Occlusion-Free Focus+Context Streamline Visualization with Deformation

Xin Tong, Chun-Ming Chen, Han-Wei Shen and Pak Chung Wong

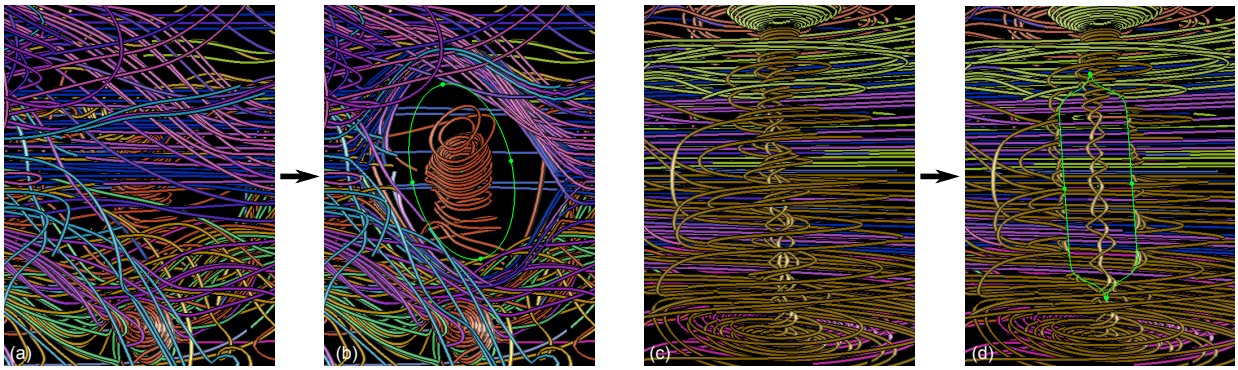


Fig. 1: (a) and (b) illustrate our interactive 3D lens with the point model. (a) Before applying the lens, the vortex is hidden by the streamlines in the front. (b) After applying the lens with the point model, the streamlines in front of the lens are removed to revel the vortex. (c) and (d) illustrate interactive lens with the line model. (c) Before applying the lens, the inner structure of the vortex is hidden. (d) The inner structure is now revealed after applying the lens with the line model. The outside of the vortex is cut into two halves and pushed to the side.

Abstract— Occlusion presents a major challenge in visualizing three-dimensional flow fields with streamlines. Displaying too many streamlines at once makes it difficult to locate interesting regions, but displaying too few streamlines risks missing important features. A more ideal streamline exploration model is to allow the viewer to freely move across the field that has been populated with interesting streamlines and pull away the streamlines that cause occlusion so that the viewer can inspect the hidden features in detail. In this paper, we present a streamline deformation algorithm that supports such user-driven interaction with three-dimensional flow fields. We define a view-dependent focus+context technique that moves the streamlines occluding the focus area using a novel displacement model. To preserve the context surrounding the user-chosen focus area, we propose two shape models to define the transition zone that accommodates the deformed streamlines, and solve the displacement of the contextual streamlines interactively with a goal of preserving their shapes as much as possible. Based on our deformation model, we design an interactive streamline exploration tool using a lens metaphor. Our system runs interactively so that users can move their focus and examine the flow field freely.

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

1 Introduction

Streamlines are commonly used for visualizing three-dimensional flow fields. Streamlines show the trajectories of particles moving along the directions of the flow field and thus provide insight into intricate behaviors of the flow field. Flow features like sinks, sources, saddles, and vortices can be visually identified by streamline visualization. However, when too many streamlines are shown, getting a clear view of important flow features is difficult. Even though a single streamline does not cause much occlusion compared with geometry of higher dimensions, such as surfaces or volumes, mixing many streamlines of different depths together can generate a very confusing image. This is not only because streamlines in the front will occlude those in the back but also because part of the streamlines in the back will show through; therefore, discerning the exact depths of the visible streamline segments becomes very difficult.

Although through interactive seeding of streamlines one can control the amount of occlusion and visual cluttering, it makes finding specific flow features–and hence understanding their surrounding context–more difficult. To reach a balance between displaying too much information and displaying too little, focus+context(F+C) techniques provide a nice solution for highlighting the details in the focus area without losing the context. In this paper, we present a streamline deformation technique to achieve a F+C view of three-dimensional streamlines. The previous streamline deformation approaches [3] [21] deform the three-dimensional space of the flow field, which indirectly deforms the streamlines embedded in the new space. The main drawback of the space deformation approaches is that a deformed continuous 3D space [21] can fail to generate a complete void to eliminate occlusion. Also, because those methods are object space approaches, they may not adapt to arbitrary view directions.

Rather than indirectly deforming the occluding streamlines in the deformed space, we present a novel view-dependent deformation model that deforms the streamlines directly by moving its vertices in the two-dimensional screen space. To achieve the deformation, the user first defines a *focus region*, which can be an arbitrary location in screen space. Alternatively, the focus region can be defined as an area in screen space enclosing user-selected focus streamlines. Then, streamlines occluding the focus region are deformed and moved out of the focus region 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 line of the focus region and moves the streamlines away from the line to its sides. Based on the shape of the focus area, choosing between these two models can ensure a smooth transformation of

- *Xin Tong, Chun-Ming Chen, Han-Wei Shen are with The Ohio State University. E-mail: {tong,chenchu,hwshen}@cse.ohio-state.edu.*
- *Pak Chung Wong is with Pacific Northwest National Laboratory. E-mail: pak.wong@pnnl.gov.*

the deformed streamlines. Because our deformation is performed in the screen space, the deformation can automatically adapt to new view directions.

To show the efficacy of our deformation model, in this paper we present an interactive tool to explore streamlines in three dimensional space. We create a tool that uses a lens metaphor to allow users to freely move away streamlines from selected areas on the screen to reveal the structure underneath. We also present additional examples to define the focus area, including streamline bundles, location-based streamlines, and multi-focuses. A graphics processing unit (GPU)-based parallel computation of the deformation allows the exploration to be done in real-time. To test the effectiveness of our system, we applied our deformation framework to explore several real flow field datasets to find hidden features in the streamlines.

