2017

Hybrid rendering of exploded views for medical image atlas visualization

Tim McGraw

Alejandro Guayaquil-Sosa

To appear in *Computer Methods in Biomechanics and Biomedical Engineering*
Vol. 00, No. 00, Month 20XX, 1–16

Hybrid rendering of exploded views for medical image atlas visualization

Tim McGraw[a]* and Alejandro Guayaquil-Sosa[a]

[a] *Purdue University, West Lafayette, IN, USA*;

(*v1.0 released February 20xx*)

Medical image atlases contain much information about human anatomy, but learning the shapes of anatomical regions and making sense of the overall structure defined in the atlas can be problematic. Atlases may contain hundreds of regions with complex shapes which can be tightly packed together. This makes visualization difficult since the shapes can fit together in complex ways and visually obscure each other. In this work we describe a technique which enables interactive exploration of medical image atlases that permits the hierarchical structure of the atlas and the content of an underlying medical image to be investigated simultaneously. Our method enables a user to create visualizations of the atlas similar to the exploded views used in technical illustrations to show the structure of mechanical assemblies. These views are constrained by the geometry of the atlas and the hierarchical structure to reduce the complexity of user interaction. We also enable the user to explode the atlas meshes themselves. The atlas meshes are registered with a medical image which is displayed on the cut surfaces of the meshes using raycasting. Results from the AAL human brain atlas are presented and discussed.

Keywords: biomedical visualization; volume rendering; brain atlas

1. Introduction

A medical image 'atlas' containing a description and visualization of human anatomy is often consulted during diagnoses, surgical planning or for educational purposes. The atlas images are often acquired from dissection photographs or histological slices which have been labeled by an expert. The labels may describe the tissue classes or outline specific anatomical structures. In medical image analysis, the term 'atlas' is also applied to images generated from a collection of datasets which are processed to obtain a represention of an average subject. The construction of an atlas involves registering multiple images of the same modality, e.g. MRI, into a common coordinate system, then processing them to construct a single representative image. The result is an image which is independent of the minor anatomical variations in the individual images. From this 'template' image a 'label' image can be created to describe the tissues and structure in the atlas. The atlas can be used to plan surgical procedures, as educational material (Kikinis et al. 1996), or as source of prior anatomical knowledge in automated medical image analysis tasks such as segmentation (Lötjönen et al. 2010; Wang et al. 2013) and image reconstruction (Custo et al. 2010).

An image atlas and an image registration procedure can be used to automatically label new medical images. The process of registering two images yields a coordinate transformation from one image to the other. A given medical image can be automatically labeled by registering an atlas template with that image. Then the coordinate transformation can be used to map the labels from the atlas to the target image.

Atlases are often visualized by color coding the labeled regions and overlaying them on the

*Corresponding author. Email: tmcgraw@purdue.edu template image or another registered medical image, as shown in Figure (2). A 3D visualization approach is to extract the boundaries between regions as triangle meshes, and then rendering these meshes. To provide visual context, the surfaces may be rendered with multi-planar reformatted images or volume renderings. In this paper we propose an alternative approach that permits the user to explore the atlas meshes and gain valuable insight into the spatial relations between labeled atlas regions and a medical image. Our technique generates interactive exploded views using a hybrid rendering method which combines aspects of polygonal surface rendering and raycasting. The user can then explore the 3D structure of the atlas at variety of levels - cerebral hemisphere, lobe, atlas region and image voxel.

The technique presented here can be applied to the problem of visualizing an arbitrary medical image by fully segmenting the image and extracting surface meshes, or by registering an existing atlas to that image. A potential application of our technique is the development of interactive educational materials for medicine, anatomy and radiology. It has been found in many previous studies that dynamic and interactive visualizations can lead to improved learning outcomes and spatial reasoning (Höffler and Leutner 2007; Sitzmann 2011).

The main contributions of our work are

- An efficient hybrid rasterization/raycast rendering method for visualizing exploded meshes. This permits image data to be displayed on cut surfaces while showing the shape of the anatomical region for visual context.
- A technique for interactively generating exploded views based on the hierarchical relations among objects in the atlas, which permits individual atlas regions to be displayed and their place within the global structure of the atlas to be understood.
- Results of applying this technique to the AAL brain atlas

In the remainder of this paper we will review the related literature, describe the implementation details of our approach, present the results and discuss conclusions and future work.

2. Related Work

Our technique is closely related to recent work in the field of illustrative visualization, especially 'focus and context' which are concerned with enabling users to focus on specific details of a large dataset while still providing relevant contextual information about the structure of the data. In this section we provide an overview of the related work.

Clipping planes work by by simply discarding the data on one side of a plane. The use of nonplanar clipping geometry has been explored (Weiskopf et al. 2003), including using heightfields as a clipping primitive (Trapp and Döllner 2013). None of these approaches overcome the main drawback of clipping, which is that all voxels within the clipping volume are treated equally. This can result in valuable contextual information being removed. Even with nonplanar clipping surfaces it is difficult to create clipping geometry which removes only unwanted features while keeping the desired contextual data. Díaz et al. (2012) addressed this problem by adaptively clipping only selected tissue classes.

Exploded views **improve the spatial layout of a scene to emphasize the structural relations among components in a compound object. For example, Li et al. (2008) generated 3D exploded views of Computer Aided Design (CAD) models based on the hierarchical relations between parts, subassemblies and assemblies. An explosion graph is used to compute displacements which prevent the objects from overlapping. Cutting planes are used to split objects which contain other parts. A related technique for anatomical surface models by McInerney and Tran (2015) is used in a system called 'Aperio' that can generate cutaways and exploded views of anatomical surface models. Unlike other approaches, explosions in 'Aperio' can displace parts along curved paths.**

Volume splitting techniques generalize the concept of exploded views to volumetic datasets.

By applying different spatial transformations to selected subregions of a volume the regions can be separated and rotated to generate unoccluded views of anatomical structures. McGuffin et al. (2003) presented a taxonomy of approaches to transforming the various semantic layers of a volume dataset, including peeling, cutting and hinging. Islam et al. (2007) described using 'spatial transfer functions' to split the outer layers of a volume, allowing the inner layers to be clearly seen. Bruckner and Groller (2006) generate view-dependent exploded views using force-directed layout. Grimm et al. (2004) presented a method for efficiently rendering multiple overlapping volumes which were split. Correa et al. (2007) developed a method for deforming volumes such that specified areas of interest were revealed. In our technique we use geometric primitives to split and explode surfaces to reveal a 3D medical image which would otherwise be invisible. By working within a user-selected level of the anatomical hierarchy we allow the user to easily control the extents of the splitting - hemisphere, lobe, region, or a combination.

