Anima

Release 4.0

Jan 06, 2021

First steps

1 First steps **3**
- 1.1 Compiling Anima from source . . . 3
- 1.2 Installing Anima from binary releases . . . 4
- 1.3 Installing Anima scripts . . . 4

2 Tools documentation **7**
- 2.1 Diffusion imaging tools . . . 7
- 2.2 Registration tools . . . 12
- 2.3 Segmentation tools . . . 17
- 2.4 Group comparison tools . . . 19
- 2.5 Relaxometry tools . . . 21
- 2.6 Denoising tools . . . 23
- 2.7 Basic image processing tools . . . 24

3 Scripts documentation **27**
- 3.1 Atlasing scripts . . . 27
- 3.2 Diffusion imaging scripts . . . 30
- 3.3 Relaxometry scripts . . . 32
- 3.4 Segmentation scripts . . . 32

4 Citing Anima or Anima scripts **35**

5 Licensing **37**
- 5.1 License . . . 37

Welcome to the documentation of Anima and Anima scripts. Thanks for your interest. Here are a few directies to installing and what Anima and Anima scripts may do for you. Mailing lists are available for any questions on the tools at anima-users@inria.fr

CHAPTER 1

First steps

If this is your first time here, you may be very well interested in installing Anima. This may be done either from *source compilation* or by *installing a binary package of your choosing*. Once you have done so and if you are interested in getting our Anima scripts (scripts that use Anima tools to perform complex processing series), look for the installation steps *here*.

1.1 Compiling Anima from source

1.1.1 Requirements

Anima is multi-platform and will compile and run on the major platforms: Windows, OSX and Linux. It requires two major things:

- CMake (>= 3.1.0) as a cross-platform Makefile generation software (we use its super-project ability to also download and compile its dependencies)
- A compilation environment: Visual studio 2015 on Windows, Xcode and developer tools on OSX, gcc/g++ or any other C++ compiler on Linux

1.1.2 Code architecture

Anima itself follows a modular structure, organized in 6 main modules, each one associated to a folder on the main repository:

- Math-tools. This module contains generic mathematical tools: statistical tests, statistical distributions, optimizers, spherical harmonics...
- Filtering. This module contains image filters: smoothing, denoising...
- Diffusion. This module includes diffusion model estimation as well as tractography algorithms.
- Registration. This one holds registration tools: resamplers, interpolators, transformations, registration algorithms...

- Segmentation. segmentation tools
- Quantitative MRI. Estimation of quantitative parameters from MRI, MR simulation

Some of these modules are dependent on others (e.g. math-tools is the base to everything). If you know what you are doing, you can choose to compile only a subset of these.

1.1.3 Compilation Instructions

Anima is available as a superproject, including all its modules and links and instructions for its dependencies. Building ANIMA is simple:

- clone the repository from github (use the first line by default, the second if you have set up your SSH keys):

```
git clone https://github.com/Inria-Visages/Anima-Public.git src
git clone git@github.com:Inria-Visages/Anima-Public.git src
```

- then, run CMake in a new build folder, change any options if you wish to change the default compilation (which downloads and compiles all dependencies and tools): *mkdir build ; cd build ; ccmake ../src* (use the CMake graphical interface on Windows)
- build using your environment (a *make* or *ninja* will be enough on Linux and OSX, open Visual Studio on Windows)

And there you go, the *bin* folder in the build directory will contain all tools described in this documentation.

1.2 Installing Anima from binary releases

You can download and install Anima as pre-compiled binary packages. We provide regular builds, so-called releases, on oour website for the three major OSes: Windows, macOS and Linux (Ubuntu or Fedora).

To install these builds, simply follow these steps:

- Download the latest binary release from our website
- Unzip the downloaded file to your prefered folder
- Add this folder to your path and the tools will be available from the command line

1.3 Installing Anima scripts

1.3.1 Requirements

Anima scripts does not require many things. But here is a list of what is required:

- Python (>= 3.0): for most scripts, Python is required. Python 2 or 3 are supported although python 3 and more recent is advised
- numpy, scipy, pandas and pydicom are required for some scripts. The following command should do the trick to install them:

```
pip install numpy pydicom pandas scipy
```

- Anima: either built from source or the releases. Some scripts (e.g. in diffusion) require non public code that are in the releases but not the open-source code unless you have access to our repositories. See there how to *compile it from source* or *install the binaries*

- Anima scripts data: data required for some scripts to work (brain extraction and diffusion scripts)
- A cluster with an OAR scheduler for the atlasing scripts

1.3.2 Anima scripts installation instructions

- Get the additional data repository by cloning it. Be careful, you should have installed Git LFS first to be able to get it

```
git clone https://github.com/Inria-Visages/Anima-Scripts-Data-Public.git
```

- Get the release of Anima scripts from our website or clone the repository using

```
git clone https://github.com/Inria-Visages/Anima-Scripts-Public.git
```

- Copy / paste the **example-config.txt** file in **.anima/config.txt** in your home folder
- Update the paths in your config file to match where Anima binaries are and where the additional data folder is (use full paths, no tilde)
  - **anima-scripts-public-root** should link to where you cloned this repository
  - **anima** should point to where you have the Anima executables
  - **extra-data-root** should point to where you have cloned the additional data repository

And now you are all setup to go, you can now read about all scripts in the other sections of the documentation.

CHAPTER 2

Tools documentation

More detailed documentation on the different tools available in Anima current release is provided here. It includes documentation on the following tool categories:

- *Diffusion imaging*
- *Registration*
- *Segmentation*
- *Patient to group comparison*
- *MR relaxometry*
- *MR denoising*
- *Basic image processing tools*

2.1 Diffusion imaging tools

This section describes only the core estimation and tractography features of Anima. For registration of diffusion models and EPI distortions correction, please refer to the *registration page*.

2.1.1 Diffusion imaging

Multiple compartment models

Anima provides tools for the estimation and processing of multi-compartment models. So far, we have implemented results of our research from papers on estimation and model averaging of MCM. More works on tractography are to come in future releases.

MCM estimation

The **animaMCMEstimator** tool provides an implementation of MCM estimation from DWI data, following the maximum likelihood method described in [8]. It provides in its public version tools to estimate models including isotropic compartments (free water, isotropic restricted water, dot compartment, and directional compartments (stick, zeppelin, tensor, NODDI [12]). The binary release also allows for the estimation of the DDI compartment model [9].

Example: this estimates a multi-compartment model with two components (`-n 2`), each anisotropic component being a tensor (`-c 3`), a free water compartment (`-F`) and an isotropic restricted water compartment (`-R`), using the variable projection method (`--ml-mode 2`) and the Levenberg Marquardt algorithm.

```
animaMCMEstimator -i DWI.nii.gz -o MCM_n2.mcm -O MCM_n2_b0.nrrd -g DWI.bvec -b DWI.
↪bval -n 2 -c 3 -F -R --optimizer levenberg --ml-mode 2
```

More options of this tool are available in the documentation of **animaMCMEstimator**. Interestingly the number of random restarts has a large influence on computation time (but also on precision). We fix it by default to 6 but getting to smaller values will make the estimation faster.

The output file format is the MCM format that may be read by medInria from version 3.0. The format is an XML file linking to individual compartment images and weights.

Example: This .mcm file references image files describing each compartment (together with a flag for its type). In addition, it references a vector image containing for each voxel the respective weights of each compartment.

```
<?xml version="1.0"?>
<Model>
<Weights>testing_weights.nrrd</Weights>
<Compartment>
<Type>FreeWater</Type>
<FileName>testing_0.nrrd</FileName>
</Compartment>
<Compartment>
<Type>IRWater</Type>
<FileName>testing_1.nrrd</FileName>
</Compartment>
<Compartment>
<Type>Tensor</Type>
<FileName>testing_2.nrrd</FileName>
</Compartment>
<Compartment>
<Type>Tensor</Type>
<FileName>testing_3.nrrd</FileName>
</Compartment>
```

MCM model averaging

In addition to estimation, we provide two ways of performing model selection and averaging:

- **animaMCMEstimator** proposes model selection based on AICc criterion (option `-M`): in that case, all models from 0 to N will be estimated and the one with the optimal AICc will be kept
- The binary Anima tools include one called **animaMCMModelAveraging** that implements the method proposed in [10]. This method uses outputs from model estimations from 0 to N fiber compartments and their AICc scores (produced by **animaMCMEstimator**) to compute an average MCM volume with simplification to the optimal number of fibres in each voxel. This tool is not yet open source and as such will be distributed only as binary versions in the Anima releases.

