

Laboratory of Neuro Imaging Pipeline Environment, v1.0.3

Concepts: David E. Rex and Arthur W. Toga

Design: David E. Rex

Implementation: Jeffrey Q. Ma and David E. Rex

Support: NCRR (RR13642), ICBM NIMH/NIDA (P20 52176)

Copyright 2001, Laboratory of Neuro Imaging

Thank you to Roger Woods for providing the Tracer program.

Tracer Copyright 2001, Roger P. Woods

HOW TO SET UP AND RUN THE LONI PIPELINE	1
TERMS AND CONDITIONS:.....	1
CONTENTS OF THE .ZIP FILE	2
QUICK INSTALLATION:.....	3
THE LONI PIPELINE ENVIRONMENT – USAGE INSTRUCTIONS	3
BACKGROUND.....	3
INTERFACE	3
INSTALLATION.....	4
USAGE SUMMARY.....	5
MODULE LISTINGS.....	6
EDITING A DEFINED MODULE	6
EDITING INDIVIDUAL I/O TABS	7
CREATING A NEW PROGRAM EXECUTION MODULE.....	8
PIPELINE EXECUTION AND CREATION.....	9
LIST PROCESSING	11
CLIENT / SERVER.....	12
PLUGINS.....	12
TROUBLESHOOTING	13
DEMONSTRATION PIPELINES INCLUDED WITH THE DISTRIBUTION:	14
DEMO PIPELINES:	14
1. Tissue Volumes Running Time: 1 minute.....	14
2. Multiple Tissue Volumes	15
3. Intracranial Volume - Linear Spatial Normalization.....	15
4. Linear Aligned Average.....	15
5. Hemispheric GM Volumes	16
6. Make Atlas - Tissue Probabilities.....	16

HOW TO SET UP AND RUN THE LONI PIPELINE

TERMS AND CONDITIONS:

1. Permission is granted to use this software without charge for non-commercial research purposes only. You may make verbatim copies of this software for personal use, or for use within your organization, provided

that you duplicate all of the original copyright notices and associated disclaimers.

2. YOU MAY NOT DISTRIBUTE COPIES of this software, or copies of software derived from this software, to others outside your organization, without specific prior written permission from the University of California at Los Angeles or the Laboratory of Neuro Imaging.
3. THE SOFTWARE IS PROVIDED "AS IS," AND THE UNIVERSITY OF CALIFORNIA AT LOS ANGELES, THE LABORATORY OF NEURO IMAGING, AND THEIR COLLABORATORS DO NOT MAKE ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, NOR DO THEY ASSUME ANY LIABILITY OR RESPONSIBILITY FOR THE USE OF THIS SOFTWARE.
4. This software is for research purposes only and has not been approved for clinical use.

CONTENTS OF THE .ZIP FILE

Demos.txt	Describes the included demonstrations
Disclaimer.txt	Above disclaimer
Instructions.html	Usage Instructions - Web Document
Pipeline.jar	LONI Pipeline - Java Executable
README	
Troubleshooting.txt	Basic quick and dirty help

EXECS DIRECTORY:

execs/ij.jar	ImageJ
execs/MultiTracer.jar	Advanced Analyze Image and MINC Viewer and ROI Tracer - Java Executable
execs/showInfo.jar	Simple program to display text in a message window
execs/Tracer.jar	Analyze Image Viewer and ROI Tracer - Java Executable

MACROS DIRECTORY:

macros/*	ImageJ macros
----------	---------------

MODULES DIRECTORY:

Modules/AIRlogo.gif	A.I.R. logo from Roger Woods
Modules/loni_logo.gif	LONI logo from the Laboratory of Neuro Imaging
Modules/microscope.gif	ImageJ logo from ij website

MODULES VIEWERS:

Modules/Viewers/ImageJ.pm	ImageJ Pipeline Module Descriptor
Modules/Viewers/MultiTracer.pm	MultiTracer Pipeline Module Descriptor
Modules/Viewers/showInfo.pm	showInfo Pipeline Module Descriptor
Modules/Viewers/Tracer.pm	Tracer Pipeline Module Descriptor

MODULES DEMOS:

Modules/Demos/hemi_measure_GM.pm
Modules/Demos/icv_sn.pm
Modules/Demos/linear_avg.pm
Modules/Demos/make_atlas.pm

Modules/Demos/tissue_volumes.pm
 Modules/Demos/tissue_volumes_mult.pm

MODULES DEMOS PIPELETS:

Modules/Demos/Pipelets/show_label_volume.pm

PLUGINS DIRECTORY:

plugins/* ImageJ plugins

PROPERTIES DIRECTORY:

properties/pipeline.properties Default Pipeline properties file

QUICK INSTALLATION:

- Make sure Java version 1.3 or greater is installed on your machine.
- Unzip the archive into its own directory.
- On a fully Java enabled system (most windows and Solaris platforms with Java installed, as well as many other systems) select the Pipeline.jar file for execution (double-click on its icon). On a system with java installed, but not fully integrated, please execute the pipeline with "java -jar Pipeline.jar" or provide the fully qualified path names if required by your setup.
- If you are behind a gateway with network address translations (NAT) then enable 'callback=false' at the end of the pipeline properties file. This will also allow you to use http-tunneling.

