Visualization in the SCIRun Problem Solving Environment

David M. Weinstein, Steven Parker, Jenny Simpson, Kurt Zimmerman, and Greg M. Jones

1 Introduction to SCIRun

1.1 Motivation and History

Located at the crossroads of scientific applications, computer science, and numerical methods is the emerging field of Computational Science. With strongholds in applications ranging from chemistry to physics, from genetics to astronomy, computational science is growing into prominence throughout the scientific world, taking a position next to "theoretical" and "experimental", as another branch of nearly every scientific discipline.

Each scientific discipline has its own terminology, its own specific problems of interest. But from a broader perspective, their similarities often outnumber their differences. Many problems of interest are based around a physical model of some system or domain; they often attempt to predict the result of well-defined, equation-driven processes that take place within that domain; and the solutions to these problems are often most easily understood when recast into an interactive visual representation. Further, it is often not sufficient to run a single simulation of a system, but rather the scientist typically wants to investigate and explore the problem space, setting up different initial conditions, system parameters, and so on and then comparing the results.

Because of these consistent commonalities, it seems plausible that a general-purpose framework could be designed to assist scientists and engineers from a broad range of disciplines in investigating their respective computational science problems. Such a framework could be thought of as a "computational science workbench"; a scientist would have a broad range of tools at hand for modeling, simulating, visualizing, and iteratively exploring a problem space. The framework would be easy to use, a visual programming environment where the scientist could dynamically hook together computational components, just as an experimentalist would hook together mechanical components in a lab. And, perhaps most important for scientists working on large-scale problems, the framework would have to be extremely efficient in how it manages and processes data.

At the Scientific Computing and Imaging Institute (SCI) at the University of Utah, we set out to produce such a computational architecture beginning in the early 1990's. Our framework, called SCIRun (pronounced "ski-run"), was initially

developed by a handful of graduate students, and targeted at the simulation of bioelectric field problems as its initial application [6, 14]. Through the mid-1990's, SCIRun grew into a more robust platform, as it was applied to more applications, including cognitive neuroscience and atmospheric simulation [4, 15, 23]. In 1997 and 1998, the SCI Institute was awarded Center grants from the DOE and NIH respectively, to continue the research, development, and support of the SCIRun system.

1.2 Overview: Dataflow Terminology

As an infrastructure, the SCIRun computational problem solving environment is a powerful collection of high-performance software libraries. These libraries provide many operating-system type services, such as memory and thread management, and inter-thread communication and synchronization; as well as development utilities, such as geometry, container, scene-graph, and persistent I/O classes.

While the SCIRun infrastructure is complex, and is likely to be somewhat opaque to non-computer scientists, SCIRun's exterior layers are, in contrast, easy to use, extend, and customize. The SCIRun user-level programming environment, described above as a "computational workbench", is a visual dataflow environment that facilitates rapid development. Figure 1 shows an example of the SCIRun Visual programming environment.

The boxes on the canvas are called *modules*, and the wires connecting them are called *datapipes*. Each module encapsulates a function or algorithm, and the datapipes carry input and output data between them. Taken as a whole, the group of modules and datapipes comprise a dataflow *network* or *net*. At run-time users can interactively instantiate, destroy, and reconnect new modules. In addition to datapipe I/O, each module also has the option of exposing additional input and output parameters through a graphical user interface. For example, as shown in Figures 2 and 3, the SolveMatrix module is a linear solver that exposes input parameters such as the solver method and the maximum error tolerance, and also reports output parameters such as convergence plots for iterative solvers.

For the SolveMatrix module, we implemented several solvers natively within SCIRun. But we have also left placeholders for users to link in other solvers. This coupling of native support and optional hooks for extensibility has been a design pattern for SCIRun. In the SolveMatrix example (see Figure 3, we implemented Conjugate Gradient, Biconjugate Gradient, and Gauss-Seidel solvers; anyone downloading SCIRun will have immediate access to those methods. Then, in order to provide support for additional solvers, we created a *bridge* to the PETSc library. If a user chooses to download and install PETSc, they can configure SCIRun to use it, and the full set of PETSc solvers can then be leveraged within

SCIRun. We have also applied this bridging mechanism to allow users to access the ImageMagick and MPEG libraries for saving images and movies, respectively. Additionally, this same bridging solution has been implemented to allow Matlab users the ability to run their Matlab scripts from within SCIRun. By leveraging other libraries and applications, we are able to stay focused on developing high-performance infrastructure, and easy-to-use interfaces, while still providing support for a wide range of application functionality. Shown in Figure 4 is an example of one such bridge where Genesis has been bridged to SCIRun for the visualization of a combined genesis/SCIRun simulation of the bioelectric field between two Aplysia motor neurons.

1.3 The Visualization Pipeline

A typical visualization algorithm, such as Streamline advection, works by computing sample positions, evaluating the value of the Field at those positions, and creating a geometric representation for those values and positions. This three step process is common to many visualization methods: isosurfacing, streamlining, volume rendering, tensor field rendering, surface potential mapping, cutting plane rendering, etc. Typically, a particular visualization algorithm will implement all three of these steps itself. Such an approach results in substantial coding inefficiencies. For example, the same geometric representations may be of interest to multiple visualization techniques (*e.g.,*rendering pseudo-colored surfaces is common to surface potential mapping, cutting plane rendering, and often isosurfacing). In the spirit of modular programming and reusable components, we have pipelined (or "networked") the majority of our visualization methods, with interchangeable modules available for each of the three stages.

2 SCIRun Visualization Tools

There exist a number of tools that are easily accessible within the SCIRun system. Following is a rundown of this toolset:

2.1 For the Visualization of Scalar Fields:

- Isosurface: visualize isosurfaces of a volume, or isocontours on a surface. Can use Marching Cubes or NOISE algorithm. Can specify a single isovalue, or a list of isovalues, or a range and quantity for evenly spaced isovalues.
- Volume rendering / MIP (via 3D textures)
- Cutting plane (via 3D textures)

- Color-mapped geometry (ShowField) - The ShowField module visualizes the geometry that makes up a Mesh inside a Field. Where possible, the field takes its color from the Data values that permeate the field.

2.2 For the Visualization of Vector Fields

- Streamlines - The StreamLines module visualizes vector fields by generating curves that interpolate the flow of vectors in a Field.
- Vector glyphs (ShowField) - The ShowField module visualizes the geometry that makes up a Mesh inside a Field. Where possible, the field takes its color from the Data values that permeate the field.
- ShowDipoles - The ShowDipoles model allows the user to edit vector positions/orientations via widgets.

2.3 For the Visualization of Tensor Fields

- Glyphs: ellipsoids, colored-boxes
- Tensorlines

2.4 Quantitative Visualization

- ShowLeads - The ShowLeads module graphs a Matrix that has rows of potentials.
- ErrorMetric - The ErrorMetric module computes and visualizes error between two vectors.
- ShowField - The ShowField module visualizes the geometry that makes up a Mesh inside a Field. Where possible, the field takes its color from the Data values that permeate the field.
- ShowColorMap - The ShowColorMap module creates a geometry overlay containing the input colormap and numerical values for its range.