Focus and context visualization techniques emphasize selected features of interest while displaying contextual information about nearby datapoints. McInerney and Crawford (2010) developed a mesh visualization technique which cuts away portions of the mesh to reduce occlusion, leaving "ribbons" of the original surface to provide visual context. Li et al. (2007) generated cutaway views of anatomical and CAD models in the visual style of technical illustrations which revealed layered interior structures. Sigg et al. (2012) developed a system for clipping away parts of a volume dataset to reveal specific areas of interest. The optimal location for cutting geometry was estimated using Monte Carlo methods. Auzinger et al. (2013) dynamically construct a cutting surface through a volume which follows blood vessel centerlines which permits vascular structures to be displayed in the context of the surrounding anatomy. A more thorough overview of focus and context in volume rendering is given by Bruckner et al. (2010).

Nonlinear deformations are often used in surgical simulations, such as those described by Herrera Asteasu (2013). These systems incorporate a realistic deformation model for multiple tissue classes in order to train surgeons for educational purposes, or to plan for future procedures on a patient. Although these systems allow exploration of anatomical structures, smooth physically-based deformations are time-consuming to simulate and can result in extreme deformations of shapes. Correa et al. (2010) presented a deformation method capable of simultaneously applying rigid deformations to selected structures and nonrigid deformations to others without computationally-expensive physical simulation. In our application, we avoid nonlinear deformations since they can distort shapes, making it difficult to recognize the anatomical regions in the atlas.

Hybrid rendering methods use combinations of different surface representations and rendering techniques to take advantage of the benefits of each one. For example, a hybrid terrain rendering technique was developed by Ammann et al. (2010). They used rasterization of polygonal meshes and raycasting of heightfields to visualize dynamic terrains. The graphics techniques of parallax mapping and relief mapping (Policarpo et al. 2005) are based on a similar approach. By performing raycasting in tangent space, these methods are able to add per-pixel displacement detail to polygonal meshes. In our method we combine triangle meshes, which represent the atlas meshes, and implicit functions, which represent the clipping geometry. Signed distance representations of the clipping geometry, specifically, have the added benefit of allowing empty space around the primitives to be efficiently skipped in many cases.

3. Methods

Our rendering method utilizes a combination of rasterization and raycasting to visualize atlas meshes in the context of a registered 3D image. We display anatomical structures which have been segmented from an atlas and permit the user to interactively explode the anatomical hierarchy. Unlike other approaches to atals visualization, in our system atlas meshes can be further split into pieces. When the user explodes individual meshes the interior of the mesh reveals an underlying 3D medical image. The mesh interior is rendered by raycasting using the segmented meshes as proxy geometry. In this section we describe how we assemble the hierarchy, the user interactions we support, and the details of the rendering process.

3.1 *Hierarchical model assembly*

Like the exploded views used for technical illustrations of mechanical assemblies (Li et al. 2008), anatomical atlases also have a hierarchical structure. Unlike most CAD applications our scene can also include a corresponding 3D medical image.

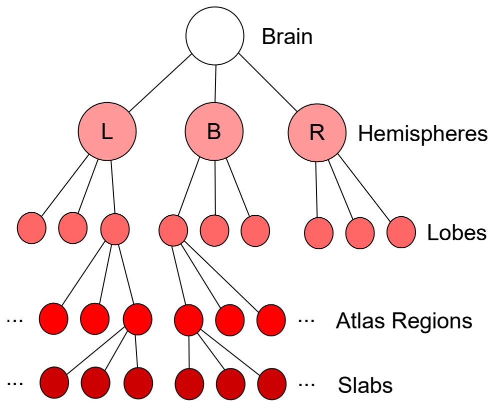

Figure 1. Hierarchical structure of the brain. From top to bottom the levels are: brain, hemisphere, lobe, region, slab.

Surfaces were extracted from the automated anatomical labeling (AAL) brain atlas which has labels for 116 gray matter structures and a template image registered into the same coordinate system (Tzourio-Mazoyer et al. 2002). A slice of the atlas is shown in Figure (2). We further group the 116 labels into brain hemispheres (left, bilateral, right) and brain lobes (listed in Figure (3)).

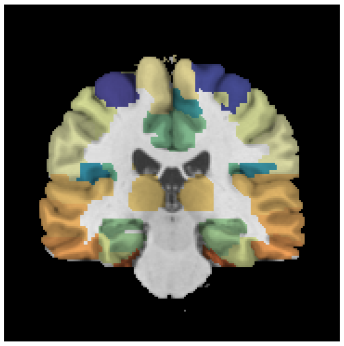

Figure 2. A coronal slice of the AAL atlas with color-coded regions overlaid on T1-weighted MRI.

The AAL atlas includes a text file which describes the mapping from numerical labels to region names. The names include a postfixed hemisphere indicator, e.g. `Precentral_L`. The regions which do not belong to the left or right hemisphere are considered bilateral. The atlas does not describe which cerebral lobe each region belongs to, but that information can be found from other atlases and medical imaging applications, such as PMOD Technologes Ltd. (2015).

A wide variety of software applications exist for processing medical image atlases, including many based on the ITK (Johnson et al. 2015) and VTK (Schroeder et al. 2006) libraries. We used ITK-SNAP (Yushkevich et al. 2006) to segment the atlas label image and generate triangle meshes. As the atlas is loaded in our application we compute an axis-aligned bounding box for each region. A complete bounding volume hierarchy, from regions to lobes to hemispheres, is then built from the bottom up.

Temporal Lobe	Hippocampus, Parahippocampus, Amygdala, Fusiform gyrus, Heschl gyrus, Superior temporal gyrus, Temporal pole: superior temporal gyrus, Middle temporal gyrus, Temporal pole: middle temporal gyrus, Inferior temporal gyrus
Posterior Fossa	Cerebellum, Vermis, Medulla, Midbrain, Pons
Insula and Cingulate Gyri	Insula, Cingulate gyrus (ant., mid, post.)
Frontal Lobe	Precentral gyrus, Superior frontal gyrus, Middle frontal gyrus, Inferior frontal gyrus, Rolandic operculum, Supplementary motor area, Olfactory cortex, Gyrus rectus, Paracentral lobule
Occipital Lobe	Calcarine fissure and surrounding cortex, Cuneus, Lingual gyrus, Occipital lobe (sup., mid. and inf.)
Parietal Lobe	Postcentral gyrus, Superior parietal gyrus,Inferior parietal gyrus, Supramarginal gyrus, Angular gyrus, Precuneus
Central Structures	Caudate nucleus, Putamen, Pallidum, Thalamus

Figure 3. Lobes and their constituent AAL labels. Most of the 116 AAL regions consist of left-right hemisphere pairs. The remainder are bilateral.