Examples:

- this computes a multi-tensor model at each voxel, with at most 3 anisotropic compartments per voxel (this number being decided based on the AICc criterion).

```
animaMCMEstimator -i DWI.nii.gz -o MCM_n3_MS.mcm -O MCM_n3_MS_b0.nrrd -g DWI.bvec -b␣
↪DWI.bval -n 3 -c 3 -FR --optimizer levenberg --ml-mode 2 -M
```

- this performs model averaging as proposed in [10], with model simplification. It uses as an input two text files, each having on each line an image file name. *listMCM.txt* contains the list of MCM files to be averaged (from 0 to N anisotropic compartments) and *listAIC.txt* contains the corresponding AICc files (all these images may be written from **animaMCMEstimator**).

```
animaMCMModelAveraging -i listMCM.txt -o MCM_avg.mcm -a listAIC.txt -m MCM_avg_mose.
↪nrrd -C
```

MCM processing tools

animaMCMAverageImages provides a way to average several volumes of MCM into just one (e.g. an atlas of those images), using the averaging and interpolation framework proposed in [11]. It works in a similar manner to the *animaAverageImages* described in the basic tools page.

animaMCMScalarMaps provides voxel-wise measures extracted from an MCM image. The currently supported parameters include: isotropic water proportions (free water and isotropic restricted), anisotropic water proportion, apparent diffusivities (mean, axial, radial), fractional anisotropy. These last parameters can be extracted either only from anisotropic compartments (weighted average over them) or the whole model.

DTI estimation and processing

DTI estimation

DTI estimation is performed using two tools in ANIMA, implementing basic matrix-based DTI estimation and extrapolation.

animaDTIEstimator takes as inputs a 4D DWI image, a set of gradient directions and b-values and estimates tensors at each voxel. Gradient directions may be in the medInria format (one line per gradient) or the bvec format. B-values may be specified using a single number or a text file (either one line for each volume b-value or a bval file). Estimated tensors may be degenerated in some places. In that case, the tool outputs either zero values or the degenerated tensors depending on the `-K` option.

Note: In all Anima tools, the tensors are stored using a 6-component vector image representing the upper diagonal part of the tensors. These values are stored in column-first order.

DTI scalar maps

animaComputeDTIScalarMaps computes the usual fractional anisotropy (FA), apparent diffusivity coefficient (ADC), axial (AD) and radial diffusivity (RD) maps from a tensor image.

Log-Euclidean tools

These tools implement Arsigny et al. log and exponential maps on tensors:

- **animaLogTensors** computes the log map of tensors. The `-S` option switches between the vector representation and matrix representation of the log (sqrt(2) scaling factor on non diagonal terms).
- **animaExpTensors** computes the exponential map of log-vectors. The `-S` option is the equivalent of the one in **animaLogTensors**: it divides non diagonal values by sqrt(2).

ODF estimation and processing

In all Anima tools, the ODFs are represented in the real spherical harmonics basis proposed by Descoteaux et al. in [2]. Coefficients are stored in vector images as explained in that publication.

ODF estimation

animaODFEstimator estimates ODFs at each voxel using one of two estimation methods: (1) Descoteaux et al. [2] with or without regularization, with or without ODF spherical deconvolution [3], and (2) Aganj et al. [4] providing naturally normalized ODFs at each voxel. The amount of ODF spherical deconvolution may be specified with the -s parameter, the estimation method with `-R`.

Example: this estimates ODFs of order 6 from DWI.nii.gz using Aganj et al. method.

```
animaODFEstimator -i DWI.nii.gz -o ODF.nii.gz -g grads.bvec -k 6 -R
```

Generalized FA

animaGeneralizedFA computes the generalized fractional anisotropy from an image of ODFs stored in our format.

2.1.2 Tractography

Anima implements tractography based on the three supported models: DTI, ODFs and MCM. It can be further divided into two classes of tractography methods: deterministic and probabilistic. All algorithms output fibers either in .vtk, .vtp (VTK format) or .fds (a meta-fibers format that can easily be read by medInria).

Deterministic tractography

Deterministic tractography algorithms are described in [5]. They implement FACT [6] for DTI and a modified version of it to handle crossing fibers for ODFs. Those algorithms progress step by step following the local directions provided by the local model available and stopping if some criterions are met (local fiber angle, fiber length, FA threshold, ...).

- **animaDTITractography** implements DTI based deterministic tractography.
- **animaMCMTractography** implements multi-compartment models based deterministic tractography.

Probabilistic tractography

Probabilistic tractography tools implement for MCM, ODF and DTI our multi-modal particle filtering framework for probabilistic tractography [7]. It relies on the simultaneous propagation of particles and their filtering relative to previous directions and the current model. This method further implements clustering of the particles to retain multi-modality, i.e. branching fibers.

- **animaDTIProbabilisticTractography** implements the filter for DTI tractography.
- **animaODFProbabilisticTractography** implements the filter for ODF tractography.

- **animaMCMProbabilisticTractography** implements the filter for multi-compartment models tractography.

Tractography tools

Application of transformations

animaFibersApplyTransformSerie works in the same way as resampler tools provided on the *registration page* except that it applies a series of transformations to a set of fibers. Please refer to that section for more details.

Counting fibers in image voxels

animaFibersCounter takes as an input a geometry image `-g`, and uses the input `-i` to know how many fibers go through each pixel of that image. The output may be either a fiber count or a fiber proportion (`-P` flag) i.e. the previous result divided by the number of fibers.

Filtering fibers

animaFibersFilterer uses a regions of interest (labeled) image to filter a set of fibers. The ROI image is a label image provided with the option `-r`. The `-t` and `-f` options can be given multiple times and are used to tell which labels a single fiber should go through (`-t`) and which labels should not be touched (`-f`).

Example: this filters the input fibers telling each fiber can be kept if it touches labels 1 and 2, but not 3.

```
animaFibersFilterer -i fibers.fds -o filtered_fibers.fds -r roi_image.nrrd -t 1 -t 2 -
↪f 3
```

2.1.3 References

1. Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. *Log-Euclidean Metrics for Fast and Simple Calculus on Diffusion Tensors*. Magnetic Resonance in Medicine, 56(2):411-421, August 2006.
2. Descoteaux, M., Angelino, E., Fitzgibbons, S., Deriche, R. *Regularized, Fast, and Robust Analytical Q-Ball Imaging*. Magnetic Resonance in Medicine 58, 497–510, 2007.
3. Descoteaux M, Deriche R, Knösche TR, Anwander A. *Deterministic and probabilistic tractography based on complex fibre orientation distributions*. IEEE Transactions on Medical Imaging, 28(2):269-86, 2009.
4. Iman Aganj, Christophe Lenglet, Guillermo Sapiro, Essa Yacoub, Kamil Ugurbil, Noam Harel. *Reconstruction of the orientation distribution function in single-and multiple-shell q-ball imaging within constant solid angle*. Magnetic Resonance in Medicine, 64(2):554-566, 2010.
5. Nicolas Wiest-Daesslé, Olivier Commowick, Aymeric Stamm, Patrick Perez, Christian Barillot, Romuald Seizeur, Sylvain Prima. *Comparison of 3 Diffusion Models to Track the Hand Motor Fibers within the Corticospinal Tract Using Functional, Anatomical and Diffusion MRI*. MICCAI 2011 Workshop on Computational Diffusion MRI (CDMRI'11), pp 150-157, Sep 2011.
6. Susumu Mori, Barbara J. Crain, V. P. Chacko, Peter C. M. Van Zijl. *Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging*. Annals of Neurology, 45(2):265–269, 1999.
7. Aymeric Stamm, Olivier Commowick, Christian Barillot, Patrick Perez. *Adaptive Multi-modal Particle Filtering for Probabilistic White Matter Tractography*. Information Processing in Medical Imaging, pp 594-606, 2013.

8. Aymeric Stamm, Olivier Commowick, Simon K. Warfield, Simone Vantini. *Comprehensive Maximum Likelihood Estimation of Diffusion Compartment Models Towards Reliable Mapping of Brain Microstructure*. 19th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2016.

9. Aymeric Stamm, Patrick Pérez, Christian Barillot. *A new multi-fiber model for low angular resolution diffusion MRI*. IEEE International Symposium on Biomedical Imaging, 2012.

10. Aymeric Stamm, Olivier Commowick, Patrick Pérez, Christian Barillot. *Fast Identification of Optimal Fascicle Configurations from Standard Clinical Diffusion MRI Using Akaike Information Criterion*. IEEE International Symposium on Biomedical Imaging, 2014.

11. Renaud Hédouin, Olivier Commowick, Aymeric Stamm, Christian Barillot. *Interpolation and Averaging of Multi-Compartment Model Images*, 18th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 354-362, 2015.

12. Hui Zhang, Torben Schneider, Claudia A. Wheeler-Kingshott, Daniel C. Alexander. *NODDI: Practical in vivo neurite orientation dispersion and density imaging of the human brain*, NeuroImage, 61:4, 1000-1016, 2012.

