HingeSlicer: Interactive Exploration of Volume Images Using Extended 3D Slice Plane Widgets

Tim McInerney
tmcinern@scs.ryerson.ca

Sara Broughton
s3brough@scs.ryerson.ca

Dept. of Computer Science
Ryerson University, Toronto, ON

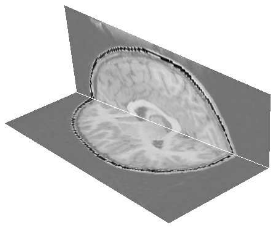
Figure 1: Hinged slice plane widget used to examine an MR volume image of the brain.

Abstract

We present a 3D interaction model for exploring volume image data by extending the capabilities of 3D slice plane widgets. Our model provides the ability to navigate through a volume image in a fast, intuitive manner, using object-relative user navigation. Employing a cut-fold-slide analogy, 3D slice plane widgets are rotated and translated relative to each other. The planes can be progressively cut to extend existing views and form staircase-like arrangements, minimizing occlusion and visual clutter problems that result from multiple, disconnected slice planes. Extending existing views also allows cutting actions to be easily "mended", providing users with the ability to return to a previous "good" view and explore again. A user makes cuts by drawing "hinge" lines on a slice plane widget, in any orientation, dividing the slice plane into two pieces. These pieces can fold (rotate) around the hinge line or slide (translate) with respect to each other, allowing the user to retain a better contextual understanding of the 3D spatial relationships between structures and of 3D structure shape.

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

Keywords: Three dimensional interaction techniques, medical visualization, scientific visualization.

1 Introduction

Medical volume images, such as CT or MR scans, are commonly viewed and processed using 3D slice plane widgets. While true 3D views generated with volume rendering or iso-surface extraction provide 3D shape information, it is often very difficult to find parameter settings that generate a clear 3D view of the target anatomical structure(s) buried within the volume, especially for noisy images. Slice plane widgets on the other hand, which provide 2D cross-sectional views, are fast and simple to generate and manipulate, enable precise volume navigation, and allow analysis and measurement of object details and object boundaries.

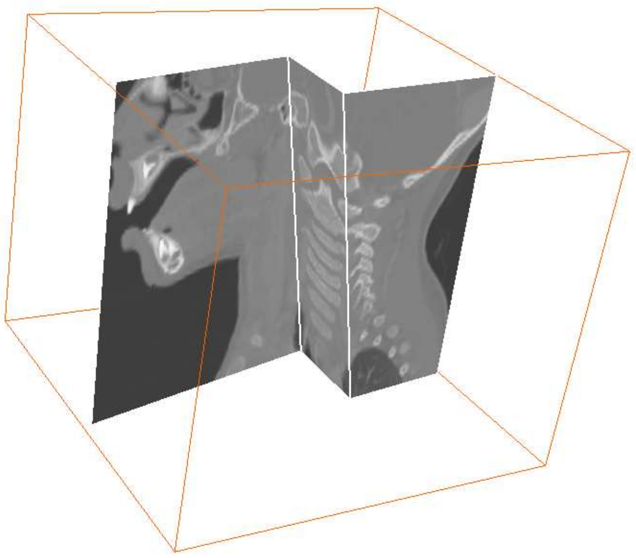
Figure 2: Interactively exploring 3D volumetric data sets by cutting, folding, and sliding a 3D slice plane widget.

Despite these properties, 3D image slice planes widgets suffer from several well-known problems. Orthogonal slice planes in standard orientations (axial, coronal, sagittal, i.e. x-y, x-z, y-z planes, respectively) are not object-aligned and it is difficult for even expert users such as radiologists to mentally reconstruct 3D object shape from these standard 2D projections. While oriented slice planes provide more view control, rotating and translating a single slice plane in volume image coordinates is difficult and tedious - it is easy to become "lost" within the volume since no contextual information is retained during the manipulation. Anatomical structures bend and twist away from slice planes so that a single view is insufficient to allow users to retain an overall understanding of the 3D spatial relationships between structures, and of 3D structure shape. On the other hand, multiple disconnected, independently-controlled slice plane widgets quickly create visual clutter and occlusion.

It is well known that true 3D views of anatomical structures are needed to gain a qualitative understanding of 3D spatial data for many analysis and surgical planning tasks. Volume rendering and iso-surface extraction are adequate for creating 3D views of some structures whose voxels appear as distinct, homogeneous regions in a volume image. However, it is often necessary to use sophisticated segmentation algorithms to extract, reconstruct, and render a geometric surface representation of the structures. This segmentation step is also essential for quantitative analysis. One of our primary motivations for extending the capabilities of 3D slice planes is to provide better monitoring, steering, and editing capabilities in order to improve the the efficiency, accuracy, and repeatability of 3D interactive (i.e. semi-automatic) segmentation algorithms. Over the past several years, many surface model-based segmentation techniques [11, 2, 5] have been developed. One of the problems with these and other interactive 3D segmentation methods is that once the algorithm is initiated, the ability to steer it is limited, as complex 3D interaction issues arise. For example, how does one display the data and the model together such that the user can view the progress and success/failure of the segmentation, and then steer the model back on course? These 3D interactive control and display issues result in an often time-consuming and tedious manual editing phase of the segmentation - a key factor for overburdened radiologists.