For detailed installation instructions that allow proper modifications to be made for specific system enhancements please refer to the installation section of the Instructions file.

In order to add more modules to the LONI Pipeline environment and get full instructions on how to properly use the environment please refer to the Instructions file provided.

THE LONI PIPELINE ENVIRONMENT – USAGE INSTRUCTIONS

BACKGROUND

The LONI Pipeline is a simple graphical environment for constructing complex scientific analyses of data. It provides a visually intuitive interface to data analysis while also allowing for diverse programs to interact seamlessly. The Pipeline allows researchers to share their methods of analysis with each other easily and provides a simple platform for distributing new programs, as well as program updates, to the desired community. The environment also takes advantage of supercomputing environments by automatically parallelizing data-independent programs in a given analysis whenever possible. Finally, the LONI Pipeline can run in a client-server mode, allowing access to compute servers running analysis software that benefits from a dedicated machine with vast computational resources.

INTERFACE

The LONI Pipeline's graphic interface consists of listings of predefined modules, pipelets, and pipelines (left); a message window (bottom right); and a workspace (right). Analysis protocols, or pipelines, are constructed and modified in the workspace. Modules are connected together by data pipes, carrying data from program to program, and even from pipeline to pipeline. Changes to options, inputs, and outputs can be made by selecting the proper module tab to edit, or by selecting the proper module, or encapsulated pipelet, to edit. The message window provides the user with useful information—mostly during an execution run—on an individual module's status, current file transfers in progress, and executions of underlying processing programs.

INSTALLATION

The LONI Pipeline is distributed as a single Java jar program, pipeline.jar. It can be run instantly upon download on any Java (version 1.2 or greater) enabled system using its default settings. Command line arguments or a properties file may be used to change many default settings in the Pipeline environment.

To execute the LONI Pipeline Environment on a Java enabled system that has already associated .jar files with the Java environment consists of simply selecting pipeline.jar ('pipeline') for execution (i.e. double-click the pipeline icon on Windows based and Macintosh systems). On systems with Java installed, but not automatically associated with the .jar files, execution of the Pipeline from the command line is simply 'java -jar c:\pipeline\pipeline.jar' (replace 'c:\pipeline\' with the correct relative path to the pipeline installed directory).

The following settings for the LONI Pipeline may be modified via the java command line [using the `-D<name>=<value>` option (multiple `-D` calls are allowed on a command line)] or by a file called 'pipeline.properties' located in the property.dir (below). Default settings for the pipeline environment are as follows (note: `<pipeline.dir>` is the location of the pipeline.jar file):

`max_running_threads = 1`

Tells the Pipeline Environment how many processors to utilize. If it is greater than 1, the `-native` option should also be added to the java command line so that java is forced to use the OS's POSIX compliant threading system instead of its default internal green thread system (i.e. 'java -native -Dmax_running_threads=4 -jar pipeline.jar').

`property.dir = <pipeline.dir>/properties/`

Location of the pipeline.properties file, if one exists.

`sys_mod.dir = <pipeline.dir>/modules/`

Where the system modules are for all users to access. For organizational purposes, this directory may have a complex directory structure of its own. This structure will be replicated in the module listing, allowing the .xml files describing modules and pipelets to be categorized by the pipeline administrator.

`user_mod.dir = <user_home.dir>/PIPE_MODULES/`

Location of an individual user's modules, pipelets, and pipelines. This directory also may have a complex directory structure beneath it, allowing categorization of the various processes.

`user.dir = <current_working_directory>`

Where the Pipeline Environment is started from. All relative pathnames to files will be started from this location.

LookAndFeel = javax.swing.plaf.metal.MetalLookAndFeel

The Java based look and feel the LONI Pipeline Environment will use.