2.2 Registration tools

This part describes registration features in Anima. It includes linear and non linear registration (of anatomical and DTI images), as well as EPI distortion correction and tools for applying series of transformations to images. One note on transformations that come out of these tools: they are all in real coordinates (as opposed to voxel coordinates).

2.2.1 Linear registration

Anima provides linear registration (**animaPyramidalBMRegistration**) using a block-matching algorithm as presented in [1,2]. It may use BOBYQA or exhaustive optimization for finding matches (`--opt` parameter). Transformations between blocks may be translations, rigid or affine (experimental) transformations (`-t` parameter). The result global transform (writable in any ITK supported format) may be a translation, rigid or affine transformation. Both this executable and the next one use a local squared correlation coefficient as the default similarity measure between blocks (`--metric` parameter). Registration may be initialized from a previously computed transformation (`-i` option) or from intensity based PCA translation or rigid transformation (`-I` option). In addition, we have introduced new constrained affine transformations [13] (`--ot` option) enabling the quantification of growth factors of e.g. the brain on specific chosen directions, and the extraction of the nearest rigid or similarity transformation from an affine registration (`--out-rig` and `--out-sim` options) [14].

Example: this registers Floating.nii.gz on Reference.nii.gz using a global affine transform (`--ot` parameter), on a multi-resolution pyramid of 4 levels (`-p` parameter), but stopping at the one before last level (`-l` parameter). For matching, it uses only those blocks where the standard deviation is above 10.

```
animaPyramidalBMRegistration -r Reference.nii.gz -m Floating.nii.gz -o Floating_On_
↪Ref.nii.gz -O transform_aff.txt -s 10 -p 4 -l 1 --ot 2
```

2.2.2 Non linear registration

Anatomical registration

Non linear registration of anatomical images (**animaDenseSVFBMRegistration**) shares many parameters with linear registration. It however estimates a stationary velocity field between two images [3], modeling a dense non linear transformation between the images. The output is therefore a vector image containing the SVF. This executable implements the method proposed in [4]. It requires that the two input images have the same sizes and orientations. It is usually highly recommended to perform non linear registration after global rigid/affine registration.

Example: this registers Floating_aff.nii.gz on Reference.nii.gz on a multi-resolution pyramid of 3 levels (`-p` parameter), stopping at full resolution (`-l` parameter).

```
animaDenseSVFBMRegistration -r Reference.nii.gz -m Floating_aff.nii.gz -o Floating_On_
↪Ref.nii.gz -O transform_nl.nii.gz -p 3 -l 0
```

DTI registration

Non linear DTI registration (**animaDenseTensorSVFBMRegistration**) implements the same algorithms as in scalar images registration with finite strain tensor reorientation or preservation of principal direction (PPD). Parameters are the same except the similarity metrics which implement those proposed in [4] (tensor oriented generalized correlation). Note that the block variance may be much smaller for tensors and it is advised to set the minimal variance (`-s`) to 0 by default.

MCM registration

Non linear MCM (multi-compartment models) registration (**animaDenseMCMSVFBMRegistration**) implements the same algorithms as in scalar images registration with finite strain tensor reorientation or preservation of principal direction (PPD). Parameters are the same except similarity metrics which implement those proposed in [12] (MCM SSD and correlation surrogate). As for tensors the block variance may be much smaller than for scalar images and it is advised to set the minimal variance (`-s`) to 0 by default.

2.2.3 EPI artifacts correction

Eddy current distortion correction

Anima includes a simple, yet experimental, tool for Eddy current distortion correction **animaEddyCurrentCorrection**. It performs by registering linearly then non linearly each sub-volume of the EPI series to the first sub-volume. Each non linear transformation is computed only in the phase encoding direction as suggested by other publications. Please note that the Eddy current distortion correction tool does not include susceptibility distortion correction that is accounted for by the distortion correction tool in the next section.

Example: this will perform Eddy current distortion correction on DiffVolume4D.nrrd in the direction specified by `-d` (`-d 1` denotes the Y direction in voxel coordinates).

```
animaEddyCurrentCorrection -i DiffVolume4D.nrrd -o DiffVolume_Corrected.nrrd -d 1
```

Susceptibility distortion correction

Anima proposes two tools for EPI distortion correction of images:

- **animaDistortionCorrection** which implements the method proposed by Voss et al. [5]
- **animaBMDistortionCorrection** which implements the symmetric block-matching method proposed in [6]. It uses the same kind of parameters as previous block-matching registration algorithms.

From our experience, we found out it is best to use both for correcting for distortion: **animaDistortionCorrection** being used as the initialization for **animaBMDistortionCorrection**. Both tools use two 3D images with reversed phase encoding directions as their input, the direction that was used in the acquisition may be specified using the `-d` parameter (0 = X axis, 1 = Y axis, 2 = Z axis). The transformation found may then be applied to a whole 4D volume using our resampling tools.

Example: this first computes an initial correction field using Voss et al. method, with a smoothing sigma of 2 voxels (`-s` parameter). Then it starts from the initial transformation Init_Correction.nii.gz and computes a more precise BM_Correction.nii.gz dense transformation (note that contrary to previous algorithms, these transformations are not SVFs but dense displacement fields). It also outputs the average of AP and PA images after correction into BM_Correction.nii.gz.

```
animaDistortionCorrection -f AP_Image.nii.gz -b PA_Image.nii.gz -o Init_Correction.
↪nii.gz -s 2
animaBMDistortionCorrection -f AP_Image.nii.gz -b PA_Image.nii.gz -i Init_Correction.
↪nii.gz -o BM_Corrected_Image.nii.gz -O BM_Correction.nii.gz
```

2.2.4 Symmetry plane computation and constrained registration

In addition to traditional registration, we provide tools to compute and use the inter-hemispheric symmetry plane of an image [7,8]. This is based on two tools:

- **animaSymmetryPlane** [7] computes the symmetry transformation of an image (about its inter-hemispheric plane) and outputs both that transform (`-O` parameter) and a transformation that brings the image on its symmetry plane (`--out-realign-trsf`)
- **animaSymmetryConstrainedRegistration** implements constrained global rigid registration [8] utilizing two input symmetry plane transforms to restrict the search space.

Example: If one wants to register two images A.nii.gz and B.nii.gz, three steps will be necessary: realign A on its symmetry plane, realign B on its symmetry plane, and use both transformations as inputs to make a constrained registration of A and B. The output transformation brings the original B on the original A with a rigid transformation. The `-F` option activates a faster constrained registration but which may lose a little accuracy (see [8]).

```
animaSymmetryPlane -i A.nii.gz -o A_realign.nii.gz --out-realign-trsf A_sym.txt
animaSymmetryPlane -i B.nii.gz -o B_realign.nii.gz --out-realign-trsf B_sym.txt
animaSymmetryConstrainedRegistration -r A.nii.gz -m B.nii.gz --ref-sym A_sym.txt --
↪moving-sym B_sym.txt -F -o B_on_A.nii.gz -O B_on_A_rig.txt
```

2.2.5 Transformation tools (applying, arithmetic, jacobian)

EPI distortion correction

EPI distortion correction works in a slightly different way as other resampling tools. The tool provided is called **animaApplyDistortionCorrection**. It takes as inputs a 4D image with regular phase encoding direction (`-f` parameter) and optionally a 4D image with reversed phase encoding direction (for better correction, `-b` parameter). Then, using transformations coming from the previous tools, it corrects for distortion (if `-b` is provided the output will be the average of the two corrected images).

Example: this applies the previously obtained transormation to the whole DWI volume to correct its distortion.

```
animaApplyDistortionCorrection -f DWI_AP.nii.gz -t BM_Correction.nii.gz -o DWI_
↪Corrected.nii.gz
```

Constructing series of transformations descriptions

All other transform application tools require the input transformations to be given as an XML file which describes a series of transformations. It can take several option but the simple example is the following:

```
animaTransformSerieXmlGenerator -i transform_aff.txt -i transform_nl.nii.gz -o␣
↪transforms.xml
```

It creates the description of the two transformations (the specified order is the order in which they will be applied).

Applying a transformation to images

Applying a transformation requires the previous XML description file. Three tools are available:

- one for scalar images - **animaApplyTransformSerie**
- one for tensor images - **animaTensorApplyTransformSerie**
- one for multi-compartment model images [10] **animaMCMApplyTransformSerie**

All tools require a geometry image to tell in which space the resampling will take place (`-g` parameter). If the transformation series is globally linear, it may be applied to a gradient file of diffusion images. **animaApplyTransformSerie** now supports 3D and 4D images (in the latter case, the transformation is applied independently to each of the 3D sub-volumes). Diffusion model resamplers have an option to either apply finite strain re-orientation of the models or preservation of principal direction (PPD) re-orientation: `-P` activates PPD re-orientation, the default is finite strain.

