Biomedical Computing and Visualization

Chris R. Johnson and David M. Weinstein

Scientific Computing and Imaging Institute
School of Computing
University of Utah
50 S. Central Campus Drive, Salt Lake City, UT 84112, US

crj@sci.utah.edu, dmw@sci.utah.edu

Abstract

Computers have changed the way we live, work, and even recreate. Now, they are transforming how we think about and treat human disease. In particular, advanced techniques in biomedical computing, imaging, and visualization are changing the face of biology and medicine in both research and clinical practice. The goals of biomedical computing, imaging and visualization are multifaceted. While some images and visualizations facilitate diagnosis, others help physicians plan surgery. Biomedical simulations can help to acquire a better understanding of human physiology. Still other biomedical computing and visualization techniques are used for medical training. Within biomedical research, new computational technologies allow us to "see" into and understand our bodies with unprecedented depth and detail. As a result of these advances, biomedical computing and visualization will help produce exciting new biomedical scientific discoveries and clinical treatments. In this paper, we give an overview of the computational science pipeline for an application in neuroscience and present associated research results in medical imaging, modeling, simulation, and visualization.[1]

Keywords: Biomedical computing, imaging, problem solving environment, visualization.

1 Introduction

The next decade will see an explosion in the use and the scope of biomedical computing and visualization. Advanced, multimodal imaging and visualization techniques, along with new computational methods, will change the way many biomedical researchers and clinicians do their work. The combination of biomedical imaging and visualization with biomedical simulations will produce information about anatomical structure that is linked to functional data, in the form of electric and magnetic fields, mechanical motion, and biochemistry, and genetics. Such an integrated approach will provide comprehensive views of the human body in progressively greater depth and detail. However, such integration will require significant advances in biomedical computing software infrastructures and corresponding advances in multi-scale biomedical computing, imaging, and visualization algorithms (Johnson 2004).

Over the past two decades, the techniques of computer simulation and visualization have had a substantial impact on the field of biomedicine, as they have on other areas of science and engineering. Computer simulation allows biomedical researchers to subject increasingly sophisticated quantitative and qualitative conceptual models of biological behavior to rigorous quantitative simulation and analysis.

2 Neuroscience Application: Neural Source Imaging

Many times each second, the brain sends electrical impulses racing through the body's web of nerve cells to the motor neurons, where they initiate the electrochemical reactions that cause muscles to contract. Several decades ago, scientists recognized that these excitation currents produce an electrical field that can be detected as small voltages on the scalp. In 1924, German psychiatrist Hans Berger recorded the first electroencelphogram (EEG). The EEG electrode measures the small electrical activity from the brain and contains continuous trains of activity. The practice through which one can infer the inter-cranial sources that give rise to these measurements is termed the *neural source imaging* or *inverse EEG* problem. Neural source imaging is a fundamental problem in neuroscience. Learning precisely which regions of the brain are active at a particular time is a central problem in fields ranging from cognitive science to neuropathology to surgical planning.

While the modern technologies of electrode design and electronic recording apparatus differ significantly from their predecessors, the EEG waveforms are essentially the same as those recorded by Berger. Even with the substantial advances in EEG technology, most of the machines in clinical use today provide relatively coarse descriptions of the overall electrical activity of the heart or brain. This limitation in resolution is primarily due to the fact that standard EEG measurements represent the cumulative electrical activity of the brain as a very small number of simple point sources of bioelectricity. Physicians uses these glimpses to help spot disorders by comparing the patient's EEGs with an atlas of waveforms

that correspond to particular disease states. Compressing all this information into a small number of features is very efficient, but can lack the sensitivity and spatial resolution required for diagnosing many illnesses.

In some difficult cases, physicians turn to other techniques that are more invasive, costly, and painful and in rare cases, to exploratory surgery. In some cases of epilepsy, for example, physicians must establish whether the source of this abnormal electrical activity is well localized, and hence operable. At present, this diagnosis may require the application of electrodes directly to the surface of the brain.

Using computer modeling, imaging, simulation and visualization, we are developing diagnostic tools that may reduce the need for these cases of preoperative surgery, by simulating and visualizing the electric fields emanating from the brain. Using large-scale, three-dimensional computer models of the head and brain, we can produce more detailed visual representations of the electrical activity within the brain than the currently used brain snapshots from standard EEGs. A primary goal is to develop these techniques based on painless, risk-free voltage measurements from the head surface and gain information that is now primarily available through highly invasive diagnostic procedures.

