The LONI Pipeline Processing Environment

David E. Rex, Jeffrey Q. Ma, and Arthur W. Toga*

Laboratory of Neuro Imaging, Department of Neurology, David Geffen School of Medicine at UCLA, Los Angeles, CA 90095-1769, USA

Received 3 December 2002; revised 25 February 2003; accepted 18 March 2003

Abstract

The analysis of raw data in neuroimaging has become a computationally entrenched process with many intricate steps run on increasingly larger datasets. Many software packages exist that provide either complete analyses or specific steps in an analysis. These packages often possess diverse input and output requirements, utilize different file formats, run in particular environments, and have limited abilities with certain types of data. The combination of these packages to achieve more sensitive and accurate results has become a common tactic in brain mapping studies but requires much work to ensure valid interoperation between programs. The handling, organization, and storage of intermediate data can prove difficult as well. The LONI Pipeline Processing Environment is a simple, efficient, and distributed computing solution to these problems enabling software inclusion from different laboratories in different environments. It is used here to derive a T1-weighted MRI atlas of the human brain from 452 normal young adult subjects with fully automated processing. The LONI Pipeline Processing Environment’s parallel processing efficiency using an integrated client/server dataflow model was 80.9% when running the atlas generation pipeline from a PC client (Acer TravelMate 340T) on 48 dedicated server processors (Silicon Graphics Inc. Origin 3000). The environment was 97.5% efficient when the same analysis was run on eight dedicated processors.

© 2003 Elsevier Science (USA). All rights reserved.

Keywords: Dataflow model; Visual programming; Data processing environment; Automated analysis

Introduction

Neuroimaging analysis often requires multiple processing steps as raw data is transformed into interpretable results. Studies focused on gross anatomic structure (Wright et al., 1995; Rapoport et al., 1999; Rosas et al., 2002; Sowell et al., 2002), functional anatomy (Woods, 1996; Corbetta et al., 1998; Worsley et al., 1998; Dumoulin et al., 2000; Zeineh et al., 2001; Pouratian et al., 2002), various physiological measurements (Woods et al., 1994; Maier et al., 1995; McClure et al., 1998; Mega et al., 1999; Opitz et al., 1999), cyto- (Geyer et al., 1996; Mega et al., 1997; White et al., 1997; Amunts et al., 1999) and chemoarchitecture (Zilles et al., 1991; Koutcherov et al., 2000), gene localization (Reiss et al., 1995; Lohmann et al., 1999; Thompson et al., 2001; Cannon et al., 2002), and their interactions (Mazziotta et al., 2001) employ both common, as well as unique, processing procedures as part of the analysis. Many of the procedures are increasingly computationally intensive and often automated.

Several software packages encapsulating these processing steps have evolved (Table 1). Some are monolithic implementations, such as SPM T4 (Friston et al., 1990, 1991, 1995a, 1995b, Worsley and Friston 1995), BrainVoyager, SureFit and Caret (Carman et al., 1995; Drury et al., 1996; Van Essen and Drury, 1997; Van Essen et al., 1998, 2001), and MEDX, which provide the user with all-in-one approaches to analyzing data. Others are more modular environments, such as AIR (Woods et al., 1998a, 1998b), FSL (Jenkinson and Smith, 2001; Zhang et al., 2001; Smith, 2002), AFNI (Cox, 1996; Cox and Hyde, 1997), BrainSuite (Shattuck and Leahy, 2001, 2002; Shattuck et al., 2001), the Montreal Neurological Institute’s Brain Imaging Software Toolbox (Collins et al., 1994; MacDonald et al., 1994, 2000; Sled et al., 1998), and FreeSurfer (Dale et al., 1999, Fischl et al., 1999a, 1999b, 2001, 2002, Fischl and Dale,
2000), providing the user with access to individual programs from each package.

There are advantages and disadvantages to monolithic and modular approaches. In monolithic environments, there is a unified complete analysis solution in one package. The individual processing steps are already linked; there should be no file format incompatibilities; and the environment works as an integrated system. These monolithic environments, however, are less amenable to alteration. When new procedures are needed, programming expertise and familiarity with code and interfaces are required to allow new algorithms to be incorporated. Alternatively, researchers may go outside of the environment requiring them to ensure compatibility and force a modular approach on the analysis. They may also choose to wait for the authors of the monolithic implementation to agree that the methodological enhancements are worthwhile and include them in the next release. For these reasons, monolithic environments do not always possess the most current and best algorithms for the specific study at hand. Finally, these environments do not lend themselves to distributed computing paradigms important in population-based, high-resolution, multifactorial brain mapping studies.

Modular environments (Fig. 1) on the other hand, provide flexibility and complete control in the design of analyses. Many of these environments, like FSL, FreeSurfer, and AFNI, provide enough modules, and an interface, for a complete analysis akin to the monolithic environments. Others, such as AIR and BrainSuite, focus their efforts for particular needs of the brain mapping community. Programs and subroutines from these packages can be called as individual steps in a compute sequence. Programs can be reconfigured and mixed and matched from various packages. Users can also build their own programs to supplement or replace steps in the sequence resulting in new methodologies and analyses. To date, a disadvantage of modular environments is the difficulty of ensuring interoperability between the various steps. It is also difficult to become an expert on all the packages available from a multitude of laboratories. Often several different programs can each adequately accomplish a task in an analysis. Depending on the dataset and analytic goals, one package may be better suited than another. Choosing the most appropriate programs and assembling the best analysis can be a complex and arduous task. Usually this requires more expertise than a monolithic environment, but offers considerably more flexibility.