Example: this applies the transforms in transforms.xml to resample Floating on Reference.

```
animaApplyTransformSerie -i Floating.nii.gz -g Reference.nii.gz -t transforms.xml -o␣
↪F_resampled.nii.gz
```

Applying a transformation to fibers or meshes

Anima also comes with a tool to apply transformations obtained from image registration to meshes or fibers. It accepts vtk, vtp and fds (our fiber format for medInria) files. This tool, named **animaFibersApplyTransformSerie**, works in the same way as animaApplyTransformSerie. The two main differences are the following:

- the input transformation is inverted by default as image transformations are encoded in Anima as the inverse of the underlying space transformation. This way, animaApplyTransformSerie and animaFibersApplyTransformSerie are similar in their uses. Use the `-I` option to invert the transformation series if necessary.
- There is no need for a geometry as this is specific to images

Computing the Jacobian of a transformation

A tool to compute the Jacobian or its determinant of a displacement field transformation is provided with the tool **animaDisplacementFieldJacobian**. If may handle SVF transformations using the `-S` option. More options for this tool are provided when using the `-h` option.

Example: this computes the Jacobian matrix of the input SVF after its exponentiation. The Jacobian matrix is stored as a 9 component vector image stored in row first.

```
animaDisplacementFieldJacobian -i inputField.nrrd -S -o dispFieldJacDeterminant.nrrd
```

Linear transformations arithmetic

We provide a tool named **animaLinearTransformArithmetic** to compose and perform log-Euclidean operations on linear transformations as proposed by Arisgny et al. [9]. The tool proposes regular composition (`-c`), addition (`-a`), subtraction (`-s`), multiplication by a constant (`-M`), division by a constant (-D) in the log-Euclidean framework.

Example: this performs the log-Euclidean addition of the two linear input transformations (in the ITK format) in the log-Euclidean framework.

```
animaLinearTransformArithmetic -i transform.txt -a addedTransform.txt -o␣
↪outputTransform.txt
```

Dense field transformations arithmetic

We provide a tool named **animaDenseTransformArithmetic** to compose and perform log-Euclidean operations on dense field (diffeomorphic) transformations as proposed by Arisgny et al. [11]. The tool proposes regular composition or BCH approximation to the composition of SVFs in the log-Euclidean framework (`-c`). It can also take the exponential of an SVF, dense diffeomorphism logarithm is the only operation not implemented yet.

2.2.6 References

1. Olivier Commowick, Nicolas Wiest-Daesslé, Sylvain Prima. *Block-Matching Strategies for Rigid Registration of Multimodal Medical Images*. 9th IEEE International Symposium on Biomedical Imaging (ISBI), pp. 700–703, 2012.
2. S. Ourselin, A. Roche, S. Prima and N. Ayache. *Block Matching: A General Framework to Improve Robustness of Rigid Registration of Medical Images*. Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery (MICCAI), volume 1935 of LNCS, pp. 557–566, 2000.
3. Olivier Commowick, Nicolas Wiest-Daesslé, Sylvain Prima. *Automated diffeomorphic registration of anatomical structures with rigid parts: application to dynamic cervical MRI*. 15th International Conference on Medical Image Computing and Computer Assisted Intervention, pp.163-70, 2012.
4. Ralph Suarez, Olivier Commowick, Sanjay Prabhu, Simon K. Warfield. *Automated delineation of white matter fiber tracts with a multiple region-of-interest approach*. NeuroImage, 59 (4), pp.3690-3700, 2012.
5. H.U. Voss, R. Watts, A.M. Ulugc, D. Ballona. *Fiber tracking in the cervical spine and inferior brain regions with reversed gradient diffusion tensor imaging*. Magnetic Resonance in Medicine, 24(3):231–239, 2006.
6. Renaud Hédouin, Olivier Commowick, Elise Bannier, Benoit Scherrer, Maxime Taquet, Simon Warfield, Christian Barillot. *Block-Matching Distortion Correction of Echo-Planar Images With Opposite Phase Encoding Directions*. IEEE Transactions on Medical Imaging, in press available online, 2017.
7. S. Prima, S. Ourselin, N. Ayache. *Computation of the Mid-Sagittal Plane in 3D Brain Images*. IEEE Transactions on Medical Imaging, 21(2):122-138, February 2002.
8. Sylvain Prima, Olivier Commowick. *Multimodal rigid-body registration of 3D brain images using bilateral symmetry*. Medical Imaging: Image Processing, SPIE, 8669, pp.866911, 2013.
9. V. Arsigny, O. Commowick, N. Ayache, X. Pennec. *A Fast and Log-Euclidean Polyaffine Framework for Locally Linear Registration*. Journal of Mathematical Imaging and Vision, 33(2):222-238, February 2009.
10. Renaud Hédouin, Olivier Commowick, Aymeric Stamm, Christian Barillot. *Interpolation and Averaging of Multi-Compartment Model Images*, 18th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 354-362, 2015.
11. V. Arsigny, O. Commowick, X. Pennec, N. Ayache. *A Log-Euclidean Framework for Statistics on Diffeomorphisms*, 9th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 924-931, 2006.
12. O. Commowick, R. Hédouin, E. Caruyer, C. Barillot. *L2 Similarity Metrics for Diffusion Multi-Compartment Model Images Registration*, 20th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 257-265, 2017.

13. A. Legouhy, O. Commowick, F. Rousseau, C. Barillot. *Anisotropic similarity, a constrained affine transformation: application to brain development analysis*, ISMRM, 2018.

14. A. Legouhy, O. Commowick, F. Rousseau, C. Barillot. *Unbiased Longitudinal Brain Atlas Creation Using Robust Linear Registration and Log-Euclidean Framework for Diffeomorphisms*, International Symposium on Biomedical Imaging, 2019.

2.3 Segmentation tools

This part describes segmentation features in Anima. It, as of now, includes multiple sclerosis segmentation tools and graph-cut segmentation.

2.3.1 Basic segmentation tools

We provide a set of basic tools related to segmentation that can be used easily.

Dice score computation

animaDiceMeasure computes the Dice score between two labeled segmentations. For binary images, it computes the Dice or Jaccard scores. For multi-labeled images, it either outputs the individual label scores or the total overlap score as in Klein et al. [1] (`-T` option).

Image thresholding

animaThrImage thresholds an image, using one or several thresholds.

Example: this thresholds the input Image.nii.gz to keep only voxels whose values are between 0.5 and 1.5.

```
animaThrImage -i Image.nii.gz -t 0.5 -u 1.5 -o BinaryImage.nii.gz
```

animaOtsuThrImage computes the Otsu automatic thresholding in several classes of an input image.

Majority voting label fusion

animaMajorityLabelVoting provides an implementation of majority voting: i.e. a fusion where each voxel is affected the most probable occurence of a set of labels. It takes as an input a list of segmentation images (`-i`: either as a 4D image or a text file with each line indicating the filename of a segmentation) and outputs the majority vote in an image given with `-o`.

Image holes filler

animaFillHoleImage provides a simple tool that fills small holes in segmentations that are not connected with the background.

Image masking

animaMaskImage simply takes an image and a binary mask and applies the mask to the input image.

Isosurface extraction

animaIsosurface takes as an input image and extracts its isosurface at the level given by the `-t` option. Decimation and smoothing of the resulting surface can also be controlled.

2.3.2 Graph cut segmentation

We provide an implementation of Lecoeur et al. graph cut segmentation algorithm [1], using the spectral gradient for multimodal images. It is implemented in the **animaGraphCut** tool.

2.3.3 Multiple sclerosis lesions segmentation

We provide an implementation of one of the multiple sclerosis (MS) segmentation algorithms developed in the Visages team [2,3]. This method is based on a tissue classification of multiple channel images and uses graph cut to delineate T2 hyper-intense lesions. It takes as an input three modalities among T1, T2, DP and FLAIR as well as a brain mask.

Example: this command line uses T1, T2 and FLAIR images as an input, as well as brain_mask.nii.gz as a skull-stripping mask and compute their GCEM segmentation.

```
animaGcStremMsLesionsSegmentation -m brain_mask.nii.gz -a 1 --rej 0.2 --min-fuzzy 1 --
↪max-fuzzy 2 --intT2 3 --intFLAIR 2 --rb --ml 3 -i T1.nii.gz -j T2.nii.gz -l FLAIR.
↪nii.gz --ini 2 -o lesion_seg.nii.gz --out-csf csf_seg.nii.gz --out-gm gm_seg.nii.gz␣
↪--out-wm wm_seg.nii.gz --out-gc gc_seg.nii.gz
```

The parameters are as follows:

- `-a` option specifies the type of algorithm (0 is STREM [2], 1 is GCEM [3]).
- `--rej` : percentage of outliers rejection
- `--min-fuzzy` : minimal value for fuzzy rules
- `--max-fuzzy` : maximal value for fuzzy rules
- `--ml` : minimal lesion size (mm3)
- `--ini` : initialization method (0: atlas, 1: based on DP, 2: based on FLAIR)
- `-o` : output lesion segmentation
- `--out-csf` : output CSF map
- `--out-gm` : output GM map
- `--out-wm` : output WM map

2.3.4 Segmentation validation tools

We now provide tools that were used for the validation of the MS segmentation challenge held in 2016 and that can be used for other tasks as well.

