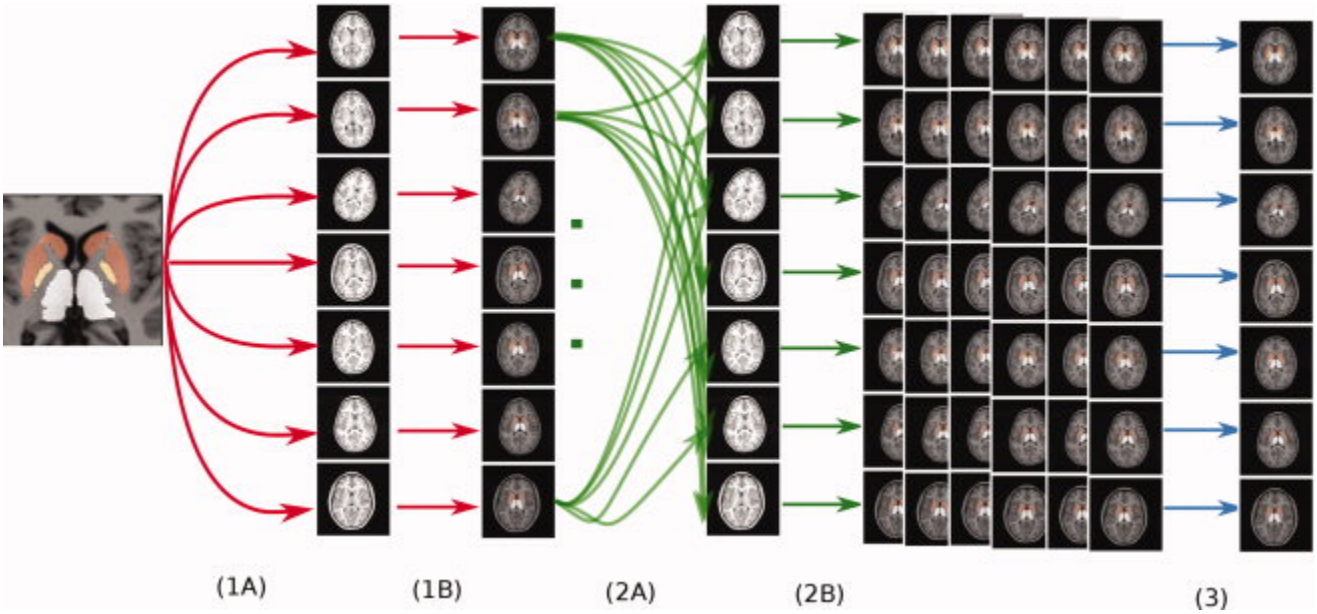
Creating Atlases with MAGeT

Introduction

The MAGeT algorithm, described in detail in Chakravarty et. al, is a way to automatically generate labeled atlases for MRI images based on existing templates. This method results in increased accuracy over previous approaches and reduces the amount of manual segmentation time often required to create labels. A schematic representation (Figure 1 from Chakravarty et. al) of this method is shown below:



In the following sections, I will refer to steps 1A, 1B, 2A, 2B and 3 from this figure to describe the relevant features in the code.

Pre-requisites

Input Atlases

In order to run this software, you must have at least one starting image with a set of labels and a mask for that image. Multiple input atlases can be used, but at least one is required. This atlas should be as accurate as possible, as it is read in step 1A and the labels are then propagated to all of the input brains, as in step 1B. *If you do not have at least one atlas (with corresponding labels and mask), you cannot use this algorithm.*

The following naming scheme **MUST** be used for an atlas/label/mask group:

- name.mnc
- name_labels.mnc
- name_mask.mnc

Specifically, the name that precedes _labels.mnc and _mask.mnc must match the name of the atlas itself. **If you do not use this naming scheme, the MAGeT code will not run correctly.** The following is an example of an acceptable naming scheme:

- MWM_version_1_average.mnc
- MWM_version_1_labels.mnc
- MWM_version_1_mask.mnc

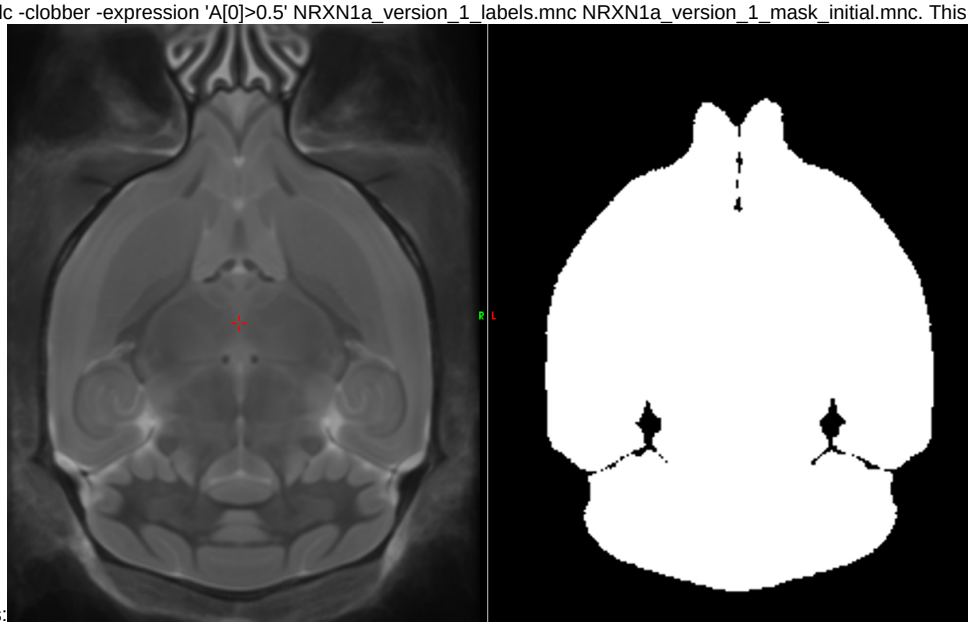
When running MAGeT at MICe, existing atlases can be found in /axiom2/projects/software/mouse-brain-atlases. The most up-to-date and accurate atlases we have generated can be found here. They were created by a combination of existing methods and hand segmenting. More information about these atlases can be found on the Mouse Brain Atlases wiki page. When running MAGeT on HPF, access the atlases in /hpf/largeprojects/MICe/tools/atlases.

All input atlases must be in a folder together, and this folder cannot contain any other files. This folder is specified as the --atlas-library command line option, and using /axiom2/projects/software/mouse-brain-atlases will not work, as there are additional files in this directory.

Masks

As is discussed in the previous section, a mask must be included with the standard set of atlas/label pairs in order for the algorithm to run properly. The reason for this mask is two-fold. First, the algorithm can be used to generate a mask for input images, should you need one. In order to do this, an initial mask must be provided. Secondly, even if the images themselves are not masked, specifying a mask limits the region over which the alignment between two images takes place, increasing the accuracy of the alignment.

If you have an existing atlas/label pairing and would like to create a mask, here is a simple way to do it:

1. minccalc -clobber -expression 'A[0]>0.5' NRXN1a_version_1_labels.mnc NRXN1a_version_1_mask_initial.mnc. This will create a mask that looks like this: 
2. After creating the mask as shown in the image above, you will want to "fill" in the dark areas within the mask. This can be accomplished by doing a series of dilations and erosions. Note that the exact number of dilations and erosions needed depends on the brain in question, and how tightly masked it needs to be. Dilations and erosions can be done using the mincmorph command: mincmorph -clobber -successive DDDDDDDDEEEEEEEE NRXN1a_version_1_mask_initial.mnc NRXN1a_version_1_mask.mnc. The resulting mask will look like this: 

Alignment

This code assumes that all brains (both atlases and inputs) are already more or less in the same space. In nomenclature many of us at MICe are familiar with, this means that they are already in LSQ6 space. Eventually, the MAGeT code will do this alignment for you, if needed. For now, brains that have undergone six-parameter alignment must be used.

Usage

MAGeT is written using the Pydpiper framework. The latest version is installed at MICe with our current set of libraries and executables in: /axiom2/projects /software/arch/linux-x86_64-eglibc2_11_1. For installation elsewhere, see the Installation section of this page, as well as the PydPiper wiki page and github repository for more information.