On systems where many of the above options are modified, it is simple to create the pipeline.properties file to remove clutter from the java command line and, when available, to allow proper execution by selecting the pipeline icon. Here is an example of such a file:

```
#properties file of the pipeline.jar java executable

#look and feel class
lookAndFeel=javax.swing.plaf.metal.MetalLookAndFeel

#Directory where the system modules live
sys_mod.dir=/usr/lshare/java/bin/properties/pipeline_mods/

# the number of exec threads which can be run at any time
max_running_threads=4
```

Another possibility for the Pipeline's execution may be from a distributed environment, where different machines, and different users, are given various access to the pipeline. This would be controlled by a shell script (i.e., tcsh) that makes the needed decisions and executes the LONI Pipeline:

```
#!/bin/tcsh -f

echo "\n   *** Starting the LONI PIPELINE PROCESSING ENVIRONMENT ***"
echo "**** Note:  The Pipeline should NOT be run in the background ****"

set userd = ${PWD}
set jard = /usr/lshare/java/bin
set maxt = 1                # default: use only 1 processor

if ($HOST == big_machine)    set maxt = 5
if ($HOST == other_big_machine) set maxt = 4
if ($HOST == huge_machine)  set maxt = 20

/usr/java/bin/java -native -Dproperty.dir=${jard} -Duser.dir=${userd} -
Dmax_running_threads=${maxt} -jar ${jard}/pipeline.jar

echo "****Exiting the LONI PIPELINE PROCESSING ENVIRONMENT****"
exit
```

USAGE SUMMARY

- Running the Pipeline Environment ('pipeline' on LONI machines or double-clicking the 'pipeline.jar' icon on most Java enabled Windows environments) will bring up a graphical user interface that has already imported all the system modules, pipelets, and pipelines. As the user defines their own modules and pipelets they will be entered in the user's personal space and automatically retrieved on subsequent pipeline use. To switch between user defined and system defined modules, click on the desired listing at the top of the module listings.
- A category of modules or pipelets may be accessed by double-clicking on its folder in the module listings.
- Open a module or pipelet into the workspace by double-clicking on it.
- To add a module or encapsulated pipelet from the listings to the workspace drag-and-drop it. If it is the first change to a pre-existing pipe/module you will be prompted for a new name for this pipeline (note, no information is saved until the environment is told to do so).

- To change the contents of an input or output tab double-click on it. A dialog box will appear where input may be entered or files may be selected for input or output.
- To connect a module's outputs to inputs of other modules, click on the desired output and then on the desired input. A pipe will be drawn connecting the two. Information passing out of one output will be automatically transferred to the next module for you. If there is no filename provided at the output a temporary file is used that is later deleted. If a filename is provided at the output then the data is put in that permanent file.
- To edit a module's options, inputs, and outputs all at once, double-click on it in the workspace and a dialog box will appear with all the information necessary for a single module. If it is an encapsulated pipelet that has been double-clicked, then that pipelet will open in a new workspace to be modified.
- To run a pipeline choose Execution->Run. The environment will check that all necessary inputs exist, and if so, it will execute the analysis pipeline defined in the workspace. If not it will ask the user if they wish to run in interactive mode, providing the missing inputs as needed.

MODULE LISTINGS

Previously defined modules, pipelets, and pipelines may exist in many different lists of modules. The "System Module List" is for those entities that every user on the system can see and use. They are robustly defined and expected to accurately describe their underlying programs. The "User's Module List" contains modules defined by the user that can only be seen and used by that user. A connected server's module listing will appear for each LONI Pipeline application server the user has connected to. It will provide the client with all usable modules, pipelets, and pipelines available on that server.

Any directory structure imposed under the locations to look for modules [usually <pipeline.dir>/modules/ for system modules and <user_home.dir>/PIPE_MODULES/ for individual user modules (look under installation for details)] will be replicated in the module listings. This is useful for organizing groups of modules into categories.

Right-clicking on a module, pipelet, or pipeline in the Module Listings will give a menu allowing the selected entity to be opened into the workspace, or deleted from the listing. If double-clicked a module will appear in the workspace, a pipelet or a pipeline will open into the workspace showing its internal modules and connecting pipes. If drag-and-dropped all entities will appear as a single module box within a surrounding pipeline. The user will be prompted for a new name for the entity they are defining, though nothing is saved till the user selects save.

EDITING A DEFINED MODULE

Modules may be opened into the Pipeline workspace by double clicking on them in the module listings. If the module represents a pipelet or pipeline it will open in its entirety, showing all encompassed modules and connections. Within the workspace, double-clicking on any module will open that particular entity. If it is an encapsulated pipelet, a new workspace will appear in a new window with the pipelet's modules and connections displayed for editing. If it is a basic module the user double-clicks on, a module-editing window will be opened for the user to modify all inputs, outputs, flags, and the command line associated with the selected module.

- All required inputs and outputs are available for designation. Optional inputs, outputs, and flags must be checked in order to use them or modify their values.

