

# CBS Tools for MIPAV & JIST

## User Guide



### Foreword

This guide describes some freely available software tools for MR image processing in neuroscience developed at the Max Planck Institute for Human Cognitive and Brain Sciences (MPI-CBS). The software, documentation and associated data are property of the MPI-CBS and governed by the license terms within the package. The algorithms included in the software package are described in the corresponding articles; this guide is aimed at helping you install and run the software in a standard way.

# 1. Background information

## ***What are MIPAV, JIST and the CBS Tools?***

**MIPAV** (Medical Image Processing, Analysis and Visualization) is a software platform for medical image analysis and visualization developed at the Center for Information Technology of the National Institutes of Health (NIH) by Dr. McAuliffe and his team.

(see <http://mipav.cit.nih.gov/> )

**JIST** (Java Image Science Toolkit) is a processing environment providing advanced pipeline capabilities as an extension of MIPAV for processing large amounts of medical imaging data and performing modular analyses developed principally at the MASI laboratory (Vanderbilt University) and IACL (Johns Hopkins University) by Profs. Landman and Prince and their teams.

(see <http://www.nitrc.org/projects/jist/> )

The **CBS Tools** is a collection of high-resolution computation modules for JIST developed at the Max Planck Institute for Human Cognitive and Brain Sciences by Dr. Bazin and his team. Because these three software packages work together, all three must be installed and configured to run the CBS Tools. In addition, you may need the TOADS-CRUISE module package as well as the ANTS software package in order to run some specific CBS Tools modules.

In this release, we offer a **bundle version** with a tested version of MIPAV, JIST, TOADS-CRUISE and ANTS compatible with the current CBS Tools (see <http://www.nitrc.org/projects/toads-cruise/> and <http://www.nitrc.org/projects/ants/> ) as well as a lighter **update version** with only the CBS Tools, in case you already have a compatible version of the other software installed. As all these software packages are being actively developed, they may become temporarily incompatible. Be careful of version numbers when installing or upgrading these packages manually. New releases are not systematically tested for compatibility.

## ***What tools and release versions are included in the CBS Tools bundle?***

The CBS Tools bundle currently includes CBS Tools 3.0, JIST-CRUISE 3.0 (25/03/2014), and DOTS 2A. **It requires** to use **MIPAV version 6.0.1** or above (warning: MIPAV 7.0.x is currently *not supported* on some linux platforms). **Optionally**, some modules use **ANTs version 1.9.x** (warning: ANTs wrappers have only been tested on linux so far, and older or newer versions of ANTs may require some additional customization). The ANTs package is not included in the CBS Tools bundle and should be installed independently.

## 2. Installation

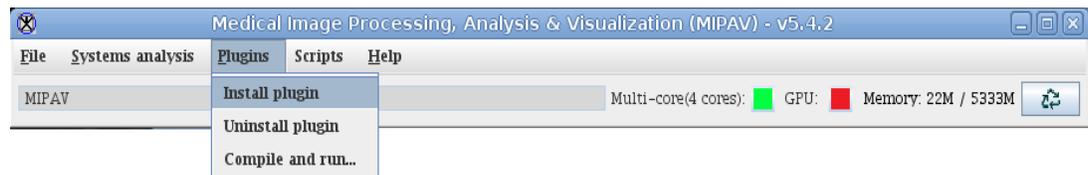
1. Download the required software packages:

to use the CBS Tools, you will need a copy of MIPAV, JIST and optionally some tools from the TOADS-CRUISE and ANTS packages. Other JIST module packages are also available that include many useful tools to help with general data processing, see the JIST website for a list. The **easiest** option to install all the software is to download the **CBS Tools bundle**, which includes a compiled version of all needed packages. Alternatively, you can find the latest version of these packages from the following web sites:

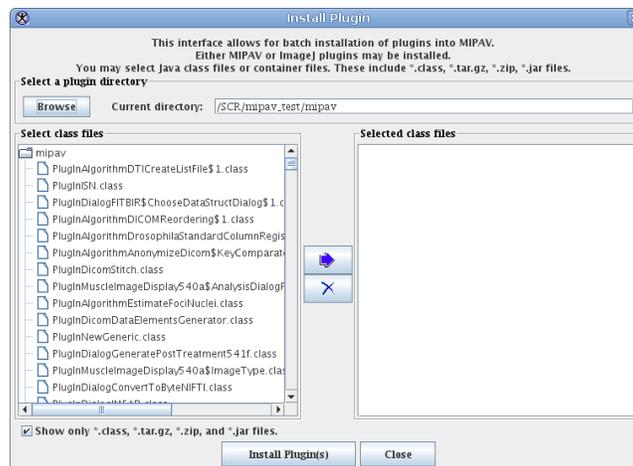
CBS Tools	<a href="http://www.cbs.mpg.de/institute/software/cbs-hrt/">http://www.cbs.mpg.de/institute/software/cbs-hrt/</a>
MIPAV	<a href="http://mipav.cit.nih.gov/">http://mipav.cit.nih.gov/</a>
JIST	<a href="http://www.nitrc.org/projects/jist/">http://www.nitrc.org/projects/jist/</a>
TOADS-CRUISE	<a href="http://www.nitrc.org/projects/toads-cruise/">http://www.nitrc.org/projects/toads-cruise/</a>
ANTS	<a href="http://www.nitrc.org/projects/ants/">http://www.nitrc.org/projects/ants/</a>

Although the latest version of each package includes the latest developments and bug fixes, they may also introduce compatibility conflicts from time to time.

2. Install MIPAV: run the MIPAV installer (see <http://mipav.cit.nih.gov/> for details if needed).
3. Run MIPAV. From the main MIPAV menu bar, select '*Plugins-> Install plugin*'.



The program may ask you to restart MIPAV, in which case you close MIPAV, restart, and repeat. You will get the '*Install Plugin*' window:



4. Next, you need to select the directory where you downloaded the packages for the CBS Tools, JIST and TOADS-CRUISE with the '*Browse*' button. The packages will appear in the '*Select class files*' list, and you need to select them and press the arrow to add them to the '*Selected class files*' list. Then you can press the '*Install Plugin(s)*' button. After a few seconds, you should get a message window listing all the installed plug-ins. You can now '*Close*' the '*Install Plugins*' window. If you look again under the '*Plugins*' menu bar, you will see several new sub-menus including JIST, TOADS-CRUISE and CBS Tools. The installation is complete!

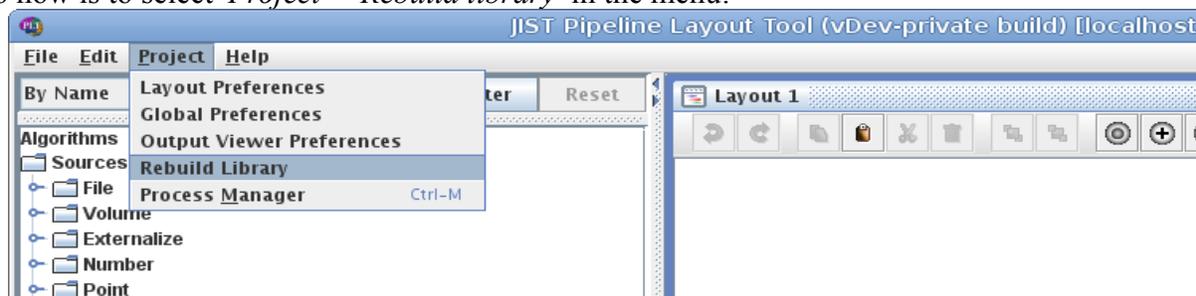
### 3. Working with JIST pipelines

#### **JIST Set-up**

There are a few additional steps in order to use the JIST pipeline environment. First, select *'Plugins->JIST->JISTLayoutTool'* in the main menu bar of MIPAV. For this first time, a browser will appear and request you to select which directory to use to store the JIST module definitions. We **recommend** that you create a new directory for this called *'modules'* inside the *'mipav'* directory of your home folder (to create a new folder directly from the browser, you can use the third button of the toolbar shown below).



