Application of Information Technology ■

Server-based Approach to Web Visualization of Integrated Three-dimensional Brain Imaging Data

ANDREW V. POLIAKOV, PHD, EVAN ALBRIGHT, MS, KEVIN P. HINSHAW, PHD, DAVID P. CORINA, PHD, GEORGE OJEMANN, MD, RICHARD F. MARTIN, PHD, JAMES F. BRINKLEY, MD, PHD

Abstract The authors describe a client-server approach to three-dimensional (3-D) visualization of neuroimaging data, which enables researchers to visualize, manipulate, and analyze large brain imaging datasets over the Internet. All computationally intensive tasks are done by a graphics server that loads and processes image volumes and 3-D models, renders 3-D scenes, and sends the renderings back to the client. The authors discuss the system architecture and implementation and give several examples of client applications that allow visualization and analysis of integrated language map data from single and multiple patients.

■ **J Am Med Inform Assoc.** 2005;12:140–151. DOI 10.1197/jamia.M1671.

Three-dimensional (3-D) imaging has become an important and integral part of biomedicine due to the three-dimensional nature of the body and to new technologies that make 3-D imaging possible. Nowhere is this importance more evident than in neuroscience, which now has the ability to image the thinking brain. In these kinds of functional imaging studies, structural (anatomic) image volumes are collected using magnetic resonance imaging (MRI) or computed tomography (CT). The structural image volumes are processed to reduce artifacts and noise, to spatially co-register the subject's data with standard template datasets and brain atlases, to segment the brain into tissue types, and to extract surfaces and 3-D models.[1] The image data may be further processed using more specific and/or advanced techniques such as nonlinear volumetric deformations[2,3] or surface-based deformable maps.[4,5] Functional data from various modalities, including functional magnetic resonance (fMRI), electroencephalography/magnetoencephalography (EEG/MEG), and many others are then mapped onto the volume- and surface-based structural data to obtain location-specific 3-D images depicting the functional activity of the brain.

Affiliations of the authors: Structural Informatics Group, Department of Biological Structure (AVP, EA, KPH, RFM, JFB); Department of Psychology, (DPC); Department of Neurological Surgery, University of Washington, Seattle, WA (GO).

This work was funded by Human Brain Project grant DC02310.

The Skandha4 toolkit was originally designed and implemented by Jeff Prothero while he was a programmer in the Structural Informatics Group from 1985 to 1997. Since then, several developers, including the authors of this report, have contributed special purpose modules and applications based on this toolkit.

Correspondence and reprints: Andrew V. Poliakov, PhD, Structural Informatics Group, Department of Biological Structure, University of Washington, Seattle, WA 98195; e-mail: <poliakov@u.washington.edu>.

Received for publication: 08/12/04; accepted for publication: 10/16/04.

Visualization of these integrated functional and structural image data requires advanced techniques, including surface and volume rendering of complex multidimensional 3-D scenes. However, these techniques are not yet widely available to the average researcher, as they in many cases present a challenge for an average user workstation due to limited disk space, memory, and computational power. Even with high-performance workstations, the application programs needed to display and render 3-D datasets may not be easily installed or run. For special-purpose 3-D image data, including brain imaging, the difficulties are even greater because much of the visualization software is only available in research laboratories and is not yet "productized," i.e., it has not been well tested and may be changing rapidly while technical support may be limited.

One way to address these issues is interactive rendering on the client workstation using virtual reality modeling language (VRML) viewers or other Web browser plug-ins, as for example, several experimental systems that have been developed for teleradiology.[6–8] Although these tools are becoming more widely available, they are not yet standardized or supported on all platforms, and they still require installation of nonstandard plug-ins. Moreover, integrated 3-D medical imaging data often require large volumes of data to be sent over the Internet and loaded into a 3-D viewer, which is still impractical at most network speeds.

In this report, we describe our experience with an alternative approach to stand-alone or client-side rendering, which has been in operation in the University of Washington Structural Informatics Group (SIG) for the past several years.[9] In this approach, an existing stand-alone visualization tool is modified to become a remote visualization server, which then sends only relatively small 2-D image snapshots of the rendered scene to a lightweight client such as a standard Web browser or 2-D Java applet. Although such an approach does not permit real-time interaction with the scene (at least until Internet 2 becomes widely available), it does provide access to advanced 3-D visualization capabilities for any desktop workstation that is connected to the Internet. As we illustrate in this report, such access can be highly valuable for the type of collaboration involved in our local brain imaging research. A similar approach applied to other stand-alone visualization applications should also be a useful addition to teleradiology and online medical record systems.

Providing this kind of Web-based access to complex 3-D visualization raises challenges that are not present in widely popular common gateway interface- (CGI-) or java server pages–based (JSP-based) Web systems, primarily because of the inherent complexities of large-scale 3-D datasets. In the remainder of this report we describe our approach to meeting these challenges. We first describe some existing remote visualization systems. Based on this survey and our own experience we formulate design goals and describe the architecture of our system as well as current status and usage for neuroscience research. We conclude with our assessment of how well our system meets the goals and describe some research directions for more completely meeting these goals.

Background

The Human Brain Project (HBP),[10] a national effort to organize and share data about the human brain, provides many examples of systems for integrating and visualizing 3-D brain imaging data.[1] As in other 3-D medical imaging applications, most of these systems are implemented in stand-alone software that must be downloaded and installed on the local user workstation. However, these and other research efforts are increasingly providing some form of Web-based access to their existing stand-alone applications. For example, BrainWeb[11] and ICBM View[12] provide Web-based slice viewers that allow a user to navigate through three orthogonal slices of a 3-D image volume. Similar slice viewers have been developed for the Visible Human[13] and various interactive brain atlases.[14–16] Several sites also provide some sort of Web-based visualization of 3-D models. Examples include the SPL Anatomy Browser,[17] Cortical Flat Mapping,[18] and the UCLA Laboratory of NeuroImaging.[19] However, most of these applications resort to pregenerated Web pages and noninteractive or quasi-interactive techniques like "canned" 3-D scenes and movies.

Client-server systems have also been developed for 3-D visualization in teleradiology, although most of these systems have been experimental. Examples include client-side VRML renderers[6–8] and custom client-server applications that are optimized for rendering speed.[20–22] Although the performance of these systems can be excellent for applications such as surgical planning,[22] the specialized hardware and software required currently limit their widespread use.

Some applications we are aware of that have a similar approach to our own are Webcortex,[23] a Web front end to a precursor of the FreeSurfer brain surface reconfiguration program[24]; WebCaret,[25] a Web front end to a different surface reconfiguration program called Caret[26] and a general purpose Web front end to SGI's OpenInventor; and Cosmo3D applications.[27] In all these applications, the server, which is a modified version of the corresponding stand-alone application, renders a 3-D scene, sends a 2-D snapshot of the rendered scene to a standard Web browser client, and manipulates (rotates, moves) the scene in response to mouse clicks initiated at the browser. The primary difference among these applications is that the underlying server application is different; therefore, the corresponding Web application has different functionality, and the client-server communication protocol takes a different form. In particular, the system described in this report uses a novel Lisp-based application that is highly suited to the server-based approach, providing a network API that permits multiple client applications to be easily written and integrated with database management systems. In addition, the system has been in active use for more than three years by collaborating neuroscientists.