To see the list of command line options available for MAGeT, simply type MAGeT.py --help. Below, we list MAGeT specific. All options are contained in option groups, and while some are specific to MAGeT only (identified as MAGeT options), several of the options are used both in MAGeT and in other Pydpiper modules.

```
Usage: MAGeT.py [options] input files

General registration options:
    General options for running various types of registrations.

    --pipeline-name=PIPELINE_NAME
                        Name of pipeline and prefix for models.
    --registration-method=REG_METHOD
                        Specify whether to use minctracc or mincANTS for non-
                        linear registrations. Default is minctracc.
    --mask-dir=MASK_DIR
                        Directory of masks. If not specified, no masks are
                        used. If only one mask in directory, same mask used
                        for all inputs.

  MAGeT options:
    Options for running MAGeT.

    --atlas-library=ATLAS_LIB
                        Directory of existing atlas/label pairs
    --no-pairwise       Pairwise crossing of templates. Default is true. If
                        specified, only register inputs to atlases in library
    --mask              Create a mask for all images prior to handling labels
    --mask-only         Create a mask for all images only, do not run full
                        algorithm
    --max-templates=MAX_TEMPLATES
                        Maximum number of templates to generate
    --masking-method=MASK_METHOD
                        Specify whether to use minctracc or mincANTS for
                        masking. Default is minctracc.

  LSQ12 and NLIN registration protocol:
    Option to specify a protocol and override default LSQ12 and NLIN
    parameters.

    --lsq12-protocol=LSQ12_PROTOCOL
                        Can optionally specify a registration protocol that is
                        different from defaults. Parameters must be specified
                        as in the following example:  applications_testing/tes
                        t_data/minctracc_example_linear_protocol.csv  Default
                        is None.
    --nlin-protocol=NLIN_PROTOCOL
                        Can optionally specify a registration protocol that is
                        different from defaults. Parameters must be specified
                        as in either or the following examples:  applications_
                        testing/test_data/minctracc_example_nlin_protocol.csv
                        applications_testing/test_data/mincANTS_example_nlin_p
                        rotocol.csv  Default is None.
```

The following command line options are part of the PydPiper framework, and are discussed in more detail on the How to Launch Pydpiper Executors wiki page:

```
--uri-file  --use-ns --create-graph --num-executors --time  --proc  --mem  --ppn  --queue  --sge-queue-opts --
restart --output-dir --execute --no-execute
```

⚠ --output-dir, which is one of the pydpiper options, is the name of the directory where you would like to put the pydpiper output and backups. The default directory is the directory from which the program is launched.

In the specified output directory, a subdirectory is created for each input brain. The input brains are the brains that you would like to create labels for. Within each input brain subdirectory, additional subdirectories (labels, resampled, tmp, transforms, log) are created. The final set of voted labels for each brain can be found in the labels directory. For the brain img_19jan12.6_dc-lsq6, the voted labels are: img_19jan12.6_dc-lsq6/labels/img_19jan12.6_dc-lsq6_votedlabels.mnc

Specific examples of how one might use these command line options are found in the Examples section below.

General registration options are used in this module, as well as others and are described here: General Registration Options. *Note that the --pipeline-name option is currently not used in MAGeT, and specifying a pipeline name will not change the output or location of any of your files*. If this changes we will update this space.

MAGeT options are options specific to MAGeT only:

1. **--atlas-library** is a directory where all of your starting atlas/label/mask triads should be stored. The code assumes that all of the atlases you wish to use for step 1A are in this directory, and it will use ALL of the atlases in this directory. Proper naming conventions (described above) must be used or the code will fail at this step.
2. **--no-pairwise** is an option that by default is set to False. If --no-pairwise is specified, the algorithm will only complete steps 1A and 1B. This means that for each input brain, a set of labels will be calculated from each of the input atlases in the atlas library, and voxel voting will be performed on each of these sets of labels. The inputs themselves will NOT be used as templates and further crossed, as shown in steps 2A and 2B. If only one atlas is specified in the atlas library and the --no-pairwise option is also specified, this amounts to the standard atlas-to-atlas protocol that is described here.
3. **--mask** is turned off by default (**and this option is broken in MAGeT.py version 1.10 and older!**). If you specify this option, the algorithm will create a mask for each of your input files in addition to a set of labels. It does this by using the mask specified for each atlas in the atlas library and executing steps 1A and 1B and voting on the resulting mask to get the best fit. After doing this, a mincmath command will apply the generated mask to each input file, creating a masked version for each of your inputs. (The atlases in the atlas library are also masked.) The algorithm then uses the masked version of each input file and executes the entire algorithm (1A, 1B, 2A, 2B, 3), calculating labels for each masked file. The final mask for each file can be found in the labels directory and the masked version of the file can be found in the resampled directory.
    a. For img_19jan12.6_dc-lsq6, the mask is: img_19jan12.6_dc-lsq6/labels/img_19jan12.6_dc-lsq6_mask.mnc
    b. For img_19jan12.6_dc-lsq6, the masked version of the file (created by multiplying the original input file by the mask specified in 4a) is: img_19jan12.6_dc-lsq6/resampled/img_19jan12.6_dc-lsq6_masked.mnc
4. **--mask-only** is False by default. Specifying this option means that a mask will be created for each of your input brains, but MAGeT will not be performed on the labels. Specifying --mask together with --mask-only is not necessary. Only one needs to be specified, but if both are specified, --mask-only will be executed.
5. **--max-templates** is 25 by default. This means that if you have 20 input brains, all steps will be performed on all brains. If you have 50 input brains and the default number of templates is unchanged, then steps 1A and 1B will be performed on 25 brains only, and then those 25 brains will be used as templates for steps 2A and 2B.
6. **--masking-method** can be specified. The default method for each is minctracc, though a mincANTS protocol can also be used.

