# Using Deformations for Browsing Volumetric Data

Michael J. McGuffin* Liviu Tancau† Ravin Balakrishnan‡

Department of Computer Science, University of Toronto, http://www.dgp.toronto.edu

## Abstract

Many traditional techniques for "looking inside" volumetric data involve removing portions of the data, for example using various cutting tools, to reveal the interior. This allows the user to see hidden parts of the data, but has the disadvantage of removing potentially important surrounding contextual information. We explore an alternate strategy for browsing that uses deformations, where the user can cut into and open up, spread apart, or peel away parts of the volume in real time, making the interior visible while still retaining surrounding context. We consider various deformation strategies and present a number of interaction techniques based on different metaphors. Our designs pay special attention to the *semantic layers* that might compose a volume (e.g. the skin, muscle, bone in a scan of a human). Users can apply deformations to only selected layers, or apply a given deformation to a different degree to each layer, making browsing more flexible and facilitating the visualization of relationships between layers. Our interaction techniques are controlled with direct, "in place" manipulation, using pop-up menus and 3D widgets, to avoid the divided attention and awkwardness that would come with panels of traditional widgets. Initial user feedback indicates that our techniques are valuable, especially for showing portions of the data spatially situated in context with surrounding data.

**CR Categories:** I.3.6 [Computer Graphics]: Methodology and Techniques—interaction techniques; H.5.2 [Information Interfaces and Presentation]: User Interfaces—interaction styles

**Keywords:** volumetric data, volume data, deformations, browsing, layers, interaction techniques, 3D widgets

## 1 Introduction

Volumetric data can contain an enormous amount of densely packed data points, or *voxels*. Visualizing such data is challenging. While in the real world, we typically only perceive the surfaces of objects, computational visualization of volumetric data should ideally not have such a restriction. Where possible, we should be able to see the data *throughout* the volume simultaneously. However, this is especially difficult to achieve on a flat 2D display surface, where the user is, in some sense, forced to pick one point of view at a time, and where the number of voxels in the data can easily exceed the number of available pixels.

At least 3 general strategies exist for "peering inside" volumetric data:

1. Making some or all of the volume semi-transparent, allowing the user to see inside or through layers of data
2. Cutting away or removing portions of the data, to eliminate occlusion of inner regions
3. Spatially transforming or deforming the volume, to displace, project, break apart or separate outer portions and reveal inner portions

Strategies 1 and 2 have been widely used in volume visualization systems. The control over transparency in strategy 1 is often achieved by adjusting a *transfer function* which maps voxel values to colour, opacity, and other properties used in rendering. Strategy 2 includes all the common boolean masks used to "carve away" parts of a volume, such as cutting planes, cutting boxes, cutting spheres, etc., and more generally includes any technique that selects and displays a subset of the data, such as showing the voxels bounded by an isosurface.

As pointed out by Carpendale et al. [1997], although transparency and removal of outer data both make inner data more visible, they also result in loss of context. This can make it difficult for users to form an integrated mental picture of the entire volume.

Strategy 3 is based on deforming the data in some manner. We use the term "deformation" somewhat loosely here, to refer not only to smooth, non-rigid transformations, but also piece-wise rigid transformations, discontinuous transformations, and combinations of these. Thus, strategy 3 includes techniques such as "exploded views" (often used in assembly manuals for mechanical devices) that simply translate parts away from each other, as well as more exotic transformations, which, apart from some recent research [Carpendale et al. 1999; Carpendale et al. 1997; Kurzion and Yagel 1997; LaMar et al. 2001], have remained largely unexplored for the purpose of browsing volumetric data.

Our goal in using deformations is to increase the visibility of the inner portions of the volume, without completely removing the surrounding data that normally occludes the inside. This is akin to focus+context schemes that allow a user to "zoom in" on data of interest, while using remaining screen space to show the surrounding context. Appropriately chosen deformations could, for example, split open a volume, showing displaced structures side-by-side, making it easy for the user to see how they connect, and allowing the user to mentally stitch them together into a whole. Deformations with familiar real-world analogues (e.g. cutting and peeling the skin off a fruit, or the layers off an onion) are also likely to be readily understood by users.

In this paper, we describe a prototype system that implements different metaphors for deformation-based browsing of volumetric data. Since strategy 3 above seems to be the least explored, we have focused our research mainly on it, without, for example, implementing support for transparency. However, there is no reason that the techniques in this paper could not be profitably extended to work in conjunction with strategies 1 and 2.

As will be seen, a key element of our approach is to support differential treatment of the various *semantic layers* in a data set.

*mjmcguff@cs.toronto.edu
†liviu@pathcom.com
‡ravin@cs.toronto.edu

IEEE Visualization 2003,
October 19-24, 2003, Seattle, Washington, USA
0-7803-8120-3/03/$17.00 ©2003 IEEE

By "semantic layers", we mean subsets of the data that are useful or meaningful to the user. These layers could be defined geometrically, for example as sections created with parallel planar cuts. More typically, layers would depend on the voxel data values, for example boundaries that are found during segmentation or isosurface extraction. In the context of medical visualization, there is at least anecdotal evidence that anatomists, for example, prefer to remove tissue layer by layer [Höhne et al. 1992], rather than making arbitrary planar cuts.

In the following sections, we review related work, identify design issues and tradeoffs to consider when choosing a deformation, describe our prototype system, report some initial user feedback, and offer conclusions and thoughts on future directions.