Dice measure

Anima comes with two tools for computing the Dice measure:

- **animaDiceMeasure** computes Dice scores between two label images (i.e. with each pixel having an integer label). It can either output the Dice scores for each label individually or compute the total overlap score as proposed by Klein et al. [5] (`-T` option)
- **animaFuzzyDiceMeasure** computes the generalized Dice score between fuzzy segmentations of a structure as proposed by Crum et al. [6]

Both tools have an option to output the Jaccard score instead of the Dice score (`-J` option), both scores being related by a monotonic function.

Detected components

animaDetectedComponents provides an evaluation tool that counts the number of connected components in a reference binary segmentation that are actually detected by a test segmentation. This does not mean that the components are very well segmented but rather that the test segmentation detect sufficiently the components. The output is a CSV file detailing for the reference image the number of connected components, their sizes and if they were detected by the test segmentation.

Segmentation performance analyzer

animaSegPerfAnalyzer includes all metrics that were computed for the MS segmentation challenge at MICCAI in 2016. This tool was used to compute the results available as supplementary material here. Please refer to our article [4] for more details on the measures.

2.3.5 References

1. Jérémy Lecoeur, Sean Patrick Morrissey, Jean-Christophe Ferré, Douglas Arnold, D. Louis Collins, Christian Barillot. *Multiple Sclerosis Lesions Segmentation using Spectral Gradient and Graph Cuts*. Medical Image Analysis on Multiple Sclerosis (validation and methodological issues), MICCAI workshop, 2008.
2. Daniel García-Lorenzo, Sylvain Prima, Douglas Arnold, Louis Collins, Christian Barillot. *Trimmed-likelihood estimation for focal lesions and tissue segmentation in multisequence MRI for multiple sclerosis*. IEEE Transactions on Medical Imaging, 30 (8), pp.1455-67, 2011.
3. Daniel García-Lorenzo, Jérémy Lecoeur, Douglas Arnold, D. Louis Collins, Christian Barillot. *Multiple Sclerosis lesion segmentation using an automatic multimodal Graph Cuts*. 12th International Conference on Medical Image Computing and Computer Assisted Intervention, LNCS 5762, pp.584-591, 2009.
4. O. Commowick et al. *Objective Evaluation of Multiple Sclerosis Lesion Segmentation using a Data Management and Processing Infrastructure*. Scientific Reports, 8(1), 2018
5. Klein, A, Andersson, J, Ardekani, BA, Ashburner, J, Avants, B, Chiang, M-C, Christensen, GE, Collins, DL, Gee, J, Hellier, P, Song, JH, Jenkinson, M, Lepage, C, Rueckert, D, Thompson, P, Vercauteren, T, Woods, RP, Mann, JJ, Parsey, RV. *Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration*. NeuroImage. 46(3): 786-802. 2009.
6. W.R. Crum, O. Camara and D.L.G. Hill. *Generalized Overlap Measures for Evaluation and Validation in Medical Image Analysis*. IEEE Transactions on Medical Imaging. 25(11):1451-1461. 2006.

2.4 Group comparison tools

2.4.1 Patient to group comparison

General patient to group comparison

Patient to group comparison tools provides ways to compare a single patient image (scalar, log-tensor fields, vector fields, ODFs,...) to a group (population) of control images. The most general executable for it is **animaPatientToGroupComparison**. It handles any vector image type that represents data in a vector space (e.g. scalar images, log-tensor images obtained from **animaLogTensors**, stacked multimodal data...). It implements the test proposed in [1] but without direction sampling of the data. It also handles dimensionality reduction through PCA on the control database (`-E` and `-e` options).

Example: this tests at each voxel testedImage against the list of controls (specified in listControls.txt: one line = one image to load). For speed reasons, it is highly desirable to use computationMask.nii.gz, which can be the intersection of brain masks of the controls and the tested image. It outputs p-values and z-scores of the test.

```
animaPatientToGroupComparison -i testedImage.nii.gz -I listControls.txt -o zScore.nii.
↪gz -O pValues.nii.gz -m computationMask.nii.gz
```

ODF patient to group comparison

animaPatientToGroupODFComparison implements the same test as in [1] but for ODF data. In addition to parameters of **animaPatientToGroupComparison**, the user has to specify a set of gradient directions (bvec or text file) on which the ODF will be sampled.

Non local patient to group comparison

Non-local patient to group comparison tools implement the method proposed in [2] for DTI images. These consist of three tools. The main one is **animaNLMeansPatientToGroupComparison** which performs the comparison itself by looking for patches around a given position, similarly to NL-Means denoising. It adds four options (`-d`, `-D`, `--dbcovave`, `--dbcovstd`) to perform the average and standard deviation tests on the test image. Those four options require images as an input. They are provided by two other tools:

- **animaLocalPatchMeanDistance** takes as an input the list of controls and provides two outputs: the one from `-o` is to be used for `-d` input of **animaNLMeansPatientToGroupComparison**, the one from `-O` is to be used for `-D` input of **animaNLMeansPatientToGroupComparison**)
- **animaLocalPatchCovarianceDistance** takes as an input the list of controls and provides two outputs: the one from `-o` is to be used for `--dbcovave` input of **animaNLMeansPatientToGroupComparison**, the one from `-O` is to be used for `--dbcovstd` input of **animaNLMeansPatientToGroupComparison**)

2.4.2 Group comparison tools

We provide an implementation of a population comparison tool proposed by Whitcher et al. [3] and used for neonates comparison in [4]. This test is based on the computation of inter- and intra-group distances computation and their use to derive a statistical test of groupies voxel differences. As such our implementation uses as an input vector images for which a L2 distance may be computed (such as log-vectors of tensors or scalar images) and outputs a map of p-values (not corrected for multiple comparisons (see next section).

Example: this computes the voxelwise Cramers tests p-values given its inputs. dataList.txt lists on each line the input files of all groups, firstGroup.txt and secondGroup.txt lists the indexes of patients in each group, starting with index 0.

```
animaCramersTest -l dataList.txt -m computationMask.nrrd -o outputPValues.nrrd -f␣
↪firstGroup.txt -s secondGroup.txt
```

2.4.3 Multiple comparisons correction

When doing the above mentioned tests, multiple comparisons are being made that need to be corrected for. **animaFDRCorrectPValues** implements FDR correction as presented by Benjamini and Hochberg [5]. It provides as an output thresholded p-values at q-value specified by the `-q` option.

2.4.4 Low memory tools

All preceding tools have their **animaLowMem** counterparts, as they usually require a lot of memory to run which may not be available on a single computer. These low memory tools include options to split the input images in every direction and process either a single sub-part of them or all of them in a sequential order.

animaMergeBlockImages can then use the output description file and a geometry image to rebuild the final result.

Warning: these low memory tools are much slower as they spend most of their time reading images. However, they can be run on a cluster if available.

2.4.5 References

1. Olivier Commowick, Adil Maarouf, Jean-Christophe Ferré, Jean-Philippe Ranjeva, Gilles Edan, Christian Barillot. *Diffusion MRI abnormalities detection with orientation distribution functions: A multiple sclerosis longitudinal study*. Medical Image Analysis, 22(1):114-123, 2015.
2. Olivier Commowick, Aymeric Stamm. *Non-local robust detection of DTI white matter differences with small databases*. 15th International Conference on Medical Image Computing and Computer Assisted Intervention, Oct 2012, Nice, France. 15 (Pt 3), pp.476-84, 2012, LNCS.
3. B. Whitcher, J.J. Wisco, N. Hadjikhani and D.S. Tuch. *Statistical group comparison of diffusion tensors via multivariate hypothesis testing*. Magnetic Resonance in Medicine, 57(6):1065-1074, June 2007.
4. O. Commowick, N. I. Weisenfeld, H. Als, G. B. McAnulty, S. Butler, L. Lightbody, R. M. Robertson and S. K. Warfield. *Evaluation of White Matter in Preterm Infants With Fetal Growth Restriction*, In Proceedings of the Workshop on Image Analysis for the Developing Brain, held in conjunction with MICCAI'09, September 2009.
5. Y. Benjamini and Y. Hochberg. *Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing*. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 57, No. 1 (1995), pp. 289-300.