LSQ12 and NLIN registration protocol:

Each pairwise registration in MAGeT includes a linear and non-linear component. Although there are default protocols set for each, an alternate set of protocols may be specified.

1. **--lsq12-protocol** A .csv file for the 12-parameter component of the registration may be specified.
2. **--nlin-protocol** A .csv file for the non-linear component of the registration may be specified. The format of this file is different for minctracc and mincANTS and needs to match the specified --registration-method.

NOTE: The current default lsq12 and nonlinear registration protocols are not optimized for MAGeT. To use the optimized MAGeT protocols, use the following .csv files

--lsq12-protocol=applications_testing/test_data/default_linear_MAGeT_prot.csv

--nlin-protocol=applications_testing/test_data/default_nlin_MAGeT_minctracc_prot.csv

These files are available on the pydpiper github repository, or in the applications_testing/test_data/ subdirectory of the pydpiper source code on your system.

At MICe you should these:

--lsq12-protocol=/axiom2/projects/software/arch/linux-3_2_0-36-generic-x86_64-eglibc-2_15/src/pydpiper/applications_testing/test_data/default_linear_MAGeT_prot.csv

--nlin-protocol=/axiom2/projects/software/arch/linux-3_2_0-36-generic-x86_64-eglibc-2_15/src/pydpiper/applications_testing/test_data/default_nlin_MAGeT_minctracc_prot.csv

On HPF you should these:

--lsq12-protocol=/hpf/largeprojects/MICe/tools/protocols/linear/default_linear_MAGeT_prot.csv

--nlin-protocol=/hpf/largeprojects/MICe/tools/protocols/nonlinear/default_nlin_MAGeT_minctracc_prot.csv

NOTE: You must also specify --masking-nlin-protocol=/hpf/largeprojects/MICe/tools/protocols/nonlinear/default_nlin_MAGeT_minctracc_prot.csv

Examples

All of the examples described in this section assume an input of three atlases (A1.mnc, A2.mnc, A3.mnc) and their corresponding labels and masks, as well as three images to register: img_A.mnc, img_B.mnc, img_C.mnc. The atlases are all in a directory called simple-atlas-names and the images are in a directory called test_data.

1. Run the full multi-atlas MAGeT algorithm on the MICe cluster. For a regular image registration pipeline, the input data should be your "lsq6" resampled files. I.e., the files that were resampled using only a rigid body registration (6 parameters)

```
MAGeT.py --atlas-library=simple-atlas-names --output-dir=full_maget --num-executors=10 --proc=1 --mem=2
--registration-method=minctracc --max-templates=20 --queue=sge --lsq12-protocol=/hpf/largeprojects/MICe
/tools/protocols/linear/default_linear_MAGeT_prot.csv --nlin-protocol=/hpf/largeprojects/MICe/tools
/protocols/nonlinear/default_nlin_MAGeT_minctracc_prot.csv --masking-nlin=/hpf/largeprojects/MICe/tools
/protocols/nonlinear/default_nlin_MAGeT_minctracc_prot.csv  --files /point/to/your/lsq6/files*.mnc
```

2. Run a standard multi-atlas segmentation procedure on the brains on your desktop without doing the full MAGeT procedure:

```
MAGeT.py --atlas-library=simple-atlas-names --output-dir=multi --no-pairwise --num-executors=4 --proc=1
--mem=2  --registration-method=minctracc test_data/lsq6_files*.mnc
```