When you select the new directory and press *'Open'*, the JIST layout tool will appear. The second thing to do now is to select *'Project->Rebuild library'* in the menu:



A dialog will ask if you wish to delete all existing module definitions, which you can accept by default (it ensures the module definitions are kept clean if you do multiple installations). Once the library has been rebuilt, you should see that the module list on the left now has many *'Algorithms'* modules, including the CBS Tools under *'CBS'*.

Finally, we **recommend** you to look at the *'Global Preferences'* (under *'Project'*) and select your preferred data format and compression (we suggest *'nii'* and *'gzip'*, which produces compressed NIFTI files). It is also recommended to keep the *'Debug level'* to 0 or 1 unless there are particular problems with some modules, as the debug procedures can slow down the computations.

#### **Building a JIST pipeline**

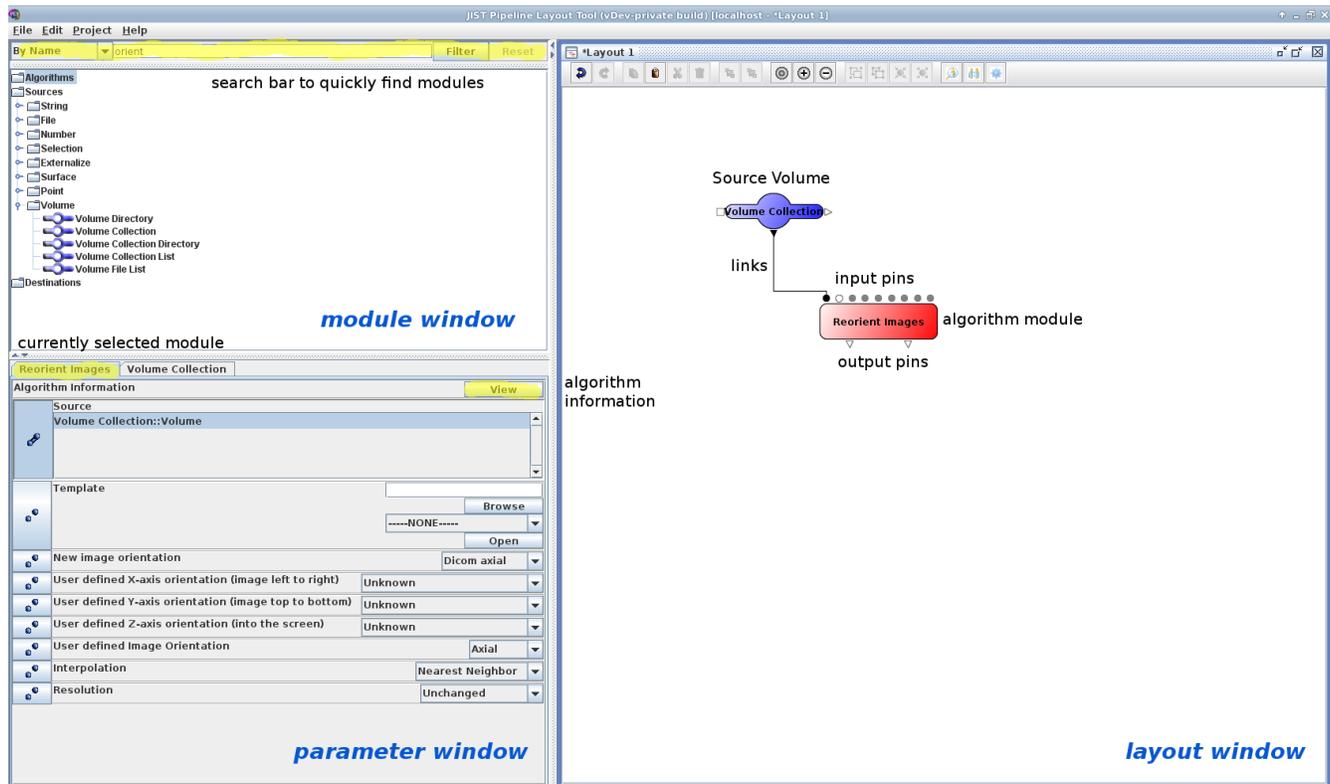
*JIST pipelines* define a computational experiment, specifying the input data, algorithms to be used and outputs to be generated in the form of modules linked in a visual diagram. They are built in the **JIST Layout Tool** and executed with the **JIST Process Manager**. First, open the JIST Layout Tool (in MIPAV, *'Plugins->JIST->JISTLayoutTool'*). The interface looks like this:

On the left side, there is a list of existing modules currently in the JIST library (Algorithms, Sources, Destinations, *'module window'*). Under it, a window that displays the list of parameters for a selected module placed on the layout window (*'parameter window'*). On the right, the *'layout window'* where the pipeline is created from the module library.

The JIST pipelines have two main components types: Sources or input modules (in blue) and algorithms modules (rectangular, generally in red). To create a simple pipeline, you first select a source

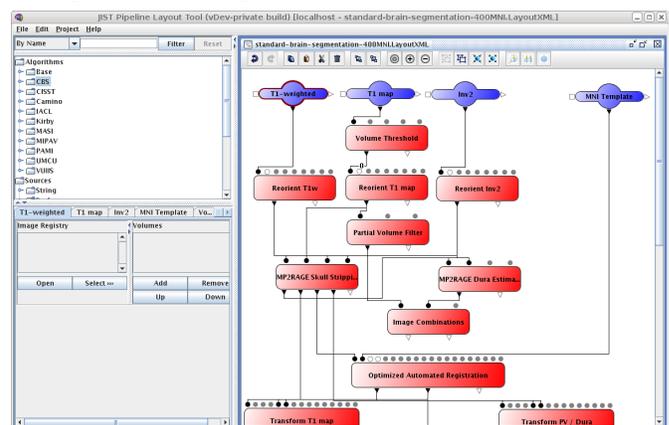
type corresponding to your data (generally Volume for volumetric MR image data) and one or several modules that perform the processing you want to do. You can drag and drop them into the layout window, and connect their pins to specify inputs and outputs of connected modules. Many algorithms have a short description available under the 'Algorithm information' button in the parameter window and also as hover text in the module window, and the name/description of the different pins appears when you click on them.

For instance:



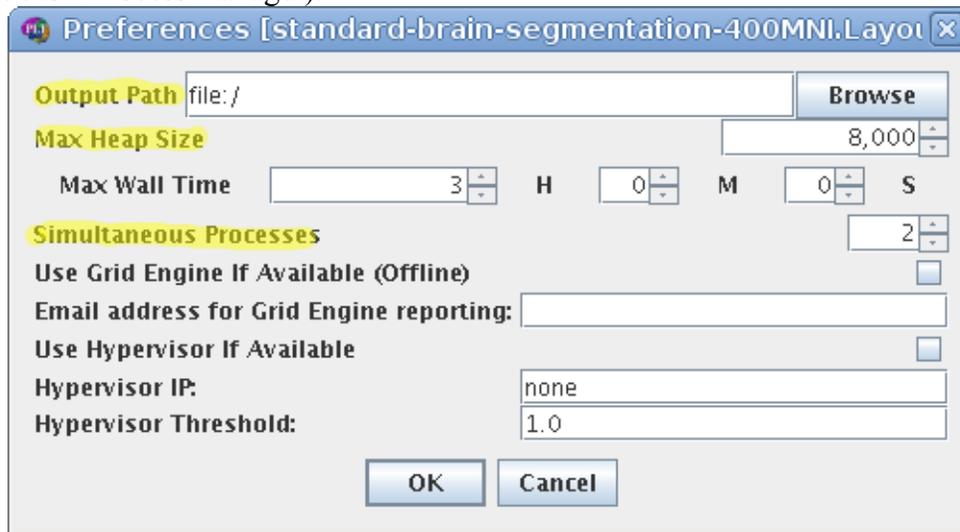
This layout will take the data specified in the *Volume Collection* and change its orientation. Highlighted in yellow are the search bar, to quickly find specific modules, the algorithm information button and the module tabs in the parameter window.

