View-Dependent Streamline Deformation and Exploration

Xin Tong, *Student Member, IEEE*, John Edwards, Chun-Ming Chen, *Student Member, IEEE*, Han-Wei Shen, *Member, IEEE*, Chris R. Johnson, *Fellow, IEEE*, and Pak Chung Wong, *Member, IEEE*

Abstract—Occlusion presents a major challenge in visualizing 3D flow and tensor fields using streamlines. Displaying too many streamlines creates a dense visualization filled with occluded structures, but displaying too few streams risks losing important features. We propose a new streamline exploration approach by visually manipulating the cluttered streamlines by pulling visible layers apart and revealing the hidden structures underneath. This paper presents a customized view-dependent deformation algorithm and an interactive visualization tool to minimize visual clutter in 3D vector and tensor fields. The algorithm is able to maintain the overall integrity of the fields and expose previously hidden structures. Our system supports both mouse and direct-touch interactions to manipulate the viewing perspectives and visualize the streamlines in depth. By using a lens metaphor of different shapes to select the transition zone of the targeted area interactively, the users can move their focus and examine the vector or tensor field freely.

Index Terms—Flow visualization, streamline, white matter tracts, focus+context, deformation, occlusion

1 Introduction

Streamlines are commonly used for visualizing three-dimensional (3D) vector and tensor fields. Streamlines show the trajectories of particles moving along the directions of flow and provide insight into intricate flow features such as sinks, sources, saddles, and vortices. When too many streamlines are shown, however, getting a clear view of important flow features without occlusion is difficult. Even though a single streamline does not cause much occlusion compared with higher-dimensional geometry, mixing many streamlines of different depths together can generate a very confusing image.

Although through interactive seeding of streamlines one can control the amount of occlusion and visual clutter, it makes finding specific flow features, and hence understanding their surrounding context, more difficult. To reach a balance between displaying too much information and too little, focus+context (F+C) techniques provide a nice solution. In this paper, we present a streamline deformation technique to achieve an F+C view of 3D streamlines. Earlier streamline deformation approaches [1], [2] deform the 3D space of the flow field. The main drawback of the space deformation approaches is that it is not easy for users to control the deformation of continuous 3D space to remove occlusion completely.

Adjusting transparency is another common method used to expose occluded features [3], [4]. But it is difficult to set the transparency value and define the semi-transparent region so that a clear view of both the F+C objects can be obtained. For example when visualizing a vortex in the Hurricane Isabel dataset, as shown in Figs. 1a and 1b, the vortex-shaped streamlines are originally occluded in (a) and then become visible after making some streamlines more transparent in (b). However, the context of the flow around the vortex is mostly lost if the occluding streamlines are too transparent, and with transparency, the depth relationships among streamlines are more difficult to discern. Furthermore, depth sorting is required to render semi-translucent objects correctly, but it is difficult to sort a large number of line segments in real-time. Cutaway method is another common technique to deal with occlusion. It not only removes all the objects occluding the *focus features*, interesting features that users focus on, but also removes the context features in that region, as show in the example of Fig. 1c.

In our previous paper [5], we presented a view-dependent deformation model for interactive streamline exploration. After users have defined a *focus region* in screen space, streamlines occluding the focus region are deformed and gradually moved away based on two deformation models, a point model and a line model. The point model moves streamlines away from the center of the focus region, while the line model cuts the streamlines along the principal axis of the focus region and moves the streamlines to both its sides. Because occlusion has a view-dependent nature and our deformation is performed in screen space, occlusion can be more effectively removed. Besides, the animation of the deformation gives users the connections between the deformed streamline shapes and their original shapes, and allow users to mentally reconstruct the original shapes as contexts. Compared to the other methods mentioned above, our

- *X. Tong, C.-M. Chen, and H.-W. Shen are with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210. E-mail: {tong, chenchu, hwshen}@cse.ohio-state.edu.*
- *J. Edwards and C.R. Johnson are with the Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112. E-mail: {jedwards, crj}@sci.utah.edu.*
- *P. C. Wong is with the Pacific Northwest National Laboratory, Richland, WA 99352. E-mail: pak.wong@pnnl.gov.*

Manuscript received 30 June 2015; revised 27 Oct. 2015; accepted 13 Nov. 2015. Date of publication 19 Nov. 2015; date of current version 1 June 2016. Recommended for acceptance by T. Dwyer, S. Liu, G. Scheuermann, S. Takahashi, and Y. Wu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2015.2502583

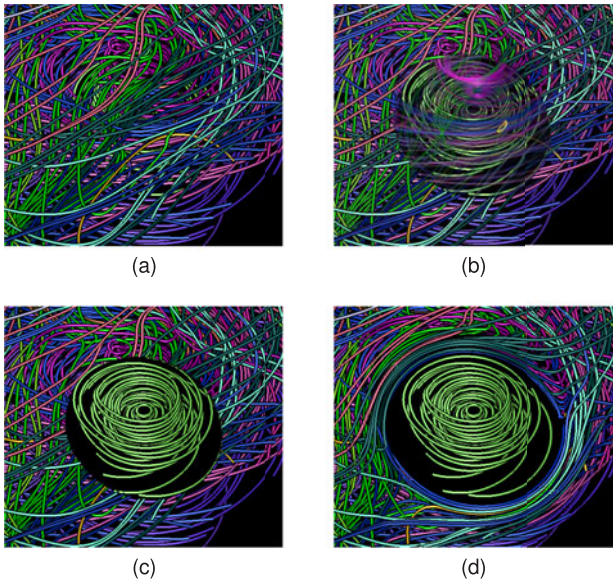


Fig. 1. Three methods are compared to reveal the feature in Hurricane Isabel dataset, a group of vortex-shaped streamlines, originally occluded by other streamlines. (a) original rendering with the features occluded; (b) transparency method is applied; (c) cutaway method that removes both the occluding factors and the contexts; (d) our deformation method is applied, which completely removes the occlusions with context information preserved.

deformation technique can better preserve the context streamlines in the vicinity of the focus feature and through the graduate deformation transition, as shown in Fig. 1d and the accompanying video. As an application of our deformation model, an interactive 3D lens was presented to allow users to freely move streamlines away from selected areas on the screen using both mouse and direct-touch interaction. Two real vector field datasets, Hurricane Isabel and Solar Plume, were used to demonstrate our deformation framework.

In this paper, we extend the previous technique by introducing two new lenses, *layered lenses* and *polyline lens*. After receiving positive feedback from scientists about the techniques presented in the previous paper, we decided to enhance the interactive lens with additional functionality to suit different users' needs. The layered lenses are a set of lenses stacking on top of each other in screen space, each of which deforms the streamlines in their respective ranges in a specific but coherent layer. By deforming different layers differently, we can show the features and contexts of different layers more clearly. The polyline lens is a variation of the previously proposed open blind lens. Instead of cutting with a straight line, the polyline lens can cut the streamlines with a series of connected line segments and deform the surrounding streamlines smoothly to the side. It is much easier to fit features of curvy or concave shapes to the focus region of a polyline lens. To demonstrate the effectiveness of our new lenses, we apply our technique on a tensor field dataset of a brain tumor patient. Our technique is used to deform the white matter tracts generated from the tensor field dataset in order to explore the relationship between the brain tumor and the tracts. Additionally, neurosurgeons and neurologists were invited to evaluate our system and provide valuable suggestions on further studies.

2 Related Work

Overcoming occlusion is an important but challenging problem in 3D visualization. Several approaches have been proposed in the past to avoid or remove occlusion. Li et al. [6] argued that the use of transparency fails to convey enough depth information for the transparent layers. They use cutaways to remove occlusion and expose important internal features. McGuffin et al. [7] used a deformation approach to allow users to cut into, open up, spread apart, or peel away parts of the volumetric data in real time, which makes the interior of the volume visible while preserving the surrounding contexts. An occlusion-free route is visualized by scaling the buildings that occlude the route. Hurter et al. [8] used an interactive dig tool to deform the volumetric data by simply pushing the data points, which reduces occlusion.

In 3D streamline visualization, many streamline selection or placement approaches have been proposed with a goal to minimize occlusion or visual clutter. Mattausch et al. [9] applied magic volume, region of interest driven streamline placement, and spotlights to alleviate the occlusion problem. Li and Shen [10] proposed an image-based streamline generation approach that places seeds on the 2D image plane, and then unprojects the seeds back to 3D object space to generate streamlines. Occlusion is avoided by spreading out streamlines in image space. Marchesin et al. [11] defined the overlap value, the average number of overlapping streamlines for each pixel in the image space, to quantify the level of clutter and then remove the streamlines that have high overlap values on their projected pixels. Lee et al. [12] proposed a view-dependent streamline placement method. In their method, streamlines will not be generated if they occlude regions that are deemed more important, characterized by Shannon's entropy. Another method to alleviate streamline occlusion is to reduce the opacity of the occluding streamlines. Park et al. [3] applied multi-dimensional transfer functions (MDTFs) based on physical flow properties to change the color and opacity of streamlines. Xu et al. [4] proposed to make the streamlines in lower entropy regions more transparent to reduce occlusion. Günther et al. [13] provided a global optimization approach to render streamlines with varying opacity in order to achieve a balance between presenting information and avoiding occlusion. Brambilla [14] measures the degree of occlusion for stream surface and split the surface along a cutting curve to reduce the degree of occlusion. The above occlusion-aware streamline placement methods and transparency modulation methods have their downsides. The problem for the streamline removal methods is that some interesting streamlines may not be shown when they occlude many other streamlines. On the other hand, for the transparency modulation methods, it is difficult to judge the relative depths among the semi-transparent streamlines, and those streamlines can become a distraction. Our F+C streamline deformation method can solve the occlusion problem with better user control while making all the input streamlines easier to see.

