Surface Projection Method for Visualizing Volumetric Data

by

Peter Lincoln

A senior thesis submitted in partial fulfillment of

the requirements for the degree of

Bachelor of Science
With Departmental Honors

Computer Science and Engineering

University of Washington

June 2006

Presentation of work given on: ____

Thesis and presentation approved by: ____

Date: June 22, 2006

1 Project Introduction

Doctors, researchers, and other medical professionals collect and analyze many types of data. At the lowest level, this data may exist only as raw values, a structured mass of numbers. At a more usable level, the data can be converted to much more decipherable images. One goal of the Human Brain Project[1] is to find new ways of producing such images to aid in better understanding how the brain operates. While many tools exist to perform a large variety of imaging techniques, they sometimes lack the cohesiveness of a single unified interface.

A new program called MindSeer—developed by Eider Moore in the Structural Informatics Group (SIG) at the University of Washington—works to address this issue by providing a single interface for medical personnel to access their patients' medical data. One of the types of viewable data that MindSeer supports is volumetric data; however, the existing methods of visualizing that data in MindSeer is limited to only seeing a few 2D slices of the 3D data at a given time. The goal of this project was to explore, develop, and implement additional visualization methods for volumetric data within MindSeer. This paper discusses the implementation of one such visualization method, the surface projection method, and compares it to other existing methods.

2 Background

Before discussing how the surface projection method works, it is necessary to provide an introduction to types of medical data and existing means of displaying such data. Additionally presented is a discussion of the problems with these existing methods of data visualization.

2.1 fMRI Described

Magnetic Resonance Imaging (MRI) is an imaging technique most often used in medical contexts for creating high resolution images of the interior of the human body. It began as a means to produce a single image slice through the human body, but it has evolved into a volumetric technique.[2] In either the image or volumetric form, it can be used to analyze physical structure as well as neural activity within the brain.

Functional MRI (fMRI) is a variation on MRI specifically for localizing activity within the brain. It operates based on the knowledge that neural activity within the brain is accompanied by an increased blood flow in the same area.[3] This increased blood flow, however, is not accompanied by a corresponding increase in the consumption of oxygenated blood. As a result, the relative amount of deoxygenated hemoglobin (the protein in blood that carries oxygen to all cells in the body) decreases in comparison of oxygenated hemoglobin. Because deoxyhemoglobin is paramagnetic—it emits a magnetic field in response to an external magnetizing field—this proportional decrease in deoxyhemoglobin results in a detected alteration in the MRI signal. Repeating this scan over a stimulated brain versus an unstimulated brain can provide a basis for comparison. By mathematical analysis, the MRI signal from the stimulated brain can then be transformed into volume scalar field where each value represents deviations from the average activity recorded in the unstimulated brain. Following this computational analysis, it becomes necessary to visualize the data.

2.2 Standard Methods of Visualization & Their Problems

There are a couple of standard methods of visualizing fMRI and other types of volumetric data. The simplest such method consists of 2-D image slices through the volume, an example of which is visible below in Figure 2-1.