The rest of the paper is organized as follows. In Section 2, we briefly review several solutions for occlusion-removal in the context of 3D rendering, with a focus on F+C techniques and deformation methods. Then, we discuss our algorithm and two deformation models in Section 3. We describe how to apply our deformation model to create an interactive tool in Section 4, and present some examples to define focus streamlines in Section 5. We show the results of applying our techniques to explore two datasets in Section 6 and discuss important implementation details in Section 7. The performance of our GPU-based implementation is discussed in Section 8, followed by some concluding remarks in Section 9.

2 Related Works

Overcoming occlusion is an important but challenging problem in 3D visualization. Several approaches have been proposed in the past to avoid or remove occlusion. In direct volume rendering [11], transfer functions are designed to assign proper voxel transparencies to reveal hidden features in the data. Li *et al.* [13] 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.* [17] 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. Qu *et al.* [19] proposed to zoom in the routes in visualizing 3D urban environments. In this work, roads are broadened by 2D seam carving, and landmarks are enlarged with a grid-based zooming technique. An occlusion-free route is visualized by scaling the buildings that occlude the route.

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.* [16] applied magic volume, region of interest driven streamline placement, and spotlights to alleviate the occlusion problem. Li and Shen [12] 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 the image space. Marchesin *et al.* [15] defined the overlap value, the average number of overlapping streamlines for each pixel in the image space, to quantify the level of cluttering and remove the streamlines that have high overlap values on their projected pixels. Lee *et al.* [10] 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.* [18] applied multi-dimentional transfer functions (MDTFs) based on physical flow properties to change the color and opacity of streamlines. Xu *et al.* [24] proposed to make the streamlines in lower entropy regions more transparent to reduce 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 keeping all the input streamlines more easily to be seen.

Fous+Context techniques have been used by different applications that magnify the focus objects while preserving the surrounding context. The techniques include fisheye views [5, 20, 6] and magnification lens [9, 23, 25]. In flow visualization, 3D lenses have been applied to show the focus region with greater details [4, 16]. van der Zwan *et al.* [22] blended several levels of detail with a halo-like shading technique to simultaneously show multiple abstractions. These methods can reduce occlusion in 3D but do not completely keep the focus objects out of occlusion.

Our approach is more related to the following works with F+C flow visualization using spatial deformation. Correa *et al.* [3] 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 in their method the deformation is performed in data space, the focus can be occlusion-free for only certain view directions but not for some other views. Tao *et al.* [21] 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 does not completely solve the occlusion problem for certain views. Both Correa and Tao's methods are view-independent, which means the deformation can fail to remove occlusion for certain views. 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 for an arbitrary view.

3 Algorithm

The goal of our algorithm is to expose interesting features in the *focus* area by deforming the occluding streamlines. In our deformation model, depending on the approximate shapes of the focus area, two shape models, the point model and the line model, are used. For each of the two shape models, we design a screen-space deformation algorithm that displaces vertices of the occluding streamlines. The deformation creates a void region to allow a clear view of the interesting features from arbitrary view directions.

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. The context streamlines are the rest. 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 Figure 2. The focus region is a user-specified region in the screen space that contains the interesting features; the transition region is the area that is immediately adjacent to the focus region and contains the deformed streamlines; and the context region is the rest of the screen space that contains undeformed streamlines. Although streamlines are defined in 3D space, our deformation takes place only in the 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 in the focus region to the transition region. To make space for the streamlines, streamline segments that are originally in the transition region will also be compressed and moved towards the outer boundary of the transition region. Essentially, the occluding streamlines in the focus region and the streamlines originally in the transition region are deformed and compressed in the transition region and leave the feature of interest to the view in the focus region occlusion free. The occluding streamlines are deformed but never removed,

providing the context to the focus area. Any other streamlines outside of these two regions remain unchanged.

There are two design goals for our deformation model:

1. The deformed streamline should preserve its shape, 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 each other, compared to other pairs of connected vertices. Otherwise, the streamlines will tend to be jagged and a long edge between two connected vertices may cut across the focus region.

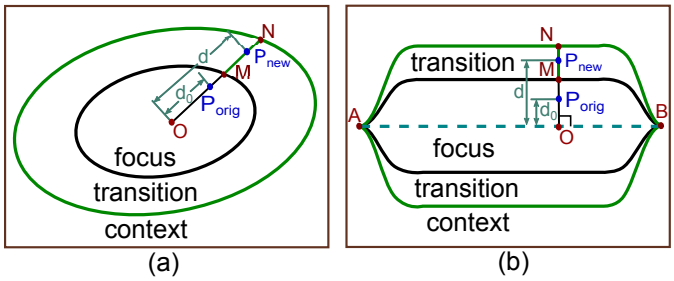


Fig. 2: Sketches of the two deformation models: (a) the point model (b) the line model. The region inside the black boundary is the focus region. The region between the black and green boundaries is the transition region. The region outside the green boundary is the context region. During the deformation, 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.

In our deformation model, the deformation of a streamline is achieved by displacing its projected vertices in the screen space. Basically, during the 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 distribute the deformed streamlines in the transition region with their shapes preserved, 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 area. Figure 2 illustrates these two models. The point model is designed for focus areas that have a circular shape, while the line model is for focus areas that have a linear shape. Both shapes are typical for streamlines.

Point Model As shown in Figure 2a, the point model is composed of a 2D focus area (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 area can also be represented by a convex polygon. Each of the convex polygon and the ellipse-based focus areas has its 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 area in the point model is the reference point from which the streamlines are moved away.

Line Model Figure 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 (inside the black boundary) that represents the focus region, and an expanded area outside of the inner focus (the area between the black and the green boundaries). 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. Line AB and the opening width can be arbitrarily defined by the user or generated from the axes of the minimal enclosing ellipse if focus streamlines are selected. 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 area is clear. More details about using the line model for deformation will be explained below in Section 3.3.