- If files, values, or text are explicitly entered, they will be utilized precisely as typed in the first input window after the argument description. Many options may contain previously designated default values for the user. These are only suggestions from the author of the module, and may be modified at will.
- If the input is received from another module it should only be modified after the pipe is explicitly deleted (right-click the input tab and choose “delete connection”). It will have a “-OutFile#” associated with it and it’s “Read From” section will contain information about the module it is dependent on.
- Output files may be explicitly defined, described as manipulations based on input variables (described below), or left blank for temporary files to be generated, used, and discarded when the user exits the pipeline.

Right-clicking on a module will bring up a menu allowing the user to:

- *Open* the module (as described above)
- *Delete* the module from the current workspace being edited
- *Help* – ask for help on the selected module.

EDITING INDIVIDUAL I/O TABS

The input (entering in the top of a module) and output (leaving through the bottom of a module) tabs of a module may be modified individually. Double-clicking on a tab brings up a dialog box where the user can enter filenames, values, or textual information to be used by the underlying program. If the I/O tab’s use is optional for the correct execution of the module, there will be a check box to turn it on and off. When selected the user may enter the information by hand, or select a file via the browse button. If a list of inputs is to be processed, that list can be provided as a file (*filename.list*) and, if desired, modified with the “create list” button (more on this under List Processing). A list can also be produced at outputs by providing a file list, or by using the manipulation of the input variables to a given module to produce its outputs (described below).

Variable Manipulation: A powerful, but simple, way to produce output names for modules is to use a module’s inputs to define its outputs. The LONI Pipeline provides this ability by binding all input tabs on a module to variables and allowing those variables to be manipulated into an output argument or filename. This becomes especially useful when processing many files through the same pipeline—each output will be unique as long as it is dependent upon unique inputs for its formation. Variable manipulation works as follows:

- Each input tab is bound to a variable that is the index number of that tab. Tabs indexed from left to right.
- The variable is accessed by using the dollar sign and braces as a key:
 - `#{2}` -- would access the text stored in the second input tab.
- Text can be added to either side of the variable:
 - `#{2}` = brain1.img
 - `/data/study/ + #{2}` = /data/study/brain1.img
 - `#{2} + .gz` = brain1.img.gz
 - `/data/ + study/ + #{2} + .gz` = /data/study/brain1.img.gz
- Text may be subtracted off the end of the variable:
 - `#{2} – 1.img` = brain

- /data/study/ + \${2} – 1.img + 3.img = /data/study/brain3.img
- The variable can also be manipulated internally:
 - \${1} = /data/study/brain1.img -- the input variable
 - \${1:f} = brain1.img -- only the filename
 - \${1:d} = /data/study/ -- only the directory
 - \${1:r} = /data/study/brain1 -- rootname (no extensions)
 - \${1:b} = brain1 -- basename (no directory or extensions)
 - \${1:e} = img -- terminal extension
- And the actions may be combined:
 - \${3} = /data/study1/brain4.img
 - \${3:d} – study1/ + results1/ + \${3:b} + _scored.img = /data/results1/brain4_scored.img

If outputs are generated by variable manipulation, and one of the used inputs is an argument list, then a new variable will be generated for each item in the list based on the corresponding input for the proper place in the list.

Right-clicking an I/O tab will give a menu of options to the user:

- *Edit* – brings up the dialog box described above.
- *Delete connection* – removes the connection associated with the selected input.
- *Extend the tab* – for inputs and outputs that are allowed to be used multiple times in a program, this option will add a new tab that mirrors the abilities of the tab it was created from. Now data from many sources can be funneled into the same module.
- *Delete the tab* – allows extended tabs (above) to be deleted.
- *Exportable* – tabs in an encapsulated pipelet may be placed on their outer module layer by selecting this option.
- *Overwriting* – if selected outputs will clobber previous existing files of the same name, if unselected the pipeline will check for existing outputs and suspend execution when a file already exists allowing the user to fix the conflict or terminate execution.
- *Group the list* – if an input is a list it can be run as multiple instantiations of the same module, or it can be grouped as one large input to a single module (more on this under List Processing).

CREATING A NEW PROGRAM EXECUTION MODULE

The LONI Pipeline Environment uses XML descriptor files to encode all program modules, pipelets, and full pipelines. Folding new programs into the environment is simple. Dialog boxes prompt the designer for the proper information to be encoded in the program's XML description and the environment constructs the file for the user. The following is a step-by-step description of the creation of a new module:

- 1) Choose File->New, Ctrl-N, or right-mouse click in the workspace and choose New. A dialog box will appear to assist the user in creating a new program module for use in the LONI Pipeline Environment.
- 2) Fields to be supplied in the New Module window:
 - a. *Name* – A descriptive name for the module to be used in the module listings and to identify the module in the workspace.
 - b. *Location* – Where the program can be found on the system.
 - c. *Command* – The actual executable's name on the system. This will be augmented by inputs and outputs as they are added. Any needed command line options that aren't described by an input or output tab should also be added to this command section.

