Baby Brain Toolkit

Fbrain ERC project: Computational Anatomy of Fetal Brain

November 7, 2014

Contents

1 Introduction

BTK stands for Baby Brain Toolkit. This toolkit is developed in the context of the Fbrain ERC project: "Computational Anatomy of Fetal Brain"[1]. Studies about brain maturation aim at providing a better understanding of brain development and links between brain changes and cognitive development. Such studies are of great interest for diagnosis help and clinical course of development and treatment of illnesses. Several teams have begun to make 3D maps of developing brain structures from children to young adults. However, working out the development of fetal and neonatal brain remains an open issue. This project aims at jumping over several theoretical and practical barriers and at going beyond the

[1] http://icube-miv.unistra.fr/fr/index.php/ERC_FBrain

formal description of the brain maturation thanks to the development of a realistic numerical model of brain aging.

1.1 Copyright

This software is governed by the CeCILL-B license under French law and abiding by the rules of distribution of free software. You can use, modify and/ or redistribute the software under the terms of the CeCILL-B license as circulated by CEA, CNRS and INRIA at the following URL: http://www.cecill.info.

As a counterpart to the access to the source code and rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors have only limited liability.

In this respect, the user's attention is drawn to the risks associated with loading, using, modifying and/or developing or reproducing the software by the user in light of its specific status of free software, that may mean that it is complicated to manipulate, and that also therefore means that it is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the software's suitability as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions as regards security.

Any researcher reporting results using BTK should acknowledge this software by citing the following article:

1. F. Rousseau, E. Oubel, J. Pontabry, M. Schweitzer, C. Studholme, M. Koob, J.-L. Dietemann: BTK: An Open-Source Toolkit for Fetal Brain MR Image Processing. Computer Methods and Programs in Biomedicine, pp. 65–73, Vol. 109, Num. 1, January 2013, also available on HAL.

1.2 Installation

1.2.1 Dependencies

Baby Brain Toolkit (BTK) needs the following packages[2]:

- CMake, Tclap, OpenMP, ANN, Doxygen and Git. These libraries can be installed using the following command line:
  - for debian-based distributions: `apt-get install build-essential libgl1-mesa-dev libxt-dev cmake cmake-curses-gui libtclap-dev libgomp1 libann-dev doxygen git-core`
  - for MacOSX using macports[3]: `port install cmake tclap libANN doxygen git`
- Insight Toolkit (ITK) version 4.6.1
  - Download the tar.gz archive of ITK v4.6.1
  - Extract the archive
  - Then open a terminal and type :

```
mkdir ITK-build
cd ITK-build
ccmake ../(your-ITK-Source-folder)/
```

This will bring up the CMake configuration screen. Press `[c]` for configure and then use `[t]` to toggle the advanced mode. Make the following changes:

```
BUILD_TESTING = OFF
CMAKE_BUILD_TYPE = Release
Module_ITKReview = ON
BUILD_DOCUMENTATION = OFF
```

[2] BTK has been tested under MacOSX 10.10 (Yosemite)
[3] When configuring in cmake, please use /opt/local/include for TCLAP_DIRECTORY

Then press [c] to configure and [g] to generate the make file. Finally, go to the build folder of ITK (ITK-build) and type `make` at the prompt to obtain the final build of ITK.

- Visualization ToolKit (VTK) version 5.10.1 [4]
  - Download the tar.gz archive of VTK v5.10.1
  - Extract the archive
  - Then open a terminal and type :

```
mkdir VTK-build
cd VTK-build
ccmake ../(your-VTK-Source-folder)/
```

This will bring up the CMake configuration screen. Press [c] for configure and then use [t] to toggle the advanced mode. Make the following changes:

```
BUILD_TESTING = OFF
BUILD_EXAMPLE = OFF
CMAKE_BUILD_TYPE = Release
VTK_USE_INFOVIS = ON
VTK_USE_RENDERING = ON
```

Then press [c] to configure and [g] to generate the make file. Finally, go to the build folder of VTK (VTK-build) and type `make` at the prompt to obtain the final build of VTK.

- Python, Numpy (Optional) These libraries can be installed using the following command line:
  - for debian-based distributions: `apt-get install python python-numpy`
  - for MacOSX, numpy can be easily installed using ScipySuperpack.

1.2.2 Download and compile the BTK sources

- Get the BTK sources: `git clone https://github.com/rousseau/fbrain.git`
- Then:

```
mkdir fbrain-build
cd fbrain-build
ccmake ../fbrain
```

Make the following changes on the CMake configuration screen :

```
ITK_DIR = (YOUR-ITK-BUILD-FOLDER)
VTK_DIR = (YOUR-VTK-BUILD-FOLDER)
CMAKE_BUILD_TYPE = Release
```

then :

```
make
```

If you want generate the doxygen documentation the BUILD_DOCUMENTATION option must be set to ON in CMake configuration :

```
BUILD_DOCUMENTATION = ON
```

Then for generating the documentation run :

```
make doc
```

Most of the programs of the BTK suite use the OpenMP library for multi-threading purpose[5]. The number of cores used can be tuned using the following command line (in this example, 4 cores will be used): `export OMP_NUM_THREADS=4`

[4] Please note that this version of VTK can be installed on Mac OSX using macports.
[5] Please note that the current Apple compiler does not fully support OpenMP.

2 The BTK pipelines

BTK allows to implement a pipeline for the processing of fetal images, i.e. the reconstruction of anatomical and diffusion data, and the final tractography, all expressed in the same local coordinate system. This processing can be summarized in the following steps: 1) image conversion, 2) anatomical image reconstruction, 3) recontruction of the diffusion sequence, 4) registration of diffusion to anatomical data and 5) tractography.

2.1 Image conversion

BTK supports and has been tested by using images in Nifti format. However, images are frequently available in DICOM format and an image conversion is required. This can be performed by using dcm2nii, Slicer or other softwares.

Let say that you have 3 (possibly orthogonal) anatomical images:

```
ana01.nii.gz
ana02.nii.gz
ana03.nii.gz
```

and one set of DWI images:

```
dwi.nii.gz
dwi.bvec
dwi.bval
```

Please check that images are not flipped.

NOTE: The set of three files .nii.gz, .bvec, and .bval used to describe a DW sequences are represented in BTK just by the basename ("dwi" in the previous example), and the sequences must be provided in this way to the different applications. This allows to have shorter command lines and the use of consistent filenames.

2.2 Anatomical image reconstruction

The anatomical image reconstruction pipeline is depicted in Figure 1. The corresponding command lines are shown in Figure 2. This pipeline consists in: 1) converting the data from DICOM to NIFTI, 2) manual masking of the brain (optional step which can improve the reconstruction results) and image cropping (based on the size of the masks), 3) image denoising, 4) motion correction and super-resolution, 5) reorientation of the image using landmarks placed using Slicer.

