Illustrative Deformation for Data Exploration

Carlos D. Correa, *Student Member, IEEE*, Deborah Silver, *Member, IEEE*, and Min Chen, *Member, IEEE*

Abstract—Much of the visualization research has focused on improving the rendering quality and speed, and enhancing the perceptibility of features in the data. Recently, significant emphasis has been placed on focus+context (F+C) techniques (e.g., fisheye views and magnification lens) for data exploration in addition to viewing transformation and hierarchical navigation. However, most of the existing data exploration techniques rely on the manipulation of viewing attributes of the rendering system or optical attributes of the data objects, with users being passive viewers. In this paper, we propose a more active approach to data exploration, which attempts to mimic how we would explore data if we were able to hold it and interact with it in our hands. This involves allowing the users to physically or actively manipulate the geometry of a data object. While this approach has been traditionally used in applications, such as surgical simulation, where the original geometry of the data objects is well understood by the users, there are several challenges when this approach is generalized for applications, such as flow and information visualization, where there is no common perception as to the normal or natural geometry of a data object. We introduce a taxonomy and a set of transformations especially for illustrative deformation of general data exploration. We present combined geometric or optical illustration operators for focus+context visualization, and examine the best means for preventing the deformed context from being misperceived. We demonstrated the feasibility of this generalization with examples of flow, information and video visualization.

Index Terms—Volume deformation, focus+context visualization, interaction techniques

1 Introduction

The purpose of visualization is to gain insight of complex structures through images and interactions. One primary objective of visualization is to aid us in building a spatio-temporal mental model of a phenomenon, process or physical quantity. The tools that we have in 3D data exploration to help build a mental model typically include real-time rendering, view transformation, transfer functions, segmentations and a collection of *focus+context* (F+C) techniques. However, most of these methods are active in “viewing” but passive in “handling”, which is quite different from our everyday activities for exploring a complex or unfamiliar object. With the prevalence of 3D visualization far and wide, it is important to explore new methodologies, which can enhance our comprehension and understanding of data, and enable us to build a mental model efficiently and accurately with active “handling” as well as “viewing”.

In science, engineering, medicine and education, hand-drawn illustrations often include two classes of elucidative methods. Firstly, multiple artistic painting styles are commonly employed to enhance, hide or emphasize different features of the object. This observation has led to a number of F+C techniques in visualization, such as cutaway views [11], ghosting [2] and importance-driven rendering [28]. Secondly, deformation is sometimes applied to parts of an object in order to depict the stages and the outcomes of a procedure, to uncover hidden features, or to reveal the spatial relationships between different components of the object. This observation has led to computer-generated surgical illustrations [8].

The aim of this work is to combine these two classes of elucidative methods into a single interactive framework, for the generation of effective F+C visualization, and to deploy the framework for data exploration in a wide range of applications. We call this framework *illustrative deformation*, in the sense that it is inspired by hand-drawn illustrations, and enables depiction of focus and context through a combination of rendering styles and geometric transformations.

- *Carlos D. Correa and Deborah Silver are with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, E-mail: cdcorrea, silver@caip.rutgers.edu*
- *Min Chen is with Department of Computer Science, University of Wales, Swansea, E-mail: m.chen@swansea.ac.uk*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

There are several challenges in producing effective F+C visualization using illustrative deformation. These include:

Generalization — It is highly desirable to have a common technical framework for a variety of applications and data types such as discrete points, lines, surfaces and volumes. Following the taxonomy by Tory and Möller [25], a general framework should support both continuous data (such as a volume) and discrete data (such as points and lines).

Geometric Integrity — There has been a reluctance to employ deformation to scientific data for fear that it would prevent accurate interpretation of the data. We believe that, on the contrary, with careful marking-up of contextual structures, and maintaining the integrity of the geometric and optical properties of feature of interest, illustrative deformation can aid data interpretation without leading to incorrect interpretation.

Interactivity — It is highly desirable to allow users to explore data interactively. Because this exploration becomes an active manipulation of data, it is important to provide interaction feedback and cues of the manipulation to the user. These challenges are the focus of this paper. Our main contributions are: (1) we introduce a novel approach to illustrative deformation, which enables focus preservation and context mark-up, based on a combination of multiple geometric and optical transformations. This is a new interactive F+C technique, allowing deformation-based data exploration to be deployed in scientific and information visualization, in addition to its traditional applications, such as surgical illustration (2) We present a generalized technical framework that supports the illustrative deformation of discrete data using deformable implicit representations. This framework allows seamless integration of continuous and discrete representations, facilitates complex optical and geometric transformations on discrete data without requiring complex geometry intersection tests, and offers unprecedented scalability when compared to explicit representations. We also present a novel mechanism for ensuring the geometric integrity of discrete data when undergoing deformation. We show how these operations can be implemented in traditional rendering pipelines, and in contemporary GPUs. (3) We demonstrate the effectiveness and feasibility through novel application of illustrative deformation in flow, video and information visualization. The initial feedback from potential users indicates that this new technique provides an intuitive means for actively exploring data and constitutes an important aid for scientific visualization

2 Related Work

An effective means for data exploration is the use of F+C techniques, which highlight focal objects in detail while depicting contextual objects in brief to provide an overview. F+C techniques, such as fisheye views [21], perspective wall [17], hyperbolic space [19] and rubber sheets [22], have been deployed extensively in information visualization.

F+C visualization is an intrinsic part of volume visualization. In a broader sense, without a F+C visualization, a volume dataset is just a solid volume cuboid containing a mixture of meaningful and insignificant information. Solutions that rely on the manipulation optical attributes and rendering styles include: *Selective rendering* allows the visualization of specifically selected parts of an object to enhance visibility of occluded parts. Fundamentally, the manipulation of transfer functions (e.g., [12, 14]) is a means for re-balancing parts of volume data in the focus and those in the context by changing the optical attributes. Other techniques that involve more semantic selection include volume decomposition [24], and opacity peeling [20]. *Non-photorealistic techniques* (e.g., [26, 16]) enable difference emphases to be rendered in different illustrative rendering styles, especially when illustrative rendering styles are combined with photorealistic styles in the same visualization. *Cutaway and ghosted views* allow occluded objects to be rendered by fading [28, 27] or removing [11, 30, 3] occluding parts. *Magic lenses* allow the changing of the parameters of the viewing system to magnify the desired features, such as in [29, 15, 5].

The above-mentioned F+C techniques rely on the manipulation of viewing attributes of the rendering engine and optical attributes of the data objects. Some solutions cannot effectively resolve the occlusion problem. Others can, but at the cost of decreasing the useful contextual information. Often the contextual information is completely suppressed. The use of deformation in volumetric objects has been proposed for animation, visualization and as a tool for computer graphics in general [6]. Most of previous approaches enable continuous deformation of biomedical objects. Recent approaches allow cutting a dataset using 3D widgets [18], the generation of surgical cuts [8] or exploded views [4] by allowing breaks in the rendering of volume objects. In such approaches, deformation is considered strictly a geometric transformation problem. Optical properties of the deformed object are usually defined to preserve the original optical properties of the undeformed object.