In this paper we present a tool to extend the capabilities of 3D slice plane widgets. We provide the ability to navigate through a volume image in a fast, intuitive manner, using anatomical structure-specific widget instantiation. In addition, research has shown that manipulating objects relative to each other is easier than using absolute coordinates [3]. Consequently, we use an interactive cut-fold-slide analogy (Figure 2) to realize relative slice plane widget manipulations and extended views . The planes can be progressively cut to form a staircase-like arrangement, minimizing occlusion and visual clutter. Relative widget manipulation and view extension help to retain better overall contextual understanding of the shape and spatial arrangement of the anatomical structures within the volume. In addition, cuts can be "mended" so that the user can return to a "good" previous view and explore again. In this way users are able to quickly cut and expose the interior of the volume image, exploring (and optionally processing) object-aligned cross-sections, before returning and repeating with a new cut. A user makes cuts by drawing "hinge" lines on a slice plane, in any orientation, dividing the slice plane into two pieces. The user folds (rotates) one piece with respect to the other along this hinge line. Alternatively, a piece may slide (translate) along the hinge line in the slice plane normal direction. This sliding action automatically creates an orthogonal connecting slice plane between the two pieces that grows as the pieces are pushed away from each other, and shrinks as they are brought closer together. This allows a user to interactively explore cross sections anywhere along an anatomical structure by drawing the hinge lines perpendicular or parallel to its major medial axes.

2 Background

Most medical image visualization packages [6, 7] provide 3D slice plane widgets for generating and manipulating multiple arbitrarily-oriented cross-sectional views. These image plane widgets are typically disconnected from each other and are positioned (rotated and translated) independently within the volume image. In addition, at least one commercial package, Slicer Dicer from VisuaLogic [9], allows sub-volumes of a volume image to be cut out by interactively expanding a rectangular region on an axial, coronal, or sagittal slice plane.

Weiskopf et al. [10] describe volume clipping - an interactive technique for cutting away selected parts of 3D volumetric data sets, taking advantage of 3D texture mapping abilities of modern graphics hardware. Their technique is capable of constructing complex 3D clip geometries, including geometries with curved boundaries. While this approach is very general, it may be difficult to interactively explore a region of the volume image around a particular anatomical structure using absolute positioning and the sculpting model. Parts of the object may be inadvertently cut away and each cutting action is independent of the others. Furthermore, unlike the HingeSlicer, cuts are performed in volume image coordinates rather than relative to an existing clip geometry.

McGuffin et al. [4] attempt to overcome the problems inherent with cutting tools that reveal the interior of volumetric data sets but remove potentially important surrounding contextual information. Their alternate strategy for volumetric data browsing uses deformations, where the user can cut into and open up, spread apart, or peel away parts of the volume in real time, attempting to retain surrounding context. However, their technique, while visually compelling, is fundamentally geared toward displaying and exploring pre-segmented volumes, where voxels have been classified into tissue type (such as in anatomical atlases). Users can apply deformations to selected semantic layers (e.g. the skin, muscle, bone) facilitating the visualization of relationships between layers. In addition, the 3D widgets they describe perform a single specific deformation and cannot be extended or compounded to allow a more detailed exploration around or through a specific object.

Carpendale et al. [1] develop a browsing method to explore 2D geospatial data that changes over time. The information is presented as a 3D spatio-temporal block (with two spatial dimensions and one temporal dimension) by stacking sequential temporal layers one atop the other. A variety of representations and interactions with the spatially arranged temporal data are possible, including orthogonal spatial-temporal slice planes. In addition, to examine the changes from one time unit to the next, the authors have developed a browsing method that allows one to move through the spatio-temporal cube in a similar manner to which one would turn pages in a book. The block opens, displaying two adjacent temporal layers.

Several researchers have developed visualization techniques to display 2D views alongside or within 3D views. Typically the 3D view (e.g. the surface of an enclosing object) is limited to an outline or semi-transparent surface to cut down on occlusion and clutter while still providing contextual information. For example, Tory et al. [8] present the 3D view in the center of the display, with 2D views surrounding it in close proximity. The 2D views are translated but not rotated from their original orientations.

3 Interactive HingeSlicer

The HingeSlicer is implemented in C++ using the open-source Visualization ToolKit (VTK) [7], and therefore runs under Linux and Microsoft Windows. We use the VTK class *vtkImagePlaneWidget* and extend it to add the HingeSlicer capabilities. The vtkImagePlaneWidget class implements a 3D widget for re-slicing volume

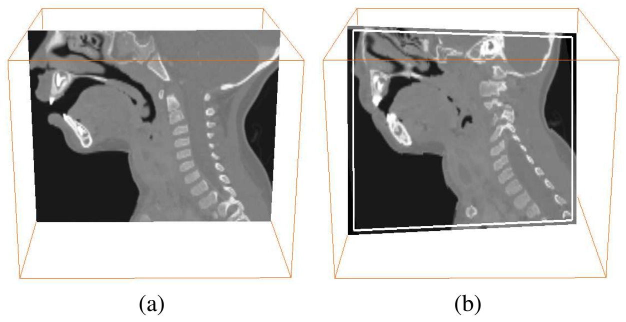

Figure 3: (a) 3D slice plane widget showing a 2D cross-sectional view of a CT volume image. (b) The slice plane can be rotated by first positioning the mouse within a delineated margin (left, right, top, bottom) and then dragging it.

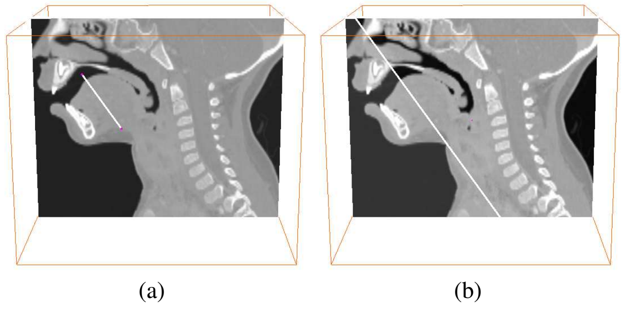

Figure 4: (a) The user draws a hinge line oriented with respect to an object of interest. (b) HingeSlicer uses the line segment to cut the plane into two pieces.

image data (Figure 3a). That is, the widget defines an image slice plane that can be interactively rotated around the plane center point into an arbitrary orientation, as well as translated along a direction normal to the slice plane. Rotation (Figure 3b) is achieved by using the mouse to highlight a set of “margins” (i.e. left, right, top, and bottom border regions of the slice plane). Positioning the mouse close to an edge of the widget while holding down the middle mouse button automatically highlights the margins. Subsequent dragging of the mouse generates slice plane rotation, with the rotation axis running through the plane center and oriented parallel to the line defining the margin (and hence the corresponding edge of the slice plane). Positioning the mouse within the central region of the slice plane (while holding down the middle mouse button) and then dragging generates slice plane translation in the plane-normal direction. We have found this interaction model to be simple and intuitive and decided to retain as much of it as possible.

