# A Client-Server Software Application for Statistical Analysis of fMRI Data

by

Vijay Singh Choudhary

Bachelor of Technology in Civil Engineering,
Indian Institute of Technology Bombay (2002)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004



Author .....
Department of Civil and Environmental Engineering
May 7, 2004

Certified by..
.....
Randy Gollub
Affiliated Faculty of Harvard-MIT Division of Health Sciences and Technology
Thesis Supervisor

Certified by.....
.....
Steven R. Lerman
Director of Center for Educational Computing Initiatives and
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by..
.....
Heidi M. Nepf
Chairman, Committee for Graduate Students

# A Client-Server Software Application for Statistical Analysis of fMRI Data

by

Vijay Singh Choudhary

Submitted to the Department of Civil and Environmental Engineering
on May 7, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science

## Abstract

Statistical analysis methods used for interrogating functional magnetic resonance imaging (fMRI) data are complex and continually evolving. There exist a scarcity of educational material for fMRI. Thus, an instructional based software application was developed for teaching the fundamentals of statistical analysis in fMRI.

For wider accessibility, the application was designed with a client/server architecture. The Java client has a layered design for flexibility and a nice Graphical User Interface (GUI) for user interaction. The application client can be deployed to multiple platforms in heterogeneous and distributed network. The future possibility of adding real-time data processing capabilities in the server led us to choose CGI/Perl/C as server side technologies. The client and server communicates via a simple protocol through the Apache Web Server. The application provides students with opportunities for hands-on exploration of the key concepts using phantom data as well as sample human fMRI data. The simulation allows students to control relevant parameters and observe intermediate results for each step in the analysis stream (spatial smoothing, motion correction, statistical model parameter selection etc.). Eventually this software tool and the accompanying tutorial will be disseminated to researchers across the globe via Biomedical Informatics Research Network (BIRN) portal.

Thesis Supervisor: Randy Gollub
Title: Affiliated Faculty of Harvard-MIT Division of Health Sciences and Technology

Thesis Supervisor: Steven R. Lerman
Title: Director of Center for Educational Computing Initiatives and Professor of Civil and Environmental Engineering

# Acknowledgments

First and foremost, I would like to extend my sincerest thanks to my advisors, Randy Gollub and Steven Lerman, for all their help and support during the project. This work has been made possible only with their continual guidance and encouragement over past one year. Randy helped me a lot in learning the fMRI and neuroscience domain knowledge throughout the course of project. Regular counselling and suggestions from Steve were of great help. I would also like to thank Rick Hoge, my project co-advisor, who assisted me greatly with the implementation, specially on the server side.

I would like to thank Ian Lai, the predecessor of the project, who was always available whenever I had technical difficulties in design or implementaion.

I sincerely thank all my colleagues and staff at Center for Educational Computing Initiatives (CECI), MIT for providing me a very professional working environment. Specially, I am grateful to Mesrob Ohannessian who helped me alot in understanding image processing concepts and Jed Northridge for his help, whenever needed.

I would like to thank my friends for making my stay at MIT enjoyable as well as my family, for their constant support. I must also thank Greg Llacer for assisting with the administrative portions of the project.

Finally, I thank Harvard-MIT division of Health Sciences and Technology for supporting this project through VaNTH educational initiative, with grants from National Science Foundation (NSF).

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Functional Magnetic Resonance Imaging (fMRI) is a new medical imaging technology providing functional, as opposed to anatomical, mapping of the human brain. Brain activity indications in fMRI data is based on the observation that increased neural activity leads to increases in localized cerebral blood flow, blood volume, and blood oxygenation. In fMRI this is referred to as Blood Oxygen Level Dependent (BOLD) contrast. FMRI can provide detailed images of localized brain activity with a spatial accuracy of millimeters and a temporal resolution of seconds [1]. Thus, FMRI is currently one of the best techniques for studying the function of the brain regions that underlie visual perception in humans. This relatively new research tool has found widespread use in a variety of applications at the intersection of biomedical engineering and neuroscience, for example, mapping the boundaries between functional regions of the brain, identifying tumor margins prior to surgery, and investigating the pathology underlying diseases such as schizophrenia [2].

In a typical fMRI experiment, the signal can be overshadowed by noise, making the detection of activation-related signal changes difficult. Though, by applying statistical analysis methods to the time series of signals from each voxel in the brain permits optimal extraction of the signal of interest. This thesis describes the technical details of a Web-based software application developed to help researchers in understanding the fundamentals of statistical analysis of fMRI experiment data.

## 1.1 Motivations

While magnetic resonance imaging (MRI) was introduced for clinical use in the 1970s, functional magnetic resonance imaging (fMRI) was discovered only in the early 1990s [2]. Although the field of fMRI research is still relatively young, it has experienced explosive growth; in the years 1999-2001 more than 900 abstracts were submitted to the International Conferences on the Functional Mapping of the Human Brain [3]. While some investigators in the field have expertise concerning the details of fMRI data analysis, others simply apply free or commercial software packages to their data. These packages often include a multitude of parameters that can be optimized by an fMRI expert, but are rarely adjusted by others [2]. Since the packages have preset defaults that may not be appropriate for all situations, investigators lacking a proper understanding of fMRI data analysis may draw false conclusions from their data if they do not have a proper knowledge in data analysis.

Students and researchers wanting to learn more about fMRI data analysis have limited resources available to them. The most widely available resources are a few recently published textbooks [4, 5] and documentation that accompanies particular software packages. There are also a number of courses and workshops on fMRI, but these represent a limited resource that is not available to all researchers in the field. Thus, there exist a need of educational material for learning about the steps and assumptions underlying standard fMRI data analysis. Hence, we developed an on-line, interactive software application which would help students and researchers to acquire insights required to use existing software packages in an informed fashion and adapt them to their own purposes. The focus of the application is on the fundamental processing steps and parameters commonly used in fMRI data analysis. For these reasons, this educational material will be invaluable to researchers working in domain of fMRI.

The intended audience includes advanced undergraduate and graduate students, as well as investigators who wish to use fMRI in their research but are not familiar with the methods and techniques of fMRI data analysis. This module is intended to

be a standalone source for learning about fMRI data analysis, although it may also be a useful adjunct to existing courses at various universities.

## 1.2 Aims and Objectives

The main objective was to develop an educational module (a software application and accompanying tutorial) to assist advanced undergraduate and graduate students in learning the fundamentals of statistical analysis of fMRI data. This module should allows students to learn about each of the steps in fMRI data analysis, pre-processing, signal to noise estimation, and statistical inference. The software application should provide students with opportunities for hands-on exploration of key concepts for each step in the analysis stream using phantom data, as well as sample human fMRI data. The accompanying tutorial should provide appropriate background materials and guidance, in order to promote exploration, understanding of key concepts and to encourage constructive use of the interactive demonstration.

Another objective was to modify the current tutorial being used in one of the labs for the course "HST.583 Functional Magnetic Resonance Imaging: Data Acquisition and Analysis" through the lens of How People Learn (HPL) framework. The HPL framework and major learning objectives of this educational module are discussed in detail in chapter 3.

## 1.3 Scope of the Work

The software tool and the tutorial covers the basic preprocessing steps and parameters commonly used in analysis of fMRI data, as described in detail in section 5.2. Topics such as fMRI experimental design, the physics of fMRI data acquisition, and advanced methods for data processing and analysis are beyond the scope of this module.

## 1.4 Overview of the Thesis

**Chapter 2**, *Background on fMRI Data Acquisition and Analysis*, builds background on basics of fMRI, fMRI experiment design, noise issues and statistical analysis of data. Also, it presents a compiled report on available educational material for teaching fMRI.

**Chapter 3**, *Educational Goals and Challenges*, describes the enhancements to the tutorial motivated by *How People Learn (HPL)* framework, a strategy for designing effective learning environment

**Chapter 4**, *Web-based Client-Server Application*, covers the overall requirements, architecture and platform technologies chosen for the Web-based interactive software application.

**Chapter 5**, *Detailed Client Design*, covers the details of the design decisions and implementation of the complete client. Chapter describes some of the software development concepts being used in the current web application such as modularity, layered architecture and data abstraction.

**Chapter 6**, *Detailed Server Design*, covers the details of the server design and implementation. In detail, it covers pre-processing of data, server architecture and communication protocols being used with client.

**Chapter 7**, *Conclusions and Future Work*, summarizes some of the key points of the client/server application. Also, it suggests future scope of improvements in the application.

# Chapter 2

# Background on fMRI Data Acquisition and Analysis

## 2.1 Introduction: What is functional imaging of the brain?

Magnetic resonance (MR) imaging uses radio waves and a strong magnetic field to provide clear and detailed pictures of internal organs and tissues. Functional magnetic resonance imaging (fMRI) is a relatively new procedure that uses MR imaging to measure metabolic changes that take place in an active part of the brain or as a result of a stimulus. Physicians know the general areas of the brain responsible for speech, sensation, memory, and other functions. However, the exact locations can vary from individual to individual. Injuries and disease, such as stroke or brain tumor, can even cause aforementioned functions to shift to a different part of the brain. fMRI can help radiologists or physicians to determine precisely which part of the brain is handling critical functions such as thought, speech, movement, and sensation. This information can be critical to planning surgery, treatment for stroke, or other interventions to treat brain disorders.

## 2.2 Basics of fMRI

To understand the statistical issues inherent in fMRI, it is essential to first gain an understanding of how the imaging process itself is thought to work. In this section, author outlines the physics and biophysics underlying fMRI image acquisition.

As described by Ogawa, Lee et al. [6] "Atomic nuclei with an odd number of protons or an odd number of neutrons are affected by magnetic fields. Exposing such nuclei to a strong magnetic field will cause them to try to align themselves parallel or anti-parallel to the field. The parallel orientation has a slightly lower energy than does the anti-parallel orientation. This causes more nuclei to align themselves in the parallel orientation, which, collectively, results in an overall magnetization of the object in the field. The alignment of the nuclei isn't perfect in either direction; instead, the atoms precess about the field at a fixed frequency. "Precession" refers to the revolution of the axis of rotation of the atoms. Frequency of precession varies for each type of nucleus and is related linearly to the strength of the magnetic field. When radiofrequency energy at the frequency of precession of the nuclei is injected into the system, the level of energy increases temporarily, but then returns to equilibrium. The energy emitted in the return to the starting state is at the frequency of precession. Both the absorption and the emission of energy are therefore selective, in that only nuclei that are near the appropriate precession frequencies will be affected. This is the key aspect of resonance. It is the selective absorption and emission of energy that produce the MR (magnetic resonance) signal. Magnetic resonance imaging (MRI) involves gathering data on the precession of the atomic nuclei, resulting in high-resolution images. The strength of the signal is proportional to the number of nuclei of a specific type. Hence the method allows us to count nuclei with particular properties. MR images are typically three-dimensional, representing volumes." The images are divided into volume elements, or voxels; the amplitude of the recorded signal at each voxel in each image is the average nuclear density of the chosen element (usually hydrogen).

With functional MRI, we use a series of MR images collected over time to gather in

formation about neuronal activation in the brain during the course of a scan. While the scan is being performed, subjects may be asked to carry out various cognitive processing tasks; the images will then convey information on which regions of the brain were active and hence involved in the particular task under study. The connection between neuronal activation and the MR images is believed to be as follows. When resting brain neurons become active, the rate of blood flow to the neighborhood of these neurons increases, as glucose is delivered to the regions in question. This is known as the hemodynamic response. As the rate of firing increases for the neurons, their metabolism also increases. The increase in metabolism results in an influx of oxygenated blood to the affected region. Oxygen levels rise in the nearby blood vessels, since active neurons do not consume much more oxygen than when at rest. The magnetic properties of oxygenated and deoxygenated hemoglobin differ (as demonstrated by Pauling in 1935), and this difference affects the measured MR signal through what is called the Blood Oxygenation Level Dependent (or BOLD) effect. Hence, the MR signal from the neighborhood of a neuron should change as the concentration of oxygenated blood around the neuron changes. MR imaging is sensitive enough to detect these functionally induced changes in blood oxygenation in the human brain [6].