Of course, **pipelines are generally more complex**. To keep them manageable, there are three useful options in JIST: first, any module can be renamed in the layout to indicate its purpose rather than its underlying algorithm (double-click on its name to edit it). Second, you can add post-it notes anywhere in the layout to add important information about the experiment. Third, multiple modules can be turned into collapsible algorithm groups. More advanced options to organize modules and layouts are also available, see the JIST website under '*More information*' below for details.



## Setting up a JIST pipeline parameters

Important settings for running the layout are located in the Layout Preferences (both in the JIST Layout Tool and the JIST Process manager):



You need to choose a **destination directory** where all the intermediate output from the JIST pipeline is written. Because JIST generates output data for every step of the way, it is usually recommended to *create an empty directory* for that output (by default, the output directory is the one where the layout has been saved). Once the processing is done, one can simply copy the desired results out of the JIST processing tree and into a specific destination, either manually or with the Copy Data module.

It is also recommended to set there the maximum amount of memory to be used by a single module ('*Max Heap Size*', in MB) and the maximum number of processes to run in parallel ('*Simultaneous Processes*'). These numbers depend on the computer's hardware. For our high-resolution processing algorithms, large amounts of memory are required at times, and we recommend a max heap size of 15-32GB. Having multiple processes reduces computation times, but it can slow down or crash some processes if the total amount of memory used goes beyond the computer's memory size.

## Running a JIST pipeline

Once you have built a given JIST pipeline (either created manually as above or from our library of standard pipelines), all is left is to run the corresponding experiments and process the data. Open the **JIST Process Manager** (Ctrl-M inside the JIST Layout Tool, or from the MIPAV Plug-ins menu), and load the pipeline layout. A series of experiments will appear, one for each module and input data set, indexed by data sets as '*exp-0000*', '*exp-0001*', ... , and by module as '*-A*', '*-B*', '*-BA*', ... depending on modules inter-dependencies. Note that changing the connections between modules will affect this ordering and may require to re-run parts of the layout.

The screenshot shows the JIST Process Manager window with a table of experiments. The table has columns for Experiment, Module, Algorithm, Priority, Time (Actual / C...), Memory (Used / ...), Hypervisor Estim..., Status, and Progress. The status of each experiment is listed in the Status column.

Experiment	Module	Algorithm	Priority	Time (Actual / C...	Memory (Used / ...	Hypervisor Estim...	Status	Progress
0000-A	Volume Threshold	Threshold	1	- / -	- / -	NA / NA	READY	
0000-B	Reorient T1w	MipavReorient	2	- / -	- / -	NA / NA	READY	
0000-C	Reorient Inv2	MipavReorient	3	- / -	- / -	NA / NA	READY	
0000-AA	Reorient T1 map	MipavReorient	4	- / -	- / -	NA / NA	NOT READY	
0000-AAA	Partial Volume Filtr	JistBrainPartia	5	- / -	- / -	NA / NA	NOT READY	
0000-CA	MP2RAGE Skull Str	JistBrainMp2rag	6	- / -	- / -	NA / NA	NOT READY	
0000-CB	MP2RAGE Dura Es	JistBrainMp2rag	25	- / -	- / -	NA / NA	NOT READY	
0000-CAA	Optimized Autom	FLIRT	7	- / -	- / -	NA / NA	NOT READY	
0000-CAB	Image Combinatio	JistToolsImageC	9	- / -	- / -	NA / NA	NOT READY	
0000-CAC	Transform T1 ma	TransformVolume	12	- / -	- / -	NA / NA	NOT READY	
0000-CABA	Transform PV / D	TransformVolume	10	- / -	- / -	NA / NA	NOT READY	

