Server-based Approach to Web Visualization of Integrated 3-D Medical Image Data

Andrew V. Poliakov, Ph.D., Evan Albright, M.S, David Corina Ph.D., George Ojemann, M.D., Richard F. Martin, Ph.D. and James F. Brinkley, M.D., Ph.D.
Structural Informatics Group, Department of Biological Structure
University of Washington, Seattle, Washington USA

ABSTRACT

Although computer processing power and network bandwidth are rapidly increasing, the average desktop is still not able to rapidly process large datasets such as 3-D medical image volumes. We have therefore developed a server side approach to this problem, in which a high performance graphics server accepts commands from web clients to load, process and render 3-D image volumes and models. The renderings are saved as 2-D snapshots on the server, where they are uploaded and displayed on the client. User interactions with the graphic interface on the client side are translated into additional commands to manipulate the 3-D scene, after which the server re-renders the scene and sends a new image to the client. Example forms-based and Java-based clients are described for a brain mapping application, but the techniques should be applicable to multiple domains where 3-D medical image visualization is of interest.

INTRODUCTION

The Web has made it possible for the average user to access unprecedented amounts of information. Initially, most of this information was in the form of static html pages. Increasingly, the information is dynamically generated using web technologies like common gateway interface (CGI), Java applets, Active Server Pages, Java Server Pages, etc., that connect to backend servers or legacy applications. This approach has lead to the emergence of the application service provider (ASP) model, in which computationally demanding or high bandwidth operations are performed on a server, and the results are presented to the user via the Web. Such an approach is becoming increasingly commercially viable because the client need not worry about the details of the complex operations performed on the server.

A prime candidate for the ASP approach is 3-D medical image visualization and analysis. Although 2-D images are commonly sent to the client, where they are either displayed directly or processed further, 3-D image volumes are generally too large to be sent in a reasonable time over current networks, and none but the highest level client machines have the computational power to process and visualize 3-D image datasets. Even with high performance client hardware the application programs needed to display and render 3-D datasets are not easily installed or run by the average user. For special-purpose 3-D image data the difficulties are even greater because much of the software is only available in research labs and is not yet “industrial strength”.

In this paper we describe an example ASP approach to 3-D medical image data visualization over the Web, in which computationally intensive operations such as 3-D rendering are done at a server, and 2-D image snapshots of the renderings are sent to the client. A remote user visualizes and manipulates the 3-D data using either a standard forms-based interface or a Java applet. The techniques are described in terms of a system for mapping and visualizing language areas in the brain, but they should be applicable to other areas such as teleradiology, distance learning, or surgical planning.

LANGUAGE MAPPING IN THE BRAIN

As a part of the Human Brain Project, the Structural Informatics Group (SIG) at the University of Washington is developing software tools for processing, integrating and visualizing multimodality data for language mapping. Prior to surgery for intractable epilepsy, structural magnetic resonance (MRI) and functional magnetic resonance (fMRI) image volumes are collected. The structural MRI image volumes provide an anatomical substrate on which to map functional information. 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. A goal of the mapping project is to integrate and visualize these diverse forms of language data in order to better understand language organization in the brain. The epilepsy cases present a unique opportunity because the CSM data provide a “gold standard” against which the fMRI and other non-invasive methods can be compared.

The software tools we are developing run on high-end Linux or Silicon Graphics computers in SIG, and allow i) reconstruction of 3-D models of the cortical surface, veins and arteries from MRI scans, ii) reconstruction of the 3-D location of the CSM sites with respect to the 3-D models, iii) integration of fMRI image volumes with the 3-D structural models, iv) quantitative correlation between fMRI and CSM language areas, and v) visualization of the integrated data. These capabilities are implemented in two stand-alone applications, the Brain Mapper[1]and the Brain Visualizer[2].

These stand-alone applications are limited to computer facilities within SIG. Although they can be accessed remotely, our collaborators found it inconvenient to use them, in part due to the sluggish response that an application might have on a remote X server. Running the application locally requires high performance hardware and massive data storage, and involves maintaining and updating the software. These reasons, plus the fact that the tools were originally designed with the ability to function as remote servers, created an opportunity to develop a web-based approach.