3 Remote/Collaborative Visualization

In the last few years, scientists and researchers have given a great deal of attention to the area of remote visualization of scientific datasets within collaborative environments. Remote visualization refers to the process of running an application on

one machine, often a supercomputer, and viewing the output on another machine in a different geographical location. Collaborative visualization is the use of tools (chat windows, annotations, synchronous viewing controls, etc.) that enable multiple geographically separated collaborators to directly exchange and simultaneously view information related to specific visualizations.

The recent interest in these tools has developed because researchers often use interactive viewing as the primary method of exploring large datasets. Researchers often need to extend this interactivity in order to collaborate remotely with colleagues in other locations. Additionally, the ability to use remote high-end computation resources is also driving the need for remote visualization tools. Certainly, visualization on grid based systems is also a driving demand for remote visualization tools.

Remote / Collaborative visualization is by no means a new problem As such, many algorithms have been developed as solutions. General strategies available for achieving remote visualization fall roughly into four categories:

1. Traditional XWindows remote display
2. Image/Pixel streaming
3. Geometry/Texture rendering
4. Some hybrid of the above methods

While most remote visualization tools based on the aforementioned methods successfully allow multiple parties to view images from different locations, most also face problems with efficiency and user interactivity at some level.

In general, popular recent approaches that address these shortcomings focus on improving two different areas of remote visualization:

1. Increasing network bandwidth utilization
2. Adjusting the amount of rendering performed on a local server versus a remote client in order to optimally utilize resources, such as available bandwidth between the server and client.

In particular, researchers and developers often use the client-server paradigm as a logical means to partition rendering responsibilities, efficiently utilizing valuable resources on both local and remote machines.

3.1 Current Work on Remote/Collaborative Visualization in SCIRun

Our work in implementing remote / collaborative visualization functionality in SCIRun has been largely experimental thus far. To this end, we built upon a prototype remote visualization application that applies the client-server paradigm, along with several rendering methods, as an attempt to offer greater flexibility for remote viewing. In addition to using multiple rendering methods, we have experimented with different networking protocols for data transfer to compare efficiency and accuracy. While we have learned a great deal from our research with this prototype application, we have run into fundamental design problems that have prompted us to sleight our remote visualization extension for redesign as a component in the design of the next generation of SCIRun, which will see the SCIRun system move toward a component architecture.

Presently, standard XWindow remote viewing is used for remote display of SCIRun.

3.2 Future Work on Remote/Collaborative Visualization in SCIRun

The future of remote / collaboartive visualization in SCIRun is closely tied to the Common Component Architecture (CCA) that is planned for SCIRun2. Roughly, the long-term plan for SCIRun (SCIRun2) is to consider everything to be a component, including the computing modules and the user interface. Presumably, the remote client will be a component which uses a CCA protocol to communicate with other components.

As for our rendering method, our current plan is to utilize a hybrid of XWindows remote display and image streaming to transfer image data from the computing engine to the remote client.

In the process of designing the remote visualization component, we will adhere to a list of user driven requirements that must be met. These include:

- Minimum x frames per second
- UI that matches SCIRun local UI both visually and functionally
- Synchronized image manipulation for multiple remote viewers with locked controls
- Exact representation of models - OR - Level of Detail (LOD) control
- Usability with “thin” client machine
- Usability with limited network bandwidth

- Chat window
- Annotation layer
- Compliance with SCIRun2 architecture and communication protocols
- Possibly some ability to record and replay sessions

The end goal is to have a remote user interface which has the same appearance and functionality as the local SCIRun user interface, but with added collaborative tools and remote viewing capabilities.

4 SCIRun Applications

As mentioned above, one of the original applications of SCIRun was to Bioelectric Field Problems. With the award of our NCRR grant from the NIH to create the Center for Bioelectric Field Modeling, Simulation, and Visualization; we have continued to focus on bioelectricity, creating modules, networks, and documentation to allow users to investigate both forward and inverse bioelectric field problems. In order to keep the core of SCIRun general-purpose, we have created a separate Package to house the components that are specific to bioelectricity. Taken together, the BioPSE Package, and the SCIRun architecture comprise the BioPSE System [21], as shown in Figure 5. Similarly, the Uintah Package [1, 9, 13] is an extended set of functionality targeting combustion simulation. In fact, a number of grants have now leveraged the SCIRun core, adding specific components to address the needs of different various applications. These applications range from bioelectric fields, to combustion simulation, to magnetic fusion. Each of these applications is briefly described below.

4.1 Modeling, Simulation, and Visualization of Bioelectric Fields [16, 19, 20, 22]

Our hearts and brains are electric organs. Electric activation at the cellular level causes the heart to beat, and it is the basis underlying our cognitive processes. However, unlike neurotransmitters and metabolic processes, electric patterns can be instantly detected at sites remote from the position of activation. By placing an ECG electrode on a patient's chest, we can "watch" the series of electrical events that make up a heart beat; by placing EEG electrodes on a patient's head, we can "watch" the electric activity as the patient thinks and reacts.

Cardiologists and neurologists are primarily interested in two types of bioelectic field problems: the forward problem, and the inverse problem. In the forward

problem, the question at hand is: given a pattern of source activation, determine the electric activity that would result through the rest of the domain (see 6). Such studies are used when investigating internal implantable defibrillator designs. The inverse problem is typically more interesting, though unfortunately also less numerically stable: given a set of remote measurements, determine the position and pattern of source activation that gave rise to those remote measurements (see 7).

The equations governing the flow of electricity through a volume conductor are very well understood. The goal of the Bioelectric Problem Solving Environment (BioPSE) is to simulate those governing equations using discrete numeric approximations. By building a computational model of a patient's body, and mapping conductivity values over the entire domain, we can accurately compute how activity generated in one region would be remotely measured in another region. The tools for modeling, simulating, and visualizing these bioelectric field phenomena comprise BioPSE.

4.2 Visualization for the Study of Magnetic Fusion

Alternate energy sources are becoming increasingly important as the world's finite resources, such as fossil fuels, are depleted. One promising source of unlimited energy is controlled nuclear fusion. Specifically, magnetic fusion is a type of nuclear fusion in which scientists harness energy by using magnetic fields to confine fusion reactions taking place within hot plasma. Since magnetic fusion research is computationally intensive, many software tools are needed to support it. Visualization tools are particularly critical to helping fusion scientists analyze their data. As part for our work within the DOE SciDAC sponsored National Fusion Collaboratory (http://www.fusiongrid.org/), we have created the Fusion package in SCIRun in order to help meet this need [3, 18].

Specifically, the Fusion package in SCIRun is designed to satisfy the goal of providing fusion scientists with visualization software tools that allow exploration of their data on a Linux workstation. The Fusion package consists of a set of SCIRun modules which, in concert with other standard SCIRun modules, allow the reading, visualization, and analysis of Fusion data that is in MDSPlus format. The system provides fusion researchers with flexible visualization options and feedback they need in order to properly adjust input parameters for the next iteration of data processing.