2.5 Relaxometry tools

2.5.1 Estimation

T1 relaxation

We include an implementation of the DESPOT1 [1] algorithm for T1 relaxation time estimation: **animaT1RelaxometryEstimation**. It optionally can take a B1 inhomogeneity map as an input to correct for it (`-b` parameter). It outputs two images: T1 relaxation times (`-o` output) and M0 map (proportional to proton density, `-O` output).

T2 relaxation

We include two ways to estimate T2 relaxation times from T2 relaxometry sequences. Those two algorithms suppose acquisitions were made with an equal echo spacing that is specified to the algorithm. Those two tools are:

- **animaT2RelaxometryEstimation** estimates T2 relaxation using the regular exponential decay equation with a single T2 value. It may take an input T1 map for better estimation, and outputs M0 and T2 maps.
- **animaT2EPGRelaxometryEstimation** estimates T2 relaxation using the EPG algorithm with a single T2 value, to account for stimulated echoes due to B1 inhomogeneity [2]. It may take input T1 and B1 maps for better estimation, and outputs M0 and T2 maps.

Multi-compartment T2 estimation

We provide from Anima 3.0 new tools for multi-compartment T2 and myelin water fraction estimation. We also provide implementations of previous methods from the literature on multi-peak T2 estimation. While this second approach is appealing, we find it more stable to use multi-compartment T2 estimation rather than multi-peak and rather provide those for comparison purposes.

- **animaGammaMixtureT2RelaxometryEstimation** implements variable projection estimation of the parameters and weights of three T2 Gamma distributions using two different modes (toggled by the `-C` option) [3,4]: only middle T2 compartment mean estimation or all class mean parameters estimation. In both cases, all weights are deduced from the estimated parameters using variable projection
- **animaGMMT2RelaxometryEstimation** implements, for robustness to clinical acquisitions, a fixed parameter estimation of a Gaussian T2 mixture [7]. In this implementation, only the weights of the three T2 compartments are estimated, their PDFs being fixed according to prior knowledge on the tissues.
- **animaMultiT2RelaxometryEstimation** provides an implementation of several methods of the literature for multi-peak T2 estimation [2,5,6]. It provides several types of regularization: Tikhonov or non local regularization. Again these methods make the deduction of myelin water fraction more difficult and quite sensitive to the regularization.

2.5.2 MRI simulation

Several MR simulation tools are included in ANIMA, which simulate sequences from relaxation time maps. All of them are described in detail here.

2.5.3 References

1. Deoni, S. *High-resolution T1 mapping of the brain at 3T with driven equilibrium single pulse observation of T1 with high-speed incorporation of RF field inhomogeneities (DESPOT1-HIFI)*. J Magn Reson Imaging 26(4):1106–1111, 2007.
2. Kelvin J Layton, Mark Morelande, David Wright, Peter M Farrell, Bill Moran, Leigh Johnston. *Modelling and estimation of multicomponent T2 distributions*. IEEE transactions on medical imaging, 32(8):1423–34, 2013.
3. Sudhanya Chatterjee, Olivier Commowick, Onur Afacan, Simon K. Warfield, Christian Barillot. *Multi-Compartment Model of Brain Tissues from T2 Relaxometry MRI Using Gamma Distribution*. ISBI 2018.
4. Sudhanya Chatterjee, Olivier Commowick, Simon K. Warfield, Christian Barillot. *Multi-Compartment T2 Relaxometry Model Using Gamma Distribution Representations: A Framework for Quantitative Estimation of Brain Tissue Microstructures*. ISMRM 2017.
5. Prasloski et al. *Applications of stimulated echo correction to multicomponent T2 analysis*. MRM, 67(6):1803-1814, 2012.

6. Yoo et al. *Non-local spatial regularization of MRI T2 relaxation images for myelin water quantification*. MICCAI, pp 614-621, 2013.

7. S. Chatterjee, O. Commowick, O. Afacan, B. Combes, A. Kerbrat, S.K. Warfield, C. Barillot. *A 3-year follow-up study of enhancing and non enhancing multiple sclerosis lesions in MS patients with clinically isolated syndrom using a multi-compartment T2 relaxometry model*, ISMRM, 2018.

2.6 Denoising tools

2.6.1 Noise generation

animaNoiseGenerator adds Gaussian or Rician noise to a scalar image at a user specified SNR (an external reference input image is necessary to compute the average reference signal value). The -n may be used to output multiple samples of noisy images from the input.

Example: this adds Gaussian noise (relative to the average signal computed from refSignalImage.nii.gz) to Image.nii.gz. It will provide as outputs 10 images named OutputImage_??.nii.gz

```
animaNoiseGenerator -i Image.nii.gz -o OutImage.nii.gz -G -n 10 -r refSignalImage.nii.
↪gz
```

2.6.2 NL-Means denoising

NL-Means denoising is an implementation of the two papers from the team to denoise images corrupted by Gaussian [1] or Rician noise [2].

NL-Means for scalar images

animaNLMeans implements patch-based denoising for scalar images, considering it as a single image without a temporal dimension. It includes all parameters described in [1], including patch half size (real size being 2N+1), patch half search neighborhood, beta parameter for weighting, ... One can choose between [1] and [2] with the -W option.

Example: this performs denoising of Image.nii.gz with the default parameters, a beta parameter of 0.5 and assuming Rician noise.

```
animaNLMeans -i Image.nii.gz -o OutImage.nii.gz -W 1 -b 0.5
```

NL-Means for images with a temporal dimension

animaNLMeansTemporal works in the same way as **animaNLMeans** except that it considers the last dimension of the input image as a temporal dimension and therefore processes each temporal volume independently (useful e.g. for DWI images).

2.6.3 References

1. Pierrick Coupé, Pierre Yger, Sylvain Prima, Pierre Hellier, Charles Kervrann, Christian Barillot. *An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images*. IEEE Transactions on Medical Imaging. 27(4): 425-441. 2008.

2. Nicolas Wiest-Daesslé, Sylvain Prima, Pierrick Coupé, Sean Patrick Morrissey, Christian Barillot. *Rician noise removal by non-Local Means filtering for low signal-to-noise ratio MRI: applications to DT-MRI*. 11th International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 5242 (Pt 2), pp.171-179, 2008.

2.7 Basic image processing tools

2.7.1 Image arithmetic

animaImageArithmetic allows you to add, subtract, multiply or divide images either by a constant or another image. It relies on ITK filters.

Example: this computes Image1 - Image2 into the result image Diff.nii.gz

```
animaImageArithmetic -i Image1.nii.gz -s Image2.nii.gz -o Diff.nii.gz
```

2.7.2 Image averaging

animaAverageImages provides a tool to average multiple images (vector or scalar) into a single one.

Example: this computes the average from all images listed in listFiles.txt. listMasks.txt specifies a list of masks of interest applicable for each image.

```
animaAverageImages -i listFiles.txt -m listMasks.txt -o outputImage.nrrd
```

2.7.3 Image collapse

animaCollapseImage is a simple tool for converting a vector image (e.g. 3D vector file) into a (N+1)D image (e.g. a 4D scalar image).

2.7.4 Images concatenation

animaConcatenateImages takes several images and concatenates them into an (N+1)D image. Options allow to set the origin and spacing of the new dimension, as well as an optional base (N+1)D volume to which the additional images will be concatenated.

Example: this will concatenate Base, Addon1 and Addon2 into a single (N+1)D image on the last dimension.

```
animaConcatenateImages -i Addon1.nii.gz -i Addon2.nii.gz -b Base.nii.gz -o Data.nii.gz
```

2.7.5 Image resolution changer

animaImageResolutionChanger provides a way to change the image resolution (in the sense voxel resolution in mm). It automatically recomputes the origin and adapted image size from the old and new resolutions.

Example: this resamples the input image so that its voxel resolution is 2x2x2 mm.

```
animaImageResolutionChanger -i Image.nrrd -o Image_Result.nrrd -x 2 -y 2 -z 2
```

2.7.6 Image vectorization

animaVectorizeImages takes several ND scalar volumes and outputs a single ND vector image. It can also take a text file listing several volumes as the input.

Example: this will create a vector image with vector dimension 2, where the first dimension will contain Image1 and the second Image2.

```
animaVectorizeImages -i Image1.nii.gz -i Image2.nii.gz -o VectImage.nii.gz
```

2.7.7 Connected components

animaConnectedComponents computes connected components from a binary image. Optionally, it may take a minimal component size, under which the components will be removed.

2.7.8 Image conversion

animaConvertImage serves several roles: it can display information on an image, reorient it and write it to another file format.

Example: this reorients the image in the coronal plane, displays its information (`-I`) and saves the result in a NRRD file.

```
animaConvertImage -i Image.nii.gz -I -R CORONAL -o Image_Coronal.nrrd
```

2.7.9 Image cropping

animaCropImage extracts a sub-volume from an image using the ITK ExtractImageFilter. The lower case arguments(x<xindex>, y<yindex>, z<zindex>, t<tindex>) are the starting indexes of the input region to keep. The default value is 0. The upper case arguments (X<xsize>, Y<ysize>, Z<zsize>, T<tsize>) are the sizes of the input region to keep. The default value is the largest possible size given the corresponding indexes.