Design Objectives

The ideal 3-D visualization system would be as integral to client-side Web browsers as 2-D image rendering is to current Web browsers. Internet connections would have sufficient bandwidth to deliver complex scenes to the desktop, where they would be rendered at interactive speeds using standardized software. Such an integrated system is not yet widely available because of low-speed Internet connections, inadequate client-side hardware, and nonstandard visualization software. Although advances in computer hardware, Internet 2, and standards such as X3d[28] promise to alleviate these bottlenecks, the system we describe is designed to be applicable to the hardware and software environment currently available to most neuroscience researchers. Thus, the system should meet the following requirements:

1. Provide 3-D visualization of integrated brain image data.
2. Provide the ability to interact with the 3-D visualization to rotate, move, or zoom the camera; to add or delete structural and functional data sets; and to render the scene in different styles.
3. Work for a broad range of users, even those with relatively low-powered desktop hardware.
4. Require a relatively low-bandwidth Internet connection.
5. Require little or no specialized software installation at the client workstation other than a standard Web browser and the lowest common denominator version of Java.
6. Provide the ability to be integrated in larger information systems.
7. Provide patient confidentiality since our application, as well as many other potential applications of this approach, deals with patient imaging data.
8. Provide fast response time.
9. Provide, at the server end, the ability to accept multiple 3-D images and derived 3-D model formats.
10. Provide, at the server end, the ability to run on multiple hardware platforms.

In the following sections we demonstrate that the system we have designed satisfies requirements 1 through 7, and, to a limited extent, requirements 8 through 10. Because of this, the system has proven very useful in our local neuroscience research. As noted in the discussion, our current work is aimed at more completely satisfying the remaining requirements.

System Description

Our basic approach is to modify an existing stand-alone application so that it can function as a visualization server. We therefore first describe the stand-alone application before describing how we converted it to a visualization server.

Stand-alone Application

The primary driving neuroscience application for the applications we have developed is processing, integrating, and visualizing functional brain imaging data to understand how language is organized in the brain.[29] A premise of this and related brain imaging work is that because each imaging modality provides different information, the integration of multiple modalities will provide more insight into the underlying biological processes than could a single modality alone.

In our studies, the primary data are obtained from patients undergoing neurosurgery for intractable epilepsy. Before surgery, structural MRI and fMRI image volumes are collected. The fMRI image volumes depict areas of the brain that are activated during language tasks. During surgery a procedure called *cortical stimulation mapping* (CSM) is used to locate areas of language on the cortex that need to be avoided during surgery.[30] A primary goal is to integrate and visualize these functional language data (CSM, fMRI) in terms of the underlying anatomy of the brain obtained from structural MRI, since anatomy (or structure) provides a common substrate on which to map function. The epilepsy cases present a unique opportunity because the CSM data provide a "gold standard" against which fMRI and other noninvasive methods can be compared.

Like many other HBP efforts, the software tools we have developed run on high-end Linux or Silicon Graphics computers. These tools allow (1) reconstruction of 3-D models of the cortical surface, veins, and arteries from MRI scans; (2) reconstruction of the 3-D location of the CSM sites with respect to the 3-D models; (3) integration of fMRI image volumes with the 3-D structural models; (4) quantitative correlation between fMRI and CSM language areas; and (5) visualization of the integrated data. These capabilities are implemented in two stand-alone applications, the Visual Brain Mapper[31] and the Brain Visualizer.[32] These stand-alone applications are limited to computer facilities within the University of Washington Structural Informatics Group. Downloading and installing the applications on a local desktop requires high-performance hardware and massive data storage measured in hundreds of gigabytes and involves maintaining and updating the software. These reasons, plus the fact that the tools were designed originally with the ability to function as remote servers, created an opportunity to develop a Web-based approach.

Overview of the Client-server System

Figure 1 shows the architecture of our server-side visualization approach. The Graphics Server, which is the key component of this architecture, awaits commands over a dedicated port that is accessed by Java applet or Common Gateway Interface (CGI)[33] client programs. In response to commands that make up the server API, the server loads MRI and fMRI image volumes, 3-D surface models, and CSM sites; renders the resulting 3-D scenes; and saves the rendered images as 2-D snapshots that are accessed by the client over the Web. User interaction at the Web browser or Java applet causes additional commands to be sent to the server, which processes those commands, renders the image, and returns a new snapshot to the client. The server can also be used to perform tasks that do not necessarily result in a rendered 3-D scene, e.g., for computing various analyses of the integrated data.

The Java and CGI clients can run as stand-alone Web applications or can be a part of more complex applications. In particular, we have started to integrate Web-based visualization into our Web-based Language Map Experiment Management System (EMS),[34,35] which is used to manage metadata about the 3-D volumes and models (e.g., file locations, relationships), as well as other aspects of the study, such as patient demographics, transcripts of experimental trials, and surgical photographs.

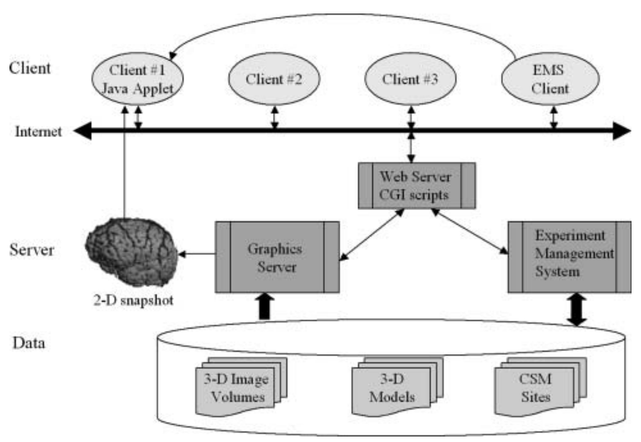


Figure 1. System Architecture. The Graphics Server, based on our in-house Skandha4 graphics toolkit, is a key component of the server-based approach. The server awaits ASCII (Lisp) commands over a dedicated port that can be accessed by a Web-based applet or CGI client programs (for example, clients 1 through 3 as described in the text). In response to these commands the server loads 3-D image volumes, 3-D models, and CSM sites; renders a scene; and sends a 2-D snapshot back to the client. The file locations of the 3-D data, as well as additional metadata, are maintained in an EMS that is accessed by the Graphics Server and by a CGI-based EMS client. The EMS client can invoke the Java applet (client #1) from a dynamically generated Web page that displays experimental data from a selected patient.

Server Side

The brain map Graphics Server is a modified version of our stand-alone Brain Map Visualizer and Visual Brain Mapper applications, which, in turn, are implemented in our Skandha4 graphics toolkit.[36,37] Skandha4 combines a subset of Common Lisp—useful for fast interactive programming and prototyping—with the ability to add precompiled C-based primitive functions that significantly accelerate computationally demanding routines. One class of such functions is a set of drivers for IRIX GL and OpenGL, which are libraries that provide access to 3-D rendering, either in hardware or software. Although other languages such as perl implement similar language-specific wrappers over GL, and at least one product, AutoCad, provides a Lisp-based scripting language for graphics (AutoLisp),[38] Skandha4 provides extensive additional C-based functionality, as well as the ability to easily run in server mode.