The point model and the line model are suitable for streamlines of different shapes. If we apply the point model to straight streamlines distributed in an area in Figure 2b, then the streamline vertices will be moved too far away from the centroid point and hence some vertices will have to travel a long distance to the region around the point A or B, resulting in a large distortion for the deformed streamlines. On the other hand, if we apply the line model to streamlines that have a circular shape, we have to cut the streamline and hence unnecessarily change the connectivity of the streamlines. The overall shape of the streamlines decides which model to use. In our algorithm, we use a minimal enclosing ellipse to measure the roundness of the focus area. If the major radius of the ellipse is much larger than the minor radius, then we use the line model; otherwise, we use the point model.

3.3 Deformation Model

Instead of computing the new vertex positions for the deformed streamlines all at once, our method achieves the deformation by displacing its vertices iteratively. 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 area. 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 velocity and a moving direction. Mathematically, the vertex movement can be written as:

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

where P' is the new position of the vertex, P is the old position, $v \times \vec{w}$ representing the movement that makes the point leave the focus area, and $v_c \times \vec{u}$ makes sure that the new point position is not too far from its neighbors. Hereafter we call $v \times \vec{w}$ as the *major displacement*, and $v_c \times \vec{u}$ as the *minor adjustment*. 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 area 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 Figure 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 Figure 5. If the line model is used, shown in in Figure 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 the point model and the line model, 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. Based on how far the streamline vertex has traveled, we change the value of v in different iterations. Note that v can be positive or negative. When v is positive, the vertex moves along the direction of $\vec{w}$. When v is negative, the vertex moves along the opposite direction of $\vec{w}$. When $v = 0$, the vertex stops moving.

Assuming d_0 is the distance between O and the vertex's original position P_{orig}, i.e. $d_0 = |\vec{OP_{orig}}|$, and d is the distance between O and the vertex's final position P_{new} after deformation, i.e. $d = |\vec{OP_{new}}|$, because the vertex moves outwards, we have $d > d_0$. 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:

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

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 $|\vec{OP}|$ keeps increasing during the deformation, the speed of v keeps decreasing, until it becomes 0. This is when the vertex stops moving and arrives at its final position P_{new}.

As shown in Figure 2a and Figure 2b, if we draw a line from O along the $\vec{w}$, this line intersects with the boundary of the focus region (the black boundary) at M and intersects with the outer boundary of the transition region (the green boundary) at point N. If we ignore the minor adjustment for now, the major displacement moves a vertex from its original position P_{orig} to the new position P_{new} along the line OM.

We now discuss how to compute the distance d, which determines the final position of each streamline vertex after the deformation. First, to move the vertex out of the focus area and into the transition area, we should have $d > |\vec{OM}|$. Because we want to preserve the shape of the streamline according to our design goal, d is made to be a monotonically increasing function of the original distance d_0, which transforms $|\vec{OP}|$ from d_0 to a larger value d. We use a function G that takes a normalized value of d_0 as such a function to transform from d_0 to d

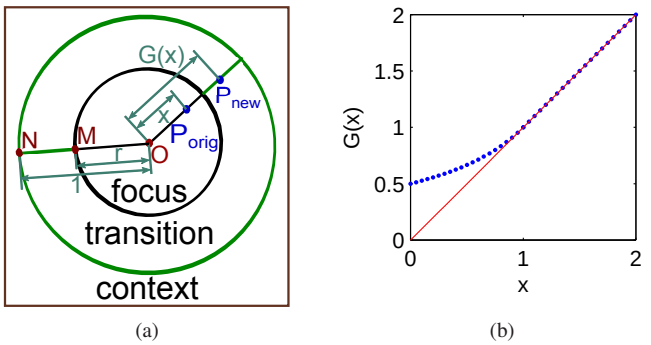
$$d = G\left(\frac{d_0}{T}\right) \times T \tag{4}$$

where T is the distance between O and the outer boundary of the transition region along the moving direction of $\vec{w}$, i.e. $T = |\vec{ON}|$. d_0 is normalized by diving its value by T.

The Figure 3a is an illustration of the point model similar to Figure 2a , marked with the several 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. $r = \frac{|\vec{OM}|}{T}$. Because d_0 varies in the range $[0, |\vec{OM}|]$, $\frac{d_0}{T}$ varies in the range $[0,1]$. Because d varies in the range $[|\vec{OM}|, |\vec{ON}|]$, $\frac{d}{T}$ 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

The goal of the displacement function G is to transform a value in $[0,1]$ to a larger value in $[r,1]$. Instead of using a linear function, we apply the idea from the transformation function of fisheye lens[20] to design a non-linear displacement function G that gives us a smoother transition of the deformation across the region boundary. Without loss of generality, here we assume a point model with a circular boundary shown in Figure 3a to explain the idea. Below, we first give the design goal of the function G, and then solve G.



(a) (b)

Fig. 3: (a) Illustration of the point model in the normalized space. (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.

Our first criterion is that, as shown in Figure 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:

$$G(0) = r \tag{5}$$

$$G(1) = 1 \tag{6}$$

Secondly, the function G must be a monotonically increasing function to make sure that the deformed streamlines and their vertices have the same relative positions after the deformation. This way, the deformed streamline keeps a similar shape and the chance for neighboring streamlines to cut cross each other decreases. A positive value of the first derivative of G makes sure the G is monotonically increasing, written as:

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

To describe the next criterion, we need to clarify the meaning of the first derivative of the displacement function $\frac{dG(x)}{dx}$. If we apply the deformation based on G in the entire space of the focus region and the transition region, we get a distorted space. $\frac{dG(x)}{dx}$ describes the amount of distortion in the deformed space at a point whose distance to O is x. We know that in the region outside of the transition region, referred to as the context region, no streamlines are deformed and the space is not distorted. So $G(x) = x$ for $x > 1$ in the context region, and thus $\frac{dG(x)}{dx}$ is 1 for any points in context region. To ensure that the amount of distortion smoothly changes from the transition region to the context region at their boundary point N in Figure 3a, $\frac{dG(x)}{dx}$ should be continuous at that point. So we know the $\frac{dG(x)}{dx}$ at point N should also be 1, that is:

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

Finally, our last design criterion is that, from a position near O to a position far from O, the amount of distortion should also change monotonically. During the displacement, the deformed streamlines are squeezed into a smaller space. In other words, the input x is in the range $[0,1]$, while the output $G(x)$ changes its value in the range of $[r,1]$. Because the range of $G(x)$ is smaller than the range of x, we have $\frac{dG(x)}{dx} < 1$ for $x \in [0,1)$. Thus, 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 $\frac{dG^2(x)}{dx^2}$. Because the distortion is monotonically increasing, the second derivative of G should be greater than 0, written as:

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

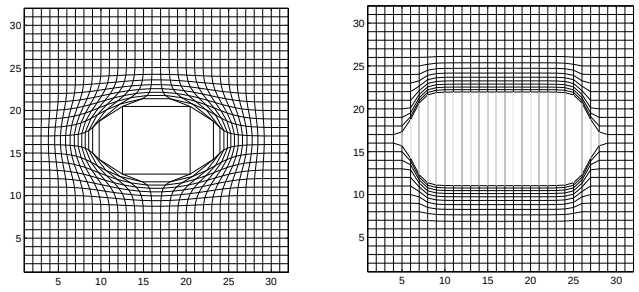
Combining the four criteria from Equation 5 - 9, we can solve the displacement function G. There is more than one solution for G, and we use the simplest one as:

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

To show that the above function satisfies the four criteria, we plot Equation 10 in Figure 3b. The figure clearly shows that Equation 5 and Equation 6 are satisfied. The blue dotted line is the values of G. The red line is a reference function, which means no displacement in the entire domain. We see that G is monotonically increasing, which satisfies Equation 7. Its slope reaches 1 at the point of $x = 1$, which satisfies Equation 8. The slope gradually increases, which satisfies Equation 9. The value of r is the intersection point of the blue line with

the y-axis. The larger r we use in G, the smaller transition region we get, the smaller space the deformed streamlines need to be squeezed in, and hence more distortion will happen to the deformed streamlines.

If we use the displacement function G to move a regular grid with our two shape models, we get the deformed grids shown in Figure 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 from the focus region. The existence of the void region is the difference between our method and the traditional fisheye lens methods. We also notice that for the same amount of distortion (same value of r) in space we can create a larger void space with the line model shown in Figure 4b than the point model shown in Figure 4a. Besides, in the point model, some parts of the space are stretched, and other parts are compressed; while in the line model, all the spaces are compressed, and none of the space is stretched.



(a) point model (b) line model

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

3.3.3 Minor Adjustment

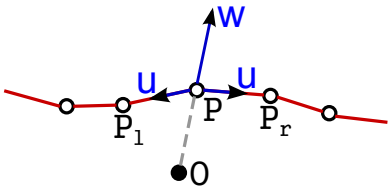


Fig. 5: The two displacement directions $\vec{w}$ and $\vec{u}$ for the point at P, using the point model as example. P_l and P_r are the two vertices connected to P on this streamline.

The minor adjustment does not contribute much to the movement of a vertex but helps to make the vertices uniformly distributed on a streamline and thus satisfies the second design goal of our deformation model. In the general process of particle advection to generate streamlines, there is a maximum step size to avoid the edge between two adjacent vertices becoming too long, or the streamline will tend to be jagged. During our deformation process, some edges can be stretched more, especially when using the point shape model, 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.

A simple solution to avoid this long edge problem during deformation is to insert vertices into the long edge to break it into short edges. However, this is not efficient, especially when the deformation is done by GPU where it is more difficult to add more geometry. So our solution is not to insert additional vertices but instead to move the vertices along the streamline to make the vertices uniformly distributed across the streamline.

As shown in Figure 5, 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. This shortens the longest edge in each iteration, and eventually no edge is much longer than the other edges.

The minor adjustment $v_c \times \vec{u}$ 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:

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

When the length of the left edge $|\vec{PP_l}|$ is larger than the length of the right edge $|\vec{PP_r}|$, the moving direction is the same as the vector $\vec{PP_l}$ and vice versa. If the lengths of the left edge and right edge are equal, then $\vec{u} = \vec{0}$, which means no minor adjustment is needed.

Note that the minor adjustment may move the vertex in a direction other than $\vec{OP}$, which ends up changing $\vec{w}$ in the next iteration. This change is not recoverable for the later iterations. So when the view direction or the focus region is changed during the deformation, if we continue the deformation with the new value of $\vec{w}$ or T, 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 only solution is to recover the streamlines' original positions and redo the deformation from the first iteration.

4 Interactive 3D Lens

In this section we introduce a streamline exploration tool, interactive 3D lens, based on the deformation algorithm described above. The interactive lens is useful when users do not have prior knowledge about data and want to freely explore the streamlines that have been computed. To overcome the occlusion problem, the lens can be placed anywhere in image space with an adjustable depth to peel away the occluding streamlines layer by layer. Specifically, the lens defines a focus area, and any streamlines inside the lens in image space and closer to the viewer than the far side of the lens are moved away as the context streamlines.

We design our interactive 3D lens as a 3D cylindrical object, shown as the black cylinder in Figure 6 . The axis of the cylinder is perpendicular to the screen, i.e., the z axis of the NDC(Normalized Device Coordinates) space, and the surface of the lens is placed on the xy plane of the NDC space. The length of the cylinder is used as the *lens depth*. The lens will move away any streamlines that intersect with the lens cylinder and keep other streamlines unchanged. To reveal streamlines at different depths, users can change the length of the lens.