The idea that blood flow changes can be correlated to changes in brain function is an old one, presented as early as the end of the last century by the British physiologists Roy and Sherrington (1890). They postulated the existence of an "automatic mechanism" that regulated the blood supply to the brain in a manner dependent on variations in activity [6]. Subsequent research has confirmed this hypothesis, although the exact nature of the system is still unknown. Functional magnetic resonance imaging (fMRI) is a step in the further understanding of this process.

## 2.3 Applications of fMRI

The range of applications of fMRI to neuroscience is growing rapidly. Here is the list of the some of the research areas in which fMRI is proving to be an important tool

[4]:

1. Defining neurophysiological correlates of human behavior;
2. Defining ways in which brain functions can be modulated;
3. Establishing a 'system-level' description for the brain basis of learning;
4. Defining 'networks' for cognitive processing

Other than these, fMRI is also becoming common in clinical applications as well. To list few of these general areas:

1. Mapping of the functional area in the damaged brain;
2. Providing state or trait markers (Whether underlying abnormalities are present only during periods of illness, with return to normal between episodes, or persist independent of clinical status can be addressed through fMRI studies [7]);
3. Defining mechanism of reorganization or compensation from injury.

## 2.4 Designing an fMRI Experiment

In a typical fMRI scanning sequence, over a hundred successive echo-planar images (EPI) are taken at a rate of 1 every 2 to 6 seconds, which gives a total of 4 to 10 minutes for the functional part of the experiment. EPI is a type of magnetic resonance imaging that uses only one nuclear spin excitation per image and therefore can obtain images in a fraction of a second rather than the minutes required in traditional MRI techniques. Since the FMRI measures the relative signal change over time under different stimulus conditions, a control condition is necessary to determine whether the change in the signal is due to the test stimulus condition. Because the change in the MR signal lags behind the change in neural activity by a few seconds (typically 5 sec), the duration of a condition should be in the range of 20 to 60 sec, in which 5 to 15 scans would then take place. To provide maximum contrast between the different stimulus conditions, the order of these conditions should be rotated

somewhat cyclically. Thus, the design of the stimulus should take these constraints into account. Most importantly, the stimulus display must be synchronized with the scanning sequence.

## 2.5 Analysis of fMRI experiments data

The goal of fMRI analysis is to detect, in a robust, sensitive, and valid way those parts of the brain which show increased intensity at the points in time that stimulation was applied. A single volume is made up of individual cuboid elements called voxels. An fMRI dataset from a single session can either be thought of as $t$ volumes, one taken every few seconds, or as $v$ voxels, each with an associated time series of $t$ time points [4].

The basic problem in analysis of functional imaging experiments is to identify voxels that show signal change varying with the changing brain states of interest across the serially acquired images. This becomes a challenging problem for fMRI data because the signal changes are small and the number of voxels simultaneously interrogated across the image is very large. One of the potentially most significant artifacts for fMRI that distinguishes it from other functional imaging techniques is its susceptibility to motion from the movements, either of the head or brain (e.g. with the respiratory or cardiac cycle). So the idea is to address the ways in which the data can be prepared to minimize artifacts and maximize sensitivity for the detection of activation changes. The aim of fMRI analysis is to identify in which voxels' time-series the signal of interest is significantly greater than the noise level. For this, a 4D dataset is initially pre-processed, i.e. prepared for statistical analysis.

### 2.5.1 Pre-processing of fMRI Data

Once fMRI data has been acquired, the preprocessing starts by reconstructing the raw 'k-space' data into images. The next step applied is slice-timing correction; because each slice in each volume is acquired at slightly different times, it necessary to adjust the data so that it appears that all voxels within one volume had been

acquired at exactly the same time. Each volume is now transformed (using rotation and translation) so that image of the brain within each volume is aligned with that in every other volume; this is known as motion correction.

After artifact removal two general approaches to maximizing the signal-to-noise ratio for the time course data then are applied typically: spatial and temporal filtering (smoothing). fMRI is being used to detect a signal change that lasts for only a limited period of time and covers just a small region of the brain. A general result of signal detection is that blurring of a signal (in this case both the dimensions of space and time need to be considered) can enhance the signal-to-noise, hopefully without significantly affecting the activation signal. Also, generally data have large number of spuriously activated voxels that appear to be sites of significant brain activation but are really just an artifact- these typically disappear with the spatial smoothing. After this, each volume's overall intensity level is adjusted so that all volumes have the same mean intensity - this intensity normalization can help reduce the effects of global changes in intensity over time and provide the means to compare images across subjects and sessions. Reduction in low and high frequency noise is normally desired as final step; each voxel's time series is filtered by linear or non-linear tools in order to achieve this [4].

In our software application, we have pre-processed several complete data sets in advance so that we can have good speed of data exchange over the network. If we do the processing of data in real time, it would just add latency to the response from the server, often as much as 15 or 20 minutes. So for demonstrating the effects of pre-processing of data, we do guide the user in a way that he/she understands the basics of pre-processing, its effects and importance before doing statistical analysis.

### 2.5.2 Statistical Analysis

After the pre-processing steps, statistical analysis is carried out to determine which voxels are activated by the stimulation. This can be a simple correlation analysis or more advanced modeling of the expected hemodynamic response to the stimulation. Various possible statistical corrections can be included, such as correction for

smoothness of the measured time series at each voxel. The main output from this step is a statistical map which indicates those points in the image where the brain has activated in response to the stimulus. Mostly each voxel's time series is analyzed independently ('univariate analysis') but there are also 'multivariate' methods. For example, standard general linear model (GLM) analysis is univariate. It's not in the scope of this thesis to explain the complete GLM Analysis. There are many valid ways of performing statistical comparisons between signals in images associated with different brain states and their time courses of change. A common approach is to generate a map of t statistics (the ratio of the mean signal intensity to its standard error) on a voxel-by-voxel basis and use this to identify voxels with significance level exceeding a chosen threshold (i.e. $t > 3$, which might correspond in a particular case to $p < 0.01$).

In the current software application, we model the signal that we record from an fMRI session as a linear combination of the actual signal and the noise. With GLM analysis, we use the signal and noise as regressors, and solve for the coefficients for the linear combination. The mathematical equation for the model is $y = \beta\, x + e$, where y is the observed signal, x is the estimated signal, $\beta$ is the parameter estimate for x, and e is error term that corresponds to our noise. Our application users have the option of choosing different signal and noise models for the analysis. Once analysis parameters are selected, user can request the results by selecting statistical maps (t-maps or p-maps) view in the module. In addition to seeing the default statistical map that shows every voxel, user can also specify a threshold such that only the activation is visible. Also, we have a module for representation of the regressors used in the analysis with the design matrix and stimulus covariance matrix. The design matrix is a visual representation of the regressors used in the analysis, and consists of one or more columns. The first column represents the paradigm convolved with the hemodynamic response function, and the subsequent columns represent polynomials used in detrending. The stimulus covariance matrix is necessary when testing contrasts involving multiple parameters under the general linear model.

## 2.6 Previous Work in teaching fMRI

Most of the currently available educational material for fMRI focuses on the physics of image acquisition and experimental design, but few resources exist for fMRI data analysis, and those that do focus primarily on theory or one specific software package [8]. There are a few recent of text books [5, 6] that provide chapters on fMRI data analysis and delve into detail about the theory behind the statistics of general linear model (GLM) analysis. There also exist some journal and research publications on statistical analysis, but all of them talk about new cutting edge methods. Several researchers have posted online material on the Web to explain the basics of fMRI with some detail [9] - [16], but they cover data analysis briefly and sometimes in the context of software analysis packages [14, 16], such as SPM [17] or Brain Voyager [18].

### 2.6.1 Software packages for the analysis of fMRI data

This section describes some of the extensively used packages in the MRI and fMRI community. AFNI (Analysis of Functional Neuroimages) [19] is a flexible package that allows graphical display of image data and analysis using the correlation method, developed by Bandettini et al, among others [20]. "Plug-in" modules are available to help users customize their analyses and extensive documentation on these and other aspects of the program can be found on the AFNI website maintained at National Institute of Health (NIH) by Bob Cox, accessible from http://afni.nimh.nih.gov/afni/.

Statistical Parametric Mapping, or SPM [21], was originally developed for Positron Emission Tomography, another imaging technique, and was extended to fMRI. The approach used by this package is voxel based, assuming a parametric statistical model at each voxel. General linear models describe the variability in the data in terms of experimental and confounding effects and residual variability [22]. At each voxel, hypotheses regarding the model parameters can be assessed and images can be created based on the calculated test statistics. There is extensive documentation about SPM at the site http://www.fil.ion.bpmf.ac.uk/spm/.

Other free software analysis tools available are KHORFu and FIASCO (Func-

tional Imaging Analysis Software: Computational Olio). Other then these, there are also some commercial software packages such as MEDx and AIR (Automated Image Registration). But the interesting thing is that most of these software packages have their own default parameters, and researchers just use them for analyzing their data without actually understanding the importance of these. Most of them expect user to know about the basics of fMRI and statistical analysis in advance, so they are not suited for new researchers or students learning about this domain. Although, we are making use of the SPM package for pre-processing of the data in advance for our client-server application, we are incorporating experts' knowledge in selecting the parameters and will teach our audience about the same in the accompanying tutorial.

Another interactive educational tool that exists for fMRI is Dview, a Matlab program developed by Richard Hoge at the MGH-NMR Center [23]. Dview was used in the MIT Health Sciences and Technology (HST) course "Functional Magnetic Resonance Imaging: Data Acquisition and Analysis," or HST.583, in several of the lab sessions [23]. It provides an image viewing tools that allows students to navigate through brain volumes, and a statistical processing module that allows students to perform some simple analyses. A lab manual with a self-paced tutorial accompanied each lab session in the course, guiding students through using Dview to examine and compare various data sets. The development of our web based client/server application is guided from the ideas of this stand-alone tool.

### 2.6.2 Courses and workshops

There also exist semester-long courses on fMRI at various universities [8]. Courses specifically covering fMRI at University of Michigan, University of Waterloo, Medical college of Wisconsin, UCLA, and the Harvard-MIT Division of Health Sciences and Technology (HST) [24] include several lectures covering data analysis and the statistics underlying the analyses.

Another source of fMRI education comes through 1 to 5 day long workshops organized by different institutes. Some of them are like the one-day analysis session at the Oxford Centre for Functional Magnetic Resonance Imaging of the Brain (FMRIB)

[25] to a three day workshop at the Functional Imaging Research Center at the Medical College of Wisconsin [26] with an hour of analysis lecture and a two-hour session using AFNI. The Athinoula A. Martinos Center for Structural and Functional Biomedical Imaging at Massachusetts General Hospital (MGH) holds a 4 day-long course offered 3 times per year that includes multiple lectures and hands on experimental design, data acquisition and analysis (using their own DView Analysis tool) [27]. In addition, conferences sponsored by the Organization Human Brain Mapping (OHBM) [28] and the annual meeting of the International Society for Magnetic Resonance in Medicine (ISMRM) [29] also feature tutorials in fMRI analysis.

While the educational opportunities for fMRI are growing as the courses and workshops spread throughout the country, they have limits on the number of people that can enroll or sign up, and often are quite expensive. The proposed software application and tutorial, on the other hand, will be provided freely to the public, and does not place a time constraint as the courses and workshops do on the attendees.

# Chapter 3

# Educational Goals and Challenges

## 3.1 Introduction

Education in any new emerging area offers a number of challenges to all the constituents of the educational process - students, teachers and researchers. This chapter talks about some of the educational challenges and goals in teaching fMRI through the lens of the "How People Learn (HPL)" framework. New advances in learning science and educational use of technology have provided frameworks for reexamination of instructional paradigm in any domain. So the author makes an effort to explicitly present the goals and motives behind the development of this software application and accompanying tutorial. This development effort is one of the many initiatives taken by the Vanderbilt-Northwestern-Texas-Harvard/MIT Engineering Research Center (VaNTH/ERC) for Bioengineering Educational Technologies, with grants from the National Science Foundation, aiming at improving the short- and long-term outcomes of bioengineering education. The immediate use of this module and tutorial is a semester long graduate course taught at MIT, named as "HST.583 Functional Magnetic Resonance Imaging: Data Acquisition and Analysis". Also, at the same time the software module will be made available freely to entire biomedical imaging community though the support of the BIRN (Biomedical Informatics Research Network [40], www.nbirn.net).