To explore a volume using the HingeSlicer, the user draws lines to specify the location and orientation of a cut, along with a key to activate the cut. The user then uses the mouse, middle mouse button, and several keys to perform different widget manipulations. In the following sections, each action/manipulation and its interface is specified in detail.

3.1 Cutting and Folding

To begin volume image exploration with the HingeSlicer, the user typically starts with a single slice plane widget in a standard orientation (i.e. axial, coronal, sagittal). The user then rotates and translates this plane into a desired position to focus on a specific anatomical structure. The user may then draw an arbitrarily oriented line segment (the hinge line) on the slice plane by depressing and holding the left mouse button, dragging the mouse to extend and orient the line, then terminating it by releasing the mouse button (Figure 4(a)). A cut is then activated by a pressing the space-bar key. Line segments are typically drawn perpendicular or parallel to a major medial axis of the target structure, allowing the user to quickly expose a key cross-section. A clipping plane, automatically constructed from this hinge line, cuts the current slice plane, dividing it into two pieces (Figure 4(b)). Each piece can now be individually folded (rotated) around the hinge line, which acts as the axis of rotation (Figure 5). A piece is folded by selecting a margin opposite to the hinge line. A folded plane interpolates the volume image using cubic interpolation. Folding a piece of a slice plane along a hinge line allows a user to use knowledge of object shape and interactively explore parts of an anatomical structure with curved boundaries, while the other piece maintains a recognizable view of the structure. In Figure 5(a-b) for example, the user is able to examine (and measure) the cross section of the child’s jaw bone in this CT volume image by folding the left slice plane, while the right-hand slice plane maintains the contextual view of the head.

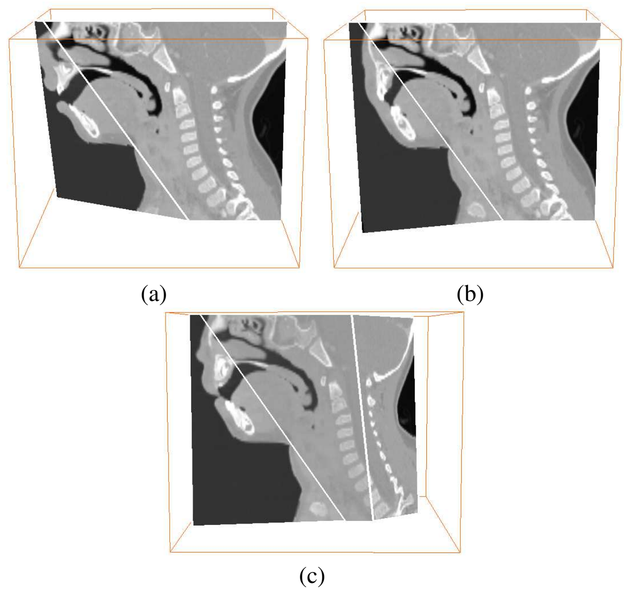

Figure 5: (a)(b) The user interactively folds (rotates) one piece with respect to the other, allowing the examination of object-aligned cross-sections. In this example, the user is examining the lower jaw.(c) Cut slice plane pieces may themselves be cut to progressively create extended views, allowing further exploration of the volume image.

Slice planes created by a cutting action may themselves be cut by drawing further hinge lines (Figure 5(c)). This allows the user to progressively extend a 2D view by forming new, connected views that follow the geometry of an object. Many anatomical structures are elongated and bend and/or twist in 3D space. A single 3D planar view or multiple, disconnected planar views are often inadequate for visualizing and measuring these structures. Extending the view helps to retain surrounding visual context. Attempting to achieve this with multiple disconnected planes results in visual clutter and occlusion, forcing the user to constantly toggle the visibility of each plane.

3.2 Sliding

The two pieces of the subdivided slice plane widget (the *hinge* slice plane widgets) can also slide (translate) in the pre-cut plane-normal direction away or towards each other. Sliding is performed by positioning the mouse within the central region of the slice plane, holding down the middle mouse button and dragging. This sliding action causes the automatic creation of a *connector* image plane, orthogonal to (and connecting) the two hinge pieces (Figure 6(b)). The connector slice plane automatically grows or shrinks as one hinge piece slides away or towards the other (Figure 6(b-d)). This mechanism allows the user to progressively examine or analyze object cross sections. A hinge line is drawn (typically perpendicular to a major axis of the object), the user pushes the two pieces apart and performs the examination.

The user may also push on the connector slice itself to examine other cross-sections (Figure 6(e-f)). This action grows and shrinks the connected hinge-plane pieces. Connector sliding is disallowed if one of the connected hinge planes is almost coplanar with it. Alternatively, more cuts can be made on the hinge plane pieces to allow viewing of other cross sections.

Finally, the user may select a margin on the hinge slice plane that is adjacent to the hinge line (rather than an opposite or non-adjacent margin, as is the case for folding) and slide one end of the slice plane (Figure 6(g-h)) while the other end remains fixed. This action is labeled an asymmetric *slide* rather than a rotation. Although the hinge slice plane is rotating, it is also stretching in such a way that the connector plane still grows and shrinks in the pre-cut plane-normal direction. Due to this stretching effect, visually the movement appears more like a slide than a rotation. Allowing a hinge slice plane to slide in this restricted manner provides it with more folding flexibility (different fold directions of the hinge planes can now be explored) while also providing enough constraint to maintain visual simplicity of the hinge-connector-hinge arrangement. Allowing a pure rotation here instead of the constrained slide, would quickly create concave and/or non-planar slice planes, destroying the simplicity of the cut-fold-slide interaction model.