Table 1
Software packages for neuroimaging data analysis

<table>
<thead>
<tr>
<th>Software</th>
<th>Available from</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM</td>
<td>Wellcome Department of Cognitive Neurology</td>
<td><a href="http://www.fil.ion.ucl.ac.uk/spm/">http://www.fil.ion.ucl.ac.uk/spm/</a></td>
</tr>
<tr>
<td>BrainVoyager</td>
<td>Brain Innovation B.V.</td>
<td><a href="http://www.brainvoyager.com/">http://www.brainvoyager.com/</a></td>
</tr>
<tr>
<td>SureFit and Caret</td>
<td>Van Essen Lab</td>
<td><a href="http://stp.wustl.edu/">http://stp.wustl.edu/</a></td>
</tr>
<tr>
<td>AIR</td>
<td>Roger Woods, M.D.</td>
<td><a href="http://bishopw.loni.ucla.edu/AIR5/">http://bishopw.loni.ucla.edu/AIR5/</a></td>
</tr>
<tr>
<td>FSL</td>
<td>FMRI B Image Analysis Group</td>
<td><a href="http://www.fmrib.ox.ac.uk/fsl/">http://www.fmrib.ox.ac.uk/fsl/</a></td>
</tr>
<tr>
<td>AFNI</td>
<td>Robert Cox, Ph.D.</td>
<td><a href="http://afni.nimh.nih.gov/afni/">http://afni.nimh.nih.gov/afni/</a></td>
</tr>
<tr>
<td>BrainSuite</td>
<td>Neuroimaging Research Group and the Laboratory of Neuro Imaging</td>
<td><a href="http://neuroimage.usc.edu/brainsuite/">http://neuroimage.usc.edu/brainsuite/</a></td>
</tr>
<tr>
<td>Brain Imaging Software Toolbox</td>
<td>Brain Imaging Centre at the Montreal Neurological Institute</td>
<td><a href="http://www.bic.mni.mcgill.ca/software/">http://www.bic.mni.mcgill.ca/software/</a></td>
</tr>
<tr>
<td>FreeSurfer</td>
<td>CorTechs and the MGH NMR Center</td>
<td><a href="http://surfer.nmr.mgh.harvard.edu/">http://surfer.nmr.mgh.harvard.edu/</a></td>
</tr>
<tr>
<td>Khoros w/Cantata</td>
<td>Khoral Inc.</td>
<td><a href="http://www.khoral.com/">http://www.khoral.com/</a></td>
</tr>
<tr>
<td>SCRRun</td>
<td>Scientific Computing and Imaging Institute at the University of Utah</td>
<td><a href="http://www.sci.utah.edu/">http://www.sci.utah.edu/</a></td>
</tr>
<tr>
<td>AVS/5</td>
<td>Advanced Visual Systems</td>
<td><a href="http://www.avs.com/">http://www.avs.com/</a></td>
</tr>
<tr>
<td>FISWidgets</td>
<td>University of Pittsburgh</td>
<td><a href="http://neurocog.lrdc.pitt.edu/fiswidgets/">http://neurocog.lrdc.pitt.edu/fiswidgets/</a></td>
</tr>
<tr>
<td>LONI Pipeline Processing Environment</td>
<td>Laboratory of Neuro Imaging</td>
<td><a href="http://www.loni.ucla.edu/NCRR/Software/Pipeline.html">http://www.loni.ucla.edu/NCRR/Software/Pipeline.html</a></td>
</tr>
</tbody>
</table>

Organization and logistics

Brain mapping studies process an increasingly large amount of raw data. The data can originate from different scanners and institutions and may be generated from different acquisition parameters and be stored in various formats. During the processing of this data, the creation of many intermediate datasets can be difficult to handle, organize, and store. Ensuring all analysis steps have been performed properly and consistently, while checking for failures in the processing and confounding differences in the results, can be troublesome for even the most diligent investigator. Flaws can go unnoticed until a late stage in processing. Some errors may never be detected.

Resources

Considerable computational power may be needed for increasingly complex population-based analyses. Resource limitations in memory, the number of available processors, disk sizes and speeds, and network speeds can all reduce the feasibility of an algorithm and its practical utility. New techniques that use multivariate statistics (McKeown, 2000;
Calhoun et al., 2001; Suzuki et al., 2002), multiple modalities (Dale and Halgren, 2001; Mohamed et al., 2001), and vector or tensor field calculations (Chung et al., 2001; Arfanakis et al., 2002; Haueisen et al., 2002; Ruiz-Alzola et al., 2002; Westin et al., 2002) are impractical, or impossible, on large datasets without parallel and/or distributed computing solutions.

Interoperability

Individual steps in an analytic sequence may not always work with one another. The output of one step may be incompatible with the input of its successor, especially if these programs were developed at different institutions. This is not limited to simple file conversions. Certain criteria may need to be fulfilled by the data itself. This can include specifications such as allowed value ranges and data types, spatial dimensions and dimension order, and limits on voxel sizes. Data criteria might also include quality or consistency limits such as signal-to-noise ratio, contrast-to-noise ratio, or the preservation of intensity gradients of a certain kernel size. Ensuring the data is properly formatted for the next processing step can take as much time and effort as the analysis. This can become a large obstacle to many collaborative efforts.