Currently, data generated using the NIMROD simulation code package is being used as a test bed for developing the SCIRun Fusion package with the extension of the SCIRun package to other data sources planned in the near future reference. The NIMROD package is publicly available code (http://www.nimrodteam.org/) designed to study three-dimensional, nonlinear electromagnetic activity in labora-

tory fusion experiments while allowing a large degree of flexibility in the geometry and physics models used in simulations. Using the visualization capabilities offered in the the SCIRun Fusion package makes it possible to visualize how the magnetic field moves within the pressure field represented in a NIMROD dataset. In this paradigm, the NIMROD simulations are done on supercomputers and then SCIRun running on a linux desktop is used to analyze the resulting data in order to appropriately adjust parameters for the next simulation.

Following is a brief description of an instantiation of the visualization pipeline created for NIMROD data. Once the NIMROD data is in SCIRun, a hexahedron mesh is build using the EditFusionField module which takes into account toroidal geometry and then infuses it with the pressure values at the nodes using the ManageField module. Next the data is passed downstream to several of the visualization modules available in SCIRun.

The simplest and most general visualization module is the ShowField module which displays a visual representation of the pressure values. The user can also pass in a ColorMap to the second port, which will translate the pressure values into colors in the rendering.

The next visualization module in the pipeline is the Isosurface module. Given a scalar field the Isosurface module will extract triangular faces that approximate the ovule surface through the domain. Since the input is pressure, isobaric surfaces are generated using this module. As with the ShowField module, the pressure values are again mapped to color via an input ColorMap. Via a user interface, a user can choose from several different Isosurface extraction algorithms, and can set a number of extraction and display options.

Using this visualization pipeline researchers are provided with the feedback needed to adjust their input parameters for the next iteration of data processing. In the case of the NIMROD data, SCIRun makes it possible to visualize how the magnetic field moves within the pressure field of the dataset(see Figure 8 and Figure 9).

The long-term goals for the Fusion package are to develop a generalized visualization system that can support a wide variety of fusion data and to bring more of the computing portion of the fusion visualization pipeline into SCIRun itself.

4.3 The Simulation of Accidental Fires and Explosions [1, 5, 13, 17]

Funded by the DOE as part of the Accelerated Strategic Computing Initiative (ASCI) to form the Center for the Simulation of Accidental Fires and Explosions (C-SAFE). This work is primarily focused on the numerical simulation of accidental fires and explosions, especially within the context of handling and storage of highly flammable materials (see Figure 10. The objective of C-SAFE is to pro-

vide a system comprising a problem-solving environment in which fundamental chemistry and engineering physics are fully coupled with non-linear solvers, optimization, computational steering, visualization and experimental data verification. For this work a derivative of SCIRun, coined Uintah has been developed. The Uintah PSE has been built specifically to handle very large datasets, which are typical in the CSAFE work. In this case, attempting to render an entire dataset can easily overwhelm the graphics hardware. To help us explore these datasets, we have incorporated into Uintah/SCIRun Multiresolution and a Multipipe volume renderers.

4.3.1 Multiresolution Volume Rendering

Multiresolution techniques enable interactive exploration of large-scale data sets while providing user-adjustable resolution levels on a single graphics pipe. A user can get a feel for the entire dataset at a low resolution, while viewing certain regions of the data at higher resolutions. A texture map hierarchy is constructed in a way that minimizes the amount of texture memory with respect to the power-of-two restriction imposed by OpenGL implementations. In addition, our hierarchical level-of-detail representation guarantees consistent interpolation between different resolution levels. Special attention has been paid to the elimination of rendering artifacts that are introduced by non-corrected opacities at level transitions. By adapting the sample slice distance with regard to the desired level-of-detail, the number of texture lookups is reduced significantly, improving interaction.

4.3.2 Multi-Pipe Volume Rendering

Multi-pipe techniques allow for interactive exploration of large-scale data at full resolution. Textures and color transfer functions are distributed among several rendering threads that control the rendering for each utilized graphics pipe or graphics display. On each draw cycle, view and windowing information is stored in a shared datastructure. While rendering is performed, compositing threads are supplied with the composite order for each partial image. Upon completion of the rendering, the rendering threads store the resulting image in a local structure. When all renderers have completed, compositing threads copy the partial images to a final image buffer, using alpha blending techniques. Care is taken to prevent blending artifacts in the final image by properly overlapping the texture data sent to each renderer and by pre-multiplying the colors in the transfer function by their corresponding alpha values.

4.4 Radiology and Surgical Planning

Most imaging systems currently used in medical imaging generate scalar values arranged in a highly structure rectilinear grid. These fields can be visualized by a variety of methods including: isosurface extraction, direct volume rendering, and maximum intesity projections (MIP). The key difference between these techniques is that isosurfacing displays actual surfaces, while the direct volume rendering and MIP methods display some function of the values seen along a ray throughout the pixel. Ideally, the display parameters for each technique are interactively controlled by the user.

Interactivity is fast becoming a fundamental requirement of medical visualizations. While only a few years ago radiologists and surgeons viewed inherently three dimensional images as two dimensional films, the use of interactive, three dimensional visualization tools has blossomed in medical imaging in the last few years. This is certainly true for the field of surgical navigation. Interestingly, the radiological exams of today are generating very large datasets (10's-100's of Mb) making interactivity a challenging requirement. While the commodity graphics card is making great strides, it is only recently that a texture memory of 256 Mb has been introduced. The medical imaging community is continually increasing the resolution, power, and size of their imaging tools, consistently outpacing the graphics card industry. Meanwhile, the medical imaging research community is now producing datasets in the multi-gigbyte range with new generation small animal imagers.

Several visualization tools incorporated in SCIRun have been applied specifically to large medical datasets. Examples of these tools are volume bricking and ray-tracing.

4.4.1 Volume Bricking [12]

While volume rendering can be performed in hardware on most modern graphics processing units (gpu's) via three-dimensional texture mapping, the amount of memory available on a particular gpu (commonly referred to as "texture memory") limits the size of the models that can be volume renderered. With large scale data, it is not uncommon for a dataset to be several times larger than the available gpu memory. A common solution is to break the dataset up into smaller chunks, each of which is small enough to fit into the memory at hand. This process is known as bricking. The bricks are then loaded into texture memory one at a time. After each brick is loaded, the corresponding texture is then mapped to a series of polygons drawn perpendicular to the view. To avoid artifacts bricks are sorted from furthest to nearest, based on the location of the view point and the location of the brick.

One must also take care to make sure that polygons drawn in neighboring bricks are aligned to avoid artifacts at the brick boundaries. Using this Volume bricking approach, datasets that are many times larger than the available gpu memory, can be processed and rendered nearly interactively.

4.4.2 Interactive Ray-Tracing [2, 12]

The basic ray-volume traversal method used in our ray-tracer allows us to implement volume visualization methods that find exactly one value along a ray. Fundamentally, for each pixel of the image a ray is traced through a volume to compute the color for that pixel. The computational demand of ray-tracing is directly dependent upon the number of pixels (i.e. resolution of the viewing screen) being used and less dependent on dataset size, this method allows both interactive isosurface extraction and maximum-intensity projection on very large datasets.