3.3 Moving

This action is related to sliding. As with sliding, moving is performed by positioning the mouse within the central region of a hinge slice plane and holding down the middle mouse button. However, in this case the user also simultaneously depresses and holds the control key and then drags the mouse. The direction of this slide is given by a vector that lies within the hinge plane and is orthogonal to the hinge line. For example, in Figure 7(a-c), the user is moving the left-most hinge slice plane to the left. This causes the connector plane to automatically change its orientation to maintain its attachment. In Figure 7(d), the user has reverted to a regular slide, pushing the two hinge planes closer together along the newly oriented connector plane. The moving action provides very flexible and intuitive volume navigation control. The user is able to perform a series of slides, folds, and moves to quickly explore around and through the entire target anatomical structure.

3.4 Mending

The cut-fold-slide analogy provides a simple yet powerful method of extending an existing 2D view. This approach also lends itself well to an undo or "mending" action, allowing the user to progressively back up to a previous "good" view (Figure 8) and explore another region with a new cut. This property is especially important for controlling and editing interactive segmentations. To mend two hinge slice plane pieces, the user folds and slides one or both pieces until they are roughly aligned as they were before the cut (Figure 8(c)). Pressing the 'm' key activates the mend and the pieces are joined back together. Currently we allow planes to be mended in the reverse order in which they were cut.

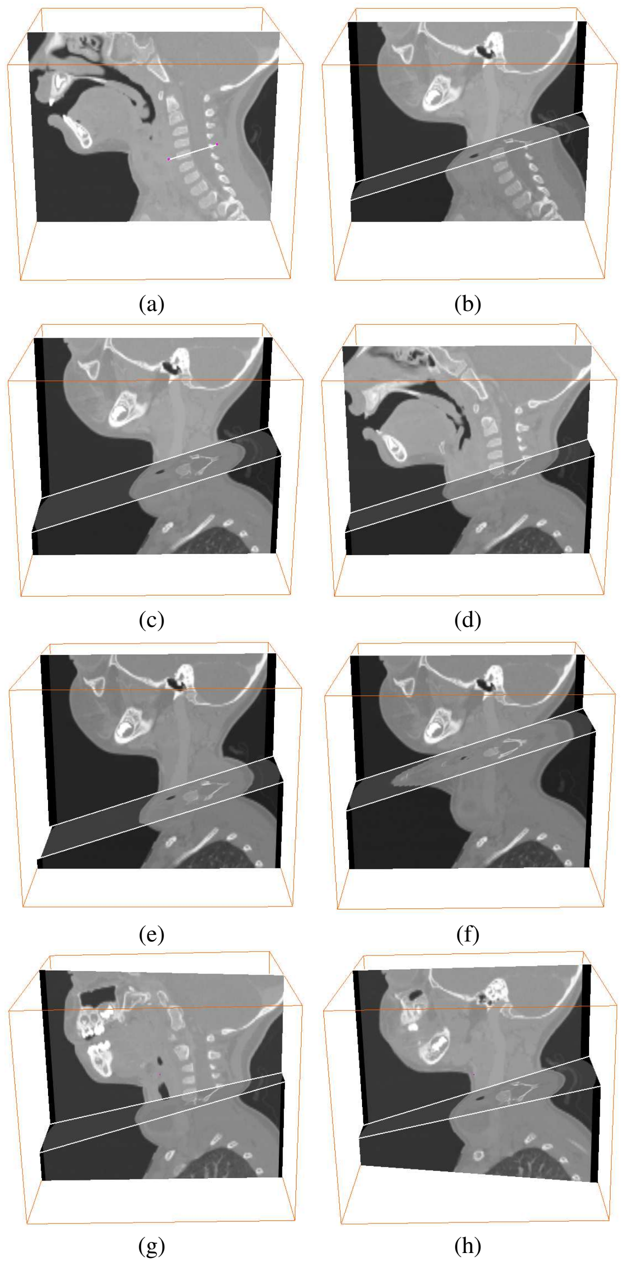

Figure 6: (a)-(d) Cut slice plane pieces can slide in a plane-normal direction away and towards each other, automatically growing/shrinking an orthogonal connector image slice. Object-aligned (in this case, vertebrae-aligned) cross-sections can then be examined.(e)(f) The connector slice plane can also slide, automatically growing/shrinking the connected hinge planes. (g)(h) One end of a hinge plane can slide while the other end remains fixed.

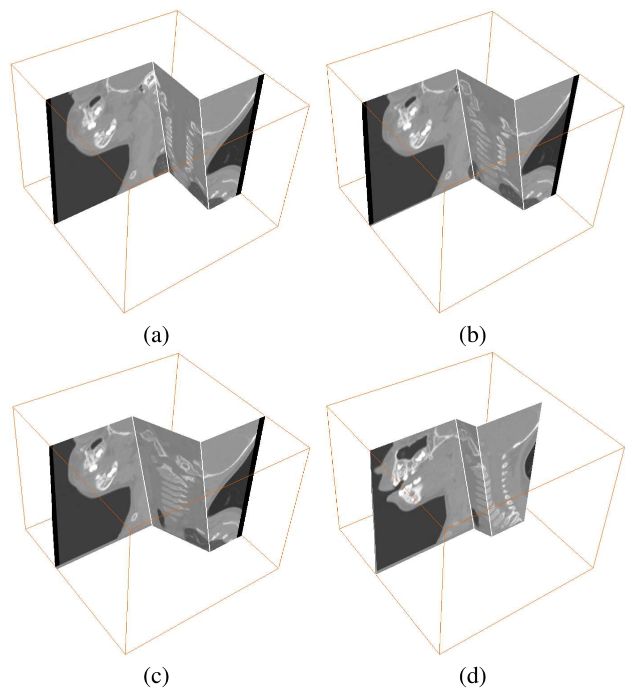
Figure 7: (a)-(c) Hinge slice planes can be grabbed and moved along a vector within the plane and orthogonal to the hinge line. This action drags along the attached connector plane, forcing it into a new orientation. (d) The hinge planes now slide along the the new connector plane orientation.