Reproducibility

A more general problem in neuroimaging is the difficulty in reproducing analyses as described in a publication. Often details are left out of the methodological description. These details, though crucial for reproduction, aren’t needed for the conveyance of the novel analyses presented in the published work.

Solutions

Scripts are often programmed to solve the difficulties of a modular approach to brain mapping analysis. Scripting languages, such as perl and tcsh, offer many powerful constructs and abilities, such as conditionals, loops, and job control. However, many of these scripts are often written for a single purpose or one time usage. In many situations they may be highly modified over time to adapt to new analyses, keeping old solutions tenuously intact that should have been discarded. The ability to distribute these scripts is dependent on the similarities between environments with no guarantees they will work without modifications. Issues such as resource requirements, temporary files, heterogeneous hardware needs with remote executions, and parallel processing are solved in ad hoc and often laboratory-specific fashions. The programming effort required to fashion a script that solves these issues in a standardized manner would be considerable. These issues are addressed with the development of the LONI Pipeline Processing Environment.

The goal of the LONI Pipeline Processing Environment is to provide a cross-platform, distributed environment for the design, distribution, and execution of brain mapping analysis protocols. It is engineered to be intelligent, efficient, simple, and flexible. The environment was developed with the aim of greater access to various techniques, algorithms, programs, and computing power. The improvements in interoperability can facilitate collaboration between developers of specific tools and cooperation among those with unique datasets. The LONI Pipeline Processing Environment makes studies with a very large number of subjects more obtainable. It enables a high degree of sensitivity and accuracy by supporting the mixing and matching of programs from various packages developed in different laboratories.

Methods

The LONI Pipeline Processing Environment

We developed an analysis strategy that provides the ability to link together individual executable programs, or modules, into a processing tree, or pipeline. A pipeline contains the connectivity information between processing modules that results in an analysis methodology. It allows files, intermediate results, and other information to be accurately passed between individually connected modules. A pipelet is the encapsulation of a pipeline that only completes a subsection of the processing. Allowing pipelets to be formed and treated as modules in larger pipelines provides a conceptual abstraction layer. A pipeline execution environment controls the details of the computation requiring only the connections, or pipes, between the modules that establish the relationships of the pipeline to be specified. Modules, pipelets, and pipelines can be saved to disk at any stage of development and recalled at a later time for modification, use, or distribution.

Module and pipeline descriptions

The description of the modules and pipelines is implemented in the Extensible Markup Language, or XML (information available from http://www.w3c.org/XML/). The LONI Pipeline Processing Environment does not require an application programming interface, or API, to be used. Any well-defined command-line driven processing program or routine can be represented as a module, given default execution arguments and files, saved, and recalled for later use and encapsulation in larger pipelines. The description of a module is accomplished with the aid of a graphical user interface that takes the user through the addition of input arguments, output arguments, input streams, and/or output streams. Arguments may be described as files of specific types, flags, or values and, if need be, are added in the order in which they should be provided to the underlying program. They may be optional or required and can be dupli-
Fig. 1. Modular environments allow the mixture of programs and algorithms to achieve tailored results. The flexibility afforded in these designs may provide increased sensitivity with more accurate results, however, the expertise required to utilize various packages is considerable and the transfer of data from one package to another is often problematic. Programs shown here include Octree Spatial Normalization (Kochunov et al., 1999), or OSN, AFNI (Cox, 1996; Cox and Hyde, 1997), the Brain Surface Extractor (Shattuck et al., 2001), or BSE, Alignlinear (Woods et al., 1998a) and Tracer from AIR, and Mincsample, Mincreshape, and Display (MacDonald et al., 1994) from MNI’s Brain Imaging Software Toolbox. All of these can work together with proper conversion between the six file formats present.

Fig. 2. The LONI Pipeline Processing Environment is a visual programming interface for the simplified design, execution, and dissemination of neuroimaging analyses. Modules can be added, deleted, and substituted for other modules within a pipeline analysis. The environment handles bookkeeping and information transfer between modules and within a pipeline. Its Java-based design and client/server abilities provide a platform independent solution to the interoperation of various modular packages for image processing in brain mapping.
cated when multiple instances of the same argument type are allowed in a program. Each additional argument, other than a flag, provides an additional input or output for the described module that may be used in a pipeline’s connection scheme.

Interface

The graphical user interface, or GUI, and execution engine for the LONI Pipeline Processing Environment is programmed as a lightweight environment in Java 1.3 (available from Sun Microsystems, http://java.sun.com/). It is designed for maximum portability and platform independence. It will run on any system with a Java Virtual Machine of version 1.3 or newer. The design of the interface is that of a visual programming environment (Fig. 2). The GUI allows the user to link together the modules, change the flow of a predesigned pipeline, or replace specific analysis modules for others. There is also a seamless client/server interface that allows the execution of external modules, pipelets, and pipelines defined on outside servers.

The LONI Pipeline Processing Environment presents modules and pipeline analyses to the user in organized tree structures (Fig. 3). There is a user specific section, a system specific section, and a separate section for any connected Pipeline Server. Each of these sections can be subdivided into folders for the logical presentation of programs, packages, and analyses.