The ray-volume transversal method has been implemented as a parallel ray tracing system that runs on both an SGI Reality Monster, which is a conventional shared-memory multiprocessor machine and a linux cluster with distributed memory. To gain efficiency several optimizations are used including a volume bricking scheme and a shallow data hierarchy. The graphics capabilities of the Reality Monster or cluster are used only for display of the final color image. This overall system is described in a previous paper [11]. Conventional wisdom holds that ray tracing is too slow to be competitive with hardware zbuffers. However, when rendering a sufficiently large dataset, ray tracing should be competitive because its low time complexity ultimately overcomes its large time constant [7]. This crossover will happen sooner on a multiple CPU computer because of ray tracing's high degree of intrinsic parallelism. The same arguments apply to the volume traversal problem.

Examples of Ray Tracing Large Datasets

- The Visible Female:

  The visible female dataset, available through the National Library of Medicine as part of its Visible Human Project, [10] was used to benchmark this ray tracing method (see Figure 11). Specifically, we used the the computed tomography (CT) data which was acquired in 1mm slices with varying in-slice resolution. This rectilinear data is composed of 1,734 slices of 512x512 images at 16 bits. The complete dataset is 910 MBytes. For the skin isosurface, we generated 18,068,534 polygons. For the bone isosurface, we generated 12,922,628 polygons. With this number of polygons, it would be challenging to achieve interactive rendering rates on conventional high-end graphics hardware. Our method can render a ray-traced isosurface of this data at multiple frames per second using a 512x512 image on multiple processors (for

exact performance measures see [12].

- Small Animal Imaging:

  A recent example of ray tracing is the work done with Dr. Richard Normann and his group at the University of Utah [8]. In this case a relatively small dataset of 131 Mb was rendered interactively across approximately 20 processors on an SGI Origin 3800. Again, no graphics hardware was required for the rendering, except to display the image. The imaging was done to examine an implantation of the Utah Electrode Array (see figure 11, insert) into the cochlear nerve of a feline. In this case the investigators used high resolution CT imaging of the cats head to verify the location of the electrode array in the cochlear nerve. The imaging was accomplished with a GE EVS-RS9 small animal computed tomography (CT) scanner. There are distinct CT values for air, soft tissue, bone, and the electrode array, enabling the use of a combination of ray-tracing and volume rendering to visualize the array in the context of the surrounding structures, specifically the bone surface. Visualization results were improved by smoothing the voxels outside the electrode array in order to better distinguish the boney structures.

  As shown in Figure 12, the 27 micron resolution of the CT scan allows definition of the cochlea, the modiolus (on the right), the implanted electrode array, and the lead wires (in purple) that connect the array to a head mounted connector. The resolution of the scan even allows definition of the shanks and tips of the implanted electrode array. Volume rendering also allows the bone to be rendered as translucent, as on the left half of this image, enabling the electrode to be clearly viewed. Thus, the combination of high-resolution scanning, image processing, and interactive visualization tools such as ray-tracing, allows non-invasive verification of the implantation site in an anatomical structure that is completely encased in the thick temporal bone.

  The small animal imaging systems, such as the CT scanner used in this work, are capable of producing extremely large dataset sizes, in the 4-6 Gb range. Additionally, the combination of multiple datasets from multiple imaging modalities, such as micro CT and micro PET (positron emission tomography) combinations, will compound the problem of large dataset size. As scientists begin wanting to interact with these datasets methods such as ray-tracing and distributed visualization will be at the forefront of this work. It will certainly be quite sometime before stand-alone, commodity graphics cards will be able to interactively handle this size of dataset.

5 Getting Started in SCIRun

SCIRun and a variety of other software packages from the SCI Institute are available at the SCI software website http://software.sci.utah.edu/ Additionally, documentation including download and installation instructions, sample datasets, and sample visualization networks are also available at the website. Following is a brief description of the SCIRun documentation.

5.1 Documentation

One of the greatest efforts in transforming our existing collection of research codes into a friendly environment for external users was the generation of the various forms of requisite documentation. We drafted documentation standards and investigated tools for integrating that documentation within and extracting documentation from our software system. The result is a hyperlinked "living document" that can be browsed from our website and is also included with our software distribution. The collection of documentation has been organized into a library of five manuals: an *Installation Guide*, a *Tutorial*, a *User Guide*, a *Developer Guide*, a *Reference Guide*, and *Frequently Asked Questions (FAQ)*.

5.1.1 The *Installation Guide*

The *Installation Guide* provides instructions for installing SCIRun from RPM's (Linux) and from Source Code (Linux and SGI), its third party libraries, the (optional) PETSc library, sample data sets, and SCIRun documentation. The PETSc library adds equation solvers to SCIRun's SolveMatrix module (see PETSc Installation (Optional)).

5.1.2 The *SCIRun Tutorial*

The *SCIRun Tutorial* is an interactive introduction to SCIRun for new users. Since SCIRun is a large system, this tutorial provides a broad overview of SCIRun concepts, and a core set of SCIRun user skills. There are seven chapters in this document. By the end of the tutorial, the user will have a grasp of dataflow programming, SCIRun architecture, and the specific modules and datatypes used along the way. The tutorial begins with Chapter 1, which demonstrates the construction of a simple, yet functional SCIRun network. This demonstration is extended in Chapters 2-7, with additional functionality and complexity.

5.1.3 The *User Guide*

The *User Guide* describes how to get started using SCIRun. It includes a discussion of the dataflow programming paradigm and problem solving environments. It explains basic concepts such as how to run and use SCIRun, and how to write dataflow programs using SCIRun. The *User Guide* also includes descriptions of all the SCIRun modules, including datatypes used, functions performed, and explanations of the user interface elements.

5.1.4 The *Developer Guide*

The *Developer Guide* contains descriptions of the various SCIRun programming utilities, including our resource management tools (memory, threads, persistent objects, exceptions). For each tool, we describe how the tool fits into SCIRun, the philosophy of why and when a developer would use that tool, and usage examples.

5.1.5 The *Reference Guide*

The *Reference Guide* contains the API specifications for all of the tools in SCIRun. This information is extracted directly from the source code using the doxygen documentation system. For each class in SCIRun, the documentation contains a complete description of the class, as well as cross-referenced hyperlinks to related classes.

5.1.6 Frequently Asked Questions (*FAQ*)

The last book in the documentation library is the *FAQ*. The *FAQ* has been subdivided into "Technical" and "User" sections. The "Technical" section contains answers to technical questions that arise when compiling and linking SCIRun and its required thirdparty software. The "User" section contains questions and answers about the behavior of SCIRun, modules, and the various messages the system issues.

5.2 Getting Help

Getting help in SCIRun is relatively easy. We have built a users group email list, if you have questions at any time, email the scirun-users@sci.utah.edu list and either a SCIRun software engineer or another user will answer your questions.

References

[1] J.D. de St. Germain, J. McCorquodale, S.G. Parker, and C.R. Johnson. Uintah: A massively parallel problem solving environment. In *Ninth IEEE International Symposium on High Performance andDistributed Computing*, pages 33–41. IEEE, Piscataway, NJ, Nov 2000.

[2] D.E. DeMarle, S.G. Parker, M. Hartner, C. Gribble, and C.D. Hansen. Distributed interactive ray tracing for large volume visualization. *IEEE Symposium on Parallel Visualization and Graphics*, page (accepted), October 2003.

