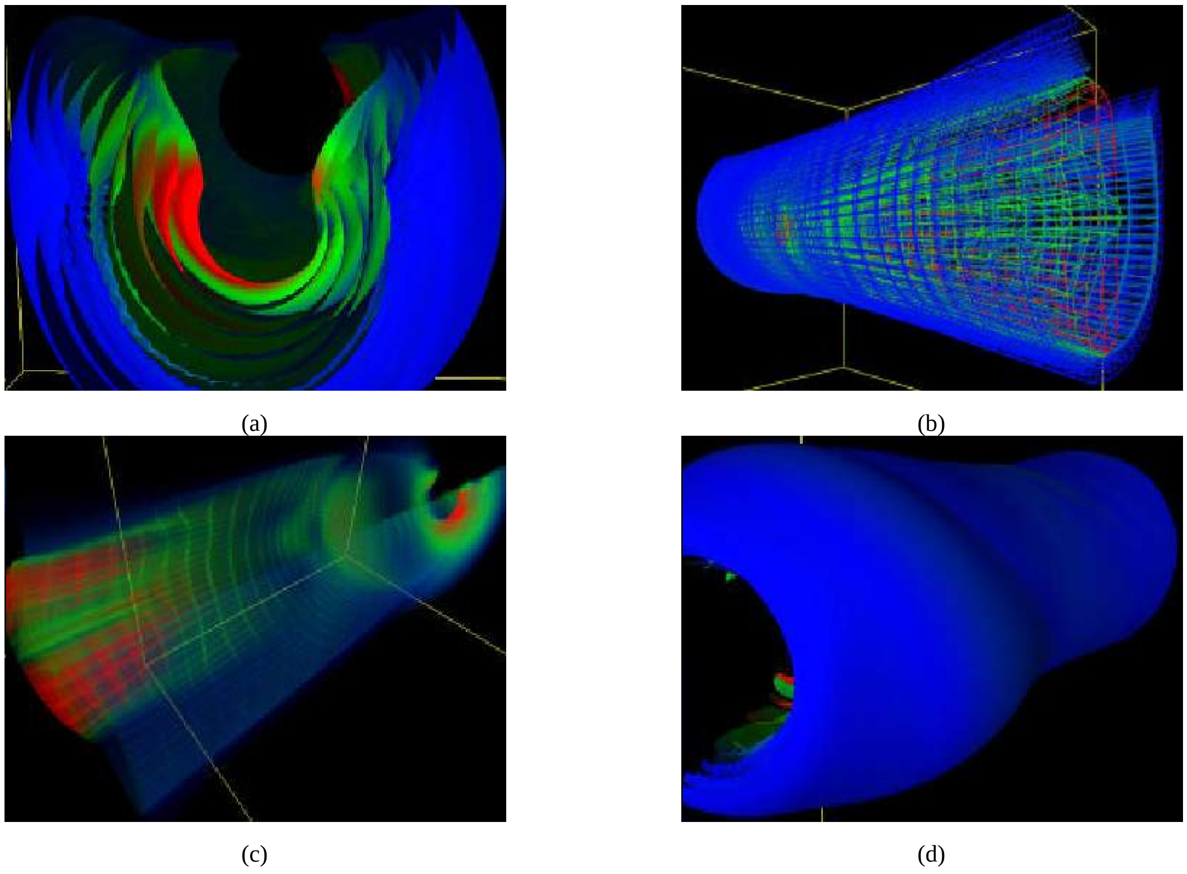


Figure 4: Anisotropic deformation on the cylinder with hemispherical cap rate of strain tensors. (a) is rendered with opaque surfaces; (b) is rendered in wireframe; (c) is rendered with transparent volumes; (d) is rendered with opaque surfaces. All of them are colored with the magnitude of tensor forces.

identify the ratio of distances of one point to its different neighbors, which are important indicators of the anisotropy of the tensor value in a local area. Figure 4(c) is taken from the left side and is rendered in transparent volumes. We clearly see the distribution of tensor force magnitudes in this figure. The red part on the bulge shows high compressive forces over that area. Figure 4(d) is taken from the bottom of the tensor field and rendered with opaque surfaces. In this image, the outer shape of the transformed tensor field is shown. We can see that the front part of the object is much bigger than its rear part, which means the tensor values are larger in the front. The maximal error rate is 6.35%.