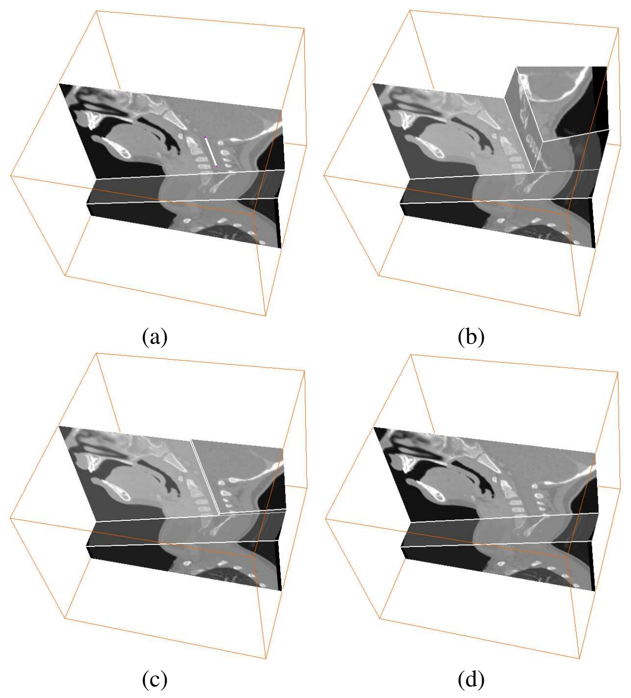
Figure 8: Extending existing views with cutting-folding-sliding provides for progressive mending (undo) actions, allowing the user to return to a "good" view. In (c), the user has moved the hinge planes into rough alignment. (d) A keystroke activates the mend, returning the view to the state before the second cut (a) was performed.

3.5 "Staircase"-ing

Rather than repeatedly cutting, sliding, and mending, a user may also successively cut and slide at several locations along a major axis of an object. This series of actions creates a staircase-like arrangement of the slice planes (Figure 9), providing a more global view of a structure. Aside from global visualization, one of the main ideas behind staircase views is to allow a user to more effectively initialize, monitor, steer, and edit an interactive segmentation algorithm. We are currently developing a new 3D segmentation technique that will integrate a deformable surface model with the HingeSlicer tool, specifically using staircase views to steer and edit the model.

Staircasing often results in the formation of a special type of connector slice plane, which we refer to as a "shared" connector. For example, the two hinge planes formed by the second cut in (Figure 8(b)) have two connector planes. The first connector (referred to as the primary connector) connects the two hinge planes along the hinge line formed by the second cut, as usual. The other connector plane (the "floor" in Figure 8(b)) is a secondary, shared connector. It is connected to not only the two hinge planes formed by the second cut, but also to the *connector* plane created from the first cut. Shared connectors occur when cuts are made such that the resulting hinge lines intersect a previous hinge line (as is the case the Figure 8 and Figure 9). To prevent visually confusing slice plane arrangements from occurring, we currently place restrictions on shared connectors and on the hinge planes that are connected to them. These restrictions are discussed in Section 3.8.

3.6 Child Orthogonal View Planes

Most visualization packages provide the ability to create an arrangement of two or three oriented 3D slice plane widgets that are approximately orthogonal to each other. While perhaps not as effective as a true 3D view, this arrangement allows the user to examine and manipulate three different object cross-sections together and provides some understanding of 3D object shape, with relatively little effort (i.e versus setting transfer function parameters in a volume rendering to generate a clear 3D view of a vaguely defined target structure). This is especially true for objects whose overall shape is relatively spherical or slightly elliptical. Unfortunately, these orthogonal planes are typically independently initialized and positioned within the volume image. Furthermore, for objects that are either elongated, highly-curved, contain significant protrusions, or some combination thereof, orthogonally arranged view planes are much less effective. Tedious and confusing manipulations of each plane are required to progressively explore the 3D shape and position of such objects.

With the extended views of the HingeSlicer, supplemental orthogonal view planes can be more easily tied to each slice plane (Figure 10(b)). We currently allow two "child" orthogonal planes (each orthogonal to each other as well as to the "parent" slice plane) to be interactively created for each slice plane. To create and position a child slice plane widget, the user draws a line on the parent and presses the 'c' key. The length of the line is used to initialize the size of the child plane. A child plane can be translated along the parent plane (Figure 10(c)), can be interactively "grown" (i.e. stretched) by "pulling" on a margin of the plane with the mouse while holding the 'control' key, and can be rotated about an axis normal to the parent. In addition, as a parent hinge plane is folded along its hinge line, the child planes "follow" along, maintaining orthogonality (Figure 10(b)). Finally, all slice plane widgets can be interactively rendered transparent (with an optional opaque outline rendered to maintain context) using the mouse to select the plane and the 'v' key to toggle transparency (Figure 10(d)). Small orthogonal child planes provide another option for a user to explore a specific region of a complex-shaped object, without affecting the

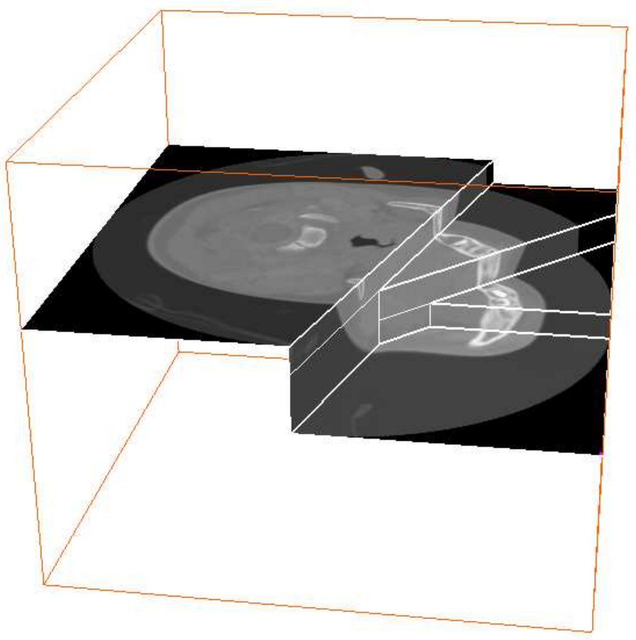

Figure 9: Exploring the left lower jaw of by drawing a series of hinge lines perpendicular to the jaw medial axis. This compound cross-sectional view was created in seconds using the Hinge Slicer. Multiple object-aligned cuts and folding/sliding are used to form these staircase-like arrangements of image cross-sections, providing a more global overview of an object.