[3] D.P. Schissel for the National Fusion Collaboratory Project. An advanced collaborative environment to enhance magnetic fusion research. In *present at the Workshop on Advanced Collaborative Environments*, 2002.

[4] C.R. Johnson, M. Berzins, L. Zhukov, and R. Coffey. Scirun: Applications to atmospheric diffusion using unstructured meshes. In M.J. Baines, editor, *Numerical Methods for Fluid Dynamics VI*, pages 111–122. Oxford University Press, 1998.

[5] C.R. Johnson, S. Parker, D. Weinstein, and S. Heffernan. Component-based problem solving environments for large-scale scientific computing. *Journal on Concurrency and Computation: Practice and Experience*, (14):1337–1349, 2002.

[6] C.R. Johnson and S.G. Parker. A computational steering model applied to problems in medicine. In *Supercomputing 94*, pages 540–549. IEEE Press, 1994.

[7] J.T. Kajiya. *An Overview and Comparison of Rendering Methods, A Consumer's and Developer's Guide to Image Synthesis*. 1988.

[8] Gordon Kindlmann, Richard A. Normann, Arun Badi, James Bigler, Charles Keller, Richard Coffey, Greg M. Jones, and Chris R. Johnson. Imaging of the utah electrode array, implanted in cochlear nerve. In *Digital Biology: The Emerging Paradigm*, Bethesda, Maryland, November 6-7 2003. NIH - Biomedical Information Science and Technology Initiative Consortium (BISTIC).

[9] J. McCorquodale, J.D. de St. Germain, S. Parker, and C.R. Johnson. The uintah parallelism infrastructure: A performance evaluation on the sgi origin 2000. In *High Performance Computing 2001*, Mar 2001.

[10] Nat'l Library of Medicine (U.S.) Board of Regents. Electronic imaging: Report of the board of regents. *NIH Publication*, 90-2197, 1990.

[11] S. Parker, W. Martin, P.-P. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive ray tracing. In *Symposium on Interactive 3D Graphics*, Apr 1999.

[12] S. Parker, M. Parker, Y. Livnat, P. Sloan C., and P. Shirley. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and ComputerGraphics*, July-September 1999.

[13] S.G. Parker. A component-based architecture for parallel multi-physics pde simulation. *Elseveier Science*, page (to appear), 2003.

[14] S.G. Parker and C.R. Johnson. Scirun: A scientific programming environment for computational steering. In *Supercomputing '95*. IEEE Press, 1995.

[15] S.G. Parker and C.R. Johnson. Scirun: Applying interactive computer graphics to scientific problems. *SIGGRAPH (applications/demo)*, 1996.

[16] O. Portniaguine, D. Weinstein, and C. Johnson. Focusing inversion of electroencephalography and magnetoencephalography data. In *3rd International Symposium On Noninvasive FunctionalSource Imaging*, volume 46, pages 115–117, Innsbruck, Austria, Sep 2001.

[17] R. Rawat, S. G. Parker, P. J. Smith, and C. R. Johnson. Parallelization and integration of fire simulations in the uintah pse. March 12-14 2001.

[18] A.R. Sanderson and C.R. Johnson. Display of vector fields using a reaction-diffusion model. SCI Institute Technical Report UUSCI-2003-002, June 2003.

[19] R. Van Uitert, D. Weinstein, and C.R. Johnson. Volume currents in forward and inverse magnetoencephalographic simulations using realistic head models. *Annals of Biomedical Engineering*, 31:21–31, 2003.

[20] R. Van Uitert, D. Weinstein, C.R. Johnson, and L. Zhukov. Finite element eeg and meg simulations for realistic head models: Quadratic vs. linear approximations. volume 46, pages 32–34, 2001.

[21] D. Weinstein, P. Krysl, and C. Johnson. The biopse inverse eeg modeling pipeline. In *ISGG 7th International Conference on Numerical Grid Generation in Computation Field Simulations*, pages 1091–1100. The International Society of Grid Generation, 2000.

[22] D.M. Weinstein, J.V. Tranquillo, C.S. Henriquez, and C.R. Johnson. Biopse case study: Modeling, simulation, and visualization of three dimensional mouse heart propagation. *International Journal of Bioelectromagnetism*, 5:(accepted), 2003.

[23] D.M. Weinstein, L. Zhukov, and C.R. Johnson. An inverse eeg problem solving environment and its applications to eeg source localization. In *NeuroImage (suppl.)*, page 921, 2000.

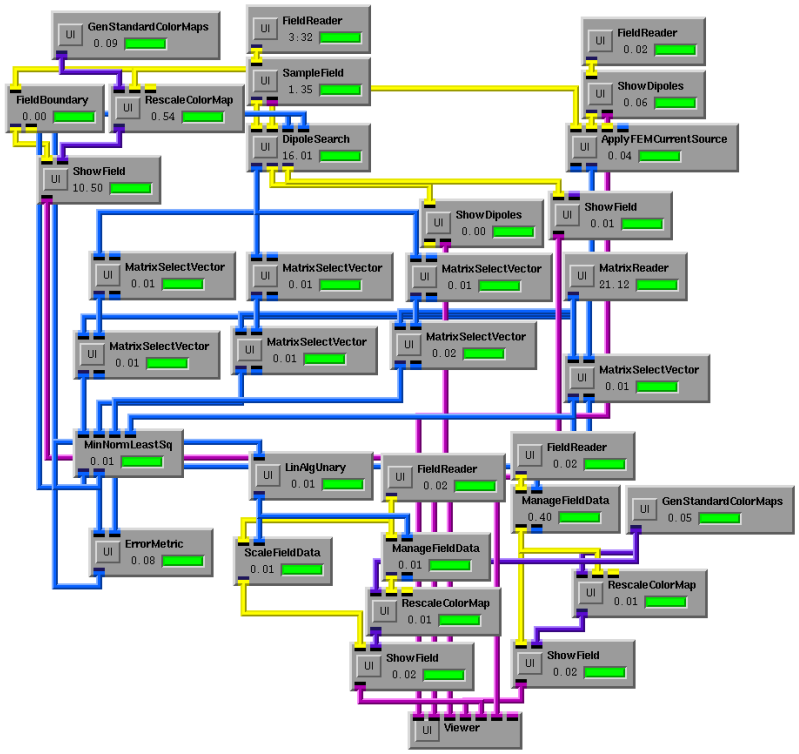

Figure 1: A SCIRun (pronounced ”ski-run”) dataflow network. Each module encapsulates a function or algorithm, while the datapipes carry input and output data between the modules.

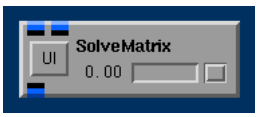

Figure 2: An example of a SCIRun module. The SolveMatrix module is a linear solver that exposes input parameters such as the solver method and the maximum error tolerance, and also reports output parameters such as convergence plots for iterative solvers. The ”UI” button produces a module specific interface allowing the user to adjust parameters specific to that module.