## 2 Background

Many of our deformation techniques are inspired by surgical metaphors, where the user cuts into and opens up data. There have been attempts to create high-fidelity simulations of surgical procedures [Pflesser et al. 1995, for example] for education, training, and rehearsal. These often involve the use of virtual reality, haptic feedback, and the simulation of the physical properties, such as elasticity and hardness, of the tissues being operated on. Bruyns and Montgomery [2002] describe virtual tools that look and behave like scalpels, scissors, and forceps, allowing a 2- or 3-dimensional mesh to be cut and peeled open. While this approach has the advantage of easily understood, very literal, metaphors, it imposes an interaction style limited to what is possible in the physical world, without fully exploiting the additional capabilities of the computational medium. Our present work, in contrast, allows the user to explore and visualize volumetric data in ways that would be physically impossible. For example, in our system, users can peel away bone just as easily as skin, or change the location of an incision after the cut has been made by "swimming" the location of the cut through the volume. Because we are not concerned with simulating a physical process such as surgery, the user can browse data in a more light-weight and free-form style. We can also build intelligence into our browsing tools, so that they, for example, automatically detect boundaries between layers of data and do not cut across these boundaries.

Another difference between medical applications in general, and our work, is that medical specialists usually have a good idea of the underlying anatomy of a volume and thus can estimate where to look to find features of interest. In contrast, our techniques are designed to support general purpose exploratory browsing, and could be used with volumetric data of unknown content. This makes focus+context techniques all the more appropriate, since showing more of the data on the screen can make it easier and faster for users to find interesting data.

Looking beyond surgical applications, the deformation of volumetric data for general visualization has also been explored. For example, Laidlaw [1995] segmented scans of a banana and a human hand, and created animations of their skin peeling off. Our work, however, uses deformations for real-time, interactive browsing.

Kurzion and Yagel [1997] describe a "discontinuous ray deflector" that gives the appearance of cutting into a volume and spreading open the voxels. This is similar to the "book" metaphor used by Carpendale et al. [1999] where data are spread open like pages of a book. The same book metaphor has been used in traditional anatomical diagrams, where organs are shown cut in half and spread apart on consecutive pages of a book [Agur and Lee 1999, for example pp. 622–623, 718, 719]. In the next section, we describe our own Hinge Spreader tool which also uses this metaphor. Our work, however, also extends the existing repertoire of deformations with other tools.

Focus+context techniques have also been proposed for volumetric data. Carpendale et al. [1997] describe a *visual access distortion* technique that clears a path of visibility to a point of interest by pushing occluding data away from the line of sight. This "cleared path" remains on the line of sight as the scene is rotated, giving the appearance of a constantly shifting deformation. Thus, the rotation and deformation of the data are coupled. In our system, however, the user deforms the data in a desired way, and can then view the deformed data from any angle; i.e. rotation and deformation are separate actions. Although this introduces a risk that the user may have to rotate the scene more deliberately to gain a clear line of sight, the user also has more freedom and control to look at the deformed data in different ways.

LaMar et al. [2001] describe a focus+context technique that magnifies a region inside a volume. This magnified region is visible to the user if the surrounding data is semi-transparent, or if a cutting plane is used to reveal the inside. As will be seen in the next section, our own Sphere Expander tool may seem similar in that it expands regions of data. However, the Sphere Expander pushes voxels away from a central point rather than magnifying data. Thus, it can be used not only to enlarge an existing cavity or hole in the data, but also to create a hole where none previously existed. Furthermore, our Sphere Expander can be applied differentially to the layers in a data set, yielding new interaction possibilities.

## 3 Approach

At the outset of our research, we imagined three main actions that might be supported by a volume browsing system: (i) selecting a region of the volume, (ii) changing the appearance (i.e. transfer function, including opacity) of the selected region, and (iii) spatially transforming or deforming the selected region. Together, these three actions could be composed to achieve the same effect of potentially any existing browsing technique. Although we have not yet implemented the full vision of these three composable actions, we have explored issues surrounding region selection and deformation of a region.

Users commonly select regions of a volume using geometric primitives, such as halfspaces (planes), spheres, or boxes. The deformation tools in our system similarly have simple geometric shapes, and are constrained to act only on voxels within these shapes. However, the features of interest within a volume may have irregular shapes. Höhne et al.'s [1992] Anatomical Atlas supported "selective cutting" tools, that were sensitive to the layers in the data, and could be made to only act on certain layers. Thus, a cutting plane could be used to first remove skin, then bone, etc. The tools in our system are also sensitive to the layers in a data set, and can act differentially on them. Users can treat each layer separately, making selection of related voxels simpler and more implicit.

Unfortunately, the subsets of a volume data set are not always best thought of as *layers* — take for example the internal organs in a human body, or even vascular structures. Furthermore, volumetric data is often noisy and difficult to cleanly segment into distinctive subsets. Nevertheless, we chose to assume the existence of layers, and focus on how a user might manipulate such layers, because this inspired unique interaction techniques.

It is informative to compare the layers in a volumetric data set to a stack of cards or papers, and to consider the various ways in which these could be browsed. Mander et al. [1992] describe different ways of browsing virtual piles of documents, emulating the effect of riffling or thumbing through a physical pile of paper. Beaudouin-Lafon [2001] designed novel interaction techniques for overlapping windows. One of these allowed a user to peel back the corner of one or more windows, to take a peek at occluded windows. We identify a few other methods for manipulating layers in Figure 1.

In keeping with the layers-as-a-stack-of-cards analogy, there are three different points of view one might want of a given layer: first, a *dorsal* view of the back/top/outer surface of the layer; second,

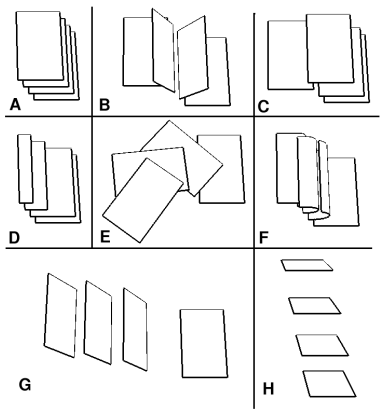


Figure 1: Techniques for browsing the layers depicted in A. B: *leafing* through the layers like pages of a book. C: pulling out an individual layer. D: compressing upper layers to reveal lower layers. E: *fanning* layers open like a hand of cards or like a Chinese fan. F: *peeling* layers back. G: *flipping* layers over — here the user is flipping the 3rd layer from the left, and the other layers to the left are pushed along like dominoes. H: an exploded view of the layers.

a *ventral* view of the underneath/bottom/inner surface of the layer (this is visible when the layer is flipped or turned over in some way); and third, a *cross-sectional* view, showing the thickness of the layer from the side, in which case it is often useful to see neighbouring layers stacked above and below the current layer.