- d. *Help File* – An optional help file for the module may be provided here to assist any users with the details and functioning of the particular module. The file should be an html file found either on the current system, or on the web, and provided via a URL.
- 3) *Command line description* - Arguments are added to the command in the order they would appear if run via a command line interface. Any arguments the user wishes to not allow the editing of can be added by hand to the command section. The arguments to be accessed by the pipeline interface are generated by the “Add Input” and “Add Output” buttons at the bottom of the module creation window. They will add placeholders for the pipeline environment’s use as they are created. A new dialog box will be opened for every input or output added:
- a. *Add Input*
 - i. *Description* – To convey to the user what is to be provided by this input.
 - ii. *Type* – The specific kind of input this is to be (a file, a value, text, or a flag to turn an option on or off). If it is a file, the specific type of file can be mentioned (usually the filename extension).
 - iii. *Synopsis* – Describes how to format the argument if it isn’t a simple argument. For instance, flags that appear on command lines alone, or in conjunction with another argument, would be placed in this box.
 - iv. *Extendable* – A toggle informing the environment whether multiple copies of this argument are allowed on the command line. If so, and if more are generated in a pipeline’s creation, they will be sequentially inserted in the command line in place of the placeholder.
 - v. *Optional / Required* – If an argument is required, the LONI Pipeline Environment will check for its existence before a Run is allowed. If it is optional, it may be provided, or not. A check box will be provided in the module for optional arguments so they can be toggled on or off. If on, they are considered required for that run. If off, they are ignored and not inserted in the command line.
 - b. *Add Output*
 - i. *Description, Type, Synopsis, Extendable, Optional / Required* – Same as above
 - ii. *Overwriting* – The Pipeline will check for output files before it executes a module. If overwriting is on it will clobber any files named the same as an output. If overwriting is off the environment will stop execution and prompt the user for what it should do.
 - c. *Delete* – Allows for the correct removal of an input or output added by the “Add” buttons.
- 4) When all inputs and outputs have been added, the designer of the module can default all the values, text, filenames, and flag toggles to their liking. If text, values, or files are added to optional arguments and they are toggled off, the arguments will still be ignored. The defaults provided will be usable by the user when they toggle to argument back on, or new arguments can be provided at that later time.
- 5) *Finish* or *Save* the module. Files are saved as XML descriptor files. They must have a .xml extension to be automatically imported by the LONI Pipeline. The module will appear in the “User’s Module List” under the name given to it and is ready to be used. For a module to appear in the “System Module List” it must be explicitly placed there by an administrator of that machine.

PIPELINE EXECUTION AND CREATION

Execution: Existing pipelines may be selected by double-clicking on them in the Module Listings. They will appear in the workspace. Inputs and outputs may be modified and, when ready, the user can execute the

pipeline by choosing Execution->Run from the main menu. The LONI Pipeline Environment will run any data-independent modules in parallel, allowing multiple concurrent processes up to the maximum number of threads allowed on a given machine. If using a server for certain modules, those modules will also be made parallel on the server using as many processors as designated by the server's administrators.

If required inputs are missing from the pipeline, either those that are always required or optional inputs that have been turned on, the user will be prompted to cancel the run, or run in interactive mode. In interactive mode the user can make modifications to the pipeline while it is running. If a required input is still missing when the module is to be executed, the user will be prompted for that input while execution of the pipeline is paused. The pipeline will resume execution when the user selects "Resume to run" from the module right-click menu.

The LONI Pipeline Environment will also check for the existence of outputs before modules are run. If an output exists and overwriting for that output is turned off, the pipeline will signal an error to the user and pause execution for the user to rectify the error. After overwriting is toggled on or the filename is changed the user can select "Resume to run".

Creation: A new processing pipeline may be composed through the modification of already existing pipelines, the connecting of predefined pipelets, and by the combining of independently defined program execution modules. If the available modules do not fit the user's needs, they may define new modules as previously described. Editing of pipelines may occur as follows:

- To add a module or encapsulated pipelet from the module listings to the workspace drag-and-drop it. If this is the first addition to the workspace the user will be prompted for a name for the developing pipeline.
- To delete a module right-click on the module and select delete.
- Modules may be edited in their entirety in the pipeline being formed by double-clicking on the module or right-clicking and choosing open.
- I/O tabs may be edited by double-clicking on them or by right-clicking and choosing edit.
- Mouse-over events will display to the user what a specific tab expects as its input, or produces as its output.
- Pipes are used to link modules for the passage of data files and arguments. Click on an output and then on an input of the same type to connect the two.
- To delete a connection right-click on the input tab the connection goes to and choose "delete connection".
- When encapsulated pipelets are added to the workspace, only I/O tabs that are marked as exportable will be accessible outside of the pipelet. This is accomplished by a right-click on the tab and toggling the exportable option on. It will appear on the outer module layer to allow the passage of data and arguments.
- Opening a module for an encapsulated pipelet (right-click-> open or double-click the module) will open a new workspace in another window. This allows options, arguments, and exportable status to be easily modified within the pipelet.
- To save a constructed pipeline select "Save As" under the File main menu or right-click on the workspace background to get the menu. Files are saved as XML descriptor files and the file should be saved with a .xml extension to be automatically parsed on startup by the LONI Pipeline. The constructed pipelet or pipeline will appear in the "User's Module List".