2.3 Reconstruction of the diffusion sequence

The processing pipeline described in this section is dedicated to DWI of the fetal brain (depicted in Figure 3). The corresponding command lines are shown in Figure 4. It is similar to the anatomical image pipeline but it includes processing tools dedicated to DWI: 1) data conversion using dcm2nii, 2) sequence normalization (the B0 image are registered together using affine transforms and then averaged to produce one B0 image), 3) B0 image extraction and manual masking using ITKSNAP, 4) motion correction and RBF interpolation to compute the reconstructed DW sequence, 5) reorientation of the image using landmarks placed using Slicer. This final step (sequence reorientation) is very important since it allows the user to visualize the DW data using consistent color coding schemes. Please see also Section 3.4.

2.4 Tractography

If you have followed the previous steps correctly, at this point you should have the reconstructed anatomical and diffusion data spatially aligned, and ready to perform the tractography. To do this, BTK provides **btkTractography** (Section 3.5).

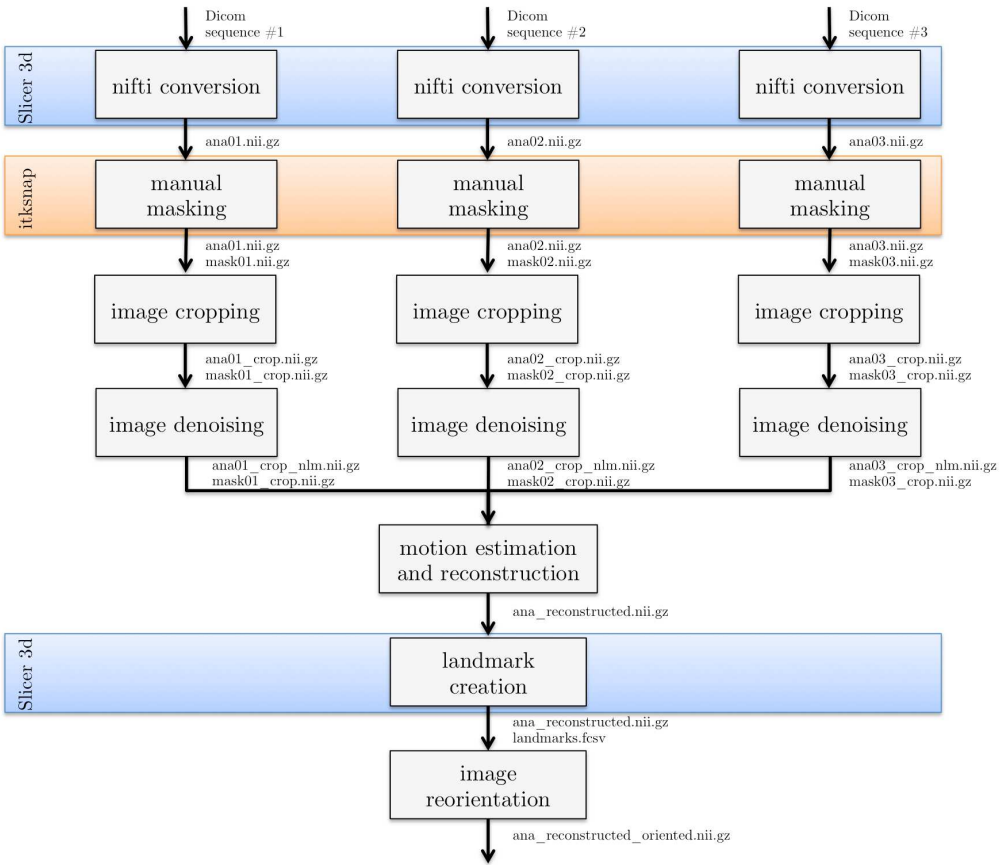


Figure 1: Overview of the processing pipeline for anatomical data in BTK. Slicer3D (www.slicer.org) is used to convert the DICOM data to NIFTI format and for the placement of landmarks on which is based the reorientation of the 3D reconstructed image. The use of Slicer for these steps allows the user to check the orientation consistency between the three anatomical images (axial, coronal, sagittal for instance). ITKSNAP (www.itksnap.org) is used to create a rough mask of the brain (this step is optional).

3 Applications

3.1 Denoising [2]

btkNLMDenoising This program applies a non-local mean filter to a 3D image for denoising purpose. Usage: `-i ana.nii.gz -o ana_nlm.nii.gz` . The best results are usually obtained by using a mask (or a padding value). This is an implementation of the method proposed by Coupé *et al.* in [2].

btkNLMDenoising4DImage This program applies a non-local mean filter to each 3D image of a 4D image, for denoising purpose. Usage: `-i dwi.nii.gz -o dwi_nlm.nii.gz` . The best results are usually obtained by using a mask (or a padding value).

3.2 Anatomical reconstruction

btkImageReconstruction This program allows to obtain a high-resolution image from a set of low-resolution images, typically axial, coronal, and sagittal acquisitions (similar to [7]).

Minimal usage: `btkImageReconstruction -i ana01.nii.gz ⋯ -i anaN.nii.gz -o ana_reconstructed.nii.gz --box`.

Recommended usage: `btkImageReconstruction -i ana01.nii.gz ⋯ -i anaN.nii.gz -m mask01.nii.gz ⋯ -m maskN.nii.gz -o ana_reconstructed.nii.gz --mask`.

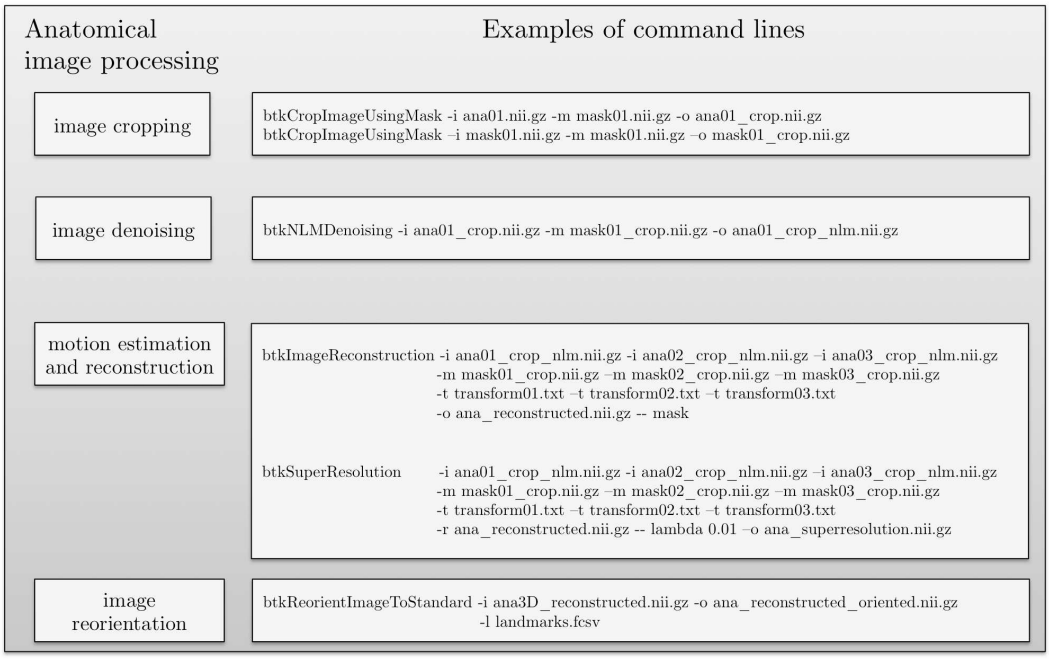


Figure 2: Command lines corresponding to the processing pipeline for anatomical data in BTK. On the left are shown the blocks used in the anatomical data pipeline (Figure 1) and on the right the correspond command lines.