Interactive browsing techniques that enable the various manipulations of Figure 1, and that also support dorsal, ventral, and cross-sectional views — possibly simultaneously — of one or more layers, are more likely to afford the user with flexible and useful vantages. We have tried to incorporate these elements in our designs.

Figure 1 already hints at some interesting ways in which layers could be deformed for visualization. Other deformations of interest can be inspired by surgical metaphors, whereby data might be cut into and spread open in different ways. Two questions to consider when choosing a deformation are: should the deformation be rigid or non-rigid, and what kind of continuity conditions should be satisfied by the deformation?

Rigid deformations, i.e. rotating and/or translating out a piece of a layer, have the advantage of preserving lengths and volume, which could be important for performing measurements, or simply for assurance that the data being visualized has not been distorted. On the other hand, non-rigid deformations, such as curvilinear "peeling", encompass a much broader range of possibilities, and may be more realistic in medical contexts for giving an impression, if only approximate, of how tissue would deform if it were physically peeled.

Regarding continuity, one issue is how a deformed region of data should remain "connected", if at all, with the rest of the volume. We return to this question in section 4.6.

Finally, there is a risk that deformations might sometimes render data unrecognizable, or change the spatial arrangement of voxels in ways that are unfamiliar or difficult to understand. To counter this, we use smooth animations to show changes or transitions in the shape of the data. For example, if the user invokes a tool that peels back a layer, rather than suddenly "snapping" the layer into a fully peeled state, the layer is continuously peeled in real time, to show the user what is happening. The benefits of using animation for smooth transitions have been documented by others [Bartram 1997; Grossman et al. 2001; Robertson et al. 1991; Woods 1984]. Essentially, users more easily maintain a mental model of the data across transitions, spending less time assimilating new states.

# 4 Prototype Implementation

Our prototype volume browser was implemented in C++ using OpenGL and GLUT, and runs under Linux and Microsoft Windows.

The individual voxels of the data are rendered as points (i.e. GL_POINTS) whose size in screen space is chosen (via glPointSize()) to give the appearance that adjacent voxels are just touching. Although many techniques exist for high quality volume rendering, for example using hardware texturing and trilinearly interpolating voxel values, we chose to render individual points to keep our prototype simple and maximally flexible. Any deformation that remaps the voxel locations can be supported by our system, since each voxel is rendered on its own. This gives us the freedom to focus on exploring interaction techniques, rather than optimized, high quality rendering.

There is currently no support in our system for transparency. Although we suspect our techniques could be enhanced with good use of transparency, we wanted to first isolate and identify the qualities and issues that are unique to deformations.

On a 1.7 GHz laptop with an *nVidia GeForce4 Go* graphics card, 32 MB of video memory, and 512 megabytes of RAM, our system can render over 500000 voxels at 13 full screen frames per second, or over 4000000 voxels at 2 full screen frames per second. Since real time interaction is critical, our system downsamples large data sets and renders them at a lower resolution during interaction. Full resolution rendering is performed after the system has been idle for a given timeout (e.g. one second), or whenever the user explicitly requests it. An even better implementation might render different parts of the volume at different resolutions. For example, only the portion of the volume currently in the user's focus could be rendered at full resolution, without precluding real-time interaction.

To support arbitrary transformations of voxel positions, we explicitly store the position of each voxel, rather than storing a 3D bitmap. This is also more efficient for sparse data sets. Voxel positions are stored in an octree, where each node of the octree has a bounding box and a colour (black, white, or both). A dividing plane can be applied to the octree, and voxels can be quickly categorized by colouring them black or white, according to the side of the plane they lie on. Intersections or unions of halfspaces can be coloured by applying multiple planes. Operations on the octree, such as rendering voxels, deforming voxel positions, or copying voxels into a second octree, can be applied to the whole octree, or to only a subset of a given colour.

As a minor optimization, each voxel position is not stored in its own leaf node. Instead, each leaf node stores a small number (e.g. 8) of voxels in an array that can be traversed more quickly than an equivalent subtree with one voxel per leaf. Rather than storing a colour flag for each voxel, we save memory by storing black voxels in the first $n$ elements of the array, and white voxels in the remaining elements. Changing the colour of a voxel requires swapping a single pair of elements and adjusting the value of $n$.

To support operations that treat each layer of data differently, each layer of voxels is stored in a separate octree. Thus, having $N$ layers requires $N$ octrees. This does not, however, imply using $N$ times more memory than a single octree for all the layers would. Each layer typically exhibits some spatial coherence, and can be stored efficiently in an octree.

Each voxel has an associated normal, and is rendered with lighting to provide the user with shading cues. Voxels near a surface of the data set have a normal computed from their neighbourhood — this computation is slow, but need only be done once, at load time. Voxels in the interior of the volume are initially not visible, and have a zero normal. However, deformations can cut or split open the volume and reveal these interior voxels. Thus, the normals of interior voxels are dynamically recomputed, in a fast but approximate way, based on the current deformation, and based on a guess

of the orientation of the closest surface. Although the result is only approximate, the depth cues resulting from lighting the scene were found to be preferable over having no lighting.

The browsing tools in our system are positioned, oriented, and resized using 3D widgets [Conner et al. 1992], i.e. objects embedded in the 3D scene that can be clicked and dragged. 3D widgets are also used to control the parameters of the different deformations. Each draggable component of our 3D widgets is highlighted when the mouse cursor passes over them, to hint to the user which elements can be dragged. The shape of the widgets also suggests how to use them: arrow widgets are for translation or adjustment of a linear quantity such as the radius of a sphere; circle and arc widgets are for rotation or adjusting angles. In an early version of our prototype, we noticed perceptual problems with the visual design of our 3D widgets. We thus improved them by adding more depth cues (Figure 2).