3 Computational Science Pipeline

In order to solve the neural source imaging problem from above, we must perform several steps that involve elements of what we call the computational science pipeline: experimental data acquisition (patient image acquisition), mathematical modeling (physical equations that describe bioelectric fields), geometric modeling (segmentation, mesh generation), material modeling (electrical conductivity and diffusion tensor), numerical approximation (large-scale parallel finite element analysis, linear solvers, nonlinear optimization), visualization (of the geometric model, material model, and solutions), and validation (of the models and solutions).

Figure 1 schematically illustrates the "Inverse EEG Pipeline" we have constructed for efficient and interactive neural source imaging. In addition to creating efficient algorithms for each task, it is also important to create useful, integrated software, such as the SCIRun (Parker 1997, Weinstein 2005) software system described in the next section. We now briefly describe the stages within the inverse EEG pipeline.

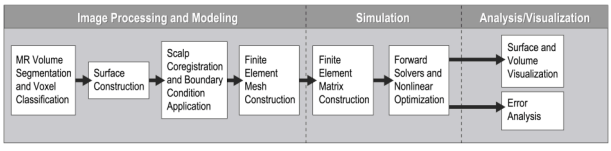

Fig 1: The Inverse EEG Computational Pipeline.

3.1 MRI Volume Segmentation and Voxel Classification

Our pipeline takes raw MRI data from a scan of a patient's cranium as anatomic input. This stage of the pipeline is accomplished using modules from the Insight Toolkit (ITK) (Yoo 2002) within SCIRun, using, for example a level set algorithm (Lefohn 2003), as shown in Figure 2. The output from this process is a tagged volume of voxels, each labeled with a tag to identify the primary material contained within that voxel. For our application, we are specifically interested in: air, skin, bone, cerebro-spinal-fluid, grey matter and white matter.

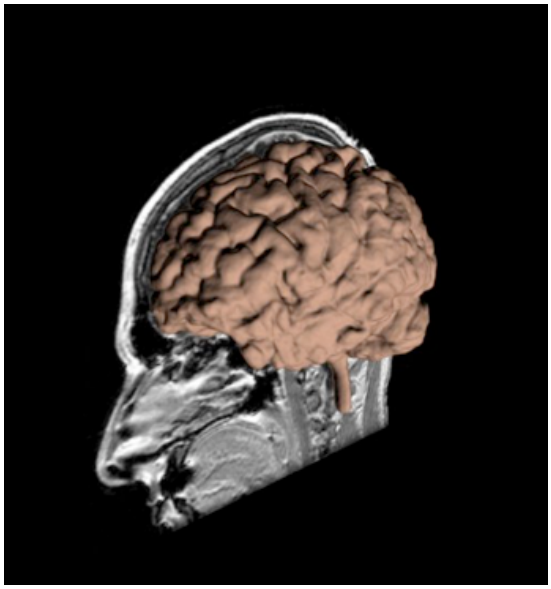

Figure 2: Result of segmentation of the brain using a level set algorithm.

3.2 Surface Construction

From the classified voxels we extract the set of boundary surfaces via a flood-fill/seed-growing style algorithm. Each boundary then corresponds to a discrete material region. Unfortunately, because the segmentation process can leave some noise in the data, we often have on the order of 10,000 surfaces after this process is completed, with many of these surfaces only bounding a single voxel. To reduce the number of surfaces, we pre-process the segmented volume with an assimilation algorithm that annexes regions containing less than some threshold number of voxels into the largest neighboring region. This process has the positive effect of reducing the complexity of the model (where fewer surfaces implies lower complexity), but it can also be destructive if the threshold is set too high. As a result, this was one of the parameters we studied in this study. The surfaces that result from this extraction have a characteristic "staircase" jaginess, since they are composed of voxel faces. To smooth out this data into more physiologically correct (and numerically stable) surfaces, we apply a scanline surface algorithm (Weinstein 2000) to these non-manifold surfaces. The result is shown in the first frame of Figure 3.

