MUSE Software Manual

Release v3.0 (r352)

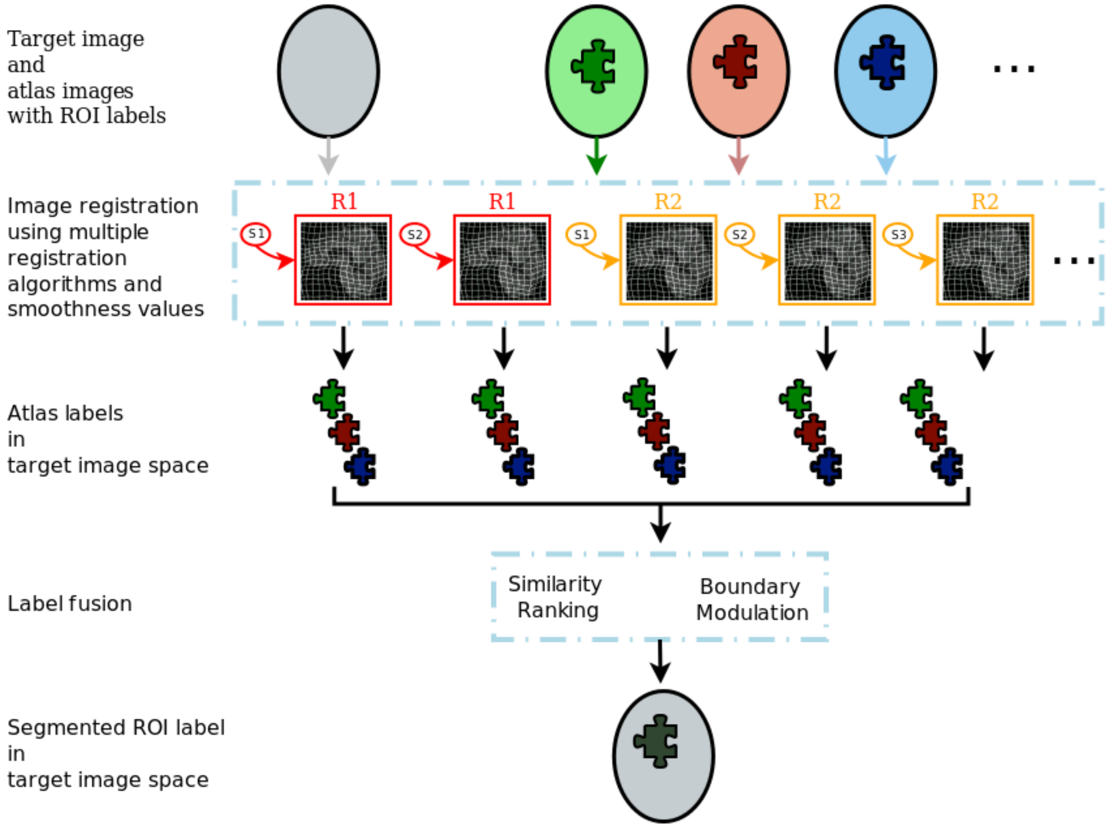
Jimit Doshi

November 04, 2015

Contents

1 About the Algorithm

MUSE generates a large ensemble of candidate labels in the target image space using multiple atlases, registration algorithms and smoothness values for these algorithms. The ensemble is then fused into a final segmentation. An illustration of the MUSE algorithm is given in the following figure.


Target image and atlas images with ROI labels
...
Image registration using multiple registration algorithms and smoothness values
S1
R1
S2
R1
S1
R2
S2
R2
S3
R2
...
Atlas labels in target image space
Label fusion
Similarity Ranking
Boundary Modulation
Segmented ROI label in target image space

2 Download

2.1 Software License

The MUSE software is freely available under a BSD-style open source license that is compatible with the Open Source Definition by The Open Source Initiative and contains no restrictions on use of the software. The full license text is included with the distribution package and available online.

2.2 Documentation

MUSE Manual: Online version of this manual.

MUSE ChangeLog: Summary of changes, new features, and bug fixes.

2.3 System Requirements

Operating System: Linux

2.4 Register for Download

Please **register online** to receive an email with the download links of the software.

3 Installation

See the BASIS guide on software installation for a complete list of build tools and detailed installation instructions.

3.1 Prerequisites

Dependency	Version	Description
BASIS	2.1.0	A meta-project developed at SBIA to standardize the software development.
DRAMMS	1.4.1	A registration algorithm developed at SBIA to warp images.
ANTS	1.9.x	A registration algorithm developed at PICSL to warp images.
MICO	1.0.0	A segmentation algorithm developed at SBIA to segment images.
AFNI		Using the version built on 2008_07_18_1710
FSL	4.1.5	A comprehensive library of analysis tools for brain imaging data
NIBABEL	1.2.0	A python package for read and write access to common medical file formats
NUMPY	1.6.1	A python package for scientific computing

* The versions listed are the minimum versions of the softwares for which the MUSE package was tested.

3.2 Job Scheduler

If you have access to a computing cluster which has a job scheduler/queuing software (SGE, PBS etc) installed, it can be used to significantly reduce the (wall-clock) time it will take for the MUSE software to produce the results. During the installation process, you can initialize the `SCHEDULER` variable with the particular version of your job scheduler. Currently, there are four options that are supported. You can select the one that best fits your system:

```
SGE  - Sun Grid Engine
PBS  - Portable Batch System
NONE - No queuing system (default)
MISC - User defined setting
```

If you have a different queuing software and you select the "MISC" option, you need to modify the **src/schedulerSettings/SettingsMISC.sh** file within the package with the appropriate options and arguments that are specific to your queuing system. You can refer to the corresponding files for SGE and PBS as examples.