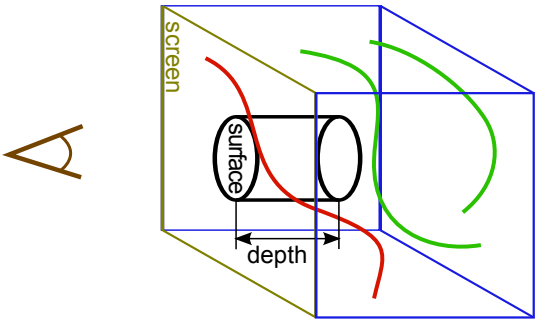


Fig. 6: The 3D lens in NDC space. The blue cube denotes the 3D NDC space. The yellow square denotes the 2D screen space. The lens has an ellipse-shaped surface on the plane of the screen. The red streamline intersects with the black 3D lens, and thus will be deformed. The two green streamlines do not touch the 3D lens, and thus will remain unchanged.

In our implementation, because only the streamlines passing through the lens are to be deformed, we test the intersection between the lens cylinder and all the streamlines in the NDC space. To

test whether a streamline passes through the lens, we simply check whether there exists at least one vertex from this streamline that is enclosed by the cylinder and do not check the intersection of line segments between two consecutive streamline vertices with the cylinder to improve the performance. As the example shows in Figure 6, there are three streamlines (one red and two green). Only the red streamline that intersects with the lens will be deformed. Once the streamlines that intersect the lens are identified, we use our deformation model to deform the streamlines.

In the interactive system, users can use the mouse 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 depths. Note that the axis of the lens is always parallel to the view direction, so users can change the view angle to reveal different portions of the field. An interactive demonstration of the lens can be found in the accompanying video.

Figure 1 shows examples of the interactive 3D lens using the point and the line deformation model, respectively. In Figure 1a, the straight streamlines in the front occlude the vortex-like streamlines in the back. In Figure 1b, we use a lens with the point deformation model to move the straight streamlines away and reveal the vortex gradually through animation (please see the accompanying video). In Figure 1c, users want to see the inner structure of the vortex. So a lens with the line deformation model of open blinds is used to break the outside of the vortex into two parts. In Figure 1d, the inner structure, which was previously occluded, now becomes visible. Users can also interactively decrease the lens depth to allow the deformed streamline to go back to its original position.

In summary, the interactive 3D lens uses our view-dependent deformation model to explore and reveal the hidden streamlines in the flow field. 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. 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. Finally, users can rotate and adjust the shape of the lens surface interactively to explore the field in different orientations.

5 Focus Streamlines

In the previous section we showed that the interactive 3D lens based on our deformation model can help the user freely explore the streamlines. Because our method can reveal the occluded streamlines while preserving the context, it can also be applied to highlighting user-specified feature streamlines defined by the user. Below we describe two ways to define the focus streamlines. One is through picking *streamline bundles*, which are groups of streamlines with similar shapes and close proximity (Section 5.1). The other is choosing *streamlines by locations*, which shows the streamlines that pass through a user-selected region (Section 5.2). We use these two methods to demonstrate the utility of our deformation model but note that the selection of the focus streamlines is not limited to these two methods. For example, users can query all the streamlines by using some geometric predicates, such as having their average curvature, torsion or curl within a particular range. Because the focus streamlines may distribute over multiple screen locations, a single focus region enclosing all the focus streamlines may be too big to display the focus streamlines tightly. Thus, in Section 5.3 we discuss how to group the focus streamlines by their screen locations to form multiple convex focus regions and apply deformation to each focus region independently.

In this paper, we use two 3D flow field datasets to generate streamlines as examples, including Hurricane Isabel and Solar Plume. Hurricane Isabel was a strong hurricane in the west Atlantic region in September 2003. The resolution of the dataset is $500 \times 500 \times 100$. Solar Plume is a simulation of the solar plume on the surface of the Sun. The resolution of the dataset has a resolution of $126 \times 126 \times 512$.

5.1 Streamline Bundles

To allow the user to select streamlines of interest, one approach is to first bundle similar streamlines together. Because in general the flow directions in local regions change smoothly, the corresponding streamlines will have similar shapes if they are spatially close to each other. Based on this idea, we cluster the input streamlines into *streamline bundles* based on their locations and shapes and show the bundles to the user. The user can then select one or more bundles of streamlines as the focus streamlines and use our deformation model to remove the occlusion and reveal their spatial locations and the surrounding context. Because a streamline can be long and may change its complexity in different regions, we want to separate a streamline into multiple segments so that the streamline segments with similar shapes are clustered together. To segment the streamlines, we use a distribution-based segmentation algorithm proposed by Lu et al. [14]. In their method, the shape of a streamline is represented as a distribution of several flow features such as curvature, torsion and curl. Then the streamline is split at a vertex where the difference of the feature distributions between the two halves is the maximum. The splitting is performed recursively until the difference between the two halves is less than a threshold or the segment is too short to be split. After this segmentation, within each segment the complexity does not vary much.

After the segmentation, we cluster the streamline segments using hierarchical clustering, which builds a tree structure and can output a desired number of clusters selected by the user. We use the *mean of closest point distance* [2] to compute the distance between two streamlines.

Fig. 7: Eight streamline bundles from the total 36 bundles of Isabel dataset.

Figure 7 illustrates the user interface that lists some of the streamline bundles from the Isabel hurricane dataset. We can see a cluster with circular streamlines from the hurricane eye region and some with elongated streamlines from other regions. The user can select one or several of them as the focus streamlines. A demonstration of using streamline bundles to perform deformation is shown in the accompanying video.

5.2 Streamline Selection by Locations

The focus streamlines can also be selected directly by their spatial 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.

As shown in Figure 8, the green cube represents the user-selected region, and the streamlines that pass through this region are used as the focus, while the surrounding context streamlines are deformed away. The user can move the cube along its axes and the system can immediately recalculate the focus streamlines and perform the deformation.

