The PipeDream Neuroimaging Pipeline for Cross-Sectional and Longitudinal Studies of Cortex and Connectivity

Release 0.1

P.Cook, B. Avants, N. Tustison, J. Duda, J. Gee

May 22, 2012

Penn Image Computing And Science Laboratory
University of Pennsylvania

Abstract

PipeDream uses open-source software to help neuroimaging researchers perform advanced processing with state-of-the-art techniques. PipeDream performs data reconstruction, data organization, advanced brain extraction, bias correction, three tissue classification and cortical thickness estimation. Diffusion tensor processing, unbiased longitudinal analysis and cortical parcellation are available as advanced options. PipeDream relies on prior knowledge encoded in an optimal template. PipeDream also helps one review the processing and perform quality assurance. The tool documents our current "best practice", implementable with free software, for MRI-based studies of the brain.

Contents

Introduction

For the latest updates to the code and this document, please see the PipeDream online repository at https://github.com/cookpa/pipedream.

This document covers the installation of PipeDream and related software, and the use of the PipeDream tools to process an example data set.

1 Documentation and getting help

This document is designed to help you get up and running using PipeDream. You can get additional help on the command line by running any command without any arguments. If you need further assistance or have any feedback, contact us at mailto:cookpa@mail.med.upenn.edu.

2 Installation

PipeDream assumes the basic unix tools that exist on GNU / Linux. On most Linux installations, everything you need to carry out installation will be available by default.

The instructions below explain how to obtain and build the software we use in PipeDream. Some of these tools are not affiliated in any way with PipeDream or the PICSL lab. They come with their own licensing agreements, which can be found at the web pages for each tool given below. Since PipeDream is an open-source pipeline, we've chosen to integrate only tools that are freely available (and preferably open source) for academic use.

2.1 Prerequisites on Mac OS X

PipeDream can run on a Mac OS X system, however you may need to install some extra software. Install Xcode, which provides the necessary compilers, git, and other useful tools. Next, we recommend the Homebrew package manager http://mxcl.github.com/homebrew/.

Mac OS X is based on BSD Unix, so there may be some differences in the implementation of unix tools. As of 10.7 (Lion), there are no known incompatibilities with PipeDream. However, you may optionally install the GNU Core Utils, which you can do with Homebrew:

```
brew install coreutils
```

This will install the GNU tools, however they will be prefixed with a g, so "ls" will be called "gls" and so on. You can create symlinks in `/usr/local/bin` to get around this, such that "ls" etc point to the GNU tools.

2.2 CMake

CMake is required for compiling ITK, ANTS, and GDCM, among other things. It can be downloaded from its website http://cmake.org/. Hombrew users can install it by typing `brew install cmake`.

CMake is a very powerful tool, and will do lots of configuration behind the scenes to enable code compilation on your machine. We will just need to know how to use the `ccmake` program. The options presented to you in CMake may differ slightly across platforms. For example, on Mac you will be asked which version of OS X to target, but overall the build process is very similar. This is the beauty of CMake, under the hood building on different systems is very different, but with CMake almost all of the necessary work is done for you.

2.3 ITK

The first thing you'll need to compile is ITK. At the time of writing, both ITK and ANTS are in a development phase, so you'll want to use the latest stable ITK. Clone the repository with git and checkout the nightly-master branch.

```
mkdir ~/src
cd ~/src
git clone http://itk.org/ITK.git
git checkout -b my_nightly_master origin/nightly-master
```

The above only needs to be done once, to update the source later you can update by pulling the remote nightly-master:

```
git pull origin nightly-master
```

More information about git can be found in the free online git book `http://git-scm.com/book`.

Once the source code is checked out, we will proceed to build. We always build outside the source tree.