Illustrative deformation, discussed in this paper, attempts to balance the visibility of important parts (or *focus of attention*), while maintaining contextual information, by using a combination of optical transformations and deformation interactively. Considering deformation and optical transformations as interdependent operations has several advantages: First, optical transformations can be used as a visual feedback of the deformation. Second, because deformation alone may not solve the occlusion problem, optical transformations can be used in combination with deformation so that contextual information is still present. Previous approaches to volume deformation usually apply the transformation with little regards to the features of interest. In our approach, deformations are applied in a feature-sensitive manner. It can preserve the visual integrity of focus features, while marking up contextual features. It can be applied to multiple features, and complex features (e.g., user-defined curves).

One inherent advantage of deploying F+C techniques in volume visualization is that most volume objects to be visualized are familiar to the viewers. It is relatively easy to establish a perceptive view of the contextual parts of an object, even when it is peeled open or exploded apart. Hence it remains an interesting conceptual conundrum and a technical challenge whether or not illustrative deformation can be effectively deployed as a F+C technique for data objects that are more abstract or unfamiliar to the viewers. Answering this conundrum and addressing this challenge is the main motivation of this paper.

3 Focus and Context in Illustrative Deformation

In illustrative deformation, a F+C visualization is the product of a series of transformations on graphical primitives. In Section 4, we will describe a generalized technical framework for accommodating a variety of such primitives for representing continuous as well as discrete data to be visualized. For F+C visualizations, one can consider the following types of transformations.

Geometric Transformation: It causes changes to the shape of an object by applying deformation to its geometry in object space. The deformation can be *rigid* or *elastic*.

Optical Transformation: It causes changes to the appearance attributes (opacity, color, texture) of parts of an object. Making parts of an object disappearing is also considered as an optical (rather than geometric) transformation. Thus, cutting planes [30], ghosted views [2] and importance-driven rendering [28] are all optical transformations.

Viewing Transformation. It causes changes to the viewing system and camera parameters, resulting in some visual distortion, for instance, fisheye views [21] and magnification lenses [15, 29]. In principle, viewing transformations can be seen as restricted geometric transformation in image space. Because its co-existence with object-space geometric deformation in the same visualization may lead to confusion, we do not consider such transformation in this paper.

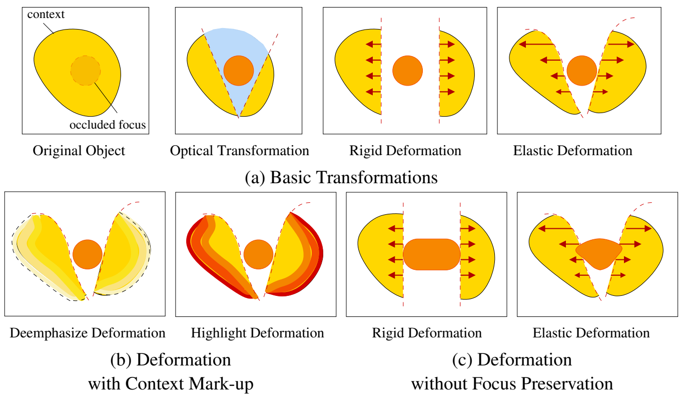

Fig. 1. Taxonomy of Illustrative Deformation Tools.

Fig. 1(a) shows an abstract representation of two types of transformation, geometric deformation and optical transformation. In this paper, we discuss the effects of combining the two types of transformations in order to obtain an effective and unambiguous F+C visualization involving deformation. The ambiguity typically arises from the difficulty in comprehending the original geometry of the key features in focus. Hence it is necessary to ensure *focus preservation*, such that no geometric or optical transformation is applied to the parts of the object in focus. As illustrated in Fig. 1(c), without a preserving the geometry of the focus, the geometric integrity of visualization would be compromised. The ambiguity can also arise from the confusion between focus and context, and not knowing what has been transformed geometrically or optically. Hence it is necessary to *mark up* the context, or parts of the context close to the focus, to illustrate the transformed context. Fig. 1(b) shows two examples of marking-up the deformed context by using optical transformation for deemphasizing and highlighting respectively.

In the following subsections, we introduce the notion of *focus of attention* (FoA), which facilitates *feature-sensitive transformation* with both focus preservation and context mark-up.

3.1 Focus of Attention (FoA)

The most important goal of a F+C visualization based on deformation is to focus the viewer's attention on certain features of interest. *Focus of Attention* (FoA) defines a region of a data domain, or a feature of a data object, which is of particular interest in data exploration and will remain untransformed (geometrically or optically) during deformation. Similar to the color and opacity specification, an FoA can be defined via a transfer function, typically, $H : \mathbb{R} \rightarrow \mathbb{R}$, or scalar field $H : \mathbb{E}^3 \rightarrow \mathbb{R}$, where $\mathbb{R}$ denote the set of all real numbers and $\mathbb{E}^3$ denote 3D Euclidean space. We call H the *Levels of Desired Attention* (LDA). The former implies the dependence of an FoA to the values of the dataset concerned, whilst the latter removes such a restriction, allowing an FoA be defined in many different ways (e.g., interactively by the viewers, or dynamically by a computational-steering monitor). In the following discussions, we assume that H is a scalar field, and refer the LDA value at $\mathbf{p}, \mathbf{p} \in \mathbb{E}^3$ as $H(\mathbf{p})$ directly. Without losing generality, we also assume that the values of H fall in the subdomain of $[0,1]$.

The values of H are typically divided into three ranges: $H(\mathbf{p}) \in [0, c_{FoA}] \cup (c_{FoA}, f_{FoA}) \cup [f_{FoA}, 1]$, where $0 \leq c_{FoA} \leq f_{FoA} \leq 1$. All the points with $H(\mathbf{p}) \in [f_{FoA}, 1]$ are said to be *in the focus*, and no geometric and optical transformation should be applied to these points. All the points with $H(\mathbf{p}) \in [0, f_{FoA})$ are said to be *in the context*, and they can be transformed. The values in the second interval (c_{FoA}, f_{FoA}) indicate those points in a transitional context region, which are typically needed to be marked up to alleviate the potential ambiguity or confusion. A weighting function can thus be defined based on $H(\mathbf{p})$, for example, as:

$$\eta(\mathbf{p}) = \begin{cases} 0 & H(\mathbf{p}) \in [0, c_{FoA}] \\ \frac{H(\mathbf{p}) - c_{FoA}}{f_{FoA} - c_{FoA}} & H(\mathbf{p}) \in (c_{FoA}, f_{FoA}) \\ 1 & H(\mathbf{p}) \in [f_{FoA}, 1] \end{cases} \quad (1)$$

This example weighting function, $\eta(\mathbf{p})$, specifies linearly the level of context marking-up in the transitional context region. Non-linear functions can also be defined in a similar manner.

3.2 Focus Preservation

There are a number of approaches to define deformations in 3D space. Most surface-based deformation approaches use explicit forward transformation of vertices. In forward deformation, preserving the focus is an easy task, and can be done by modulating the transformation with the weighting function η. However, volumetric objects are typically deformed using an inverse transformation. Here we show how focus preservation can be achieved when defining deformation as an inverse operation.