Fig. 4. The global variables table separates important inputs and outputs from the inner workings of a pipeline. Variables may be defined for text, values, files, or lists of files and accessed for any input, output, or new variable definition. Variables are accessed with a dollar sign and braces and may be modified with a “:m” suffix, where m is the proper modifier, or the addition, “+,” or subtraction, “−,” of text. The modifiers “r” and “b” shown here represent the input’s root, everything except a file’s extension, and base, the root without the path prefix, respectively.

Fig. 3. Modules and pipelines are presented in multiple organized and cross-referenced tree structures. Separate structures exist for the current user’s personal modules and pipelines, the system the client is running on, and each Pipeline Server the client is connected to. Each structure can be subdivided into groups of folders for the logical presentation of programs, packages, and analyses.

Tables for the definition of global variables are available.
in the LONI Pipeline Processing Environment. The architect of a pipeline uses these tables to make certain inputs and outputs of a pipeline easily accessible (Fig. 4). The user need not delve into the architecture of the underlying pipeline in order to change these inputs and save outputs. Instead, the appropriate global variables are changed to run the pipeline on new data. Variable modifiers, such as accessing the directory a file is in, a filename without the directory information, the base name of a file, or the extension on a file, may be invoked when using a global variable. Variables also may be defined using previously defined variables, modified versions of those variables, and explicit changes to the variable text. This allows for outputs to be defined based on input variables and sets of inputs to be defined based on similar naming schemes. Inputs and outputs can be saved in the same directory, a similar directory structure, or with related filenames.

Execution model

The LONI Pipeline Processing Environment uses a dataflow model (Karp et al., 1967) to provide its parallel processing architecture. The dataflow model is based on a directed graph (Davis and Lowder, 1981) with processing functions, or modules, as nodes and data pipes as edges dictating the flow of data between the functions. The beginning of a data pipe is output data from one module. The pipe directs the data to the inputs for other modules. In this scheme, each module is self-aware and executes upon receiving all of its input data. When a module completes

Fig. 5. This server-based pipeline performs a Talairach registration (Collins et al., 1994), tissue classification (Shattuck et al., 2001), and cortical surface generation (MacDonald et al., 1994) from T1-weighted MRI whole-head volumes. This is a fully automated pipeline requiring no input other than the files to process. The display program from the MNI (http://www.bic.mni.mcgill.ca/software/Display/Display.html) was added as a client-side module to visualize the results. Gray matter is shown in red, white matter in green, and CSF in blue.
execution, the other modules that receive data from it are signaled that needed inputs have arrived. The signaled modules will initialize execution if they are not waiting for additional data. All modules that are not data dependent on one another can execute in parallel without any explicit specification. When no modules are left in a running state, the pipeline has completed its execution.

All intermediate and output data in a dataflow are stored in temporary files unless specified otherwise. Filenames may be explicitly provided or dynamically generated using the variable manipulation language described previously. A unique filename is generated for each temporary file with a filename extension based on the file type encoded in a module’s description. Temporary files are discarded when a workspace is cleared or a new pipeline is selected for editing or execution.

A simple batch-processing scheme is provided in the LONI Pipeline Processing Environment. In the default case, a file list is provided as an input to a module and the execution of each item in the list is in a separate, but identical, module. This results in a listing of outputs in one-to-one correspondence to the file list input. Full parallel execution of file lists in the pipeline is allowed through the dataflow execution model. No special cases need be derived. Modules may also have a file list input that is grouped. All the items in the list are treated as identically typed input arguments to a single module. This is often the case when group statistics are calculated. Data are first preprocessed individually then combined for simultaneous input for processing across the group. Argument lists and multiple input lists to modules are also allowed.

To ensure that the dataflow model does not overwhelm available computing power, there is a first-in–first-out, or FIFO, launch queue. Modules that are ready to execute will wait on the queue for a processor to become available. When the LONI Pipeline Processing Environment is set up on a new computer, the number of accessible processors is set. This value is usually less than or equal to the number of processors available on the platform. A module will move from the launch queue to a running state only when the number of currently running processes is less than the maximum allowed.

Grid computing and batch scheduling systems may also be integrated with the LONI Pipeline Processing Environment to allow processing to occur over a cluster. In this case, the maximum number of simultaneous processes would be set to the limits of the computational grid, and the environment would depend on the grid management software to perform the process scheduling and resource allocation. Modern grid management software can distribute processing based on available processors, memory, operating environments, access to specialized hardware, and various other factors, providing a powerful and heterogeneous extension to the abilities of a pipeline environment.

**Client/server**

The client/server strategy built into the LONI Pipeline Processing Environment is both application and cycle serving. It is platform independent using Java’s remote method invocation, or RMI, libraries. When a client logs into a Pipeline Server, it is presented with all the modules, pipelets, and full pipelines the server can run. The client user can immediately select a pipeline from the server, load it with data, and execute it on the server. Alternatively, selected modules or pipelets from the client and the server can be connected together. These complex pipelines can be saved in the user’s module listing and recalled for later use. Server-based pipelines and pipelets may only be accessed when the user is connected to the appropriate servers.

When server modules or pipelets are executed, their inputs are transferred to the server and the underlying modules are placed in the server’s launch queue for execution. When completed, outputs are transferred back to the client only if they are explicitly saved or needed for a client module to run. Multiple server connections are allowed in a pipeline session. Modules and pipelets may be mixed from many different servers in a single workspace. The client will act as a master to all the connected servers and dictate all the file transfers and module launches that are to occur. If files are provided to the pipeline through URLs, the client or...
appropriate server will download them for use by the appropriate module.