F+C techniques have been used by different applications that magnify the focus objects while preserving the surrounding context. The techniques include fisheye views [15], [16], [17] and magnification lens [18], [19], [20], [21]. Magic

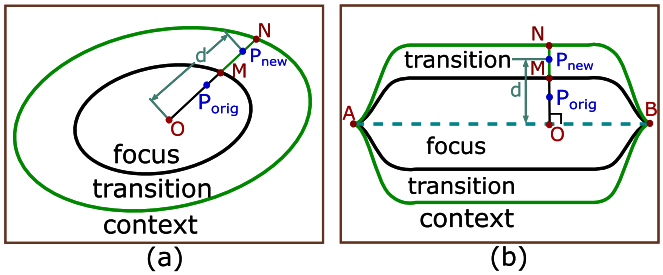


Fig. 2. Sketches of the two deformation models: (a) the point model and (b) the line model. The region inside the black boundary is the focus region. The region between the black and green boundaries is the transition region. The region outside the green boundary is the context region. During deformation, the vertex moves from P_{orig} to P_{new}. In (a) O is the center of the black ellipse, while in (b) O is the intersection point between the line AB and the perpendicular line of AB passing through P_{orig}. M and N are the intersection points between the line OP_{orig} and the two boundaries.

lens [22] changes the presentation of objects to unveil hidden information. In flow visualization, 3D lenses have been applied to show the focus region with greater details [9], [23]. Gasteiger et al. [24] use a magic lens to attenuate the focus attribute while showing the context attribute within the lens. Krüger et al. [25] use 2D lens to control the visibility of features. Van der Zwan et al. [26] blend several levels of detail with a halo-like shading technique to simultaneously show multiple abstractions. NPR lens [27] and the Edgelens [28] interactively distort the features within a 2D lens to emphasize effects and reduce edge congestion, respectively. Among those referenced F+C works, some of them do not solve the occlusion problem. Some methods [9], [23] can reduce occlusion in 3D but do not completely keep the focus objects out of occlusion. Some other methods [24], [25], [26] can completely remove occlusion, but they remove the context information (the occluding objects) at the same time. Only the Edgelens [28] work solves the clutter problem without removing context.

Our approach is more related to the following works with F+C flow visualization using spatial deformation. Correa et al. [1] proposed an illustrative deformation system for F+C visualization of discrete datasets. Deformation is used to expose the internal focus region, and an optical transformation is applied to mark up the context region. Because the deformation is performed in data space, the focus can be occlusion-free for only certain view directions. Tao et al. [2] devised a deformation framework specifically for streamlines. This method deforms the data grid, and generates streamlines based on the deformed grid. It magnifies the streamlines in the focus while compressing the context region. In their method, because deforming space cannot move individual streamlines according to their specific locations, it is more difficult to avoid occlusion from certain view angles. Both Correa et al.'s and Tao et al.'s methods are view-independent, which means the deformation can fail to remove occlusion completely. Besides, both methods require user input of 3D locations on a 2D screen, which makes direct user manipulation difficult. In contrast to these two deformation approaches, our new approach displaces the streamline vertices in 2D screen space and hence can achieve efficient occlusion-free rendering and easy user control for an arbitrary view.

3 Algorithm

The goal of our algorithm is to expose interesting features in the focus region by deforming the occluding streamlines. We use two shape models, the point model and the line model, to perform deformations. Choice of which model depends on the shape of the focus region. These two shape models can generate effective deformation for different shapes of focus region, and is easy for users to control interactively. For each of the two shape models, we design a screen-space deformation algorithm that displaces vertices of the occluding streamlines. The point and line models are the algorithmic underpinnings for our interactive lenses described in Section 4.

3.1 Algorithm Overview

The input to our algorithm is a set of densely distributed streamlines, which can be roughly divided into *focus streamlines* and *context streamlines*. Focus streamlines are what the user is interested in visualizing without any occlusion, e.g., a cluster of streamlines with a similar shape, or a group of streamlines passing through a user-specified region. The remainder are context streamlines. Any streamlines that block the focus streamlines will be deformed and moved to the side. To perform the deformation, our F+C deformation model divides the screen space into three regions: *focus region*, *transition region*, and *context region*, as shown in Fig. 2. The focus region is a user-specified region in screen space that contains the features of interest; the transition region is the area that is immediately adjacent to the focus region used to contain the deformed streamlines; and the context region is the rest of screen space that contains undeformed streamlines. Although streamlines are defined in 3D space, our deformation takes place in 2D screen space, i.e., streamlines are deformed without changing their original depth. For this reason, our shape model described below in Section 3.2 is defined in 2D space.

The goal of the deformation is to compress and move the occluding streamline segments from the focus region to the transition region. To make space for these streamlines, streamline segments that were originally in the transition region will also be compressed and moved towards the outer boundary of the transition region. Essentially, the deformation makes sure that the features of interest in the focus region are occlusion-free to the view. The occluding streamlines are deformed but not removed, providing the context to the focus region. Any other streamlines outside of these two regions remain unchanged.

We have two design goals for our deformation model:

1. The deformed streamline should preserve its shape as much as possible, even though the shape is compressed. In other words, the relative positions of the streamlines and their vertices should be preserved.
2. After the deformation, the vertices should be distributed on the streamline as uniformly as possible. In other words, any two connected vertices on a streamline should not be placed too far from or too close from each other, compared to other pairs of connected vertices. Otherwise, the streamlines will be jagged and a long edge between two connected vertices may cut across the focus region.

In our algorithm, the deformation of a streamline is achieved by displacing its projected vertices in screen space. During deformation, we displace the deformed vertices away from the center of the focus region. The amount of the displacement is determined by each streamline vertex's distance to the center of the focus. We design a *displacement function* to place the deformed streamlines in the transition region, preserving their shapes as much as possible, in order to satisfy our first design goal. In addition, an adjustment is applied to the vertex displacement to make the deformed streamlines satisfy our second design goal.

3.2 Shape Models

We designed two shape models, a point model and a line model, to represent the shape of the focus region. Fig. 2 illustrates these two models. The point model is designed for focus regions that have a circular shape, while the line model is for focus regions that have a linear shape. Both shapes are typical for streamlines. The first section of the accompanying video demonstrates and compares the two shape models. We note that besides the simple regular shapes (point and line), a more complex irregular shape could be used, e.g., a skeleton or principal curve of the streamline cluster and their surrounding curved tube-shaped regions. However, the resulting context streamlines would be distorted, making it hard for users to mentally recover their original shapes. Therefore, they are not considered in this work.

3.2.1 Point Model

As shown in Fig. 2a, the point model is composed of a 2D focus region (the inner black ellipse in the figure), a transition area (the area between the inner black ellipse and the outer green ellipse), and its center O. The inner focus region can also be represented by a convex polygon if desired. The convex polygon and the ellipse-based focus regions have their own advantages. The convex polygon model can more tightly cover the focus streamlines, while the ellipse focus has a relatively smoother and more regular shape and can be represented analytically. The center of the focus region in the point model is the reference point from which the streamlines are moved away.

3.2.2 Line Model

Fig. 2b shows our line model. The line model is composed of a principal axis line (line AB), a linear bounding area immediately outside of the principal axis that represents the focus region, and the transition area that is an expanded area outside of the inner focus. We call this bounding shape *open blinds*, because it looks like two window blinds that are open. Around the two end points A and B, the shape of the open blinds is represented by the $tanh$ function. During the deformation, we cut the streamlines in the open blinds region by the axis line and move the streamlines to both sides of the axis line along the direction normal to AB, until the focus region is clear.

The point model and the line model can be selected based on the shape of streamlines automatically. If we apply the point model to straight streamlines distributed in an area as in Fig. 2b, for example, some vertices will have to travel a long distance to the region around the point A or B, resulting in a large distortion. On the other hand, the point model is good for circular focus regions to avoid unnecessary cutting of the streamlines happening in the line model. The overall shape of the focus streamlines determines which model to use. To measure the overall shape, we use the roundness of the focus region, which is a minimum enclosing ellipse of the focus streamlines. If the major radius of the ellipse is much larger than the minor radius, i.e., low roundness, then we use the line model; otherwise, we use the point model.

3.3 Deformation Model

Our method achieves the deformation by displacing its vertices iteratively to preserve the smoothness of the streamlines throughout the whole process, as required by our second design goal. This means the relative positions of the vertices on a streamline need to be updated constantly; otherwise, the distance between two adjacent vertices on a streamline can become too large, and consequently, the line segment between them can cut across the focus region and still cause occlusion. To achieve this, the force-directed algorithm [29], which considers the relative positions of the points by moving them iteratively to generate the final layout, inspires the design of our approach. In our method, a vertex does not just follow a linear path and move towards a single predetermined direction. Instead, the deformation is computed through multiple iterations and generates an animation sequence. In each iteration, the vertex adjusts its moving direction so that its relative position is preserved throughout the animation. This animation sequence provides the context, and the final layout of the streamlines provides an occlusion-free view of the focus region, as shown in Fig. 1d.

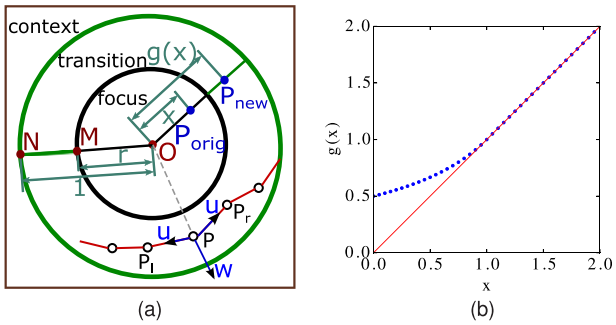
When a streamline is deformed, in each iteration the position of a vertex on the streamline is modified based on two considerations. First, the vertex should gradually move out of the focus region. Second, the vertex should not be placed too far away from its neighboring vertices. Based on these considerations, we control the displacement of a vertex using two subcomponents, each of which is represented by a speed and a moving direction. Mathematically, the vertex movement can be written as:

$$P' = P + v \cdot \vec{w} + v_c \cdot \vec{u}, \tag{1}$$

where P' is the new position of the vertex, P is the old position, $v \cdot \vec{w}$ represents the movement that moves the point out the focus region, and $v_c \cdot \vec{u}$ makes sure that the new point position is not too far from its neighbors. Hereafter we refer to $v \cdot \vec{w}$ as the *major displacement*, and $v_c \cdot \vec{u}$ as the *minor adjustment*. The two directions $\vec{u}$ and $\vec{w}$ are shown in Fig. 3a. Below we explain each of the terms in detail.