## 3.2 HPL framework

Recent research in learning sciences and review of pedagogical methods have produced many good frameworks for designing and creating an effective learning environment [30, 31]. This innovative project is based on principles of learning within the How People Learn model described in the National Academy of Sciences report "How People Learn: Brain, Mind, Experience, and School" [30]. This framework says that learning can be enhanced if the learning environments are grounded in four basic principles. As described by Harris et al., "the learning environment is [31]:

(a) ***learner-centered*** in the sense that it takes into account the knowledge, skills, preconceptions, misconceptions, and learning styles of the students;

(b) ***knowledge-centered*** in the sense that it helps students learn with understanding by organizing the knowledge around "key concepts" or "big ideas" of the subject domain area, along with understanding the conditions under which different aspects of the knowledge is applicable;

(c) ***assessment-centered*** in the sense that it provides numerous opportunities for students to obtain feedback on their understanding so that it can be refined as needed, and numerous opportunities for a professor to obtain information on students' understanding of material so that teaching may be adjusted as needed; and

(d) ***community-centered*** in the sense that it fosters norms that encourage both students and faculty to learn from one another."

A diagram of these concepts is shown in Figure 3-1.

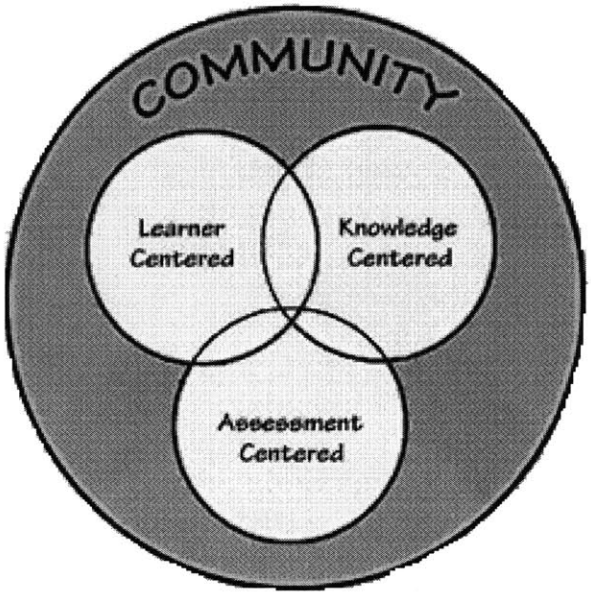


Figure 3-1: The How People Learn (HPL) environment; Source [30]

## 3.3 Enhancements to tutorial motivated by learning theory

Looking through the lens of HPL framework, we are redefining the accompanying tutorial to make the effective usage of this software application in conveying the fundamental concepts of fMRI data analysis. The use of simulation in educational settings is most effective when students are working towards a clear goal, yet the assigned tasks are not too narrowly defined [32]. As a consequence, we are reorganizing the tutorial literature used with the software simulator such that we first of all clearly define the major learning objectives. We also describe detailed learning objectives and key concepts of each of these major learning objectives.

To make a student understand concepts thoroughly, we walk them through guided exploration of the software simulation and create situations where key concepts are presented. Students examine the unprocessed image data, learning about the characteristics of the fMRI signal and then sequentially move through the pre-processing

and statistical analysis steps that enable further exploration of the image data. This is done for different data sets. Students are directed to make comparisons, with emphasis on how processing choices affect the ultimate interpretation of data.

The four major learning objectives explicitly described in the tutorial for the stand-alone version of the same application by Lai, Gollub et at. [2], are:

- "Understanding temporal and spatial correlation in fMRI data;
- Understanding how to construct a statistical model for fMRI data;
- Identifying sources of noise and how they affect fMRI signals; and
- Understanding the effects of motion correction and spatial filtering on the outcome of statistical analysis of fMRI data".

The interactivity of any teaching environment is a very important feature of learning. Interactivity makes it easy for students to revisit specific parts of the environment to explore them more fully and to test ideas, take some decisions and receive feedback. In this case, students can see how different statistical analysis models have different effects on data. Non-interactive environments are much less effective for creating contexts that students can explore and reexamine, both individually and collaboratively. Most of the stand-alone software being used in pre-processing does not provide immediate feedback, because it takes considerable amount of time to process the data. In this tools environment, the user can go back to change the parameters and immediately see the effects on outcomes.

# Chapter 4

# Web-based Client-Server Application

## 4.1 Background

The standalone prototype of the fMRI data analysis application, which was developed by Lai, et. al. [2] in fall of 2002, had some limitations which led to the development of this web based application. The primary goal was to port the tutorial to a platform that would allow the learning tool to be available to the wider scientific community through the World Wide Web. Some of the limitations which became motivations for this development are listed here:

- The standalone demo application required a Matlab license and access to image files
- Since the Matlab based standalone demo uses the image data located in the Athena locker, it could only be accessed through Athena to users with a valid MIT account
- Interactivity of the demo was limited by the size of the image data and the fact that many of the processing steps were actually accomplished "on the fly" by the tutorial software analysis capabilities. So, latency in the data retrieval was an issue

With these issues in mind, Lai [8] started (under the supervision of Dr. Randy Gollub and Dr. Rick Hoge) developing an initial version of the client/server application which could be made accessible to "everyone from anywhere". The first version of the application used the Matlab scripts and Matlab web server on the server side and Java GUI based client on the front end. But the prototype application didn't really solve the problem of the latency in data communication between front-end and backend. It was later determined that the Matlab web server was not efficient, specifically for this application, in handling multiple client requests and processing the requests on the fly for real-time communication. In addition, issues related to licensing of the Matlab software on the server machine created obstacles to broader access. Limitations of the Matlab web server and additional cost due to licensing of the same, forced the author to look for alternative server technologies and web server which could solve these problems and still achieve the goals of the application. Author explains the modified architecture and selection of particular set of platform technologies in this chapter.

## 4.2 Client/Server Application

### 4.2.1 Requirements

The client/server solution should be able to address most of the problems and meet several of the requirements. It should be sufficiently interactive such that users do not experience undue delays when examining and exploring the brain data. The graphical user interface (GUI) of the application should be user-friendly and should display the brain slices/data in a manner that is consistent with standards used in the medical imaging community. The client should be able to run on all platforms such as Linux, Windows, Macintosh, and others. Improvements in the data set selection for the tutorial should be such that there is sufficient contrast in the stimulation and noise sources (such as body motion) so that fMRI data analysis concepts could be taught better.

An interactive application with least latency would require pre-processing of the data and carrying out statistical computations in advance on the server side. It should be the goal to shift as much of visualization work to the client as possible for better performance. For example, constructing the image (of brain slices) from raw data and implementing the autocorrelation function can be done at the client. So the work load of image construction on the server could easily be moved to client with the use of Java imaging technologies [33].

The software application should be made available to majority of researchers across the globe who are interested in learning fMRI. To make the application freely available to public, the selection of technologies should be such that it does not require users to buy a license for using it. For this reason only, the distribution of the standalone Matlab based software was not considered as it requires that the user purchase a Matlab license.

### 4.2.2 Architecture

Various options for selection of architecture were considered before deciding upon a client/server design. One option was to convert the Matlab based prototype to a standalone application which is easy to distribute. Matlab does offer a Runtime Server that allows Matlab programs to be converted to standalone applications [34], but that still does not solve the issue of requiring users to download an enormous quantity of data in the case of standalone prototype. Though, applications developed for use with the Matlab Runtime Server can take advantage of any of the math, language, and visualization features in Matlab and the Matlab toolboxes [34], the solution was not found in consistent with our requirement of not requiring user to buy a license to run the application. Thus it could be desirable to keep data and the GUI interface on separate computers.

Another choice was to develop a powerful, robust, portable, extensible, and open-source Java (v 1.4) application (applet) comparable to a 3D image visualization application developed by Chris et al. [35], but for a different scientific need. The convenient way to visualize 4D medical imaging data set is as three 2D slices through

the same voxel location in the volume and a time series showing the intensity value of the selected voxel. In order to provide remote data access with a level of performance comparable to traditional stand alone applications, the issue of file loading ("I/O"), which is significantly slower, needs to be addressed. In practice, the performance of regular Internet connections is unpredictable; moreover, the transfer rate can vary by up to three orders of magnitude (1000x) among different types of network connections [35]. So, we had two options with respect to how and when to download the 4D image data:

1. ***All up-front:*** all of the data is downloaded and stored in client's memory before the user can view and interact with any of it. This guarantees the best interactive response of the viewer; however, the user has to wait for all the data to download before the client could work, which can be impractical in low bandwidth situation.

2. ***On demand:*** download slice image data only when and if the user wants to view that particular slice. This minimizes the data downloads (and the amount of memory required by the applet/application), but the interactive response time is highly dependent on the server and on the network speed.

Given that we do not expect user to have accessibility to store the huge data-sets required for this application (complete data sets with all pre-computation could require as much as 20- 30 gigabytes),the *all up-front* option was found to be impractical. Also, as we expect the request/response communication over the network to be light weight (i.e. data transferred per call are small in size) for our application, the second option of *on demand* supply of data to the client found more viable and attractive.

Given that data and interface should be decoupled, a client/server model would serve a reasonably good architecture. So author chose to build an application based on client/server architecture: the data reside on the server machine, and a GUI interface on the client side communicates with the server to retrieve appropriate data (brain slices, time series data etc) for the user. How much of the processing should be done on either side was decided based on several constraints and on the goal of maximizing

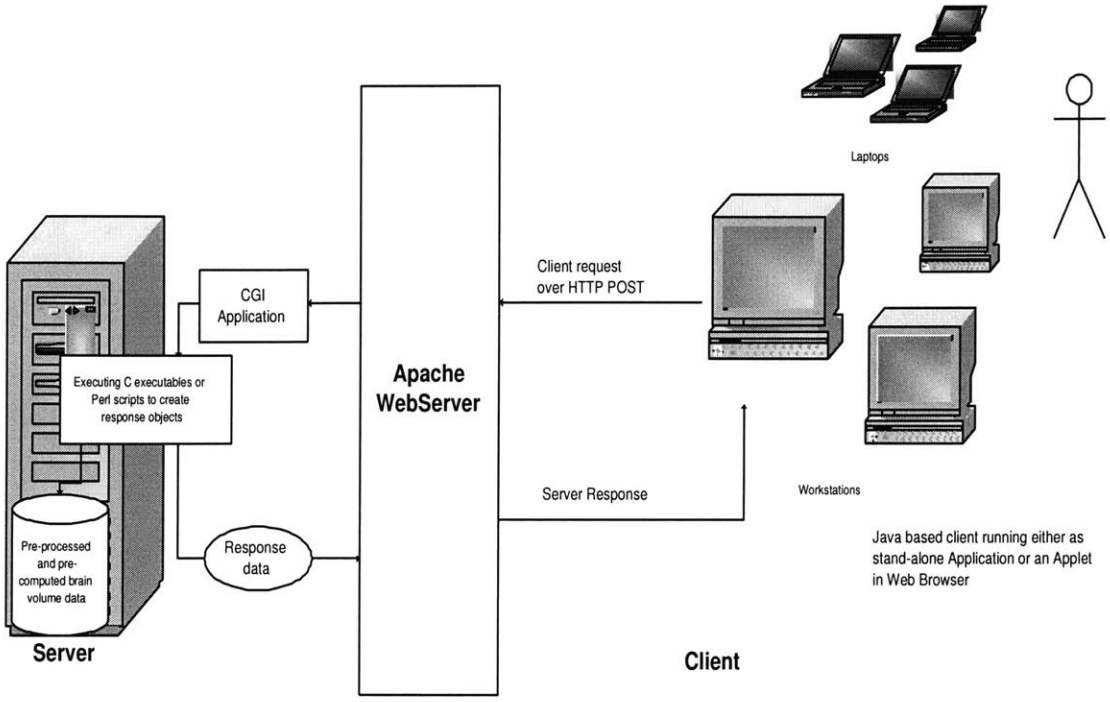


Figure 4-1: Client/Server application architecture

the efficiency of the system. Other reasons which make the Client/Server architecture (as shown in Figure 4-1) appropriate for this application are listed below:

1. As the application requires the user to navigate through multiple data sets (each one of approximately 30 megabytes in size), it is not reasonable to transfer a complete data set to the client. Rather it was decided to store all these large data sets at one common server which could be used by different and/or multiple clients. Also, in a networked environment, shared data should be stored on the servers, rather than on all computers in the system. This makes it easier and more efficient to manage access.