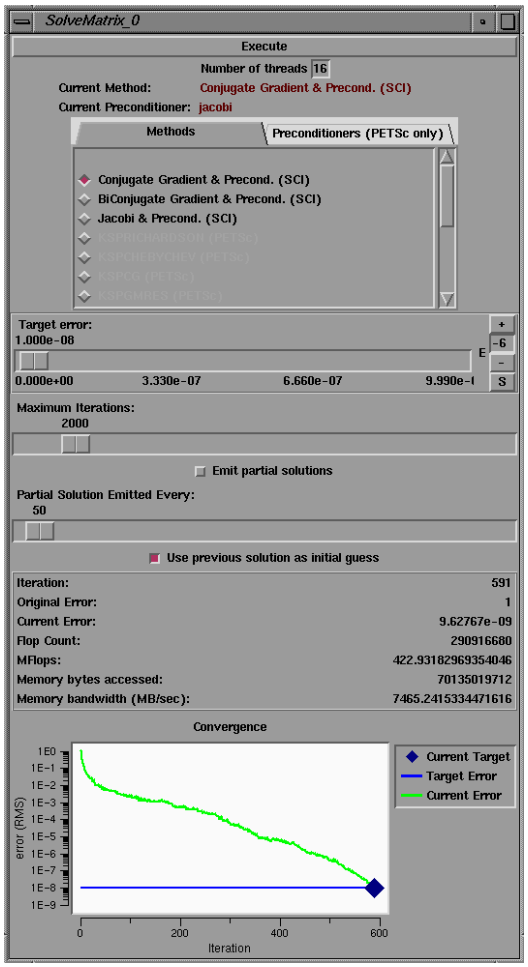

Figure 3: The solve matrix user interface (UI). This UI allows the user to interact with the model. In this case, the UI allows the user to chose various solvers such as the Conjugate Gradient, Biconjugate Gradient, and Gauss-Seidel solvers. Additionally, the convergence of the solver is also displayed in the UI. In order to provide support for additional solvers, there is also a *bridge* to the PETSc library.

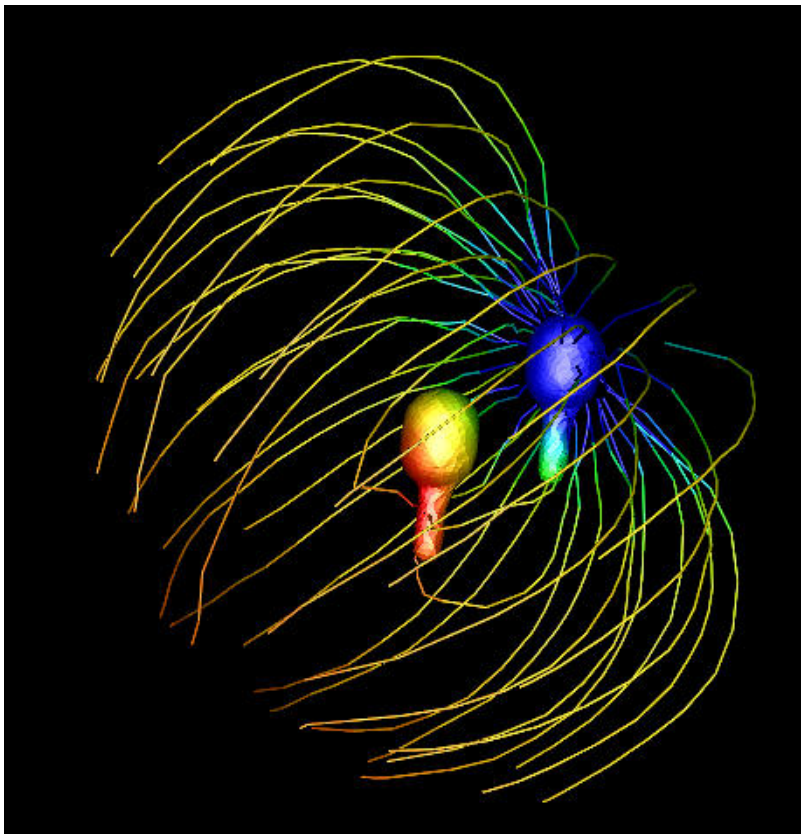

Figure 4: Simulation of two Aplysia motor neurons using the bridging capabilities between Genesis and SCIRun. First, Genesis solves the time-dependent Hodgkin-Huxley equations for each compartment in each cell. One result of the Genesis simulation is neuron membrane current density, which is passed to SCIRun through an SQL database. SCIRun uses the current density to solve the forward field problem in the volume surrounding the cell. In this picture, streamlines show current flow within the volume; voltage is encoded by color (blue negative, red positive). Image courtesy of Chris Butson, University of Utah, Dept. of Bioengineering.

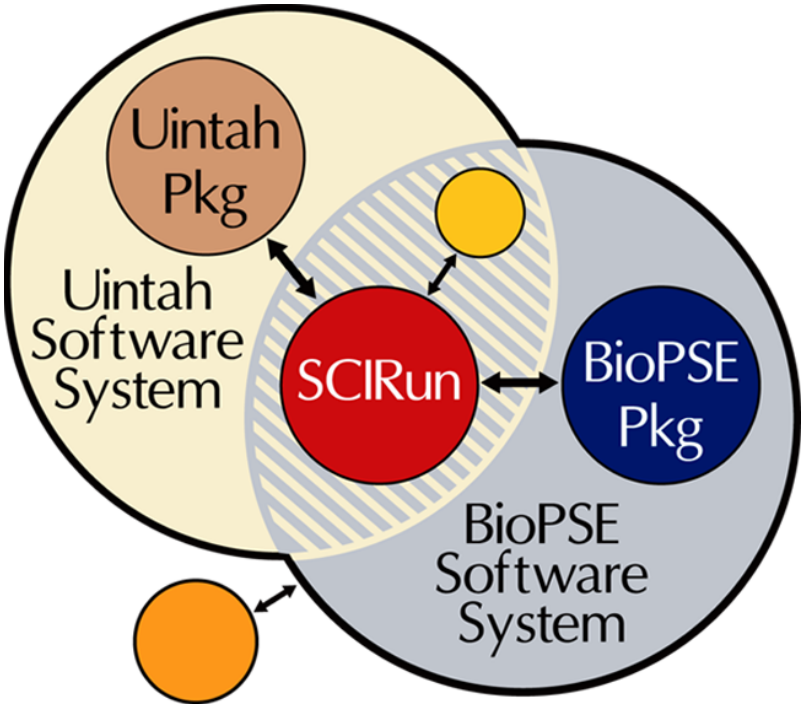

Figure 5: The relationship of the core infrastructure of SCIRun to the specialty packages, BioPSE and Uintah. In order to keep the core of SCIRun general-purpose, we have created a separate Package to house the components that are specific to bioelectricity. Taken together, the BioPSE Package, and the SCIRun architecture comprise the BioPSE System. Similiarly, the Uintah Package is an extended set of functionality targeting combustion simulation.