At the bottom of the window, there is a status bar that reads: "Scheduler Stopped: Succeeded (0) Failed (0) Running (0) Queued (0) Total (25) Elapsed Time (Actual: - / CPU: -)".

For each data set and module, it displays the expected **inputs and outputs** (bottom window) and the processing status. To start a single process, select the corresponding line and press the 'run' button (multiple selections with the usual key combinations are possible). To start the entire pipeline, select instead 'Scheduler-> Start Scheduler'. Note that the JIST Layout Tool is not needed if you just want to run a pipeline from a saved layout. If the layout is modified, the process manager may be halted and need to be reloaded. Results that have been obtained with an earlier version of the layout are labeled as 'out of sync' and must be recomputed (with the 'restart' button) or overridden (right-click of the mouse on the selected experiment).

Results can be easily **inspected** by selecting a 'completed' process and opening the input or output images with the volume or surface rendering option (in the menu that appears at the right-click of the mouse). The images are then automatically loaded in MIPAV.

If problems arise, **information** about each module's activity and error messages are collected in the *debug window* (opened with the 'View Debugging Information' button). Inspecting the lists of inputs for the module can also help in case some input data is not accessible. The module may then take a 'failed' status, and the processing continues for other modules and data sets.

## Using JIST modules in scripts

In addition to the JIST pipeline interface, JIST modules can be run directly from the command line and integrated in scripts. This option is a bit technical, and described in detail in: <http://www.nitrc.org/plugins/mwiki/index.php/jist:CommandLineInterface>

## More information

There are more detailed tutorials and explanation on the JIST website: <http://www.nitrc.org/plugins/mwiki/index.php/jist:MainPage>

## 4. CBS Tools Overview

### **Standard layouts**

The **layouts included with the CBS Tools** (under `mipav/plugins/layouts-r3.0/`) follow the principles described above. It is important to keep in mind the **memory requirements** of the main computational modules, which will also vary depending on the processing resolution. A computer with about 8GB of memory should be able to handle data processed at up to 0.8mm resolution, while a 0.4 resolution may require 24GB or more. Computation times are also variable, between a few minutes and a few hours (the average computation time for the *'MGDM Multi-contrast Brain Segmentation'* module varies from under 1h to about 6h for 0.8mm and 0.4mm resolutions, respectively). The layouts are all annotated to explain the main steps and describe where to find specific input **templates and atlases**. These are generally under `mipav/plugins/atlases/` and `mipav/plugins/Atlas/`, where are located data and templates from the CBS Tools and JIST-CRUISE respectively.

### **Main algorithms**

**Brain segmentation:** the CBS Tools include a multi-contrast algorithm for segmenting the brain into its main sub-structures. The method has been optimized for high resolution data and can handle routinely resolutions of up to 400 $\mu$ m on a computer with enough memory. It accepts a large number of MR contrasts and can be extended easily to new data types. See (Bazin et al., 2013) in References for details. *Modules: MGDM Multi-contrast Brain Segmentation*

**Cortical surface processing:** cortical surface extraction, inflation and mapping is provided through tools adapted from the TOADS-CRUISE module package, see (Han et al., 2004) and (Tosun et al., 2007) for details. These tools work well at high resolution with meshes of up to a million vertices. In addition, many tools for handling data on cortical surfaces are provided. *Modules: Full CRUISE Cortex Extraction, Surface Mesh Inflation, Surface Mesh Mapping, etc.*