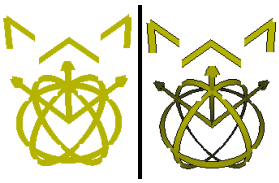

Figure 2: Example 3D widgets before and after design changes that enhanced depth cues. On the right, shading and variation in thickness are used, and intentionally exaggerated, to suggest depth. A thin halo of black pixels is also drawn to ensure contrast with whatever data may be in the background. The arrow widgets are used for translation along an axis, the circle widgets are used for rotation around an axis, and the 'L'-shaped widgets are for translation within a plane.

Another challenge encountered was that some deformations have many adjustable parameters, and if each of these is controlled with a separate 3D widget that is always visible, the screen becomes cluttered with widgets. We therefore identified situations where certain widgets were not likely to be used, and changed our prototype to only display a given 3D widget if the current state warrants it. For example, Figure 10 shows a set of layers that are fanned open. Only after they are fanned open do additional 3D widgets appear, attached to each layer.

Figures 3 through 14 show our system browsing a scan of a human head that was pre-segmented into 5 layers. At any given time, only one browsing tool is active, which the user chooses from a popup radial menu [Callahan et al. 1988] or Marking Menu [Kurtenbach and Buxton 1993] (our menu is only one level deep, and so could be described as either a radial menu or Marking Menu). The active tool only affects the currently selected layers of the data, leaving unselected layers unchanged. When a layer is selected or unselected, an animation shows the layer's transition from one state to the other, for example, from a deformed state to an undeformed state.

Two mechanisms were implemented for selecting/unselecting layers. First, the selection state of each layer can be toggled individually through a set of hotkeys assigned to each layer — but these could just as easily be virtual check boxes in a menu. Second, there are two special items in the popup radial menu, the right and left items, that also control layer selection. These items can be invoked with quick flick gestures to the right or left, and have the effect of (a) selecting the outermost unselected layer, or (b) unselecting the outermost selected layer, respectively. Thus, the user can, for example, make 5 flick gestures to the right, causing each layer, from the outermost to the innermost, to be successively selected. If the currently selected tool peels layers away, this would have the effect of successively peeling each layer in the natural ordering.

To support this behaviour, the system needs some notion of which layers are inside or outside other layers. We manually assigned a global, fixed ordering, from outside to inside, of the layers in our head data set. This ordering is important not just for selection, but also for deformations that automatically deform layers to different degrees, such as the fanning out in Figure 10, where inner layers are rotated by a larger angle than outer layers. Although the fixed ordering is acceptable for our head data set, in general this would not be a viable solution. It is possible for the semantic layers in a data set to not have any single ordering from outside to inside. An improved prototype would compute a locally acceptable ordering of layers on the fly, given the current location and orientation of the deformation tool. Such an ordering might be computed by sampling the relevant data along parallel rays, and finding the "average ordering" of layers encountered along the rays.

No collision detection is performed between layers. Hence, layers can interpenetrate as they are manipulated or animated. However, as long as the user is manipulating a region of data where the the inside-to-outside ordering of layers is reasonably accurate, the interpenetration of layers is minimal.

The deformation tools in our system will be described in the following subsections. We also implemented more traditional cutting tools, specifically: a cutting plane, cutting hinge, cutting sphere, and cutting box. Figure 3 shows two of these in action. Just as in Höhne et al.'s Atlas [Höhne et al. 1992], our cutting tools are sensitive to the layers in the data, and only remove voxels from the currently selected layers. Thus, they can be thought of as intelligent scalpels that cut no deeper than the innermost selected layer. An alternative way of thinking of these tools, especially the cutting box, is that they behave like 3D magic lenses [Viega et al. 1996], in that they make the currently selected layers fully transparent.

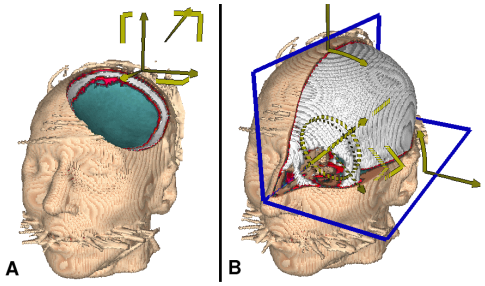


Figure 3: Examples of cutting tools. A: a cutting sphere. B: a cutting hinge. Each cutting tool can be made to cut away all layers (as in A) or only a subset of layers (as in B).

### 4.1 Hinge Spreader Tool

The Hinge Spreader (Figure 4) tool is a dihedral shaped object that pushes all the voxels between the hinge to either side. As mentioned in the Background section, this deformation can be used to create views of data that resemble anatomical dissections spread open like a "book" [Agur and Lee 1999].

3D widgets enable positioning and orientation of the tool, and also allow the angle of the hinge to be adjusted. Note that the cutting hinge in Figure 3 B has the same dihedral shape as the Hinge Spreader, but *removes* the voxels within the hinge, rather than displacing them.

As with all our tools, the Hinge Spreader only deforms voxels of selected layers. When layers are selected or unselected, an animation shows the voxels of that layer transition from a deformed state to a resting state, or vice versa. Interestingly, the Hinge Spreader can be used to create views that look like exploded diagrams (Figure 5) when applied to only a subset of the layers.