5.3 Grouping Focus Streamlines into Multiple Focuses

Different from using the interactive 3D lens where the focus is always in a single local region, the user-selected feature streamlines may scatter in different regions in the domain. For example, the user can select

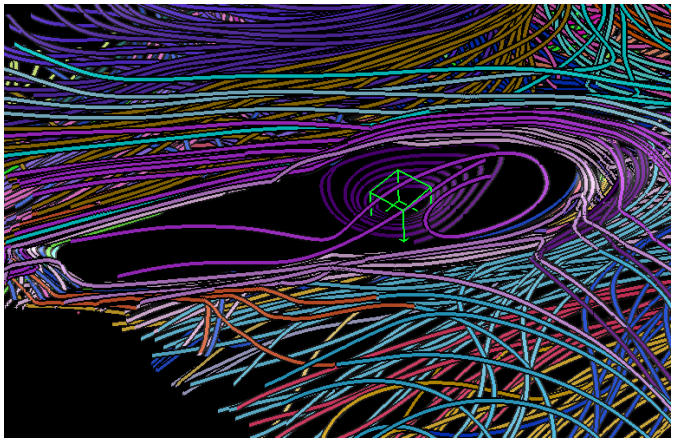


Fig. 8: Streamlines passing through the green cube are selected as the focus. A point deformation model with a convex polygon shape focus region is used in the deformation.

several clusters from different regions. In such a case, a single deformation shape model may open up the space too much, which results in a large empty space in the focus area. To fix this problem, instead of using only one focus, we group the selected streamlines on the screen into multiple focus areas based on their screen locations and generate one focus area for each group. This allows us to generate tighter focus areas and avoid deforming too many streamlines.

As we divide the feature streamlines into multiple groups, we make sure that the groups do not overlap too much in screen space. Two streamlines that are too close to each other are grouped together. We use the axis-aligned bounding box (AABB) to represent one streamline or one group of streamlines to accelerate the grouping process. If the AABBs of two groups of streamlines overlap or are closer than a threshold in any dimension, we merge the two groups into one group. We perform such grouping in a bottom-up manner, that is, we first treat each streamline as a group, and then merge two groups if the two groups' AABBs are too close, until none of the groups are closer to each other than a threshold. Note that we try to make the threshold of minimum group distance large enough so that the transition regions of two groups will not overlap. Thus, we prevent a vertex from receiving displacements from two different deformation focuses.

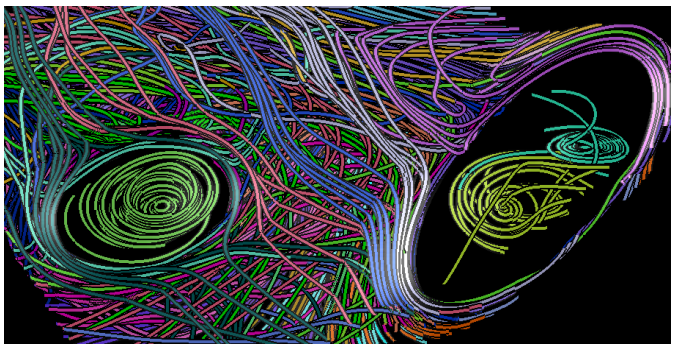


Fig. 9: Multi-focuses deformation with the ellipse-shaped focus region.

In Figure 9, the streamlines from three vortex-like bundles in the Isabel dataset are picked as the focus streamlines. Because the streamlines from two of the bundles on the right overlap with each other on the screen, our grouping method places them in the same group. Then two focus regions are generated from the two groups of the focus streamlines. Each focus region performs its deformation independently. Suppose we instead use only one big focus region to enclose all the focus streamlines; the streamlines between the two focus regions in Figure 9 will all be moved away from their original positions, which results in unnecessary loss of context. Thus, grouping streamlines in screen space provides smaller focus regions and preserves more context.

6 Exploration

With the proposed deformation algorithm, as well as the interactive 3D lens and focus streamline selection methods, we design an interactive system for streamline exploration. In this section, we use our exploration system to explore the Plume dataset in three different ways: exploring the streamlines at different depths, from different view directions, and at different locations.

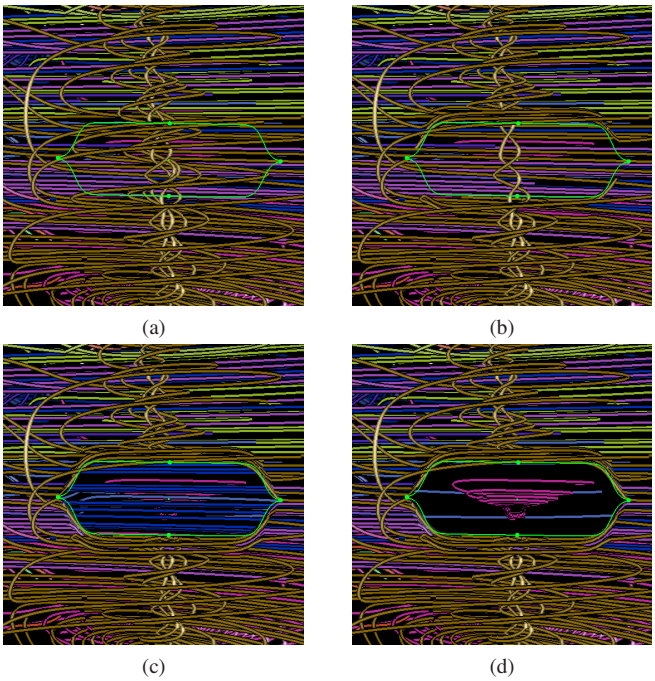
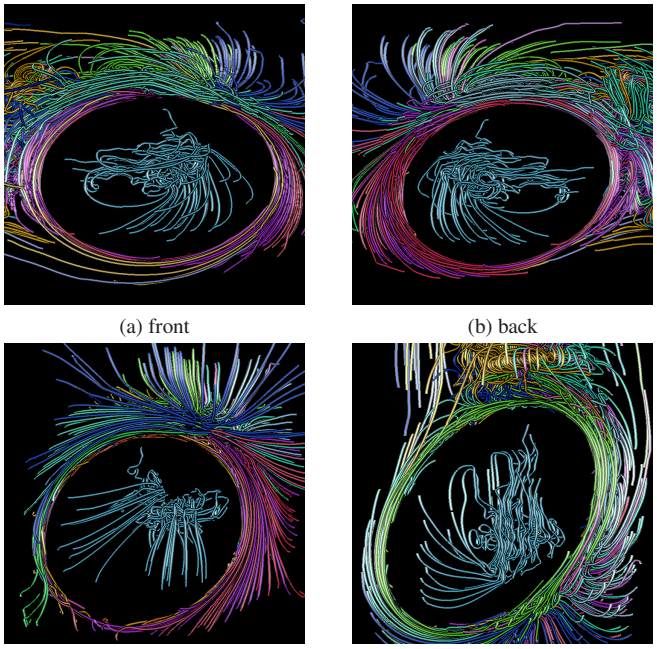


Fig. 10: Exploring streamlines at different depths by the interactive 3D lens with different lens depths.