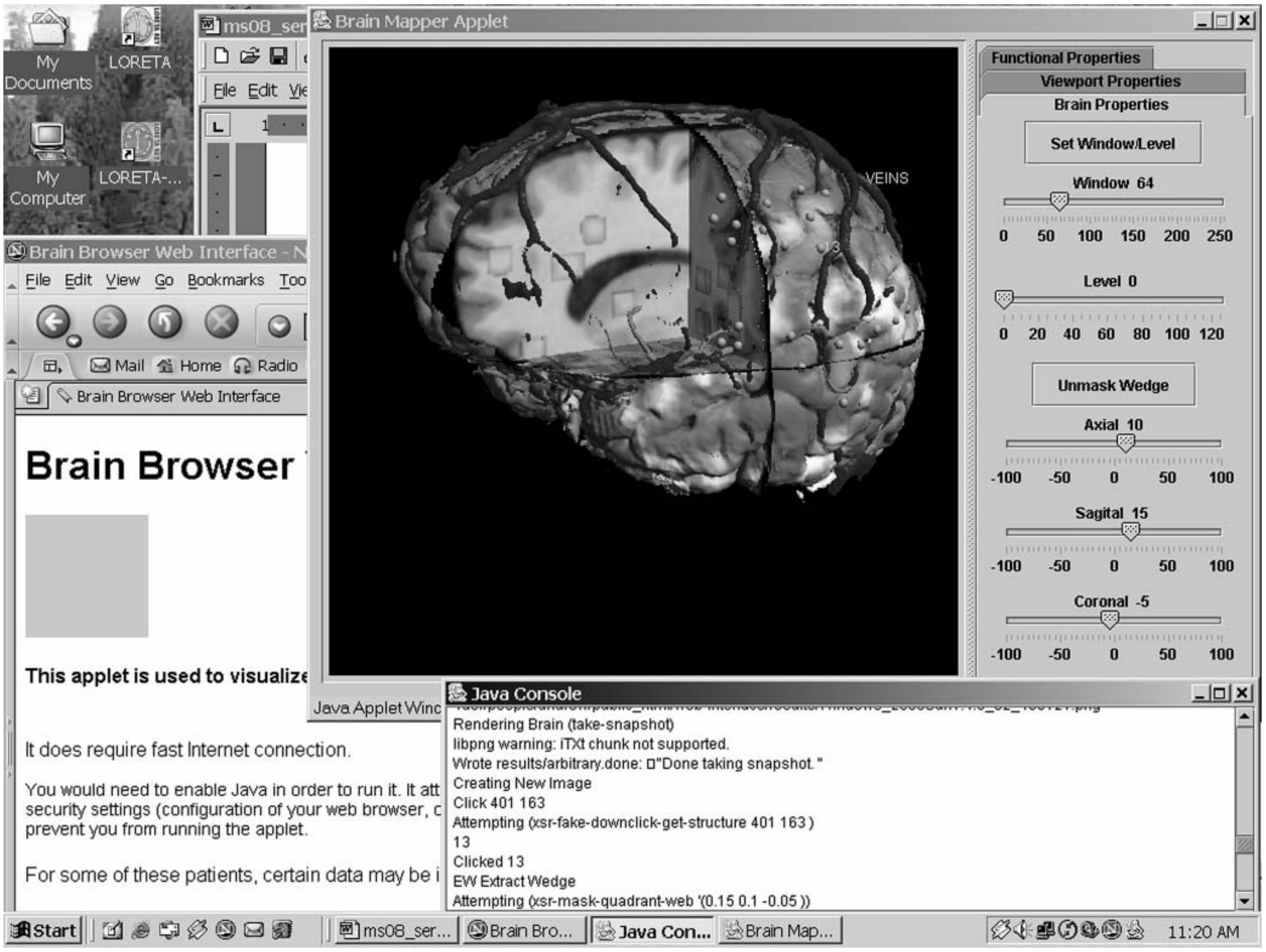
For example, in Skandha4, graphical objects are defined at the Lisp level as lists of "graphical relations," where a graphical relation is itself a list of parallel arrays of binary data defined at the C level, each of which defines one dimension of the object. In a 3-D surface mesh the vertices are described by a graphical relation consisting of three parallel arrays defining the X,Y,Z coordinates, and the triangular facets are described by a graphical relation consisting of three parallel arrays specifying the indices into the vertex arrays that define each triangle. The advantage of this uniform representation is that a single C-based drawing function may be written to iterate through an object, changing the drawing style depending on the presence or absence of particular graphical relations in the list defining the object. For example, a bit array may be added in parallel to the facet graphical relation to specify which triangles should be set to invisible. When the drawing function sees this bit array it only draws triangles whose bits are set. If the array is not present it draws all the triangles. This capability is used in Figure 2 to set to invisible all triangles defining the surface of the brain intersected by two cutting planes. The only changes to the code are the addition of the bit array defined by the position of the cutting planes.

This basic functionality is augmented by plug-in modules containing C-based Lisp primitives that may be included or excluded during compilation. We have developed several such modules for processing MRI data; reconstructing 3-D models of the cortical surface, veins, and arteries; importing and exporting MRI volumes, images, and 3-D models in various file formats such as GE Genesis format, Analyze 7.5, and MINC image volumes, tiff, png, and VRML; and visualizing and analyzing integrated data.

Skandha4 also includes a graphical user interface (GUI) module that provides a local event-driven interface to the Lisp functions. By separating the processing functions from the GUI it is possible to create different front-ends for the same functionality. In particular, server mode is implemented by bypassing the GUI functions and calling the processing functions directly.

In server mode Skandha4 implements a preforking server model. That is, on initializing and (optionally) loading data and/or code from a specified file, the parent process forks several child processes in advance, so that each of them can handle a new connection from a client. The server then binds to a specified port, and a connecting client need only implement a simple ASCII networking protocol to access all the functionality of the server. When the client disconnects, the child process terminates, and the parent process forks a new child, which is then available to serve a new connection from another client.

In server mode, scene rendering is typically performed off screen using the Mesa software implementation of OpenGL.[39] In this mode the scene as rendered in the image



Brain Browser

This applet is used to visualize

It does require fast Internet connection.

You would need to enable Java in order to run it. It att
security settings (configuration of your web browser, c
prevent you from running the applet.

For some of these patients, certain data may be i

```
Rendering Brain (take-snapshot)
libpng warning: iTXt chunk not supported.
Wrote results/arbitrary.done: □"Done taking snapshot."
Creating New Image
Click 401 163
Attempting (xsr-fake-downclick-get-structure 401 163 )
13
Clicked 13
EW Extract Wedge
Attempting (xsr-mask-quadrant-web '(0.15 0.1 -0.05 ))
```

Figure 2. Screen shot of the desktop with the Brain Visualization applet running. The Web page that launches the applet is in the lower left, the applet itself is in the upper right, and the Java console is in the lower right. The user can query structure labels by clicking on them, i.e., VEINS and CSM site 13 are shown in the figure. The Brain Properties panel shown allows the user to control the appearance of the MRI slices using Window and Level sliders and to show the cut-away view using the Axial, Sagittal, and Coronal sliders.

buffer does not appear on screen, but instead is saved in an image file which can then be returned to the client program.

All these features make Skandha4 well suited as a backend graphics server to support interactive visualization on the Web. In addition to the Web interfaces described in this report, Skandha4 in server mode is also used for interactive knowledge-based 3-D scene generation by a Web-based application that dynamically constructs 3-D anatomical scenes from model primitives.[40,41] The server functionality for the 3-D scene generator is achieved simply by starting the server process with a different set of Lisp files than those used for the brain map Graphics Server. More information about Skandha4, including downloadable source code, can be found on the project Web site.[42]

Server API

Any client connecting to the brain map Graphics Server on the specified port has access to the entire augmented Lisp programming environment of Skandha4. However, implementing a Web-based visualization client typically requires sending only a limited number of commands corresponding to those sent by the GUI controls in the standalone Brain Visualizer.[32] For example, the client that we developed for visualizing an individual patient's data (client Example 1 described below and shown in Figures 2 and 3) only utilizes commands from a small set of wrapper functions. In general the name of each Lisp function in a module is prefixed by the name of the module that contains it.

Selection and Loading of Data

`(xmri-subjects)`—returns a list of identifiers of all subjects available in the EMS

This list is read on startup from a file that is generated by the EMS and that contains an identifier for each subject as well as the file paths of image, graphics, and map files associated with that subject.

`(xsr-select-subject identifier)`—select the current subject

`(xsr-load-dataset)`—load structural MRI image volumes and derived surfaces as well as the list of functional volumes available for the selected subject. The specific filenames for these objects are specified in the file generated by the EMS.

Brain Properties

`(xsr-mask-quadrant-web '(sagittal axial coronal))`— create a cut-away view by masking a portion of a 3-D brain surface model, texture mapping the associated MR image volume on the sides of the masked quadrant. The masked triangular facets are set to be invisible by the addition of a bit array to the graphical relation defining the brain surface.

`(xsr-unmask-web)`—restore the unmasked volume.

`(xsr-change-window-level window level)`— MRI visualization parameters.