3.2 *Exploded view generation*

Trees and directed acyclic graphs are hierarchical data structures which have many applications in computer graphics. They can be used to represent relative transformations between objects in a scene graph or the transformations of bones in a skeleton used for character animation (Marschner and Shirley 2015). Although medical image atlases have an implicit hierarchical structure, there is no underlying skeleton which defines the locations of joints or constrains the types of transformations between structures. Relying on the user to manually specify all of this information would be burdensome, time-consuming and error-prone. We simplify interaction by permitting the user to select rotation axes and positions from the precomputed bounding volume hierarchy.

CAD models describing mechanical systems are usually associated with an assembly process which can be used to define an exploded view. Parts are combined into subassemblies, and subassemblies are mounted to assemblies. Biological systems which grow into place don't have a similar assembly process. In this work we use the hierarchical structure of the atlas, and the hierarchy of mesh bounding volumes to guide the development of exploded views. Specifically, we use the parent-child relationships and the location of bounding boxes edges to constrain the set of transformation we allow when exploding the atlas. The axes of the bounding boxes which are constructed during hierarchical model assembly are used as explosion directions and rotational axes. These directions initially correspond to the coronal, axial and sagittal radiological imaging planes. However, after applying transformations to nodes in the hierarchy the principal directions may vary among nodes.

Transformations are represented at each node, i, of the hierarchy by a unit quaternion, $\mathbf{q}_i$, a rotation axis position, $\mathbf{p}_i$, and a translation vector, $\mathbf{d}_i$. **As transformation parameters are updated we do not change them discontinuously, but instead change them smoothly by linearly interpolating the translation and position, and using spherical linear interpolation (slerp) on unit quaternions (Shoemake 1985) to improve visual coherence.** We also permit the user to undo and redo the transformation at each node with a button press by interpolating to an identity transformation and back again. We compute the displayed centerlines of explosions using this same transformation interpolation function.

We support rotational fanning and linear (axis-aligned and radial) explosions at the hemisphere, lobe region and slab levels of the hierarchy. During interaction the user may select a desired hierarchical level and the bounding boxes of selected entities at that level are displayed.

Radial explosions of selected nodes n_i in the hierarchy are performed by applying displacement

$$\mathbf{d}_i = s(\mathbf{c}_i - \mathbf{c}_{i \to parent}), \tag{1}$$

where s is a degree-of-explosion parameter which the user selects with a slider, and $\mathbf{c}$ are bounding box centers.

Axis aligned explosion displacements are given by

$$\mathbf{d}_i = s\mathbf{e}\left(\frac{w_{sorted(i)}}{2} + \sum_{j=sorted(0)}^{sorted(i-1)} w_j\right), \tag{2}$$

where $\mathbf{e}$ is the direction of the explosion (selected from 3 conventional imaging directions), and w is the bounding box width in that direction. The set of sorted indices is obtained by sorting based on the bounding box center coordinate corresponding to the axis direction. For $s = 1$ this transformation results in the bounding boxes of the exploded nodes having no overlap along direction $\mathbf{e}$.

Fanning transformations for meshes are performed by computing the polar coordinates (r, θ) of the displacement vector from the bounding box center of the root of the hierarchy to the bounding box center of the selected nodes, and then scaling θ. This is done by applying a rotation to each mesh given by $\mathbf{q}_i = (s\theta_i, \mathbf{a}_s)$, $\mathbf{p}_i = \mathbf{b}_0$ where $\mathbf{b}_s$ is the center of the root and $\mathbf{a}_s$ is the selected axis of rotation.

To fan slabs of meshes, the rotation axis is a user-selected edge of the mesh bounding box and the center point is an endpoint of that edge. The rotation angle of each slab is set by $\theta_i + s * instanceID$. This transformation rotates the slabs relative to each other to make the interior more visible. In the next section we will describe how we map the 3D image onto the interior of these slabs.

At render time each node's transformation, $\{\mathbf{q}_i, \mathbf{p}_i, \mathbf{d}_i\}$, is converted to a matrix $M_i = T_d T_p^{-1} R_q T_p$, where R_q is a rotation matrix computed from $\mathbf{q}_i$ and T_d and T_p are translation matrices computed from $\mathbf{d}_i$ and $\mathbf{p}_i$ respectively.

The object-to-world transformation matrix for objects at each level of the tree are obtained by walking up the tree toward the root, multiplying matrices on the left at each step. For example, the full transformation matrix for a slab would be given by $M_{brain} M_{lobe} M_{region} M_{slab}$.

3.3 *Hybrid rasterization / raycasting*

The second-to-bottom level of our scene hierarchy consists of the segmented atlas meshes, but users may visualize the interior of these meshes by exploding them into pieces. Exploded meshes are represented by a combination of triangle meshes and implicit surfaces (represented as signed distance functions). Once exploded, the interior of a mesh is raycast to show the underlying medical image on the cut surfaces. Other approaches to cutaway meshes use constructive solid geometry (CSG) at the object-level to generate new meshes (McInerney and Tran 2015) or use image-space stencil buffer techniques (Li et al. 2007). Although the object-level CSG approach can split meshes during a brief pause, we maintain consistent realtime framerates by using raycasting to achieve the same visual effect. This approach also permits us to change the the cutaway volumes dynamically without modifying the atlas mesh geometry. However, the image-space cutaway approach we use is not a general-purpose CSG solution. We cannot, for example, compute the union of a mesh and another primitive. But our approach does support the subtraction and intersection operations required to expose the interior of atlas meshes. Our technique fits naturally into a volume rendering framework where rays are formed in the fragment shader based on rendered proxy geometry.