If you give arguments size of zero the corresponding dimension will be collapsed.

Example: for a 4D image 4x4x4x4, the arguments `--xindex 1 --zindex 1 --zsize 2 --tindex 3 --tsize 0` will result in an image 3x4x2 where the x dim corresponds to [1,2,3] of the input, y[0,3], zindex[1,2] and tindex is collapsed, only the last sequence has been kept.

2.7.10 Image smoothing

animaImageSmoother simply applies Gaussian smoothing with a specific sigma value to an image using Young - Van Vliet's recursive smoothing filter implemented in ITK [1].

2.7.11 Morphological operations

animaMorphologicalOperations computes usual morphological operations (erosion, dilation, opening, closure), with a specified radius expressed in millimeters.

2.7.12 References

1. Irina Vidal-Migallón, Olivier Commowick, Xavier Pennec, Julien Dauguet, Tom Vercauteren. *GPU & CPU implementation of Young - Van Vliet's Recursive Gaussian Smoothing Filter*. Insight Journal (ITK), 2013, pp.16

CHAPTER 3

Scripts documentation

And finally we provide a documentation of the main scripts available in Anima scripts that basically use Anima to make fancy things like:

- *Atlasing*
- *Diffusion imaging*
- *Relaxometry*
- *Segmentation* (brain extraction and multi-atlas segmentation)

3.1 Atlasing scripts

This section describes atlasing scripts of Anima scripts. These scripts are special in Anima scripts as they use shell and the OAR job submission system.

3.1.1 Overall goal

Two scripts are available here that have to be run on a OAR cluster to work properly. They follow a modified Guimond et al. [1] method to compute an average unbiased atlas from respectively anatomical or diffusion tensor images.

The method proceeds iteratively by registering the images onto a current reference to compute the reference at the next iteration. The main modifications with respect to the original Guimond et al. method is to use the log-Euclidean framework on diffeomorphisms [2] to compute the average transformations.

The two scripts then differ on the registration method used: anatomical image non linear registration [3] or DTI non linear registration [4].

3.1.2 Data structure for atlas construction

The datasets are supposed to be named correctly and placed in the right folders so that the atlasing scripts will find them. Data should be organized like this:

```
Main folder
+-- Images
|   +-- Prefix_1.nii.gz
|   +-- Prefix_2.nii.gz
|   +-- Prefix_3.nii.gz
|   .
|   .
|   .
+-- Masks
|   +-- Mask_1.nii.gz
|   +-- Mask_2.nii.gz
|   +-- Mask_3.nii.gz
|   .
|   .
|   .
```

The main folder can be located anywhere, the prefix of the images names (i.e. **Prefix**) and their folder name (i.e. **Images**) are up to the user (but should not include spaces, special characters or accentuated characters by precaution). The images folder contains the images from which the atlas will be built (anatomical or DTI depending on the script), all suffixed from 1 to N without losing contiguity.

The Masks folder and the subsequent names should be as shown here. They concern however only the anatomical images script for which they provide the mask of interest for each image that will be considered for average image construction.

3.1.3 Atlasing scripts

With this folder structure, two scripts are available that follow the same idea: **animaBuildAnatomicalAtlas.py** for anatomical atlas creation and animaBuildDTIAtlas.py** for DTI atlas creation. Both should be run from an OAR frontend machine as they start many jobs (N per iteration where N is the number of images, times a total of M iterations).

To run them, use one of the commands that follow after going to the main folder in the terminal (depending if you are making a DTI atlas or anatomical one):

```
~/Anima-Scripts-Public/atlasing/anatomical/animaBuildAnatomicalAtlas.py -p Images/
↪Prefix -n <N> -i <M> -c <nCores>
~/Anima-Scripts-Public/atlasing/dti/animaBuildDTIAtlas.py -p Images/Prefix -n <N> -i
↪<M> -c <nCores>
```

This runs an atlas construction over N images, performing M iterations of atlas creation and asking for nCores cores on each computer of the cluster for each job.

Results are provided in the main folder in the form of an average image (**averageForm<M>.nii.gz** or **averageDTI<M>.nii.gz** depending on the script). Transformations bringing each individual image on the atlas can be found in the **residualDir** subfolder added by the script:

- *Prefix_<i>_linear_tr.txt*: affine transform from image i toward **averageForm<M-1>.nii.gz**
- *Prefix_<i>_nonlinear_tr.nrrd*: non linear transform from affinely transformed image i toward **averageForm<M-1>.nii.gz**
- *sumNonlinear_inv_tr.nrrd*: average transformation whose inverse should be applied to get transformed images onto **averageForm<M>.nii.gz**

3.1.4 Online atlasing (iterative centroid)

Online atlasing as explained in [6] works in a different way as the previous scripts. Assuming you hav an atlas output products from the previous scripts, the iterative centroid algorithm adds additional images to form an atlas without having to create it from the ground up. Its call works in the same way as the previous ones:

```
~/Anima-Scripts-Public/atlasing/anatomical_iterative_centroid/
↪animaBuildAnatomicalICAtlas.py -p Images/Prefix -s <startPoint> -n <N> -c <nCores>
```

The *-s* option gives the number of images already in the atlas. The script will then use the original atlas and additional images up to N to compute the new atlas. It assumes the same data structure as the previous scripts

3.1.5 Longitudinal atlasing

A longitudinal atlas is constituted of a set of 3D atlases each representing an average model at a given age. In this case, the 3D atlases are computed up to a rigid transformation accounting both for global and local transformation in the unbiasing step.

We provide in the following sub-sections the necessary scripts to compute longitudinal atlases as detailed in [5], strongly based on the previous scripts. A preliminary weighting step is necessary such that, in the creation of an atlas at time *t*, more importance is given to subjects closer in age to *t*.

Weighting

This script takes as an input all of the following:

- List of all images paths in a txt file (i.e. image.txt). One path per line
- List of associated ages in an other txt file (i.e. age.txt)
- List of desired atlas ages in an other txt file (i.e. atlasAge.txt)

```
~/Anima-Scripts-Public/atlasing/longitudinal_preparation/
↪animaComputeLongitudinalAtlasWeights.py -a age.txt -i image.txt -o outputFolder -n␣
↪n -A atlasAge.txt -p Images/Prefix
```

This script call runs the preparation of everything needed to compute a 4D atlas composed of a set of 3D atlases representatives of ages contained in atlasAge.txt, made using about *n* subjects. It also prepares the folder structure to compute the different atlases in the following step.

```
output folder
+-- atlas_1
.
.
.
+-- atlas_i
|  +-- Images
|  |  +-- Prefix_1.nii.gz
|  |  .
|  |  .
|  |  .
|  +-- weights.txt
.
.
.
```

Atlasing scripts with longitudinal parameters

After the preparation step, to compute each sub-atlas i, simply run one of the atlasing scripts with the appropriate options:

```
cd outputFolder/atlas_i
~/Anima-Scripts-Public/atlasing/anatomical/animaBuildAnatomicalAtlas.py -p Images/
↪Prefix -n <N> -i <M> -c <nCores> --rigid -w weights.txt -b 2
```

In the previous line, **animaBuildAnatomicalAtlas** may be replaced by **animaBuildDTIAtlas** to compute a DTI longitudinal atlas.

3.1.6 References

1. A. Guimond, J. Meunier, J.P. Thirion. *Average brain models: A convergence study*, Computer Vision and Image Understanding, 77(2):192-210, 2000.
2. V. Arsigny, O. Commowick, X. Pennec, N. Ayache. *A Log-Euclidean Framework for Statistics on Diffeomorphisms*, 9th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 924-931, 2006.
3. Olivier Commowick, Nicolas Wiest-Daesslé, Sylvain Prima. *Automated diffeomorphic registration of anatomical structures with rigid parts: application to dynamic cervical MRI*. 15th International Conference on Medical Image Computing and Computer Assisted Intervention, pp.163-70, 2012.
4. Ralph Suarez, Olivier Commowick, Sanjay Prabhu, Simon K. Warfield. *Automated delineation of white matter fiber tracts with a multiple region-of-interest approach*. NeuroImage, 59 (4), pp.3690-3700, 2012.
5. Antoine Legouhy, Olivier Commowick, François Rousseau, Christian Barillot. *Unbiased Longitudinal Brain Atlas Creation Using Robust Linear Registration and Log-Euclidean Framework for Diffeomorphisms*, International Symposium on Biomedical Imaging, 2019.
6. Antoine Legouhy, Olivier Commowick, François Rousseau, Christian Barillot. *Online Atlasing Using an Iterative Centroid*, MICCAI, 2019.