LIST PROCESSING

Any argument in a pipeline may also be given as a list of arguments. This is useful to allow the simple application of the same processing steps to many files, to use multiple values and variations of inputs on a single file, to send many files into a single analysis module for contrasts to be applied across all the files, or for some combination of these possibilities. If a list is an input for a module, the default operation for the module will be to run in a looped mode, akin to a foreach in many shells. This means that each argument in the list will spawn an execution of the module with all the other arguments filled in according to their single input, or according to other lists to be traversed in parallel. If multiple lists are available to a module they will all be traversed argument for argument till the end of the longest list, with shorter lists looping over their inputs if necessary. All of the generated instances of the module's operation will be added to the run queue and processed in parallel (up to the number of processors available on the system or server). This can also be seen as generating many command lines with differing arguments.

If an argument list to a module is "grouped" then it is treated as an extended argument to a single module. Loops for multiple argument substitution to many equivalent modules will not be generated due to a grouped list. Instead this is equivalent to supplying many arguments to a single command line. If there happens to be another list argument for the module that is not grouped, it will cause the generation of multiple executions of the module based on the non-grouped lists, and using the grouped list in every instance.

A common usage scenario for a pipeline will be to have an input list for the processing of individual files in a file-by-file fashion. That list will generate new lists for each of the following modules and eventually feed into a module that groups the list and analyzes the data across the study. This may be the end of the pipeline, or it may feed into other modules using the result of that grouped analysis for the further processing of the individual files, again possibly feeding into a module that groups lists.

To use a list as an argument:

- The list is a text file with a .list extension (for instance: control_subjects.list).
- The file has an argument on each line.
- It may be created previous to the use of a pipeline or generated by the user from within the pipeline environment.
 - 1) Select an I/O tab to edit.
 - 2) Enter the name of a list or a file to store a list in.
 - 3) Click on "Create List" if the list needs modification.
 - To add files, navigate the directory structure and select the files for the list
 - To add text or numerical arguments to a list type them under "Selection" and click "Add."
 - To remove an entry, select the entry in the list, and click on "Remove."
 - When finished click "Done" and the file will be saved under the name provided previously.
- To treat a list as a grouped list, right-click on the appropriate input tab and select "group the list."
- When a module has at least one non-grouped list as an input, it will always produce lists at its outputs.
 - If no output is provided, temporary files will be used and a temporary list will be formed.
 - Variable manipulation can be used to produce multiple saved outputs and will automatically generate a temporary list when necessary.

- The list may also be formed by providing a .list file to be used for filenames for the outputs.

CLIENT / SERVER

By connecting to a LONI Pipeline Server a user gains access to the modules, pipelets, pipelines, and files existing on that server. From the user's perspective the modules, pipelets, and pipelines are used just as any local ones would be. The processes, however, that are defined on the server will execute on the server, bringing the power of server's processors to the client. The process execution model continues to be fully data-driven in client/server mode. Data-independent modules will be run in parallel up to the number of processors the client or server is defined to use. Access to files on the server, which possess proper permissions, is also allowed in order to provide access to templates and models the modules may need, and even to access databases of files the server would be allowed to use.

Use of a pipeline server:

- The administrator of a pipeline server must explicitly provide the user with an account in order to gain access that server.
- Access to the server from a pipeline client is obtained by selecting "Connect to servers" from the Server main menu.
 - The user enters the server's address in the server dialog and selects an action.
 - Login – to gain access to the selected server; they will be prompted for their user name and password.
 - Change Passwd – to change their current password for that server; they will be prompted for their user name, old password, and two copies of their new password.
- Once logged into the server a new Module Listing will appear with all the modules, pipelets, and pipelines that are available on the server.
 - These entities can be used just like any on the client side.
 - When editing I/O tabs of server based modules, arguments or files (including lists) can be Local or Server based.
 - If only a Local or Server argument is specified, that argument or file will be used.
 - If both a Local and a Server argument or file is entered, the Server argument or file will override the Local one.
 - In order to save outputs locally they must be given filename in the Local section of the output tab (directly, through a list, or via variable manipulation).
- When a pipeline is executed that contains server-based modules, all file transfers between the client and server will happen automatically as needed.
- Any client pipelets or pipelines that depend on a server will generate an error if access is attempted without connecting to the correct server first.
- Programs that are not defined on the server will not execute on the server.