SYSTEM ARCHITECTURE

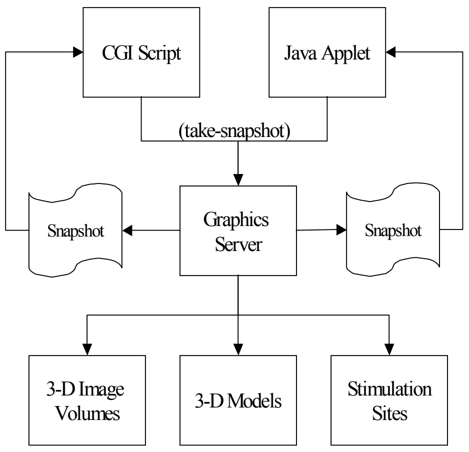


Figure 1.

Figure 1 shows the architecture of our server-side visualization approach. The graphics server accepts Lisp-like commands over a dedicated port that is accessed by a Java or CGI client program. The server loads integrated fMRI and MRI 3-D image volumes, 3-D models, and stimulation sites. In response to Lisp-like commands such as (take-snapshot) the server renders the resulting 3-D scenes, saving 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 Lisp-like commands to be sent to the server, which processes those commands, renders the image, and returns a new snapshot to the client The following sections describe this process in more detail.

Server side

The graphics server, as well as many of our other visualization and analysis tools, utilizes Skandha4 -- a general-purpose in-house graphics toolkit[3]. Skandha4 combines a subset of Common Lisp -- useful for fast interactive programming and prototyping -- with the ability to add pre-compiled C-based primitive functions that significantly accelerate computationally demanding routines. Skandha4 supports 3-D graphics, with drivers for IRIX GL and OpenGL available. It includes a module for processing and storing MRI data, with support for the Montreal MINC 3-D image volume file format[4,5], which is becoming a standard for the Human Brain Project. Skandha4 also includes a GUI module that provides a local event-driven interface. All the functionality of Skandha4, including the GUI module, is accessible via Lisp functions. By separating the processing functions from the GUI functions it is possible to create different front-ends for the same functionality. For example, the standalone Brain Mapper and Brain Visualizer applications were created by combining functions in the GUI module with those in the image processing and rendering modules. The web applications described in this paper were created by calling these same functions remotely.

Skandha4 is designed to operate in either server mode or standalone mode. A connecting client needs to implement a simple ASCII networking protocol[3] to access all the functionality of the Skandha4 modules. In server mode, Skandha4 implements a pre-forking model, i.e. the server forks several subprocesses in advance, so that each of them can handle a new client. This, plus the ability to perform off-screen rendering, makes Skandha4 well suited as a backend graphics server to support interactive visualization on the web.

In addition to the Brain Browser web interface described in this paper, Skandha4 in server mode is also used for intelligent 3-D scene generation by an educational application that dynamically constructs 3-D anatomical scenes from 3-D model primitives[6]. 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 Browser.

Client side

Forms-based CGI web interfaces. Figure 2 shows an example web interface we developed to present stimulation sites from multiple patients in a common frame of reference. Although simple as far as web interfaces go, this CGI script serves the useful purpose of giving our collaborators remote access to the data and computational tools, which otherwise they would find hard to use. The interface is implemented as a perl script that makes use of the perl CGI module. It allows the user to select a group of patients for which stimulation sites can be visualized after being transformed into Talairach space[4]. To provide a visual reference for the site locations, we used a 3-D model derived from the average of 305 brains of normal patients[5]. The sites identified as language-essential were distinguished by size and color. Rotating and zooming was implemented to let the user adjust the rendering of the 3-D scene. Performance of this web interface was improved by preloading the average brain model into the graphics server.