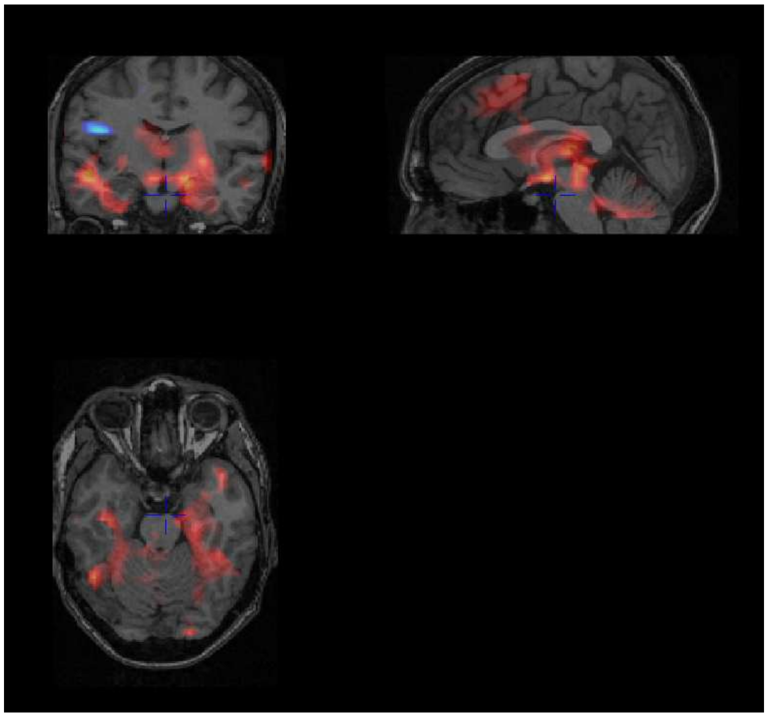


Figure 2-1 – A Sample Slice View of fMRI Data

The slice view type of data interface enables an analyst to cycle through each interpolated layer and hunt for active areas at interactive rates. If the desired point in the brain is known, then the analyst could choose to just examine that particular area.

Slice view, however, has a major limitation: only thin slices of the data can be seen at a given instant. It requires the analyst to visualize the whole 3D volume in their mind by examining each of the 2D slices. As a result, it becomes difficult, in a single glance, to observe the entire volume of brain activity.

Volume rendering is another imaging technique that attempts to alleviate the problem of being unable to see the entire brain at one glance. It operates by using specialized algorithms or hardware to render an entire volume-space by painting voxels (3-D pixels). It can generate translucent solids and inner surfaces quite visibly. However, on general purpose computer hardware, this rendering method operates at sub-interactive rates, making it very slow to use. On very expensive, specialized hardware it becomes possible to use volume rendering at interactive rates; however, the equipment costs are high and the portability of the machine is low. Centralizing computer of this type and providing network support could increase accessibility but decreases the interactivity rate.

3 The Surface Projection Method

Considering the issues present with image slices and volume rendering, a different rendering method must be considered: the surface projection method. The surface projection method combines the speed and interactivity of image slices with the quick interpretation that volume rendering provides. This section discusses the motivation behind this method and how this method is implemented within the MindSeer program.

3.1 Motivation & Design

The surface projection method operates by combining a volumetric dataset, such as an fMRI dataset, with a 3D triangle mesh, such as a brain surface model. To compute a color at a given point, an identical method to the image slices can be used: sample a data value at a point and map it to a color. This, in effect, creates a texture map over the surface representing data values at the surface. Modern consumer-level graphics hardware is highly optimized for handling triangles and textures; as a result, once the surface map is calculated, the analyst is free to manipulate the model at interactive rates.

While surface data may be useful in certain cases, many analysts are more interested in activity deeper in the brain. In order to support these and other cases, this implementation was designed to be extendable. In this way, data from deep inside the brain can be projected outwards or subsurface activity can be blended to be visible on the surface.

3.2 Infrastructure & Implementation

This implementation of the surface projection method is built upon the MindSeer platform. MindSeer is a Java application that enables medical personnel to readily access a patient's medical data. It supports a variety of formats of data, including volume data, surface models, images, text files, and simulation sites. Furthermore, MindSeer is designed using the Factory-style of programming, which means that it is quite extendable, thus allowing for the implementation of support for additional data types and additional viewing methods. Skandha, another SIG approach[4], provides similar support for the surface projection method; however, it was specifically designed for use in the client-server architecture and was not optimized to take advantage of the client's hardware. Using MindSeer to implement this visualization method enables the program to not only operate in a similar client-server mode but also in a standalone mode, which enables access to the user's optimized 3D graphics hardware.