The 3D displacement mapping introduced in [7] offers a powerful and efficient technical framework for geometric transformation. It can facilitate rigid as well as elastic transformation. However, unlike [7, 8], this work has extended this framework significantly to accommodate a more general definition of multiple and dynamic FoA, discrete data and multi-space deformation (see Section 4 for details). An elastic deformation can be defined as an inverse transformation $T : \mathbb{E}^3 \mapsto \mathbb{E}^3$:

$$T(\mathbf{p}) = \mathbf{p} + D(\mathbf{p}) \quad (2)$$

for each point $\mathbf{p} \in \mathbb{E}^3$, where $D : \mathbb{E}^3 \mapsto \mathbb{R}^3$ is a 3D displacement map. This definition can also be used to represent rigid transformations. However, for rigid transformations, a more compact representation (using a rotation matrix and a translation vector) is preferred.

The preservation of the focus can be achieved geometrically by applying:

$$T(\mathbf{p}) = \mathbf{p} + (1 - \eta(\mathbf{p}))D(\mathbf{p}) \quad (3)$$

This ensures that the transformation is injective for the points in focus, which is critical to the visualization. A given sampling point $\mathbf{x}$ in the focus will be mapped to a distinct $\mathbf{x}' = T(\mathbf{x})$ because $1 - \eta(\mathbf{x}) = 0$. However, the transformation is not injective for the points in context, and it may happen that a sampling point in the context maps to a point in the focus, which is undesirable. This may occur when $\eta(\mathbf{p}) = 0$, but $\eta(T(\mathbf{p})) = 1$. To alleviate this, one may carefully design the weighting function based on H, so that a point in the context is not mapped into the focus, for example, by using a distance field representation of the focus region.

We also define focus preservation as an optical transformation. We use two field representations as a generic form for an optical specification (e.g, color and opacity) of an object: $F_c : \mathbb{E}^3 \rightarrow [0,1]^3$ to denote the original color specification, and $F_\alpha : \mathbb{E}^3 \rightarrow [0,1]$ to denote the original opacity specification. Both can be captured, simulated or derived from the original data values using transfer functions. The conventional deformation, such as for surgical illustration [7], normally assures the preservation of optical specification for all points with:

$$c(\mathbf{p}) = F_c(T(\mathbf{p})), \quad \alpha(\mathbf{p}) = F_\alpha(T(\mathbf{p}))$$

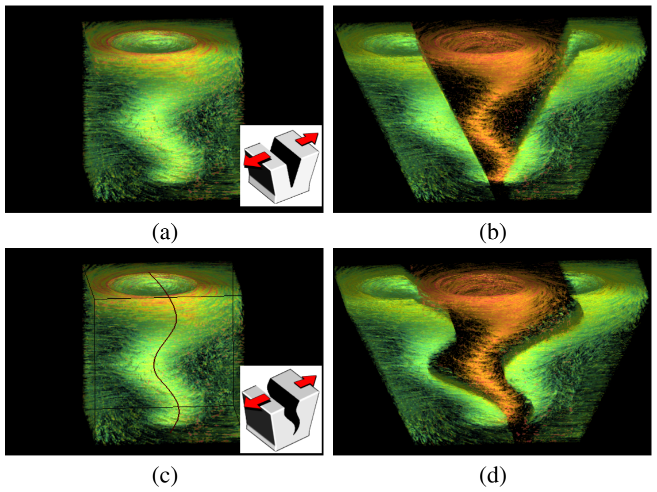

Fig. 2. Illustrative Deformation of a Tornado Dataset. Top: Here, the context regions are simply defined by splitting the volume along a vertical line. Bottom: The contextual regions are defined by splitting along a user-drawn line that matches the shape of the focus.

where $c(\mathbf{p})$ and $\alpha(\mathbf{p})$ are the color ad opacity to be displayed at $\mathbf{p}$.

Let $C_{con}(\mathbf{p})$ and $\alpha_{con}(\mathbf{p})$ be the transformed color and opacity for the points in the context. Their specification can be the original representations F_c and F_α, or mark-up functions as defined in Section 3.3. Then, the color and opacity to be displayed at a sample point $\mathbf{p}$ are:

$$c(\mathbf{p}) = \begin{cases} F_c(\mathbf{p}) & \eta(\mathbf{p}) = 1 \\ \eta(\mathbf{p}) F_c(\mathbf{p}) + \big(1 - \eta(T(\mathbf{p}))\big) C_{con}(T(\mathbf{p})) & \eta(\mathbf{p}) < 1 \end{cases} \quad (4)$$

$$\alpha(\mathbf{p}) = \begin{cases} F_\alpha(\mathbf{p}) & \eta(\mathbf{p}) = 1 \\ \eta(\mathbf{p}) F_\alpha(\mathbf{p}) + \big(1 - \eta(T(\mathbf{p}))\big) \alpha_{con}(T(\mathbf{p})) & \eta(\mathbf{p}) < 1 \end{cases} \quad (5)$$

Note that the color is preserved, as $c(\mathbf{p}) = F_c(\mathbf{p})$, for points in the focus. Moreover, it ensures that points in full context are not mapped back to the focus. In such cases, $\eta(\mathbf{p}) = 0$. When $\eta(T(\mathbf{p})) = 0$, $c(\mathbf{p}) = C_{con}(T(\mathbf{p}))$, which is the context region, and when $\eta(T(\mathbf{p})) = 1$, $\alpha(\mathbf{p}) = 0$, which denotes empty space. On the other hand, points in the transitional region result in a blending of the deformed and undeformed positions. In this case, the focus and the context share a transition region, which is blended optically. Although some points in the transition region may be mapped to points in the focal region, the result is visually acceptable, as it depicts the FoA and the context as having a fuzzy boundary.

3.3 Context Marking-up

Simply preserving color and opacity of all parts of an object may be alright for applications such as surgical illustration, but it is certainly not adequate for illustrative deformation of objects (e.g., a flow field and a graph) that are not intuitively recognizable in real life. It is thereby necessary to distinguish untransformed focus region from the transformed context region. Since maintaining the visual integrity of the focus region cannot be compromised, it is best to mark up the context region.

Mark-up effects can take many different forms, including both optical and geometric transforms. When alleviating the ambiguity due to deformation is the main concern, optical transformation can be used to highlight or deemphasize the deformed context region. Typical highlighting effects include introducing easily-recognizable colors (such as bright prime colors) and increasing opacity of points in the transitional context region. On the contrary, typical deemphasizing effects are transforming color hues towards bland colors (e.g. grey), or increasing transparency. Examples are shown in Fig. 4 and Fig. 10, where the deformed glyphs are rendered more transparently.