Java-based applet. Java provides a much more flexible programming environment than forms-based CGI for web-based applications, at the cost of slower startup and potential incompatibility with the web browser. Our Java-based web interface is implemented in Java 1.1 (supported by both Netscape 4.x and IE 5.x), with the Swing GUI. This "Brain Browser" applet provides nearly the same functionality as that of our stand-alone Brain Visualizer application[2], which was developed in order to visualize integrated data from structural MRI, functional MRI and surgical stimulation sites.

Figure 3 is a screen shot of the entire desktop with the running applet. 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 applet is configured to echo in the Java console all the commands sent to the server.

When started, the applet connects to the server and lets the user choose a patient. The server loads MRI 3-D image volumes for anatomy, veins and arteries, and the corresponding 3-D models. The user can then control the server using the GUI shown on the right-hand side of the applet. User interaction with the GUI is translated into a sequence of commands that are sent to the server. For example, in figure 3 the user has already loaded the CSM sites (small round spheres) by clicking a button that sends the (xsr-load-map-sites) command to the server, as shown in the Java console window. The user has loaded one of the fMRI volumes, which is used to brighten the color of the 3-D brain model in areas of fMRI activation, and has adjusted the Axial, Sagittal and Coronal sliders shown in figure 3 These sliders select three cutting planes, which are sent to the server to mask out a segment of the 3-D brain surface model. The server sets to invisible those surface facets that are within the segment to be masked, then texture-maps the structural MRI intensity values onto the three cutting planes, renders the image, and sends a snapshot back to the user. With the controls shown in the figure the user could then click the "Unmask Wedge" button, which would send the (xsr-unmask-wedge) command to restore the masked surface, or could click the "Remove Brain Map" button, which would send the (xsr-remove-map-sites) command to remove the stimulation sites from the image.

The user can also click one of the other tabs to access additional controls. The "Functional Properties" tab brings up controls for manipulating fMRI visualization properties, and the "Viewport Controls" tab controls the camera, including rotation and zoom.

A working demo of this applet can be found via the demos page of the UW Human Brain Project http://sig.biostr.washington.edu/projects/brain/demos.html. Response time is about 12 - 20 seconds when the graphics server is running on a quad Intel Pentium III 550 Mhz processor. Most of this time (10-15) seconds is spent rendering the 3-D scene on the server.

DISCUSSION

In this paper, we describe a server-side approach to visualization of large 3-D image datasets. This approach makes sense at this time because 1) Internet bandwidth is not yet high enough to support massive transfer of large image datasets; 2) Client-side rendering technologies (e.g. Java-3D, VRML viewers) are still in development, and may be too slow to efficiently render large 3-D models (on the order of a million triangles) derived from 3-D medical image volumes. These factors are of course changing daily, and in the near future most processing may take place at the client. In the meantime the server-side approach offers a working solution that can migrate to the client as software and hardware improves. By concentrating the computationally intensive operations at the server, a single upgrade to server processing power can dramatically improve performance for many clients. Software improvements such as parallel rendering should improve the performance even more, as we have shown in preliminary studies.

The server-side approach has of course been possible for a long time, and non-web based access to servers via X Windows or VNC have become popular.

There is nothing wrong with these approaches, but they do require high bandwidth connections, and the choice of platform may be limited. (i.e., the need for a client that can run an X server). Running applications remotely in this manner has well known security implications, and accessing a computer behind a firewall typically requires a special setup. More importantly, such an approach requires that the users be proficient, at least somewhat, with a particular operating system and environment.

On the other hand, creating a custom web interface can make an application truly accessible for anyone. Such an interface makes possible collaboration among a distributed and diverse group of researchers, who can use the interface to share information and to stimulate new ideas. This situation is increasingly becoming the case with the UW Human Brain Project, which involves a collaboration between groups in the departments of Biological Structure, Radiology, Neurosurgery, and Psychiatry. Individual researchers in these departments routinely use these web-interfaces to visualize integrated language-related data without having to come to the SIG lab or needing to understand how to install or use the underlying applications.