This new visualization method utilizes the existing support in MindSeer for reading volumetric data files, such as fMRI, and 3D models. MindSeer also provides access to the Java3D library, which enables the program to take advantage of the client's 3D graphics card. Prior to this addition, users could view the volume data as a slice view as shown in Figure 2-1 and separately view the model data in a 3D format. The existing program also supported an inset slice view visible directly on the brain. While an inset slice view can help an analyst locate the image in relation to the brain by being able to see the slice within the brain, it still stuffers from the limited view inherent to the slice view method. The surface projection implementation uses the existing data structures and built-in 3D model viewer, but applies an additional coloring algorithm to override the existing coloring scheme with a calculated vertex-based coloring scheme based on the volumetric data. These implementation additions and utilizations are presented on page 4 in Figure 3-1.

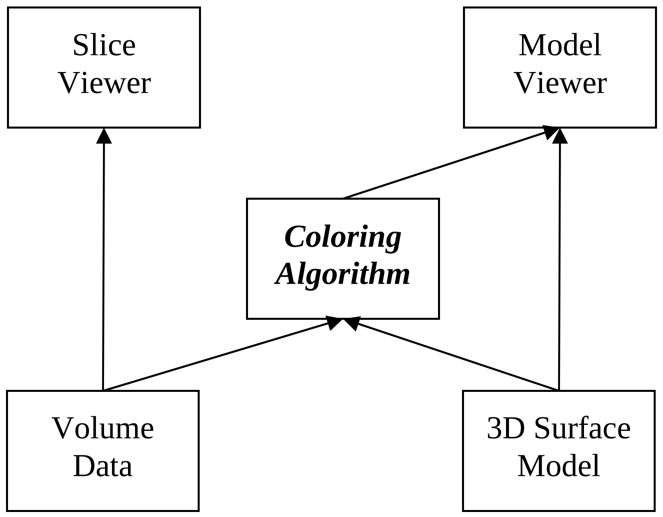


Figure 3-1 – High-Level Implementation Architecture Diagram

The coloring algorithm for the surface projection method is implemented in a pipelined format. This programming style was selected to best enable extensibility; in order to change any element of the pipeline, a programmer need only implement the interface used by a particular stage of pipeline. The pipeline consists of the following stages, illustrated below in Figure 3-2:

- Pre-filter the volume data using a filter kernel
- Sample the volume data to compute a value at each vertex of the model
- Perform aggregate summary calculations over each sampled dataset for later use
- If multiple volume datasets are present, unify them into a single dataset
- Perform color calculations, then create and display the textured model

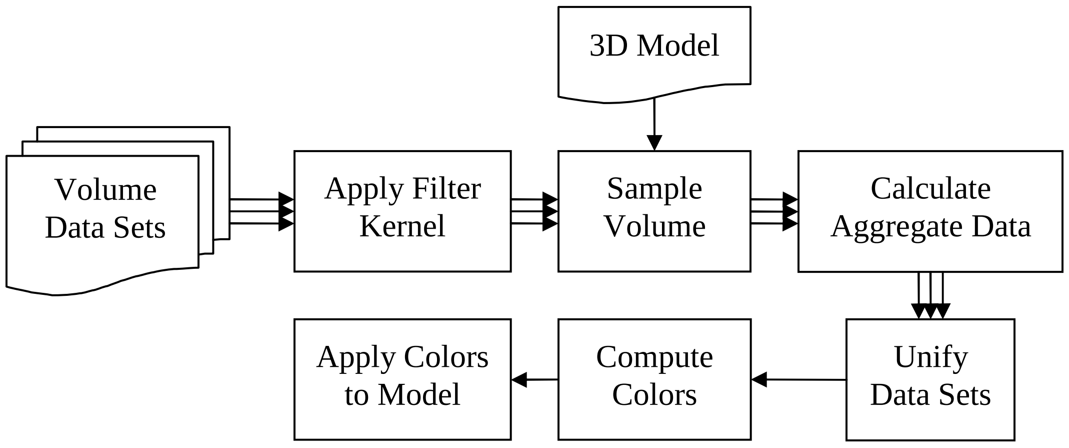


Figure 3-2 – Coloring Algorithm Pipeline

The pipeline begins with each of the datasets pre-loaded: the volume datasets to project and the model to project upon. Each dataset is inherently transformed to use common coordinate axes to eliminate the need for later conversions. The first processing step, which is optional, applies a 3D Gaussian filter kernel over the data. This step smoothes the raw volume data as well as bring data just below the surface to the surface. If this stage is enabled, then all further stages use the filtered data instead of the raw data.