2. Clients would not be responsible for performing any data processing. Clients can concentrate on requesting input from users, requesting desired data from the server, and then analyzing and presenting this data using the display capabilities of the client workstation or the terminal.

3. Substantial Computational Requirements: Even if the data all reside on the server, the amount of disk space and memory required for computation may be too large to expect of the client computer. Also, interactivity might not be good if the bandwidth between the server and client is low; transmitting the data from the server to client would take too much time, given that each brain volume data takes 30 megabytes. Hence computation should be done on the server side for whatever possible client requests. So a dedicated server is needed for substantial computing requirement to process client requests.

4. Client can be designed with no dependence on the physical location of the data. If the data is moved or distributed to other database servers (or file servers) the application continues to function with little or no modification.

In the current architecture we expect the server to perform all computations as desired by the user via the client, and that the server transmits to the client the end result of computations. Here also a similar problem could arise that the server sends the client the entire result which could be anywhere in size from a simple line graph to a four-dimensional brain volume data. Assuming that the user could only see part of the data at a time, because of 2-D presentation of the results in GUI, it is sufficient to keep the result on the server side, and for the client to request the portion that user is interested in viewing from the server. So, the objective is to transfer only as much of data as is needed by the client. A similar approach was also adopted in the Dview [23] (standalone prototype) for displaying its 3- and 4- dimensional brain data, except instead of having the data transmitted over the network from server to client as in this case, it was transmitted from disk to memory. Dview was not loading the huge data set directly into memory, but rather just the portion displayed to user. Thus, the client/server architecture fits with our requirements for location of computation and location of results to be at server and visualization work at client side.

#### Web Server Selection

As discussed earlier, because of the problems with the Matlab web server in terms of efficiency and interactivity in the previous version of this application, we evaluated other available web server options. The Apache web server was found to be best suited to our application requirements. However, the Apache configuration was modified to our needs. Apache is considered as one of the most popular web servers since April 1996 [36], as it is robust, could be deployed easily and is available for free.

### 4.2.3 Platform technologies

#### Server Side

Many options were considered for the server platform before finally deciding upon one. One option was to keep server as a complete Matlab based solution for both computation and Web-serving. This approach would allow much of the old standalone demo code to be readily used, but would require modification of the output data into a form the client accepts. The major disadvantages of this option include limitations of the Matlab Web Server and dangers of code reuse because of few unknown bugs in the final version. In the environment of Matlab Web Server, input variables are submitted to the server though URL-encoded form data through the HTTP POST request, and the server then computes the output and sends back the output HTML document as the result [37]. This is much the same way any standard web server, such as Apache, handles client requests. An option similar to this one was adopted by Lai [8] in the prototype version except that some of the data was pre-preprocessed and pre-computed. Another major issue with this option was getting a license for running Matlab on the server machine.

Another option was to move completely away from Matlab and to use another programming language, both for computation and serving the data to the client. The major advantage of this approach is that code could cleanly be built from scratch and that it provides an opportunity to get rid off some of the bugs in standalone prototype code. The language of choice could be required to have basic toolboxes for

building simple web server or it should be able to provide web serving functionality by using standard web server, such as Apache, invoking CGI script to execute the server programs. The major limitation of this option was that common languages such as Java and C do not have Matlab's matrix-manipulation facilities, which would have to be written before server can function, and would take a fair amount time to develop. Also, the standalone version was taking advantage of libraries for reading and processing data files in the common MINC (Medical Image NetCDF, is a file format for medical imaging data) format, which according to MINC documentation does not exist for languages other than Matlab, C, and Fortran [38], and conversion routines would still have to be written to bridge the gap from the MINC library output to a format that a C library would accept.

The third option considered includes complete pre-processing and computation for statistical analysis of the data in advance. To achieve that, Matlab/Perl scripts or already existing software packages could be used. And then, the main effort would consist of writing the actual server code in Java or some other language. This approach could preserve the Matlab computational facilities and the advantage of building the server from scratch with no bugs. Depending on the format of output file generated after running Matlab scripts or another fMRI data analysis software package on the data, this option would require additional libraries for reading the pre-computed data for the client request. The one disadvantage of this approach is that it would generate at least an order of magnitude more data than used in standalone prototype where many of the computations were done on request in real-time. Given enough disk space on the server this might not necessarily be a problem. The fact that there were no written libraries in existence for Java to read the pre-computed data, and it was not possible to write these libraries in the given time frame, the choice was made to choose another language to replace Java in this option.

Ultimately, the author chose to use a hybrid of the second and third options. Keeping the interactivity of the client in mind, most of the data for time-consuming processing steps and statistical analysis were pre-computed. The Apache Web Server dispatches CGI/Perl scripts to fetch the pre-computed data by running some of the

conversion routines written in C and Perl, performs any necessary minor computation such as coordinate transformation etc, and returns the output to the client. More detailed design of the server is presented in Chapter 6.

### Client Side

Some of the options in selecting client side technologies for the application were being discussed by Lai in his report [8]. Author summarizes some of his thoughts and presents additional ideas for the final selection of Java 2 Platform. Initially, standard HTML forms supplemented with some JavaScripts were considered adequate for the client. It was originally believed by the author that interaction between user and program does not go far beyond viewing static images and plots. However, we subsequently recognized the need for a very interactive client where the user could navigate through time-series brain data with the ability to click on the time-series plot and/or the three ordinal views of the brain (transverse, sagittal, coronal). An intermediate solution of having a Java applet specifically for displaying interactive plots was explored by Ian [8] where the server would transmit the data required for the plot, and the applet would take care of plotting the data on client side. But the communication between the applet and the HTML forms containing the processing steps and brain images was found to be too cumbersome. This is because, at present, Java and HTML can be a difficult combination due to limitations in how Java can interact with the Web browser [39]. Ultimately, it was decided that having entire client written in Java would offer more power, robustness, flexibility, and modularity in development. Finally, the client will be distributed or deployed either as a Java Applet or a regular Java Application. Since the client is expected to be used long term, the latest version of Java platform (Java 2 Platform, standard edition 1.4) was chosen, with the assumption that most Web browsers would support it soon.

# Chapter 5

# Detailed Client Design

The client of the Web-based application is markedly different from the stand-alone prototype in the user interface and design. This chapter covers details of the user interface of the client, the main data abstractions used, the architecture, and the package structure. The final client architecture and interface has evolved from a prototype version developed by Lai [8] to a fully functional robust application which can be deployed to multiple platforms in heterogeneous, distributed networks.

## 5.1 Design Goals

The design requirements of the programming language for the client were driven by the nature of the computing environments in which this software application would be deployed. We expect our users to run this application on multiple platforms, so Java was the obvious choice of development language for its reasonable performance and portability for a GUI intensive application like this. Other design goals for the client development were:

- Applications should be designed to offer more interactive and responsive user interfaces than HTML clients running inside a web browser.
- Developing the application under the umbrella of layered architecture for its flexibility to future changes.

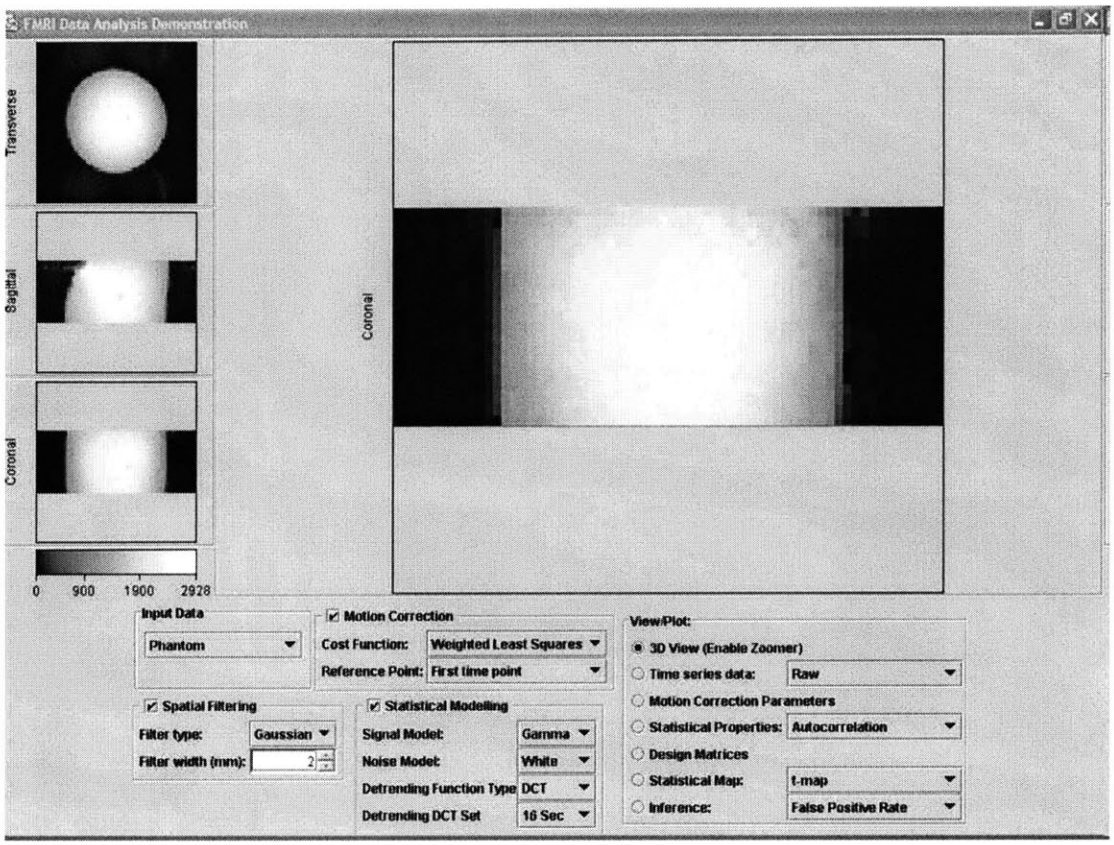


Figure 5-1: Screenshot of Web-based Java Client for fMRI Analysis of Phantom data (3D View)

## 5.2 User Interface

The user interface for the Web-based fMRI Data Analysis client is based on that of the standalone prototype, with some important differences. First, the final client integrates the viewer and the panel of processing parameters so that the user does not have to switch his attention between windows. Some of the features in the final GUI of the application are just extension to the basic features implemented in the prototype application developed by Ian [8]. Additional features such as color map legend for images, zoomed and mosaic view of individual cross sectional slices are implemented from scratch in the final client. Screenshots of the integrated interface are shown in Figure 5-1 and Figure 5-2 with different data sets.