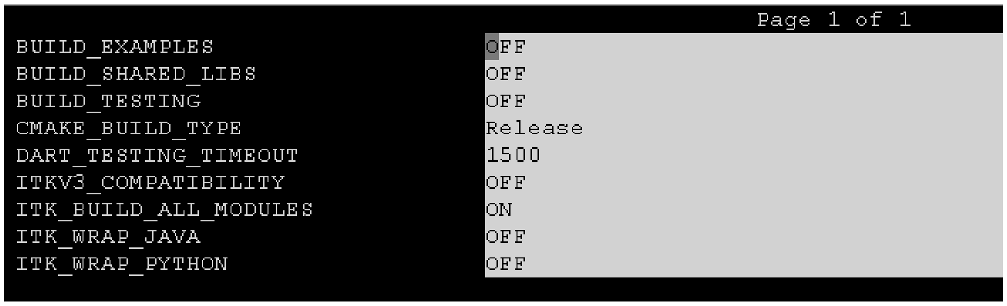
```
mkdir -p ~/bin/itk
cd ~/bin/itk
ccmake ~/src/ITK
```

This will bring up the `ccmake` interface. Initially, you will see the message `EMPTY CACHE`, meaning CMake hasn't done any configuration yet. Hit C to do an initial configuration. CMake will choose as many settings as it can automatically, and present others to you for review. Navigate to the options and hit enter to toggle boolean options or to enter text where needed. Turn off building examples and testing to save compilation time. Make sure the `CMAKE_BUILD_TYPE` says "Release" - this is important for ANTS too, since debug code is substantially slower. Also make sure that `ITK_BUILD_ALL_MODULES` is on. Hit C again to reconfigure. You should now have a CMake screen that looks like figure **??**. Hit G to generate the make files and exit.

Back at the terminal, you can now start compilation with `make`.

2.4 ANTS

ANTS will be checked out and built in a similar way to ITK, except that we use Subversion instead of git (in the future, ANTS will move to git).



```
                                                  Page 1 of 1
BUILD_EXAMPLES                  OFF
BUILD_SHARED_LIBS               OFF
BUILD_TESTING                   OFF
CMAKE_BUILD_TYPE                Release
DART_TESTING_TIMEOUT            1500
ITKV3_COMPATIBILITY             OFF
ITK_BUILD_ALL_MODULES           ON
ITK_WRAP_JAVA                   OFF
ITK_WRAP_PYTHON                 OFF
```

Figure 1: ITK configured in `ccmake`.

```
cd ~/src
svn checkout https://advants.svn.sourceforge.net/svnroot/advants/trunk ants
mkdir ~/bin/ants
cd ~/bin/ants
ccmake ~/src/ants/Examples
```

As before, set `BUILD_TESTING` off and ensure that the build type is set to “Release”. Reconfigure until everything is set, then generate the makefiles and make.

After building ANTS, it’s a good idea to record the version information of ANTS and ITK. This will help the developers reproduce any bugs or performance issues you encounter. Record the source code version information with `git` and `svn`.

```
cd ~/src/ITK
git show > ~/bin/ants/version_info.txt
svn info ~/src/ants >> ~/bin/ants/version_info.txt
```

2.5 GDCM

GDCM is available through Sourceforge, which means you can get it using Git or by downloading a tarball from its website `http://sourceforge.net/projects/gdcm/`. It builds with CMake. Building should be straightforward following the procedure above. Set `GDCM_BUILD_APPLICATIONS` on, PipeDream needs these applications to parse DICOM headers.

2.6 MRIcron

MRIcron is distributed in binary form from its web page `http://www.nitrc.org/projects/mricron`. The package contains several tools, including the `mricron` image viewer and the `npm` statistical tool. In PipeDream we use the `dcm2nii` program that converts DICOM series to NIFTI images.

2.7 Camino

Camino is a diffusion MRI toolkit. It can be downloaded from http://camino.org.uk. It builds directly from the package, assuming you have Oracle Java installed.

```
cd ~/bin/
tar -xzvf /path/to/camino.tgz
cd camino
make
```

2.8 Other tools

These tools are not required by PipeDream but may be useful to many users.

ITK-SNAP

ITK-SNAP is useful for image viewing and segmentation, and is avaliable from http://www.itksnap.org.

convert3d

The `c3d` command wraps many of the image processing filters of ITK in a command-line tool. It can handle complex operations in memory using an operations stack. See http://www.itksnap.org/pmwiki/pmwiki.php?n=Convert3D.Convert3D.