Security

The Pipeline Server has been made secure in several ways. Account access is encrypted and passwords are confirmed through a one-way hash. The client is not allowed to launch any program on the server that is not defined in the server’s process table. Static files on the server are accessible only to the server modules and may not be transferred to the client for its use. Only temporary files written by the pipeline are accessible to the client. Furthermore, only server generated temporary files can be written to the server; the clients cannot explicitly write any files. Finally, it is recommended that the Pipeline server be run under a restricted account that only has access to the components it needs. It is up to the server administrator to ensure that modules do not contain programs that may be used to compromise the server or any computer to which it may be connected.

Results

Modules from AIR 5.2.3, BrainSuite, FSL 1.3, FreeSurfer, the Montreal Neurological Institute’s Brain Imaging Software Toolbox, and the Laboratory of Neuro Imaging have been described and are available through a Pipeline server located at the Laboratory of Neuro Imaging. Included in these definitions are modules for the skull stripping of MRI data (Shattuck and Leahy, 2002; Smith, 2002), RF inhomogeneity corrections (Sled et al., 1998; Shattuck et al., 2001; Zhang et al., 2001), tissue classification (Shattuck et al., 2001; Zhang et al., 2001), linear (Woods et al., 1992; Collins et al., 1994; Woods et al., 1998a; Jenkinson and Smith, 2001) and nonlinear (Woods et al., 1998a) registration of volumetric data, cortical surface extractions (MacDonald et al., 1994; Dale et al., 1999), skull and scalp finding software (Shattuck and Leahy, 2002), and many accessory tools for volumetric image processing.

Many individual processing pipelines have been built and are available on the Pipeline server at the Laboratory of Neuro Imaging. They are used for the nonlinear alignment of MRI data, skull stripping, coregistration, tissue classification, and cortical surface extraction.

Fig. 7. This modified “AIR Make Atlas” pipeline utilizes chirp-z and 3D windowed-sinc interpolation for the linear and nonlinear registration steps, respectively. It also includes a tissue classification step to generate probability fields for gray matter, white matter, and CSF locations in the atlas space. The global variables table shows the important input and output files for this pipeline. The native volume and binary mask lists contain 452 subject volumes in correspondence. The outputs include a full affine aligned atlas with gray, white, and CSF probability fields and a fifth order polynomial aligned atlas to the linear defined space with gray, white, and CSF probability fields.
Fig. 8. The ICBM452 T1-weighted MRI atlas is an intensity average of 452 normal young adult brains existing in a linearly derived, through 12-parameter affine transformations, least distant space from all 452 subjects. The final registration to this space is a fifth order polynomial warp allowing substructures to be more clearly defined in the intensity average. Tissue classification for all 452 subjects is performed in ICBM452 space and probability fields for each tissue type are derived in atlas space. White matter probabilities are shown in green, gray matter in red, and CSF in blue.
of a high-resolution template of gross anatomic delineations to human imaging data, improved skull stripping utilizing a rough registration to a cropped template, average positional and intensity atlas construction from raw datasets, registration of datasets to an atlas space, and the Talairach space (Collins et al., 1994) registration of raw data for tissue classification and surface model generation (Fig. 5). These pipelines were built from modules defined on the server and require no other input than the datasets to be processed. The use of these pipelines on a 64 processor Silicon Graphics Origin 3000 has thus far generated over 300,000 module executions with over 4 terabytes of data generated and/or processed.

Data were analyzed from the International Consortium for Brain Mapping (Mazziotta et al., 1995, 2001), or ICBM, using the LONI Pipeline Processing Environment. Four hundred fifty-two normal adult subjects from three imaging centers were selected to create an average intensity atlas in the average space for the group (Woods et al., 1999). The raw data were skull-stripped using the Brain Extraction Tool (Smith, 2002), or BET, in ICBM305 space (Collins et al., 1994) and the masks of the subdural space were projected back onto the raw data. The procedure, “Aligned BET” (Fig. 6), is located on a LONI Pipeline server, but was controlled by an Intel Pentium III 450 MHz based laptop (Acer TravelMate 340T) with 192 Mb of RAM. The LONI Pipeline server is running on a 64 processor Silicon Graphics Origin 3000 with 32 Gb of RAM, though it was only granted 48 dedicated processors for this test. The skull-stripping pipeline took 1 h 37 min to finish with all 452 subjects, which comprised 1.1 Gb of compressed data. There were 3616 jobs started from the LONI Pipeline server. It generated 4068 intermediate files using 27.0 Gb of disk, uncompressed, and produced an output of 452 masks defining the subdural space for each input MRI.

The masks compressed to 30 Mb of disk space.

The pipeline to build the atlas was derived from the “AIR Make Atlas” pipeline on the same LONI Pipeline server. This pipeline derives affine transformations (Woods et al., 1998b) from all subjects to a single subject. These transforms are averaged to define a “least distant space” for every subject to be transformed into. The data is transformed into the common space and an intensity average is made. All the subjects are realigned with affine transformations to the average template and again averaged to form a linearly defined atlas in both intensity and spatial domains. A final step, to gain definition in the atlas, is a fifth order polynomial nonlinear alignment (Woods et al., 1998b) to the atlas space for each subject. Each subject is transformed into the atlas space and averaged to form the final atlas.