Following the pre-filter stage, the sampling stage occurs. The simplest method samples the volume at each vertex. If a vertex does not lie on an exact corresponding data point in the volume dataset, then trilinear interpolation is used to derive a data value. Another implemented alternative is a radial average sampling method. In this calculation, a line is traversed from the center of the brain to each vertex. Data values are regularly sampled along this line and averaged. This enables deep brain activity to become projected upon the surface. In either of these methods, if multiple volume datasets are being applied to a single model, then this process is repeated for each dataset.

The next stage computes aggregate values over the entire set of sampled data. In order to provide optimal interactive rates while using the inset slice view, MindSeer divides the model into a collection of cutaway nodes; as a result, each node must store its own color texture. By the design of this pipeline, each node is unified and colorized separately. Certain combining and coloring algorithms, described below, may require aggregate values over the entire dataset to create accurate results. For this reason, this stage remains separate from others in the pipeline.

The next two stages combine a set of datasets to compute a single data value for each vertex and then map a single color to each vertex. Most useful combiners separate the two steps of combining the data values and calculating the color values; however, one method simply averages the resulting colors for each input data set. For this reason, these two stages are internally executed by the pipeline as a single stage. Some of the implemented combiners include an averaging data combiner, a normalized averaging data combiner, and AND-like operator. In cases where only a single input dataset is used, then typically the combination stage either performs no changes or normalizes the data values to the range zero to one or negative one to one; this is dependent on the intended result for multiple input datasets.

Vertex color calculation involves blending the selected data color scheme with the base model color. Several different schemes have been implemented and are presented below in Figure 3-3. In each case, the selected base color for the brain model is gray.