2.9 PipeDream

```
cd ~/bin
git clone git://github.com/cookpa/pipedream.git
cd pipedream
git checkout -b mybranch origin/master
cp config/pipedream_config_example.sh config/pipedream_config.sh
```

Edit the `config/pipedream_config.sh` file. Most of the variables are paths to the other software mentioned above. You should also review the `qsub` command that will be used to submit jobs to the queue.

Add the `pipedream/bin` directory to your PATH.

```
export PATH=/path/to/pipedream/bin:$PATH
```

You can add this line to your `.bash_profile` to give you access to PipeDream at login.

3 Running the pipelines

The pipedream neuroimaging pipeline has five stages:

1. Organize data
2. Pre-process individual modality images
3. Construct a template.
4. Run the brain processing pipeline.
5. Check results.
6. Perform statistical analyses.

We cover each step in turn in the following sections. [1]

4 Data Organization and study design

PipeDream is designed to work with DICOM data, and contains tools for properly organizing DICOM files and converting them to NIFTI-1 format. If you do not have access to DICOM data, you can replicate the data organization and pick up the pipeline at the NIFTI stage, see **??**.

In all processing, images are identified by a unique subject identifier and a scan date. Subject identifiers are chosen by the user. They should be alphanumeric and not include special characters that may cause problems for scripts. Alphanumeric includes all letters and numbers, plus the underscore character _. Other characters have special meaning in various software packages and may cause problems.

The choice of subject identifiers should be consistent for imaging and non-imaging data that will be part of your study.

4.1 DICOM data organization

PipeDream organizes DICOM data from a given subject by date and series number. It does not sort subjects by subject identifier, because the correct choice of identifier is often not present the header. The user needs to choose the subject identifier at run time. The rest of the organization can be done by `dicom2series`.

In the absence of public data in DICOM format, we will use a hypothetical example data set. We have a base directory `$RAW_DICOM_DIR` that contains subjects numbered 001 through 100. Within `$RAW_DICOM_DIR/001`, the DICOM data for subject 001 is stored in some unknown format, perhaps copied from a CD written at the scanner. We can process all the data with the following script.

```
for subj in `seq -w 1 100`; do
  find -L ${RAW_DICOM_DIR}/$subj -type f -print0 | xargs -0 dicom2series \
    ~/subjectsDICOM/$subj 1 0
done
```

[1]This document is a work in progress. Please check for updates with each release.

Here we use the `find` utility to list all files within each subject's data directory. The use of `-print0` and `xargs` make the command robust to large numbers of files, and spaces in the file or directory names. Non-DICOM files will be skipped automatically.

DICOM index files can sometimes confuse `dicom2series`. These are usually files called `DICOMDIR`. If they are present you can remove them from the above command by adding `-not -name DICOMDIR` to the `find` command above.

The output data will be organized hierarchically by subject, acquisition time, and then by series. The subject identifier is completely under the user's control and is specified on the command line. The acquisition time is determined from the data and is written in the format YYYY_MM_DD_HHMM where the first three items are the study date, and the final "HHMM" is hour and minute of the study time. The series are named by series number, protocol name, and series description, in the format NUMBER_PROTOCOL_DESCRIPTION. The output DESCRIPTION is the concatenation of the protocol name and series description in the data, minus any duplication. Thus if your protocol name is "DTI30DIR" and your series description is "DTI30DIR_ADC" the output description will be "DTI30DIR_ADC" not "DTI30DIR_DTI30DIR_ADC".