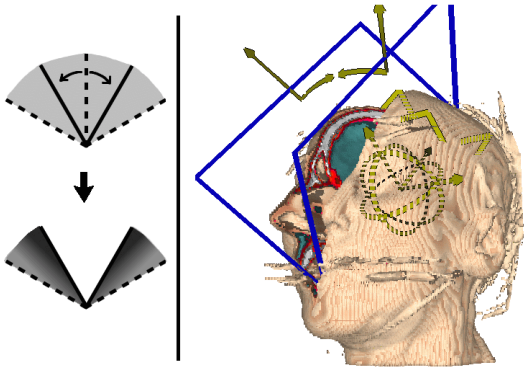

Figure 4: The Hinge Spreader. Left: a sketch of the voxels before and after deformation. Solid black lines show the hinge seen by the user. Dashed lines delimit the voxels affected by the deformation. Voxels are pushed away from the bisector of the hinge, compressing surrounding voxels that are within twice the hinge's angle. Right: a face is split down a line through the nose. Note that both halves of the nose are still present – no voxels have been removed or cut away, they have simply been pushed aside.

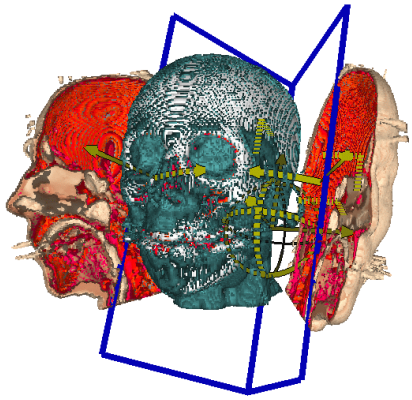

Figure 5: The Hinge Spreader, acting only on layers outside the skull, has been pushed all the way through the head, and therefore spreads both halves of the skin off the skull. This provides a kind of "exploded view".

The hinge form factor of this tool, and of the cutting hinge, have interesting properties with respect to interface design. First, the positioning, orientation, and angle of a hinge could be easily controlled with a hand-held prop, much like Hinckley et al.'s [1998] use of a cutting plane prop in their "doll's head" interface. Users have a strong mental model of the shape and function of a hinge, and would probably be able to use a hinge prop as successfully as Hinckley et al.'s cutting plane. In addition, the use of a hinge has certain advantages over a plane. A hinge opened up to 180 degrees reduces to a plane as a special case, and so is more general than a plane. Acute hinge angles allow for more context to be maintained close to a focal point. Finally, two-handed techniques are possible where a user holds two hinge props, and could make fast, compound cuts or spreads of the data.

### 4.2 Sphere Expander Tool

The Sphere Expander tool (Figure 6) pushes voxels away from a central point. The centre and radius of the sphere are controlled with 3D arrow widgets. Placing this tool outside and near the surface of a volume creates a dent in the volume, which in itself may not be useful for browsing. However, when placed inside a volume, the Sphere Expander can be used to inflate the voxels of the volume outward. Since we render each voxel as a point, sufficient inflation eventually makes the voxels sparse enough to see through — a kind of cheap transparency. The Sphere Expander can also be used to create a hole in a layer, by selecting only that layer, and by placing the centre of the sphere on the layer (Figure 6, right hand side). Interestingly, we did not initially know whether the Sphere Expander tool would turn out to be useful. After implementing it, however, we discovered that our layer-based architecture allowed for situations like that in Figure 6.

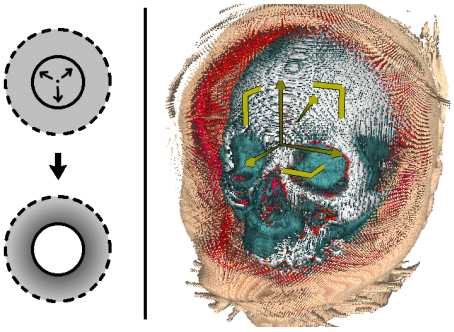

Figure 6: The Sphere Expander. Left: voxels before and after deformation. All voxels contained in the solid sphere are pushed outside, compressing surrounding voxels that are within twice the sphere's radius. Right: the Sphere Expander, acting only on layers outside the skull, is centred on a point on the face above the nose. This opens up a hole in the face and lifts much of the skin off the skull, creating a kind of "window" through which we can see the skull and surrounding skin.

### 4.3 Box Spreader Tool

In the same spirit as the previous two deformation tools, the Box Spreader (Figure 7) pushes all voxels outside the shape of the tool, in this case a box. This tool was inspired by *rib spreaders*, instruments used in chest cavity surgery. The Box Spreader could be used to cut in to the virtual chest of a human data set, and spread open the outer layers of the chest, revealing internal organs. When the box is made wide enough, however, the deformation can eventually lift outer layers off a data set, as in Figure 7.

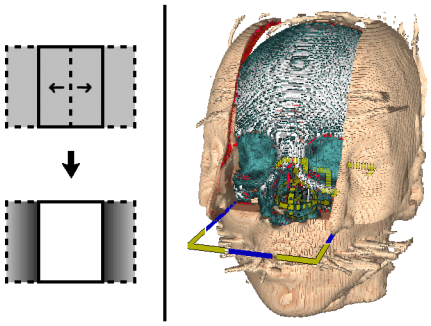

Figure 7: The Box Spreader. Left: voxels before and after deformation. All voxels contained in the box are pushed sideways, compressing surrounding voxels that are within twice the box's width. Right: the Box Spreader, acting only on layers outside the skull, cuts the upper face in half and pushes each half off the skull.

### 4.4 Leafer Tool

The Leafer is shaped like a tray (Figure 8). The voxels above each half of the tray can be hinged open using 3D widgets. Selecting or unselecting layers causes them to smoothly rotate out or back in to place. A rapid succession of selections initiates an animation with layers temporarily spaced out, or "leaved" (Figure 9), affording the user a brief glimpse of the shape of individual layers. This style of browsing inspired the name of the Leafer tool. Note that once the animation is complete, however, all selected layers are rotated open with the same angle.

The three views of layers mentioned in Section 3 are all made available, simultaneously, with the Leafer. Figures 8 shows the areas where layers are seen from a dorsal, ventral, or cross-sectional view. Selection or unselection of a single layer causes that layer to transition from one view to the other.