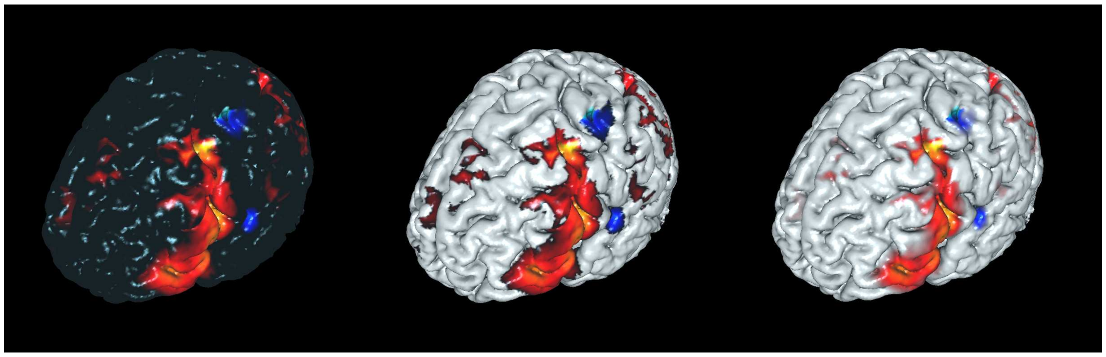
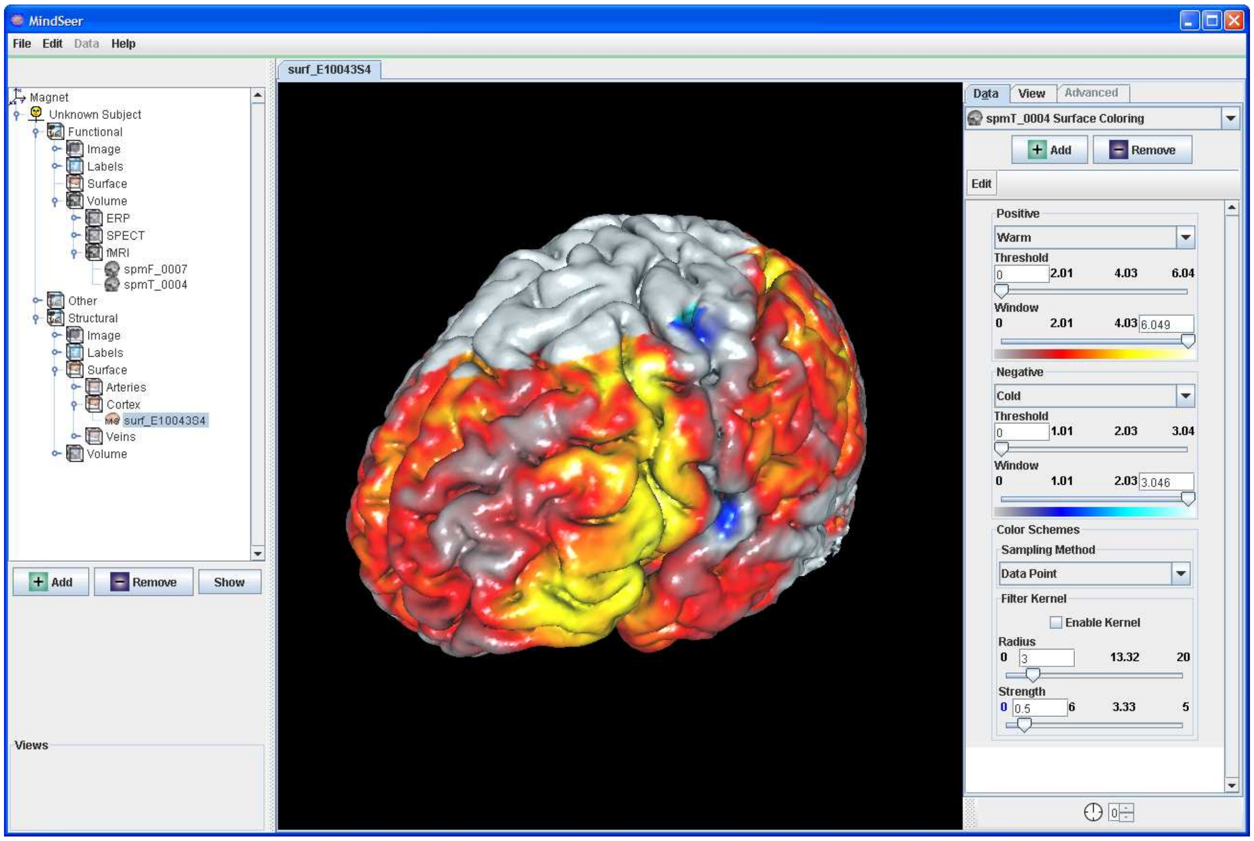


Figure 3-3 – A Set of Blending Examples; from left: Volume Color Only, Step Function, and Smooth Blending

Each of these three blending methods promotes a different mode of viewing. The Volume Color Only method yields the greatest contrast between the base model and the volume data. The Step Function provides both a means to see where data exists (black to red to yellow) and where it is not present (gray). The Smooth Blending operates similarly to the Step Function; however, where the volume color is dark (black to dark red), then it smoothly blends with the base color to provide a clean resulting image. All subsequent figures use the Smooth Blending scheme.

3.3 Features & Results

This section presents a discussion of the features and results. Figure 3-4, visible below, presents the basic user interface when a volume dataset has been applied to a surface model.