We use a signed distance representation of the cutaway pieces since this permits a wide variety of shapes to be represented, and also allows those shapes to be deformed and combined (Reiner et al. 2011). Alternative stencil-based approaches represent cutout geometry as a mesh which requires an additional draw call for each CSG primitive, and makes the results subject to artifacts due to undertessellated cutout meshes. The undertesselation artifact is most apparent when using smooth CSG primitives like spheres and cylinders. Instead our cutaway geometry is represented mathematically in the fragment shader. A useful shape supported by our method is the slab bounded by a pair of infinite planes. The signed distance to a slab defined by two axis aligned planes is given by

$$d(x) = |x - x_{\text{offset}}| - w/2, \tag{3}$$

where w is the width of the slab, and the center of the slab is located at $x = x_{\text{offset}}$. As seen in a representative plot of this function in Figure (4) the function values are negative between the two planes, and positive otherwise. Distance to slabs with different positions and orientations is computed by inverse transforming point coordinates. For example, to transform the slab by the affine transformation matrix, M, we first compute $[x', y', z', w]^T = M^{-1}[x, y, z, w]^T$ then compute the distance $d(x')$.

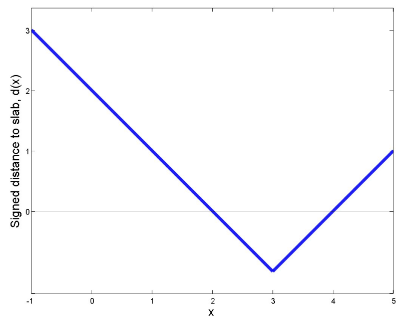

Figure 4. Slab signed distance function with $x_{\text{offset}} = 3$, and slab width $w = 2$. Positive distances are outside the slab defined by two parallel planes ($x = 2$ and $x = 4$). Negative distances are inside the slab.

We render meshes exploded into multiple slabs by using OpenGL's (Shreiner et al. 2013) geometry instancing functionality. This feature permits multiple copies of a mesh to be rendered with a single draw call which improves the efficiency of the application. Instanced rendering results in the value of the built-in vertex shader variable, `gl_InstanceID`, being incremented for each copy of the mesh. We use this value to advance the position of the slab for each instance, as shown in Figure (5). We also keep an array of slab transformation matrices in a uniform buffer object which allows us to transform the instances relative to each other when generating images of exploded slabs.

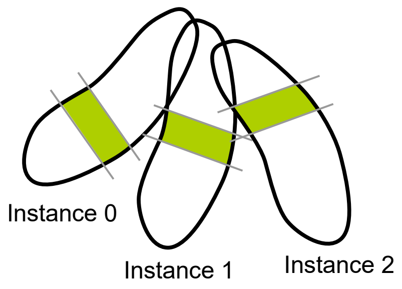

Figure 5. Each instance of the model is rendered with a different slab offset ($x_{\text{offset}} = x_0 + w * instanceID$).

Our rendering process starts in a similar way to many GPU-based volume rendering approaches (Engel et al. 2006)– by rendering proxy geometry back faces to floating-point textures to compute ray endpoints in the first rendering pass, then rendering the geometry front faces in the second pass to determine ray start points. Since each slab is raycast independently, each slab needs a ray start and endpoint for each fragment, so we render each instance in this pass to a separate layer of a layered texture. This prevent instances from interfering with each other when they overlap, as shown in Figure (5). A layered depth buffer is also required for correct hidden surface removal of each instance. To correctly handle the case of concave proxy geometry, we render the *furthest* ray endpoint in the first pass by clearing the depth buffer to 0.0 (rather than the default of 1.0) and changing the depth comparison function to keep fragments whose depth is *greater* than the framebuffer contents.

The front faces of the proxy geometry are rendered in the second pass to determine ray start points. In the fragment shader we create a ray which starts at the front face position, and ends at the back face position which is read from the layered back face texture for that instance. Three different configurations of the ray start, r_{start} and end point, r_{end}, relative to the slab distance function, d, are possible. All three cases are illustrated in Figure (6).

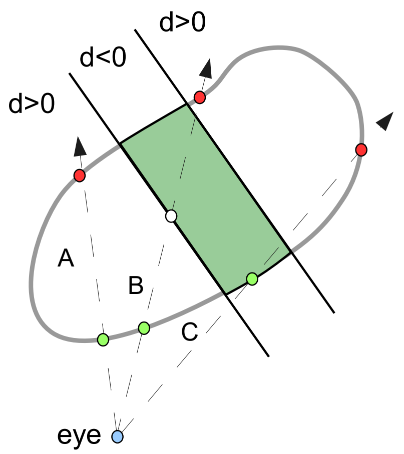

Figure 6. Hybrid surface rendering / raycasting process. Green dots denote front face ray intersections and red dots denote back face ray intersections. When tracing the ray from the front to back face there are three possible situations: (A) the ray does not intersect the slab, (B) the ray intersects front face, then the slab, (C) the ray intersects front face inside of slab.

In the first 2 cases the signed distance to the slab at the ray start point, $d(r_{start}) > 0$:

- (Case A) Skip empty space outside the slab by iteratively stepping by distance $d(r)$ along the ray (sphere tracing) (Reiner et al. 2011). If r_{end} is passed before the slab is entered there is no intersection and the fragment can be discarded.
- (Case B) At the location where the front face of the slab is intersected (white dot), read the fragment color from the 3D texture, and compute the new fragment depth from the intersection point.
- (Case C) If $d(r_{start}) < 0$ the point r_{start} is surface rendered by computing lighting for the mesh surface.

Intermediate rendering results are shown in Figure (7). Multiple instances of the postcentral gyrus mesh are exploded away from each other, and raycasting cuts away the parts of the mesh that do not intersect the slab.

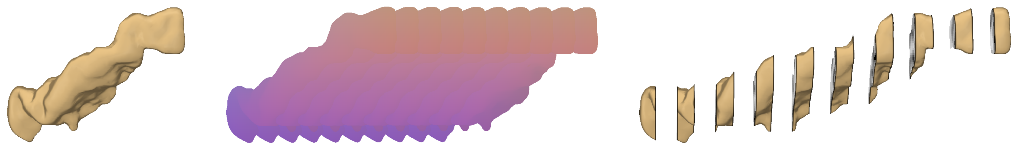

Figure 7. Postcentral gyrus (left), 10 exploded instances of the mesh front faces (center), raycasting results (right)

The method we describe for hybrid rendering of exploded meshes requires a constant maximum number of instances, n_{inst}, to be known before rendering. A detailed description of the initialization and hybrid rendering processes are presented in Algorithm (1).