The AIR Make Atlas pipeline was modified (Fig. 7) to use chirp-z interpolation (Tong and Cox, 1999) to apply the linear transformations and a (13 mm)3 windowed-sinc interpolation (Hajnal et al., 1995) for the nonlinear transformations. Additions were also made to apply the previously generated masks of subdural space to the raw data as a first step in the processing and to use the Partial Volume Classifier (Shattuck et al., 2001), or PVC, to generate gray matter, white matter, and CSF masks of each subject in the atlas space. These masks were averaged to produce gray matter, white matter, and CSF probability fields in the atlas space. The pipeline was run on 48 dedicated processors from the SGI server and controlled by the Intel based laptop. It took 19 h 2 min to generate the atlas and tissue based probability fields from all 452 subjects (Fig. 8). There were 7697 jobs launched from the LONI Pipeline server generating 15,391 intermediate files taking up 46.3 Gb, uncompressed, of temporary disk space. The saved results spanned nine volumes of compressed data occupying 25.4 Mb of disk. They consist of a preliminary common space volume, an affine generated atlas volume with gray, white, and CSF probability field volumes, and a fifth order polynomial nonlinearly generated atlas volume in the affine average atlas space with corresponding gray, white, and CSF probability field volumes.

Additional atlases were created for the males, females, left-handers, and right-handers in the dataset. The subjects in the handedness atlases were selected to enforce an equal number of males and females in each atlas. The modifications to the processing pipeline required to generate these separate atlases consisted of changing the input file lists and the location of the output files as appropriate. The processing statistics for each atlas are shown in Table 2.

Further analyses comparing nonlinear intersubject registrations of ICBM data have been performed using modules from the Research Imaging Center at the University of Texas Health Science Center at San Antonio, or UTHSCSA, modules from AIR, and modules from the MNI. Converters for the descriptor file format were provided by UTHSCSA (information available from http://ric.uthscsa.edu/projects/
chsn/des.html) and integrated with the LONI Pipeline Processing Environment. This allowed the Octree Spatial Normalization (Kochunov et al., 1999), or OSN, software from UTHSCSA to be integrated in client environments and used in place of server-based AIR or MNI modules for nonlinear intersubject MRI registration. OSN requires input data to be classified for gray matter, white matter, or CSF. PVC provided this prior to the OSN module execution and data were resampled to match OSN’s input requirements (Fig. 9).

**Discussion**

The total processing time required to assemble the ICBM452 atlas and noted subatlases on the 48 dedicated SGI processors was 2 days 9.5 h. Using sequential executions, or the LONI Pipeline Processing Environment with only a single processor available, would take 39 times as long,1 or 93 days, to generate the atlases. If this process were executed at a command line by a single user, assuming an 8-h work day, it will take up to 3X longer, or 279 days.

The pipelines found on the LONI Pipeline server are common analysis methods. Interaction with the inner workings of the pipeline is not usually necessary. Providing inputs, and sometimes outputs, in the global variables table is all that is required for new datasets to be analyzed. Outputs that are not explicitly defined by the user are often derived from the input data and are visible in the global variable table. Any undefined outputs are discarded. The specific analysis pipelines used for a publication can be placed on the server for others to access and execute on new datasets.

If a pipeline is to be used with unforeseen data, or there are new procedures to make an analysis more sensitive, faster, or produce different results, it can be changed easily. Flags, options, kernel sizes, etc. are all placed in module descriptions and are available to the user for modification. An entire analysis that runs on T1-weighted MRI data may only require a tissue classification step to have its input flag changed from T1 to PD for a proton density dataset to be analyzed. It is possible that this option is not available for the current module being used. Another tissue classifier that works on PD-weighted MRI volumes can replace it. Finally, if there is no module available on the server to accomplish a task, it can be installed on the client side. The transfer of data between the servers and the client will be automatically managed for the user.

**Interoperability and accessibility**

The LONI Pipeline Processing Environment allows many of the interoperability problems that exist between modular packages used in the brain mapping community to be solved in one environment and saved for later use. We have shown the successful execution of pipelines that integrate work from the AIR, FSL, and BrainSuite packages as well as individual programs from the MNI, LONI, and UTHSCSA laboratories. These programs used different file formats, Medical Image NetCDF (available from the MNI, http://www.bic.mni.mcgill.ca/software/minc/minc.html), or MINC, the descriptor file format, and two flavors of the Analyze image format (available from the Mayo Clinic, http://www.mayo.edu/biz/PDF/ANALYZE75.pdf; SPM modifications from the Wellcome Department of Cognitive Neurology, http://www.fil.ion.ucl.ac.uk/spm/distrib.html#AzeFmt), and required a variety of preprocessing steps to be applied before execution. The details of these pipelines can be generalized for the robust interoperability of packages and separated into pipelets for later use. These pipelets can be further improved upon, and possibly replaced by more accurate steps.