Although this paper describes a specific application in brain mapping, the techniques should be applicable to other areas that involve visualization of large 3-D image datasets. Current teleradiology systems primarily deal with 2-D images. Inclusion in these sites of interactive 3-D models obtained from patient-specific 3-D image datasets could potentially improve clinical diagnosis. Remote access to 3-D image visualization servers from the operating room or radiation treatment room could improve surgical and radiation treatment planning without requiring expensive installation of custom software.

3-D image visualization servers could be coupled with web-accessible multimedia databases, such as the one we are building for the UW Brain Project[7], in order to integrate remote experiment management with remote visualization capabilities. When accessed via a Web-based medical record system such as the UW's Mindscape[8], a 3-D visualization server could bring advanced imaging tools directly to the primary care provider. All these applications could greatly improve access to biomedical information, without requiring any investment in software technology besides a personal computer, a web browser, and a fast Internet connection.

ACKNOWLEDGEMENTS

This work was funded by Human Brain Project grant DC02310. We also thank Intel for a generous equipment grant.

REFERENCES

1. Hinshaw, K.P. and J.F. Brinkley, Incorporating constraint-based shape models into an interactive system for functional brain mapping, in Proceedings, American Medical Informatics Association Fall Symposium. 1998 pp. 921-925.

2. A. V. Poliakov, K. P. Hinshaw, C. Rosse and J. F. Brinkley, Integration and Visualization of Multimodality Brain Data for Language Mapping, Proceedings, American Medical Informatics Association Fall Symposium Washington, D.C., 1999.

3. J. F. Brinkley and J. S. Prothero, Slisp: A flexible software toolkit for hybrid, embedded and distributed applications, Software -- Practice and Experience, vol. 27, pp. 33-48, 1997.

4. Automatic 3D Inter-Subject Registration of MR Volumetric Data in Standardized Talairach Space D. L. Collins, P. Neelin, T. M. Peters and A. C. Evans, Journal of Computer Assisted Tomography, 18(2), 192-205, 1994.

5. A. C. Evans and D. L. Collins and S. R. Mills and E. D. Brown and R. L. Kelly and T. M. Peters, "3D statistical neuroanatomical models from 305 MRI volumes", Proc. IEEE-Nuclear Science Symposium and Medical Imaging Conference, 1813-1817, 1993.

6. B. A. Wong and J. F. Brinkley, Dynamic 3-D scene navigation in web-based anatomy atlases, Proceedings, American Medical Informatics Association Fall Symposium Orlando, Florida, 1998.

7. R. M. Jakobovits and J. F. Brinkley, Managing medical research data with a Web-interfacing repository manager, Proceedings, American Medical Informatics Association Fall Symposium, Nashville, pp. 454-458, 1997.

8. Tarczy-Hornoch, P., T.S. Kwan-Gett, L. Fouche, J. Hoath, S. Fuller, K.N. Ibrahim, D.S. Ketchell, J.P. LoGerfo, and H.I. Goldberg, Meeting clinician information needs by integrating access to the medical record and knowledge resources via the Web. Proc AMIA Annu Fall Symp, 1997: pp. 809-13.