Algorithm 1 Overview of initialization and rendering process

```
procedure INITIALIZATION
Create framebuffer object (FBO)
Attach layered color texture to FBO (number of layers = $n_{inst}$)
Attach layered depth texture to FBO (number of layers = $n_{inst}$)
Create shader A: surface rendering with lighting
Create shader B: render back face instanced positions to layered texture (gl_Layer =
gl_InstanceID)
Create shader C: hybrid renderer
end procedure

procedure RENDERING
Clear back buffer and depth buffer
Render unexploded meshes to back buffer using shader A
    for each exploded mesh do
        Bind FBO
        Clear depth to 0.0
        Set depth comparison mode to 'greater'
        Enable front face culling
        Update transformation matrices and upload to uniform buffer
        Enable shader B
        for each of the $n_{inst}$ instances of the mesh do
            Render back face position of instance $i$ to layer $i$ (layered rendering)
        end for
        Unbind FBO, draw to back buffer
        Set depth comparison mode to 'less'
        Enable back face culling
        Enable shader C
        for each of the $n_{inst}$ instances of the mesh do
            Compute front face position from rendered mesh
            Fetch back face position of instance $i$ from layer $i$ of texture
            Perform raycasting or lighting
        end for
    end for
end procedure
```

In the case of using the slab as a cutaway shape it might appear that this rendering technique could be optimized by eliminating many triangles from the proxy geometry with user-defined clipping planes. However the proxy meshes must be closed for raycasting to function correctly. A more sophisticated capping strategy could be pursued using the geometry shader, however handling arbitrary cutaway shapes would be more difficult. In this work we did not experience performance issues on modern desktop videocards. To better support laptop and tablets we will consider optimizations in future work.

An example of using multiple cutaway shapes when exploding meshes is shown in Figure (8). To generate this effect we alternated between slabs and cylinders when cutting each instance. The final result is similar to the "RibbonView" (McInerney and Crawford 2010) focus and context visualization technique. This figure also shows the edge darkening approach we use to emphasize the boundaries between surface-rendered and raycast surfaces. We modulate the surface color with a sigmoid function applied to the signed distance function. By using OpenGL shading language functions which compute screen-space derivatives we can impose a constant width of the lines for all view distances. We compute the color scaling value, s from the signed distance function as `float w = fwidth(dist); float s = 0.3 + 0.7*smoothstep(-2.0*w, -4.0*w, dist);`.

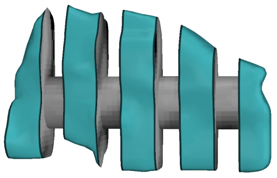
Figure 8. A variety of shapes my be incorporated into the hybrid rendering process. A visual result similar to RibbonView can be obtained by replacing the odd numbered slabs with cylinders.

We do not address the problem of volume rendering the slabs in this work. Modern A-buffer implementations, such as the per-pixel linked list Yang et al. (2010); Knowles et al. (2014) techniques used to implement order independent transparency can be applied in this case to maintain a list of multiple ray entry and exit points through multiple instances of (possibly) concave proxy geometry. In preliminary visual mockups this approach was found to yield visually complex and possibly confusing results, so we concentrate on opaque cutaways.

Lighting can be computed for the slab surfaces by using the slab normals, and for other shapes by estimating the gradient of the SDF. However, in the results presented in the next section we leave the slab unlit to avoid distorting the apparent intensity of the medical image.

Although our rendering technique was described in terms of OpenGL features, our method can also be easily implemented in the DirectX or Vulkan graphics APIs.

4. Results

Our atlas visualization technique was implemented in C++ and executed on 2 systems. System 1 was an Alienware X51 R2 workstation with a 3.6 GHz Intel Core i7-4790 CPU, 8 GB RAM, Nvidia GeForce GTX 960 with 1024 shader cores and 2 GB GDDR5 dedicated video RAM. System 2 was a Surface Pro 2 Tablet with a 1.6 GHz Intel Core i5-4200U CPU, 8 GB RAM, Intel HD Graphics 4400 with 20 shader cores and 1793 MB shared video RAM.

The results were obtained by loading the 116 meshes segmented from the AAL atlas label image - a total of 575, 460 triangles. Due to instancing, however, our technique may result in the processing and rasterization of up to $n_{inst} = 10$ times that number. The 3D image is a $181 \times 217 \times 181$ T1-weighted MRI image.

The left side of Figure (9) demonstrates the linear explosion method. At a value of $s = 1$ the meshes of the frontal lobe are exploded such that their bounding boxes do not overlap in the lateral direction. In the right side of Figure (9) the meshes of the temporal lobe have been exploded

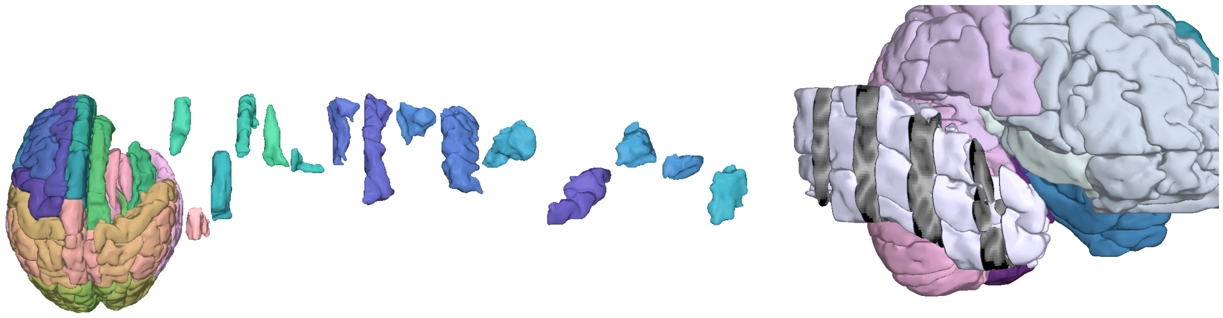
Figure 9. Exploded view of right frontal lobe of the human brain (left) and exploded meshes in the right temporal lobe rendered using our hybrid approach which displays a 3D image in context with segmented anatomical surfaces.