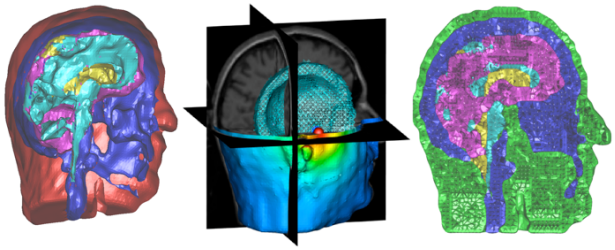

Figure 3: The result of surface construction and mesh generation from segmented MRI data.

3.3 Scalp Co-registration and Boundary Condition Application

In addition to the raw MRI data, we use two other clinically obtained raw datasets in our pipeline. These datasets are the basis for the functional data that will be mapped onto the scalp surface and serve as Dirichlet boundary conditions. The first of these datasets are the potentials recorded through electrodes attached to the patient's scalp. The second dataset is a list of point locations in space, obtained with a pointing device and a magnetic tracker. These points are used to spatially locate the electrode positions on the MR dataset, and consist of electrode positions and a cloud of points digitized off the patient's scalp. The first step in the co-registration process is to match the point cloud to the scalp surface extracted in the surface construction stage. This is done with a semi-automatic algorithm that applies affine transformations to the point cloud to minimize the summed squared distances from the points in the point cloud to the surface. The second step of this process is to apply the boundary conditions to the scalp surface. We simply determine the closest scalp point to each of the electrodes and assign it the corresponding potential with a Dirichlet boundary condition.

3.4 Finite Element Mesh Construction

To construct our finite element mesh, we use a spatial subdivision algorithm that subdivides space into uniform cubic voxels and places a single mesh node in each voxel. Voxels that correspond to air are not included in this process. The placement of each node is chosen based on two criteria: if a surface passes through the voxel, the node is constrained to lie on the surface; nodes must maintain a minimal distance from each other (a Poisson disk constraint, applied between neighboring voxels is used to guarantee this property). The edge lengths of these cubic voxels will directly determine the number of nodes generated. Since fewer nodes will result in less accurate meshes geometrically, we varied this parameter to evaluate the effect of geometric inaccuracies on the cortical solution. After generating all of the nodes, we use the CAMAL mesh generator (Sandia 2004) to construct a tetrahedral mesh. Each element in the mesh is tagged with a material/conductivity.

3.5 Finite element matrix construction

The finite element matrix is constructed by discretizing a generalized Poisson equation, $\nabla \cdot \sigma \nabla \Phi = -I_V$, where Φ is the voltage, σ is the electrical conductivity tensor, and I_V is the electric current per unit volume. The details of the finite element theory and implementation are described in detail in (Johnson 1997). The result of this algorithm is a sparse, symmetric, positive-definite stiffness matrix that encodes all of the geometry and electrical conductivity information of the problem.

3.6 Nonlinear Optimization

In order to solve the source localization problem, one needs to use nonlinear optimization. This involves solving the discretized Poisson equation above multiple times in order to find the global minimum of a misfit function defined as the difference between the measured voltages on the surface of the scalp and the computed solutions assuming a model neural source. We used both a multi-restart simplex search and a simulated annealing algorithm to find the global minimum of the misfit function. Both algorithms recovered the same neural sources, modeled as dipoles. The simplex search algorithm was restarted eight times for each source in order to improve the likelihood that we had localized the global minimum. We validated our recovered minima through an exhaustive search of the domain. Details of the algorithm and implementation can be found in (Weinstein 2000).

3.7 Visualization

Researchers at the SCI Institute and collaborators have created several novel visualization techniques to visualize scalar, vector, and tensor fields (Kniss 2005, Livnat 2005, Scheuermann 2005, Whitaker 2005, Zhang 2005). As an example of a new multi-field visualization technique, we applied a combination of stream surface visualization with simple tensor field visualization to look at the effects of including anisotropy within a realistic head model for the EEG source localization simulation. Figures 4 and 5 illustrate the visualization of the effects of white matter anisotropy using these techniques. We can observe a correlation between the primary direction of the conductivity structure of the white matter fiber bundles and the direction of the return currents. The visualization of return currents in bioelectric field problems can reveal important details about the distribution of sources, interactions at conductivity boundaries, and the effect of geometric distortion on bioelectric fields. By integrating the stream surfaces with a visualization of the diffusion tensors representing the white matter, we can better understand the structural, spatial relationships (Wolters 2005).