7 CONCLUSIONS

Two new deformation based techniques are introduced in this paper. Each of them is an improvement over existing methods. The normal vector deformation method allows the interrogation vector to vary over space and time, and also allow the entire volume to be deformed. It currently suffers the same limitation as hyperstreamlines in that the resulting deformation is different when different layer orientation are used to sweep the volume. We care continuing our investigation at data dependent interrogation vector to address this limitation. The anisotropic deformation does not require an interrogation vector nor a tensor decomposition. It deforms the volume by adjusting lengths connecting a node to its neighboring nodes subject to minimizing the resultant displacement force. This force is obtained by multiplying the local tensor with the displacement vector. However, we note that a general tensor can be decomposed into a symmetric and a rotational component using polar decomposition. And since the rigid rotation does not affect the lengths, the procedure is essentially only visualizing the symmetric portion of the tensor. In addition, this method does not provide an exact solution but rather an approximation. We are continuing to improve the accuracy of this approach. Aside from these two approaches, we are also looking at other methods for visualizing tensor fields using the same idea of letting the tensor field manifest itself in the space it resides in as well as the interaction of matter and energy going through it. Finally, we are continuing to obtain feedback from users on the effectiveness and ways of improving these techniques.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their suggestions on how to improve the paper, and to members of the Advanced Visualization and Interactive Systems laboratory at UCSC for some utility code to get the project going sooner. This work is supported in part by NSF ACI-9908881, NASA Cooperative Agreement NCC2-1260, and LLNL Agreement No. B347879 under DOE Contract No. W-7405-ENG-48.

REFERENCES

[1] E. Boring and A. Pang. Interactive deformations from tensor fields. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proceedings IEEE Visualization '98*, pages 297–304. IEEE Computer Society Press, 1998. Tensor / Flow.

[2] W.C. de Leeuw and J.J. van Wijk. A probe for local flow field visualization. In G.M Nielson and D. Bergeron, editors, *Proceedings IEEE Visualization '93*, pages 39–45. IEEE Computer Society Press, 1993. Flow Visualization.

[3] T. Delmarcelle and L. Hesselink. Visualization of second order tensor fields and matrix data. In *Proceedings of Visualization 92*, pages 316–323, CP–34, 1992.

[4] T. Delmarcelle and L. Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications*, 13(4):25–33, July 1993.

[5] J.A. Ekaterinaris and L.B. Schiff. Delta wing at 40 degrees angle of attack. http://www.nas.nasa.gov/Research/Datasets/Ekaterinaris/index.shtml, 1990.

[6] F R Gantmacher. *Theory of Matrices*. Chelsea Pub Co, 1960.

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

[8] R.M. Kirby, H. Marmanis, and D.H. Laidlaw. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In David Ebert, Markus Gross, and Bernd Hamann, editors, *Proceedings of Visualization 99*, pages 333–340, San Francisco, 1999.

[9] David Laidlaw, Eric Ahrens, David Kremers, Matthew Avalos, Russell Jacobs, and Carol Readhead. Visualizing diffusion tensor images of the mouse spinal cord. In *Proceedings of Visualization 98*, pages 127–134, 1998.

[10] Xundong Liang, Bin Li, and Shenquan Liu. The deformed cube: a visualization technique for 3D velocity vector field. In *Image Analysis Applications and Computer Graphics. Third International Computer Science Conference. ICSC'95*, pages 51–58. Springer, 1995.

[11] D. J. Tritton. *Physical Fluid Dynamics*. Van Nostrand Reinhold Company, 1977.

[12] W.Schroeder, C.Volpe, and W.Lorensen. The stream polygon: A technique for 3D vector field visualization. In *Proceedings: Visualization '91*, pages 126–132. IEEE Computer Society, 1991.