PLUGINS

Plugins Folder:

This folder is where user-written plugins go. It contains two sample plugins. "RedAndBlue_.java" is an example acquisition plugin for acquiring or generating images. "Inverter_.java" is an example image filter. Many more are available at "<http://rsb.info.nih.gov/ij/plugins/>". Plugins with an underscore in their name are

automatically installed in the Plugins menu. Plugins in subfolders are installed in submenus. Assign keyboard shortcuts using the Plugins/Shortcuts/Create Shortcut command. Install plugins in other menus using the Plugins/Shortcuts/Install Plugins command.

Compiling Plugins:

To compile and run a plugin, use either the Plugins/Compile and Run command or the Plugins/Edit command. Both commands require that ImageJ be running on a Java Virtual Machine that includes the javac compiler, such as the JRE 1.1.8 distributed with the Windows and Linux X86 versions of ImageJ.

The javac compiler is also included with the Java Development Kits from Sun but the tools.jar file must be in the classpath before ImageJ can use it. This requires that you run ImageJ using a command something like

```
java -cp ij.jar;C:\jdk1.3\lib\tools.jar ij.ImageJ
```

Alternately, put a copy of tools.jar in the Java extensions folder, located at something like

```
C:\Program Files\JavaSoft\JRE\1.3\lib\ext
```

and run ImageJ by double clicking on ij.jar.

To compile plugins in the Macintosh you must download and install the MRJ Software Development Kit (SDK) from

<http://developer.apple.com/java/text/download.html#sdk>

TROUBLESHOOTING

Pipeline Client execution:

- Make sure there are no spaces in the filenames or directories leading up to the LONI Pipeline processing environment (Java doesn't like them).
- Make sure you have Java 1.3 or newer in your path.
- If you are running on an Apple, you must be using MAC OS X.
- There are problems with Java 1.4+ on SGI. Please use Java 1.3.X. for that platform only.

Pipeline Server connections:

- If you are behind many firewalls the server is not allowed to initiate communications, only respond to your queries. To ensure proper executions please enable 'callback=false' in your properties/pipeline.properties file.
- Some firewalls do not allow you to connect on port 80 with http-tunneling (our system default). Please try changing the rmi port in the properties/pipeline.properties file to 'rmi_port=1099'.

ImageJ Plugins and Macros:

- If ImageJ complains it cannot find the plugins or macros folders they need to be moved to the location ImageJ is being executed from. Normally the pipeline execution directory is the proper

place, but this may differ on various OSs and with various startup scripts. Move the plugins and macros directories from the Pipeline.jar directory to the ImageJ execution location to fix this problem.

Note: This location is usually specified in the error ImageJ gives upon its execution: 'Plugins folder not found at <location>' where <location> is where ImageJ is looking.

For other problems please contact help@loni.ucla.edu.

DEMONSTRATION PIPELINES INCLUDED WITH THE DISTRIBUTION:

Note: All the demo pipelines are to be run while connected to the LONI Pipeline Server: inire.loni.ucla.edu

All Demonstration Pipelines are located in: System Module List -> Demos. They are detailed below. Approximate running times are given. The pipelines get more complex and have longer running times as demos progress. Running time can also be influenced by the server load. If things are taking much longer than estimated, our servers are probably busy doing lots of brain mapping. Don't worry, your jobs are still queued to run and will get processed when resources become available. We also are going through an upgrade cycle to provide our collaborators with even more computing power through the LONI Pipeline Server that will allow analyses to run even faster!

The Pipelines demonstrated are deliberately simple, and only use 1, 2, 5, or, at most, 20 input MRI scans. Users are free to change the inputs and outputs to the demos and play with the environment with their own scans. Just remember that we have all the inputs set to be server side to save on transfer times during a demonstration. If your data is on your client, then make sure to toggle the 'Server File?' options off on the proper input tabs for each pipeline.

If you are having trouble, then feel free to email help@loni.ucla.edu for assistance. When mailing please make sure to give as much detail about your problems as possible.

DEMO PIPELINES:

RUNNING:

1. Log in to the server under your provided account: Server -> login
2. Select the desired pipeline by double-clicking on it, i.e.: System Module List: -> Demos -> 1 Tissue Volumes
3. Run it: Execution -> Run
 - i. If it runs on a dataset, instead of a single scan, you will be asked if you want to run in 'looped' mode. Say YES.

1. Tissue Volumes

Running Time: 1 minute

Description: This pipeline does a tissue classification of a single T1-weighted MRI of a brain into gray matter, white matter, and CSF. It displays the tissue classified brain and computes the volumes of each tissue class and the volume of the intracranial cavity. All the volumes are displayed in message windows.