Exploration with Different Depths Different streamlines of different depths may be projected to the same screen location, and the ones 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 Figure 10. In Figure 10a, 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. Then in Figure 10b, we push the lens into the volume by increasing the lens depth until the lens touches the outer portion of the vortex. At this point, the lens cuts the outer part of the vortex and pushes those occluding streamlines away to reveal the inner structure of the vortex. By continuing to push the lens into the volume, we move away the vortex and see the blue horizontal streamlines in the center of the volume, shown in Figure10c. Finally in Figure10d, 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.

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 adapt to the environment that surrounds a feature and remove the occlusion from all viewing angles. Furthermore, the user can get different context information from the different views. In Figure 11, we view a streamline bundle from four different sides of

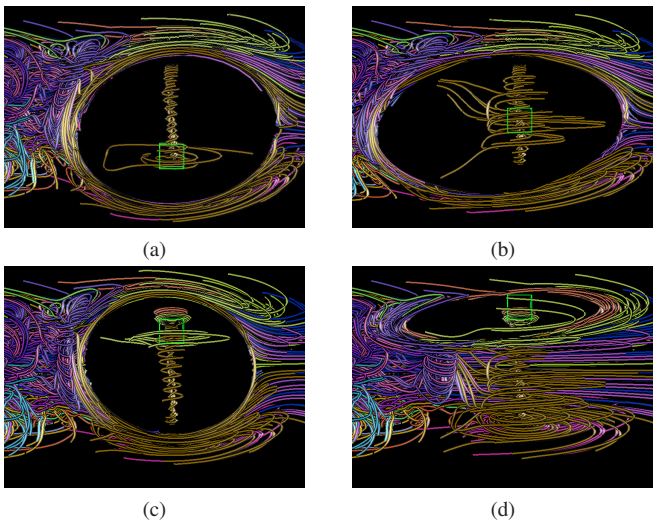


(a) front (b) back

(c) left (d) top

Fig. 11: Views of a bundle from different view directions.

the volume. This bundle shows a turbulent flow structure with straight tails. Figure 11a and Figure 11b give the views from the front and back of the bundle respectively. We can compare both views of these streamlines and see the slight difference at the center of the bundle. Note that a yellow vortex is preserved at the upper-left of Figure 11a but is not visible from the opposite view in Figure 11b. Figure 11c is a view from the left of the volume. This view of the bundle looks very different from what can be seen from the previous two viewing directions. The contexts are almost all stretched long streamlines. Figure 11d views the focus streamlines from the top. We can see some curvy vertical streamlines from this view. The yellow vortex visible in the context of Figure 11a is again visible from this view and has a more complete shape than that in Figure 11a.



(a) (b)

(c) (d)

Fig. 12: Exploring flow features at different locations. The streamline picking cube moves bottom up from (a) to (d).

Exploration with Different Locations A location-based picking tool allows the user to freely select streamlines passing through the selected region and trace the streamlines both forward and backward from that region. Here we illustrate the exploration of the streamlines from different locations around a vortex. In Figure 12, the streamlines passing though the selected location indicated by the green cube are shown. In Figure 12a, 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 Figure 12b. From the tails of the focus streamlines on the left, we know that this selected location is on the left side of the flow. We keep moving the cube up to see the focus streamlines, as shown in Figure 12c. 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. Finally, we move the cube to the top of the volume, shown in Figure 12d. From this location, we only see a small vortex, and a few horizontal streamlines showing the flow moving along that direction. Note that the four images in Figure 12 all preserve the context features, such as the purple vortex on the left and the horizontal straight streamlines on the right. A demonstration of this exploration is shown in the accompanying video.

7 Implementation Details

This section describes the implementation details of our deformation algorithm and the streamline exploration system.

As mentioned above, the focus area can be represented by a convex polygon, which is computed from the convex hull of the focus streamlines. Instead of computing the 2D convex hull from all the vertices of the focus streamlines for each view, we first generate the 3D convex hull of those vertices and then project the vertices of the 3D convex hull to the screen. This projected 2D convex hull is then used as the focus region. Projecting the 3D convex hull to the screen is similar to computing the convex hull in 2D, as discussed in article [7]. Because the 3D convex hull is invariant to different view directions and can be generated in the pre-processing stage if the feature streamlines are known already, the more expensive per-view operation can be avoided. In our implementation, the CGAL library [1] is used to extract the 2D/3D convex hull. B-spline is used to resample the 2D convex hull and make it smoother. After the convex hull on the screen is computed, an ellipse-shaped focus area is generated by CGAL.

To further speed up the computation, CUDA and the Thrust library [8] were used to perform the deformation computation. In our implementation, each streamline vertex is assigned to a thread in GPU. CUDA+OpenGL Interop is used to allow the CUDA program to get the pointer of the vertex coordinates in the OpenGL vertex buffer object(VBO) and change their locations during deformation.

During rendering, each bundle is assigned a unique color to distinguish the different features. Halo and lighting are added to the streamlines to provide the necessary depth cue.

8 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 2GB of memory. Table 1 shows the results of two test datasets, Plume and Isabel. Streamlines were first generated with random seeding for each dataset. The timing for generating the streamline bundles described in Section 5.1 is listed as *Streamline bundling* in the table. The distance computation took relatively more time. But this is done in a preprocessing stage and thus does not affect the rendering performance because the bundles are unchanged during the visualization stage.