3.3.1 Major Displacement

At each iteration, the streamline vertex moves away from the focus region along the direction $\vec{w}$ at a speed of v. The moving direction $\vec{w}$ is related to the underlying shape model in use. For the point model, as shown in Fig. 2a, $\vec{w}$ is from the centroid O to the current vertex position P, i.e., $\vec{w} \parallel OP$, which is also shown in Fig. 3a. If the line model is used,



(a) (b)

Fig. 3. (a) Illustration of the point model in the normalized space. $\vec{w}$ and $\vec{u}$ are the two displacement directions for the point at P. P_l and P_r are the two vertices connected to P on this streamline. (b) Blue dotted line: normalized displacement function g in Equation (10) when $r = 0.5$. Red line: a reference displacement function $F(x) = x$, which gives no displacement for the entire domain.

shown in in Fig. 2b, $\vec{w}$ is the normal direction of the line AB. If we draw a line through point P and perpendicular to AB, it intersects with AB at O, and we have $\vec{w} \parallel \vec{OP} \perp \vec{AB}$. For both shape models, we generalize the definition of $\vec{w}$ as:

$$\vec{w} = normalize\left(\vec{OP}\right). \tag{2}$$

The moving speed v determines the amount of major displacement in one iteration. Assuming P_{orig} is the vertex's original position, and d is the distance between O and the vertex's final position P_{new} after deformation, i.e., $d = |\vec{OP_{new}}|$. For the vertex to reach a distance of d from O, the speed of the movement for P is determined by how much the point has yet to travel, that is:

$$v = (d - |\vec{OP}|) \cdot \alpha, \tag{3}$$

where α is a constant that has a value in $(0, 1)$. It controls the magnitude of the moving speed. An empirical value of α is 0.01. Because as P moves away from O, $|\vec{OP}|$ keeps increasing, and thus the speed of v keeps decreasing, until the vertex P stops moving and arrives at its final position P_{new}.

From Equation (3), we know that d determines the final position of each streamline vertex after the deformation. Because we want to compress and preserve the shape of the streamline, we control the value of d using a monotonically increasing function to transform the original distance $|\vec{OP_{orig}}|$ to a larger value d. This function, denoted as g, takes a normalized value of $|\vec{OP_{orig}}|$ as its input and has the general form:

$$d = g\left(\frac{|\vec{OP_{orig}}|}{|\vec{ON}|}\right) \cdot |\vec{ON}|, \tag{4}$$

where $|\vec{ON}|$ is the distance between O and the outer boundary of the transition region along the moving direction of $\vec{w}$, as shown in Fig. 2. $|\vec{OP_{orig}}|$ is normalized to be between 0 and 1 by dividing its value by $|\vec{ON}|$.

Fig. 3a is an illustration of the point model similar to Fig. 2a, marked with several normalized distances to illustrate the displacement function $g(x)$. The normalized value of $|\vec{ON}|$ is 1. We define r as the normalized value of $|OM|$, i.e., $r = \frac{|\vec{OM}|}{|\vec{ON}|}$. Before deformation, vertices on the non-deformed streamline segments distribute over both the focus and transition regions, so $|\vec{OP_{orig}}|$ varies in the range $[0, |\vec{ON}|]$, i.e., $\frac{|\vec{OP_{orig}}|}{|\vec{ON}|}$ varies in the range $[0, 1]$. After deformation, the transformed vertices will all go to the transition region, so d varies in the range $[|\vec{OM}|, |\vec{ON}|]$, i.e., $\frac{d}{|\vec{ON}|}$ varies in the range $[r, 1]$. Essentially, g monotonically transforms a value in $[0, 1]$ to a larger value in $[r, 1]$.

3.3.2 Displacement Function

Instead of designing the displacement function g as a linear function, we apply the transformation function of fisheye lens [16] to design a non-linear displacement function g to control the speed of streamline vertices as discussed in the previous section and produce a smoother deformation across the region boundary. Here we assume a point model with a circular boundary shown in Fig. 3a to explain the idea. Below, we first give the design goals of the function g, and then solve g.

Our first criterion is that, as shown in Fig. 3a, a vertex located at O should be moved to the inner boundary of the transition region at M, and a vertex located at the outer boundary of the transition region at N should remain on the outer boundary. So we have:

$$g(0) = r, \tag{5}$$

$$g(1) = 1. \tag{6}$$

Second, the function g must be a monotonically increasing function to make sure that the deformed streamlines and their vertices have the same relative positions to O after the deformation. So we have:

$$\frac{dg(x)}{dx} > 0. \tag{7}$$

$\frac{dg(x)}{dx}$ describes the amount of distortion in the deformed space at a point whose distance to O is x. We know that $\frac{dg(x)}{dx}$ is 1 for any points in the context region because they will not move. To ensure that the amount of distortion smoothly changes from the transition region to the context region at their boundary point N in Fig. 3a, $\frac{dg(x)}{dx}$ should be continuous at that point. So we know

$$\left.\frac{dg(x)}{dx}\right|_{x=1} = 1. \tag{8}$$

Finally, our last design criterion is that, from a position near O to a position farther away from O, the amount of distortion should also change monotonically. The value of $\frac{dg(x)}{dx}$ should monotonically increase from a value less than 1 to the value 1 when x changes from 0 to 1. The change of the distortion amount is the second derivative of displacement function $\frac{d^2g(x)}{dx^2}$. So we get:

$$\frac{d^2g(x)}{dx^2} > 0. \tag{9}$$

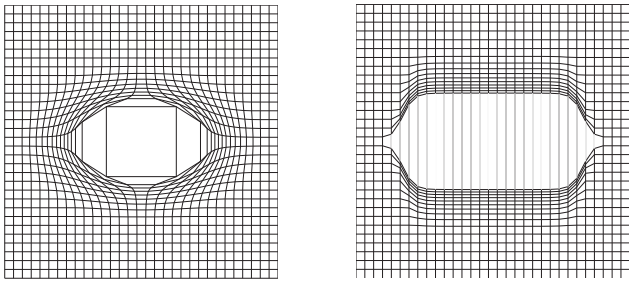


Fig. 4. Deformed grids using two shape models with $r = 0.5$.

Combining the four criteria from Equations (5) - (9), we can solve the displacement function g. There is more than one solution for g, and we use the simplest one:

$$g(x) = \frac{(r-1)^2}{-r^2 x + r} - \frac{1}{r} + 2 \qquad x \in [0,1]. \tag{10}$$

To show that the above function satisfies the four criteria, we plot Equation (10) in Fig. 3b. The figure clearly shows that Equations (5) - (9) are satisfied.

If we use the displacement function g to move a regular grid with our two shape models, we get the deformed grids shown in Fig. 4. Note that we create a void focus region at the center of the grid, because $g(x) \geq g(0) = r$ for $x \in [0, \infty)$; all the streamline vertices will be cleared out of the focus region. We also notice that for the same amount of distortion (same value of r) in the space we can create a larger void space with the line model shown in Fig. 4b than the point model shown in Fig. 4a. Besides, the deformed grid in the point model shows more stretching but less compression than the deformed grid in the line model.

3.3.3 Minor Adjustment

The minor adjustment plays an important role in making the vertices uniformly distributed on the deformed streamline and thus satisfies the second design goal of our deformation model. During our deformation process, some edges can be stretched more, which makes those portions of the streamline jagged. Furthermore, the long edge can cut across and hence still occlude the focus region, which is undesired.

As shown in Fig. 3a, the vertex P is connected to two vertices at point P_l and point P_r with two edges. A local approach to uniformly distribute the vertices over the streamline is that each vertex moves towards the farther one of the two neighboring vertices through multiple iterations so that the relative positions among the vertices are preserved. This shortens the longest edge in each iteration, and eventually no edge is much longer than the other edges.

The minor adjustment $v_c \cdot \vec{u}$ is a product of a constant adjustment speed v_c and an adjustment direction $\vec{u}$. $\vec{u}$ is a normalized direction parallel to the longer connected edge, which is defined as:

$$\vec{u} = \begin{cases} normalize(\vec{PP_l}), & \text{if } |\vec{PP_l}| > |\vec{PP_r}| \\ normalize(\vec{PP_r}), & \text{if } |\vec{PP_l}| < |\vec{PP_r}| \\ \vec{0}, & \text{if } |\vec{PP_l}| = |\vec{PP_r}| \end{cases}.$$

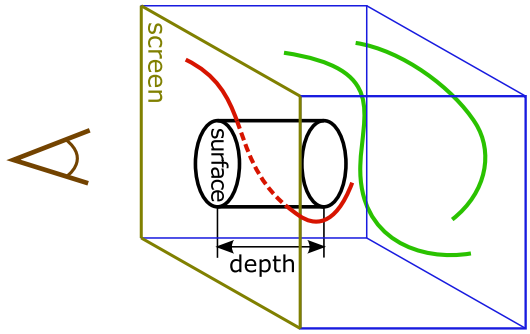


Fig. 5. The 3D lens. The blue cube denotes the 3D space. The yellow square denotes the 2D screen space. The lens has an ellipse-shaped surface on the plane of the screen. Inside the cube, there are three streamlines.