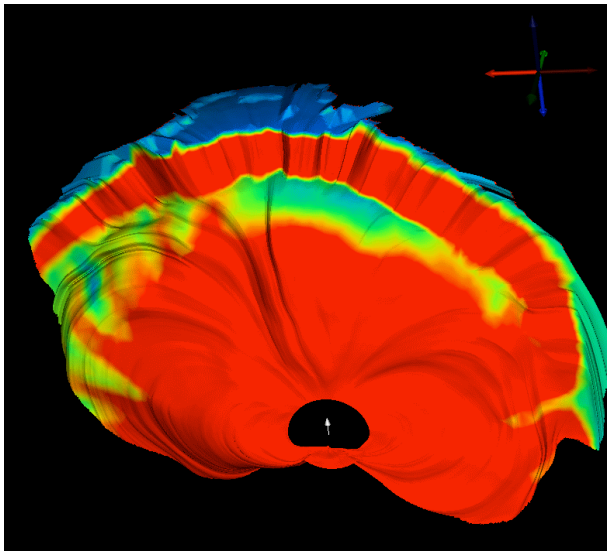

Figure 4: Visualization of return current surfaces from an EEG simulation using an isotropic conductivity model.

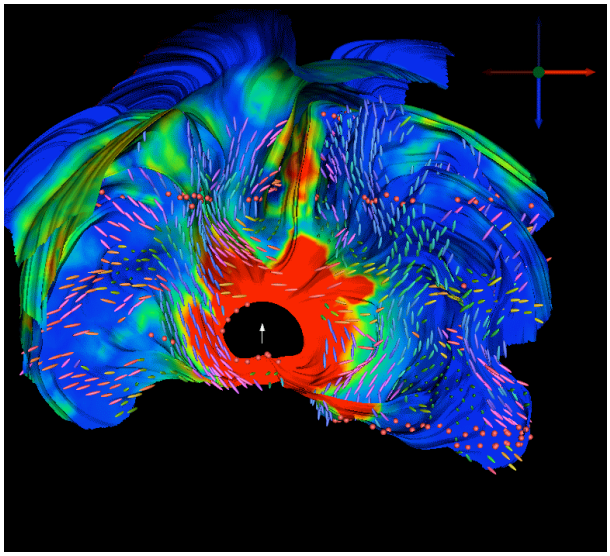

Figure 5: Visualization of return current surfaces from an EEG simulation using an anisotropic conductivity model.

In Figure 6 we show the orientation of the anisotropic white matter tracks from a diffusion tensor MR scan using a novel application of super quadric glyphs (Kindlmann 2004).

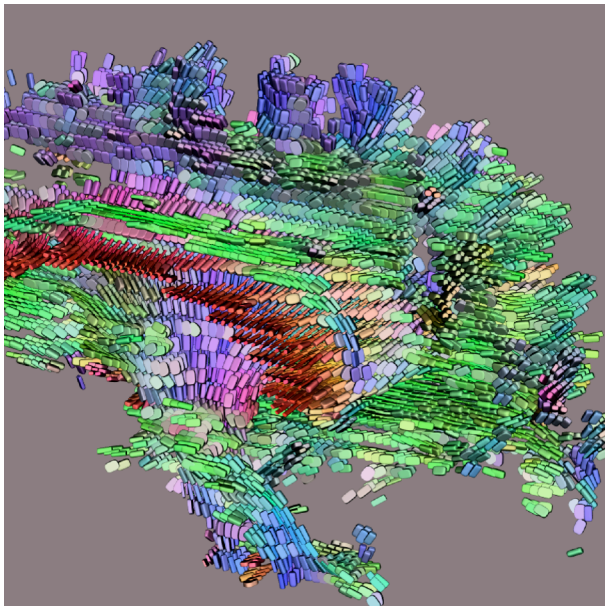

Figure 6: Visualization of half a brain DT-MRI volume using super quadric glyphs. Red indicates left/right, green indicates anterior/posterior, and blue indicates superior/inferior.

4 SCIRun: Integrated Software System

The desire to understand biological systems drives researchers to create ever-more sophisticated computational models. While such sophistication is essential to good research, the resulting complexity of the scientific computing process has itself become a major hindrance to further progress. Sources of this complexity include the number of equations and variables required to encapsulate realistic function, the size of the resulting systems and data sets, and the diverse range of computational resources (algorithms, databases, software, and hardware) required to support significant advances. Biomedical computing researchers gather multi-channel and multi-modal data from real-time collection instruments, access large distributed databases, and rely on sophisticated simulation and visualization systems for exploring biomedical systems.