Let M_c and M_α be general mark-up functions for color and opacity respectively. Let $\omega(\mathbf{p}) \in [0,1]$ be a weighting function that determines

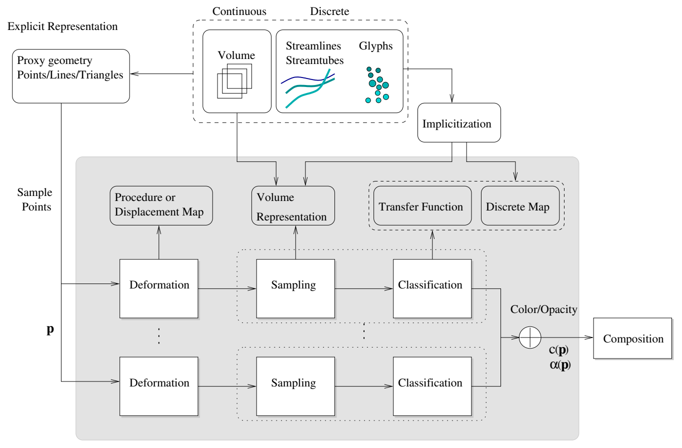

Fig. 3. Flowchart of Illustrative Deformation.

the desired degree of mark-up. Then, the color and opacity transformations for the context are:

$$\begin{aligned} C_{con}(\mathbf{p}) &= (1-\omega(\mathbf{p}))F_c(\mathbf{p}) + \omega(\mathbf{p})M_c(\mathbf{p}) \\ \alpha_{con}(\mathbf{p}) &= (1-\omega(\mathbf{p}))F_\alpha(\mathbf{p}) + \omega(\mathbf{p})M_\alpha(\mathbf{p}) \end{aligned}$$

An example weighting function is based on amount of deformation, e.g., $\omega(\mathbf{p}) = ||T(\mathbf{p}) - \mathbf{p}||$, where $||\cdot||$ denotes vector magnitude.

4 A Generalized Deformation Framework

In visualization, data and models are usually represented by volumes, surfaces, lines, glyphs and points. Volumes differ from others in that no geometry is given explicitly. Instead each volume defines a *continuous* spatial subdomain, where every point is associated with one or more values, which may encode color, opacity, and implicit geometric information, or can be used to derive such information using transfer functions. Our previous work on deformation [7, 8] was designed for such data representations in medical illustration. On the other hand, data representations composing *discrete* primitives such as lines, glyphs and points are much more common in scientific and information visualization. For such explicit representations, geometric and optical transformations are typically applied to the low-level primitives, such as vertices, and are interpolated along the high-level primitives such as faces. However, it is difficult to maintain geometric integrity and mark up contextual information without incurring costly intersection computation and neighborhood search. Our aim to extend the approach of illustrative deformation for data exploration in a broader range of applications led us to develop a generalized deformation framework that can work with discrete as well as continuous data. This framework is depicted in Fig. 3, as the composition of multiple geometric and optical transformations on sampling points **p**. These points are deformed via a displacement map or procedure, then sampled according to a volume representation (which can be an implicit representation of lines and points) and finally classified via an optical operator.

4.1 Composite LDA

Section 3 considers the use of a single FoA defined by a single LDA function. In many situations, users may wish to have multiple LDAs to influence the same visualization. Some may be feature-sensitive (e.g., defined using a transfer function), and others may be region-based (e.g., specified by the user dynamically). Fig. 3 shows the overall technical framework of our implementation. The lower part of the diagram shows a set of parallel operations for realizing composite illustrative deformation based on multiple LDAs, $H_1, H_2, \ldots, H_n$, each of which is associated with a focus weighting function $\eta_i(\mathbf{p})$ and a context weighting function $\zeta_i(\mathbf{p}), i = 1..n$. As mentioned in 3.1, with a single LDA, we normally have $\zeta(\mathbf{p}) = 1 - \eta(\mathbf{p})$. However, this is not adequate. For multiple LDAs, we use a more complex condition:

$$\sum_{i=1}^{n} \eta_i(\mathbf{p}) + \sum_{i=1}^{n} \zeta_i(\mathbf{p}) = 1, \qquad \forall \mathbf{p} \in \mathbb{E}^3$$

Consider that each LDA H_i is coupled with a geometric transformation T_i, we can generalize Eqs. 4 and 5 as:

$$c(\mathbf{p}) = \begin{cases} F_c(\mathbf{p}) & \exists i, \eta_i(\mathbf{p}) = 1 \\ \sum_{i=1}^{n} \big(\eta_i(\mathbf{p})F_c(\mathbf{p}) + \zeta_i(T_i(\mathbf{p}))C_{con}(T_i(\mathbf{p}))\big) & \forall i, \eta_i(\mathbf{p}) < 1 \end{cases}$$

$$\alpha(\mathbf{p}) = \begin{cases} F_\alpha(\mathbf{p}) & \exists i, \eta_i(\mathbf{p}) = 1 \\ \sum_{i=1}^{n} \big(\eta_i(\mathbf{p})F_\alpha(\mathbf{p}) + \zeta_i(T_i(\mathbf{p}))\alpha_{con}(T_i(\mathbf{p}))\big) & \forall i, \eta_i(\mathbf{p}) < 1 \end{cases}$$

Fig. 2 shows a deformation of the tornado dataset using a LDA defined in terms of the vector field magnitude. This is mapped onto one focus weighting function, η_a, for the central core of the data, and two context weighting functions, ζ_a and ζ_b, for the two halves of the volume. We can feed η_a and ζ_a into one pipeline in Fig. 3, and ζ_b into another. By moving the contextual parts in opposite directions, we achieve a split. The bottom figure depicts the case where the contextual regions are defined as the two halves at either side of a user-drawn curve that matches the shape of the focus. The effect is a better view of the FoA. The advantage of defining the context with multiple ζ_i is the ability to apply deformation to different parts of the context individually. For example, Fig. 10 shows a F+C view of a 3D scatter plot, where the FoA is the region along the middle part of the plot. Two context regions deform in the opposite direction during a splitting operation. A more complex example of multiple LDAs is shown in Fig. 10(e), where a global zooming operation is employed to explore the data, and a deformation is applied to the context to neutralize the zooming effects on the context.

4.2 Implicit Glyphs and Lines

In visualization, rendering the *explicit* representations of glyphs and lines is still the predominant approach. Because of the powerful capabilities of the current GPU technology, it has been possible to render discrete primitives in their *implicit* representations. In this work, we consider lines and glyphs as implicit functions defined in a volumetric domain, which can be deformed directly using space warping and rendered using a volume renderer. One such implicit function is a distance field, where voxels store the Euclidean distance to the closest primitives. In the case of a set of points, an implicit function encodes the geometry of the corresponding spheres surrounding the points; and in the case of a set of streamlines, an implicit function encodes the geometry of the corresponding streamtubes. This approach can also be extended to other data representations involving discrete primitives. For instance, interactive deformation of implicit meshes, useful for the rendering of isosurfaces, has been suggested [9]. Nevertheless, the generation of implicit representations of lines and points is very fast, in comparison to arbitrary meshes. Timing results for generative and rendering of these implicit representations are given in Section 6.2.