After hinging open the halves of the Leafer's tray, the layers that make up each half can be fanned open (Figure 10). Fanned out layers can then be pulled out and/or flipped over, showing the components of the "dissected" voxels in context with the rest of the data set.

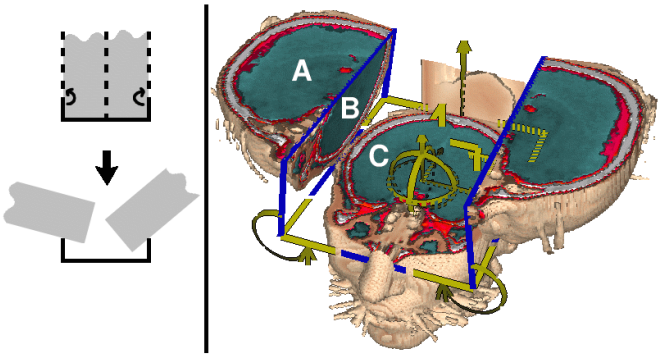


Figure 8: The Leafer. Left: voxels before and after deformation. Voxels above each half of a "tray" are (rigidly) rotated away from the centre of the tray. The top edges of the tray are the axes of rotation. Right: the Leafer is used to hinge open parts of a head. Here, the depth of the tray has been set to zero, i.e. the axes of rotation coincide with the bottom edges of the tray, whereas in the figure on the left, the depth of the tray is non-zero. Three areas are labelled A, B, and C, to show how the Leafer provides cross-sectional, ventral, and dorsal views of layers, simultaneously.

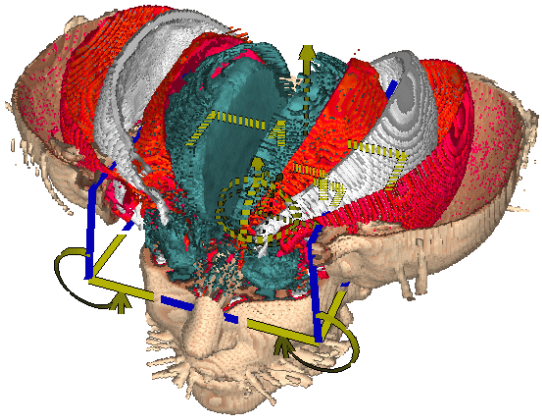

Figure 9: When the user selects/unselects a layer, the Leafer animates the rotation of the layer open or back in to place. Here, the user has selected each layer in rapid succession, and the leafer is midway through an animation opening them up. In this way, the user can "leaf" through the layers, as if they were pages of a book.

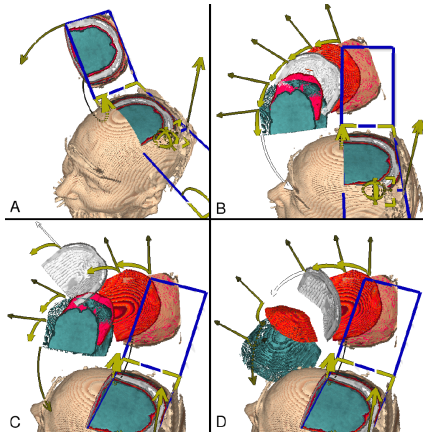


Figure 10: A: The left half of the leafer hinges open the top of the head. B: The layers that make up the hinged-open region are fanned open. New widgets appear attached to each layer. C: A translation widget is used to pull out an individual layer. D: Rotation widgets are used to flip over layers.

### 4.5 Peeler Tool

The Peeler, like the Leafer, consists of a tray that can be positioned and oriented to encompass a region of interest, and allows each half of the tray to be opened up. Unlike the Leafer, however, the Peeler uses a non-rigid, curvilinear deformation to open up the layers (Figures 11 and 12).

As with the Leafer, selecting layers when the Peeler is active initiates an animated transition, during which the user can see in between the moving layers. However, unlike the Leafer, the Peeler also affords control over the degree of peeling for each layer independently, through arrow widgets attached to each layer. This gives the user an extra level of control, allowing the user to create spaces between the layers (Figure 13) and keep them in this state for further browsing.

We also created a variation on the Peeler called the Radial Peeler (Figure 14). Each voxel is peeled radially away from the axis of the tool, as if the tool were poking a hole in the volume and turning the layers inside out, somewhat like a flower opening up.

Figure 12 sketches the deformation for the left half of the regular Peeler, where voxels are peeled to the left. If this sketch is revolved around the vertical axis $x = \rho$, where $\rho$ is the radius of the Radial Peeler, the resulting form corresponds to how the Radial Peeler deforms voxels: they are peeled away from the axis $x = \rho$.

### 4.6 Observations

Similarities can be seen between our set of tools and the layer browsing techniques of Figure 1. The Hinge Spreader, Sphere Expander, and Box Spreader all non-rigidly compress and push layers to reveal data, just like Figure 1 D. The Leafer combines the techniques of Figure 1 B, E, C and G. And the Peeler, of course, corresponds to Figure 1 F. Our tools are not the only possible combinations of techniques, and extensions are possible (such as the exploded view of Figure 1 H), but we have demonstrated the applicability of layer-based techniques for browsing volumetric data.

An interesting tradeoff to consider is how much control to give the user. The Leafer allows users to "leaf" through layers (Figure 9)

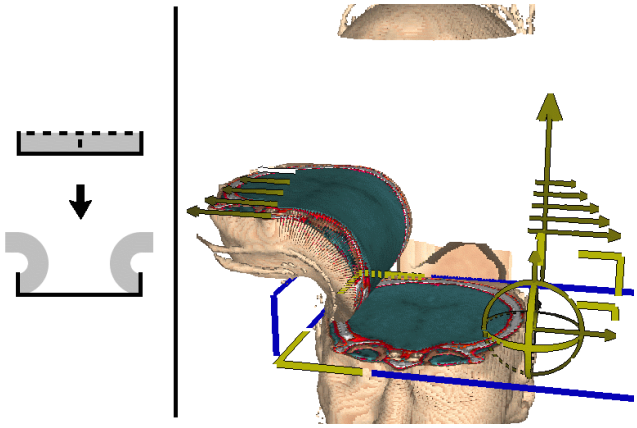

Figure 11: The Peeler. Left: voxels before and after deformation. Voxels in each half of a "tray" are peeled off the bottom of the tray. Right: the left half of the Peeler's tray is positioned to encompass the brow of the head, and this is peeled off. Notice that, above the Peeler, a thin section of the top of the head has been automatically translated upward and out of the way.