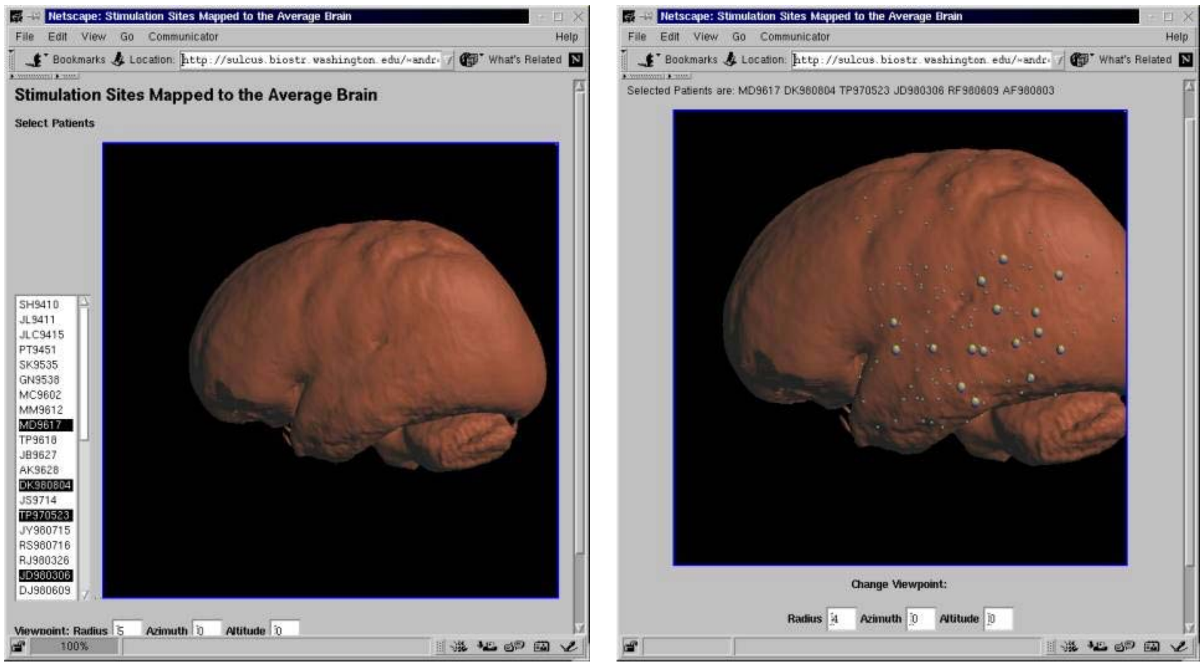
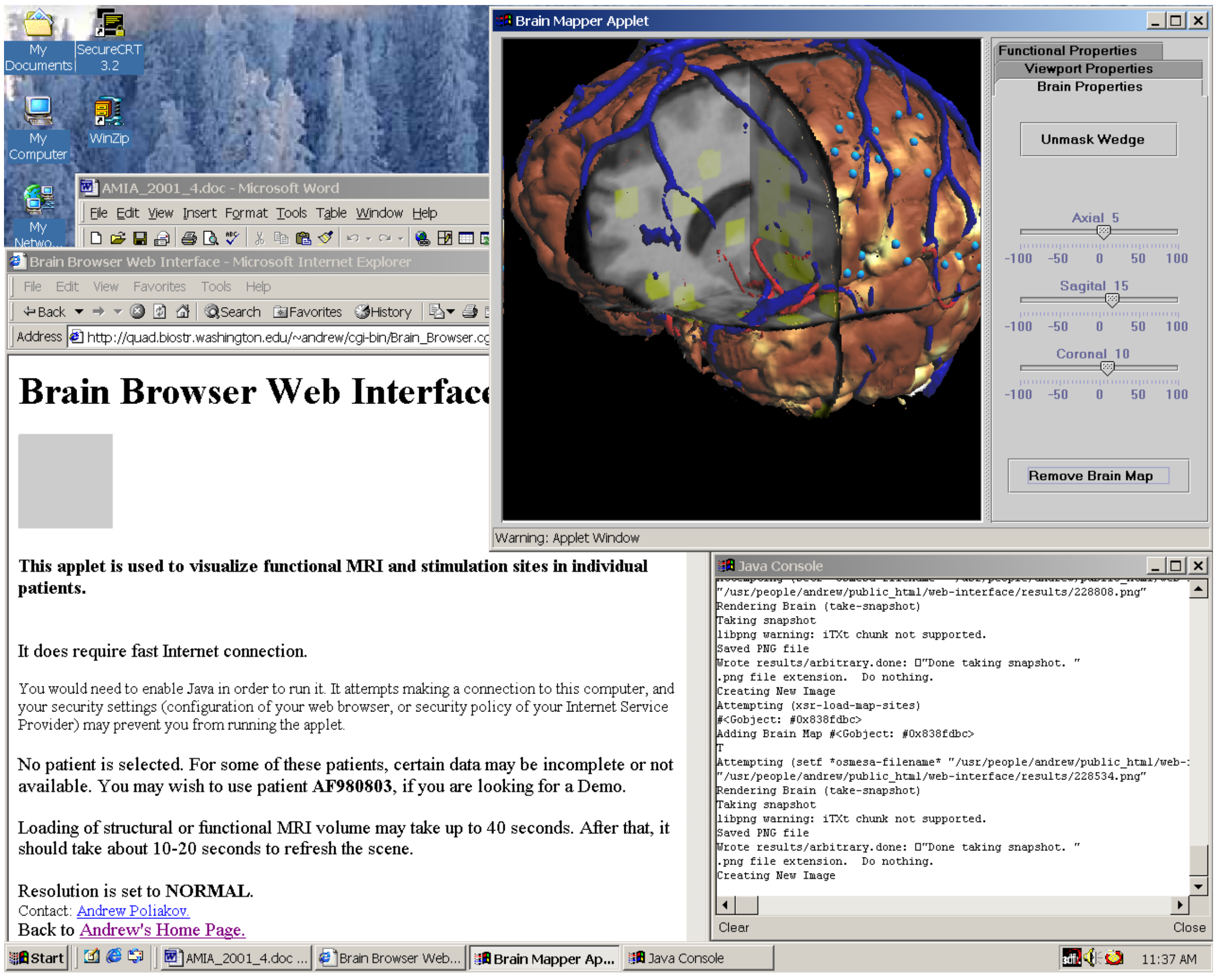