Managing such large-scale computations requires powerful hardware and efficient and transparent software that frees the user to engage the complexity of the scientific problem rather than of the tools themselves. Unfortunately, such biomedical computing software does not currently exist. The range of computational tools available is growing so rapidly that navigating this large set of possible options has become its own challenge. The need to integrate software is especially acute when scientists seek to create models that span spatial or temporal scales or cross physical systems (e.g. combining electrical with mechanical and biochemical parameters). Integration is also necessary across the various components of the modeling and simulation process. No single researcher has the skills required to master all the computational and biological knowledge needed to successfully create geometric and mathematical

models, map them to numerical algorithms, implement them efficiently in modern computers, visualize the results, and understand them as they pertain to the specific biological system under investigation. To successfully model such complex systems requires a multidisciplinary team of specialists, each with complementary expertise and an appreciation of the interdisciplinary aspects of the system, and each supported by a software infrastructure that can leverage specific expertise from multiple domains and integrate the results into a complete software system.

Problem-solving environments (PSEs)[2] provide a natural platform to support integration and leverage multidisciplinary expertise to create complete systems for biomedical computing (Bramley 2000). Such systems solve the challenges of interfacing disparate elements and provide a level of functional abstraction that greatly assists researchers dealing with complex software systems.

PSEs also provide infrastructure for vertical integration of computational knowledge. Specific elements that may be incorporated into a comprehensive PSE include knowledge of the relevant discipline(s); the best computational techniques, algorithms and data structures; the associated programming techniques; the relevant user interface and human-computer interface design principles; the applicable visualization and imaging techniques; and methods for mapping the computations to various computer architectures (Bramley 2000). A PSE can consolidate knowledge from a range of experts in these disparate areas into a system that offers the end user a powerful set of computational tools.

Within the Scientific Computing and Imaging (SCI) Institute at the University of Utah, we have a long history of research in software architecture and creating problem-solving environments for scientific computing, such as SCIRun, BioPSE, and Uintah (SCIRun 2005).

The SCIRun PSE allows the interactive creation, investigation, and steering of large-scale scientific computations. SCIRun has been under development since the mid 1990s, but it has been enhanced significantly over the past five years due to the efforts of two large research centers that have used SCIRun as their core software system. These centers are 1) The Center for the Simulation of Accidental Fires and Explosions (C-SAFE), a Department of Energy ASHLI ASAP Level 1 Center; and 2) the NIH NCRR Center for Integrative Biomedical Computing. Largely because of these efforts, SCIRun has become a comprehensive software environment for scientific computing applications. SCIRun provides a component model, based on a generalized dataflow paradigm, which allows different computational components and visualization components to be connected in a tightly integrated fashion. A dataflow model implies the following: 1) data is sent to a software component, 2) the component manipulates the data in some manner, and 3) the new data is sent downstream to the next component for further manipulation.

SCIRun can be viewed as a *computational workbench*, in which a scientist designs and modifies a simulation interactively via a component-based visual programming model. SCIRun also facilitates interactive debugging and steering of large-scale, typically parallel, scientific simulations by, for example, enabling a scientist to modify geometric models and interactively change numerical parameters and boundary conditions. As opposed to the typical off-line simulation mode - in which the scientist manually sets input parameters, then computes results, and finally visualizes the results via a separate visualization package, and then starts again at the beginning - SCIRun closes the loop, combining each of these phases of the scientific investigation of the chosen problem.

While SCIRun provides the framework and software support needed to provide the extensive functionality discussed above, the actual science is done by individual software components. The modules are stand-alone pieces of software designed by various individuals or groups and contributed to the system. It is through combining the functionality of a number of modules that interesting problems are solved.

SCIRun/BioPSE Example of EEG Simulation and Visualization

An example electroencephalography (EEG) neural source localization application is show in Figures 7 and 8. Figure 7 contains the dataflow network that implements an inverse EEG application. At the top of the network, the input data files are loaded; these include the finite element mesh that defines the geometry and conductivity properties of the model and a precomputed lead-field matrix that encodes the relationship between electric sources in the domain and the resulting potentials that would be measured at the electrodes. Further down in the network, we have a set of modules that optimize the dipole location in order to minimize the misfit between the measured potentials from the electrodes and the simulated potentials due to the dipole. Finally, we have visualization and rendering modules, which provide interactive feedback to the user.