Note that the minor adjustment may move the vertex in a direction other than the direction of major displacement $\vec{OP}$, which ends up changing $\vec{w}$ in the next iteration. This change is not recoverable for the later iterations. Therefore, when the viewpoint or the focus region is changed during the deformation, if we continue the deformation with the new value of $\vec{w}$ or $|\vec{ON}|$, then we can still keep the focus region occlusion-free, but we may not be able to preserve the shape of deformed streamlines in the transition region. The solution is to recover the streamlines' original positions and redo the deformation from the first iteration.

4 Interactive Deformation

In this section we introduce a streamline exploration tool, interactive 3D lens, based on the deformation algorithm described above. To overcome the occlusion problem, the lens can be placed anywhere in the image space with an adjustable depth to peel away the occluding streamlines layer by layer. To enhance the experience of user interaction, we use a direct-touch technique to allow users to control the lens directly on the screen with multi-touch gestures.

4.1 Interactive 3D Lens

The interactive lens is useful when users want to freely explore the computed streamlines. The lens defines a focus region with a certain depth range. Any streamline that is entirely or partly under the lens and closer to the viewer than the far side of the lens are treated as context streamline and will be moved out of the focus region in the screen space. The other streamlines are all treated as focus streamlines that will not be deformed, even when they are not the interesting features.

We design our interactive 3D lens as a 3D cylindrical object, shown as the black cylinder in Fig. 5. The lens resides in screen space with depth defined, shown as the blue cube in the figure. The axis of the cylinder is perpendicular to the screen, i.e., parallel to the z axis, and the top surface of the lens is parallel to the XY plane. The length of the cylinder is used as the *lens depth*. For the point model, the lens has an elliptical surface; while for the line model, the lens surface has an open blind shape. As the example shows in Fig. 5, there are three streamlines (one red and two green), but only the red streamline that intersects with the lens will be deformed. In the deformation, all the vertices on the red

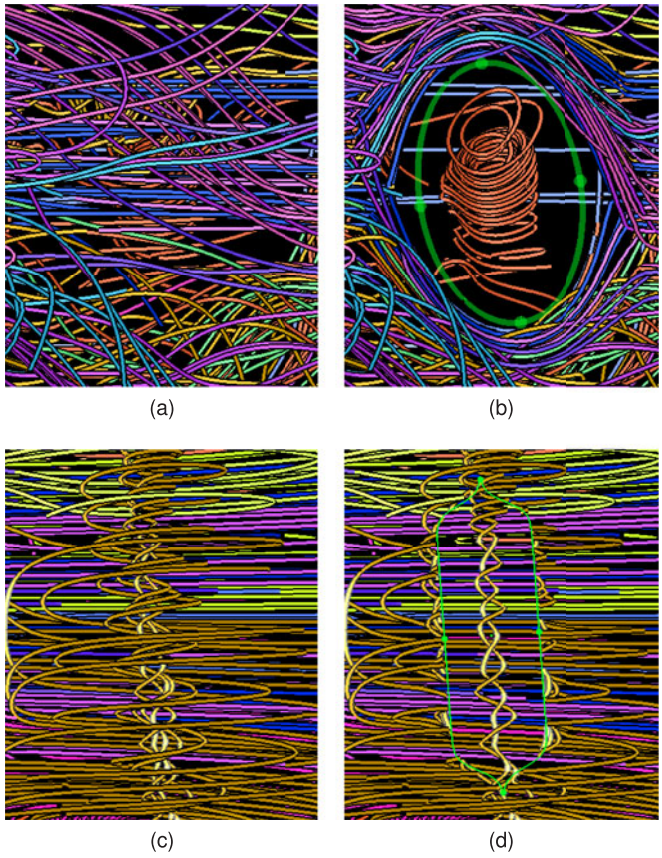


Fig. 6. A point-model lens is applied to the streamline visualization in (a) to push layers apart and reveal the hidden red curled vortex in (b). A line-model lens cuts up a flow field in (c) into two halves and pushes them aside to expose the olive colored helix twisted vortex in (d).

streamlines will be moved out of the surface region in screen space, even for the right tail of the red streamline that is not inside the cylinder.

Figs. 6a, 6b, 6c, and 6d show examples of the interactive 3D lens using the point and the line deformation models. In Figs. 6a and 6b, we use a lens with the point deformation model to move the straight streamlines in the front away and reveal the vortex gradually through animation (please see the accompanying video). In Figs. 6c and 6d, a lens with the line deformation model is used to break the outside of the vortex in two, so that the inner structure, which was previously occluded, now becomes visible. Even though the streamline around the inner structure only partially intersects with the 3D lens, we treat the entire streamline as the context streamline and let all the vertices on it be deformed out of the screen space focus region in order to ensure the continuity of deformation on this streamline.

In the interactive system, users can use the mouse buttons to modify the size, shape, and orientation of the lens surface on the screen to specify the focus region. In addition, users can use the mouse wheel to change the lens depth to explore the streamlines at different layers in z direction.

In summary, the interactive 3D lens uses our view-dependent deformation model to explore and reveal hidden streamlines in the flow field. By using the lens, users can freely move their focus to different locations and change the shapes of the lens interactively to search for interesting streamlines. Because the lens is always perpendicular to the image space, users can rotate the field and change the depth of the lens to explore the 3D space. Users can progressively discover the features at different depths even when the features are deeply buried inside the field, or move the lens around to explore different parts of the focus streamlines when they are very long. Because our deformation can be performed interactively, users can go back and forth to replace the deformed streamlines to enhance their understanding of the 3D features.

4.2 Specialized Lenses

The interactive 3D lens described above has received positive feedback and suggestions of extension from researchers. Thus, we enhance our lens with additional features to meet special needs in feature exploration. Streamline features look different at different depths and screen locations. A good interactive lens should adapt itself to those different features in order to better preserve the context and accelerate the process of finding features. In this section, we propose the *layered lens* and the *polyline lens*, which can explore features at different depths and different screen locations with improved flexibility and adaptability.

4.2.1 Layered Lenses

In camera space, since streamlines in different layers have different shapes and orientations, a lens used for the top-most layer may not be able to reveal features clearly at a different depth layer. For example, when users use the open blind lens to cut the streamlines and move them to the side, they usually want to make the cut direction follow the trajectories of the streamlines. If the directions of the streamlines vary in different layers, the cut directions should also be different in different layers. Another reason to have layered lenses is that users usually want to explore the data in a wider screen area first at the top layers, and then focus on a smaller screen area of interest to inspect the inner structures. To make the deformed streamlines on the top layers unchanged as the context while exploring the smaller scale features in the inner layers, our layered lenses can make different screen sizes of focus regions in different depths and present the users with multiple layers of contexts.

The layered lenses are composed of a set of regular lenses with increasing depths and decreasing screen sizes, as shown from left to right in the Fig. 7. The lenses are connected to each other in the depth direction. Between two adjacent lenses, the screen area of the lens with a larger depth is contained within the screen area of a smaller depth lens. A streamline passing through more than one lens will be deformed by the lens with the smallest depth, because deforming streamlines by multiple lenses can result in large distortion. Also, lenses with smaller depth have larger screen size, and thus can create a larger void region to look through. For example in Fig. 7, the green streamline intersects with both the green lens and the blue lens. Because the green lens has a smaller depth and larger screen size, the green streamline will be deformed by the green lens but not the blue lens. After the deformation, the streamlines originally located in the three lenses will be removed from the three 3D focus regions. Then, users will be able to see the yellow focus streamline and also see the three deformed streamlines (red, green and blue streamlines) as contexts in different layers.

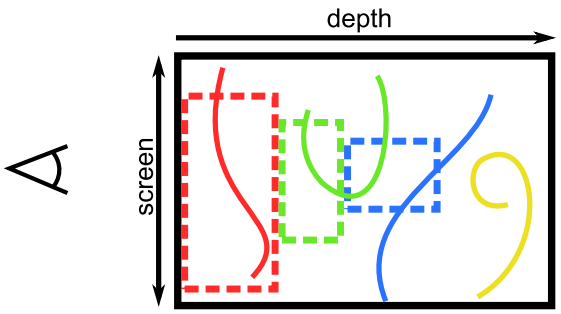


Fig. 7. Layered lenses are composed of three connected lenses, shown as dash line frames in three different colors. The red lens has the smallest depth and largest screen area, while the blue lens has the largest depth and smallest screen area. The streamlines will be deformed by the lens that has the same color with the streamline. The yellow streamline is the focus feature and will not be deformed.

To demonstrate the use of our layered lenses, we apply it on white matter tracts which are generated from a brain diffusion tensor dataset using the *3D Slicer* software [30], [31]. This dataset has a resolution of $256 \times 256 \times 51$ with a 3×3 tensor matrix on each grid point. Because this data is from a patient with a brain tumor, we visualize the tumor as a polygonal surface in addition to the while matter tracts. By visualizing the streamlines and the surface together in Fig. 8, users can explore the tracts, the tumor and their relationships. From the figure, we can see the white brain tumor through the three lenses of different sizes and orientations. We give the deformed streamlines different colors according to their affecting lens in order to differentiate them from the unchanged ones and the ones deformed by other lenses. The biggest lens is closest to the viewer and the color of its deformed streamlines is red; while the smallest lens is the deepest inside the volume and has yellow deformed streamlines. With the layered lens, different layers of streamlines can coexist in the same image, which gives users better context while exploring the 3D space.

4.2.2 Polyline Lens

As said previously, having a lens with an irregular shape, e.g., an arbitrary polygon, is not desired because the affected streamlines will be distorted too much and tend to conform to the shape of the lens. On the other hand, a feature can have an irregular shape, which can be difficult to inspect in detail by either the ellipse lens or the open blind lens. In order to allow the focus region to have more general shapes, we propose the polyline lens, which supports more complex shapes with relatively little distortion. It divides the streamlines by a series of connected straight lines, or a polyline, and pushes the context streamlines to the side. It works similarly to the open blind lens, but with more flexible shapes and can easily fit a curvy streamline.