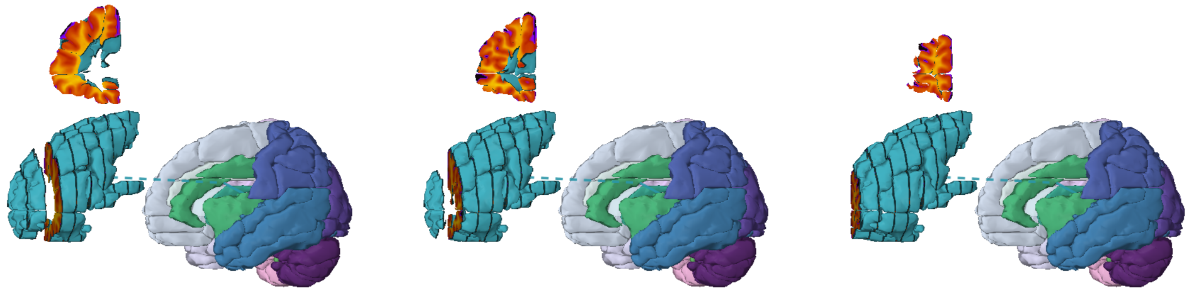

Figure 10. Exploded left frontal lobe. The selected slab is pulled out and rotated to face the camera. Results are shown for 3 different slab selections. A heatmap transfer function has been applied to the MRI image.

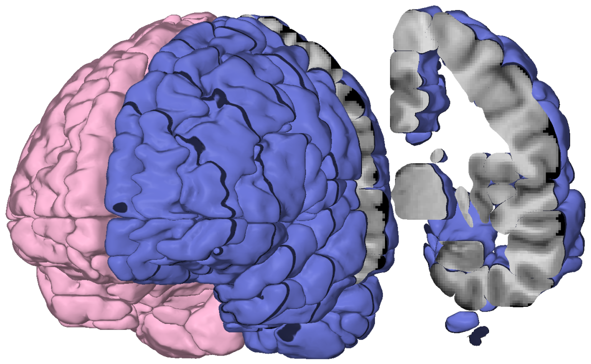

Figure 11. Exploded left hemisphere meshes. Note that the full brain is not segmented in this atlas, so the cross-sectional shape of the hemisphere has a complex topology which has been correctly displayed by our technique.

to show the MRI image. The user can select an individual slab for emphasis using the mouse wheel. The selected slab smoothly slides out and rotates to faces the viewer (see supplemental video and Figure (10)). Multiple meshes at any hierarchical level can be exploded. In Figure (11) the entire left hemisphere is exploded, and a single slab (user-selected with the mouse wheel) has been automatically translated so that the entire cross-section is visible. If multiple nodes are selected each one is transformed independently as shown in Figure (12). The mesh fanning transformation applied to the left and right hemispheres is shown in Figure (13), while the bilateral node containing the cerebellar vermis is linearly exploded. The centerlines are computed from the coordinates of the center of each bounding box by interpolating between an identity transformation and the current node transformation. **Figure (14) shows the region meshes of the brainstem (Midbrain, Pons and Medulla Oblongata) radially exploded (left) and with the meshes exploded into slabs (right). The slabs of exploded meshes are initially aligned with the axial, coronal or sagittal imaging planes, but the user can interactively rotate the slab orientation, as demonstrated in Figure (15) to display oblique slices through the image. In Figure (16) we show the effect of linearly exploding the slabs of a region. The increased spacing between them makes it easier to view the 3D image. Slabs may also be fanned about an axis as shown in Figure (17). In Figure (18) the entire right hemisphere has been selected by the user and exploded, then the postcentral gyrus and the middle frontal gyrus were unexploded to show their shapes in the context of the surrounding image intensities and the shapes of surrounding anatomical structures. Due to slight misalignment between the medical image and the atlas meshes black pixels near the boundary of the meshes may occasionally be observed.**

Timing results are presented in Table (1). The results will vary depending on how many atlas meshes are exploded and how large they are on the screen, so timing values for several different situations are presented. In all cases we used $n_{inst} = 10$ and the window size was 1280×720. The System 2* results were obtained by simplifying the meshes to a total of $77,248$ faces. This

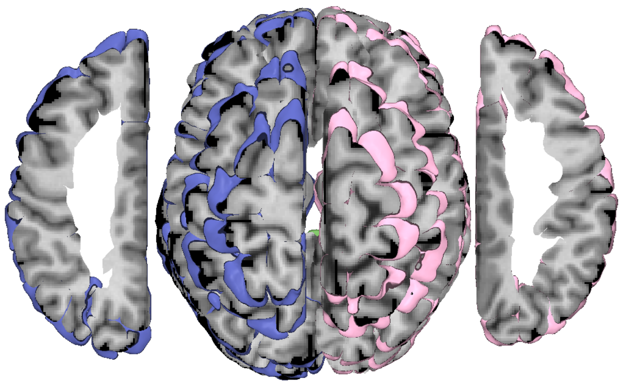

Figure 12. The selected slabs of each hemisphere explode away from the center of their parent node until they are completely visible.

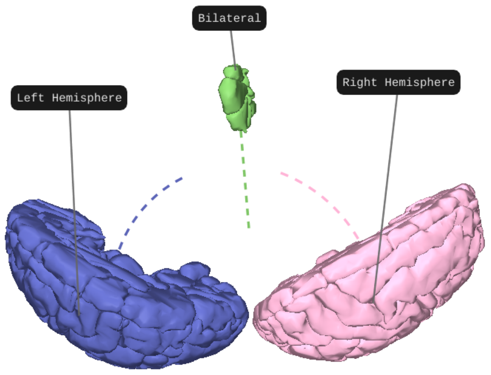

Figure 13. Fanned cerebral hemispheres shown with centerlines and annotations.

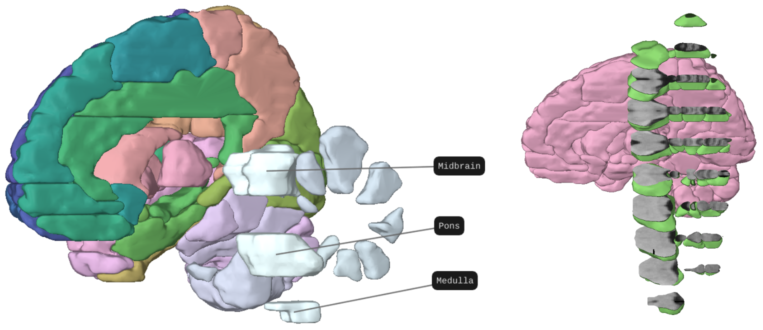

Figure 14. Brainstem exploded to show spatial relations between regions (left) and meshes exploded to image intensities (right).