Let $f(\mathbf{p})$ be the implicit representation of a set of lines or points. An example is the distance field, where voxels store the Euclidean distance to the closest line or point. Rendering of deformable streamtubes or spherical glyphs of a specific radius τ is done by using the conventional opacity transfer function: $O(\mathbf{p}) = u\big(\tau - f(\mathbf{p})\big)$ where $u(x)$ is typically a unit step function. An example of applying illustrative deformation to a set of stream tubes is shown in Fig. 4, where we compare different deformation and optical mark-up methods.

Without a very fine sampling interval, the volume rendering integral typically gives rise to amorphous lines with fuzzy boundaries. While this effect can be useful for marking up contextual information, it is not desirable in situations where solid streamlines produce more effective visualization. For solid streamlines, we hence search for the intersections with specific isosurfaces of the distance volume, instead of doing compositing. This is achieved by sampling the volume until reaching the zero-set $\tau - f(T(\mathbf{p})) = 0$, for each sample point **p**.

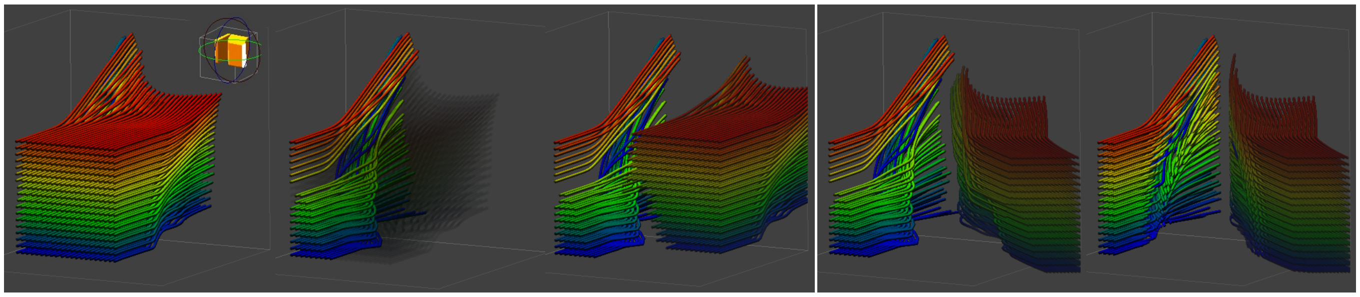

Fig. 4. Deformation to the stream tubes of the blunt fin dataset, with FoA defined to preserve the stream lines on the left (a) Original dataset (b) Optical transformation only (c) Optical transformation and rigid deformation (d) Optical transformation and elastic deformation (e) Without discrete FoA deformation, stream tubes are cut, making it difficult to understand.

Although discrete data is often represented and deformed explicitly, there are a number of advantages for performing it implicitly: (1) It is easier to incorporate volumetric data, such as registered MRI for the case of fiber tracts (Fig. 7), or vector field magnitude for the case of flow visualization (Fig. 2). (2) It is faster to incorporate soft shadows, necessary for better depth, motion and deformation cueing, as shown in Fig. 5. (3) Controlling the thickness of the glyph or streamtube is a simple operation, which does not require extra steps to control the smoothness of a mesh, and (4) It scales better to large numbers of glyphs (up to millions). This approach scales with the rendering area instead of the number of glyphs, which makes it feasible to render a large number of glyphs at interactive rates. This observation is exploited by the approach in [13], where a large number of particle data is rendered in real-time using raytracing.

4.3 LDA Specification for Discrete Data

The above specification of the weighting functions, $\eta(\mathbf{p})$ and $\zeta(\mathbf{p})$, is not suitable for discrete data. The application of such an LDA may result in breaks in the discrete glyphs. Although cutting streamtubes for fiber tracking is reminiscent of the physical cutting of white matter tracts, it is not desirable for flow or graph visualization. To consider the discrete nature of data, we make use of a labeling scalar field, L, where each voxel has a unique identifier of the closest line or point. The specification of an LDA is thus based on L rather than the implicit function representing the discrete data. From L, we can create a look-up table, P, which maps each unique label to a 3D reference point associated with the corresponding glyph. For example, for the case of a sphere, we use its center as the reference point, while for a streamtube, we use its geodesic centroid.

This gives a new form of LDA as $G(\mathbf{p}) = P(L(\mathbf{p}))$, which is referred to as discrete LDA. From $G(\mathbf{p})$, we can derive $\eta(\mathbf{p})$ and $\zeta(\mathbf{p})$ in the same way as discussed in previous sections.

An example is shown in Fig. 4(d), where a discrete LDA is defined for the blunt fin dataset. In comparison, without discrete LDA (Fig. 4(e)), streamtubes are broken. The application of Eq. (5) works only for the case of composition. For direct rendering streamtubes by intersecting with the zero-set of an implicit representation f, a different approach must be taken. We hence split f into two implicit representations for the focus and context respectively:

$$f_F(\mathbf{p}) = \begin{cases} \tau - f(\mathbf{p}) & \xi(\mathbf{p}) \geq 1-\varepsilon \\ -f_{max} & \xi(\mathbf{p}) < 1-\varepsilon \end{cases}, \quad f_C(\mathbf{p}) = \begin{cases} -f_{max} & \xi(T(\mathbf{p})) \geq 1-\varepsilon \\ \tau - f(T(\mathbf{p})) & \xi(T(\mathbf{p})) < 1-\varepsilon \end{cases}$$

where $\xi(\mathbf{p}) = \eta(G(\mathbf{p}))$ is a weighting function for a discrete LDA, f_{max} is the maximum distance to the focus of attention, and ε (typically 0), is a threshold value to account for precision errors in the raycasting process. Then, we obtain a discrete F+C visualization by sampling the combined implicit representation

$$g(\mathbf{p}) = \min(f_F(\mathbf{p}), f_C(\mathbf{p})) \tag{6}$$

An example can be seen in Fig. 5.

4.4 Multi-space Deformation

Conventionally, the geometric entities of illustrative deformation are defined using the same coordinate system. In the context of deformation, we can consider three different coordinate systems:
Ξ_v: the visual or renderable space, usually defined with a proxy geometry.
Ξ_d: deformation space, usually defined as a displacement map.
Ξ_o: original object space for continuous or discrete data, often defined as a rectilinear space for storage as a 3D texture.

Therefore, we can define the following mappings between the different spaces: $W_{v,o} : \Xi_v \mapsto \Xi_o$, $W_{d,o} : \Xi_d \mapsto \Xi_o$ and $W_{v,d} : \Xi_v \mapsto \Xi_d$, and a multi-space deformation is defined as:

$$T(\mathbf{p}) = W_{v,o}(\mathbf{p}) + W_{d,o}\left(D(W_{v,d}(\mathbf{p}))\right) \tag{7}$$

Fig. 8 shows an example of video visualization, where the 3D video volume is rendered into a torus space to better use the screen space [10]. Because the original data and the displacement are defined as a 3D texture, then $W_{v,o} = W_{v,d}$ are mappings from toroidal to rectilinear spaces, while $W_{d,o}$ is the identity transformation.

5 Applications

We examine below the use of illustrative deformation in different application domains.