The deformation of the polyline lens mostly follows the deformation of the line model, except for the vertices near the joint of two connected line segments. For example in Fig. 9, polyline AO_1B is part of a polyline lens, which cuts the streamlines and pushes them out of the focus region. Our line model cannot be used to define the movement of the vertices in the red and green regions, because the red/green focus region has completely different shape with the red/green transition region, i.e., one is a triangle and the other is a trapezoid. For example in Fig. 10a, the red region with white dotted

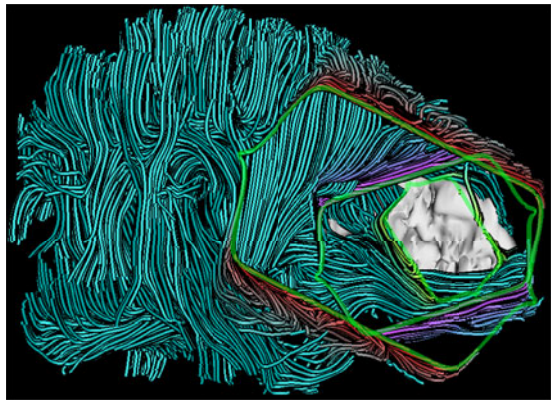


Fig. 8. Using layered lenses to open the brain tumor data set. Deformed streamlines are colored by lens and deformation magnitude, e.g. red streamlines are deformed by the top-most lens, and regions of darker red are deformed more than regions of lighter red.

texture is a triangular focus region; while the red region with checkerboard texture is a trapezoidal transition region. By simply using line model and pushing vertices along the direction perpendicular to the region boundaries, the streamlines in the dots region plus checkerboard region cannot be uniformly fitted in the checkerboard region. To give another example, there is a blue streamline in the red region in Fig. 10a and one in the green region in Fig. 10b. When they follow the line mode and move along the arrow dash lines, the middle two vertices will diverge in the red region and converge in the green region. As a result, the deformed streamline vertices in the red/green transition regions are either too sparse to be smooth (in the red region) or too congested to be visible (in the blue region).

To achieve a smooth and continuous deformation around the joint area of the polyline lens, we want to compress the red region space into its upper transition region, and

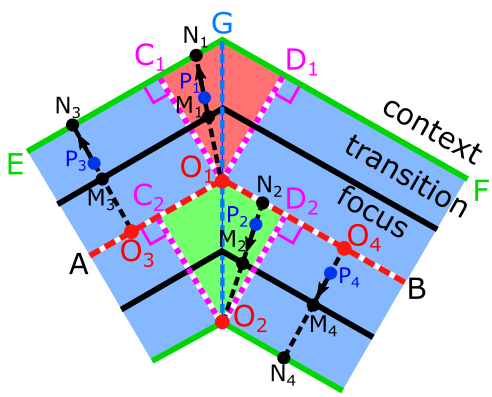


Fig. 9. Illustration of polyline lens. The red dashed lines, AO_1 and O_1B, are two connected line segments that compose a polyline lens. The region inside the black lines is the focus region. The region between black lines and the green lines are the transition regions. Outside the green lines are the context regions. The deformation regions of two line segments intersect on line GO_2. Draw lines from point O_1 perpendicular to line EG and FG, which intersect at point C_1 and D_1, respectively. Draw lines from point O_2 perpendicular to line AO_1 and BO_1, which intersect at point C_2 and D_2. The deformation region in the entire domain is divided into three regions, marked with red, green and blue background colors. P_1, P_2, P_3 and P_4 are four vertices located in three different color regions. The black arrows show their moving directions during deformation. The extension lines of the directions are the black dash lines, which intersect with the region boundaries at $M_{1,2,3,4}$ and $N_{1,2,3,4}$.

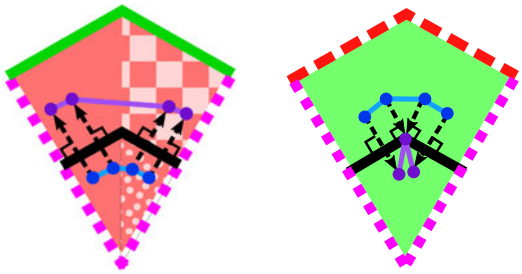


Fig. 10. Magnified views of the red and green regions in Fig. 9. The blue streamlines become the purple streamlines after deformation.

compress the green region space into its lower transition region. We notice that such compression will cause the vertices to move in a radial direction. For the red region, vertices move away from point O_1; while for the green region, vertices move towards point O_2. Thus, we can apply the point model in the two regions in the joint areas. For the red region, we can directly apply the point model and push vertices away from the center O_1. On the other hand in the green region, instead of pushing, vertices are pulled towards the center O_2.

To determine how far a vertex should move, we still use our displacement function $g(x)$ to compute the vertex's desired distance to the focus region center by Equation (4). For example, there are four vertices $P_{1,2,3,4}$ in Fig. 9, which are from the regions of three different colors and follow the four black arrows to move.

Vertex P_1 is in the red region and follows the point model. It is pushed away from O_1 until reaching a distance of d_1 from O_1, where

$$d_1 = g\left(\frac{|\vec{O_1P_1}|}{|\vec{O_1N_1}|}\right) \cdot |\vec{O_1N_1}|. \tag{11}$$

Vertex P_3 is in the blue region and follows the line model. After deformation, its distance to O_3 is

$$d_3 = g\left(\frac{|\vec{O_3P_3}|}{|\vec{O_3N_3}|}\right) \cdot |\vec{O_3N_3}|. \tag{12}$$

If P_1 and P_3 are very close to each other at the boundary of red and blue region, i.e., line O_1C_1, then we have $|\vec{O_1P_1}| = |\vec{O_3P_3}|$ and $|\vec{O_1N_1}| = |\vec{O_3N_3}|$. From Equations (11) and (12), we got $d_1 = d_3$. This means that the deformation is continuous around the boundary of the red region and the green region.

Vertex P_2 is a point in the green region and follows the point model. Equivalent to pulling towards O_2, we can think of it as pushed away from N_2 and reach a distance of d_2 from N_2, where

$$d_2 = g\left(\frac{|\vec{N_2P_2}|}{|\vec{O_2N_2}|}\right) \cdot |\vec{O_2N_2}|. \tag{13}$$

Same as P_3, Vertex P_4 follows the line model and is pushed away from O_4 at a distance of

$$d_4 = g\left(\frac{|\vec{O_4P_4}|}{|\vec{O_4N_4}|}\right) \cdot |\vec{O_4N_4}|. \tag{14}$$

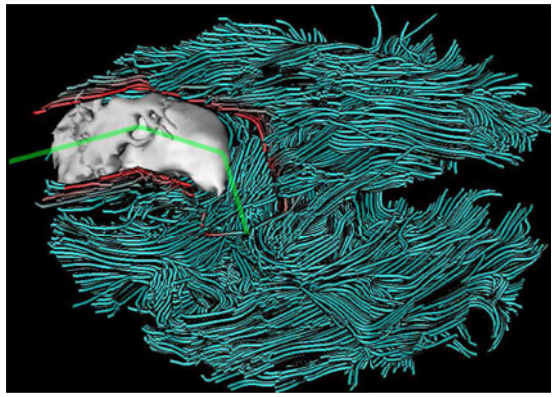


Fig. 11. Using the polyline lens to open the brain tumor data set.

If P_2 and P_4 are very close to each other around the boundary of the green region and the blue region, i.e., line O_2D_2, then we have $|\vec{N_2P_2}| = |\vec{O_4P_4}|$ and $|\vec{O_2N_2}| = |\vec{O_4N_4}|$. From Equations (13) and (14), we know that $d_2 = d_4$, which means the deformation around the boundary of the green region and the blue region is continuous.

Because the red region and the green region do not share a boundary, we do not need to verify the deformation continuity between them. Although our polyline lens uses both the point and line models in different deformation regions, its deformation is still continuous at the boundary of the regions using different deformation models.

Fig. 11 demonstrates a visualization of the brain tumor in the white matter tracts with the polyline lens. A polyline lens is drawn by the user to make it follow the curvy shape of the tumor. Streamlines around the lens are cut and pushed away from the lens region. The deformed streamlines around the joint of two connected line segments are smoothly deformed and continuously connected to the surrounding deformed streamlines. Compared to the regular ellipse lens and open blind lens, the shape of the polyline lens can closely follow the tumor's shape, which distorts the streamline less and preserves the context better.

4.3 Direct-Touch Interaction

Because our deformation model is in screen space, it would be intuitive to interact with the streamlines directly on a direct-touch display, so that users can use the screen as an interaction space and pull away the occluding streamlines with their fingers. It has been reported that touch-based visualization can benefit the users with its ability of direct manipulation and smooth interaction [32]. In our system, users cut and move the streamlines with the interactive 3D lens, which requires users to specify its location and shape. Six degrees of freedom (DOF) need to be supported for the lens: 2D translation on the screen, 1D angle representing the orientation of the lens, 1D scaling factor for the two radii of the ellipse shape lens surface, and the 1D lens depth.