Positive
Warm
Threshold 0 2.01 4.03 6.04
Window 0 2.01 4.03 6.049
Negative
Cold
Threshold 0 1.01 2.03 3.04
Window 0 1.01 2.03 3.046
Color Schemes
Sampling Method
Data Point
Filter Kernel
Enable Kernel
Radius 0 3 13.32 20
Strength 0 0.5 6 3.33 5

Figure 3-4 – The Basic User Interface for the Surface Projection Method

The main view port at the center displays the colorized model. At the left is the standard set of MindSeer controls for loading data files. At the right, a control panel is available to adjust the rendering parameters for thresholding, selecting a color scheme, selecting a sampling method, and controlling the filter kernel. Additionally, the control panel provides a color legend to link colors to interpolated data values.

Thresholding enables the user to control which range of values are relevant. Typically and analyst is looking for the areas yielding the greatest activity. This can be achieved by increasing the values on the threshold sliders until only a relevant amount of information is visible. Figure 3-5 presents an example image using this feature; the source data for this image is the same as the data used in Figure 3-4.

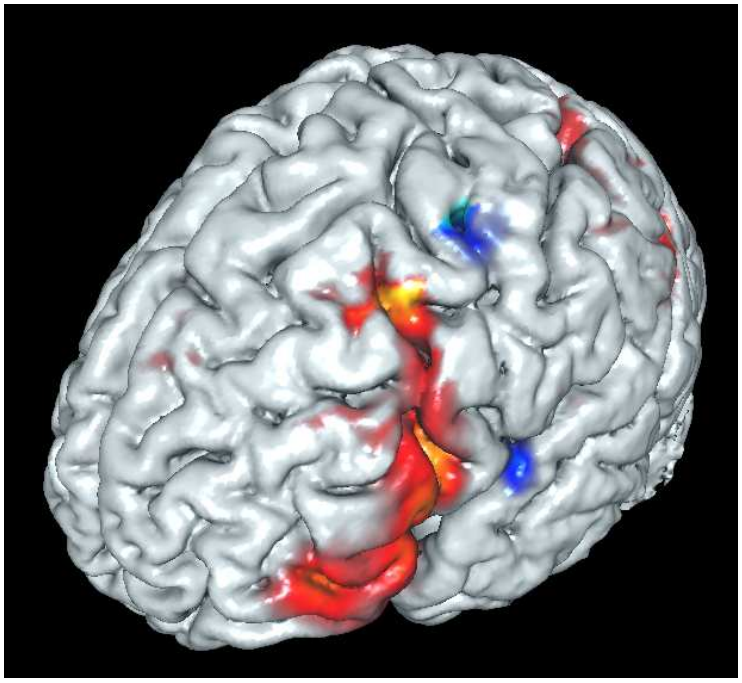
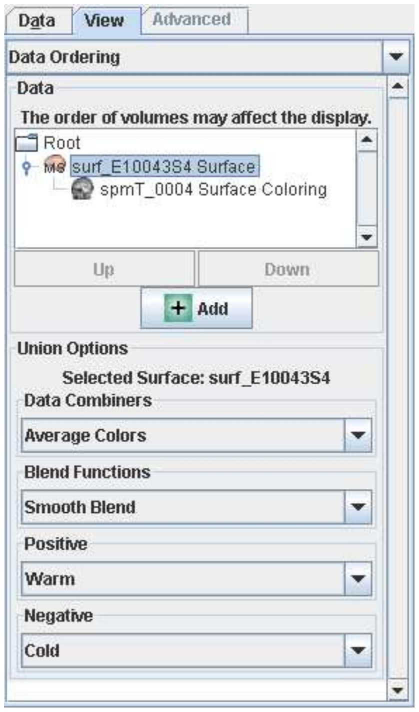


Figure 3-5 – An Example of Thresholding



Data Ordering

Data

The order of volumes may affect the display.

Root
surf_E10043S4 Surface
spmT_0004 Surface Coloring

Union Options

Selected Surface: surf_E10043S4

Data Combiners: Average Colors

Blend Functions: Smooth Blend

Positive: Warm

Negative: Cold

Figure 3-6 – The View Tab

Additional controls are available on the View tab of the control panel, as shown above in Figure 3-6. These controls provide access to the final stages of the pipeline: combining data values and blending colors. Through these controls, and analyst could highlight similarities and differences among multiple volume datasets for the same patient. A few examples of the implemented combiners are illustrated in Figure 3-7 and Figure 3-8.