5.1 Flow Visualization

Flow visualization is essential for the understanding of the dynamics of fluids and gases by representing the movement of particles visually. Flow information is usually represented in a volume by sampling the velocity vectors in a regular grid at different moments in time. However, visualizing overall movement is complicated due to the large amount of information. For this reason, a number of mechanisms have been widely used, such as the explicit rendering of stream

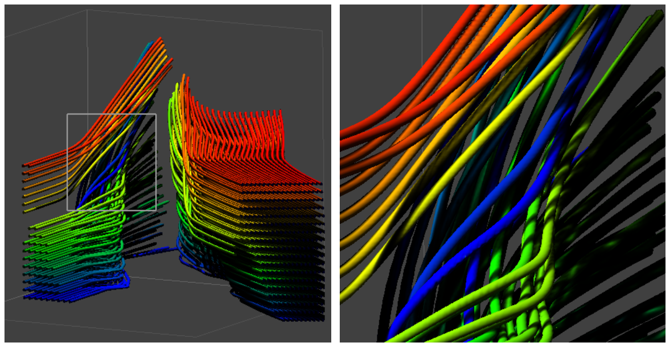

Fig. 5. Illustrative deformation of stream tubes with soft shadows.

lines, ribbons or tubes to depict flow path, or the use of Linear Integral Convolution (LIC) [23]. A challenge in both approaches is visualizing internal 3D flow. Fig. 2 shows an example of flow visualization of the tornado dataset using LIC textures. We make two important considerations for ensuring that deforming a flow does not lead to misinterpretation of data. (1) We avoid cutting through the flow. In the case of stream tubes, this is achieved with discrete FoAs, as depicted in Fig. 4. (2) We render deformed data with optical transformations, so that is understood as a means for context information.

In another experiment, we deformed data from a plasma turbulence simulation. Interestingly, the use of a continuous FoA defined radially along the centerline of the torus, provides a F+C view of flux surfaces, which are of interest to scientists. Examples can be shown in Fig. 6. By controlling the radius of the FoA, it is possible to interactively explore the different flux surfaces.

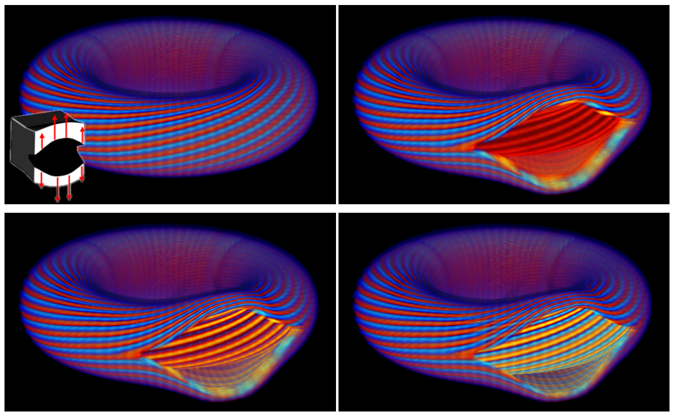
Fig. 6. Deformation of Plasma Turbulence Data. The use of FoA defined radially from the centerline of the torus provides a F+C view of flux surfaces.

5.2 Fiber Tract Visualization

Another application of deformation of implicit lines is the visualization of white matter tracts in the human brain. These are usually obtained from diffusion tensor magnetic resonance images. Visualizing DTI datasets is useful for understanding the directional qualities of brain tissue, and most common approaches use glyphs or streamlines [31]. Fig. 7 shows cutting through the corpus callosum to visualize the internal fiber bundles on one brain hemisphere, which otherwise are occluded by the other hemisphere. We found deformation very intuitive in this case, as it is reminiscent of physical cuts that may be performed on specimens.

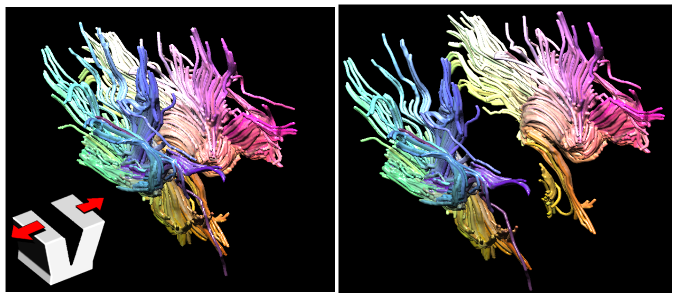
Fig. 7. Deformation of Fiber Tracts from Diffusion Tensor Imaging. Fiber Tracts are depicted as streamtubes, which can be cut and deformed interactively.

5.3 Video Visualization

Video visualization is a visual process for extracting meaningful information from a video sequence. It was introduced by Daniel and Chen as a mechanism for showing moving images in a static 3D volume [10]. In general, a video can be understood as a 3D volume by using the time dimension as a spatial dimension. Using volume rendering techniques, it is possible to have a depiction of the entire video in a single image. One of the problems with these tools is the inability to show the original frames and still focus on an area of interest, due to occlusion. Proposed solutions make use of semi-transparent volumes derived via background substraction of each frame. This solution reduced the amount of contextual information that can be derived from the visualization. In our approach, we solve this problem using deformation, as depicted in Fig. 8. Here, a retracting deformation is used to manipulate the contextual information when visualizing the signature of a person walking. We also depict the use of multi-space deformation, in particular, the use of a toroidal shape, as proposed in [10]).

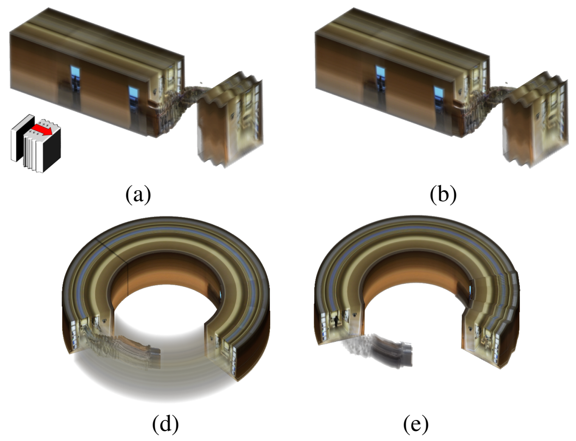

Fig. 8. Deformation in Video Visualization using a retracting deformation, depicted in (a), using multi-space deformation. (b)-(c) Rectilinear coordinate system (d)-(e) Using Toroidal coordinate system. Wave deformation is used to mark up the contextual information.

5.4 Glyph and Graph Visualization

Another need for discrete data is in the visualization of glyphs. 3D glyphs, e.g., spheres, are often used to represent points in 3D space. Here we consider two example applications: the visualization of a 3D scatter plot, where sample points are plotted in 3D using three continuous scalars as spatial coordinates and a a fourth scalar as color. Fig. 10 shows a scatter plot of data from the cosmological simulation Enzo [1]. The X,Y,Z coordinates correspond to gas energy, total energy and density of star particles, while color represents temperature. A discrete FoA is used to split the dataset and focus on a cluster of interest.