A traditional way to specify the 6DOF of the lens is through mouse interaction, i.e., adjusting the lens surface screen position and shape with mouse dragging, and adjusting lens depth with mouse wheel scrolling. With a multi-touch display, we can replace all the mouse interaction with more intuitive and efficient direct-touch interaction, as described by the following. With one finger, users

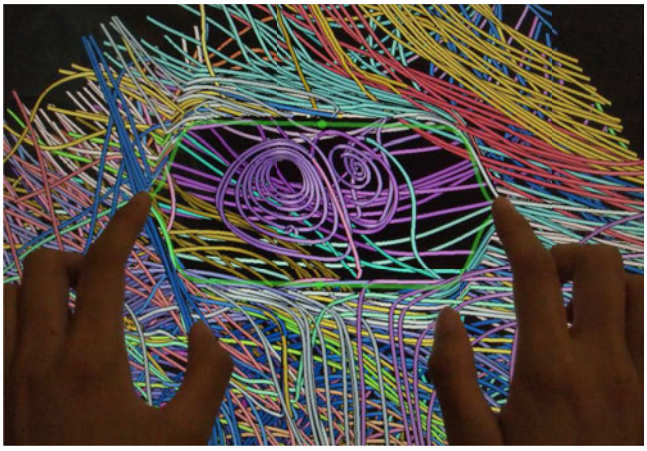


Fig. 12. Placing an open blade shape lens with two fingers on a multi-touch display.

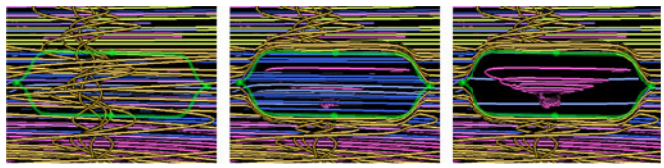
can change 1DOF or 2DOF of the lens at a time, i.e., translation or rotation or scaling. When two fingers are placed on the boundary of the lens and move, users can change 4DOF of the lens at the same time. Swiping two fingers changes the 2D screen location of the lens; rotating two fingers changes the 1D lens orientation; and pinching two fingers changes the size of the lens. Pinching out/in two fingers inside the lens allows users to increase/decrease the 1DOF lens depth and simulates pushing away or pulling back the streamlines. Users can also drag one or two fingers on screen to specify a cutting line of an open blade lens. This touch interaction simulates the process of cutting streamlines with finger tips, as shown in Fig. 12. The Hurricane Isabel dataset is used in Fig. 12. Hurricane Isabel was a strong hurricane in the west Atlantic region in September 2003. The resolution of the dataset is $500 \times 500 \times 100$. An efficient GPU implementation of the streamline cutting process makes the cutting effects responds to the cutting gestures in real-time. The viewpoint navigation is also enabled with multi-touch gestures. The accompanying video demonstrates the interactions with the multi-touch display.

5 Case Study

With the proposed deformation algorithm, as well as the interactive 3D lens and direct-touch interface, we design an interactive system for streamline exploration. We perform two case studies using our technique with two types of data: vector field data and tensor field data.

5.1 Streamlines from Vector Fields

In this section, we provide a case study that uses our system to explore the Solar Plume dataset in three different ways: exploring the streamlines at different depths, from different view directions, and at different locations. In this case study, we provide two ways to define focus streamlines, first as streamline bundles in Section 5.1.2, and second as streamlines passing through a user-specified region in Section 5.1.3. Note that users are free to use other methods to define the focus streamlines depending on their needs. Alternatively, when the 3D lens is used, users are not required to define their focus streamlines but can simply move the lens around the 3D space and search for them.



(a) (b) (c)

Fig. 13. Exploring streamlines at different depths by the interactive 3D lens with different lens depths.

The Solar Plume dataset comes from a simulation of the solar plume on the surface of the Sun. The resolution of the dataset is $126 \times 126 \times 512$. A demonstration of this case study is shown in the accompanying video.

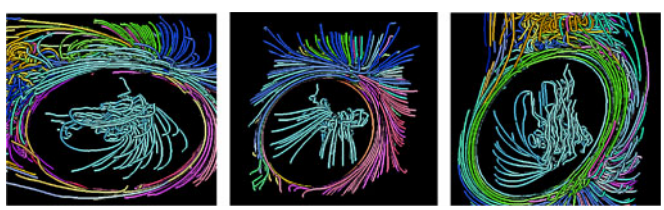
5.1.1 Exploration with Different Depths

Streamlines of different depths may be projected to the same screen location, and those with smaller depths will occlude the ones with larger depths. Our interactive 3D lens can help show the streamlines at all different depths with reduced occlusion. To do this, we can exploit the 3D lens that uses the line deformation model with open blinds to explore the streamlines, as shown in Fig. 13. In Fig. 13a, initially the lens does not touch any streamlines so no streamline is deformed. We see a vertical vortex-like object close to the surface of the volume. We push the lens into the volume by increasing the lens depth, and then we see the blue horizontal streamlines in the center of the volume, shown in Fig. 13b. Finally in Fig. 13c, we increase the lens depth even more to remove the straight streamlines and a funnel shaped vortex is revealed in the back of the volume.

5.1.2 Exploration with Different View Directions

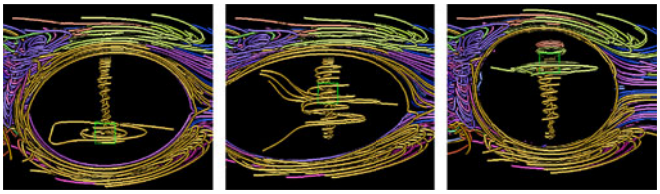
A 3D object may look very different from different views; hence it is important to view a 3D feature from different view directions to obtain a complete understanding of its shape. Because our deformation model is view-dependent, we can remove the occlusion regardless of the view direction. Furthermore, users can get different context information from different views.

In Fig. 14, we treat the streamline bundle as the focus streamlines and view them from three different sides of the volume. A streamline bundle is a cluster of streamlines that are similar in location and shape. We measure the similarity between the streamlines with the *mean of closest point distance* [33], and cluster the streamlines using hierarchical clustering. The focus region is represented by the convex hull or the minimal enclosing ellipse of the focus streamlines. The streamlines not in the bundle, including the streamlines behind the bundle, are the context streamlines and will be moved out of the focus region.



(a) front (b) left (c) top

Fig. 14. Views of a bundle from different view directions.



(a) (b) (c)

Fig. 15. Exploring flow features at different locations. The streamline picking cube moves bottom up from (a) to (c).

This bundle shows a turbulent flow structure with straight tails. From the front view of the bundle in Fig. 14a, a yellow vortex is preserved at the upper-left of the figure in the context region. In the left view of the volume (Fig. 14b), the contexts appear to be stretched long streamlines, which look very different from the front view. In the top view of the bundle (Fig. 14c), we can see some curvy vertical streamlines. The yellow vortex visible in the context of Fig. 14a is again visible from this view, which has a more complete shape than that in Fig. 14a.

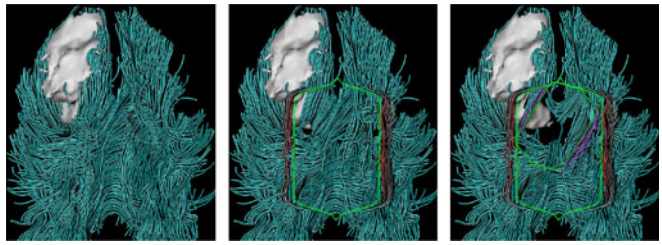
Fig. 1 is another example of streamline bundles using the Hurricane Isabel dataset. In the accompanying video, we also use the Isabel dataset to explore the bundle from different view directions.

5.1.3 Exploration with Different Locations

Users sometimes are interested in the flow behavior in a particular spatial region. In our system, users are allowed to place a small axis-aligned cube in the domain, and then the streamlines passing through this region are selected as the focus streamlines. The minimal enclosing ellipse of the focus streamlines defines the focus region. Here we illustrate the exploration of the streamlines from different locations around a vortex. In Fig. 15, the streamlines passing through the selected location indicated by the green cube are shown. In Fig. 15a, we place the cube at the bottom of the space. A narrow vortex is selected and shown with some wider vortex-shaped streamlines on the side. This image tells us that the flow passing through the selected region extends to the top and the side of the surrounding area. When we move the cube up, a new set of focus streamlines is selected as shown in Fig. 15b. From the tails of the focus streamlines on the left, we know that this selected location is connected to the left side of the flow. We move the cube to the top of the volume, as shown in Fig. 15c. As can be seen, the flow behaviors are different on the top and the bottom of the regions. The vortex on the top is located in a small region, while the vortex at the bottom extends to a wider area. Note that the three images in Fig. 15 all preserve the context features, such as the purple vortex on the left and the horizontal straight streamlines on the right.

5.2 White Matter Tracts from Diffusion Tensor Imaging

Diffusion tensor MRI (DTI) provides directional diffusion information that can be used to estimate the patterns of white matter connectivity in the human brain. Diffusion tensor imaging (DTI) and white matter tract fiber tractography have opened new, noninvasive windows into the white matter connectivity of the human brain. DTI and fiber tractography have advanced the scientific understanding of many neurologic and psychiatric disorders and have been applied clinically for the presurgical mapping of eloquent white matter tracts before invasive tumor removal surgeries[34].



(a) (b) (c)

Fig. 16. Placing the layered lenses on reshaped tracts to see how the tumor influences its surrounding tracts.

The most common technique for visualizing the white matter tracts from DTI uses the major diffusion tensor eigenvector to define the local white matter tract direction [35], [36], [37]. The result is a dense set of tracts, which can be represented as streamlines (or tensorlines [35]).

5.2.1 Exploring Reshaped Tracts

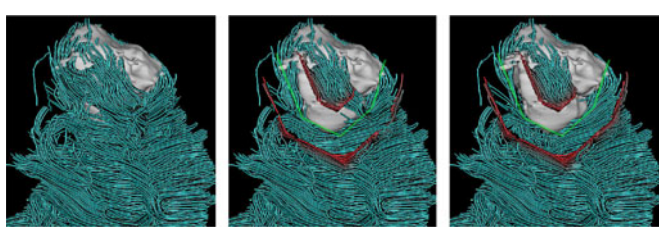
A growing brain tumor tends to compress and re-orient white matter tracts into abnormal configurations. As seen in Fig. 16a, in the center near the bottom, the tracts are distributed horizontally and connect the left and right halves of the brain. However, the tracts above and to the left of the horizontal tracts are diagonal and not symmetric because they are reshaped by the tumor.