5 DICOM to NIFTI conversion

We recommend keeping the NIFTI data in a separate directory structure. The first thing we do is define a list of protocols that should be converted. That is, the name of protocols as they appear in the `subjectsDICOM` directory, minus the series numbers. These should be written to a text file, let's say `all_protocols.txt`, one per line, for example

```
t1_mpr_AX_MPRAGE
t2_tse_obl_448_2mm
DTI_30dir_noDiCo_vox2_1000
DTI_34dir
```

Next, list the subjects that need processing. Usually this will be every subject listed in `subjectsDICOM/`

```
 ls subjectsDICOM > subjects.txt
```

Now we're ready to run

```
 dicom2nii sge subjects.txt  all_protocols.txt subjectsDICOM subjects
```

The output is organized in `subjects/subjectID/time/rawNii`. For example, `subjects/104391/2007_07_10_1024/rawNii`, which contains:

```
104391_2007_07_10_1024_0006_t1_mpr_AX_MPRAGE.nii.gz
104391_2007_07_10_1024_0011_DTI_30dir_noDiCo_vox2_1000.bval
104391_2007_07_10_1024_0011_DTI_30dir_noDiCo_vox2_1000.bvec
104391_2007_07_10_1024_0011_DTI_30dir_noDiCo_vox2_1000.nii.gz
104391_2007_07_10_1024_0013_DTI_30dir_noDiCo_vox2_1000.bval
104391_2007_07_10_1024_0013_DTI_30dir_noDiCo_vox2_1000.bvec
104391_2007_07_10_1024_0013_DTI_30dir_noDiCo_vox2_1000.nii.gz
```

```
104391_2007_07_10_1024_0015_DTI_30dir_noDiCo_vox2_1000.bval
104391_2007_07_10_1024_0015_DTI_30dir_noDiCo_vox2_1000.bvec
104391_2007_07_10_1024_0015_DTI_30dir_noDiCo_vox2_1000.nii.gz
```

The protocol "t1_mpr_AX_MPRAGE" is a T1 protocol, so the output is a 3D NIFTI image. The protocol "DTI_30dir_noDiCo_vox2_1000" is a diffusion protocol, so the output is 4D, and there are two additional files that describe the diffusion gradient vectors.

If your raw data is already in NIFTI format, you can begin processing with PipeDream at this point by copying your data into the structure above. The series numbers can be assigned arbitrarily. The acquisition date can be specified in a format of your choice, we recommend "YYYY_MM_DD", because then the text sorts in the correct temporal order. If you don't have DICOM data you probably won't know study time, but that's only a problem if you have multiple same-day imaging sessions. If that's the case, then you should add "_0001", "_0002", etc in place of the study time.

In the following sections, we discuss the various programs that do modality-specific pre-processing for the different images that PipeDream supports.

6 Modality specific preprocessing - scalar images

Scalar images, for example a T1 MRI, do not require any specialized pre-processing, however it is often convenient to rename them into a common format.

7 Modality specific preprocessing - diffusion images

The basic preprocessing steps for diffusion images are

1. Affine normalization of DWI images to the first b=0 image, for motion and distortion correction.
2. Registration of the average DWI image to a template, to define the brain mask.
3. Reconstruction of the diffusion tensor by weighted linear regression.
4. Computation of FA, MD, RGB images.

A full list of the output generated by nii2dt is given in table **??**.

8 Construct a Template

Pipedream requires that you have a template—preferably constructed from a dataset similar to your current study—with a number of companion images that contain anatomical labels. The minimum set of priors include: a brain or cerebrum mask, a cerebrospinal fluid probability map, a gray matter probability map and white matter probability map.

If you do not have a template already available—or would like a local/optimal template—then you can build one from your data by following these steps:

Table 1: default

Output	Description
allscans.scheme	The scheme file for the complete acquisition, including repeats if any.
averagedwi.nii.gz	The average of all images with $b > 0$.
brainmask.nii.gz	The brain / background binary mask, computed by registering the average DWI to a template.
dt.nii.gz	The diffusion tensors in NIfTI format.
dwi	Directory containing corrected raw DWI images
exitcode.nii.gz	Contains brain / background mask and flags for bad data.
lns0.nii.gz	Reconstructed estimate of the log of the b=0 signal.
sigmaSq.nii.gz	Estimate of the noise variance in each voxel
fa.nii.gz	Fractional anisotropy
md.nii.gz	Mean diffusivity. Units are dependent on the input b-values, which are usually $\mathrm{s/mm}^2$.