The use of a mask provide better results since it allows an accurate estimation of the initial transform, and constrains the registration to the region of interest.

The full list of optional parameters of the method can be obtained by `btkImageReconstruction --help`

btkSuperResolution To be documented.

3.3 Anatomical segmentation

btkTissueSegmentation This program allows to obtain :

- A segmentation of the brain which separate pericerebral LCR and ventricules,
- A segmentation of the cortex inside of a region of interest defined from the border between brain and pericerebral LCR,

as described in [1].

Usage: `btkTissueSegmentation -i ana.nii.gz -s intracranian_volume.nii.gz -l brainstem_segmentation.nii.gz -l cerebellum_segmentation.nii.gz -o brain_segmentation.nii.gz -o cortex_segmentation.nii.gz`

btkLabelPropagation This program applies a patch-based label propagation algorithm for segmentation [8]. As a supervised method, it takes as input a textbook containing several anatomical images and the corresponding segmentation maps.

3.4 Reconstruction of DW sequences [4]

The reconstruction of diffusion-weighted (DW) sequences aims at obtaining a sequence corrected for fetal moving and eddy-current distortions. This can be performed in BTK by using the two following applications.

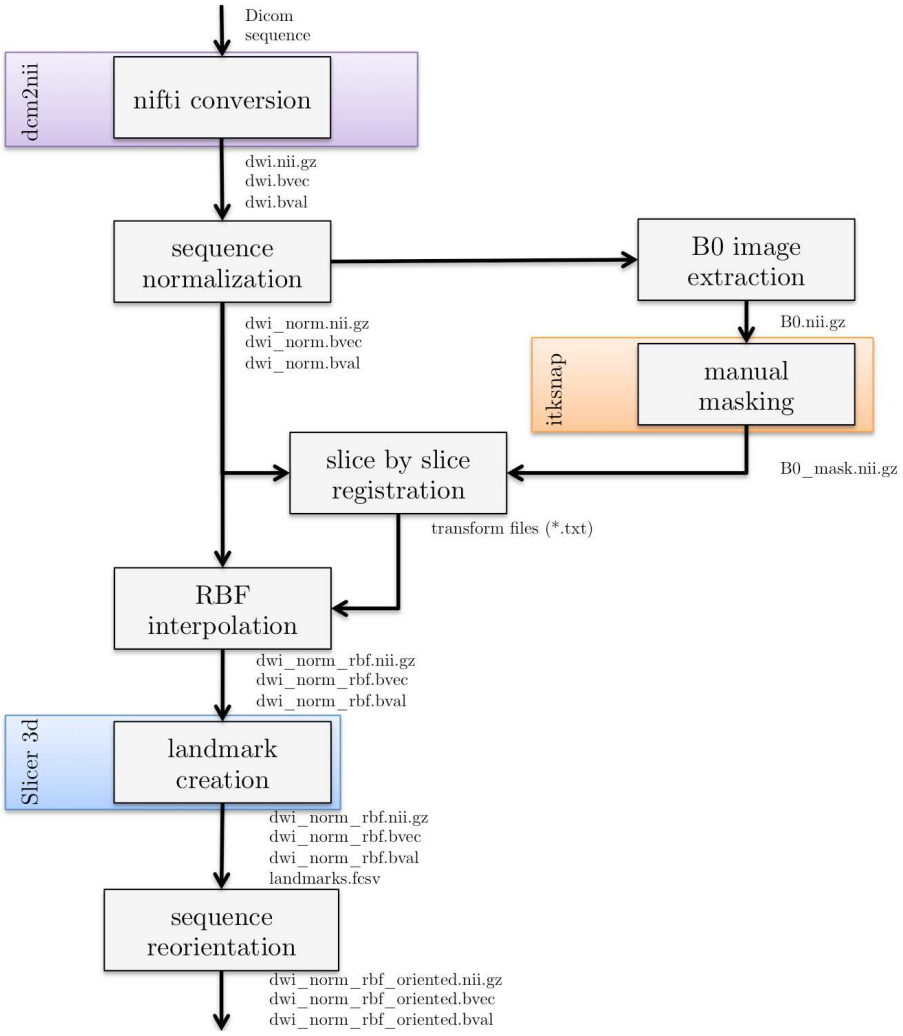


Figure 3: Overview of the processing pipeline for diffusion data in BTK. dcm2nii is used to convert the DICOM data to NIFTI format and Slicer for the placement of landmarks on which is based the reorientation of the 3D reconstructed image. ITKSNAP is used to create a rough mask of the brain.

btkGroupwiseS2SDistortionCorrection It performs a groupwise slice-by-slice registration of the image components of the sequence.

Minimal usage: `btkGroupwiseS2SDistortionCorrection -i dwi_normalized.nii.gz -o dwi_reconstructed.nii.gz -t transform_folder`.

-i dwi sequence noramlized (Utilities/btkSequenceNormalization).

-o dwi sequence reconstructed.

-t folder to save the transformation files

The slice-by-slice transform for a given DW image is saved as a set of N transforms in ITK format, with N the number of slices in the image.

NOTE: For slice registration, 20% of the samples are used for computing the image metric. As the region of interest of the slice can be small, this number of samples might be insufficient to compute the metric accurately. However, this percent of samples has been sufficient for the tested sequences.

btkRBFInterpolationS2S It performs an interpolation of scattered data generated from the application of slice-by-slice transforms. To this end, radial basis functions are used. By default, the