To further explore the tracts near the tumor, in Fig. 16b, we place an open blind lens vertically to cut the tracts and move them to the left and right sides. It becomes clearer that the orientations of the tracts change from horizontal (normal) to diagonal from bottom up. We also notice that the tumor has a vertical elongated shape, which is the same orientation of the tracts on the surface of the tumor. This observation supports the finding that the tumor cells move faster along the white matter fiber tracts in the brain [38].

In order to look at the deeper tracts near the tumor and without losing much context information, we place a second smaller layered lens on the top region of the first lens and make the cut following the orientation of the tracts, as shown in Fig. 17c. Inside the second lens, we can see the tracts are connected to the right half of the brain, but not connected to the left half. This may indicate the white matter on the left has been destroyed.

5.2.2 Exploring Tracts Around Tumor

Because tracts around the tumor are of significant interest to neurosurgeons, we can place a polyline lens



(a) (b) (c)

Fig. 17. Placing a polyline lens around the tumor to inspect the relationship between the tumor and its surrounded tracts.

around the tumor by following its silhouette, as shown in Fig. 17. Before applying the lens as in Fig. 17a, tracts are densely distributed above and around the tumor, which makes it difficult to inspect the tracts that are close to the tumor but deeper inside. After placing a polyline lens around the tumor, we create curved band shape focus region, as shown in Fig. 17b. When the streamlines on the surface are removed from focus region, we can see that the tracts above the tumor is connected to the tracts on the right side. The tracts on the left side are vertically oriented. Some of them even penetrate through the tumor. After further increasing the lens depth, we see the deeper tracts in Fig. 17c. Those tracts at the bottom right of the tumor are relatively smooth and mostly horizontally distributed.

5.2.3 Feedback from Domain Experts

To gain useful, informal feedback, we demonstrated our streamline exploration approach to five neuroscience experts: a biomedical engineer, two neuroradiologists and two neurosurgeons. All experts have used primarily 2D slice- and transparency-based occlusion reduction techniques for tract visualization. They were all very positive about the deformation approach and said that it is "smooth", "very useful" and could be a "terrific tool." One neuroradiologist noted that the 3D environment allowed him to see relationships that he could not see with his 2D tools and the biomedical engineer appreciated that the deformation approach gives greater spatial context with less clutter than transparency techniques. Two experts were interested in using the software to explore objects and features deeply embedded beneath tract streamlines for deep brain probing and, interestingly, visualization deep into the kidneys. Additional application suggestions included visualization of tract disruption in trauma cases, intraoperative visualization, tract morphology in cases of Multiple Sclerosis and other neurological disorders, and endoscopy simulation. We based the coloring of deformed streamlines (see Fig. 8) from expert feedback, as it gives a clear visual cue of true versus deformed morphology.

6 Performance

We measured the performance of our interactive streamline deformation system on a machine running Windows 7 with Intel Core i7 2600 CPU, 16 GB RAM, and an nVidia GeForce GTX 560 GPU that has 336 CUDA cores and 2 GB of memory. Table 1 shows the results of two test datasets, Plume and Isabel. In our implementation, the CGAL library [39] is used to extract 2D/3D convex hulls and ellipse-shaped focus regions. To speed up the computation, CUDA and the Thrust library [40] were used to perform the deformation computation.

Four different operations related to our deformation model described previously were tested and the timings were collected. The *Cut* operation is the first step of deformation for the line model (Section 3.2), which cuts a streamline into multiple streamlines by a straight line. The *Lens* operation runs in the interactive 3D lens mode (Section 4), which searches for the streamlines that intersect the lens, and is done only when the lens or the view is changed. The *Location* operation runs in the streamline selection stage by the location mode (Section 5.1.3), which searches for the streamlines that pass through the cubic region only after the cube location is changed. The results show that these operations only take a few milliseconds, and they are only executed when some settings are changed. Therefore, they do not affect the overall performance of our algorithm much. The deformation operation is performed at every frame, so its speed is crucial for the interactivity of the system. We measured the deformation computation time and the overall frame rates for the three models described in Section 3.2. Note that the overall frame rate was computed from the average time of drawing each frame and is not related to the constant deformation speed. It reflects the speed of our algorithm in all stages, including the CUDA-based deformation operation, data transfer, and coordinates transformation. The point model with the ellipse focus takes the shortest deformation time and has the highest frame rate, while the line model is slightly slower, but highly interactive. They are both suitable for real-time performance. The point model with the convex polygon as its focus region is much slower for deformation and has a comparatively lower frame rate because the shapes of both the ellipse and the open blinds have an analytical representation, but the convex polygon is represented by a point set. Although this model is comparatively slower, it is still moderately interactive in our implementation running at at least 30 frame per second (FPS). We also measured the performances of the three layered lenses with line model and the polyline lens composed of six line segments. Their speeds are slightly lower than the basic open blind lens, but still high enough to guarantee the interactivity of the system. Finally, we can also see that the Plume dataset has a higher frame rate than the Isabel dataset because the deformation operates on each vertex and the Plume dataset has fewer vertices.

TABLE 1
Performance Test Results

Dataset		Plume	Isabel
Count	Streamlines	921	609
	Total vertices	317,814	522,436
Operation time for preparing deformation inputs (ms)	Cut	22	36
	Lens	6	9
	Location	7	7
Deformation time (ms)	Ellipse(point model)	0.67	1.08
	Polygon(point model)	13.8	25.1
	Blinds(line model)	0.81	1.27
	3 layered lens(line model)	0.783	1.44
	Polyline lens(6 segments)	1.53	1.83
Frame Rate (FPS)	Ellipse(point model)	536	347
	Polygon(point model)	58.3	35.4
	Blinds(line model)	483	293
	3 layered lens(line model)	226	147
	Polyline lens(6 segments)	209	144

7 Conclusion and Future Work

We have presented a streamline deformation technique to achieve occlusion-free focus+context streamline visualization, by displacing the occluding streamline vertices in screen space. Our deformation model has the following advantages:

- Creates an occlusion-free view from arbitrary view directions.
- Minimizes the distortion of the context streamlines.
- Provides smooth transition when distorting the deformed streamlines.
- Provides interactive performance.

In this paper we describe the deformation model and its two variations regarding the shape model used in deformation, the point model and the line model. To allow users to freely explore the flow field without prior knowledge, our system provides an interactive 3D lens and direct-touch control to move away the streamlines in user-specified regions in screen space at a given depth. To satisfy different user requirements, we provide the layered lenses and the polyline lens that explore the features in different layers and the features of more complex shapes. Our interactive streamline exploration system is based on our deformation model and we demonstrate our system through a case study using the Plume dataset and the brain diffusion tensor dataset with expert feedback. Our deformation algorithm is easy to parallelize and can achieve high performance using GPUs, and thus can be used for interactive exploration of flow datasets. We believe our interactive streamline exploration approach can help CFD scientists, engineers and neuroscientists to freely explore the data and also help students to learn about flows and brain structure.

One limitation of our deformation model is that some deformed streamlines close to the center of the focus region may still get significant distortion as they are deformed, so the original shapes of these streamlines are not well preserved. The focus region should be relatively small and local to the feature of interest to prevent too much distortion in the context. When the viewpoint is changed abruptly, the transition of deformation may not always be smooth, so sometimes users need to restart the deformation for the new viewpoint.

Our future work is to apply a combination of deformation and transparency to solve the occlusion problem. We can use transparency on the streamlines whose shapes can not be well preserved by the deformation. Our system can also be enhanced with interactive seeding of streamlines. The deformation model can be better designed to convey more depth information of the deformed streamlines. To allow focus streamlines distributed in different regions of the image space, the system should allow deformation in multiple focus regions simultaneously. Our deformation model can also be designed to work with an animation of streamlines from time-varying flow field datasets. More interactive tools, with different shapes of focus region and more methods to select focus streamlines can be added to our exploration system to allow more flexible feature exploration in screen space.

Acknowledgments

This work was supported in part by the National Science Foundation grants IIS-1250752 and IIS-1065025; and by US Department of Energy grants DE-SC0007444, DE-DC0012495, and award 59172; and by National Institutes of Health grant P41GM103545. The Pacific Northwest National Laboratory is managed for the US Department of Energy by Battelle under Contract DE-AC05-76RL01830.

References

[1] C. Correa, D. Silver, and M. Chen, "Illustrative deformation for data exploration," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 6, pp. 1320–1327, Nov. 2007.

[2] J. Tao, C. Wang, C.-K. Shene, and S. H. Kim, "A deformation framework for focus+context flow visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 1, pp. 42–55, Jan. 2014.

[3] S. W. Park, B. Budge, L. Linsen, B. Hamann, and K. I. Joy, "Dense geometric flow visualization," in *Proc. Eurograph. - IEEE VGTC Symp. Vis.*, 2005, pp. 21–28.

[4] L. Xu, T.-Y. Lee, and H.-W. Shen, "An information-theoretic framework for flow visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 6, pp. 1216–1224, Nov. 2010.

[5] X. Tong, C.-M. Chen, H.-W. Shen, and P. C. Wong, "Interactive streamline exploration and manipulation using deformation," in *Proc. IEEE Pacific Vis. Symp.*, Apr. 2015, pp. 1–8.

[6] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin, "Interactive cutaway illustrations of complex 3d models," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[7] M. McGuffin, L. Tancau, and R. Balakrishnan, "Using deformations for browsing volumetric data," in *Proc. IEEE Vis.*, Oct. 2003, pp. 401–408.

[8] C. Hurter, R. Taylor, S. Carpendale, and A. Telea, "Color tunneling: Interactive exploration and selection in volumetric datasets," in *Proc. IEEE Pacific Vis. Symp.*, Mar. 2014, pp. 225–232.