Table 2: Output of nii2dt

1. Create a template directory where you can write a bunch of data.

2. Link your anatomical (T1-MRI) images into the directory, using "ln -s" in unix/linux/osx.

3. Call the ANTS command AverageImages to get an initial template or copy one of your subjects to a file called DIFFtemplate.nii.gz .... Make sure you don't copy a file called image.nii to a nii.gz file-type. Either way, you will get an unbiased template out of this process.

4. Call the ANTS script : sh buildtemplateparallel.sh in this directory. An example call is here:

   ```
   sh buildtemplateparallel.sh -d 3 -o DIFF -c 2 -j 2 -z DIFFtemplate.nii.gz  *T1.nii.gz
   ```

   In this example, we assume you have images of the type Sub001T1.nii.gz . Note that the buildtemplateparallel script is a complex algorithm in itself and takes time. If you can get a pre-built template, then that would be preferable.

5. Extract the brain from the output template (using, for instance, www.itksnap.org) and segment it into three tissue classes. The ANTS Atropos tool will allow you to do this. However, this should be done well in that it will impact your study outcome. Keep the four label/probability images you generated for use as priors in your study. See the file pipedreamsegment3classnowarp.sh in pipedream to see how one might achieve a good segmentation of an image in a stand-alone module.

It takes some effort to produce a good template. So, check with developers to see if one is available for your type of data.

9 Run a Cross-Sectional Study

To run a pipedream study, your data needs to be organized as above and you need a template with all the necessary prior information available. When all is ready, you will call:

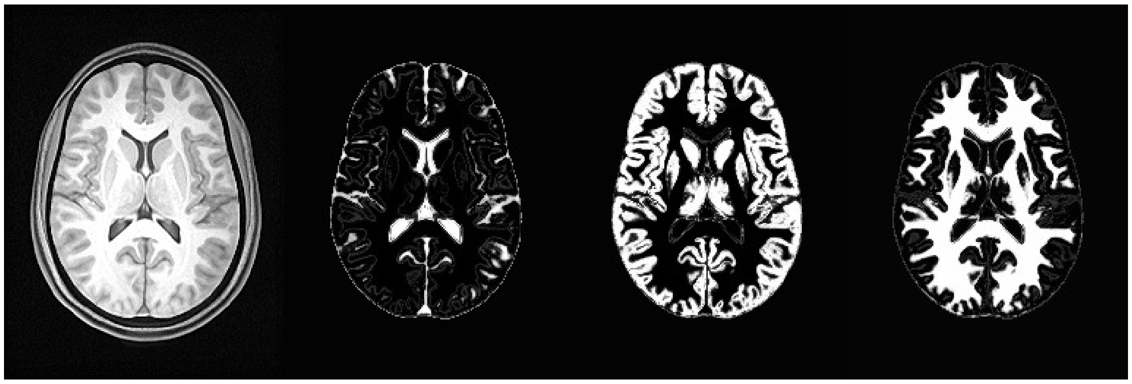


Figure 2: An example template.

```
    sh $PIPEDREAMDIR/pipedreamMapT1.sh pipedreamMapT1paramsUser.sh
```

where you likely have copied the "base" pipedreamMapT1paramsUser.sh to your local directory and edited it for your data. The output directory structure and naming will mirror your input structure. You will get cortical thickness images, brain extraction, probability maps, estimates of brain volume and—if enabled—a cortical parcellation.

The key to the study is the file pipedreamMapT1paramsUser.sh. Within that file is a series of directory and then file variables that you need to define. We've tried to "check" your definitions for you and provide a warning that will help you find a mistake but this may not always work. So, take care when making these definitions.

An important part of this step is how you distribute these processes. We provide hooks for voxbo and for the SGE qsub. There is also a PIPEDREAMISTEST variable in the user configuration file that will set "fast" parameters to allow you to check your configuration before running the full dataset. Set variable GOTOVBQ=1, GOTOSGEQ=0 (for voxbo) GOTOVBQ=0, GOTOSGEQ=1 (for sge) in pipedreamMapT1paramsUser.sh. If both are zero, you will run in serial mode.