[2] We note that there are a number alternative phrases for what we mean by a problem solving environment currently being used in the scientific software literature, including software frameworks, toolkits, scientific software environments, software workbenches, plus a number of application specific names.

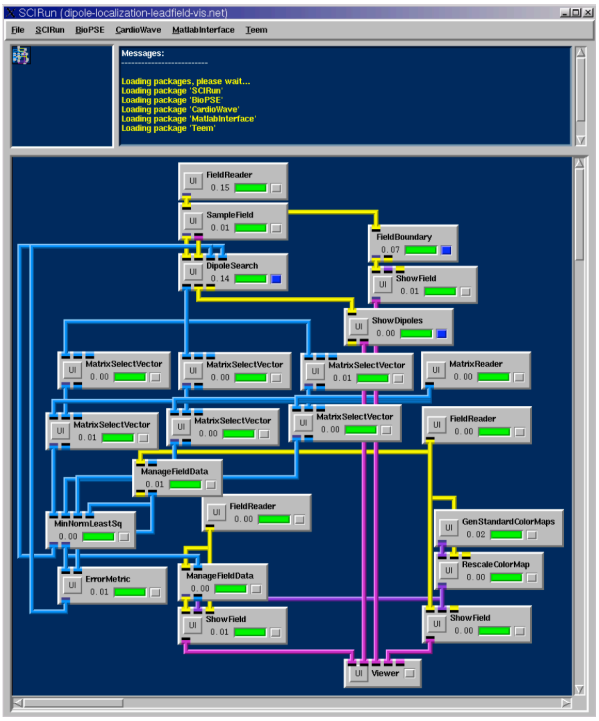

Figure 7: SCIRun/BioPSE modules combined for EEG modeling (unstructured mesh generation), simulation (finite element simulation, parallel linear system solves, and inverse source localization), and visualization (mesh visualization, isosurface extraction, and vector field visualization.

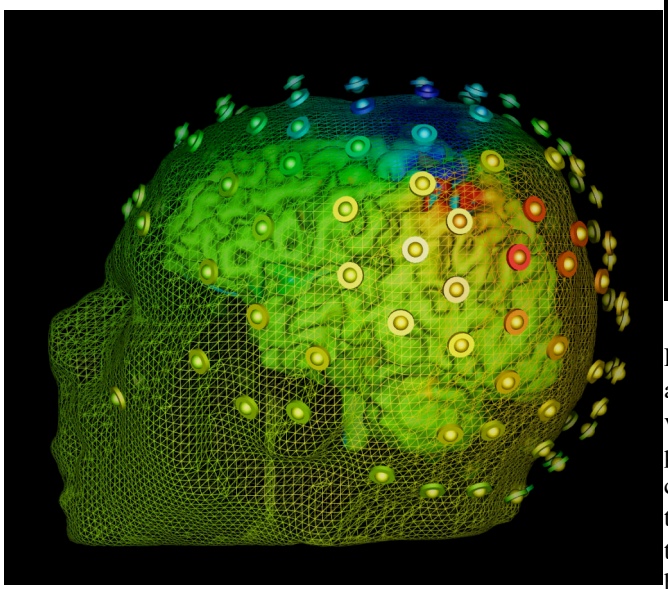

Figure 8: Visualization of simulation results of an EEG simulation localizing a neural source.

PowerApps

One of the major hurdles to SCIRun becoming a practical tool for the scientists and engineers has been SCIRun's dataflow interface. While visual programming is natural for computer scientists and some engineers, who are accustomed to writing software and building algorithmic pipelines, it is overly cumbersome for application scientists[3]. Even when a dataflow network implements a specific application (such as the bioelectric field simulation network provided with BioPSE and detailed in the BioPSE Tutorial), the user interface (UI) components of the network are presented to the user in separate UI windows, without any semantic context for their settings. For example, SCIRun provides file browser user interfaces for reading in data. However, on the dataflow network all of the file browsers have the same generic presentation. Historically, there has not been a way to present the filename entries in their semantic context, for example to indicate that one entry should identify the electrodes input file and another should identify the finite element mesh file.