The availability and accessibility of many algorithms to perform similar tasks are part of the power of the LONI Pipeline Processing Environment. Many seemingly redundant modules are included in the environment to provide choices and to allow the tailoring of an analysis to specific data and for specialized end products. When one package fails on data presented in a manner not accounted for by its authors, another package may fare better. For instance, we

---

1 The efficiency of the LONI Pipeline Processing Environment is less than 100% due to adjustable delays in the environment’s execution scheme and because of bottlenecks in dataflow. Delays on clients are to ensure Java has time for “garbage collection.” Delays are present on servers to prevent processes from accessing resources simultaneously. Delays are usually tens to hundreds of milliseconds before each individual command execution and are tunable for specific installations.
have processed datasets where the Brain Extraction Tool (Smith, 2002) performs more reliably and ones where the Brain Surface Extractor (Shattuck and Leahy, 2002) performs more reliably. These differences exist due to the sensitivity of an algorithm to noise, smoothness of gradients in the data, different tissue volumes and tissue intensity profile changes from disease, aging, or scanner protocol. The output of a preprocessing module also may be more suited to, or even aware of, the input requirements of a later analysis step. A pipeline currently available on the LONI Pipeline server uses both BFC (Shattuck et al., 2001) and N3 (Sled et al., 1998) in parallel processing streams as corrections for nonuniformity on the input MRI data. This is to ensure the proper application of a tissue classifier (Shattuck et al., 2001) that BFC is packaged with as well as a cortical surface extractor (MacDonald et al., 1994) that N3 is packaged with. The design of an algorithm often incorporates an expectation of the input data.

All of the modules and pipelines discussed in this paper are available on the LONI Pipeline server for use by the brain mapping community. This ability to distribute automated analysis programs and protocols via Pipeline servers and the transfer of XML pipeline description files ensures the accurate conveyance of new ideas from an article to the workbench. The critical factors in the analysis are reduced to the quality and structure of the input data. Collaborations between groups can be facilitated by direct access to described methodology and the assurance of an identical analytic structure. Consolidating tools and analyses into a common framework for the neuroimaging community to utilize and enhance will also be beneficial. Improvements can be disseminated along with the original protocols. The LONI Pipeline Processing Environment is available from http://www.loni.ucla.edu/NCRR/Software/Pipeline.html. Accounts for access to the LONI Pipeline Server can be requested at http://www.loni.ucla.edu/NCRR/NCRR_Colla-bApp.html. Access to the LONI Pipeline Server is currently unlimited for the brain mapping and neuroimaging communities. Validated users may access as much computational power as we can provide.

### Computational abilities

The LONI Pipeline Processing Environment enhances computational abilities in two ways, with distributed processing and through the dataflow model for parallel execution. The distributed processing provided by Pipeline servers allows the analysis of large amounts of data, the study of complex data, such as tensor fields, and the use of data mining techniques to be available for every laboratory. The Java-based servers are simple to set up requiring only the number of available processors, a modules file, and a password file to be specified. The modules file is a Java Archive file, or JAR, that contains the XML descriptions of modules, pipelets, or full pipelines already defined on a system and organized in a directory structure along with any desired icons. Accounts to the server are created by an included Java program to encode user names and encrypted passwords in a file. Pipeline servers require Java 1.3 or newer and RMI to run. HTTP tunneling for firewall compatibility is available so long as the RMI registry is connected to port 80.

The structure of the predominant number of analyses in brain mapping is such that most functions are run on each subject separately and only a few comparison steps are required to run across all the subjects at once. In the atlas generation pipeline there were 17 modules running independently on each of the 452 subjects and 10 that ran once with all 452 subjects as input. In the skull-stripping pipeline there were eight modules running independently on each of the 452 subjects and none that ran only once. Out of 11,310 module executions, 10 did not benefit from subject level parallelizations in the analysis. Of those 10, three were in the middle of the dataflow causing bottlenecks and only two of the three did not benefit from the module parallelization in our dataflow model.

Module and subject level parallelization of most analyses provides close to optimal gains in required processing time when the number of subjects is much greater than the number of available processors (Fig. 10). If there are no bottlenecks in a dataflow, gains are near optimal when the number of subjects meets or exceeds the number of processors. The gains are only near optimal because particular subjects may require more processing time and by chance execute at the end of an analysis or close to last before a bottleneck. These subjects will extend a sequence by however much longer they take to execute than the average subject. Chance executions and bottlenecks that exist in an analysis become a smaller and smaller factor as the number of subjects is increased.

Another possibility is that the number of available processors exceeds the needs of a pipeline. In this situation the time of execution is equivalent to the longest time path through the analysis pipeline. Parallelization of the original code for individual modules would enhance the efficiency and speed of these analyses. Analyses that suffer bottlenecks from many group comparison steps would also benefit from parallelization of the code for group actions. Programs that are currently provided with parallel code may be incorporated into a LONI Pipeline Processing Environment using a grid computing solution that will allocate multiple units of a resource, such as processors, to an individual program when available. Future versions of the LONI Pipeline Processing Environment will allow a module to be specified to use multiple processors. The required number of processors will be provided to the executing module as a block, and released back into the processor pool as a block at the end of execution.

### Similar outside efforts

Several other efforts complement the LONI Pipeline Processing Environment. Khoros (Konstantinides and Ra-
Khoros (Konstantinides and Rasure, 1994; Barrera et al., 1998) with Cantata is a commercial endeavor that provides a similar visual programming environment to the LONI Pipeline Processing Environment and also uses a dataflow paradigm. Khoros, however, requires an API to link code and algorithms into the environment. This requires a programmer to use the provided libraries to integrate new modules into the processing scheme. The distributed processing scheme provided with Khoros does not allow independent external servers. Instead, servers must be tightly linked with a common file system preventing independent and secure external resources from being offered.