10 Check Results

Pipedream output is organized by subject and time point in a similar way to the input data, except that structural and DTI data are placed in the same directory. Within each time point output directory, there are the following files, prefixed with the subject ID and time point ID.

Images suffixed with "norm.nii.gz" are normalized to standard space. If a mapping from the local template to standard space is provided, then all "norm.nii.gz" are in standard space. Otherwise they are in the local template space. The images relating to DT processing are only produced if you have DT data for the subject and time point.

If you have defined label images in the template space, there will be additional output containing the labels and probabilities in the subject space.

It's critical to evaluate the output data visually before doing statistics. The normalized images can be evaluated by extracting the same slice from all subjects, using ANTS:

```
    $ANTSPATH/StackSlices vol.nii.gz -1 -1 80  PipeDreamOutDir/*/*/*_thickness.nii.gz
```

Table 3: default

Output	**Description**
Affine.txt, InverseWarp.nii.gz, Warp.nii.gz	Warps from subject T1 to local template space.
brain.nii.gz	The brain-extracted and bias-corrected T1 image in subject space.
brainmask.nii.gz	The brain mask of the T1 image, used to compute brain.nii.gz.
brain_prob_[0,1,2].nii.gz	Segmentation probabilities of CSF (0), gray matter (1) and white matter in the T1 image.
deformed.nii.gz	Input head image deformed to the local template space. Useful for evaluating brain extraction and registration.
distcorr[Affine.txt, Warp.nii.gz, InverseWarp.nii.gz]	Distortion correction between T1 and DT space.
DTdeformed.nii.gz	DT image warped to the local template space
fa.nii.gz	Fractional anisotropy from DTI, in the subject space.
fadeformed.nii.gz	FA deformed into the local template space.
gmpnorm.nii.gz	Normalized gray matter probability (brain_prob_1.nii.gz).
grid.nii.gz	Deformation grid showing the warp of the T1 image to template space. Used for evaluating registration.
head.nii.gz	Input T1 image.
kappa.nii.gz	White matter curvature.
kappanorm.nii.gz	kappa transformed to standard space.
logjacobian.nii.gz	Log determinant of the Jacobian, in the local template space.
seg.nii.gz	Three tissue segmentation of the T1 brain image.
thickness.nii.gz	Cortical thickness in the subject space.
thicknorm.nii.gz	Thickness warped to standard space.

Table 4: Output of pipedreamMapT1

This will extract the 80th z-slice of every image and put it into a volume (vol.nii.gz) that you can scroll through. You can call this on any image type. If some specific dataset, e.g. subject00X, does not work, you may want to manually set the variable PIPEDREAMSUBJECTSLIST=subject00X and run a test on that subject alone, without sending to distributed computing, to see what is going on. As above, if you set PIPEDREAMISTEST=1 you can get a fast test result for this subject.

11 Compute Statistics

If everything is good, you are ready to estimate group statistics. We prefer a **R** statistical interface and can provide a script for you. However, you may run your favorite statistical package on the output of pipedream.

12 Annotated Bibliography

The original statement of the symmetric normalization and template construction methodology was given in [**?**]. A follow up study that used landmark guidance to compare the chimpanzee cortex to the human cortex was published here [**?**] – this study used *in vivo* MRI and template-based normalization to confirm volumetric numbers derived from an early 20th century post-mortem study comparing one human and one chimp. This conference article has some additional detail and alternative updates to the methodology, in particular application to shape-based interpolation [**?**]. Network based studies were performed here [**?**, **?**]. The main SyN paper is here [**?**]. Applications to neurodegeneration are here [**?**, **?**, **?**, **?**, **?**, **?**, **?**]. Hippocampus focused work is here [**?**, **?**]. The main evaluation papers include [**?**] and [**?**] for the cortex and deep brain structures whereas [**?**] evaluates the use of automated and semi-automated normalization for high-throughput hippocampus morphometry. An additional evaluation paper is being developed.