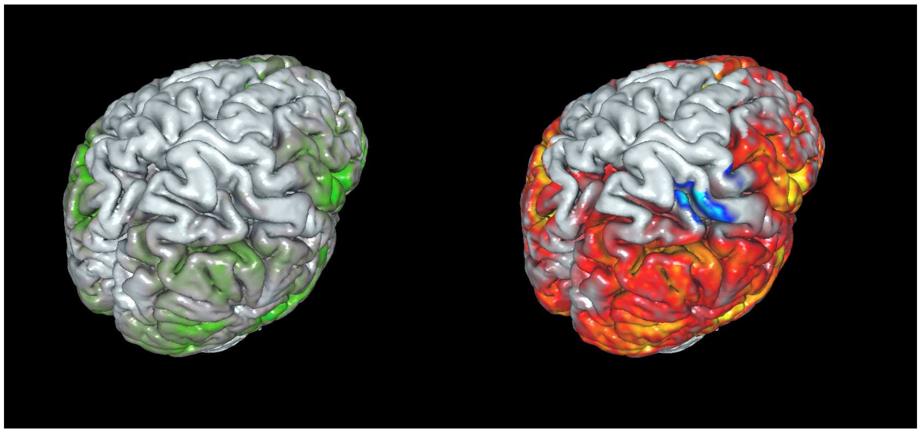


Figure 3-7 – Two Sample Input fMRI Volumes

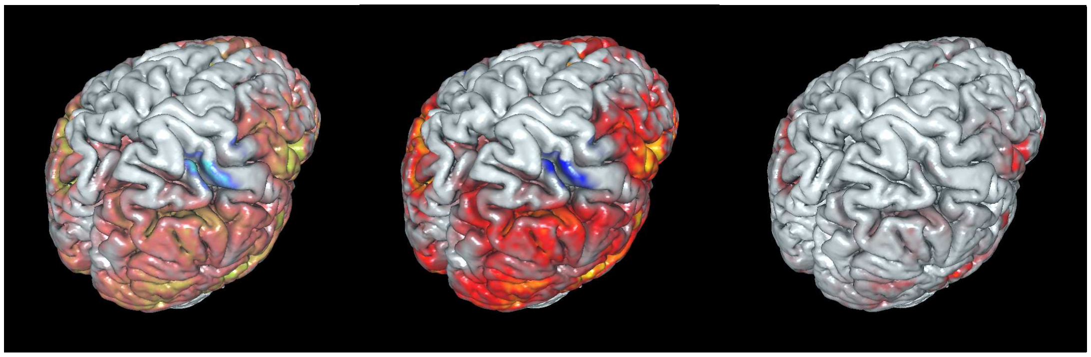


Figure 3-8 – Three Different Combiner Methods; from left: Average Colors, Scaled Average Data, and the AND Operator

The Average Colors method computes colors for each dataset and then performs an average of the colors to yield a resulting color value. The Scaled Average Data method normalizes each dataset to the range -1 to 1 and then performs an averaging operation over the data. In effect, this combiner averages the relative magnitudes of the data. The AND Operator first normalizes the absolute values of each dataset to the range 0 to 1. Each dataset is then element-wise multiplied to yield a resulting value. As a result, if most of the displayed datasets have a relatively high value at a given vertex, then the vertex is colored, else it is not colored.

3.4 Challenges

The Java language platform is designed to operate in a mostly identical fashion across all platforms; this design generally increases cross-platform compatibility. However, because each computer's operating system handling 3D graphics differently, the Java3D package attempts to abstract away the underlying native implementation of 3D graphics. Through this abstraction, Java3D created the most significant optimization challenge.