SCIRun (Parker et al., 1997) is a freely available environment for scientific modeling, processing, and visualization. It also uses a dataflow visual programming interface. SCIRun focuses on steering an analysis or model for interactive visualization and therefore requires an integrated programming effort with the environment. An API is provided that furnishes the developer with libraries, handles, and data types to tightly link new code with their existing model. This gives SCIRun complete control over the execution strategy, but limits external development. Most of the benefits of the API are lost when studies with a large number of subjects are being conducted. The benefit of interactive steering becomes unlikely and subject level parallelization can do as well without the API. AVS/5 is a commercially available visualization and processing environment similar in scope to SCIRun.

FISWidgets (Fissell et al., 2003) is a freely available API and environment that allows Java-based GUIs to be programmed and attached to any command line program. It comes with predesigned interfaces to many neuroimaging analysis packages including AIR and AFNI. It allows the packages to be integrated in a “GlobalDesktop” for sequential execution on a single workstation. The passing of filenames and variables must be explicitly specified and controlled by the user.

Similarities between the LONI Pipeline Processing Environment and these other environments include the visual interface they all provide to enhance and ease the design of methodologies and processing power.

---

**Table 3**

<table>
<thead>
<tr>
<th>Package</th>
<th>Main purpose</th>
<th>Platforms</th>
<th>API</th>
<th>Integration of previous code</th>
<th>Integration of new code</th>
<th>Modules</th>
<th>Distributed computing</th>
<th>Application serving</th>
<th>Protocol serving</th>
<th>Integration of multiple application environments</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONI Pipeline</td>
<td>Protocol development and distribution; dissemination of computing resources</td>
<td>Java</td>
<td>Not needed</td>
<td>GUI-aided description of any executable, no recompiling</td>
<td>GUI-aided description of any executable</td>
<td>Provides module descriptors to run commonly used neuroimaging software</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes, as long as they share a common filesystem</td>
<td>Free</td>
</tr>
<tr>
<td>SCIrun</td>
<td>Visual programming and steering, module development, computational efficient processing</td>
<td>UNIX, SGI IRIX</td>
<td>Yes</td>
<td>Recompilation with API</td>
<td>Compilation with API</td>
<td>Many modules provided for various mathematical and visual tasks</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Free</td>
<td></td>
</tr>
<tr>
<td>Khoros w/ Cantata</td>
<td>Visualization and module development environment</td>
<td>UNIX, Windows</td>
<td>Yes</td>
<td>Recompilation with API</td>
<td>Compilation with API</td>
<td>Many modules provided for various mathematical and visual tasks</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Commercial</td>
<td></td>
</tr>
<tr>
<td>AVS/5</td>
<td>Visualization and computation environment</td>
<td>UNIX, Linux</td>
<td>Yes</td>
<td>Recompilation with API</td>
<td>Compilation with API</td>
<td>Many modules provided for various mathematical and visual tasks</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Commercial</td>
<td></td>
</tr>
<tr>
<td>FISWidgets</td>
<td>GUI builder and sequential execution environment</td>
<td>Java</td>
<td>Yes, for wrappers</td>
<td>Java code is written from their API to wrap programs</td>
<td>Commonly used neuro imaging software that has been wrapped</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Free</td>
</tr>
</tbody>
</table>

**Note.** A comparison of the LONI Pipeline Processing Environment with other environments for scientific processing and visualization shows the similarities and key differences between them. The LONI Pipeline Processing Environment stresses ease of integration, use, and distribution of methodology and processing power.
The goal of the LONI Pipeline Processing Environment is to make the development and execution of any analysis as simple as possible. This involves handling the details of an analysis, providing the constructs necessary to include new algorithms in the environment, and building general and specific methodology.

In future releases, the LONI Pipeline Processing Environment will include automatic format conversions and fully integrated visualization and tailored display of results. Currently, modules providing access to individual external converters or visualization programs are utilized. This will be upgraded so that general conversions from one volumetric, surface, contour, or transformation format to another similar format will be selected and applied automatically to ensure compatibility between modules. Similarly, visualization schemes will be tailored in a data-driven manner. List processing enhancements, conditionals, and iteration constructs are planned for inclusion as well. The integration of database query, retrieval, and submission systems in the LONI Pipeline Processing Environment are also planned. Finally, in future releases, the LONI Pipeline Processing Environment will provide intelligent agents to assist in the design of a new analysis.
Conclusions

The advanced techniques available in neuroimaging provide a landscape where studies of unprecedented size and complexity are being formulated. The LONI Pipeline Processing Environment makes those studies accessible and achievable in a reasonable amount of time. The ability to find smaller effects using larger datasets, the assurance that all data were analyzed identically and as desired, the time saved during the processing and construction of an analysis, and the lowered costs of shared computing resources are current benefits of the LONI Pipeline Processing Environment. Future enhancements to this environment will provide integrated access to visualization, databases, and intelligent agents. We look forward to many additional modules and pipelines being added in the near future.

Acknowledgments

This work was generously supported by grants from the National Institute of Mental Health (1 P20 MH65166 and 5 P01 MH52176) and the National Center for Research Resources (2 P41 RR13642 and 2 M01 RR00865), with a supplement by the Biomedical Informatics Research Network (2 P41 RR13642 [http://www.nbirn.net/]), funded by the National Center for Research Resources. D.E.R. is supported, in part, by an ARCS Foundation scholarship and a National Institutes of General Medical Sciences Medical Scientist Training Program grant (GM08042). The authors wish to thank Joseph D. Kilisics, Allan MacKenzie-Graham, Roger P. Woods, and the other members of the Laboratory of Neuro Imaging for their help and support.

References