Notes: This pipeline opens with a global variables table in the foreground. This table is often used to provide a user with access to important inputs and outputs and shield them from the inner workings of a pipeline. It can be used to house any file, dataset, or argument to be used in an analysis.

You can also double click on any of the modules you see. If they are a base module, you will see the command details in a dialog box. If the module is actually another pipeline, or pipelet, it will open a new workspace that shows you the details of that pipeline. This is the case with the 'Show Label Volume' modules seen. Try double-clicking on one and then running the pipeline. You can watch the progression of the execution through the sub-layers of the analysis as well.

An important note is that we are showing everything to the user with viewers. The environment is actually designed to save all desired results to disk. If the user specifies an output for any modulee (or a list of outputs as the case may be) they will be made permanent instead of using the temporary space they normally use. For lists, outputs can be generated through manipulations of input variables. See the instructions and the provided pipelines on the server (Modules@inire.loni.ucla.edu -> Pipelines -> . . .) for examples of this.

2. Multiple Tissue Volumes

Running Time: 2 minutes

Description: This pipeline performs the tissue classification and volume calculations on 5 brains simultaneously. Message windows display all the volumetric information when it is done.

Notes: Demonstrated here is the parallel processing that benefits the users of the LONI Pipeline Processing Environment. The time it takes to analyze 5 subjects is only slightly more than the time to analyze 1 subject. This stays true up till the resources of the computer are finally topped out. Currently we allow external users to see over 70 processors with more than 60Gb of RAM and over 250Gb of temporary disk space.

3. Intracranial Volume - Linear Spatial Normalization

Running Time: 6 minutes

Description: Demonstrated here is the difference between doing volumetrics on a subject's brain with or without spatial normalization. Head size can often be a confounding variable in a study. Sometimes it is dealt with by normalizing the MRI data to a standard, i.e. Talairach space, or in this case ICBM452 space (Rex et al., Neuroimage 2003 19(3):1033-48). This normalization allows differences in structure size or tissue volume to be elucidated with respect to the cranium it is in.

Notes: The results seen here were taken to be extreme on purpose. A small female and a large male were chosen to be normalized. We see intracranial volume grow for the female from 1229.6cc to 1649.5cc. For the male it shrinks from 1811.3cc to 1609.6cc. They went from a difference of nearly 600cc to only 40cc with an affine (12-parameter) linear registration to the atlas space. Non-linear registrations can remove even more variability, but they also can remove the variability of the substructures that are to be studied.

4. Linear Aligned Average

Running Time: 8 minutes

Description: Shown here is a 20 subjects linearly aligned, 12-parameter affine transformation, to ICBM452 space. Their intensity information is averaged in the new space. Probability maps of the tissue types also are calculated for gray matter, white matter, and CSF. They demonstrate how likely it is to find each of these tissue types at a given point in ICBM452 space when a similar subject is linearly registered to it. Four image viewers are launched with each tissue probability field and the overall average displayed.

Notes: Again, demonstrating the parallelism of the environment, the registration time on an unloaded server is nearly identical for the 2 subjects in demonstration 3, above, and the 20 subjects in this demonstration.

5. Hemispheric GM Volumes

Running Time: 13 minutes

Description: This pipeline registers a single subject to ICBM452 space. It then registers a template with a 5th order polynomial non-linear transformation to the subject. This template is used to cut out each hemisphere of the subject into a different region of interest. The gray matter volume is then computed for each hemisphere, cerebral and cerebellar. The right and left gray matter maps of the cerebral hemispheres are displayed in volume viewers. The resultant volume calculations are shown in message windows.

6. Make Atlas - Tissue Probabilities

Running Time: 1 hour 30 minutes

Description: This pipeline generates a new atlas space from the data provided to it. In this case we only use 20 volumes (normally we would use 50 or more, and in the case of ICBM452 space, we used 452 volumes). The atlas space is in a linearly 'least distant space' from each of the subjects provided to the pipeline. Each subject is aligned to one random subject from the group. The transforms are mathematically analyzed in the 'Define Common AIR' step to find the space that is linearly closest, on average, to each subject. All subjects are resampled into this space and an intensity average of the subjects is made. Another linear registration takes place of each subject to the average template. A second resampling of each subject is made to the average template, and a second average template is computed with another intensity average. Finally a non-linear registration is done for each subject to the linearly defined template. This step preserves the boundaries of the linearly defined space, but allows sub-structures to be better defined with less blurring across their edges. From this last step a tissue classification is made of each subject and probability fields for each tissue class are generated for reference in the new space.

Notes: Most of the time spent in this pipeline is on the non-linear registrations. They can take over an hour to complete.