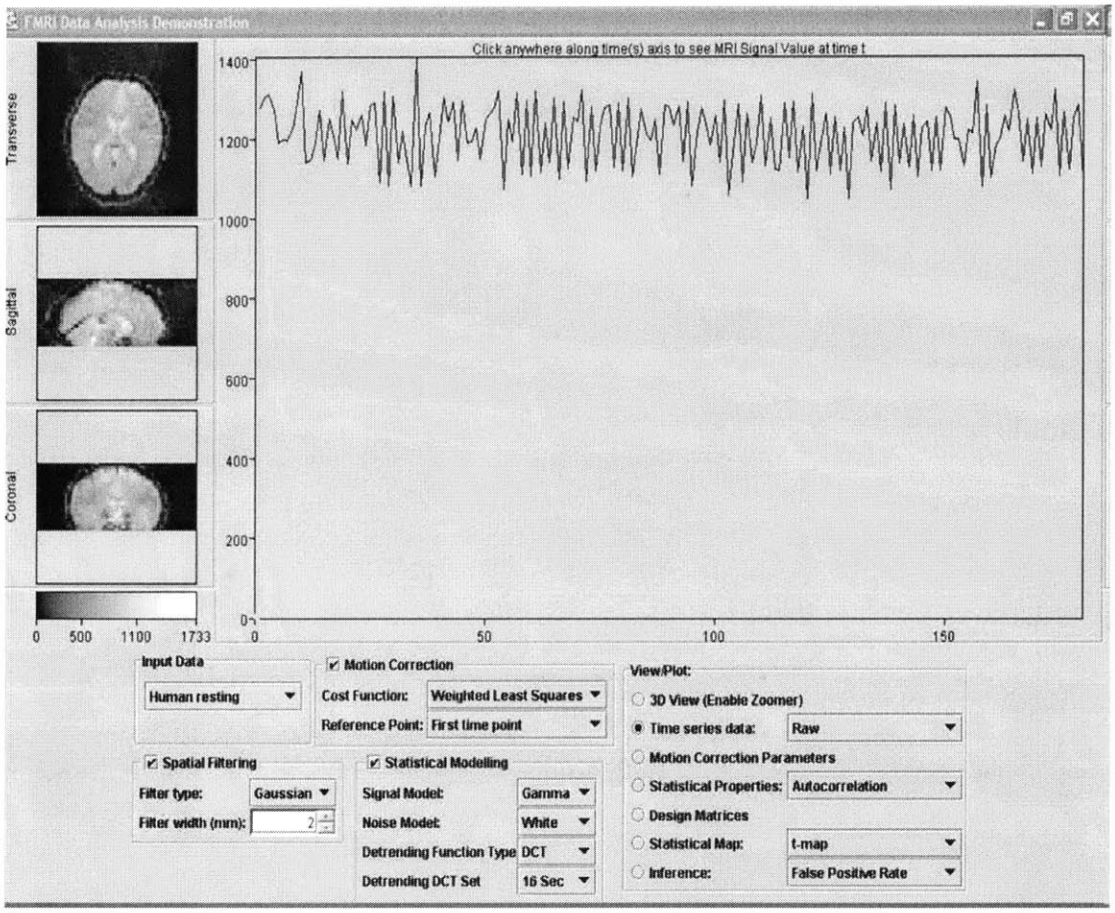


Figure 5-2: Screenshot of Web-based Java Client for fMRI Analysis of human brain data (4D View)

The viewer has a large panel to the right of the transverse, sagittal, and coronal cross-sectional slices, in which it displays either a zoomed slice of the image or the time series values for the selected voxel in 4-dimensional data or the autocorrelation function of a selected voxel. However, in addition to integrating the viewer and the panel, the final client also displays the design matrix and the motion correction parameters in the same window. When the user requests a plot or the design matrix, the image data is replaced by a panel displaying the requested data.

In the standalone prototype, the processing parameters control the computations performed on the data, and the various View and Plot buttons at different stages of processing display the data as processed up to that particular stage [8]. However, since nearly all of the data in the Web-based application is pre-computed, the need for the user to press a button to view the data after changing parameters was found unnecessary, instead the response should be close to immediate. Also, the user should be able to enable or disable a processing step or alter its parameters, and see that the data is changed automatically. Therefore, in the final client, the View and Plot buttons have been removed, and replaced by a list of different views to choose from:

- Time Series Data (raw, fitted, or hemodynamic response)
- Zoom View (Zoomed view of either of the slice in large panel area)
- Motion correction parameters (6 parameters: 3 translational and 3 rotational)
- Autocorrelation function (autocorrelation function estimated from the time series data of a selected voxel)
- Statistical properties (t-value maps or standard deviation map)
- Design matrices (displays design matrix and a stimulus covariance matrix for a given experiment paradigm)

Since each view may require certain processing steps to be turned on, the view selection defaults to raw data if turning off a processing step causes the view to become unavailable. For instance, if statistical modelling is turned off, the statistical

properties are no longer relevant; if the statistical properties were previously selected, the display changes to show raw data instead.

## 5.3 Data Abstractions

Data abstraction groups the pieces of data that describes some entity (in this case processing/view parameters), so that programmers can manipulate that data as a unit. It helps programmer to cope with the complexity of the data as it hides the details. This section talks about the data abstractions used in the current application because of its importance in flexibility to construct or change a piece of software.

### 5.3.1 View Parameter

**ViewParameters** class represents parameters chosen for viewing data in the client. The ViewParameters object represents whether a processing step is enabled, which processing parameters are selected (if a processing step is enabled), and which view is currently requested by the user. It has matching get and set methods for each of the parameter. Also, it has methods for toggling and indicating whether each processing step is enabled or not.

### 5.3.2 Graph Data

A utility package of graph data abstractions provides a mechanism for communicating the details of various graph data sent from the server to the client. For simplicity's sake, each of the graph data objects is immutable (an immutable object is an object which has a state that never changes after creation). An important reason for using an immutable object is that other objects can trust its content to not unexpectedly change.

The abstract **GraphData** object represents an arbitrary n-dimensional graph with axes and an optional title. Each **Axis** spans a predefined **Range** and can have an optional label. The optional tick marks and labels of each axis can be specified

in the server response or automatically computed using an implemented algorithm. In the current implementation, we chose not to print tick marks and labels on slices because it creates more space for actual image to display, though we have used them for plots. Since the plots used in the client are all two-dimensional, the objects representing them all derive from **Graph2DData**, which specifies a horizontal and a vertical axis.

**PlotData** represents a line graph with one or more data series plotted on the same axes. Each **DataSeries** encapsulates a list of **GraphPoint** objects, an optional **GraphStyle** specifying the color, and an optional label that may be used for a legend.

**MapData** represents a two-dimensional image map. An **ImageMap** object specifies the actual image and its dimensions, and is translated on the graph to a specified point.

The **BrainPoint** object represents a point in brain volume data, with or without a time dimension. Basically it represents x, y, z or/and t coordinates of the selected voxel in brain volume data depending upon if it is 3D or 4D data.

## 5.4 Layered Architecture

The user's selection of different processing steps and navigation of brain data is generalized to a set of parameters (implemented as ViewParameters). The client sends these parameters to the server as a request for data. After receiving a response from the server, the client then displays the data in a manner appropriate to the data returned. This view of the client's responsibilities suggests an architecture consisting of layers of interaction between the user and the server. Though, more centralized architectures were also considered for the client, but the layered architecture seems to give the most flexibility for future to the user interface and server.

The client consists of three different layers, the Graphical User Interface (GUI) Layer, the Main Layer, and the Data Retrieval Layer, as shown in Figure 5-3. Figure 5-4 shows a detailed view of the communications between layers of the client and can be used for reference.

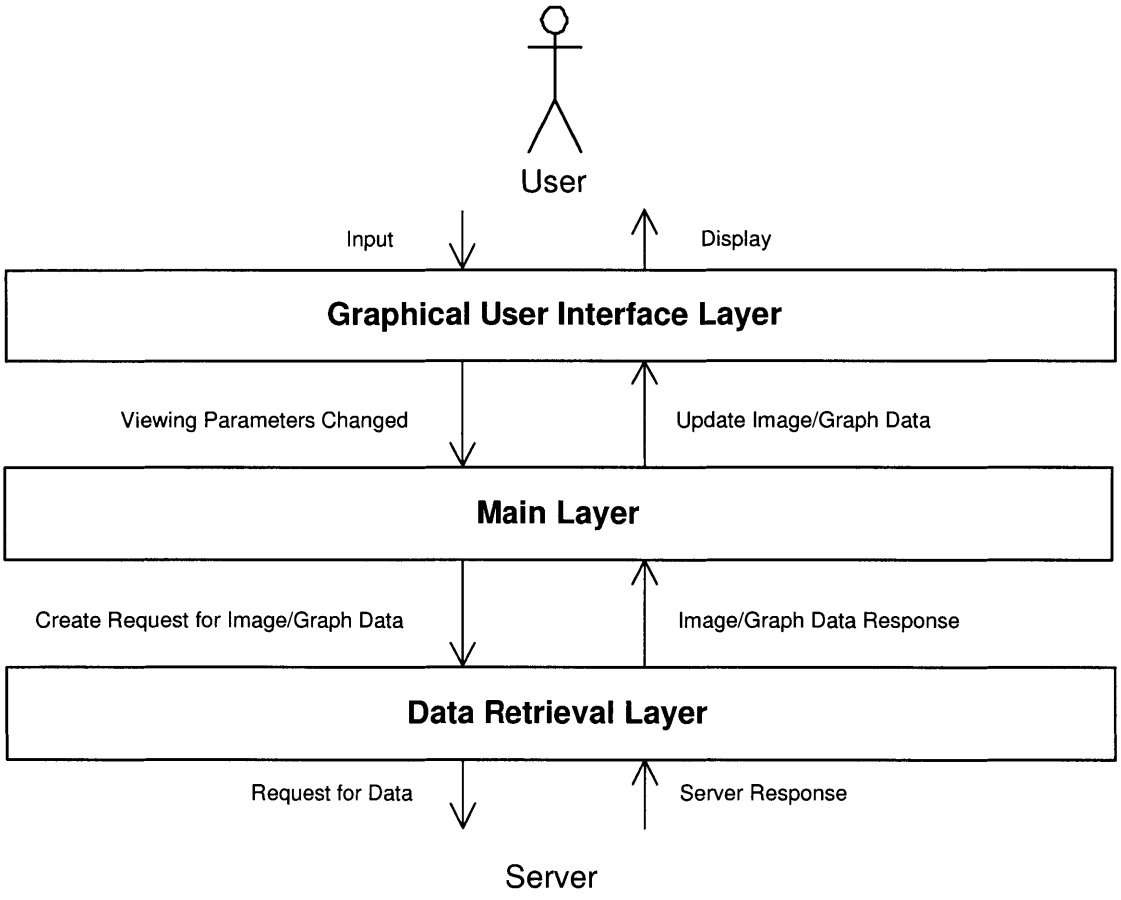


Figure 5-3: Layered architecture of web bases Java client

### 5.4.1 Graphical User Interface (GUI) Layer

As the main interface to the user, the GUI Layer displays two-dimensional brain slices and plots based on the user selected parameters. Major functions of GUI layer include providing mapping between UI (AWT/SWING) objects and more convenient data object, processing UI events, updating display and interacting with next layer down. To make the user interface easily modifiable, the GUI Layer is decoupled from the Main Layer via several Java interfaces.

#### Interfaces

The Main Layer communicates with the GUI Layer via *update methods* that instruct the user interface to refresh its data. To receive notification that the user has clicked on a plot or changed a parameter, the Main Layer registers itself as a listener of events caused by the GUI Layer.

Each interface of the GUI Layer can register and un-register listeners that are interested in its events via the *addListener* and *removeListener* methods, and return a listing of listeners via the *getname-of-interfaceListeners* method. The *finishUpdates* method informs the GUI Layer that the updates for that interface are completed, and that the given listener is ready for events again.

The alternative to having an interface for each of the possible viewers is to have a single monolithic interface for the entire GUI Layer, consolidating all of the update methods. While this approach is not altogether undesirable, the division of the interfaces into separate types of viewer seems logical because of modularity and flexibility for future development.

The list of interfaces for the GUI Layer is described below, along with the interface-specific update methods and the methods used to notify their listeners of events.