While this interface shortcoming has long been identified, it has only recently been addressed. We recently introduced *PowerApps*. A PowerApp is a customized interface built atop a dataflow application network. The dataflow network controls the execution and synchronization of the modules that comprise the application, but the generic user interface windows are replaced with entries that are placed in the context of a single application-specific interface window. Figure 9 shows the BioFEM PowerApp implementation of the neural source localization application.

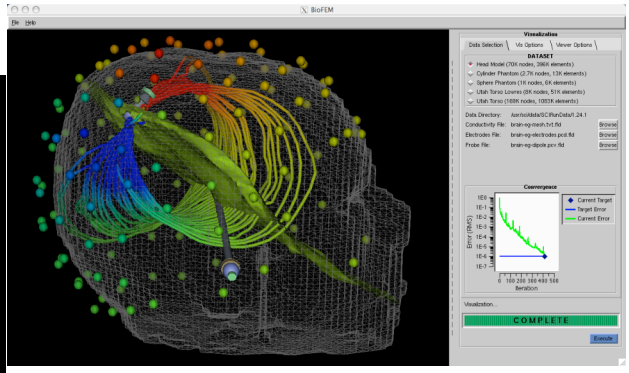

Figure 9: The BioFEM custom interface. Though the application is functionality equivalent to the dataflow version shown in Figure 7, this PowerApp version provides an easier-to-use custom interface. Everything is contained within a single window; the user is lead through the steps of loading and visualizing the data with the tabs on the right; and generic control settings have been replaced with contextually appropriate labels; and application-specific tooltips (not shown) appear when the user places the cursor over any user interface element.

[3] We note this statement is often true of software written by computer science researchers being used by application scientists and engineers.

5 Next Generation Software Architecture: SCIRun2

At the SCI Institute, we are beginning development of a

next-generation software architecture, called SCIRun2 (Zhang 2004). This system shares much of its software code-base with SCIRun, and it is our intent to evolve SCIRun into SCIRun2 over the next year.

SCIRun2 seeks to remove barriers to software component reuse by employing a flexible component architecture that enables a number of different styles of components (called component models) to be used together simultaneously. Thus far, we have been very successful in writing component wrappers to allow software packages (such as ITK, Teem, MATLAB, the CAMAL mesh generator, and so forth) to be used as modules in SCIRun. All of these undertakings have been successes: they have broadened the applicability of SCIRun, improved its performance, and have made it a more useful tool for our collaborators and for the scientific community at large. In practice, though, some of these efforts were very straightforward while others required significant custom development to overcome the technical hurdles.

SCIRun2, born of our experience developing SCIRun, provides a new internal architecture that is specifically designed to integrate component-based and object-based software such as the libraries described above, making this task of integration both simpler and more powerful.

The primary innovative design feature of SCIRun2 is a meta-component model that facilitates integration of a number of classes of tools from various, previously incompatible systems. In the same way that components plug into a traditional component-based PSE (such as the original SCIRun), SCIRun2 will allow entire component models to be incorporated dynamically. SCIRun2 facilitates the coupling of multiple component models, each of which can bring together a variety of components. In addition, the SCIRun2 architecture directly enables features that we wish to add to SCIRun, such as support for MPI-based components, a separation of the user interface from the computational engine, improved scripting support, and features for collaboration.

6 Summary

Advanced biomedical computing techniques coupled with advances in multi-modal imaging and visualization will change the way many biomedical researchers and clinicians do their work. The combination of biomedical imaging, and visualization with biomedical simulations will produce information about anatomical structure that is linked to functional data, in the form of electric and magnetic fields, mechanical motion, and biochemistry, and genetics. Such an integrated approach will provide comprehensive views of the human body in progressively greater depth and detail. However, such integration will require significant advances in biomedical computing software infrastructures and corresponding advances in multi-scale biomedical computing, imaging, and visualization algorithms.

7 Acknowledgments

This work was supported, in part, by a grant from the NIH NCRR 5P41-1RR12553-07 and from grants from DARPA, DOE, and NSF. SCIRun, BioPSE, and PowerApps software are all available as Open Source from the SCI Institute website (www.sci.utah.edu). SCIRun, BioPSE and the PowerApps are currently supported on three different platforms: Linux, Macintosh OSX, and SGI IRIX. A Windows port will be available in early 2006.

8 References

Johnson, C.R., MacLeod, R.S., Parker, S.G., and Weinstein, D.M. (2004): Biomedical Computing and Visualization Software Environments. *Communications of the ACM*, **47** (11): 64-71.