In graph visualization, vertices in a graph can be represented in

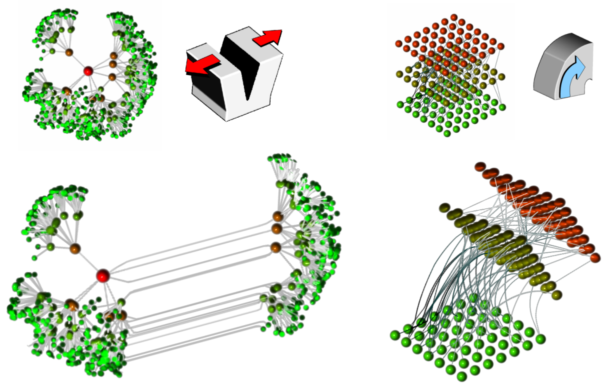
Fig. 9. Deformation of a Hyperbolic graph and a multi-tier graph using a bending transformation.

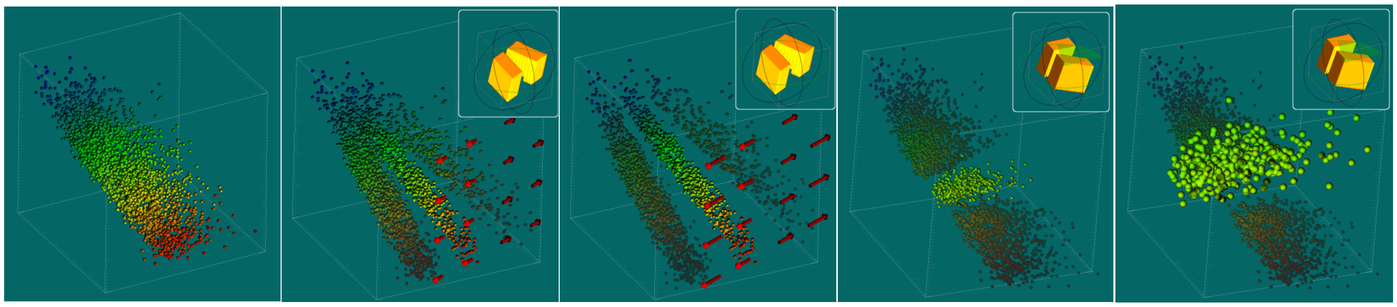

Fig. 10. Scatter Plot Visualization with Interaction Widget (inset). (a)-(c) Sequence for splitting while preserving FoA (defined along the center of the split). (d) Rotation can be used to select the region of interest along a different dimension. (e) Scaling of the FoA helps to get a better view.

a similar fashion. Deformation of edges, however, cannot follow the same series of transformations, since it is undesired to cut an edge. Instead, we deform vertices and edges independently, and we use an algebraic operation to combine the result of both, as : $c(\mathbf{p}) = C_v(T_v(g(\mathbf{p}))) + C_e(T_e(\mathbf{p}))$ where C_v and C_e are optical transformations for vertices and edges, respectively, while T_v and T_e are deformations. T_e differs from T_v in that it must be continuous. In addition, vertices are transformed discretely, as defined in Section 4.3. Fig. 9 shows two examples of deforming a hyperbolic graph, and a multi-layered graph.

6 Implementation

Our approach can be incorporated in most traditional volume rendering systems. The algebraic operations depicted in Fig. 3 can be implemented as a multi-pass process in most contemporary visualization processes, or, with the aid of the GPU, in a single-pass fragment shader. Each step in the process, identified as a box in Fig. 3, is a function that application developers can modify to suit their own needs. In our particular implementation, we use GPU-based raycasting. It proves to be a more flexible approach than slice-based volume rendering, as it enables us to use adaptive sampling of the volume, essential for the rendering of deformable stream tubes and glyphs.

6.1 Interaction Issues

One of the goals of our work is to make our approach interactive. As the complexity of the illustrative deformation and the size of the dataset grow, it becomes important to use progressive sampling. We exploit dynamic branching capabilities of new GPUs to allow variable sampling rates of the volume. Low resolution is used when the user changes the view or the deformation parameters, and high resolution is activated once the user stops interacting. There are a number of technical challenges, including handling objects in 3D space and the interaction with complex transfer functions. Because there is a lot of research devoted to handling transfer functions, we focus in two new interaction challenges introduced by our framework: (1) manipulating the deformation space and (2) presenting interaction feedback to the user.

Interaction with the deformation parameters is done by applying rotations and translations to the deformation space. By moving the deformation space, the user can remove certain parts of an object, or define the FoA. However, this process is a complex task as the user only perceives the effects of deformation, rather than the deformation itself. For this reason, we provide a “widget” view of the deformation, which appears as a volumetric icon of the deformation. The icon is automatically generated from the displacement, by deforming a volumetric cube. A number of widgets, color coded according to the axis of rotation, allow the user to rotate the deformation along a specific axis. A wireframe box is also used to depict the object space. Fig. 10 depict a series of deformations on a 3D scattered plot, obtained by translating and rotating the deformation space. Note the changes in the icon widget.

Another important aspect is the provision of interaction feedback. Because our approach does not incorporate additional modes of interaction, such as haptic or auditive, we rely on visual cues to represent interaction. One such cue is the use of deformation arrows, which represent the principal direction of the deformation as the user moves the deformation space. We automatically obtain the direction of deformation from the displacement field D. To avoid clutter, we only show the arrows corresponding to a plane in the direction of the movement of the deformation plane. An example is shown in Fig. 10(b,c), where the user translates the deformation in the z-direction, to control the amount of the splitting. The direction and magnitude of the arrows represent the direction and amount of deformation.

6.2 Preprocessing

To handle discrete data (such as lines and points) in our framework, we can generate implicit representations of such datasets. As described in Section 4, there are a number of advantages, since the process for generating an implicit representation is very fast and can be performed seamlessly. By exploiting GPU programmability, it can be included in the rendering pipeline by intercepting the rendering primitives before they are sent to the rasterization process. For this reason, this preprocess of discrete data does not hinder the generality of our approach. In this paper, we developed an implicitization process on the CPU, which computes the distance field of an object only in the vicinity of the points and lines. In our experiments, we computed a 256^3 implicit representation of 10000 and 100000 points, which took 0.487 and 3.4 sec., respectively. For lines, 10000 and 100000 line segments took 2.887 and 31.287 sec., respectively. These values are obtained using a CPU-only implementation on a XEON 2.4 Ghz laptop. We believe that the process can be accelerated considerably with a GPU-implementation, so that it can be used in real-time. This is of particular interest for time-varying particle systems or flow simulations, where points and streamlines change dynamically. Another advantage of using implicit representations, is that rendering scales better for a large number of glyphs. In our experiments, 10000 are rendered using explicit representations in about 0.3 (low quality) to 1 sec. (high quality). 100000 glyphs are rendered at about 3 sec. per frame. In comparison, our implicit renderer takes from about 0.2 sec. (in the interactive mode using progressive sampling) up to 1.8 sec. (in high-quality mode), for a viewpoint of size 512×512, regardless of the number of points.