Viewport Properties

`(xsr-set-camera distance azimuth altitude)` — set camera location and viewing angles.

`(take-snapshot)`—render the scene and save a snapshot as a 2-D image.

Functional Properties

`(xsr-add-fmri identifier)`—add an available fMRI volume to the loaded dataset.

`(xsr-change-fmri-threshold-gain threshold gain)` —set visualization parameters for functional data.

`(xsr-load-map-sites)`—add CSM sites for the loaded dataset. CSM sites are given by a file defining a set of points in the coordinate system of the 3-D surface model, each of which may also have a label defining it to be a location on the brain surface where language function was found.

`(xsr-remove-map-sites)`—remove CSM sites

Image (click operation)

`(xsr-fake-downclick-get-structure x y)`—perform a 3-D pick operation at the *(x, y)* screen coordinate specified by a mouse click, and query the label assigned to the picked 3-D structure. The label is either the anatomical name of a structural component or the number of a stimulation site.

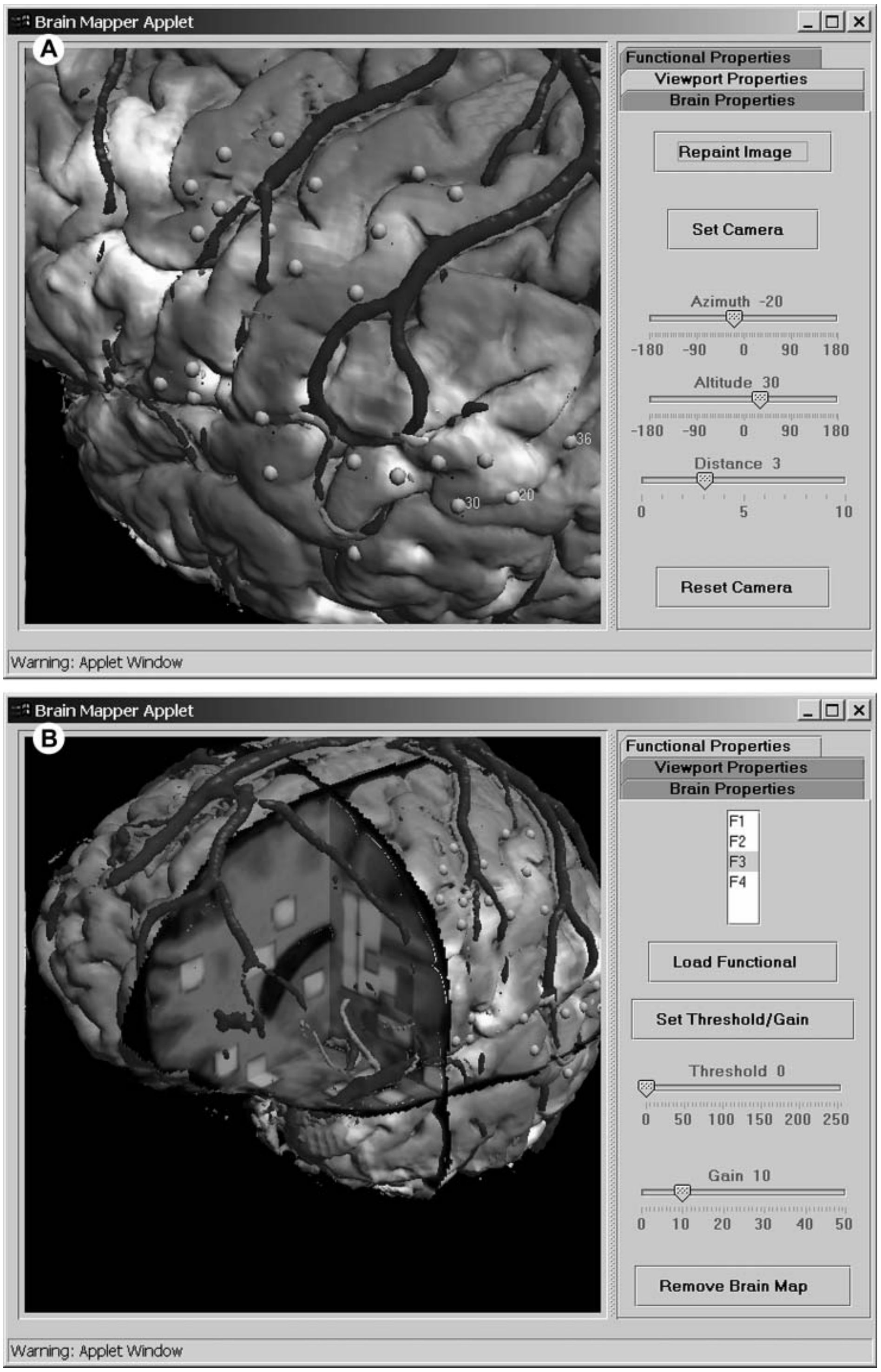
Client Side

Any of the above commands may be executed by connecting to the server machine on a specified port and simply typing the commands at the Lisp prompt. The scene is then rendered by the `(take-snapshot)` command. However, to provide a user-friendly client-side application it is necessary to create graphical user interfaces that send these commands in response to mouse clicks. We have developed several such client implementations that connect to the brain map Graphics Server. These applications are implemented as Java applets or perl CGI pages, but other approaches, e.g., Java Server Pages (JSP), can be also used. The applications we have developed include a Java applet for visualization of an individual patient's dataset, a CGI client for visualizing multiple patient CSM datasets in standard space, and various CGI clients for analyzing the relationships between fMRI and CSM measures of language.

Example 1: Visualization Applet for Individual Patient Datasets

Our Java-based Web interface is implemented in Java 1.1 (supported by both Netscape 4.x and IE 5.x and higher), with the Swing GUI. This Brain Visualizer applet provides nearly the same functionality as that of our stand-alone Brain Visualizer application,[32] which was developed to visualize integrated data from structural MRI, functional MRI, and surgical stimulation sites. Functional activation can be studied both on the cortical surface and in depth of the brain structures. Source code for the Java applet is available on the project Web page.[42]

Figure 2 is a screen shot of the entire desktop with the running applet. The applet is configured to echo in the Java console all the commands sent to and received from the server. When started via a URL without any parameters, the applet connects to the server, issues the `(xmri-subjects)` command to obtain the list of subject identifiers, displays this list in the applet, and sends the `(xsr-select-subject identifier)` command in response to the user selection. The subject identifier can also be appended to the URL at startup, in which case these first two steps are bypassed. Once the subject has been selected, the server loads the associated MRI 3-D image volumes for anatomy, veins, and arteries as well



Azimuth -20
Altitude 30
Distance 3

Threshold 0
Gain 10

Figure 3. User Interface of the Brain Visualization applet. In addition to the Select Patient panel (not shown) and the Brain Properties panel (Figure 2) the applet provides (A) a Viewport Properties panel to control the camera position and (B) a Functional Properties panel that brings up controls for selecting a functional volume, manipulating its visualization properties, and loading/removing CSM maps.

as corresponding 3-D models (`xsr-load-dataset`). The user can then control the server using the GUI shown on the right-hand side of the applet, which includes tabs corresponding to the grouping of API commands described above. User interaction with the GUI is translated into a sequence of commands that are sent to the server.

The controls are organized into three panels. The "Brain Properties" panel shown in Figure 2 allows the user to show the cut-away view and control the appearance of the MRI slices. For example, the sliders labeled "Axial," "Sagittal," and "Coronal" are used to select corresponding cutting planes. Pressing the Mask Wedge button sends the (`xsr-mask-quadrant-web '(sagittal axial coronal)`) command to mask out a segment of the 3-D brain surface model and render a new image. This command causes the server to set to invisible those surface facets that are within the segment to be masked then texture-map the structural MRI intensity values onto the three cutting planes and render the image. With the controls shown in the figure the user could then click the "Unmask Wedge" button, which would send the (`xsr-unmask-web`) command to restore the masked surface. The user could also click the "Set Window/Level" button, which would send the (`xsr-change-window-level window level`) command to change the MRI slice appearance.