Netscape: Stimulation Sites Mapped to the Average Brain
File Edit View Go Communicator Help
Bookmarks Location: http://sulcus.biostr.washington.edu/~andr What's Related
Stimulation Sites Mapped to the Average Brain
Select Patients
SH9410
JL9411
JLC9415
PT9451
SK9535
GN9538
MC9602
MM9612
MD9617
TP9618
JB9627
AK9628
DK980804
JS9714
TP970523
JY980715
RS980716
RJ980326
JD980306
DJ980609
Viewpoint: Radius 5 Azimuth 0 Altitude 0
100%
Selected Patients are: MD9617 DK980804 TP970523 JD980306 RF980609 AF980803
Change Viewpoint:
Radius 4 Azimuth 0 Altitude 0

Figure 2. Forms-based interface for mapping multiple patients to an "average" brain. Left. Patient selection, Right:. Mapped sites.



Brain Browser Web Interface

This applet is used to visualize functional MRI and stimulation sites in individual patients.

It does require fast Internet connection.

You would need to enable Java in order to run it. It attempts making a connection to this computer, and your security settings (configuration of your web browser, or security policy of your Internet Service Provider) may prevent you from running the applet.

No patient is selected. For some of these patients, certain data may be incomplete or not available. You may wish to use patient **AF980803**, if you are looking for a Demo.

Loading of structural or functional MRI volume may take up to 40 seconds. After that, it should take about 10-20 seconds to refresh the scene.

Resolution is set to **NORMAL**.
Contact: Andrew Poliakov
Back to Andrew's Home Page.

```
"/usr/people/andrew/public_html/web-interface/results/228808.png"
Rendering Brain (take-snapshot)
Taking snapshot
libpng warning: iTXt chunk not supported.
Saved PNG file
Wrote results/arbitrary.done: □"Done taking snapshot. "
.png file extension.  Do nothing.
Creating New Image
Attempting (xsr-load-map-sites)
#<Gobject: #0x838fdbc>
Adding Brain Map #<Gobject: #0x838fdbc>
T
Attempting (setf *osmesa-filename* "/usr/people/andrew/public_html/web-
"/usr/people/andrew/public_html/web-interface/results/228534.png"
Rendering Brain (take-snapshot)
Taking snapshot
libpng warning: iTXt chunk not supported.
Saved PNG file
Wrote results/arbitrary.done: □"Done taking snapshot. "
.png file extension.  Do nothing.
Creating New Image
```

Figure 3. Applet interface for visualizing integrated 3-D image data from a single patient.