[9] O. Mattausch, T. Theußl, H. Hauser, and E. Gröller, "Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines," in *Proc. 19th Spring Conf. Comput. Graph.*, 2003, pp. 213–222.

[10] L. Li and H.-W. Shen, "Image-based streamline generation and rendering," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 3, pp. 630–640, May 2007.

[11] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma, "View-dependent streamlines for 3D vector fields," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 6, pp. 1578–1586, Nov. 2010.

[12] T.-Y. Lee, O. Mishchenko, H.-W. Shen, and R. Crawfis, "View point evaluation and streamline filtering for flow visualization," in *Proc. IEEE Pacific Vis. Symp.*, Mar. 2011, pp. 83–90.

[13] T. Günther, C. Rössl, and H. Theisel, "Opacity optimization for 3d line fields," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 120:1–120:8, Jul. 2013.

[14] A. Brambilla. (2014). Visibility-oriented visualization design for flow illustration," Ph.D. dissertation, Dept. Informatics, Univ. Bergen, Norway, Dec. [Online]. Available: http://hdl.handle.net/1956/8961

[15] G. W. Furnas, "Generalized fisheye views," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1986, pp. 16–23.

[16] M. Sarkar and M. H. Brown, "Graphical fisheye views of graphs," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1992, pp. 83–91.

[17] E. Gansner, Y. Koren, and S. North, "Topological fisheye views for visualizing large graphs," *IEEE Trans. Vis. Comput. Graph.*, vol. 11, no. 4, pp. 457–468, Jul. 2005.

[18] E. LaMar, B. Hamann, and K. Joy, "A magnification lens for interactive volume visualization," in *Proc. 9th Pacific Conf. Comput. Graph. Appl.*, 2001, pp. 223–232.

[19] M. S. T. Carpendale and C. Montagnese, "A framework for unifying presentation space," in *Proc. ACM Symp. User Interface Softw. Technol.*, 2001, pp. 61–70.

[20] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman, "The magic volume lens: an interactive focus+context technique for volume rendering," in *Proc. IEEE Vis.*, Oct. 2005, pp. 367–374.

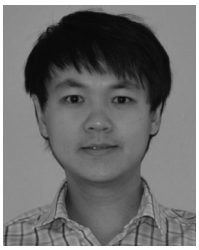
[21] X. Zhao, W. Zeng, X. Gu, A. Kaufman, W. Xu, and K. Mueller, "Conformal magnifier: A focus+context technique with local shape preservation," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 11, pp. 1928–1941, Nov. 2012.

[22] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose, "Toolglass and magic lenses: The see-through interface," in *Proc. SIGGRAPH*, 1993, pp. 73–80.

[23] A. Fuhrmann and E. Gröller, "Real-time techniques for 3D flow visualization," in *Proc. Vis.*, Oct. 1998, pp. 305–312.

[24] R. Gasteiger, M. Neugebauer, O. Beuing, and B. Preim, "The FLOWLENS: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2183–2192, Dec. 2011.

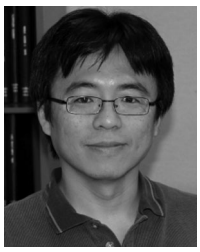
[25] J. Krüger, J. Schneider, and R. Westermann, "Clearview: An interactive context preserving hotspot visualization technique," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 941–948, Sep. 2006.
[26] M. van Der Zwan, A. Telea, and T. Isenberg. (2012). Continuous navigation of nested abstraction levels," in *Proc. Eurograph. - IEEE VGTC Symp. Vis.*, pp. 13–17. [Online]. Available: http://hal.inria.fr/hal-00781297
[27] P. Neumann, T. Isenberg, and M. S. T. Carpendale, "NPR lenses: Interactive tools for non-photorealistic line drawings," in *Proc. Smart Graph.*, vol. 4569, 2007, pp. 10–22.
[28] N. Wong, S. Carpendale, and S. Greenberg, "Edgelens: An interactive method for managing edge congestion in graphs," in *Proc. IEEE Inf. Vis.*, Oct. 2003, pp. 51–58.
[29] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw.: Practice Exp.*, vol. 21, no. 11, pp. 1129–1164, Nov. 1991.
[30] (2012). 3D Slicer, a free, open source software package for visualization and image analysis [Online]. Available: http://www.slicer.org
[31] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, J. Buatti, S. Aylward, J. V. Miller, S. Pieper, and R. Kikinis, "3d slicer as an image computing platform for the quantitative imaging network," *Magnetic Resonance Imaging*, vol. 30, no. 9, pp. 1323–1341, 2012, http://www.scopus.com/inward/record.url?eid=2s2.084867140087&partnerID=40&md5=55ae2ae8d4b0685511e8ad55eb5ce27e
[32] T. Klein, F. Guéniat, L. Pastur, F. Vernier, and T. Isenberg, "A design study of direct-touch interaction for exploratory 3D scientific visualization," *Comput. Graph. Forum*, vol. 31, no. 3pt3, pp. 1225–1234, Jun. 2012.
[33] I. Corouge, S. Gouttard, and G. Gerig, "Towards a shape model of white matter fiber bundles using diffusion tensor mri," in *Proc. IEEE Int. Symp. Biomed. Imaging: Nano Macro*, Apr. 2004, vol. 1, pp. 344–347.
[34] P. Mukherjee, J. Berman, S. Chung, C. Hess, and R. Henry, "Diffusion tensor mr imaging and fiber tractography: Theoretic underpinnings," *Am. J. Neuroradiol.*, vol. 29, no. 4, pp. 632–641, 2008.
[35] D. Weinstein, G. Kindlmann, and E. Lundberg, "Tensorlines: Advection-diffusion based propagation through diffusion tensor fields," in *Proc. IEEE Vis.*, 1999, pp. 249–253.
[36] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In vivo fiber tractography using dt-mri data," *Magnetic Resonance Med.*, vol. 44, no. 4, pp. 625–632, 2000.
[37] M. Lazar, D. Weinstein, J. Tsuruda, K. Hasan, K. Arfanakis, E. Meyer, B. Badie, H. Rowley, V. Haughton, A. Field, and A. Alexander. (2003). White matter tractography using diffusion tensor deflection. *Human Brain Mapping* [Online]. *18*, pp. 306–321. Available: http://www.sci.utah.edu/publications/lazar03/lazar.hbm.03.pdf
[38] B. H. Menze, E. Stretton, E. Konukoglu, and N. Ayache, "Image-based modeling of tumor growth in patients with glioma," in *Optimal Control in Image Processing*, C. S. Garbe, R. Rannacher, U. Platt, and T. Wagner, Eds. Heidelberg, Germany: Springer, 2011.
[39] (2012). Cgal, computational geometry algorithms library [Online]. Available: http://www.cgal.org
[40] J. Hoberock and N. Bell. (2010). Thrust: A parallel template library [Online]. Available: http://thrust.github.io/



Xin Tong received the BE degree in geoinformation science and engineering from Tongji University in 2010 and is currently working toward the PhD degree in the Department of Computer Science and Engineering of The Ohio State University. His research interests are interactive scientific visualization and large-data visualization. He is a student member of the IEEE.

John Edwards received the PhD degree in computer science from the University of Texas at Austin. He is an assistant professor at Idaho State University in the Informatics and Computer Science Department. His research interests are computational geometry and scientific visualization.

Chun-Ming Chen received the MS from Computer Science in University of Southern California and the BS from Computer Science and Information Engineering in National Chiao-Tung University, Taiwan. He is currently working toward the PhD degree of the Department of Computer Science and Engineering from the Ohio State University. His research interest is analysis and visualization for large-flow data.



Han-Wei Shen received the PhD degree in computer science from the University of Utah. He is a full professor in the Ohio State University's Department of Computer Science and Engineering. His primary research interests are scientific visualization and computer graphics. He received the US National Science Foundations Career Award and the US Department of Energys Early Career Principal Investigator Award. He is a member of the IEEE.

Chris Johnson is the founding director of the Scientific Computing and Imaging (SCI) Institute at the University of Utah where he is a distinguished professor of computer science and holds faculty appointments in the Departments of Physics and Bioengineering. His research interests are in the areas of scientific computing and scientific visualization. He founded the SCI research group in 1992, which has since grown to become the SCI Institute employing more than 200 faculty, staff and students. He serves on several international journal editorial boards, as well as on advisory boards to several national and international research centers. He was awarded a young investigator's (FIRST) Award from the NIH in 1992, the US National Science Foundation (NSF) National Young Investigator (NYI) Award in 1994, and the NSF Presidential Faculty Fellow (PFF) award from President Clinton in 1995. In 1996, he received a DOE Computational Science Award and in 1997 received the Par Excellence Award from the University of Utah Alumni Association and the Presidential Teaching Scholar Award. In 1999, he was awarded the Governor's Medal for Science and Technology from Governor Michael Leavitt. In 2003, he received the distinguished professor Award from the University of Utah. In 2004, he was elected a fellow of the American Institute for Medical and Biological Engineering, 2005 he was elected a fellow of the American Association for the Advancement of Science, in 2009 he was elected a fellow of the Society for Industrial and Applied Mathematics (SIAM) and received the Utah Cyber Pioneer Award. In 2010, he received the Rosenblatt Award from the University of Utah and the IEEE Visualization Career Award. In 2012, he received the IEEE IPDPS Charles Babbage Award, and in 2013 he received the IEEE Sidney Fernbach Award. In 2014, he was elected an IEEE fellow.

Pak Chung Wong received the PhD degree in computer science from the University of New Hampshire. He is a chief scientist and project manager at the Pacific Northwest National Laboratory. His research interests include big data analytics, graph analytics, visual analytics and visualization, earth sciences, social networks, and national security. He serves as an associate editor-in-chief of *IEEE Computer Graphics and Applications* and an associate editor of *Information Visualization*. He is a member of the IEEE.