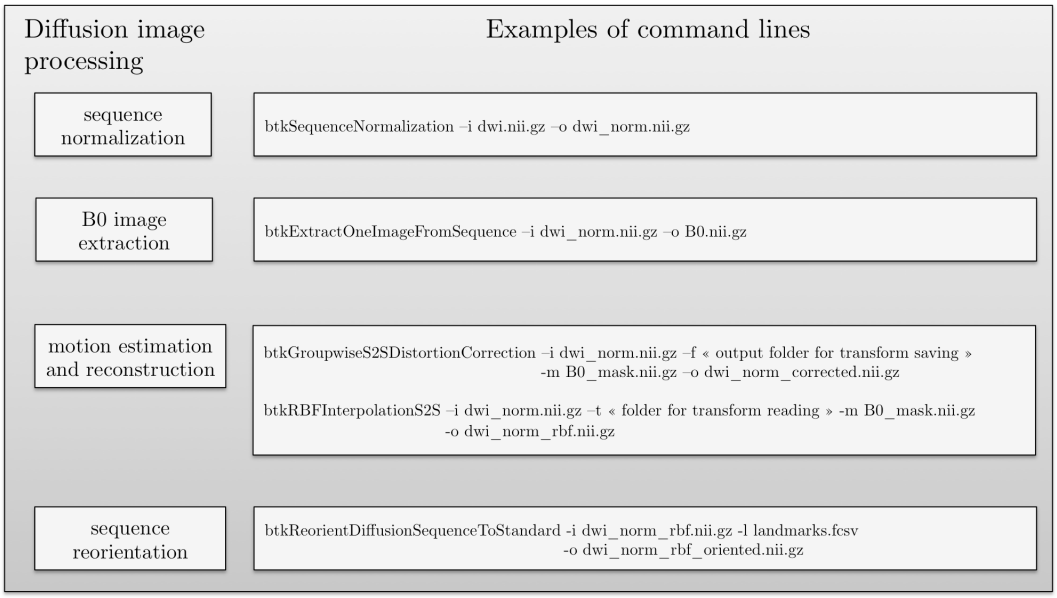


Figure 4: Command lines corresponding to the processing pipeline for diffusion data in BTK. On the left are shown the blocks used in the anatomical data pipeline (Figure 3) and on the right the correspond command lines.

output has isotropic voxels of size equal to the in-plane resolution of the input.

Minimal usage: `btkRBFInterpolationS2S -i dwi_normalized.nii.gz -m mask.nii.gz -o dwi_interpolated.nii.gz -t transformation_folder`.

-i dwi sequence normalized

-m mask for the B0 image

-o dwi sequence interpolated

-t folder with the transformations generated by `btkGroupwiseS2SDistortionCorrection`

3.5 Tractography [5]

Standard usage

Suppose you want to perform a tractography on a diffusion weighted MRI dataset. You should have a dwi image, the corresponding gradient vectors' coordinates, a mask of the brain white matter and a label image of the seeds. Assume this data is stored in files named respectively for instance `dwi.nii.gz`, `dwi.bval`, `dwi.bvec`, `mask.nii.gz` and `seeds.nii.gz`. The tractography is accomplished by the command below.

```
btkTractography -d dwi.nii.gz -m mask.nii.gz -r seeds.nii.gz
```

When the program terminates its task, the probability connection map and the fibers estimation are saved in files respectively named `map.nii.gz` and `fibers.vtk`. The connection map is a volume image of probability intensities (`i.e.` intensities between 0 and 1) with the same origin, orientation and spacing as the diffusion weighted image. The fibers are polygonal data of VTK library in world coordinates.

The file named `seeds.nii.gz` is a label map which is used to generate seeds for the algorithm. The spacing in millimeters between seeds can be adjust with the option `--seed_spacing`.

Before doing anything, be sure that the diffusion sequence has been normalized by using the utility `btkDiffusionSequenceNormalization`.

This program contains three different algorithms: a tensor streamline, an fODF streamline and a fODF particle filtering algorithm. The choice of the model to use is achieved by using the option `--tensor`, which set the diffusion model to tensor model. The choice of the algorithm is achieved by using the option `--streamline`, which set the algorithm to use to the streamline one.

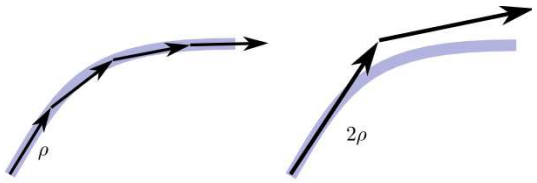


Figure 5: Effect of the step size option on a particle's trajectory. With a large step size (right), the particle may overshoot the trajectory of the ground truth.

Advanced usage

In addition to standard arguments of `btkTractography` program, there are some other parameters that let you to alter algorithm's behaviour. These options can be classified into three groups : model's options, constraints on trajectory and tracking options. The first group options allow you to tweak the model (for more details about fODF modeling, please refer to [3]). The second group options let you to control the fiber trajectory. These options provide prior informations to the algorithm. The last group options are dedicated to the tracking algorithm control.

Since the default parameters values may work in the most of cases, they are optional. A list is of optional features is avaible by using the command

`btkTractography --help` or `btkTractography -h`

and program's arguments are much more described below.

Model's order (fODF only)

The model's order (i.e. the spherical harmonics' order) can be specified by the option

`--model_order <order>` ,

for `order` $\in \{2, 4, 6, 8\}$. The default value is 4. For more details, please refer to [3].

Model's regularization (fODF only)

A Laplace-Beltrami regularization coefficient is used to assume a better estimation of the model. This coefficient can by manually modified by the option

`--model_regularization <coefficient>` ,

for `coefficient` $\in \mathbb{R}$. The default value is set as 0.006. For more details, please refer to [3].

Displacement step size

The displacement step size of a moving particle can ben adjusted as you want by using the option

`--step_size <length>` ,

where `length` $\in \mathbb{R}_+^*$. Note that this option is expressed in `mm`. The default value is fixed at 0.5 `mm`. By setting a big step size, the particles will move quickly. So the biger is the step, the faster the algorithm will finish, but as shown by Fig. 5, some informations may be missed and the particle's trajectories may overshoot the ground truth, resulting in a bad estimation.

Angular threshold

An angular threshold prevent a particle to return back. This option has to be expressed in radian and can bet set by

`--angular_threshold <angle>` ,

where `angle` $\in]0, 2\pi[$. The default value is set as a $\frac{\pi}{3}$ angle. As illustrated in two dimensions in Fig. 6, an angle threshold is used to define an allowed area for successive sampled directions. This can be seen as a global curvature parameters on trajectories. A small angle defines trajectories with a small curvature. This is a prior information on ground truth trajectory.

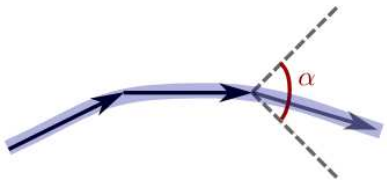


Figure 6: An angle threshold allows the algorithm to sample successive direction only in the cone defined by this angle. This illustration show the principle in two dimensions.

Figure 7: Local effect of rigidity parameter on a particle's trajectory. This parameter helps to "attract" the current displacement vector in the direction of the previous displacement vector of a particle. It correspond to a concentration paramter of a von Mise-Fisher density probability used in the prior density of the system. For instance, a rigidity of 0 leads to an equiprobable distribution, whereas a rigidty tending to infinity leads to a distribution focused on a point.

Trajectory interpolation method (streamline only)

When using streamline algorithm, by default the Runge-Kutta method at order 4 is used. The Euler's method can be used instead by setting the option `--no_RK4`.