The user can click one of the other two tabs to access additional controls. The "Viewport Controls" tab provides camera controls (Fig. 3A), which allows viewing the scene from any angle and magnification. The "Functional Properties" tab brings up controls for selecting one of the functional volumes available for this patient, manipulating its visualization properties, and loading/removing CSM sites (Fig. 3B).

Visualizing functional activation. Functional activation from fMRI is typically observed below the surface of the brain and often occurs in the deeper structures, e.g., the bottom of the sulci or the medial wall of the temporal lobe. For this reason, fMRI data cannot be readily compared with the stimulation map on the surface of the brain or even shown on a 3-D view along with the anatomical features of the brain surface reconstruction. We have implemented two alternative methods for visualizing the relationship between fMRI and stimulation data. With the first method, we highlight the surface areas that are located in the immediate vicinity of the fMRI activation sites. This technique, however, does not reveal activation in deep structures. As an alternative to this method, we implemented a method in which activation anywhere in the brain is projected onto the surface, using the point between the anterior and posterior commissures as a projection point. The user can adjust the visualization by setting the Threshold and Gain parameters (Fig. 3B). These parameters are passed to a weight function, which reflects a desired representation of fMRI characteristics for every triangle of the 3-D model of the brain. This function is then used to modify the color characteristics of the surface triangles. These methods permit direct comparison of the fMRI results and the stimulation map.

Deep activation is typically visualized by superimposing color-coded fMRI intensities onto individual structural MR slices. To achieve better visualization of individual slices in the context of the brain, we combined the 3-D models with individual slice images and implemented a cut-away view with functional activation color-mapped onto the grayscale MRI slices. The color map function is controlled by the same Gain and Threshold parameters (Fig. 3B).

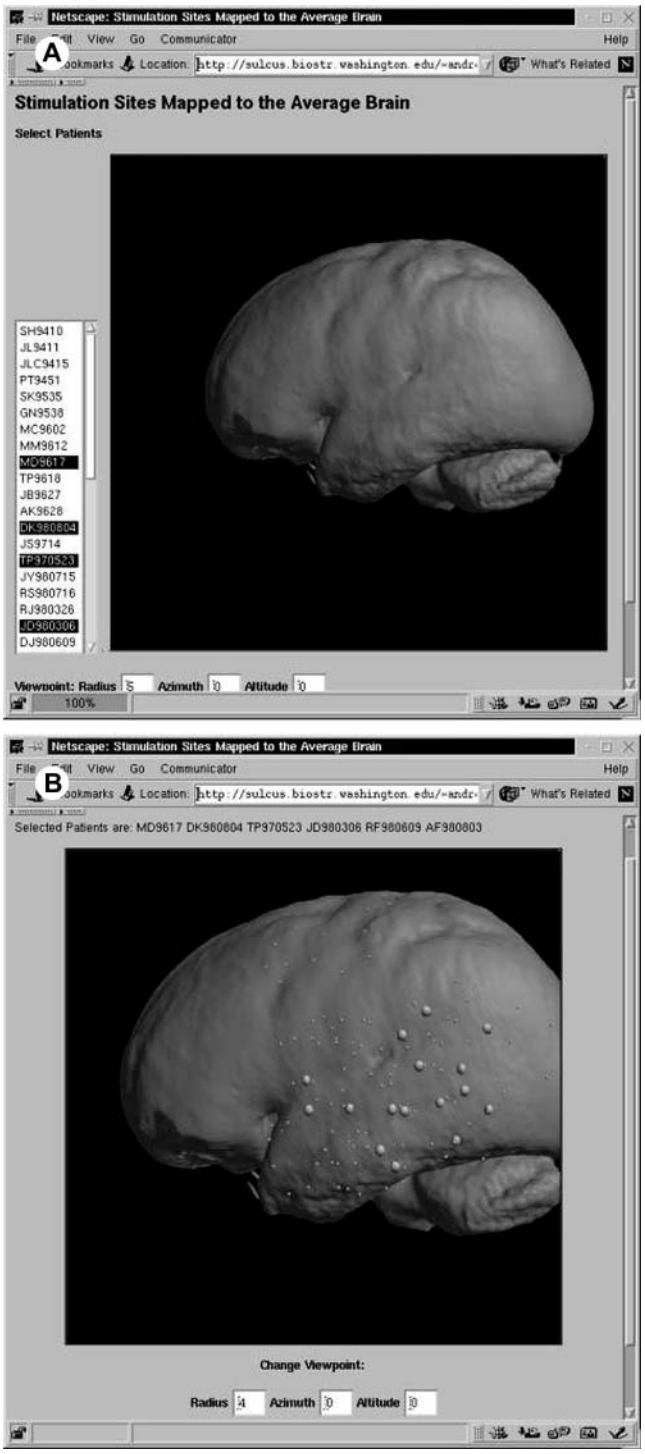
Integration with the experiment management system. In parallel with this project we continue to develop our Web-based EMS.[34,35] This application dynamically generates HTML Web pages based on searches of an underlying relational database. It organizes and manages the experimental data, including patient demographics, surgical photographs, transcripts of the CSM trials, functional and structural image volumes, 3-D models, and language maps—the metadata needed by the brain map Graphics Server.

We have begun to integrate the EMS with Web-based visualization by allowing the user to select a patient in the EMS, then click on a link to invoke the visualization applet with that patient preselected. Further development and tighter integration will enable results of complex queries (e.g., particular language error types) to be sent to the brain map Graphics Server for visualization.

Example 2: CGI Interface for Visualizing Multiple CSM Maps on the Average Brain

Figure 4 shows an example of another Web interface we developed, in this case to present pooled CSM data from a selected group of patients in a common frame of reference. The frame of reference is defined by the average structural MRI volume of 305 brains of normal patients.[43] To map the CSM sites to the average brain volume we used the MNI_AutoReg software[44] to find the best linear transform between the patient-specific coordinate system and the average brain coordinate system. This operation was performed off-line, and the resulting transforms were stored where they could be loaded into the Graphics Server and used to transform each set of CSM sites to the common frame of reference. To provide the context for this visualization, we extracted a surface from the average brain volume, using our image segmentation tools.[31]

The Web interface shown in Fig. 4A permits the user to select a subset of patients for loading and then display the transformed CSM sites for all the selected patients on the average brain (Fig. 4B). The sites identified as language essential are distinguished by size and color. Rotation and zoom controls allow the user to adjust the rendering of the 3-D scene. The Web interface is implemented as a CGI script that utilizes the CGI perl module.[33] The script accesses a separate running instantiation of the Graphics Server that is initialized by loading a set of Lisp functions that adds new functionality. In particular, the 3-D model derived from the average of 305 brains is preloaded at startup to improve response time, which, because of server memory limitations, is not possible to do for all 100+ individual patient datasets that can be visualized by the applet described in Example 1. The CGI approach is adequate in the case of the average brain because once the average brain model is preloaded, only a small amount of data need to be reloaded for each mouse click, namely, the individual CSM sites. However, it would be problematic to implement a more complex Web interface, like the one shown in Example 1, using this simple CGI approach, because of the need to preserve considerably more state information.