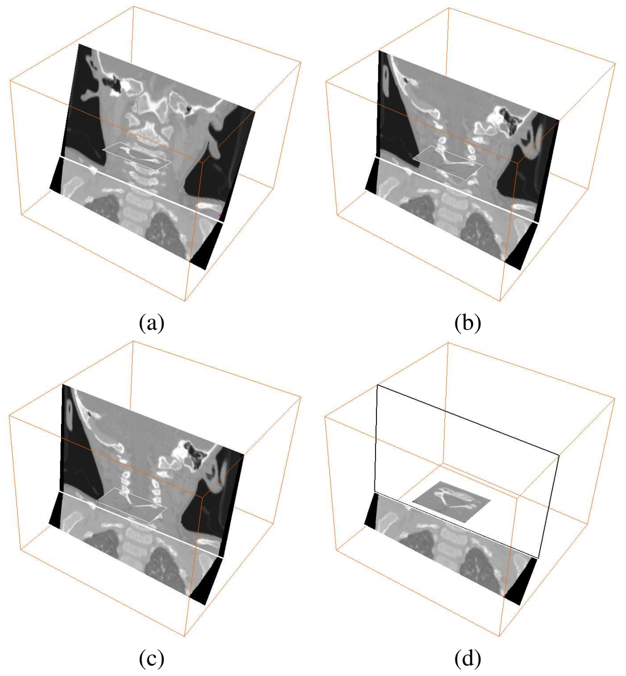

Figure 10: (a) "Child" orthogonal view planes can be selectively added to any "parent" slice plane. The child plane can be translated, scaled or rotated with respect to the parent. (b)(c) If the parent is rotated or translated, the child is constrained to maintain orthogonality. (d) The parent slice plane can be made transparent to prevent occlusion. The outline of the parent is retained for context.

underlying parent slice plane arrangement and with minimal occlusion.

3.7 Implementation

As mentioned previously, our 3D slice plane widget is a subclass of the VTK class *ImagePlaneWidget* and extends its functionality. Each 3D slice plane widget is defined and controlled by an oriented plane. The plane is specified with an origin point and two other points that together define two axes for the plane (and hence a local coordinate system - Figure 11 upper left). This plane is used to slice the volume image (using the VTK class *vtkImageReslice*) and generate an image cross-section.

The slice plane widget is displayed using an in-plane polygonal mesh. In the *ImagePlaneWidget* class, this mesh consists of one rectangle defining the spatial extent of the defining plane. In our *HingeSlicePlaneWidget* class, as the slice plane is progressively cut, the mesh becomes divided into triangles and quadrilaterals (Figure 11 upper left and right) and may no longer spans the extent of the defining plane. This mesh is texture-mapped to display the cut slice plane piece. The mesh point coordinates (and mesh point texture coordinates) are defined with respect to the widget's local coordinate system and are computed once only when a slice plane widget is created by a cutting action.

When the user selects a widget and moves the mouse, the widget transforms the movement of the mouse into a change in angle or position. The widget origin point and axes end points are then rotated around the hinge line axis or translated along the plane normal, updating the plane definition. The transformed plane is used to re-slice the volume, generating a new image cross-section spanning the full extent of the slice plane. The 3D positions of the display mesh points are recomputed with respect to the transformed plane coordinate system using their local coordinates. Similarly, local texture coordinates are also used for texturing the mesh by making use of the VTK class *TextureMapToImagePlane*. Separating a widget's display polygons from its defining oriented plane (Figure 11 lower) greatly simplifies the implementation of slice plane cutting.

Implementing a cut of a slice plane proceeds as follows. The cut line segment drawn by the user, along with the slice plane normal, are used to construct a clipping plane. The slice plane is clipped and the resulting polygonal mesh from the "inside" half of the plane are used to replace the plane's current polygonal mesh (Figure 11 upper right, dark gray shaded polygons). A copy of the defining plane is made (Figure 11 upper right, light gray shaded polygons) and the polygonal mesh from the "outside" half replaces the display polygons. Currently the display polygons of the two new hinge planes do not share the hinge line end points - each maintains a separate copy. Hinge plane connections are handled as described in the next paragraph.

As mentioned in Section 6, a cutting action automatically creates an orthogonal connector plane. We use the connector planes as their name implies - to connect hinge planes. Each connector plane keeps track of (and is defined by) the two hinge planes it connects. Conversely, each hinge plane stores a list of all its connector planes. As a hinge plane is translated or rotated, it updates its display polygon points and informs each of its connector image planes of the change. The connectors query the hinge planes for the updated hinge line end points and redefine themselves.