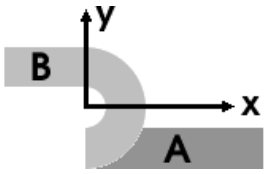


Figure 12: The deformation on the left half of the Peeler's tray. Each point $(x,y) \in A$ is mapped to a new point $(x',y') \in B$. Let $R$ be the radius at which length is preserved by the deformation. Points where $x < R\pi$ are mapped to the curved region via $(x',y') = (-y\sin(x/R), y\cos(x/R))$. Other points are shifted and rotated with $(x',y') = (-(x-R\pi), -y)$.

using animation. This shows how animation is useful not only for helping the user maintain their mental model of the data as it deforms, but can be a browsing technique in itself. In contrast, the Peeler also allows the user to individually peel each layer by different amounts (Figure 13), giving the user more control, but this comes at the cost of more 3D widgets that clutter the screen. In general, we reduced clutter from widgets by only displaying them when the state of the deformation warrants their presence. However, additional techniques for reducing the number of widgets shown at any time, without limiting the user's power, would be valuable. One possibility is to develop a kind of popup 3D widget, that is shown only when requested by the user.

The Leafer and Peeler also shed some light on the question of how to connect a deformed set of voxels to the rest of the volume. The Leafer rigidly deforms voxels, creating a sharp "seam" at the axis of rotation. By hinging open the Leafer far enough, the user can easily see this seam where the voxels connect. However, such rigid rotation can lead to interpenetration of the deformed voxels and the rest of the volume. The Peeler, on the other hand, is more continuous in the sense that the deformed voxels connect smoothly with the rest of the volume. Interpenetration of voxels is less likely, and reduced in severity if it does occur. However, in the case of the Peeler, the seam along which peeled regions connect with the volume is much harder to see, since it is usually occluded by the peeled layers. It may or may not be important for the user to be able to see these seams or contact edges, however, it is a tradeoff to consider when choosing a deformation.

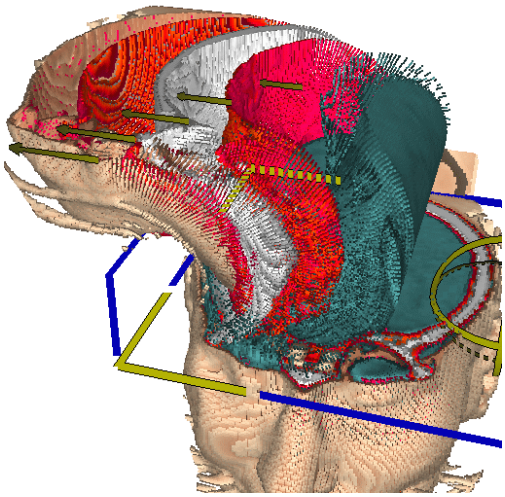

Figure 13: A close up of the peeler in action. Here, each layer has been peeled to a different degree (using the small arrow widgets attached to each layer). The spacing between layers makes the interfaces between them visible.

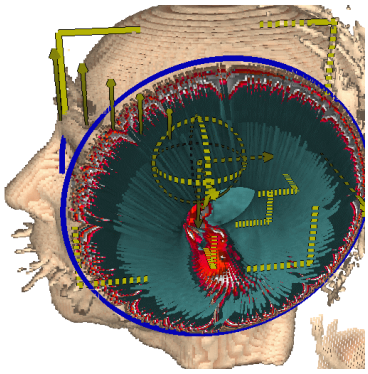

Figure 14: The Radial Peeler. Rather than peeling away two halves of a tray as in Figure 11, this tool peels away all the voxels in a cylinder. Two circles (only one of which is visible here) delimit the cylinder. Voxels are pulled through the top of the cylinder and then stretched away from the cylinder's centre. A hole along the cylinder's axis is thus opened up, allowing the user to peer inside.

## 5 Initial User Feedback

Informal trials with members of our lab led to some improvements in the visual design of the tools and 3D widgets. Furthermore, an earlier prototype of our system didn't employ animations, i.e. deformations would cause the volume to suddenly "snap" to a transformed state. During demonstrations of this prototype, people often had trouble understanding how exactly the tools were deforming voxels. Hence our incorporation of smoothly animated transitions.

More recently, we had a professional anatomist, having little experience using 3D software, try out our system during an informal, one hour session. The session consisted of a mix of designer-driven demonstration and user-driven exploration of the tools.

The anatomist found that the direct manipulation 3D widgets afforded flexible control and were easy to understand. Deforming or pulling out portions of the volume *in context*, with the rest of the volume still displayed, was found to be very valuable, for keeping track of "where you are in the whole". Animated transitions were also found to be valuable. The anatomist suggested that they would

be very appropriate in educational settings, e.g. to show layers actually peeling back, rather than showing a sudden change of state.

The anatomist also suggested that, in some situations, after a layer has been peeled away, it may not be important to continue displaying the layer, since the user's goal may be to simply see the tissues revealed underneath. However, there were other situations were the anatomist found it important to keep all data present, for example when showing the two halves of the Hinge Spreader.