A
File Edit View Go Communicator Help
Bookmarks Location: What's Related
B

Netscape: Stimulation Sites Mapped to the Average Brain

Location: http://sulcus.biostr.washington.edu/~andr

Stimulation Sites Mapped to the Average Brain

Select Patients

SH9410
JL9411
JLC9415
PT9451
SK9535
GN9538
MC9602
MM9612
MD9617
TP9618
JB9627
AK9628
DK980804
JS9714
TP970523
JY980715
RS980716
RJ980326
JD980306
DJ980609

Viewpoint: Radius 5 Azimuth 0 Altitude 0

100%

Netscape: Stimulation Sites Mapped to the Average Brain

Location: http://sulcus.biostr.washington.edu/~andr

Selected Patients are: MD9617 DK980804 TP970523 JD980306 RF980609 AF980803

Change Viewpoint:

Radius 4 Azimuth 0 Altitude 0

Figure 4. CGI Web interface for mapping multiple patient's CSM sites to the "average" brain. This Web interface permits the user to (A) select a subset of patients for loading and (B) display the transformed CSM sites for all the selected patients on the average brain. The sites identified as language essential are distinguished by size and color.

Status Report

Usage and Neuroscience Results

The Web-based visualization and analysis tools that we describe in this report are used routinely by our collaborators in the departments of Psychology, Speech & Hearing Science, Neurological Surgery, and Radiology at the University of Washington. These tools have greatly facilitated collaboration among a distributed and diverse group of researchers working on various aspects of the UW Human Brain Project. Individual researchers use these Web interfaces to visualize and analyze integrated data from several modalities without having to come to the SIG laboratory or needing to understand how to install or use the underlying applications.

More recently, the visualization applet is being used by one of us in planning language studies during neurosurgery. Before neurosurgery, fMRI activations are mapped onto the MRI-based structural models and then loaded into the database where they are available for viewing in the applet. The rendered image of fMRI activation projected onto the surface is then used to help determine during neurosurgery where to perform CSM and where to insert microelectrodes for recording of the responses of single neurons to language tasks. The goal of this study is to determine whether neurons within the fMRI activation regions show firing rates that are different from neurons that are not in the fMRI activation regions. The ability of the neurosurgeon to do this planning from his or her own office, without coming to our laboratory, has greatly facilitated this process.

The Web-based tools have facilitated scientific findings pertaining to language function of the brain that have resulted in several neuroscience publications: Corina et al.[45,46] compared functional language maps derived from CSM and fMRI. Our Web-based data integration techniques were used to examine qualitatively and quantitatively the spatial proximity between sites of naming disruption observed during CSM and fMRI activation in the same patients. The data from a common task of visual object naming indicate only moderate correspondence between these two measures.

Visualization of the distribution of the CSM sites from multiple patients superimposed on the average brain was used by Martin et al.[47] They generated multipatient maps, which summarize essential language sites for various patient groupings. Comparisons were made between language sites of patients with right-brain dominant language and those with left-brain language dominance. Other groups compared were patients with low versus high verbal IQs (VIQs) and males with low VIQs versus females with low VIQs. Serafini et al.[48] used our Web-based tools to visualize and compare several functional modalities, including fMRI, proton echo-planar spectroscopic imaging (PEPSI), and event-related potentials (ERP) during a visual object naming task.

Finally, Kinbar et al.[49] used the Web interface to analyze distributions of essential language sites with respect to the type of error made during the CSM procedure. They tabulated the frequency, location, and nature of error types, finding that semantic paraphasia was most common, followed by phonological reductions, phonological paraphasia, neologisms, and semantic/phonological blends. Semantic paraphasias were seen across wide regions of the cortex. Phonological paraphasias were more limited in extent and found with stimulation to the superior and middle temporal gyri (STG, MTG), supramarginal gyrus, and the inferior frontal gyrus. Phonological reductions were the most regionally specific and associated with posterior STG, MTG, and ITG, with a surprising lack of frontal lobe involvement.

Performance

Our collaborators typically experience response times of about 5 seconds per generated image (longer when updating the scene involves complex computational tasks). This response is composed of the following chain of events: (1) the server receives a command from the client and modifies the 3-D scene, (2) the server renders the new scene and saves it as a compressed image, (3) the client retrieves the image, and (4) the client decompresses and displays the image. Step one—modifying the scene—is highly variable. Most of the tasks (e.g., changing camera position or adjusting MRI window and level) require no more than a few milliseconds, whereas a few other commands, e.g., loading the dataset or adding functional volume, may take several seconds. Step two—rendering the 3-D scene—depends on scene complexity (i.e., the number of triangles in all 3-D models) and the image size. For example, the scene shown in Fig. 2 (about 1,000,000 triangles, image size 512×512 pixels) takes about 6 seconds on a Pentium IV 1.5 GHz computer. Steps three and four are largely beyond our control because they depend entirely on the client's computer and network capabilities. Step three—retrieving the image—depends on both the image size (769 K uncompressed and about 200 K compressed) and network throughput. Although image retrieval could be a bottleneck on a very slow network (e.g., about 40 seconds on a 50-kbs dial-up connection), it should not take more than a second on a fast network. The last step—decompressing and displaying the image—depends on the client's Java VM performance but should be well under 1 second. Therefore, rendering the scene in software (estimated as about 3 seconds for the example shown in Fig. 2) currently constitutes the bulk of the response time and presents a bottleneck that we are addressing on the server in our current work (see Discussion).

Discussion

In this report, we described a server-side approach to implementing Web-based interactive visualization and analysis tools for large 3-D neuroimage datasets. During the approximately three years that the system has been in active use, the primary lesson we have learned is that this approach works very well for the types of users with whom we work. Although the server-side approach does not permit as much interaction as stand-alone or client-side applications, and although the response is not real-time, these disadvantages are far outweighed by the convenience of having complex 3-D visualizations of patient data available directly at the user desktop on low-end machines without the necessity of installing special software and without the need to come to our central facility.

Our primary users are working biologists who only want to use the computer as a tool in the same way they use office applications or gene databases. Our experience suggests that these types of users are far more prevalent than the more computationally sophisticated users who have the knowledge and hardware to install complex client-side programs. In some ways our greatest success comes about when the systems we build are so simple to use that the users wonder why informatics is so difficult as to warrant an entire field of study. Although hardware and software are rapidly improving, we believe that the server-based approach makes sense now because client-side rendering technologies are still too complex and slow for most users. In addition, neuroimaging datasets are typically very large (in our case, approximately a gigabyte per subject), and Internet bandwidth is not yet high enough to support massive transfer of such data. Even at the client end neuroimaging data may exceed the memory capacity of a typical user workstation, resulting in low performance or even program crashes. With the server-side approach, all these issues of compatibility, data storage, and performance need only be addressed once—on the server—which makes this approach appealing for neuroimaging applications that involve visualizing large image volumes and 3-D models on workstations in the laboratories.

The system we have built meets objectives 1 through 7 described in the design objectives. It provides advanced visualization capabilities and the ability to interact with the scene (objectives 1 and 2). Because no 3-D rendering is performed on the desktop, the user application (client) can run on low-cost hardware (objective 3), and because only 2-D images are sent from the server, relatively low bandwidth network connections need be used (objective 4). Because the client is either a standard Web browser for the CGI applications or a simple 2-D Java applet, little or no software installation is required beyond that which is available on most desktops (objective 5). We believe that this feature is an important reason for the success of our approach in our collaborative research. Because the client can be accessed via a simple URL that may have the subject identifier appended to the URL, the client can easily be included in a system that can generate the proper URL, as we have done with our EMS (objective 6).

The server-side approach also makes it much easier to ensure patient confidentiality (objective 7) compared with a client-side approach in which all imaging data are sent to the client. In the latter case the image files may include header information that could be used to identify the patient if not scrubbed. More insidiously, structural MRI is of such high resolution that it is possible to reconstruct the face of the subject from an image volume. In the system we have built, only 2-D image snapshots are sent without any header information. In addition, the 2-D snapshots only show image data within the confines of the brain (e.g., Fig. 2), so the face cannot be reconstructed. Thus, the server-based approach could be instrumental in giving users access to the analysis and visualization of essential aspects of patient imaging data while eliminating concerns about patient confidentiality.

Although the system we have built partially meets design objectives 8 through 10, there is room for improvement in these areas. One desired improvement is to decrease the 5-second response time (objective 8) by reducing the rendering time at the server (although we have heard no complaints from the users about this response time, most likely because they have never had anything like this capability in the past). We are investigating both hardware rendering and parallel data processing.