Rigidity (particle filter only)

The rigidity option controls how much you want the particles to have straight trajectory. You can adjust it by

```
--curve_constraint <rigidity>  ,
```

where `rigidity` $\in \mathbb{R}_+^*$. The default value is fixed at 30. This value correspond to a concentration parameter of a von Mises-Fisher density probability used in the prior density of the system. As Fig. 7 illustrates locally in two dimensions, a high value leads to a straight trajectory.

Number of particles (particle filter only)

The number of particles in the system is set by the option

```
--number_of_particles <number>  ,
```

where `number` $\in \mathbb{N}^*$. By default, the algorithm will use 1000 particles. A poor number of particles leads to a short computation time and a poor estimation. A large number of particles leads to a long computation time and a good estimation. In general, the default number of particles is a good compromise between computation time and estimation.

Resampling threshold (particle filter only)

This option modify the resampling threshold of the system. When the number of effective particles in the system falls below this resampling threshold, the particles are resampled according a multinomial resampling. It can be adjust by

```
--resampling_threshold <percent>  ,
```

where `percent` $\in [0, 1]$ is the percent of minimal effective particles in the system. A low threshold value will result in an inefficient algorithm because the particles with low weight are not are not often eliminated. Conversely, a high threshold value leads to a bad estimation because the search space will not be explored enough.

3.6 Atlas construction [6]

There are two steps for atlas construction: filling the data script and building the atlas using the scripts. The following sections describe how you can do it.

Filling the data script

The data script (*btkAtlasData.py*) is used to give information to the executed script about the directories, the algorithms' options, which image modalities are used, etc. It has to be customized according to the data you have.

The atlas construction scripts take as input series of images (such as T2, label images, etc.), corresponding to the patients of the dataset. The input files should be named as *patientIdentifier_modalityIdentifier.nii.gz* and placed in the specified directory for the modality *modalityIdentifier*. There are basically two use cases: 1) you do not have any label image or 2) you have a label image.

If you are working without labels, and for instance you only have T2 image, you have to add the following piece of code in the modalities part of the script.

```
# T2 modality
# Identifier of the modality is T2 (input will be patientIdentifier_T2.nii.gz)
ModalityImage = 'T2'
# Initialization
modalities[ModalityImage] = {}
# T2 images will be used during registration
modalities[ModalityImage][UseInRegistration] = True
# T2 images will be used during regression
modalities[ModalityImage][UseInRegression]   = True
# The weight of T2 images is 1
modalities[ModalityImage][ModalityWeight]    = 1
# The path of T2 images is /path/to/data/T2
modalities[ModalityImage][ModalityDataPath]  = dataPath + '/T2'
# T2 is not a label
modalities[ModalityImage][IsTissueMap]       = False
# T2 is not a label, this field does not matter
modalities[ModalityImage][TissueLabel]       = 0
```

If you are working with labels, and for instance you have T2 images and labels images named *patientIdentifier_Tissues.nii.gz* in the data directory */path/to/data/Tissues*. Assume the label images contains 5 labels and you want to use only 2 labels (GM and WM). So you have to add the following piece of code in the modalities part of the script.

```
# T2 modality
# Identifier of the modality is T2 (input will be patientIdentifier_T2.nii.gz)
ModalityImage = 'T2'
# Initialization
modalities[ModalityImage] = {}
# T2 images will be used during registration
modalities[ModalityImage][UseInRegistration] = True
# T2 images will be used during regression
modalities[ModalityImage][UseInRegression]   = True
# The weight of T2 images is 1
modalities[ModalityImage][ModalityWeight]    = 1
```

```
# The path of T2 images is /path/to/data/T2
modalities[ModalityImage][ModalityDataPath]  = dataPath + '/T2'
# T2 is not a label
modalities[ModalityImage][IsTissueMap]       = False
# T2 is not a label, this field does not matter
modalities[ModalityImage][TissueLabel]       = 0

# GM modality
# Identifier of the modality is GM (input will be patientIdentifier_GM.nii.gz)
ModalityImage = 'GM'
# Initialization
modalities[ModalityImage] = {}
# GM images will be used during registration
modalities[ModalityImage][UseInRegistration] = True
# GM images will be used during regression
modalities[ModalityImage][UseInRegression]   = True
# The weight of GM images is 0.5
modalities[ModalityImage][ModalityWeight]    = 0.5
# The path of GM images is /path/to/data/GM
modalities[ModalityImage][ModalityDataPath]  = dataPath + '/GM'
# GM is a label
modalities[ModalityImage][IsTissueMap]       = True
# GM is the label 1
modalities[ModalityImage][TissueLabel]       = 1

# WM modality
# Identifier of the modality is WM (input will be patientIdentifier_WM.nii.gz)
ModalityImage = 'WM'
# Initialization
modalities[ModalityImage] = {}
# WM images will be used during registration
modalities[ModalityImage][UseInRegistration] = True
# WM images will be used during regression
modalities[ModalityImage][UseInRegression]   = True
# The weight of WM images is 0.5
modalities[ModalityImage][ModalityWeight]    = 0.5
# The path of WM images is /path/to/data/WM
modalities[ModalityImage][ModalityDataPath]  = dataPath + '/WM'
# WM is a label
modalities[ModalityImage][IsTissueMap]       = True
# WM is the label 3
modalities[ModalityImage][TissueLabel]       = 3
```

Building atlas

Once the data script *Scripts/btkAtlasData.py* has been copied in your working directory and filled according to the dataset and the algorithms' options (see previous section for some help), the atlas has to be built using dedicated scripts. First, the directory where the data script is have to be added in the python path environnement variable. For instance, in a terminal, the following command add the current directory in the python path environnement variable:

```
export PYTHONPATH = $PYTHONPATH:‘pwd‘  .
```

Finaly, the atlas construction can be achieved by calling directly the following python scripts.

btkPrepareAtlasData This optional script will split the tissue label map (if any and named *patient-name_Tissues.nii.gz*) into separate tissue probability maps. Tissue name and label are controlled

by setting up the script *Scripts/btkAtlasData.py*.

Usage: `btkPrepareAtlasData.py`

btkCreateTemplate.py This script will use the informations contained in the data script to produce a template normalization as output.

Usage: `btkCreateTemplate.py`

btkCreateLongitudinalAtlas.py This script will use the informations contained in the data script and the output of the script *btkCreateTemplate.py* to produce a longitudinal atlas over gestionnal ages (in weeks) as output. So, the script *btkCreateTemplate.py* must be executed before using this script.

Usage: `btkCreateLongitudinalAtlas.py`

3.7 Mask Extraction

Suppose you want to extract automicaly masks of your images. First of all you should have some Atlas of foetal brain for different ages (in week). In Addition you should know age of every image you want to process. The pipeline will registrate the corresponding age template with the current image, finaly you will have a new template succefully registrate on you image and you can use it as a mask. The mask extraction can be achieved by calling directly the first following python script.

btkMaskExtraction.py This script will contain all your images, you must edit it ! It contain examples. This script is the only one you should execute, it will automaticaly search and execute the second one.

Usage: `btkMaskExtraction.py`

btkMaskExtractionProcess.py This script will process the pipeline, you must edit the images_directory and template_directory. Note that your image directory should respect a hierarchy (written in the script).