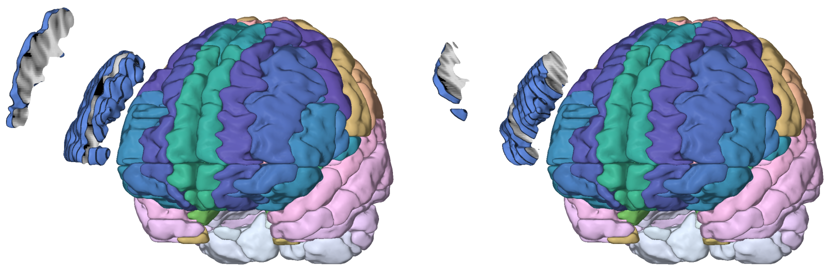

Figure 15. Slab orientation can be changed interactively to visualize oblique planes through the image.

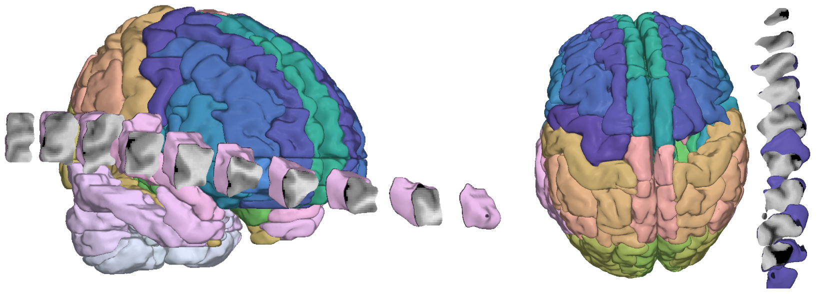
Figure 16. Slabs of exploded meshes can be spread apart to improve the visibility of the 3D image.

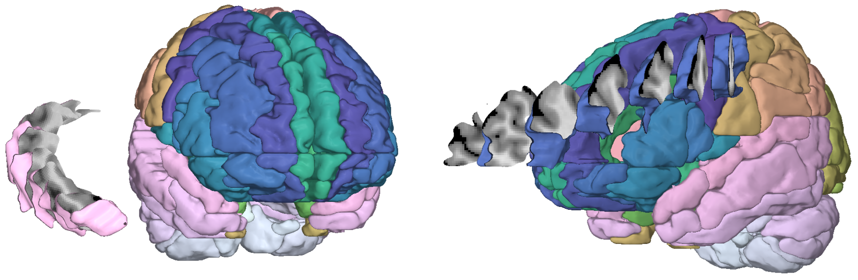
Figure 17. Slab fanning about edges of the mesh bounding box.

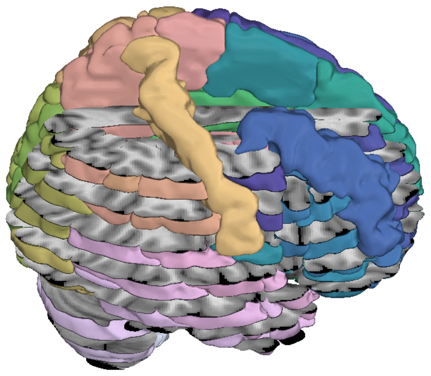
Figure 18. Regions can be interactively selected and then exploded and unexploded. Exploding an entire hemisphere into slabs, then unexploding a few select regions can provide contextual shape information about the unexploded regions, and insight into nearby image intensities.

Table 1. Timing results

	Render time (ms)		
	System 1	System 2	System 2*
0 exploded meshes	4.20	33.4	18.7
1 exploded region	5.43	37.5	21.2
1 exploded lobe	9.27	76.3	28.7
1 exploded hemisphere	24.2	177	74.1

permits our technique to run at an interactive frame rate with a minor loss of visual fidelity. We feel that this is an acceptable trade-off to get the technique running on portable hardware. For visual comparison both high and low resolution mesh results are shown in Figure (19).

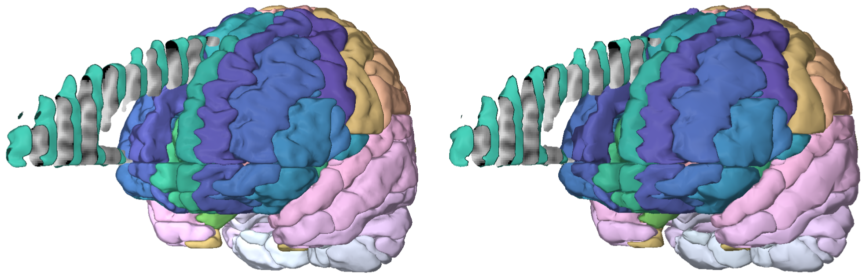

Figure 19. Visual comparison of high (left) and low (right) resolution meshes.

5. Conclusion and Future Work

In this work we have described an interactive atlas visualization system which is capable of creating exploded views based on the hierarchical structure of the data. Our hybrid rendering technique is capable of exploding anatomic meshes into slabs to reveal an underlying medical image. We detailed an OpenGL implementation of the rendering process, and presented results from the AAL neuroanatomical atlas. Our implementation is able to maintain interactive framerates, even on tablet hardware.

In future work we intend to explore the fusion of multiple imaging modalities with the atlas visualization. Functional imaging modalities, such as fMRI, PET and SPECT, can reveal valuable information about physiological processes such as blood oxygenation. Since functional images usually do not include anatomical detail it is necessary to visualize them in the context of a source of anatomical information. We feel that our visualization approach will allow users to explore functional imaging data and easily identify the anatomical sources of activity. Similarly, in visualizing structural connectivity from techniques such as diffusion tensor MRI (DT-MRI) it is important to have a visual reference for the grey matter structures which are connected by white matter fiber bundles.

In this study we did not encounter any significant performance issues, but in order to support large datasets, such as full body atlases, we will likely need to consider optimized rendering techniques and new interface approaches.

Acknowledgments

The GTX Titan X videocard used for this research was donated by the NVIDIA Corporation.

References

Ammann L, Génevaux O, Dischler JM. 2010. Hybrid rendering of dynamic heightfields using ray-casting and mesh rasterization. In: Proceedings of Graphics Interface 2010. Canadian Information Processing Society; p. 161–168.