As noted previously, rendering is currently performed using the off-screen rendering functionality provided by the Mesa3D graphics library.[39] The drawback of using this library is that it performs software rendering, which is computationally demanding and not nearly as fast as hardware rendering.

Three-dimensional graphics hardware, driven by the PC gaming industry, has become not only highly capable, but also very affordable, and Web-based neuroimaging applications should capitalize on this technology.

It is feasible to take advantage of 3-D graphics hardware by rendering on screen and saving the images, i.e., by using screen grabbers, but this approach would make the server more cumbersome and would prevent the screen from being used for other purposes. However, the latest releases of hardware-specific OpenGL implementations support hardware-based off-screen rendering capabilities, which potentially makes them ideally suited for a server-based approach to 3-D medical imaging visualization. Off-screen rendering is generally not a part of Open GL core functionality but may be provided by extensions like GLX 1.3 and others. We are currently experimenting with this approach and have achieved promising preliminary results.

Another advantage of the server-based approach is that the server can run on high-performance hardware that makes possible parallel data processing methods. Even inexpensive modern servers typically implement a Symmetric MultiProcessor (SMP) architecture with two, four, or more CPUs, while advanced servers may implement massively parallel architectures. In our group, pilot studies were conducted to explore several different approaches to parallel rendering in software using the Mesa3D library, including (1) the Parallel Virtual Machine (PVM), (2) lightweight threads, and (3) System V Interprocess Communication (IPC).[50] Lightweight threads and system V IPC approaches permit the utilization of several CPUs available on the same server, whereas the PVM approach can be used to distribute work among many computers on the network. Our preliminary studies confirm that rendering of 3-D scenes in software could be accelerated dramatically using parallel techniques.

A third approach to reducing rendering time is to take advantage of techniques that have been developed for decimating the number of triangles that need to be displayed without sacrificing image quality.[51]

Although parallel processing and decimation are promising approaches, our experience suggests that the best speedup will be obtained by using graphics cards specifically optimized for hardware rendering. On the other hand, computationally intensive tasks that do not involve surface rendering (such as spatial queries or volume rendering) may benefit from other approaches. In future work we will investigate the tradeoffs among these alternative approaches. The advantage of the server-based approach is that these enhancements will be transparent to the end user except in the sense that response time will decrease.

The server we have built can currently only accept image volumes and 3-D models in a few formats (Minc[52] and Analyze 7.5 for image volumes, and an in-house Skandha4 format for 3-D models). The server also only runs on SGI and Linux computers but not Windows. Because Skandha4 is not widely supported (like many other brain imaging toolkits), and Lisp is not the favorite of most developers, we currently are porting our stand-alone and server-based applications to a Java3D environment, which will allow us to take advantage of Java3D hardware acceleration (objective 8), compatibility with many image and 3-D model formats (objective 9), and cross-platform portability (objective 10).[53] The stand-alone application is being designed so that it can be run easily as a graphics server, using the basic techniques described in this report. In the meantime, the current code may be of interest to Lisp programmers developing similar applications.[42]

Although this report presents a specific application in brain mapping, the techniques should be applicable to other areas that involve visualization of large 3-D image datasets. Three-dimensional imaging is playing an important role in biomedicine ranging from the molecular level to macroscopic anatomy and is becoming more important in medicine for diagnosis and treatment planning. Current teleradiology systems primarily deal with 2-D images. Remote interactive visualization of 3-D models obtained from patient-specific image datasets has the potential to greatly advance such fields as telemedicine and teleradiology,[6–8] in which additional functionality is often desired beyond basic image transfer and management. When accessed via a Web-based medical record system such as the University of Washington's Mindscape,[54] a 3-D visualization server could bring advanced imaging tools directly to the primary care provider. By requiring only standard desktop tools such as a Web browser and by dealing with patient security issues at the server, such tools could become a valuable adjunct in clinical medicine as well as research.

References ■

1. Brinkley JF, Rosse C. Imaging and the Human Brain Project: a review. Methods Inf Med. 2002;41:245–60.
2. Gee JC, Reivich M, Bajcsy R. Elastically deforming 3D atlas to match anatomical brain images. J. Comput Assist Tomograph. 1993;17(2):225–36.
3. Collins DL, Holmes DJ, Peters TM, Evans AC. Automatic 3-D model-based neuroanatomical segmentation. Hum Brain Mapp. 1995;3:190–208.
4. Van Essen DC, Drury HA. Structural and functional analysis of human cerebral cortex using a surface-based atlas. J Neurosci. 1997;17(18):7079–102.
5. Fischl B, Sereno MI, Dale AM. Cortical surface-based analysis. II: Inflation, flattening, and a surface-based coordinate system. Neuroimage. 1999;9(2):195–207.
6. Evers H, Mayer A, Engelmann U, et al. Extending a teleradiology system by tools for 3D-visualization and volumetric analysis through a plug-in mechanism. In: Proc MEDINFO. 1998:1033–5.
7. Samothrakis S, Arvanitis TN, Plataniotis A, McNeill MD, Lister PF. WWW creates new interactive 3D graphics and collaborative environments for medical research and education. Int J Med Inf. 1997;47:69–73.
8. Nigel WJ, Riding M, Sadarjoen A, Blumrozen L. Bringing 3D to teleradiology. In: International Conference on Information Visualization (IV2000). 2000, p 4.
9. Poliakov AV, Albright E, Corina D, Ojemann G, Martin RF, Brinkley JF. Server-based approach to web visualization of integrated 3-D medical image data. In: Proc. AMIA Fall Symposium. 2001:533-7. Available at: http://quad.biostr.washington.edu/~andrew/cgi-bin/Brain_Browser.cgi?patient=P54. Accessed January 4, 2005.
10. Koslow S, Hyman S. Human Brain Project: A program for the new millennium. Einstein Quarterly J Biol Med. 2000;17:7–15.
11. Cocosco CA, Kollokian V, Kwan RK, Evans AC. BrainWeb: online interface to a 3D MRI simulated brain database. Neuroimage. 1997;5(4):S425. Available at: http://www.bic.mni.mcgill.ca/brainweb/. Accessed January 4, 2005.