Usage: Should be run by btkMaskExtraction.py or for advance user you can call like this : `btkMaskExtractionProcess.py subject exam age image01 image02 ...`

3.8 Fiber tracts Clustering

Fiber tracts clustering is used to map the well-known fiber bundles of white matter. This can be performed using the following applications:

btkProbabilisticSegmentationMapBasedClustering As supervised method, this program takes as input a probabilistic segmentation MAP of brain tissue which is an 4D image in Nifti format.

Usage: `btkProbabilisticSegmentationMapBasedClustering -b fiber_tractography.vtk -p 4D_prob_segmentation_map.nii.gz -o fiber_clustering.vtk`

Fig.8 shows an example of clustering results obtained using the method above.

btkDistanceFibersKMeansApproximationClustering As unsupervised method, this program makes it possible to cluster the fibers according to their anatomical regions using a distance metric based on the similarity between fibers.

Usage: `btkDistanceFibersKMeansApproximationClustering -b fiber_tractography.vtk -d distance_measure -c number_of_clusters -o fiber_clustering.vtk`

The full list of optional parameters of this method can be obtained by:
`btkDistanceFibersKMeansApproximationClustering --help`

Fig.9 shows an example of clustering results in the corpus callosum region obtained using the method above with the following input parameters: `-d 10 -c 7`.

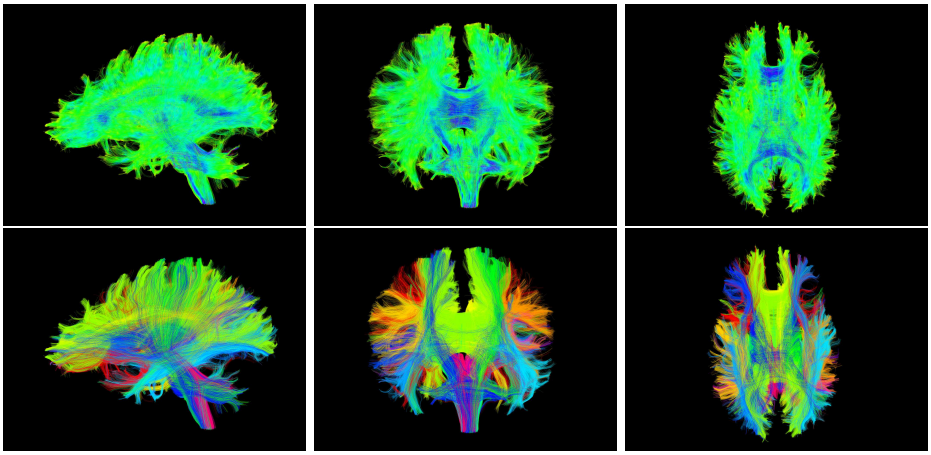


Figure 8: An example of fiber tracts clustering in the whole brain using `btkProbabilisticSegmentationMapBasedClustering` method. *The fibers from the tractography process and final clustering fiber bundles are shown in the first and second row, respectively. Three viewing angles of 3D depictions of fibers are shown (from left to right): left lateral view, anferior view and interior view.*

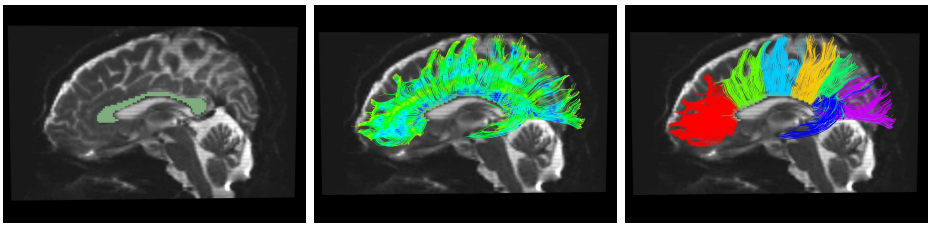


Figure 9: An example of fiber tracts clustering in the corpus callosum using `btkProbabilisticSegmentationMapBasedClustering` method. *The region of interest on the corpus callosum, fibers from the tractography process and final clustering fiber bundles are shown from left to right, respectively. The fibers are clustered according to anatomical regions as proposed by Witelson.*

4 Utilities

btkApplyMaskToImage : Multiply an image with a mask.

btkApplyTransformationToImage : Resample an image using a reference and a transformation (itk or btk).

btkAverage3DImages : Compute the mean (and variance) image of a set of 3D images.

btkAverageImagesWithReference : Creates a high resolution image from a set of low resolution images.

btkBinarizeLabels : Binarize one label image (depending on the label value chosen).

btkBinarizeMask : Binarize a float image using a threshold.

btkBinarizeTissueProbabilityMaps : Binarize a set of probability maps ([0,1]) into a short image (by taking the max probability at each voxel).

btkColorFiberTractsByOrientation : Color the fibers (polydata) by their orientation (local or global).

btkComputeChamferDistance : Compute the chamfer distance.

btkComputeCoefficientOfDeterminationForAtlas : Compute the coefficient of determination (R^2) for atlas made with regression script (see Atlas part of this document).

btkComputeFeatureSelectionResiduals : Compute the residuals of the feature selection algorithm (norm of the reconstruction error).

btkComputeOverlap : Compute the overlap (Dice coefficient and Jaccard index) between two label images.

btkComputeSoftMaskUsingOrthogonalImages : Compute soft masks of a set of images using the geometry of protocol acquisition (*i.e.* orthogonal images).

btkConvertGradientTable : Transform gradients from world to image coordinates.

btkCropImageUsingMask This program crops one (3D or 4D) image using a 3D mask.

btkDifferentialBiasCorrection : Differential bias correction method for n images (Leung et al. Neuroimage 2012).

btkDiffusionScalarMeasurement : Compute diffusion scalars (tensors and high-order; e.g. FA, GFA, GA, FMI, etc.)

btkDistanceBetweenBinaryImages : Compute Hausdorff and Mean Contour Distances between two binary images (ITK Filters: HausdorffDistanceImageFilter and ContourMeanDistanceImageFilter).

btkExtractMaskUsingBoundingBox : Extract Masks using the intersection of the bounding boxes of images.

btkExtractOneImageFromSequence This program extracts one image from a 4D sequence. It can be useful for diffusion MRI image analysis.

btkFCMClassification : Fuzzy C-means classification algorithm.

btkFSLToITKTransform : Convert a FSL transform to the ITK standard transform ($--inverse$ do the opposite).

btkGenerateSimulatedFiber : Generate a simulated fiber from coordinates of points stored into a text file.

btkGenerateVtkFileFromFiberDataTextFiles : Generate a VTK file of fibers from data stored into several text files, and visualize it.

btkHistogramMatching : Normalize the grayscale values of one image using a reference image by histogram matching (ITK filter: HistogramMatchingImageFilter).

btkImageGaussianFilter : Apply a gaussian filter on an image (ITK filter: DiscreteGaussianImageFilter).

btkImageHistogram : Analysis of images through histograms (It can save into text files histogram, cumulative distribution function (cdf) and inverse-cdf of an image).

btkImageInjection This program performs the injection of a set of images with an already existing set of transformations. This avoids the need to perform a new image reconstruction (computationally expensive) after modification of the input images (some filtering for example) or an involuntary deletion of the reconstructed image.