- **Brain2DViewerGUI** — The Brain2DViewerGUI represents a user interface for a standard brain viewer, using two-dimensional cross-sectional slices as the main navigational tool. In addition, it should be able to display a time series for a four-dimensional brain data set. It must implement the update methods

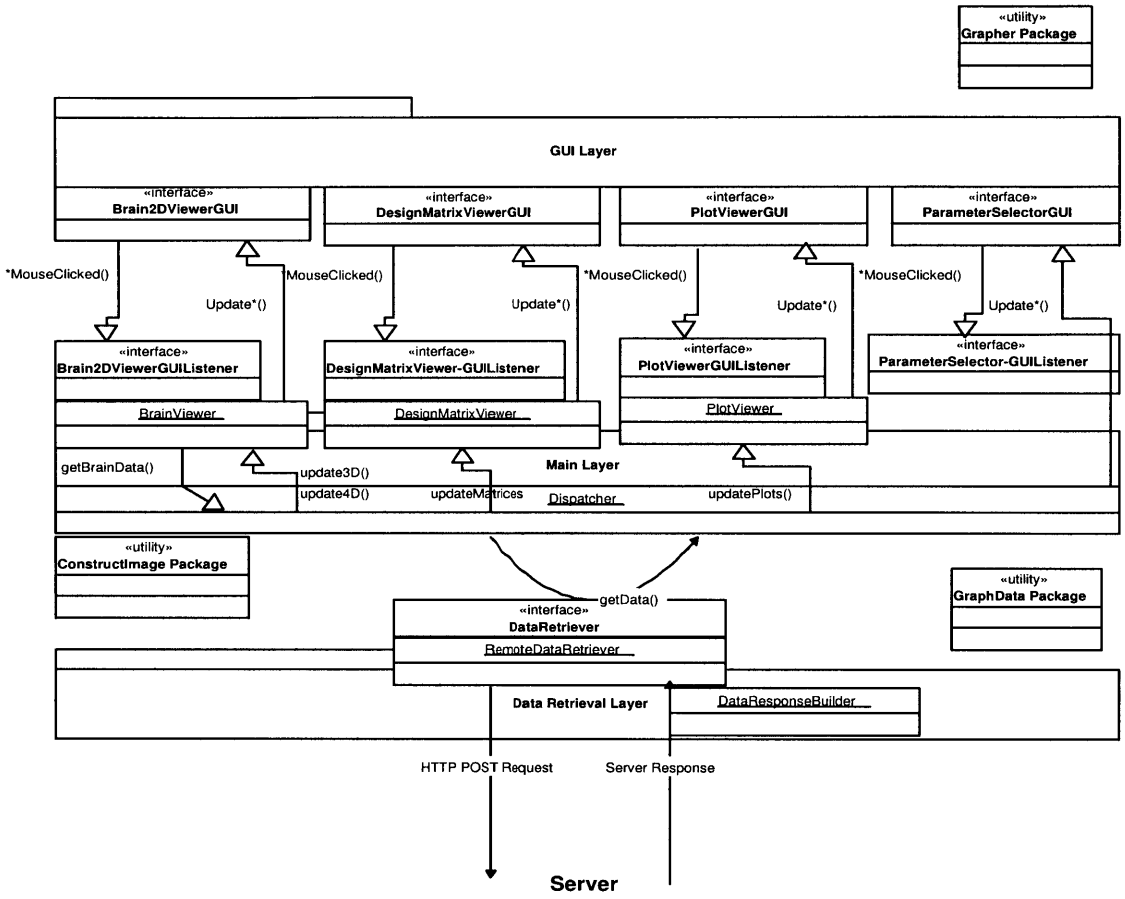


Figure 5-4: A detailed view of the communications among layers in the Web-based Java client. Interfaces are labelled as "<<interface>>, with their accompanying implementations in adjacent boxes. Method calls are represented by thin arrows; the one method which returns data, *getData*, loops back to the caller (called as **Dispatcher**). The arrows between the GUI Layer and the Main Layer represent the various *MouseClicked* and *update methods*. See Section 5.4.1 for the list of methods. Utility packages are labelled as "<<utility>>and are used by most of these layers.

| Method | Description |
|---|---|
| *updateSlice* | Instructs the brain viewer to display/update the given slice with the new data |
| *updateTimeSeries* | Instructs the brain viewer to display the given time series data of the voxel under cursor |
| *updateMosaic* | Instructs the brain viewer to display a given mosaic of brain slices |
| *updateColorMap* | Instructs the color map viewer to display the dynamic range color map for the brain slices |
| *updateZoomView* | Instruct the large panel to display the zoom view of last slice clicked |
| *mosaicViewRequested* | Invoked when the user requests a mosaic view of the given cross sectional view. The type of view - sagittal, coronal, or transverse - should be provided |
| *sliceMouseClicked* | Invoked when the user clicks on a cross section to move the cursor in space. The coordinates for the point clicked should be provided |
| *timeSeriesMouseClicked* | Invoked when the user clicks on the time series plot to move the cursor in time. The coordinates for the point clicked should be provided |

Table 5.1: Brain2dViewerGUI methods

and required to call the corresponding methods of its registered listeners on the listed conditions (methods details are listed in Table 5.1). For each of the methods that require the point clicked, the point corresponding to the axes of the graph displayed should be returned, rather than the physical screen pixel coordinates.

- **DesignMatrixViewerGUI** — The DesignMatrixViewerGUI represents a user interface that displays a design matrix and a stimulus covariance matrix for a given fMRI experimental paradigm. As described in chapter 2, the design matrix is a visual representation of the regressors used in the statistical analysis. It must implement the update methods and methods of their listeners on the listed

| Method | Description |
|---|---|
| *updateDesignMatrix* | Instructs the design matrix viewer to display the given design matrix |
| *updateRegressor* | Instructs the design matrix viewer to display the given regressor |
| *updateStimCovMatrix* | Instructs the design matrix viewer to display the given stimulus covariance matrix |
| *designMatrixMouseClicked* | Invoked when the user clicks on the design matrix to request the display of the regressor corresponding to the vertical column of the matrix |
| *sliceMouseClicked* | Invoked when the user clicks on the regressor |
| *stimCovMatrixMouseClicked* | Invoked when the user clicks on the stimulus covariance matrix |

Table 5.2: DesignMatrixViewerGUI methods

conditions (methods details are listed in Table 5.2).

- **PlotViewerGUI** — The PlotViewerGUI represents a user interface for displaying one or more two-dimensional line graphs. PlotViewerGUI is being used to display time-series data, motion corrected parameters and the autocorrelation function. When the user clicks on a plot, the PlotViewerGUI is required to call the *mouseClicked* method of its listeners, providing the coordinates of the point clicked and which plot the user clicked on. It must implement the following update methods:

  - *updatePlot*: Instructs the plot viewer to display the given plot with data series
  - *updatePlots*: Instructs the plot viewer to display the given list of plots in some appropriate visual arrangement

- **ParameterSelectorGUI** — The ParameterSelectorGUI represents a user interface for selecting view parameters. It must implement the *updateParameters* method, which instructs the parameter selector to change its display to reflect

that the given view parameters. When the user has changed the view parameters, the ParameterSelectorGUI must call the *viewParametersChanged* method of its registered listeners (which is mainly Dispatcher in this case).

#### Implementation of Viewer Interfaces

The implementation of each of the viewer interfaces simply consist of standard Swing panels, with different graphers (described in the next section) tiled to display the brain slices and plot data. A **StandardGUI** object provides a frame divided into two parts, an area where appropriate viewer for the user-selected view is displayed, and the area for the parameter selector, which simply comprises a collection of menus and buttons (Java swing objects) for the different view parameters.

#### Grapher Utility Package

A package of graphing tools was developed to simplify the implementation of the viewer interfaces. The packages consists of a **LineGrapher**, which plots line graphs, such as the times series of brain data and the motion correction parameters, and a **MapGrapher**, which plots image maps, such as the cross sections of the brain and the design matrix. Each grapher can optionally display a cursor within the axes, and uses the standard Swing event mechanism for notification of user input. The graphers accept the graph data objects described in Section 5.3.2. Because of its generality, the grapher package can be readily utilized by any other user interface designed to sit in the GUI Layer. An abstract superclass **AxisGrapher** is also provided for creating any other grapher that plots on a set of horizontal and vertical axes. Another utility package, called **ConstructImage**, was developed to build images from the array of pixel data using Java imaging and foundation classes.

### 5.4.2 Main Layer

The Main Layer constitutes the center of the client as it holds the state of the client, communicates with the GUI Layer to update the display, and sends data requests to

the Data Retrieval Layer. Major functions of main layer includes providing methods for interacting with the server object, mapping between basic data object and objects to be sent to/from server and interaction with the Data Retrieval Layer. The **Dispatcher**, **Brain2DViewer**, **PlotViewer**, and **DesignMatrixViewer** are main components of the Main Layer. Main Layer holds the core of the client logic, and is capable of serving and communicating with different implementations of the GUI and Data Retrieval Layers. Main layer can be think of as middleware component in a 3-tier architecture which drives the application in both directions.

- **Dispatcher** — The Dispatcher serves as the main control of the application. It keeps track of its contacts in the Data Retrieval Layer and the GUI Layer, as well as the viewing parameters chosen by the user. This is the key class which is being called whenever any of the Viewers need some data from the server. Dispatcher communicates the request to Data retrieval layer by launching a separate thread for each request call. An alternative design considered for current Dispatcher was to keep the back-end for the parameter selector into a separate entity, which would have the exclusive channel of communication with ParameterSelectorGUI and would communicate parameter changes back to the Dispatcher. Since the view parameters are an integral part of the state of the client and are also necessary for the Dispatcher's retrieving data on the Brain2DViewer and PlotViewer's behalf, it seemed the Dispatcher should simply assume the role of the back-end. Also, at the same time, Dispatcher can maintain central control of the view parameter information and communicate it to upper layers.

- **Brain2DViewer** — The Brain2Dviewer is corresponding back-end in Main Layer to the Brain2DViewerGUI in GUI Layer. It keeps track of the current position of the cursor in the brain, and passes requests for brain data to the Dispatcher when the user navigates through the brain slices or time series data. Finally, when the server response returns with the data, the Brain2DViewer calls appropriate update methods to refresh the Brain2DViewerGUI.

- **DesignMatrixViewer** — The DesignMatrixViewer, serving as the back-end to the DesignMatrixViewerGUI, keeps track of the design and stimulus covariance matrices (used in the experiment paradigm) being viewed and the regressors used in fitting the statistical model. It notifies the GUI to switch to a different interface when the user requests the design matrix view. When the user clicks on the design matrix, the DesignMatrixViewer fetches the regressor corresponding to the column that the user clicked, and instructs the GUI to display it.

- **PlotViewer** — As the back-end to the PlotViewerGUI, the PlotViewer keeps track of the current plots requested by the user, and passes the plots (after building them from raw data from the server using several of utility packages) to the PlotViewerGUI for display.

### 5.4.3 Data Retrieval Layer

The Data Retrieval Layer is client's interface to the server to handle all the communication between them. It sends the data request to the remote server using HTTP POST and returns with the response data from the server. So, the major function of Data Retrieval layer includes managing connection with the server, providing I/O from/to the server through the connection and finally to read the data from input stream in a desired manner.

**DataRetriever**, is the main interface of this layer. The one method, *getData*, accepts a data request in the form of a **DataRequest** object, and returns the response from the server in the form of a **DataResponse** object. Depending on the type of data returned, the DataResponse could either be a **BrainDataResponse**, a **DesignMatrixDataResponse**, or a **PlotResponse**. An alternative DataRetriever interface that was considered consisted of multiple methods for getting different types of data, instead of having a single getData method. But since view parameters contain all of the information needed for a request, it was decided that one method would be sufficient, and would offer more flexibility the case if more data types arise in the future.

**RemoteDataRetriever** (the current implementation of the data retriever), converts the provided DataRequest from the Main Layer into an HTTP POST request and sends it to the remote server. It extracts the response data from the server reply, of content-type: application/x-www-form-urlencoded, and builds plot/image from the corresponding key-value pairs and raw byte data for image. The appropriate type of DataResponse is created and returned for the graph data received. It makes use of the ConstructImage (see section on utility package) module for constructing the cross-sectional slice images and dynamically changing color map from the raw byte data.

As this layer is decoupled from the Main Layer via a simple interface, different versions of the Data Retrieval Layer can be used for different types of servers (separate implementation for access to local and remote servers).