**Laminar modeling:** with the CBS Tools, it is also possible to estimate cortical depth and geometric cortical layers that follow the volume-preserving bending measured in classical histology, see (Waehnert et al., 2013) for details. The lamination comes with utilities to map intensity values on each separate layer, measure geometric properties of the depth profiles and estimate averages in ROIs. *Modules: Volumetric Layering, Profile Sampling, Profile Geometry, etc.*

**Cortical surface registration:** the CBS Tools include a multimodal, multiresolution surface registration algorithm (MMSR) which allows to align cortical surfaces based on geometry and structural contrasts such as quantitative T1, see (Tardif et al., 2013) for details. The registration is based on SyN (and so requires ANTs to be installed) and provides symmetric diffeomorphic mappings in 3D space. In addition, many utilities are provided to deform images and transfer surface data from mesh to volumetric space and back. *Modules: Multimodal Surface Registration, Apply Deformations, Level Set To Mesh, Mesh Data To Volume, etc.*

**Cerebellum processing:** cortical surface extraction, inflation and mapping is possible for the cerebellum as well as the cerebrum, using the tools described above in 'Cortical surface processing'.  
*Modules: Full CRUISE Cortex Extraction, Surface Mesh Inflation, Surface Mesh Mapping, etc.*

**Contrast-specific MR tools:** the CBS Tools include several dedicated tools and filters for the MP2RAGE sequence, such as background masking, skull-stripping, arteries filtering, dura estimation.  
*Modules: MP2RAGE Background Masking, MP2RAGE Skull Stripping, MP2RAGE Arteries Filter, MP2RAGE Dura Estimation, etc.*

## **Data representations**

Images handled by the CBS Tools are 3D or 4D data volumes (sometimes called 'Image Volumes' or 'Volumes' in JIST) in any of the medical image formats supported by MIPAV. As the software does not expect a particular format, it will not take into account embedded transformations to treat the images (e.g. Nifti image space), but instead work on the voxel array, unless specified. Re-orientation and resampling of data coming from multiple sources may be needed.

Many algorithms in the CBS Tools manipulate 3D surfaces, which are often represented as level set functions in volumetric space. These functions represent the signed distance to the surface for each voxel, and can be turned into 3D triangular meshes easily through a marching cubes algorithm. The two representations (level sets and meshes) are roughly equivalent, but cannot be used for one another.

## **Other recommended modules from MIPAV, JIST and CRUISE**

There are many additional algorithms and utilities developed by the other laboratories who adopted the JIST and MIPAV environment. We recommend below a short selection of modules we have used from the 'IACL' plug-in collection. There are many others that may be of interest depending on your data and application, of course.

- SPECTRE: a very conservative and robust skull stripping algorithm,
- Optimized Automated Registration (OAR): MIPAV's own version of the FLIRT algorithm,
- Transform Volume: the corresponding module to transform images,
- Reorient Volume: a simple utility to set all your data to a chosen orientation,
- N3 Correction: a port of Sled et al.'s inhomogeneity correction algorithm,
- FANTASM: a fast and accurate method to segment brain tissues with many options,
- Topology Correction: a method to generate spherical topology surfaces from level sets and segmentations,
- STAPLE: a port and several extensions of Warfield et al.'s algorithm for label fusion,
- DOTS: a voxel-based method for automatic labeling of WM tracts in DTI (under 'IACL').

## ***Encapsulation of external modules***

Finally, there is also a simple interface to encapsulate command-line tools as JIST modules with the *JIST External Script Adapter* in 'Base'. A detailed description of the procedure is given here:

<https://www.nitrc.org/plugins/mwiki/index.php/jist:ExecutableWrapper>

## 5. Troubleshooting

Many problems can occur when applying a complex processing pipeline to a new set of data; and MIPAV, JIST and the CBS Tools are no exceptions. Upon encountering a problem, we recommend to check the following things first:

1. Find **which module** in your pipeline is the first to give erroneous results or to fail. This is likely the source of all other errors.
2. Check the module's **input data** for correctness, as the module before (or the original input) may be the source of error.
3. Read the module's **debugging information**, which may include error messages and/or details of the algorithm's behavior.
4. If everything seems normal, check for unusual causes, e.g. whether the **data output path** is correctly specified and writable in the Layout Preferences or if your hard drive is full.