Besides streamline bundling, four different operations for preparing the input to the deformation model described previously were tested and the timing was collected: The *Cut* operation runs in the first deformation iteration in 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

Table 1: Performance test results.

Dataset		Plume	Isabel
Count	Streamlines	921	609
	Total vertices	317,814	522,436
Streamline bundling (seconds)	Segmentation	4.8	5.6
	Distance measure	55	142
	Clustering	0.49	0.19
Operations for preparing deformation inputs (ms)	Cut	22	36
	Lens	6	9
	Location	7	7
	Grouping	8	13
Deformation (ms)	Ellipse	0.67	1.08
	Convex Polygon	13.8	25.1
	Line	0.81	1.27
Frame Rate (FPS)	Ellipse	536	347
	Convex Polygon	58.3	35.4
	Line	483	293

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 5.2), which searches for the streamlines that pass through the cubic region only after the cube location is changed. The *Grouping* operation (in Section 5.3) groups the focus streamlines based on their screen coordinates, which is called after the selected streamlines or the scene changes. 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: the point model with the ellipse bounding shape (*Ellipse)*, the point model with the convex-polygon bounding shape (*Convex polygon*), and the line model with the open blinds bounding shape (*Line*). Note that the overall frame rate 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. They are both suitable for real-time performance. The point model with the convex polygon as its focus area is much slower for deformation and has a comparatively lower frame rate. This is because both the shapes of the ellipse and the open blinds have an analytical representation, but the convex polygon is represented by a point set. With the point set, we have to iterate though every point to compute T in Equation 4. Although this model is comparatively slower, it is still moderately interactive in our implementation running at at least 30 frame per second (FPS). Finally, we can also see that the Plume dataset has a higher frame rate than the Isabel dataset. This is because the deformation operates on each vertex and the Plume dataset has a fewer number of vertices.

9 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,
- Preserves shapes of the deformed streamlines with minimum distortion,
- Provides smooth transition when distorting the deformed streamlines,
- And is well-suited for real-time GPU implementation.

In the 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 the user to freely explore the flow field without prior knowledge, our system provides an interactive 3D lens to move away the streamlines in a user-specified screen area at a given depth. Our system also allows the user to define the focus streamlines by selecting streamline bundles or by finding the streamlines that pass through a local region. In addition, our system can generate multiple focus regions by grouping the selected streamlines based on their screen locations. We develop an interactive streamline exploration system based on our deformation model. 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.

One limitation of our deformation model is that some deformed streamlines close to the center of the focus region may still get larger distortion, so the original shapes of these streamlines cannot be well preserved. 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 could not be well preserved by the deformation. More interactive tools can be added to our exploration system to allow more flexible feature exploration in screen space.

References

[1] CGAL, computational geometry algorithms library. http://www.cgal.org.

[2] I. Corouge, S. Gouttard, and G. Gerig. Towards a shape model of white matter fiber bundles using diffusion tensor mri. In *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 344–347 Vol. 1, April 2004.

[3] C. Correa, D. Silver, and M. Chen. Illustrative deformation for data exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1320–1327, Nov 2007.

[4] A. Fuhrmann and E. Groller. Real-time techniques for 3d flow visualization. In *Visualization '98. Proceedings*, pages 305–312, Oct 1998.

[5] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, pages 16–23, New York, NY, USA, 1986. ACM.

[6] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 11(4):457–468, July 2005.

[7] S. Gu, C. Lindsay, M. A. Gennert, and M. A. King. A quick 3d-to-2d points matching based on the perspective projection. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, ISVC '08, pages 634–645, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] J. Hoberock and N. Bell. Thrust: a parallel template library, 2010.

[9] E. LaMar, B. Hamann, and K. Joy. A magnification lens for interactive volume visualization. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pages 223–232, 2001.

[10] T.-Y. Lee, O. Mishchenko, H.-W. Shen, and R. Crawfis. View point evaluation and streamline filtering for flow visualization. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 83–90, March 2011.

[11] M. Levoy. Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*, 8(3):29–37, May 1988.

[12] L. Li and H.-W. Shen. Image-based streamline generation and rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 13(3):630–640, May 2007.

[13] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive cutaway illustrations of complex 3d models. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[14] K. Lu, A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and P. C. Wong. Exploring vector fields with distribution-based streamline analysis. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 257–264, Feb 2013.

[15] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma. View-dependent streamlines for 3d vector fields. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1578–1586, Nov 2010.

[16] O. Mattausch, T. Theußl, H. Hauser, and M. E. Gröller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. Technical Report TR-186-2-03-04, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Apr. 2003.

[17] M. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Visualization, 2003. VIS 2003. IEEE*, pages 401–408, Oct 2003.

[18] S. W. Park, B. Budge, L. Linsen, B. Hamann, and K. I. Joy. Dense geometric flow visualization. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'05, pages 21–

28, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[19] H. Qu, H. Wang, W. Cui, Y. Wu, and M.-Y. Chan. Focus+context route zooming and information overlay in 3d urban environments. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1547–1554, Nov. 2009.

[20] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 83–91, New York, NY, USA, 1992. ACM.

[21] J. Tao, C. Wang, C.-K. Shene, and S. H. Kim. A deformation framework for focus+context flow visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 20(1):42–55, Jan 2014.

[22] M. van Der Zwan, A. Telea, and T. Isenberg. Continuous navigation of nested abstraction levels. In M. Meyer and T. Weinkauf, editors, *In Proceedings of the EUROGRAPHICS - IEEE VGTC Symposium on Visualization 2012*, pages 13–17, Goslar, Allemagne, 2012. Eurographics Association.

[23] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman. The magic volume lens: an interactive focus+context technique for volume rendering. In *Visualization, 2005. VIS 05. IEEE*, pages 367–374, Oct 2005.

[24] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1216–1224, Nov 2010.

[25] X. Zhao, W. Zeng, X. Gu, A. Kaufman, W. Xu, and K. Mueller. Conformal magnifier: a focus+context technique with local shape preservation. *Visualization and Computer Graphics, IEEE Transactions on*, 18(11):1928–1941, Nov 2012.