## 5.5 Package Structure

The client takes advantage of Java packages to divide the classes and interfaces into logical groupings. Table 5.3 enumerates the packages in the client and gives a description for each package.

| Package | Description |
|---|---|
| edu.mit.hst583.fdad | Main package for fMRI Data Analysis client |
| edu.mit.hst583.fdad.client | Package for client application |
| edu.mit.hst583.fdad.client.gui | Package for client Graphical User Interface (GUI) layer |
| edu.mit.hst583.fdad.client.gui.standard | Standard implementation of client GUI |
| edu.mit.hst583.fdad.client.gui.grapher | Grapher utility package |
| edu.mit.hst583.fdad.client.main | Package for client Main Layer |
| edu.mit.hst583.fdad.client.data | Client Data Layer |
| edu.mit.hst583.fdad.lib | Package for shared data structures between client and server |
| edu.mit.hst583.fdad.lib.data | Package for client/server communications objects (requests and responses) |
| edu.mit.hst583.fdad.lib.graph | Package for graph data objects |

Table 5.3: The list of Java packages and their description for the Web-based client

# Chapter 6

# Detailed Server Design

The server of the Web-based application was completely redesigned and implemented from scratch with the aim to improve performance over the prototype version running with Matlab server. The final server consists of C executables and CGI/Perl scripts invoked by the Apache Web server running on server machine. The server architecture, pre-processing and communication protocol are described in this chapter.

## 6.1 Server Architecture

The **FMRIDataAnalysisDemoServer** script serves as the interface to the client. It calls the request parser to extract the parameters from the input, then calls the coordinate transformer to change client coordinates (x, y, z, t) to brain volume coordinates (row, column, slice and frame), dispatches the corresponding data generator to compute or process pre-computed data, and passes it to the response generator to output the reply to the client in the proper format. A diagram of the server architecture is shown in Figure 6-1.The subsequent sections detail each of the steps above.

### Data

As discussed earlier, all the data used for teaching through this tutorial are pre-processed in advance and are stored in a UNIX file system based database on the

server. MINC is used as the medical imaging file format for all the data used in this application. MINC, stands for Medical Image Net CDF, is a medical imaging oriented extension of the NetCDF (Network Common Data Form) file format which is widely used in many areas of scientific data storage. The main advantages of MINC are its support for coordinate systems, extensibility, self-description, and portability across different computer platforms. With a few simple tools, it is extremely easy to read in both the data and descriptive information about the data. So, we have written a C program, know as MINCRead, which outputs desired data from these MINC files for the client requests.

### Request Parser

The script *GetRequestParameters* checks the validity of the client input and generates a perl structure containing the parameters as requested by the client. As the Apache Web Server, with CGI.pm module installed, automatically creates a structure for input variables, in the implementation the parser does a relatively simple conversion between fields in the input structure and fields in the parameter structure. CGI.pm, a perl 5 library, provides a simple interface for parsing and interpreting query strings passed to CGI scripts. These extracted parameters from the client request are used to produce the filename to be used for data generation for the client.

### Coordinate Transformer

The script *XYZTtoRCSF* converts the x, y, z, and t coordinates from client request to the coordinate system used in the MINC data file. Each pixel on a 2D client GUI can be mapped to a voxel in 3D brain volume data by knowing the corresponding row, column and slice number in the data. The script uses a binary called **mincinfo** for getting information about the image data file such as step sizes, orientation of image etc, required for the coordinate transformation.

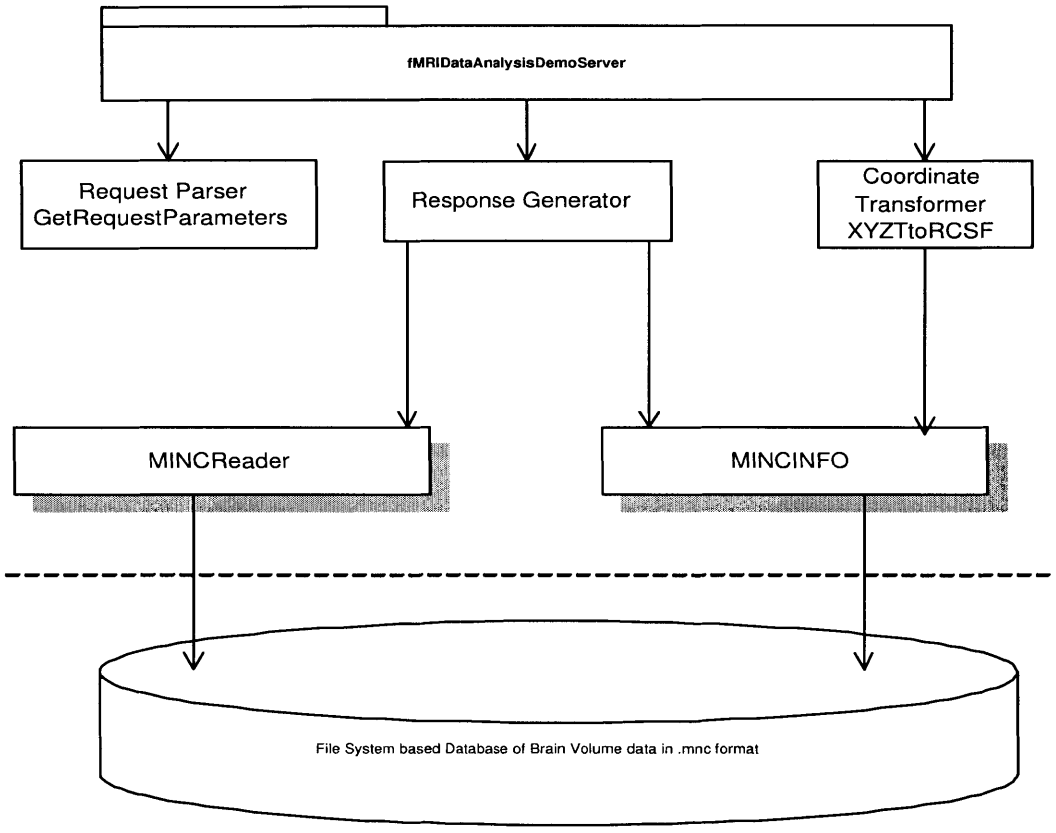


Figure 6-1: The architecture of the Web-based fMRI Data Analysis Server. Boxes with shadow represent a C executable and the rest are CGI/Perl scripts, and each arrow indicates a call from one script to another.

| Output | Command Syntax |
|---|---|
| Whole series to stdout | MINCRead file.mnc |
| One slice | MINCRead -slice <index> -frame <index> file.mnc |
| One cross-section along row | MINCRead -row <index> -frame <index> file.mnc |
| One cross-section along column | MINCRead -column <index> -frame <index> file.mnc |
| Signal at specified indices | MINCRead -row <index> -column <index> -slice <slice> -frame <index> file.mnc |

Table 6.1: A list of usage statements for MINCRead program

### Response Generator

Response Generator is a collection of perl scripts and subroutines to create a response object for the client. Depending on the type of view request from client, it generates the data, by executing MINCReader and other programs, in a format which the client expects. For instance, if "Time Series Data" view is requested then the response generator calls the corresponding subroutines responsible for producing the data for 3 slices (transverse, sagittal, and coronal) and the time course for the selected voxel.

### MINCReader

*MINCRead* is a C program capable of generating raw byte data for brain image slices (cross-sections of transverse, sagittal, and coronal) and the time series values of any selected voxel in the brain. MINCReader is called from different Response Generator scripts with different parameters as arguments depending upon the view requested. This program basically opens the MINC file specified on the command line and writes bytes containing either an image or signal to standard output. The usage statements of the program are described in the Table 6.1.

## 6.2 Precomputation of Data

As discussed earlier, in the interest of increasing interactivity, most of the data that was computed on the fly in the prototype is pre-computed for the server. This includes not only the motion-corrected and spatially filtered data, as was the case with the prototype, but also the standard deviation maps, a number of fitted statistical models, and the corresponding statistical maps. Data that can be trivially computed either at the client side or by Perl scripts on the server, such as the autocorrelation, and plots that can be extracted readily from three-dimensional brain data, such as the histogram of t-values, were not considered for pre-computation.

The amount of pre-computed data generated as a result of numerous parameters (signal model and different noise models, each with multiple options to chose from) does not really pose a problem. This is because the three dimensional maps lack the time component, and the fitted models only require at most the time data between successive stimulations in the experiments [8]. Also, given declining storage costs, we don't see data storage a problem on server machine, so interactivity was given a priority over the size of the data sets generated by pre-computation.

Since most of the data on the server side was pre-computed, it removed the need for Matlab on the server side for computation. Thus, pre-computation allowed us to shift to a server solution completely independent of Matlab.

## 6.3 Communications Protocol

As the client-server application requires transfer of document and other data from server/client to client/server, HTTP was chosen as the base protocol for communication and Apache as the HTTP Web server. The Apache Web Server can handle input in URL-encoded form data in an HTTP request using the POST method, and can return the output in a desired format [40]; the modules for communication between the client and server were designed keeping these things in mind.

| Key | Possible Value |
|---|---|
| data_set | resting, phantom, exp 1, exp 2, exp 3 |
| motion_correction | true, false |
| mc_cost_function | wls (stands for weighted list squares) |
| mc_reference_point | first time point |
| spatial_filtering | true, false |
| sf_filter_type | gaussian |
| sf_filter_width | 2, 4, 6 |
| statistical_modelling | true, false |
| stat_signal_model | gamma, FIR |
| stat_noise_model | white, AR1 |
| dt_function | dct (stands for discrete cosine transform) |
| dt_dct_set | 16 sec, 32 sec, 64 sec,128 sec, 256 sec, Inf |
| view | viewZoom, raw, autocorr, hdr, fitted, std dev map, t map, p map, t histogram, mc params, matrices |

Table 6.2: The keys and the valid corresponding values for the client request

### 6.3.1 Request Protocol

Since the client request is submitted in the form of URL-encoded form data, it consists of a set of keys and values corresponding to the view parameters desired by the client. Table 6.2 lists the different keys and their possible values. Some of the these keys have only single option, so the interesting question is why to have them at all. The reason is to keep the request protocol as much generic as possible so that it can be extended in future easily. All of the keys must be present, and all keys must have legitimate values, for the request to be considered valid by the server; otherwise the server will respond that the request is invalid.

### 6.3.2 Response Protocol

The response as returned by the Apache Web Server is in the form of a document comprising a mixture of text data and raw byte data for image/graph. For simplicity,

the server response first has plain-text data consisting of key-value pairs that represent the image/graph data, then followed by actual data for the image to be displayed by the client. Figure 6-2 shows a sample response from the server.

### Syntax and semantics of server response

The body of the server response consists of lines with a key-value pair, separated by an equals sign (=), and the raw image/graph byte data for image or graph to be built from at the client side. The key-value corresponding to dimension of the image data (for e.g. transverse-image-dim = 64 64) specifies the number of bytes to be read from the response for a particular type of image or graph.

A key consists of a string of alphanumeric characters or hyphens (-). Keys are case-sensitive. A value consists of a list of strings or numbers, separated by white space; a string is delimited by double quotation marks ("), with the backslash as the escape character. Each server response should have the key data-type, which specifies the type of data included in the response. The current possible values are brain-4d (for time series and autocorrelation views) and brain-3d (for t maps, standard deviation etc) for four- and three-dimensional brain data; matrices, for the design and stimulus covariance matrix; and plot, for one or more two-dimensional line plots. The client should ignore any keys that it does not recognize.

The key-value pairs that specify each graph (line plot or image map) have keys with the name of the image/graph as the prefix, with the suffixes in Table 6.3 expressing the different properties of the graph. For instance, if the server is returning image data called transverse, the title would have the key transverse-title, and the vertical/horizontal range would have the key transverse-vert-range/transverse-horz-range.