## 6 Conclusion and Future Directions

We have extended the range of deformations used for exploratory browsing of volumetric data. Our prototype demonstrates one way of integrating these deformations with differential treatment of the layers in a data set, as well as with 3D widgets and use of animation. We have identified various tradeoffs and design issues brought to light by our work. Initial user feedback suggests that our techniques are useful for helping a user understand and maintain context while exploring different regions of a data set.

One aspect not explored in our prototype is enhancing tool behaviour with transparency. For example, a hinge-shaped tool might make voxels enclosed by the hinge gradually more transparent as the hinge is opened up to a wider angle. The Leafer and Peeler tools could also make affected layers partially transparent, reducing occlusion of the immediate neighbourhood of these layers.

Our Leafer tool combined many techniques of Figure 1 in one particular order, but many other orderings or combinations are possible. More flexible tools could be designed, perhaps allowing the user to "construct" their own custom deformations by combining more primitive operations or widgets.

Sophisticated deformations with many parameters can clutter the screen with 3D widgets. Ideally, a widget should only be visible when the user wants to interact with it. Popup 3D widgets, or use of gestures instead of widgets, could eliminate this problem.

Finally, the implementation of true volume rendering, possibly using graphics hardware [Rezk-Salama et al. 2001], could improve visual quality and frame rates.

## 7 Acknowledgements

Thanks to Karan Singh, Sheelagh Carpendale, Anne Agur, Gord Kurtenbach, Rafeef Abugharbieh, Nicholas Woolridge, Lloyd Burchill, Kim Chua, Joe Laszlo, Gonzalo Ramos, Glenn Tsang, and the members of the Interaction Research Group at the University of Toronto, for their help throughout this work.

## 8 Supporting Material

Videos and additional screen shots of our prototype can be downloaded at http://www.dgp.toronto.edu/~mjmcguff/research/

## References

AGUR, A. M. R., AND LEE, M. J., Eds. 1999. *Grant's Atlas of Anatomy*, 10th ed. Lippincott, Williams and Wilkins.

BARTRAM, L. 1997. Can motion increase user interface bandwidth? In *Proceedings of IEEE Conference on Systems, Man and Cybernetics*, 1686–1692.

BEAUDOUIN-LAFON, M. 2001. Novel interaction techniques for overlapping windows. In *Proceedings of ACM UIST Symposium on User Interface Software and Technology*, 153–154.

BRUYNS, C. D., AND MONTGOMERY, K. 2002. Generalized interactions using virtual tools within the Spring framework: Cutting. In *MMVR Medicine Meets Virtual Reality*.

CALLAHAN, J., HOPKINS, D., WEISER, M., AND SHNEIDERMAN, B. 1988. A comparative analysis of pie menu performance. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*.

CARPENDALE, M. S. T., COWPERTHWAITE, D. J., AND FRACCHIA, F. D. 1997. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications: Special Issue on Information Visualization 17*, 4, 42–51.

CARPENDALE, M. S. T., COWPERTHWAITE, D. J., TIGGES, M., FALL, A., AND FRACCHIA, F. D. 1999. The Tardis: A visual exploration environment for landscape dynamics. In *Proceedings of SPIE Conference on Visual Data Exploration and Analysis VI*.

CONNER, D. B., SNIBBE, S. S., HERNDON, K. P., ROBBINS, D. C., ZELEZNIK, R. C., AND VAN DAM, A. 1992. Three-dimensional widgets. In *Proceedings of ACM I3D Symposium on Interactive 3D Graphics*, 183–188.

GROSSMAN, T., BALAKRISHNAN, R., KURTENBACH, G., FITZMAURICE, G., KHAN, A., AND BUXTON, W. 2001. Interaction techniques for 3D modeling on large displays. In *Proceedings of ACM I3D Symposium on Interactive 3D Graphics*, 17–23.

HINCKLEY, K., PAUSCH, R., PROFFITT, D., AND KASSELL, N. F. 1998. Two-handed virtual manipulation. *ACM TOCHI Transactions on Computer-Human Interaction 5*, 3 (September), 260–302.

HÖHNE, K. H., BOMANS, M., RIEMER, M., SCHUBERT, R., TIEDE, U., AND LIERSE, W. 1992. A volume-based anatomical atlas. *IEEE Computer Graphics and Applications 12*, 4 (July), 72–78.

KURTENBACH, G., AND BUXTON, W. 1993. The limits of expert performance using hierarchical marking menus. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 35–42.

KURZION, Y., AND YAGEL, R. 1997. Interactive space deformation with hardware-assisted rendering. *IEEE Computer Graphics and Applications 17*, 5.

LAIDLAW, D. H. 1995. *Geometric Model Extraction from Magnetic Resonance Volume Data*. PhD thesis, California Institute of Technology.

LAMAR, E., HAMANN, B., AND JOY, K. I. 2001. A magnification lens for interactive volume visualization. In *IEEE Pacific Conference on Computer Graphics and Applications*, 223–232.

MANDER, R., SALOMON, G., AND WONG, Y. Y. 1992. A 'pile' metaphor for supporting casual organization of information. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 627–634.

PFLESSER, B., TIEDE, U., AND HÖHNE, K. H. 1995. Towards realistic visualization for surgery rehearsal. In *Computer Vision, Virtual Reality and Robotics in Medicine, Proc. CVRMed '95 (N. Ayache, ed.), vol. 905 of Lecture Notes in Computer Science*, 487–491.

REZK-SALAMA, C., SCHEUERING, M., SOZA, G., AND GREINER, G. 2001. Fast volumetric deformation on general purpose hardware. In *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 17–24.

ROBERTSON, G. G., MACKINLAY, J. D., AND CARD, S. K. 1991. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 189–194.

VIEGA, J., CONWAY, M. J., WILLIAMS, G., AND PAUSCH, R. 1996. 3D Magic Lenses. In *Proceedings of ACM UIST Symposium on User Interface Software and Technology*, 51–58.

WOODS, D. D. 1984. Visual momentum: a concept to improve the cognitive coupling of person and computer. *International Journal of Man-Machine Studies 21*, 229–244.