12. Cocosco CA. ICBM View. Available at: http://www.bic.mni.mcgill.ca/cgi/icbm_view/; 2004. Accessed January 4, 2005.
13. National Library of Medicine. The Visible Human Project. Available at: http://www.nlm.nih.gov/research/visible/visible_human.html; 1996. Accessed January 4, 2005.
14. Johnson KA, Becker JA. The Whole Brain Atlas. Available at: http://www.med.harvard.edu/AANLIB/home.html; 2001. Accessed January 4, 2005.
15. Bowden DM. Braininfo: A primate brain information system. Available at: http://braininfo.rprc.washington.edu; 2001. Accessed January 4, 2005.
16. Brinkley JF, Bradley SW, Sundsten JW, Rosse C. The Digital Anatomist information system and its use in the generation and delivery of Web-based anatomy atlases. Comput Biomed Res. 1997;30:472–503. Available at: http://www9.biostr.washington.edu/da.html. Accessed January 4, 2005.
17. Golland P, Kikinis R, Halle M, et al. AnatomyBrowser: a novel approach to visualization and integration of medical information. Comput Assist Surg. 1999;4:129–43. Available at: http://splweb.bwh.harvard.edu:8000/pages/papers/AnatomyBrowser/current/index.html#curr. Accessed January 4, 2005.
18. International Neuroimaging Consortium. Cortical Flat Mapping. Available at: http://www.neurovia.umn.edu/incweb/circle-pack/; 2000. Accessed January 4, 2005.
19. Toga AW. UCLA Laboratory for Neuro Imaging (LONI). Available at: http://www.loni.ucla.edu/; 2001. Accessed January 4, 2005.
20. Luttgau A, Bendl R. Technical aspects of internet based knowledge presentation in radiology. Med Inform Internet Med. 2001;26(4):265–81.
21. Falk V, Mintz D, Grunenfelder J, Fann JI, Burdon TA. Influence of three-dimensional vision on surgical telemanipulator performance. Surg Endosc. 2001;15(11):1282–8.
22. Wilkinson EP, Shahidi R, Wang B, Martin DP, Adler JR Jr, Steinberg GK. Remote-rendered 3D CT angiography (3DCTA) as an intraoperative aid in cerebrovascular neurosurgery. Comput Aided Surg. 1999;4:256–63.
23. Sereno MI. Webcortex: Web interface to cortical surface database. Available at: http://cogsci.ucsd.edu/~sereno/webcortex.html; 2001. Accessed January 4, 2005.
24. Dale AM, Fischl B, Sereno MI. Cortical surface-based analysis. I. Segmentation and surface reconstruction. Neuroimage. 1999; 9(2):179–94.
25. Van Essen D, Dickson J, Harwell J, Hanlon D. WebCaret and SumsDB: online access to surface-based representations of cerebral and cerebellar cortex in primates and rodents. In: Human Brain Project Annual Meeting. Bethesda, MD; 2004. Available at: http://www.nimh.nih.gov/neuroinformatics/vanessen204.cfm. Accessed January 4, 2005.
26. Van Essen DC, Drury HA, Dickson J, Harwell J, Hanlon D, Anderson CH. An integrated software suite for surface-based analysis of cerebral cortex. J Am Med Inform Assoc. 2001; 8(5):443–59. Available at: http://stp.wustl.edu. Accessed January 4, 2005.
27. Engel K, Sommer O, Ertl T. A framework for interactive hardware-accelerated remote 3D-visualization. In: de Leeuw W, van Liere R (eds). Data Visualization 2000: Proceedings of the Joint Eurographics and IEEE Tcvg Symposium on Visualization in Amsterdam, the Netherlands, May 29–31, 2000. Vienna: Springer; 2000. Available at: http://www2.chemie.uni-erlangen.de/projects/ChemVis/VisSym2000.pdf. Accessed January 4, 2005.
28. Web 3D Consortium. X3D. Available at: http://www.web3d.org/index.html; 2004. Accessed January 4, 2005.
29. Brinkley JF, Myers LM, Prothero JS, et al. A structural information framework for brain mapping. In: Koslow SH, Huerta MF (eds). Neuroinformatics: An Overview of the Human Brain Project. Mahwah, NJ: Lawrence Erlbaum, 1997, pp 309–34.
30. Ojemann G, Ojemann J, Lettich E, Berger M. Cortical language localization in left, dominant hemisphere: an electrical stimulation mapping investigation in 117 patients. J. Neurosurg. 1989; 71:316–26.
31. Hinshaw KP, Poliakov AV, Martin RF, Moore EB, Shapiro LG, Brinkley JF. Shape-based cortical surface segmentation for visualization brain mapping. Neuroimage. 2002;16(2):295–316. Available at: http://sig.biostr.washington.edu/publications/online/neuroimage2002. Accessed January 4, 2005.
32. Poliakov AV, Hinshaw KP, Rosse C, Brinkley JF. Integration and visualization of multimodality brain data for language mapping. In: Proc AMIA Fall Symp. Washington, DC; 1999:349–53.
33. Stein LD. Official Guide to Programming with CGI.pm. New York, NY: John Wiley & Sons; 1998.
34. Jakobovits RM, Modayur B, Brinkley JF. A Web-based repository manager for brain mapping data. In: Proc AMIA Fall Symp. Washington, DC; 1996:309–13.
35. Jakobovits RM, Rosse C, Brinkley JF. An open source toolkit for building biomedical web applications. J Am Med Inform Assoc. 2002;9(6):557–90. Available at: http://sigpubs.biostr.washington.edu/archive/00000134/. Accessed January 4, 2005.
36. Brinkley JF, Prothero JS. Slisp: A flexible software toolkit for hybrid, embedded and distributed applications. Software—Practice and Experience. 1997;27(1):33–48.
37. Prothero JS, Hinshaw KP, Brinkley JF. Skandha4 and Slisp. Available at: http://sig.biostr.washington.edu/projects/skandha4/; 1997. Accessed January 4, 2005.
38. Hood JD. Using AutoCAD with AutoLISP. New York, NY: McGraw Hill; 1990.
39. Mesa. The Mesa 3D Graphics Library. Available at: http://www.mesa3d.org/; 2004. Accessed January 4, 2005.
40. Wong BA, Rosse C, Brinkley JF. Semi-automatic scene generation using the Digital Anatomist Foundational Model. In: Proc AMIA Fall Symp. Washington, DC; 1999:637–41.
41. Brinkley JF, Wong BA, Hinshaw KP, Rosse C. Design of an anatomy information system. Computer Graphics and Applications. 1999;19(3):38–48. Available at: http://sigpubs.biostr.washington.edu/archive/00000024/. Accessed January 4, 2005.
42. Poliakov AV, Brinkley JF. Brain Visualizer. Available at: http://sig.biostr.washington.edu/projects/brainVisualizer/; 2004. Accessed January 4, 2005.
43. Evans AC, Collins DL, Mills SR, Brown ED, Kelly RL, Peters TM. 3D statistical neuroanatomical models from 305 MRI volumes. In: Proceedings, IEEE Nuclear Science Symposium and Medical Imaging Conference. 1993:1813–7.
44. Collins DL, Neelin P, Peters TM, Evans AC. Automatic 3-D intersubject registration of MR volumetric data in standardized Talairach space. J Comput Assist Tomograph. 1994;18(2):192–205.
45. Corina DP, Steury KR, Mulligan KA, et al. A comparison of language cortex identified by cortical stimulation mapping and fMRI techniques. In: Abstracts, Society for Neuroscience Annual Meeting. Miami, FL; 1999:654.2.
46. Corina DP, Poliakov AV, Steury K, et al. Correspondences between language cortex identified by cortical stimulation mapping and fMRI. Neuroimage (Human Brain Mapping Annual Meeting, June 12-16). 2000;11(5):S295.
47. Martin RF, Poliakov AV, Mulligan KA, et al. Multi-patient mapping of language sites on 3-D brain models. In: Abstracts, 30th Annual Meeting, Society for Neuroscience. New Orleans, LA; 2000:464.20.
48. Serafini S. Functional neuroanatomy during language processing: correspondence of cortical stimulation mapping, fMRI, PEPSI and ERP during a visual object naming task [PhD Thesis]. Seattle, WA: University of Washington; 2002.
49. Kinbar K, Martin R, Hill J, et al. Naming deficits during cortical stimulation mapping: functional specificity reconsidered. In: Proceedings, 8th Annual Meeting, Cognitive Neuroscience Society (CMS). New York, NY;2001:126.

50. Albright E. Dynamic Scene Generation and Software Parallel Rendering of Anatomical Structures [MS]. Seattle, WA: University of Washington; 2000.
51. Schroeder W, Citriniti T. Decimating polygon meshes. Dr. Dobb's Journal. 1997;22(7):109–10.
52. Montreal Neurological Institute. MINC. Available at: ftp://ftp.bic.mni.mcgill.ca/pub/minc/README; 2002. Accessed January 4, 2005.
53. Moore E, Poliakov A, Brinkley JF. Brain visualization in Java3D. In: Proceedings, MEDINFO. San Francisco, CA; 2004. Available at: http://sigpubs.biostr.washington.edu/archive/00000151/. Accessed January 4, 2005.
54. Tarczy-Hornoch P, Kwan-Gett TS, Fouche L, et al. Meeting clinician information needs by integrating access to the medical record and knowledge resources via the Web. Proc AMIA Annu Fall Symp. 1997;809–13.