Recommended usage: `btkImageInjection -i ana01.nii.gz ··· -i anaN.nii.gz -m mask01.nii.gz ··· -m maskN.nii.gz -t transform01.txt ··· -t transformN.txt -o ana_reconstructed.nii.gz --mask`

btkImageMorphologicalClosing : Greyscale Morphological Closing by a ball structuring element (ITK filter: GrayscaleMorphologicalClosingImageFilter).

btkImageMorphologicalTopHat : Greyscale Morphological Top Hat by a ball structuring element.

btkImageResampling : Resample an image using: 1) a reference image (better option) or 2) specific size or spacing (mainly based on ITK filter: ResampleImageFilter)

btkImageSimilarity : Calculates mean square error, mutual information, normalized correlation and normalized mutual information of two images A and B. The use of a mask is possible.

btkImageSubtract : Pixel-wise subtraction of two image, or one image and a constant (ITK filter: SubtractImageFilter).

Recommended usage: `btkImageSubtract-i ana01.nii.gz -i ana02.nii.gz -o substraction_image.nii.gz` or `btkImageSubtract -i ana.nii.gz -c value -o substraction_image.nii.gz`

btkInverseDisplacementField : Compute the inverse of a displacement field (ITK filter: InverseDisplacementFieldImageFilter)

Recommended usage: `btkInverseDisplacementField -i field.nii.gz -o inverse_field.nii.gz`

btkIteratedBackProjection : Apply iterated back projection to high resolution image using one low resolution image.

btkMajorityVoting : Compute label map using majority voting rule.

btkMidwayHistogramEqualization : Perform midway histogram equalization for a set of 3D images (cf Delon JMIV 2004).

btkModifyImageUsingLookUpTable This program modifies one image using a look up table defined in a ascii file (2 columns, one for the original values, one for the final values). It can be useful to relabel a segmented image.

btkNiftiToNrrd This program convert a diffusion sequence in nifti format[6] to the nrrd format (*.nhdr).

Recommended usage: `btkNiftiToNrrd -i input.nii.gz -o output.nhdr`

btkNrrdToNifti This program convert an image from Nrrd file (*.nhdr and *.nrrd) to a Nifti file (*.nii or *.nii.gz). The conversion of a DWI image is possible by using the option `--dwi`.

Recommended usage: `btkNrrdToNifti -i input.nhdr -o output.nii.gz`.

Recommended usage for DWI sequence: `btkNrrdToNifti --dwi -i input.nhdr -o output.nii.gz`.

btkPSNR : Compute the PSNR between an input image and a reference image.

btkPrintImageInfo : Prints image information (size, origin, spacing, directions, anatomical orientation, pixel type).

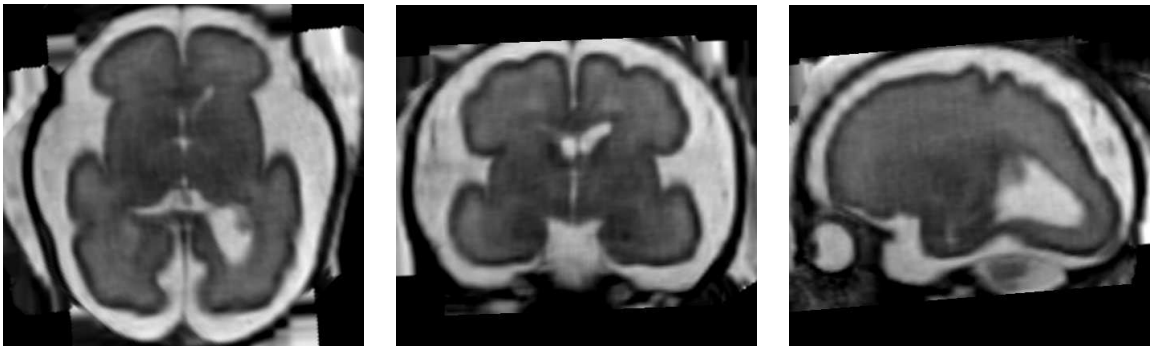
btkProbabilityMapNormalization : Normalize in a voxelwise manner a set of (positive) probability maps.

btkReconstructionComparisonTool : Compare several reconstruction methods: creates a label image explaining which reconstructed image is the best at each voxel.

btkRegisterDiffusionToAnatomicalData This program registers a DW sequence to an anatomical image.

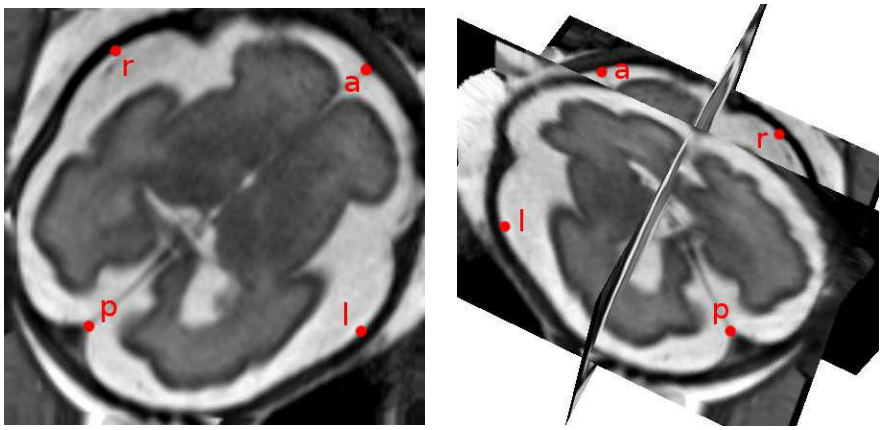
btkReorientDiffusionSequenceToStandard Reorients a DW sequence to the standard orientation. This is necessary with fetal images since the fetus is in a random orientation with respect to the scanner. This is particularly important in DWI because colormaps lack of significance, which makes difficult the identification of specific bundles.

[6] Currently there is no nifti standard for DWI, so DW images are saved as a standard nifti sequence (*.nii, *.nii.gz) and two text files containing the b-values (.bval) and the gradient directions (.bvec).



(a) (b) (c)

Figure 10: Example of an anatomical reconstruction of a fetal brain by using `btkImageReconstruction`. (a) axial, (b) coronal, and (c) sagital view.



(a) (b)

Figure 11: Placement of landmarks by using Slicer. (a) axial slice, (b) 3D view.

Recommended usage: `btkReorientDiffusionSequenceToStandard -i dwi.nii.gz -o dwi_reoriented.nii.gz -l landmarks.fcsv`.