Parker, S.G., Weinstein, D.M., and Johnson, C.R (1997): The SCIRun Computational Steering Software System. In *Modern Software Tools in Scientific Computing*, 1-40. Arge, E., Bruaset, A.M. and Langtangen, H.P. (eds). Birkhauser Press.

Weinstein, D.M., Parker, D.M., Simpson, J., Zimmerman, K., and Jones, G. (2005): Visualization in the SCIRun Problem-Solving Environment. In *The Visualization Handbook*, 615-632. Hansen, C.D. and Johnson, C.R. (eds). Elsevier.

Yoo, T.S., Ackerman, M.J., Lorensen, W.E, Schroeder, W., Chalana, V. Aylward, S., Metaxes, D., and Whitaker, R. (2002): Engineering and Algorithm Design for an Image Processing API: A Technical Report on ITK - The Insight Toolkit. In *Proc. of Medicine Meets Virtual Reality*, (586-592). Westwood, J. (ed). IOS Press Amsterdam.

Lefohn, A.E., Cates, J.E., and Whitaker, R.T (2003): Interactive, GPU-Based Level Sets for 3D Segmentation. In *Medical Image Computing and Computer Assisted Intervention* (MICCAI), 564-572.

Weinstein, D.M. (2000): Scanline Surfacing: Building Separating Surfaces from Planar Contours. In *Proceeding of IEEE Visualization 2000*, 283-289.

Sandia National Laboratories (2004): CAMAL - The CUBIT Adaptive Meshing Algorithm Library - http://cubit.sandia.gov/camal.html - Release 2.0.2.

Johnson, C.R. (1997): Computational and Numerical Methods for Bioelectric Field Problems. In *Critical Reviews in BioMedical Engineering*, **25**(1):1-81.

Weinstein, D.M., Zhukov, L. and Johnson, C.R. (2000): Lead-Field Bases for EEG Source Imaging. *Annals of Biomedical Engineering,* 28(9):1059-1065.

Wolters, C.H., Anwander, A., Tricoche, X, Lew, S., and Johnson, C.R. (2005): Influence of Local and Remote White Matter Conductivity Anisotropy for a Thalamic

Source on EEG/MEG Field and Return Current Computation. In *International Journal of Bioelectromagnetism* (In Press).

Kindlmann, G. (2004): Superquadric Tensor Glyphs. In *Proceeding of The Joint Eurographics - IEEE TCVG Symposium on Visualization 2004*, (147-154).

Kniss, J.M., Kindlmann, G., Hansen, C.H. (2005): Multidimentional Transfer Functions for Volume Rendering. In *The Visualization Handbook*, (189-210). Hansen, C.D. and Johnson, C.R. (eds). Elsevier.

Livnat, Y. (2005): Accelerated Isosurface Extraction Approaches. In *The Visualization Handbook*, (39-55). Hansen, C.D. and Johnson, C.R. (eds). Elsevier.

Scheuermann, G. and Tricoche, X (2005): Topological Methods for Flow Visualization. In *The Visualization Handbook*, (341-356). Hansen, C.D. and Johnson, C.R. (eds). Elsevier.

Whitaker, R.T. (2005): Isosurfaces and Level-Sets. In *The Visualization Handbook*, (97-123). Hansen, C.D. and Johnson, C.R. (eds). Elsevier

Zhang, S., Laidlaw, D.H., and Kindlmann, G. (2005): Diffusion Tensor MRI Visualization. In *The Visualization Handbook*, (327-340). Hansen, C.D. and Johnson, C.R. (eds). Elsevier

Bramley, R., Char B., Gannon, D. , Hewett, T., Johnson, C.R., and Rice, J. (2000): Enabling Technologies for Computational Science: Frameworks, Middleware, and Environments. In *Workshop on Scientific Knowledge, Information, and Computing*, (19-32). Houstis, E., Rice, J., Gallopoulos, E, and Bramley, R. (eds). Kluwer Academic.

SCIRun, BioPSE, and PowerApp Software. Scientific Computing and Imaging Institute. http://www.sci.utah.edu/.

Zhang, K., Damevski, K., Venkatachalapathy, V., Parker, S. (2004): SCIRun2: A CCA Framework for High Performance Computing, In *Proceedings of The 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments*.