Auzinger T, Mistelbauer G, Baclija I, Schernthaner R, Kochl A, Wimmer M, Groller ME, Bruckner S. 2013. Vessel visualization using curved surface reformation. IEEE Transactions on Visualization and Computer Graphics. 19(12):2858–2867.

Bruckner S, Groller ME. 2006. Exploded views for volume data. IEEE Transactions on Visualization and Computer Graphics. 12(5):1077–1084.

Bruckner S, Gröller ME, Mueller K, Preim B, Silver D. 2010. Illustrative Focus+Context Approaches in Interactive Volume Visualization. In: Hagen H, editor. Scientific visualization: Advanced concepts. Dagstuhl follow-ups; vol. 1. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik; p. 136–162.

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

Correa CD, Silver D, Chen M. 2010. Constrained illustrative volume deformation. Computers & Graphics. 34(4):370–377.

Custo A, Boas DA, Tsuzuki D, Dan I, Mesquita R, Fischl B, Grimson WEL, Wells W. 2010. Anatomical atlas-guided diffuse optical tomography of brain activation. NeuroImage. 49(1):561–567.

Díaz J, Monclús E, Navazo I, Vázquez P. 2012. Adaptive cross-sections of anatomical models. Computer Graphics Forum. 31(7):2155–2164.

Engel K, Hadwiger M, Kniss J, Rezk-Salama C, Weiskopf D. 2006. Real-time volume graphics. CRC Press.

Grimm S, Bruckner S, Kanitsar A, Gröller E. 2004. Flexible direct multi-volume rendering in dynamic scenes. In: Proceedings of the Vision, Modeling, and Visualization Conference 2004. p. 379–386.

Herrera Asteasu I. 2013. Volumetric visualization techniques of rigid and deformable models for surgery simulation [dissertation]. University of Navarra School of Engineering.

Höffler TN, Leutner D. 2007. Instructional animation versus static pictures: A meta-analysis. Learning and instruction. 17(6):722–738.

Islam S, Silver D, Chen M. 2007. Volume splitting and its applications. IEEE Transactions on Visualization and Computer Graphics. 13(2):193–203.

Johnson HJ, McCormick MM, Ibanez L. 2015. Template:the ITK software guide book 1: Introduction and development guidelines-volume 1.

Kikinis R, Shenton ME, Iosifescu DV, McCarley RW, Saiviroonporn P, Hokama HH, Robatino A, Metcalf D, Wible CG, Portas CM, et al. 1996. A digital brain atlas for surgical planning, model-driven segmentation, and teaching. IEEE Transactions on Visualization and Computer Graphics. 2(3):232–241.

Knowles P, Leach G, Zambetta F. 2014. Fast sorting for exact OIT of complex scenes. The Visual Computer. 30(6-8):603–613.

Li W, Agrawala M, Curless B, Salesin D. 2008. Automated generation of interactive 3D exploded view diagrams. ACM Transactions on Graphics. 27(3):101.

Li W, Ritter L, Agrawala M, Curless B, Salesin D. 2007. Interactive cutaway illustrations of complex 3D models. ACM Transactions on Graphics. 26(3):31.

Lötjönen JM, Wolz R, Koikkalainen JR, Thurfjell L, Waldemar G, Soininen H, Rueckert D, Initiative ADN, et al. 2010. Fast and robust multi-atlas segmentation of brain magnetic resonance images. Neuroimage. 49(3):2352–2365.

Marschner S, Shirley P. 2015. Fundamentals of computer graphics, 4th edition. CRC Press.

McGuffin MJ, Tancau L, Balakrishnan R. 2003. Using deformations for browsing volumetric data. In: IEEE Visualization Conference. IEEE; p. 401–408.

McInerney T, Crawford P. 2010. Ribbonview: interactive context-preserving cutaways of anatomical surface meshes. In: Advances in visual computing. Springer; p. 533–544.

McInerney T, Tran D. 2015. Aperio: A system for visualizing 3D anatomy data using virtual mechanical tools. In: Advances in visual computing. Springer; p. 797–808.

PMOD Technologes Ltd. 2015. Pmod version 3.7; [http://http://doc.pmod.com//].

Policarpo F, Oliveira MM, Comba JL. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In: Proceedings of the 2005 symposium on Interactive 3D graphics and games. ACM; p. 155–162.

Reiner T, Mückl G, Dachsbacher C. 2011. Interactive modeling of implicit surfaces using a direct visualization approach with signed distance functions. Computers & Graphics. 35(3):596–603.

Schroeder W, Martin K, Lorensen B. 2006. The visualization toolkit: an object oriented approach to 3D graphics. Kitware, Inc.

Shoemake K. 1985. Animating rotation with quaternion curves. In: ACM SIGGRAPH computer graphics. ACM; p. 245–254. vol. 19.

Shreiner D, Sellers G, Kessenich JM, Licea-Kane BM. 2013. OpenGL programming guide: The official guide to learning OpenGL, version 4.3. Addison-Wesley Professional.

Sigg S, Fuchs R, Carnecky R, Peikert R. 2012. Intelligent cutaway illustrations. In: IEEE Pacific Visualization Symposium. IEEE; p. 185–192.

Sitzmann T. 2011. A meta-analytic examination of the instructional effectiveness of computer-based simulation games. Personnel psychology. 64(2):489–528.

Trapp M, Döllner J. 2013. 2.5 D clip-surfaces for technical visualization. Journal of WSCG. 21(1):89–96.

Tzourio-Mazoyer N, Landeau B, Papathanassiou D, Crivello F, Etard O, Delcroix N, Mazoyer B, Joliot M. 2002. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. Neuroimage. 15(1):273–289.

Wang H, Suh JW, Das SR, Pluta JB, Craige C, Yushkevich PA. 2013. Multi-atlas segmentation with joint label fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence. 35(3):611–623.

Weiskopf D, Engel K, Ertl T. 2003. Interactive clipping techniques for texture-based volume visualization and volume shading. Visualization and Computer Graphics, IEEE Transactions on. 9(3):298–312.

Yang JC, Hensley J, Grün H, Thibieroz N. 2010. Real-time concurrent linked list construction on the GPU. Computer Graphics Forum. 29(4):1297–1304.

Yushkevich PA, Piven J, Cody Hazlett H, Gimpel Smith R, Ho S, Gee JC, Gerig G. 2006. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. Neuroimage. 31(3):1116–1128.