To perform these checks, the best tool is the **Process Manager**: when selecting a given module you can see the values of its input parameters ('Ancestors' or 'Descendants' window). With a right-click, a menu appears that allow you open input and output images ('Volume Rendering') as well as the 'View Debugging Info' window. Finally, both layout-specific and global preferences are accessible under the 'File' menu.

If all fails or appears to be clearly a bug, please send the following information along with the bug report: version of CBS Tools, MIPAV and JIST-CRUISE, operating system, amount of memory given to the software, dimensions and resolutions of the data. A copy of the output and error messages generated by JIST (in the Debugging Information window) and/or the terminal running MIPAV are usually very helpful as well. Note also that the CBS development team may not be able to solve problems related to MIPAV or JIST, but not to the CBS Tools.

You may then submit this information to:

<http://www.nitrc.org/projects/cbs-tools/> under the 'Tracker' section.

## 6. References

### *Brain segmentation*

Bazin, P.-L.; Weiss, M.; Dinse, J.; Schäfer, A.; Trampel, R. & Turner, R.  
A computational framework for ultra-high resolution cortical segmentation at 7 Tesla  
*NeuroImage*, **2013**.

### *Cortical surface processing*

Han, X.; Pham, D.; Tosun, D.; Rettmann, M.; Xu, C. & Prince, J.  
CRUISE: Cortical Reconstruction Using Implicit Surface Evolution  
*NeuroImage*, **2004**, 23, 997-1012

Tosun, D.; Rettmann, M. & Prince, J.  
Mapping techniques for aligning sulci across multiple brains  
*Medical Image Analysis*, **2004**, 8, 295-309

### *Laminar modeling*

Wahnert, M.; Dinse, J.; Weiss, M.; Streicher, M.; Wahnert, P.; Geyer, S.; Turner, R. & Bazin, P.-L.  
Anatomically motivated modelling of cortical laminae  
*NeuroImage*, **2013**.

### *Multi-modal Surface Registration*

Tardif, C.L.; Dinse, J.; Schäfer, A.; Turner, R.; & Bazin, P.-L.  
Multi-modal surface-based alignment of cortical areas using intra-cortical T1 contrast  
*Proceedings of the MBIA workshop*, **2013**, 226-37

### *Multi-object Geometric Deformable Models (MGDM)*

Bogovic, J.; Prince, J. & Bazin, P.-L.  
A Multiple Object Geometric Deformable Model for Image Segmentation  
*Computer Vision and Image Understanding*, **2013**, 117, 145-57

### *MIPAV*

McAuliffe, M.; Lalonde, F.; McGarry, D.; Gandler, W.; Csaky, K. & Trus, B.  
Medical Image Processing, Analysis and Visualization in Clinical Research.  
*Proceedings of the 14th IEEE Symposium on Computer-Based Medical Systems (CBMS 2001)*, **2001**.

### *JIST*

Lucas, B.; Bogovic, J.; Carass, A.; Bazin, P.-L.; Prince, J.; Pham, D. & Landman, B.  
The Java Image Science Toolkit (JIST) for Rapid Prototyping and Publishing of Neuroimaging  
Software. *Neuroinformatics*, **2010**, 8, 5-17

### *JIST-CRUISE overview*

Landman, B.; Bogovic, J.; Carass, A.; Chen, M.; Roy, S.; Shiee, N.; Yang, Z.; Kishore, B.; Pham, D.;  
Bazin, P.-L.; Resnick, S. & Prince, J.  
System for integrated neuroimaging analysis and processing of structure.  
*Neuroinformatics*. **2013** Jan;11(1):91-103.