3. Run the full multi-atlas MAGeT algorithm on your local machine, launching the executors separately from the pipeline and creating a mask first :

   This option (–mask) is broken in MAGeT.py version 1.10 and older. Don't run this unless you are using a newer MAGeT.py version

```
MAGeT.py --atlas-library=simple-atlas-names --output-dir=20June_masked_maget --registration-
method=minctracc --mask test_data/*.mnc

pipeline_executor.py --num-executors=1 --proc=4 --uri-file=uri-file-for-this-pipeline
```

   Note that in this example, MAGeT.py and pipeline_executor.py are launched in separate terminals. --uri-file may also be specified in the MAGeT.py command, but if nothing is specified a file called "uri" will be created in the same directory where the command was launched. This is what needs to be set as the uri file for the executor, or it will not run properly.
4. Run the full multi-atlas MAGeT algorithm on the MICe cluster, specifying alternate registration protocols:

```
MAGeT.py --atlas-library=simple-atlas-names --registration-method=minctracc --output-
dir=20June_masked_maget --lsq12-protocol=applications_testing/test_data/default_linear_MAGeT_prot.csv --
nlin-protocol=applications_testing/test_data/default_nlin_MAGeT_minctracc_prot.csv -output-
dir=20June_full_maget --num-executors=4 --proc=1 --mem=2 --queue=sge test_data/*.mnc
```

Note that the above example will create 4 executors on the cluster, each with one processor and 2G of RAM.

Input files note:

Since I find this a bit confusing, I will specify here for any users that are also lost. One of the components of the MAGeT.py command is "files," meaning your input files. In the examples, this is annotated as:

`--files /point/to/your/lsq6/files*.mnc`

For reference, in the MBM registration, these are located at:

Template:

--files /hpf/largeprojects/MICe/[user]/[folder_where_registration_was_run]/[name_of_registration]_processed/*/resampled/*sept2014_dist_corr_N_I_lsq6.mnc

Example:

--files /hpf/largeprojects/MICe/zsu/CCP_Raw/ExVivo/Mnc_Files/CCP_Tide/coils_appended/CCP_ExVivo_Final_processed/CCP_*_dist_corr/resampled/CCP_*sept2014_dist_corr_N_I_lsq6.mnc

And in a 2-level in vivo registration, these are located at:

Template:

--files /hpf/largeprojects/MICe/[user]/[folder_where_registration_was_run]/[name_of_registration]_first_level/*_processed/*/resampled/*sept2014_dist_corr_N_I_lsq6.mnc

Example:

--files /hpf/largeprojects/MICe/zsu/CCP_Registrations/InVivo/OxyFinal_Spring2018/TwoLevelReg_CCPOxy_Spring2018_first_level/*_processed/*.RegReady/resampled/*_N_I_lsq6.mnc

Feel free to go into those directories to see their locations if it helps.

Analyzing structure volumes from MAGeT output

When you have segmentations for all your lsq6 files, you will need to use the correct arguments for anatGetAll in RMINC to determine the volumes of the structures. You might be more familiar with having an atlas only for the final non linear average. In that case you call anatGetAll with this atlas and the method to determine the structure volumes is "jacobians" (which the default). The will use the Jacobian determinant values to integrate over the labels in the segmentation of the final non linear atlas. This is not the correct method when you have labels for the original brains (in lsq6 space). Now we can simply use the number of voxels present for each label, and the method for anatGetAll to use is "labels". For more information see (scroll down a bit after following the link...): RMINC analysis with individual segmentations

Tips

For generally useful Pydpiper tips, go to the main Tips page. For general debugging information, go to the Debugging page.

1. For best results, mask your brains. Although a mask will always be specified based on the masks read in from the atlas library, creating and applying a mask for each individual brain (either using MAGeT.py or beforehand) will likely increase the accuracy of your labels.

   
   But only when running MAGeT.py version 1.11 or higher, see note above

2. MAGeT will generate output files for you to monitor the progress of your pipeline. In addition to each of the files generated in the log directories, the following files will be created in the directory where you launched your pipeline:

   - MAGeT.py-20120620-131859.log (Timestamp based on creation.)
   - pipeline_executor.py-20120620-135720.log
   - MAGeT-pipeline-stages.txt

   The log files give information about which stages are currently running and the MAGeT-pipeline-stages.txt file gives a listing of each stage in the pipeline.