`landmarks` is a landmarks file obtained as explained for btkReorientImageToStandard.

btkReorientImageToStandard This program reorients a image to the standard orientation. This is necessary with fetal images since in general the fetus is in a random orientation with respect to the scanner. Recommended usage: `btkReorientImageToStandard -i ana.nii.gz -o ana_reoriented.nii.gz -l landmarks.fcsv`. `landmarks` is a text file containing points that define the left-right and the posterior-anterior directions. The points l and r define the left $\rightarrow$ right direction, and the points p and a define the posterior $\rightarrow$ anterior direction. Such file can be easily generated by using Slicer (version 3) as follows:

1. Open the high-resolution image by using the *Volume* module.
2. Toogle on the visibility of all slices in the 3D view. This allows to identify the left and right sides of the brain in the 2D views.
3. Place the landmarks l, r, p, and a in this order by using `[p]`.
4. Save the file (*.fcsv) by using the menu File $\rightarrow$ Save.

btkResampleLabelsByInjection : Resample a set of label images using the injection method.

btkRescaleIntensity : Rescale the intensity values of an image using short values (possibility to use a mask image).

btkSequenceNormalization : Writes a dwi sequence as a single image B0 + the diffusion images. The new B0 is the mean of all B0 images in the original sequence, or a user-provided B0.

btkSetStandardCoorSystem : Sets the direction to the identity, and the origin to the center of the image.

btkSimulateLowResolutionImage : Simulates a low resolution image from a high resolution image (reconstructed, super-resolution, or acquired) and a transformation between both images (transformation can be set or randomly computed) .

btkSimulateMotionSliceBySlice : Apply transformations on slices of input image.

btkSimulateStandardViewFromIsotropicImage : Simulates standard acquisitions (axial, coronal, and sagital) from an isovoxel by using an observational model. This is useful for example to assess the performance of reconstruction algorithms according to the subsampling factor (the reconstructed image from the simulated images is then compared to the isovoxel image).

btkSimulateStandardViewsFromImage : Compute a low resolution image from a high resolution image by using a generative model (old version of btkSimulateStandardViewFromIsotropicImage).

btkSTAPLE : Apply the multi-label segmentation STAPLE filter (ITK implementation).

btkTractDensityMap : Create tract density images using set of fibers as input.

btkTranslateImageOverTemplate : Usefull for the extraction mask pipeline, it will translate the center of image (or barycenter of masked image) on the center of the template image.

btkWeightedSumOfAffineTransforms : Compute a weighted mean of affine transforms (no check for normalized weights!).

btkWeightedSumOfImages : Compute a weighted mean of 3D images (no check for normalized weights!).

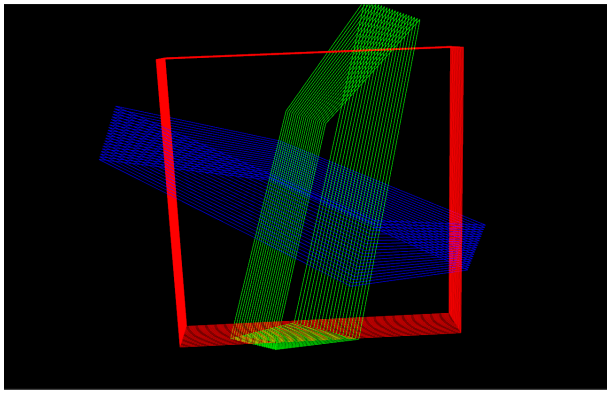


Figure 12: Example of a render of three polydatas (axial, coronal, and sagital). `btkSlicesMotionViewer`.

5 Viewers

btkSlicesMotionViewer : Construct a VTK polydata wich outline the slice for each input. Transforms can be given in output the option `-r` will open a render window. A typical use of this program is to visualize slice by slice transformation onto volume. See figure 12.

Recommended usage: `btkSlicesMotionViewer -i ana01.nii.gz ··· -i anaN.nii.gz -t transform01.txt ··· -t transformN.txt -o PolyData01.vtk ··· -o PolyDataN.vtk -r`

Note that the only required arguments are the inputs and outputs.

btkDiffusionViewer : Useful for diffusion dataset visualization (memory consuming). The option `--percent_of_sample_points` can be used to set the percent of points (of the number of voxels) being sampled for modelling display. This can be useful for memory saving and spacing control between model representations. An other useful option is the input mask (`-m`) used to mask a part of the image. Once the display window is opened, you can move the slice up and down by using the corresponding arrows of your keyboard. To display the mean directions and the modelling, you can use respectively the touch 'd' and 'm' of your keyboard. You can move the view by pointing the mouse outside of the image and then moving the mouse while keeping the left button of the mouse pressed.

Example usage: `btkDiffusionViewer -i dwi.nii.gz -m mask.nii.gz`

Acknowledgment

The research leading to these results has received funding from the European Research Council under the European Communitys Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 207667).

References

[1] B. Caldairou, N. Passat, P. Habas, C. Studholme, M. Koob, J. . Dietemann, and F. Rousseau. Data-driven cortex segmentation in reconstructed fetal mri by using structural constraints. In *Computer Analysis of Images and Pattern - CAIP 2011*, volume 6854 of *Lecture Notes in Computer Science*, pages 503–511, 2011.

[2] P. Coupé, P. Yger, S. Prima, P. Hellier, C. Kervrann, and C. Barillot. An optimized blockwise non local means denoising filter for 3d magnetic resonance images. *IEEE Trans. Medical Imaging*, 27(4):425–441, 2008.

[3] M. Descoteaux, E. Angelino, S. Fitzgibbons, and R. Deriche. Regularized, fast, and robust analytical q-ball imaging. *Magnetic Resonance in Medicine*, 58(3):497–510, 2007.

[4] E. Oubel, M. Koob, C. Studholme, J. Dietemann, and F. Rousseau. Reconstruction of scattered data in fetal diffusion mri. *Medical Image Computing and Computer-Assisted Intervention*, 2010.

[5] J. Pontabry, F. Rousseau, Oubel E., Studholme C., Koob M., and J.-L. Dietemann. Probabilistic tractography using q-ball imaging and particle filtering: Application to adult and in-utero fetal brain studies. *Medical Image Analysis*, 17(3):297–310, 2013.

[6] J. Pontabry, F. Rousseau, M. Schweitzer, C. Studholme, M. Koob, and J.-L. Dietemann. Longitudinal probabilistic atlas of the fetal brain. In *Nouvelles méthodologies en imagerie du vivant*, 2012.

[7] F. Rousseau, O.A. Glenn, B. Iordanova, C. Rodriguez-Carranza, D.B. Vigneron, J.A. Barkovich, and C. Studholme. Registration-based approach for reconstruction of high-resolution in utero fetal MR brain images. *Acad Radiol*, 13(9):1072–1081, Sep 2006.

[8] F. Rousseau, P. Habas, and C. Studholme. A supervised patch-based approach for human brain labeling. *IEEE Transactions on Medical Imaging*, 30(10):1852–1862, Oct 2011.