3.2 Diffusion imaging scripts

This section describes diffusion imaging preprocessing and model estimation scripts of Anima scripts.

3.2.1 Diffusion images preprocessing script

This script combines several preprocessing steps to prepare data with the following steps (in that order):

- if -D option is given, correct gradients to match requirements from Anima (in real space i.e. the scanner space)
- Eddy current correction and motion correction using the experimental tool from Anima **animaEddyCurrentCorrection**
- distortion correction using the method proposed in [1]
- reorientation of the DWI volume to be axial on the z-axis and have no reversed axis
- Denoising using the NL-Means method [2]
- Brain masking using the *brain extraction script*
- DTI estimation using the **animaDTIEstimator** tool

Some of these steps may be discarded using the `--no-\*` options available in the script help. The brain masking step is performed either on a provided T1 image or on the DWI first sub-volume. For distortion correction, the reversed PED image has to be provided with the `-r` option and the direction of the PED with the `-d` option. If no reversed PED image, and the T1 is, distortion will be corrected by a simple B0 to T1 non linear registration.

Warning: gradients reworking is known to work only with Siemens acquisitions, not tested on other scanners.

Example:

```
~/Anima-Scripts-Public/diffusion/animaDiffusionImagePreprocessing.py -b Diff.bval -D␣
↪Dicom/* -r B0_PA.nii.gz -d 1 -t T1.nii.gz -i Diff.nii.gz
```

Results of this scripts are:

- **Diff_preprocessed.nrrd**: preprocessed diffusion 4D image
- **Diff_preprocessed.bvec**: preprocessed diffusion gradient vectors
- **Diff_brainMask.nrrd**: diffusion brain mask
- **Diff_Tensors.nrrd**: estimated tensors
- **Diff_Tensors_B0.nrrd**: estimated tensors B0 image
- **Diff_Tensors_NoiseVariance.nrrd**: estimated noise variance from the tensor estimation (actually includes noise and model underfit error)

3.2.2 Multi-compartment models estimation script

This script is more experimental as the multi-compartment model estimation tool in Anima [3] is currently undergoing quite some changes. However, it works well and we have a script that uses preferably the results of the diffusion preprocessing script as an input.

It mainly has two modes: one for HCP-like datasets that have high quality acquisitions and enable less constrained models, and one for regular clinical data with less estimation demanding models. Several options are still available:

- **-t**: choose the model type (as explained in *diffusion documentation*
- **-n**: maximal number of anisotropic compartments (ideally choose a number in between 1 and 3)
- **–hcp**: add some additional compartments like stationary water to handle specific aspects of HCP data
- **–no-model-simplification** and **-S** control model selection / averaging option. If none are set, the script will by default follow the method proposed in [4]: compute all models from 0 to N compartments and "average" them according to their likelihood (according to the AIC criterion). If **-S** is set, a faster model selection done at the stage of the stick model estimation is performed. If **–no-model-simplification** is set, a pure N model estimation is done without model selection.

Example:

```
~/Anima-Scripts-Public/diffusion/animaMultiCompartmentModelEstimation.py -t tensor -n␣
↪3 -i Diff_preprocessed.nrrd -g Diff_preprocessed.bvec -b Diff.bval -m Diff_
↪brainMask.nrrd
```

3.2.3 References

1. Renaud Hédouin, Olivier Commowick, Elise Bannier, Benoit Scherrer, Maxime Taquet, Simon Warfield, Christian Barillot. *Block-Matching Distortion Correction of Echo-Planar Images With Opposite Phase Encoding Directions*. IEEE Transactions on Medical Imaging, in press available online, 2017.

2. Nicolas Wiest-Daesslé, Sylvain Prima, Pierrick Coupé, Sean Patrick Morrissey, Christian Barillot. *Rician noise removal by non-Local Means filtering for low signal-to-noise ratio MRI: applications to DT-MRI*. 11th International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 5242 (Pt 2), pp.171-179, 2008.

3. Aymeric Stamm, Olivier Commowick, Simon K. Warfield, Simone Vantini. *Comprehensive Maximum Likelihood Estimation of Diffusion Compartment Models Towards Reliable Mapping of Brain Microstructure*. 19th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2016.

4. Aymeric Stamm, Olivier Commowick, Patrick Pérez, Christian Barillot. *Fast Identification of Optimal Fascicle Configurations from Standard Clinical Diffusion MRI Using Akaike Information Criterion*. IEEE International Symposium on Biomedical Imaging, 2014.

3.3 Relaxometry scripts

This section describes Anima scripts for relaxometry processing.

3.3.1 T2 relaxometry script

This script combines several processing steps to compute from a 4D T2 relaxomaetry sequence (plus possible additional images) T2 maps or multi-T2 maps following [1]. This script performs the following preprocessing:

- brain masking either from the provided T1-weighted high resolution image provided using the `-m` option or by default the first volume of the input 4D image
- computation of the mono T2 maps with B1 correction using **animaT2EPGRelaxometryEstimation** [2]
- computation of multi-T2 weight maps (modeled as three Gaussians as in [1]) as well as the myelin water fraction using **animaGMMT2RelaxometryEstimation** [1]

Some of these steps may be discarded using the `--no-\*` options available in the script help. If provided, the EPG algorithms may use an external T1 image provided by the user with the `-t` option. Results of this script are specified using the `-o` and `-g` options.

3.3.2 References

1. S. Chatterjee, O. Commowick, O. Afacan, B. Combes, A. Kerbrat, S.K. Warfield, C. Barillot. *A 3-year follow-up study of enhancing and non enhancing multiple sclerosis lesions in MS patients with clinically isolated syndrom using a multi-compartment T2 relaxometry model*, ISMRM, 2018.

2. Kelvin J Layton, Mark Morelande, David Wright, Peter M Farrell, Bill Moran, Leigh Johnston. *Modelling and estimation of multicomponent T2 distributions*. IEEE transactions on medical imaging, 32(8):1423–34, 2013.

3.4 Segmentation scripts

This section describes Anima scripts for segmentation. For now, we have atlas-based intracranial extraction and multi-atlas segmentation.

3.4.1 Atlas-based intracranial mask extraction

This requires an additional atlas image for which the intracranial mask is known. These data are provided in the Anima scripts data.

The script simply takes as an input the image or images to be brain-extracted and performs a series of registrations to bring the atlas on the image. Finally the brain mask is applied to the image. For each brain-extracted image, two images are produced: `image_brainMask.nrrd` and `image_masked.nrrd`. The first one is the binary mask of the brain, and the second one the image with only the brain kept. T1 images are the ones that should work best as the atlas proposed is in the T1 modality but since the registration algorithm uses an adapted similarity measure, other MRI modalities should be fine as well as long as their field of view is similar to the one of the atlas.

Example:

```
~/Anima-Scripts-Public/brain_extraction/animaAtlasBasedBrainExtraction.py -i T1Image.
↪nrrd
```

3.4.2 Multi-atlas segmentation

As for the *atlasing* scripts, this script requires to be run on a cluster with an OAR scheduler. It provides a basic multi-atlas segmentation, i.e. does the following:

- registers a set of images with known segmentations (atlases) onto an image to be delineated
- applies transformations to known segmentations
- fuses the transformed segmentations using majority voting (**animaMajorityLabelVoting** in *segmentation tools*)

Several options are available:

- `-i`: anatomical image to be delineated
- `-a`: list of atlas anatomical images i.e. a text file with a file name on each line
- `-s`: list of corresponding atlas segmentations i.e. a text file with a segmentation name on each line
- `-o`: output label image for the input anatomical image
- `-c`: optional number of cores for each job on the cluster

Example:

```
~/Anima-Scripts-Public/multi_atlas_segmentation/animaMultiAtlasSegmentation.py -i␣
↪T13D.nrrd -a listAtlasImages.txt -s listAtlasSegmentations.txt -o T13D_segmented.
↪nrrd
```

CHAPTER 4

Citing Anima or Anima scripts

So far, there is no white paper on Anima or Anima scripts. If you are using Anima for your research, please take the time to:

- cite the relevant papers referenced in the different pages of this documentation
- add a sentence in your acknowledgments section or footnote in the methods part of your paper referencing the RRID of Anima: RRID:SCR_017017 or Anima scripts: RRID:SCR_017072 (should be written like this to ensure it is recognized by search bots)
- if possible, add as well the main website address (eg. as a footnote): https://anima.irisa.fr/

Do not hesitate to contact us if you have any doubts anima-users@inria.fr. Thanks !

CHAPTER 5

Licensing

Anima is licensed under an Aferro GPL v3 license. More details are provided in the *licensing page*.

5.1 License

5.1.1 Contact

This software is distributed under the AGPL v3 or later license. If you wish to use the code under another license, contact us at anima-licensing@inria.fr