3.3 Configure

1. Extract source files:

```
    tar -xzf muse-3.0.0-source.tar.gz
```

2. Create build directory:

```
mkdir muse-3.0.0-build
```

3. Change to build directory:

```
cd muse-3.0.0-build
```

4. Run CMake to configure the build tree by using one of the following commands:

```
cmake -D CMAKE_INSTALL_PREFIX:STRING=/Full/path/to/install/muse/
      -D SCHEDULER:STRING=???
      ../muse-3.0.0-source
```

3.4 Build

After the configuration of the build tree, the software can be build using GNU Make:

```
make
```

3.5 Test (Optional)

After the build of the software, optionally run the tests using the command:

```
make test
```

Allow 30-60 mins for the test to finish. In case of a test failure, re-run the test, but this time by executing CTest directly with the `-V` option to enable verbose output and redirect the output to a text file:

```
ctest -V >& muse-test.log
```

and attach the file `muse-test.log` to the issue report.

3.6 Install

The final installation copies the built files and additional data and documentation files to the installation directory specified using the `CMAKE_INSTALL_PREFIX` option during the configuration of the build tree:

```
make install
```

After the successful installation, the build directory can be removed again.

4 Manual

4.1 MUSE Default Command

The main command of MUSE which labels an input image into a set of desired regions of interest is named `muse`. The simplest use is:

```
muse  -i /path/to/source/sourceimage.hdr
```

This command will internally submit 22 registrations (11 DRAMMS + 11 ANTS) and 1 labelFusion job. Once finished, it'll generate the output file in the /Path/To/Source/Directory/ directory:

> *Input_n3_str_muse.nii.gz* - The fnal labeled image

Supported File Formats: NIfTI-1 (recommended)

Supported Datatypes: byte (unsigned char, uint8), int8, short, int16, uint16, float, float32, int32.

4.2 MUSE Options

To parcellate the input brain image using the default options, but without using the computing cluster:

```
muse
 -i /Path/To/Source/Directory/Input_n3_str.nii.gz
 -Q;
```

To parcellate the input brain image using the default options, but without the cerebellum and storing the results at a user-specified destination :

```
muse
 -i /Path/To/Source/Directory/Input_n3_str.nii.gz
 -D /Path/To/Destination/Directory/
 -C;
```

To parcellate the input brain image, but use the WMLS mask to exclude those regions from intensity weighting/correction:

```
muse
 --in /Path/To/Source/Directory/Input_n3_str.nii.gz
 --dest /Path/To/Destination/Directory/
 --WML /Path/To/WMLS/Directory/Input_n3_str_WMLS_mask.nii.gz;
```

To parcellate the input brain image into ROIs using the default options. Additionally, use 6 CPU cores during processing of indivudual ROIs to speed up the process:

```
muse
 -i /Path/To/Source/Directory/Input_n3_str.nii.gz
 -D /Path/To/Destination/Directory/
 -P 6;
```

To parcellate the input brain image into ROIs using Majority Voting to combine the results of different registrations:

```
muse
 --in /Path/To/Source/Directory/Input_n3_str.nii.gz
 --dest /Path/To/Destination/Directory/
 --noIC
 --noSim
 --noFuzzy;
```

To parcellate the input brain image into ROIs and keep some of the important intermediate results:

```
muse
 -i /Path/To/Source/Directory/Input_n3_str.nii.gz
 -D /Path/To/Destination/Directory/
 -k 1;
```

4.3 Calculating ROI Volumes generated from MUSE

To calculate the ROI volumes for each unique ROI within the labeled ROI image:

```
muse-calculateVolumes -i /Path/To/Destination/Directory/Input_n3_str_muse.nii.gz;
```

To calculate the ROI volumes for each unique ROI within the labeled ROI image and output the results in a csv file:

```
muse-calculateVolumes
 -i /Path/To/Destination/Directory/Input_n3_str_muse.nii.gz
 -s Input
 -o /Path/To/Destination/Directory/Input_n3_str_muse.csv;
```

To calculate the derived ROI volumes (based on a hierarchical parcellation) from the labeled ROI image:

```
muse-calculateVolumes
 -i /Path/To/Destination/Directory/Input_n3_str_muse.nii.gz
 -s Input
 -o /Path/To/Destination/Directory/Input_n3_str_muse.csv
 -d
 -v /Path/To/Source/Directory/Input_n3_str.nii.gz;
```

5 Publications

[TEMP2015]

6 People

6.1 Advisors

- Christos Davatzikos (Christos.Davatzikos@uphs.upenn.edu)

6.2 Software Development

- Jimit Doshi (Jimit.Doshi@uphs.upenn.edu)

6.3 Contributors

- Guray Erus
- Yangming Ou
- Meng-Kang Hsieh

6.4 Testers

- Harsha Battapady
- Meng-Kang Hsieh
- Xiao Da
- Martin Rozycki
- Irem Aselcioglu

References

[TEMP2015] Jimit Doshi, Guray Erus, Yangming Ou, Susan M. Resnick, Ruben C. Gur, Raquel E. Gur, Theodore D. Satterthwaite, Susan Furth, Christos Davatzikos, for the Alzheimer's Neuroimaging Initiative. MUSE: MUlti-atlas region Segmentation utilizing Ensembles of registration algorithms and parameters, and locally optimal atlas selection (Under Revision)