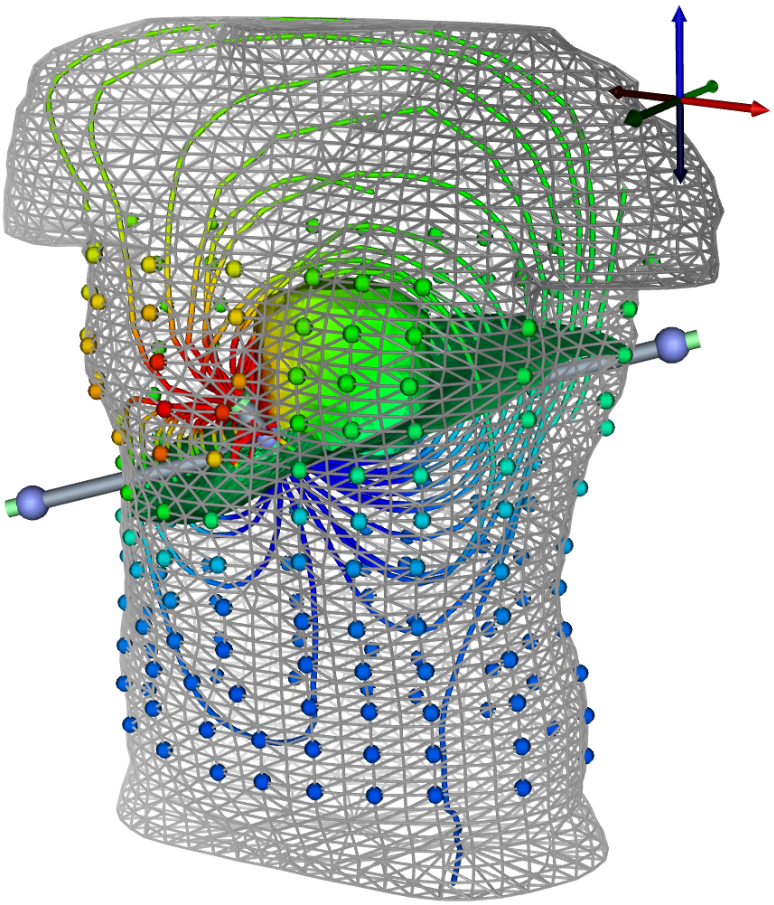

Figure 6: Visualization of a bioelectric field simulation. In this case an electric dipole is placed near the heart inside the Utah torso model. Visualized by color-mapped streamlines is the electric field set up by this dipole. Also visualized is the surface potential (indicated by color-mapped spheres on the surface of the torso) and a field potential isosurface (green surface inside the torso). This visualization was produced using the BioPSE forward-fem net with the movable dipole widget.

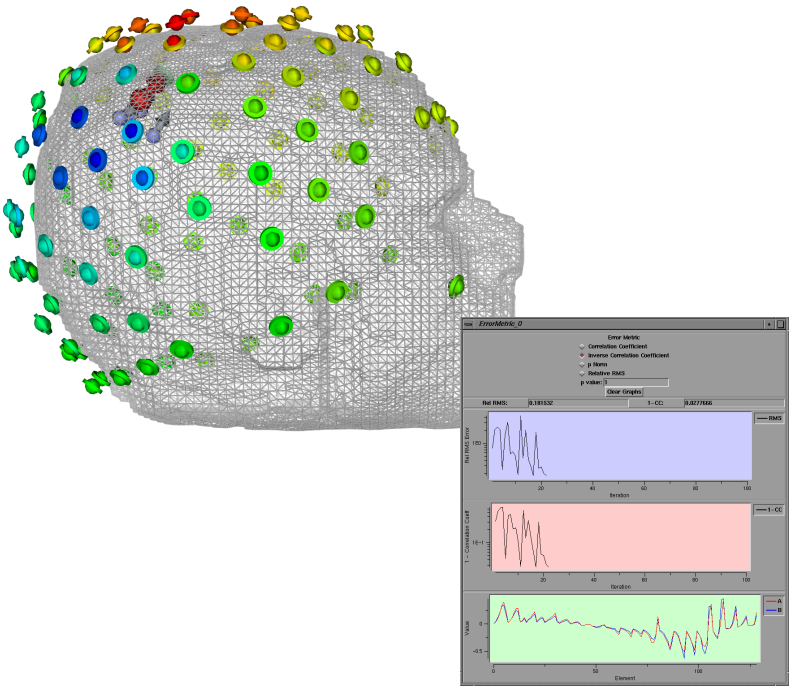

Figure 7: Visualization of an inverse EEG simplex search produced by using the dipole-localization net. The accuracy of the solution at each electrode position is also shown (disks show measured voltage, spheres show computed voltages), view window shows simplex dipoles as four arrows (connected with lines), and test dipole as a sphere. Also shown are the error metrics plotted for this particular solution.

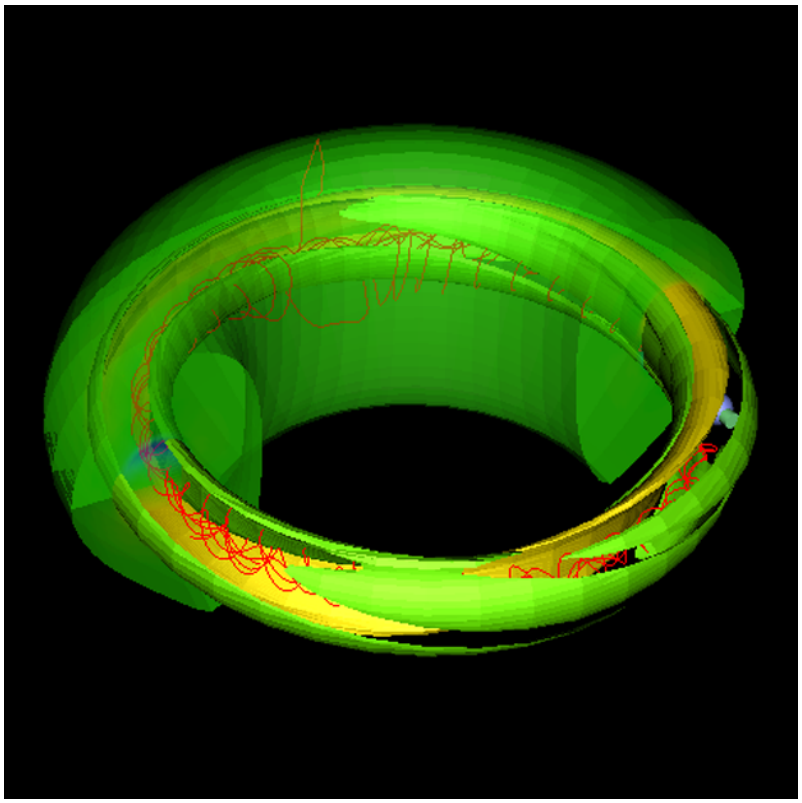

Figure 8: Simulation of an experiment inside a Tokomak Fusion Reactor visualized using the SCIRun Fusion package. This image is of NIMROD simulation data showing an isosurface of the n=0 part of the pressure field (yellow), which shows the 1/1 structure, and an isosurface of the n=2 part of the toroidal current field (green), which shows the developing 3/2 structure. Between the two isosufaces is a streamline using the sum of the n=0,1, and 2 modes of the magnetic field (red). The underlying model consists of a toroidal grid with 737,280 nodes (in 10 arbitrary phi slices) with 22 time slices. Image provided by Dr. Allen Sanderson and the National Fusion Collaboratory via support from the U.S. Department of Energy SciDAC initiative