### Required Data

For brain-3d, the server is required to return the raw image data of transverse, sagittal, and coronal for the cross sections. In addition, the cursor coordinates of last click should be returned with the keys x-coord, y-coord, and z-coord. In the case of brain-

```
Content-type: application/x-www-form-urlencoded

<!-- START FMRIDATAANALYSISDEMO DATA -->

data-type = brain-4d

transverse-image-dim = 64 64
sagittal-image-dim = 64 21
coronal-image-dim = 64 21
time-series-points = 180

transverse-horz-image-range = 0 252
transverse-vert-image-range = 0 252
.
.
.

<!-- TRANSVERSE IMAGE DATA -->
.
.... raw image data bytes are written here...
.
.
<!-- SAGITTAL IMAGE DATA -->
.
.
.
.
<!-- CORONAL IMAGE DATA -->
.
.
.
.
<!-- TIME SERIES DATA -->
.
.
.
.

<!-- STOP FMRIDATAANALYSISDEMO DATA -->
```

Figure 6-2: A sample response from the fMRI Data Analysis server

| Key Suffix | Value Description |
|---|---|
| ***Graph Specifications*** | |
| -title | a string containing the title of the plot |
| **-horz-range** | the range of the horizontal axis (low value and high value) |
| **-vert-range** | the range of the vertical axis (low value and high value) |
| -horz-axis-label | a string containing the label for the horizontal axis |
| -vert-axis-label | a string containing the label for the vertical axis |
| -horz-tick-values | the list of values along the horizontal axis where tick marks should be drawn |
| -vert-tick-values | same as -horz-tick-values, but for the vertical axis |
| -vert-tick-labels | same as -horz-tick-labels, but for the vertical axis |
| ***Data Series (in a Line Plot)*** | |
| **-horz-data-n** | the list of values along the horizontal axis in the nth data series |
| **-vert-data-n** | the list of values along the vertical axis in the nth data series |
| -color-n | the color for the nth data series |
| ***Image Map*** | |
| **-image-dim** | the dimensions (width and height) of the image (required for number of pixels to be read) |
| **-image** | an array of raw data (short integers) for the image |
| **-horz-image-range** | the coordinates along the horizontal axis for the left and right edge of the map |
| **-vert-image-range** | the coordinates along the vertical axis for the bottom and top edge of the map |

Table 6.3: The suffixes for keys specifying the different properties of graph/data in the server response. Required keys are marked bold.

4d, the line plot time-series-plot is also required, and the time coordinate should be returned with t-coord. For the data type matrices, the server is required to return the image maps of design-matrix and stim-cov-matrix. Also required are the regressors corresponding to each column n of the design matrix, regressor-plot-n. For plot, the server should return the plots plot-n for however many plots the client should display.

### Alternative options considered

The author also considered returning the response which is just text based, containing the temporary location URLs for cross section images, rather than sending the actual raw data. This requires fair amount of work on server side to construct the image. But the abovementioned option was not pursued for two reasons: the server would have to do more work and that could lead to latency in the response, (why do that when the same functionality could be achieved at client side using advanced Java imaging technologies), and,the client would have to make multiple calls to server, once for constructing it and another to fetch the image.

# Chapter 7

# Conclusions and Future work

## 7.1 Client-Server Application

This web-based client/server software application was implemented using the design and technologies described in the previous chapters. The application is operational with most of the features fully functional, including navigation through 3 and 4-dimensional brain volume data, display of statistical maps, and client side auto-correlation function. One of the observations made about the performance of the application is its increased interactivity after switching to a non-Matlab based server solution. Some interesting points to note about the application are:

- The layered architecture based design of the Java Client will allow for easy future development and integration of additional features in the GUI
- The minimal dependency in the design of server and client for each other allows complete replacement of any of these, if needed. Also, each of them can easily be expanded to provide greater functionality

We expect the Java client to run successfully on multiple platforms as an applet as well as a stand-alone application. Author will test the application in the month of June 2004 by measuring latency and other matrices of performance on MIT Athena work stations, on LINUX, Windows and Macintosh systems, etc. Also, author will

work with Biomedical Informatics Research Network (BIRN) portal [41] collaborators to seek their feedback on improving performance and easy deployment. The accompanying tutorial will also be updated before distributing the application over the web.

## 7.2 Future Work

Several additional features can be implemented in a straight forward manner, such as keeping a parameter history, navigating through the brain by specifying the coordinates, and adding a mosaic view. Other features, requiring more effort, include allowing users to upload their own raw data, and the capability for users to play with their uploaded data in the software application, after some automated scripts at the server finish the pre-processing and pre-computation steps.

The accompanying tutorial could be expanded so that it does not presume much knowledge and could be useful to all the researchers interested in learning statistical analysis of fMRI. Suitable material which is not covered in the tutorial should be referenced via appropriate links to external web sites. The effectiveness of this tutorial should be re-examined and improved by applying the "How People Learn" (HPL) framework [30]-[32], a strategy for designing effective learning environments as explained in chapter 3. This application could serve as a model prototype for creating many more of such educational tools in the future.

An eventual goal of this project is to disseminate the fMRI Data Analysis Software Application and the accompanying tutorial to researchers across the globe via Biomedical Informatics Research Network (BIRN) [41]. The application source code has already been moved to the BIRN, and the application will be made accessible to the wider biomedical community through the BIRN portal after its testing is done. As described earlier, BIRN brings together the computational resources and the data online in order to serve biomedical researcher scientists through the internet.

# Bibliography


[1] A. Harner, *An Introduction to Functional Magnetic Resonance Imaging (FMRI) for Studying Visual Perception*, PS 822 Final Project, Jan. 13, 1997.

[2] I. Lai, R. Gollub, R. Hoge, D. Greve, M. Vangel, R. Poldrack, J. E. Greenberg, *Teaching statistical analysis of fMRI Data*

[3] R. L. Savoy, *History and Future Directions of Human Brain Mapping and Functional Neuroimaging* Acta Psychologica 107 (2001): 9-42

[4] P.Jezzard , P. M. Matthews, and S. M. Smith, Eds., *Functional MRI: An Introduction to Methods*, Oxford University Press, Oxford, UK, (2001).

[5] R. Ed. Buxton, *Introduction to Functional Magnetic Resonance Imaging: Principles and Techniques*, Cambridge University Press,Cambridge, UK (2001).

[6] S. Ogawa, T. M. Lee, A. R. Kay, and D. W. Tank, *Brain magnetic resonance imaging with contrast dependent on blood oxygenation*, Proceedings of the National Academy of Science of the United States of America 87, 9868-9872, (1990).

[7] G. D. Pearlson,T. E. Schlaepfer, *Brain Imaging in Mood Disorders*, [Online document], Available HTTP: http://www.acnp.org/g4/GN401000100/CH098.html

[8] I. Lai, *A Web-Based Tutorial for Statistical Analysis of fMRI Data*

[9] J. P. Hornak, *The Basics of MRI*, [Online document], Available HTTP: http://www.cis.rit.edu/htbooks/mri/

[10] *FIDAP Basics Home Page: Spatial Smoothing*, [Online document], Available HTTP: http://lbc.nimh.nih.gov/fidap/spatialsmooth.html

[11] D. C. Noll, *A Primer on MRI and Functional MRI*, [Online document], Available HTTP: http://www.bme.umich.edu/ dnoll/primer2.pdf

[12] J. Culham, *fMRI for Dummies*, [Online document], Available HTTP: http://defiant.ssc.uwo.ca/Jody_web/fmri4dummies.htm

[13] S. Clare, *Functional MRI: Methods and Applications*, [Online document], Available HTTP: http://www.fmrib.ox.ac.uk/ stuart/thesis/

[14] M. Brett, *Cambridge Imagers: Tutorials*, [Online document], Available HTTP: http://www.mrc-cbu.cam.ac.uk/Imaging/tutorials.html

[15] *Introduction to FMRI*, [Online document], Available HTTP: http://www.fmrib.ox.ac.uk/fmri_intro/

[16] S. Smith, *FEAT: FMRI Expert Analysis Tool User Guide*, [Online document], Available HTTP: http://www.fmrib.ox.ac.uk/

[17] Wellcome Department of Cognitive Neurology, *Statistical Paramteric Mapping*, [Online document], Available HTTP:http://www.fil.ion.ucl.ac.uk/spm/

[18] B. V., *BrainVoyager: a product from Brain Innovation*, [Online document], Available HTTP: http://www.brainvoyager.com/

[19] R. W. Cox, *AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages*, Computers and Biomedical Research 29, pp. 162 - 173, (1996).

[20] P. A. Bandettini, A. Jesmanowicz, E. C. Wong, and J. S. Hyde, *Processing strategies for time-course data sets in functional MRI of the human brain*, Magnetic Resonance in Medicine 30, 161 - 173, (1993).

[21] K. J. Friston, C. D. Frith, P. F. Liddle, and R. S. J. Frackowiak, *Comparing functional (PET) images: The assessment of significant change*, Journal of Cerebral Blood Flow and Metabolism 11, 690 - 699, (1991).

[22] A. L. Nicole, F. E. William, R. G. Christopher, J. Welling, *Statistical Issues in fMRI for Brain Imaging*, July 13, 1999.

[23] R. Hoge, *fMRI Data Acquisition Lab* [Online document], Available HTTP: http://www.nmr.mgh.harvard.edu/ rhoge/HST583/doc/HST583-Lab1.html

[24] R. Gollub, *HST-583 Functional Magnetic Resonance Imaging: Data Acquisition and Analysis Fall 2002 Home Page*, [Online document], Available HTTP: http://web.mit.edu/hst.583/www/

[25] S. Smith, *FMRIB MEDx FMRI Analysis Course*, [Online document], Available HTTP: http://www.fmrib.ox.ac.uk/internal/medx/course/

[26] *Functional Magnetic Resonance Imaging: An Introductory Course*, [Online document], Available HTTP: http://www.firc.mcw.edu/course/

[27] M. Vangel, D. Greve, *The MGH/MIT/HMS Martinos Center for Medical Imaging Announcing a short course in fMRI statistics*, Available HTTP: http://www.nmr.mgh.harvard.edu/NewFiles/short_course.html

[28] *fMRI Course*, Human Brain Mapping 2001, 10 June 2001, [Online document], Available HTTP:http://www.hbm2001.ucl.org.uk/register/course10.html

[29] T. Nichols and S. Smith, *ISMRM Morning Categorical: fMRI Data Analysis*, [Online document], Available HTTP: http://www.ismrm.org/02/morningcat2('02).htm

[30] J. D. Bransford, A. L. Brown, and R. R. Cocking, Eds., *How People Learn: Brain, Mind, Experience, and School*, Washington, DC: National Academy Press, 1999. Available: http://www.nap.edu/openbook/0309065577/html/index.html

[31] T. R. Harris, J. D. Bransford, and S. P. Brophy, *Roles for learning sciences and learning technologies in biomedical engineering education: A review of recent advances*, Annu. Rev. Biomed. Eng., vol. 4, pp. 29-48, 2002. Available: http://129.59.92.138/docs/Harris_001.pdf

[32] D. Laurillard, *Learning through collaborative computer simulations*, British Journal of Educational Technology, 23(3) pp. 164-171 (1992).

[33] Sun Microsystems, *What is Java Advanced Imaging (JAI)?*, [Online document], Available HTTP: http://java.sun.com/products/java-media/jai/whatis.html

[34] *The Mathworks: MATLAB Runtime Server* [Online document], Available HTTP: http://www.mathworks.com/products/runtime/

[35] C. A. Cocosco and A. C. Evans, *Java Internet Viewer: a WWW Tool for Remote 3D Medical Image Data Visualization and Comparison*, http://www.bic.mni.mcgill.ca/users/crisco/jiv/

[36] Wikipedia, *Apache HTTP Server*, [Online document], Available HTTP: http://en.wikipedia.org/wiki/Apache_HTTP_Server

[37] *The Mathworks: MATLAB Web Server*, [Online document], Available HTTP: http://www.mathworks.com/products/webserver/

[38] *Introduction to Minc*, [Online document], Available HTTP: http://www.bic.mni.mcgill.ca/software/minc/minc.html

[39] A. Rhyno, *Java in Context*, [Online document], Available HTTP: http://www.mgmt.dal.ca/slis/etig/itcolumn/itjava.htm

[40] The Apache Software Foundation, [Online document], Available HTTP: http://www.apache.org/

[41] University of California, San Diego, *Biomedical Informatics Research Network*,[Online document], Available HTTP: http://www.nbirn.net/