6.3 Evaluation

We experimented with a number of datasets, ranging in size from 64^3 to 256^3. We obtained interactive results for all our datasets on a XEON processor with a Quadro FX 4400 GPU, using a 512×512 viewport and a sampling distance of $d = 1$ voxel. In our case, a comparative evaluation is more informative. Table 1 shows the comparison between our different methods vs. a basic rendering system. We compared for the case of volume rendering, e.g., Fig. 4 and the case of implicit rendering via ray intersection, e.g., Fig. 5. Progressive sampling is used when the user is interacting, for $d = 4$, resulting in a four-fold improvement in performance (up to 20 fps). The overhead on introducing deformation and FoA will become smaller in future GPUs. Interactivity can be seen in the accompanying video and at : `http://www.caip.rutgers.edu/˜cdcorrea/deformation`

Table 1. Summary of rendering time for different methods.

F+C Method	Volume Rendering (s)	Implicit Rendering (s)
Basic Rendering	0.132	0.075
Deformation	0.217	0.131
Deformation with FoA	0.336	0.169
Deformation with Discrete FoA	0.399	0.279

7 Validation and Conclusions

The software tool developed in this work was demonstrated to a number of scientists, using the plasma turbulence data as an example. Although the scientists did not use the system directly, as it still requires the study of effective interaction widgets, they were enthusiastic. Contrary to our initial hypothesis that scientists may be reluctant to deform the data for fear of misinterpretation, they found it very intuitive and did not raised a concern with the concept of deformation of their data. In fact, they expressed interest in focus-preserving deformations that allow them to visualize the flux surfaces, as depicted in Fig. 6. They also suggested deformation operations where the torus surface is "flattened" interactively, in order to visualize the entire flux surface in a single view. In another informal validation, images from the deformation of DTI-based fiber tracking was presented to two brain scientists, who expressed interest in our framework. They commented on the deformation of individual fiber bundles as an interesting application, which validates the need for focus-preserving deformation, especially for discrete data such as streamtubes. Although this was an informal validation, we were very encouraged by the comments. A full evaluation is still an important aspect that needs to be addressed in the near future. This will be a time-consuming task, as our framework spans a variety of applications with very different groups of users, each of which must be analyzed independently.

We presented a novel framework for data exploration through illustrative deformation, which combines active manipulation of the spatial data with opacity and color transformations. Our framework incorporates the definition of optical transformations, such as cutaways, ghosted views and clipping, with manipulation operators, such as cuts, exploded views and deformation, and can be applied to both continuous and discrete data. We showed the generality and flexibility of our approach through a number of examples, including flow visualization, video visualization, fiber tractography, scatter plots and 3D graphs. Initial comments from scientists and visualization experts are encouraging. We believe that *illustrative deformation* is an important aid in data exploration, where active "handling" of data is as important as active "viewing", which has been the predominant paradigm in visualization.

Acknowledgements

We would like to thank Dr. Han-Wei Shen, The Ohio State University, for the tornado and LIC texture datasets. Other datasets are from Nasa Advanced Supercomputing Division, University of North Carolina, Swansea University, San Diego Supercomputer Center and the SciDAC Center for Plasma Edge Simulation Project, to which we are grateful.

References

[1] ENZO - AMR cosmological simulation, UC San Diego.

[2] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving volume rendering. In *Proc. of EuroVis*, pages 69–76, 2005.

[3] S. Bruckner and M. E. Gröller. Volumeshop: An interactive system for direct volume illustration. In *Proc. of IEEE Visualization*, pages 671–678, 2005.

[4] S. Bruckner and M. E. Groller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.

[5] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, 1997.

[6] H. Chen, J. Hesser, and R. Manner. Ray casting free-form deformed-volume objects. *The Journal of Visualization and Computer Animation*, 14:61–72(12), 2003.

[7] C. Correa, D. Silver, and M. Chen. Discontinuous displacement mapping for volume graphics. In *Proc. Volume Graphics*, 2006.

[8] C. Correa, D. Silver, and M. Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1069–1076, 2006.

[9] C. D. Correa and D. Silver. Programmable shaders for deformation rendering. In *Eurographics/SIGGRAPH Graphics Hardware*, 2007.

[10] G. Daniel and M. Chen. Video visualization. In *VIS '03: Proc. of the 14th IEEE Visualization 2003 (VIS'03)*, pages 409–418, 2003.

[11] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. *Computer Graphics Forum*, 22(3):525–532, 2003.

[12] D. Ebert and P. Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *Proc. IEEE Visualization*, pages 195–202, 2000.

[13] T. Ize, A. Kensler, I. Wald, C. P. Gribble, and S. G. Parker. A coherent grid traversal approach to visualizing particle-based simulation data. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):758–768, 2007.

[14] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proc. IEEE Visualization*, pages 255–262, 2001.

[15] E. LaMar, B. Hamann, and K. I. Joy. A magnification lens for interactive volume visualization. In *Proc. IEEE Pacific Conference on Computer Graphics and Applications*, pages 223–231, 2001.

[16] A. Lu, C. Morris, D. Ebert, P. Rheingans, and C. Hansen. Non-photorealistic volume rendering using stippling techniques. In *IEEE Visualization*, pages 211–218, 2002.

[17] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The perspective wall: detail and context smoothly integrated. In *Proc. ACM CHI*, pages 172–180, 1991.

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

[19] T. Munzner. Exploring large graphs in 3D hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, 1998.

[20] C. Rezk-Salama and A. Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum*, 25(3):597–606, 2006.

[21] M. Sarkar and M. Brown. Graphical FishEye views of graphs. In *Proc. ACM CHI*, pages 83–91, 1992.

[22] M. SarkaR, S. S. Snibbe, Y. O. Tversk, and S. P. Reiss. Stretching the rubber sheet. In *Proc. ACM Symp. User Interface Software and Technology*, 1993.

[23] H.-W. Shen and D. L. Kao. A new line integral convolution algorithm for visualizing time-varying flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):98–108, 1998.

[24] V. Singh and D. Silver. Interactive volume manipulation with selective rendering for improved visualization. In *Proc. IEEE Symposium on Volume Visualization and Graphics '04*, pages 95–102, 2004.

[25] M. Tory and T. Moller. Rethinking visualization: A high-level taxonomy. In *Proc. of the IEEE InfoVis*, pages 151–158, 2004.

[26] S. M. F. Treavett and M. Chen. Pen-and-ink rendering in volume visualization. In *IEEE Visualization*, pages 203–210, 2000.

[27] I. Viola, M. Feixas, M. Sbert, and M. E. Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940, 2006.

[28] I. Viola, A. Kanitsar, and M. E. Groller. Importance-driven volume rendering. In *Proc. IEEE Visualization*, pages 139–146, 2004.

[29] L. Wang, Y. Zhao, K. Mueller, and A. E. Kaufman. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proc. IEEE Visualization*, pages 367–374, 2005.

[30] D. Weiskopf, K. Engel, and T. Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Trans. Vis. Comput. Graph.*, 9(3):298–312, 2003.

[31] S. Zhang, G. Kindlmann, and D. H. Laidlaw. Diffusion tensor MRI visualization. In *Visualization Handbook*. 2004.