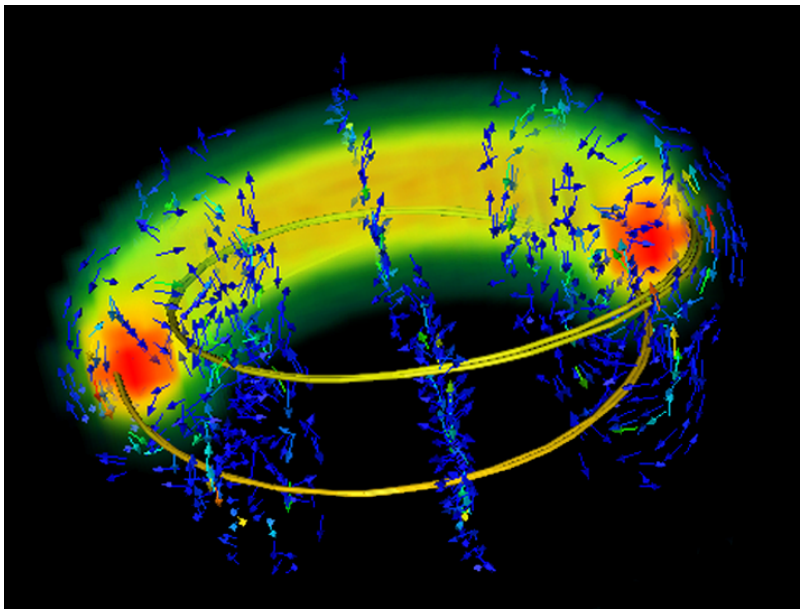

Figure 9: Simulation of an experiment inside a Tokomak Fusion Reactor visualized the SCIRun Fusion package. Frame one of two from a time sequence showing the stochastic nature of the realspace magnetic field lines. A comparison of the two frames shows that the magnetic field lines are starting to diverge earlier as time progresses. The field lines are overlaid in a volume rendering of the pressure field which provides visual cues to the location while also rendering the plasma velocity vector field. Image provided by Dr. Allen Sanderson and the National Fusion Collaboratory via support from the U.S. Department of Energy SciDAC initiative.

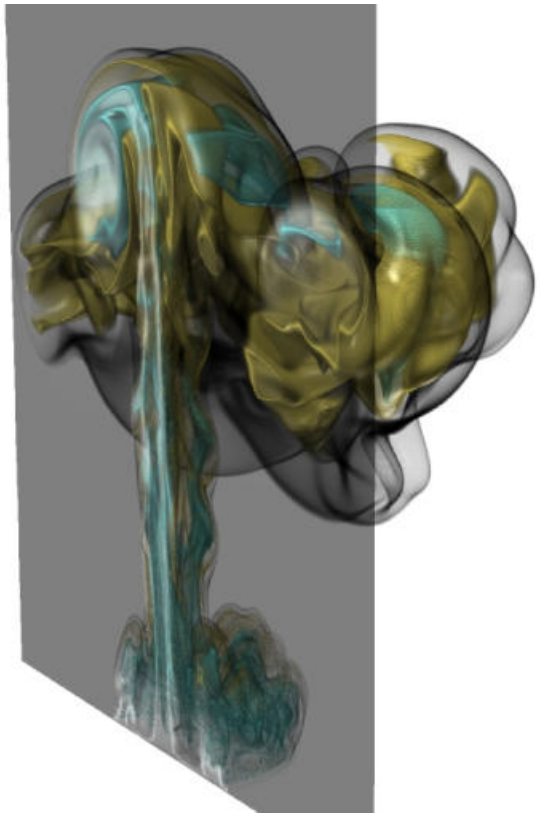

Figure 10: Simulation of a heptane pool fire. The simulation was done using the Uintah derivative of SCIRun and visualization using the ray-tracing package described later in Section 4.4.2. This image is courtesy of Center for the Simulation of Accidental Fire and Explosions (C-SAFE) working under funding from the Department of Energy as part of the Accelerated Strategic Computing Initiative (ASCI) Academic Strategic Alliance Program (ASAP).

Figure 11: Maximum intensity project (MIP) of the visible female dataset using ray tracing. The maximum intensity projection algorithm seeks the largest data value that intersects a particular ray. Ray tracing allows interactive visualization of the MIP of the 1 Gb visual female dataset.

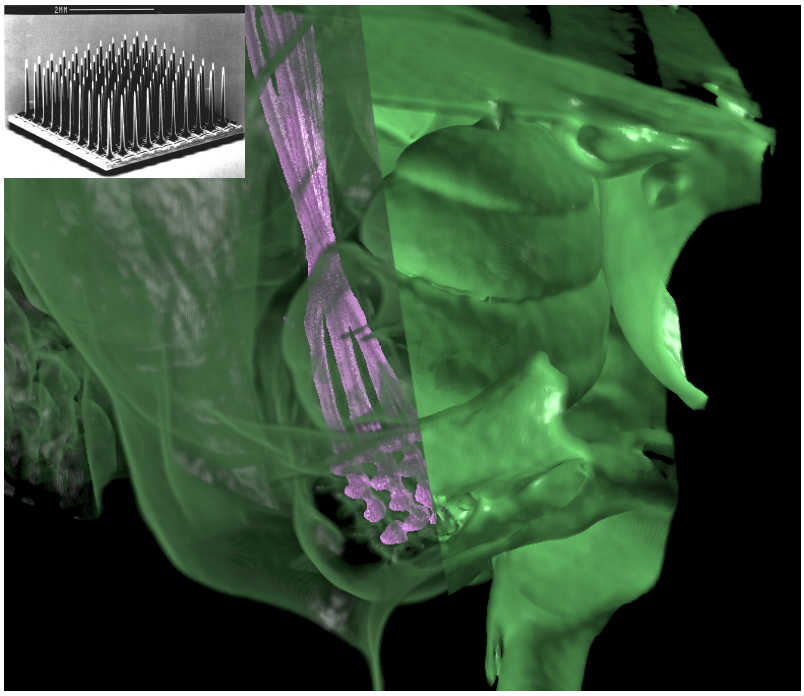

Figure 12: Visualization of the Utah Electrode array embedded the cochlear nerve of a cat. The insert is a picture of the Utah Electrode array. The 27 micron resolution of the CT scan allows definition of the cochlea, the modiolus (on the right), the implanted electrode array, and the lead wires (in purple) that connect the array to a head mounted connector. The resolution of the scan even allows definition of the shanks and tips of the implanted electrode array. Volume rendering also allows the bone to be rendered as translucent, as on the left half of this image, enabling the electrode to be clearly viewed. Thus, the combination of high-resolution scanning, image processing, and interactive visualization tools such as ray-tracing, allows non-invasive verification of the implantation site in an anatomical structure that is completely encased in the thick temporal bone. Data provided by Dr. Richard Normann and Dr. Charles Keller, University of Utah. This work was supported by the National Heart Lung and Blood Institute P20 HL68566; the National Center for Research Resources P41 RR012553; and NINDS/NIDCD NO1-DC-1-2108.