3.8 Current Restrictions and Limitations

Currently, once a slice plane is cut such that it is surrounded on two opposite edges by two other slice planes (i.e. an "internal" plane - for example, Figure 12D, the hinge plane labeled '2'), it becomes a restricted plane and may slide but cannot be folded or cut. This limitation is also true of connector planes, which are by definition

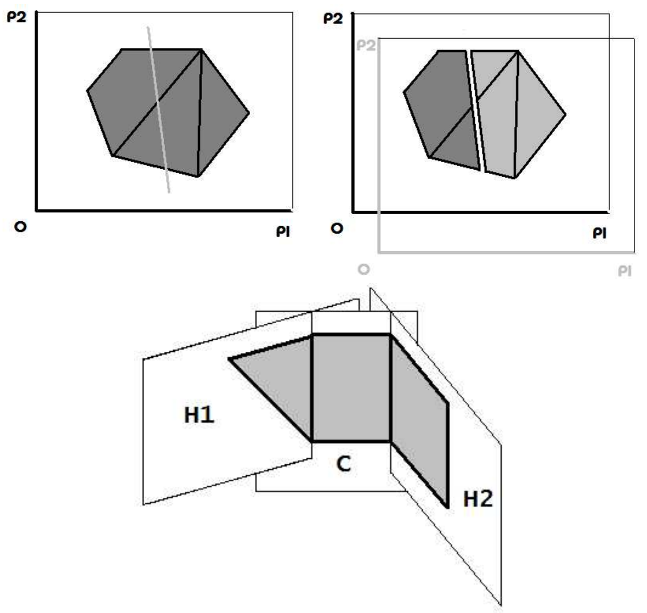

Figure 11: Upper left: diagram of slice plane widget showing defining plane and coordinate system along with display polygons. The line through the polygons is the cut line. Upper right: the slice plane has been cut into two pieces. The two resulting pieces copy the defining coordinate system and redefine their display polygons. The coordinate system of the right-hand piece (light gray) has been offset for emphasis. Lower: Diagram of two hinge pieces and their connector piece. Each is defined by a plane which cuts the volume image and each maintains its own display polygons

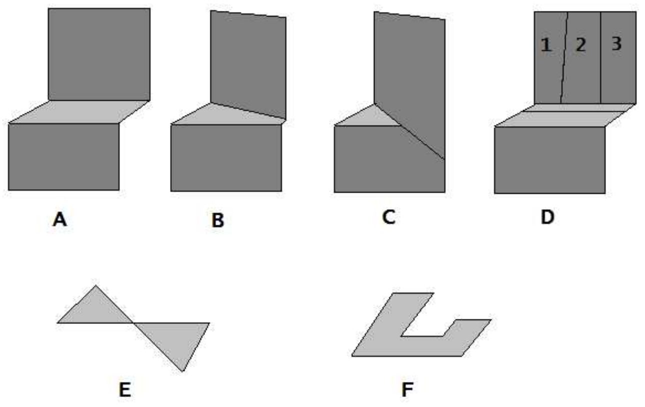

Figure 12: (A)(B)(C) The movement of hinge slice planes may cause the connector plane to self-intersect (E). (D) Many small slices from multiple cuts may share one connector plane. Movement of these planes may cause the display polygon of the shared connector plane to become concave (F).

internal. Hinge planes (i.e. planes that can fold) are referred to as "leaf" planes. This restriction is primarily an implementation issue - disallowing rotation of internal planes avoids software complexity - although it is not entirely clear how rotation should behave for these planes. One of our main design goals was to prevent a change in hinge slice plane orientation from occurring as a side effect of the user-affected change in the orientation/translation of a neighboring plane. The idea here is that the user may have spent some effort to orient a leaf plane and create a useful view. This view should not be affected unless the user explicitly initiates the change. In our experience, widget orienting is often difficult and therefore one of the primary goals of the HingeSlicer was to minimize the amount of slice plane orienting required. We have attempted to achieve this goal by drawing lines and using relative hinge plane rotation. Nevertheless, the user may need to occasionally fine-tune the orientation and position of an internal plane. In the immediate future, we plan on at least partially removing the folding restriction to make the interface more consistent. The ideal goal would be to present a single plane type to the user. However, to achieve our design goal, "rotation" of internal planes will not be true rigid rotation, but will necessarily involve sliding and stretching one end of the plane while holding the other end fixed (i.e. asymmetric sliding discussed in the last paragraph of Section 6).

Connector plane display issues can arise under various circumstances. For example, in Figure 12A-D, hinge planes may slide such that their connector plane self-intersects (Figure 12E). Furthermore, as the user folds and slides small hinge planes that share a secondary connector plane (Figure 12D), it is possible for the display polygons of the shared connector plane to become highly concave (Figure 12F). There are several solutions to these two related problems. One is to impose constraints on folding/sliding and prevent the occurrence of these anomalous arrangements. A second approach is to break up concave shaped or self-intersecting connector polygons into several convex polygons. We have chosen a third solution and restrict the connector plane display to a single quadrilateral. This means that in the case of Figure 12D, the secondary shared connector is always displayed as a single rectangle which extends beyond the small hinge planes, ensuring there is always a "floor" underneath them. In the case of self-intersection, this situation is detected and a quadrilateral convex hull is constructed and displayed. This solution was the simplest to implement and allowed us to focus on the development of the interaction model. In the future, we will move to the second solution, as it provides a more natural visual transition to the user.

Another restriction we currently impose is on hinge planes that share a connector plane (Figure 12D, planes 1 and 3). Although these planes may fold along their primary hinge line, they currently are unable to undergo asymmetric sliding. One solution to this problem would be to allow the shared (floor) connector to fold and drag all of the hinge planes that share it along with it. Another possibility would be to allow the user to remove the shared connector entirely.

4 Discussion

The idea of the hinge slicer emerged out of many years working with interactive surface/curve model-based segmentation algorithms and other 3D interactive image processing algorithms. As most visualization packages support multiple interactively oriented slice planes widgets and the volume images used in our segmentation experiments were often noisy, it was a common occurrence to monitor and interact with the surface model as it dynamically fit itself to the boundary of a target anatomical structure. However, manipulating multiple independent slice plane widgets in an attempt to track and view the current position of the model and its target position was an exercise in frustration, especially for elongated or curved objects and objects with protrusions. 3D views generated using volume rendering were of little use in these situations, as setting transfer function parameters to obtain the desired view was akin to performing another segmentation to view the progress of the current segmentation.