In Java3D, two basic methods for changing the colors of a triangle mesh: create new instances of the meshes and color arrays or modify the existing color arrays. At first glance the first option is more memory and garbage-collector intensive since it requires the repeated creation and destruction of large color arrays for each modified model. This would seem to imply that the first method would be slow. However, the interface for modifying the existing arrays was found to be much slower. Java3D provides an interface for modifying the color; however, as a result of the abstraction, color values can only be modified at a certain stage of the render loop. Additionally, due to the aforementioned structure of the cutaway nodes, each node would have to be modified individually. The problem arises in the way that Java3D permits these massive changes, only about one node in a set of a couple thousand nodes would be modified in each cycle of the render loop, thus requiring a wait time of around 30 seconds for all nodes to be updated, after all of the color calculations were completed. In contrast, the total replacement method only requires a couple seconds to replace the old colored mesh with the newly computed version. Due to the interactive benefits afforded by this method, the total replacement method is used in this implementation.

3.5 Extensibility

As mentioned previously, this implementation was designed to be quite extendable. This follows from the original design methodology of MindSeer: to be open for outside sources to easily write plug-ins to increase overall functionality. The surface projection implementation already supports a common format for volume data; however, only a few select types of 3D surface models are currently fully compatible. Should another sufficiently able user desire to apply the surface projection technique to another type of model, they would only need to implement the necessary methods to enable the existing interfaces to hook into that model type as well. Additionally, new samplers, combiners, and blenders can be added by simply implementing an interface and registering with the corresponding factory. In this way, another programmer could create a plug-in for the surface projection plug-in. A couple of possible extensions, similar to the AND operator, could highlight similarities between fMRI and stimulation sites or fMRI and magnetoencephalography data[5].

3.6 Contributions Summary

As this visualization method is implemented within the MindSeer framework, it is useful to highlight the specific accomplishments of this specific project. The main contribution consists of the new ability to combine 3D volumetric data with a 3D surface model. While the means of loading each data type into MindSeer previously existed, the pipelined process of coloring the surface according to the volume data, including each stage of the pipeline, is a direct result of this project. Additional reusable contributions include the general purpose 3D filter kernel and the color legend (present in Figure 3-4), which visually links colors to numerical data values. Future MindSeer contributors can make use of each of these new components.

4 Conclusions

Previously existing methods for viewing volumetric data have problems which the surface projection method solves. Slice views limit the user's view of the whole dataset, making it difficult to see certain correlations. On the other hand, the surface projection view enables the user to see the whole brain at once to see surface or subsurface data, thus making it easier for the analyst to analyze. Volumetric rendering has the problem of requiring special hardware or running too slow on consumer-level hardware. Surface projection solves this problem by rapidly pre-calculating a texture, which modern standard graphics hardware is strongly optimized to handle, thus enabling interactive redraw rates. In this way, the surface projection method solves the limitations of other existing methods of volumetric data visualization.

References

1. Brinkley, James F., Rosse, C. "Imaging and the Human Brain Project: A Review." Methods of Information in Medicine; 41: 245-260 <http://sigpubs.biostr.washington.edu/archive/00000123/>

2. Hornak, Joseph P. *The Basics of MRI*. E-Book. 2006. <http://www.cis.rit.edu/htbooks/mri/>

3. Columbia University. "fMRI – About Functional MRI (General)." <http://www.fmri.org/fmri.htm>

4. Poliakov, Andrew V., et al. "Server-based Approach to Web Visualization of Integrated Three-dimensional Brain Imaging Data." *Journal of the American Medical Informatics Association*. 2005; 12(2): 140:151 <http://sigpubs.biostr.washington.edu/archive/00000170/>

5. Badea, A., et al. "Surface visualization of electromagnetic brain activity." *Journal of Neuroscience Methods*. 2003; 127(2): 137-147. doi:10.1016/S0165-0270(03)00100-6 <http://www.sciencedirect.com/science/article/B6T04-48XSJGY-2/2/a364de4393c7dbb4c43f529a85890444>