The key to enhancing 3D slice plane usefulness, in our opinion, is the use of quick and simple creation, positioning and orienting (by drawing oriented lines) of a slice plane with respect to an existing plane. A second key feature is the ability to extend and mend (i.e. return to) existing good views. We have chosen to use the cut-fold-slide interaction model to achieve these abilities and the results are very promising. Drawing lines on planes across or along anatomical structures is simple and intuitive and folding and sliding are very natural actions. Does it provide the user with better (or at least faster) insight? In our opinion, with two or three connected planes, it is faster and easier to explore and follow a complex shaped object than with the traditional individual disconnected planes. We believe this ability by itself makes the HingeSlicer an effective tool (Figure 13).

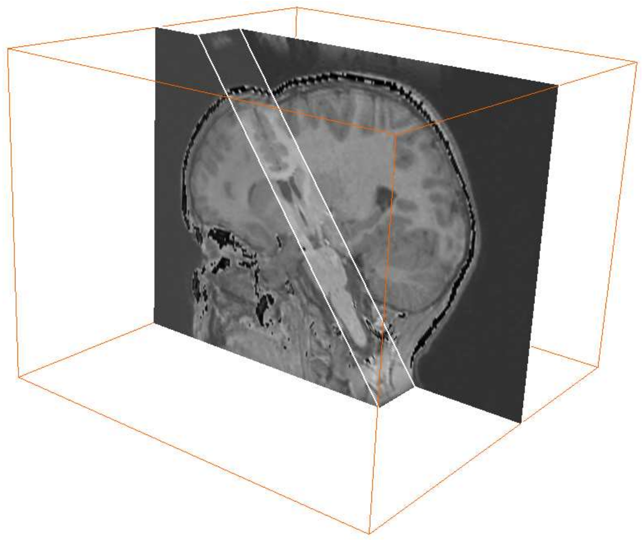

Figure 13: Exploring the caudate nucleus and lateral ventricle in an MR brain volume. Simple cut-slide-mend, cut-slide-mend sequences are often sufficient to quickly examine object-aligned cross sections along a curving object.

The usefulness of the more elaborate staircase views, involving more than three connected planes, is more open to question and perhaps can best be answered by expert users. For this reason, we will make the software publicly available in the near future. The issue is not one of view construction, as it is simple and intuitive to make several cuts in quick succession and slide/fold to create a complex view. Which view of an anatomical structure to create is the more pertinent question. There are often several possible directions of approach and choices of hinge line cuts and folds for anatomical structures. However, we suspect that this issue will be resolved simply with further user experimentation.

There are two more critical issues regarding elaborate views. One is interface consistency - the more elaborate the view, the more constraints on slice plane pieces are introduced and this may frustrate the user in some cases. Most of the interface restrictions and constraints are implementation related, but there are likely some interface consistency issues arising from the cut-fold-slide interaction model itself. A second issue is under what scenarios are the complex views necessary? In our own research, we envision that these staircase views will allow a user to view and monitor an evolving deformable surface simultaneously at different cross-sections of a target anatomical structure, allowing us to "reach out" and steer the model back on course at any one of the cross-sections. This ability has been sorely lacking in semi-automatic shape-model based segmentation systems developed to-date, resulting in a tedious post-segmentation editing phase, where the user scans through a series of slices and edits the cross-section of the model. Fast, accurate, repeatable 3D semi-automatic segmentation is a critical phase for a host of medical image analysis tasks. We are currently integrating the HingeSlicer into 3D segmentation system and hope to have a better idea of the effectiveness of these elaborate views in the near future.

5 Conclusion

We have created an interactive tool for exploring and analyzing volume images. By making use of a new 3D interaction model and extending the functionality of existing 3D slice plane widgets, the tool allows expert users to quickly and easily create custom, extended cross-sectional views of anatomical structures. By using cuts aligned with structure geometry along with a series of intuitive slice plane manipulations, users are able to examine key object cross-sections along a complex-shaped object, while maintaining a global contextual view. This is in direct contrast to most existing visualization systems which support scanning of volume images using either a single slice plane widget or multiple, disconnected, independently-controlled slice planes. Future work includes continuing to improve the consistency of the interface, combining the cross-sectional views generated with the slice planes with 3D transparent views of enclosing surfaces (such as the skin) to improve understanding of global context, and adding functionality to automate the creation of custom views.

Acknowledgements

The CT scan is courtesy of Dr. Paul Babyn, Chief Radiologist, The Hospital for Sick Children, Toronto, ON.

References

[1] M.S.T. Carpendale, D.J. Cowperthwaite, M. Tigges, A. Fall, and F.D. Fracchia. The tardis: A visual exploration environment for landscape dynamics. In *Proc. Visual Data Exploration and Analysis VI*, 1999.

[2] H. Delingette, M. Hebert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. *Image and Vision Computing*, 10(3):132–144, April 1992.

[3] K. Hinckley, R. Pausch, J.C. Goble, and N.F. Kassell. A survey of design issues in spatial input. In *Proc. ACM Symp. User Interface Software and Technology*, pages 213–222, 1994.

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

[5] T. McInerney and D. Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Trans. on Medical Imaging*, 18(10):840–850, March 1999.

[6] Mercury Computer Systems. *Amira 3.0 User's Guide and Reference Manual*. Mercury Computer Systems, Inc., USA, 2002.

[7] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Kitware, Inc., USA, 2004.

[8] M. Tory, A.E. Kirkpatrick, M.S. Atkins, and T. Möller. Visualization task performance with 2d, 3d, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):2–13, January/February 2006.

[9] Visualogic. *Slicer Dicer User's Guide, www.slicerdicer.com*. Pixotec, Inc., USA, 2006.

[10] Daniel Weiskopf, Klaus Engel, and Thomas Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Trans. on Visualization and Computer Graphics*, 9(3):298–312, 2003.

[11] P.A. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S.Ho, J.C. Gee, and G. Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, To appear 2006.