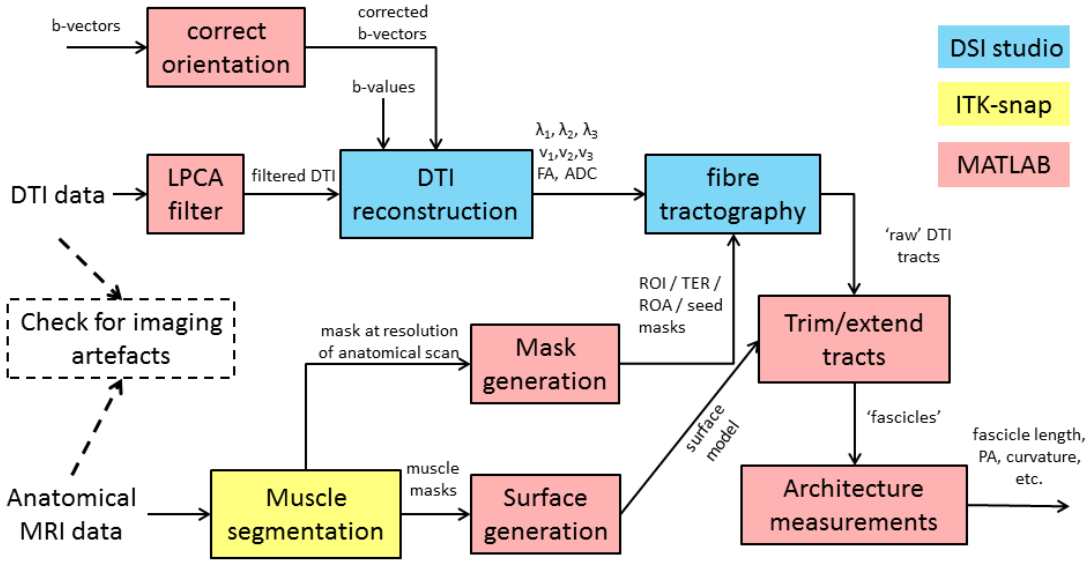
Manual for analysing DTI data

Version 1.0 - February 2017

1. Introduction

The DTI measurements of muscle architecture involve quite a few steps, as depicted in the schematic overview below. Most of these steps are now automated and just require calling the appropriate MATLAB functions at the right time. This manual is intended to be both a practical guide on how to analyse the DTI data, as well as an explanatory guide to understand what is happening behind the scenes.



The DTI analysis involves the following steps:

- Copy files from NeuRA's server, give the files easily interpretable filenames and store the files consistently in your local working directory.
- Filter the DTI data and reconstruct the diffusion tensor.
- Segment muscles on the anatomical MRI scan.
- Create masks and surfaces used for fibre tractography
- Perform fibre tracking, reconstruct fascicles and measure muscle architecture
- Measure DTI indices for all fibre tracts.
- Inspect the results.

Each section in this manual will explain one step of the process. With the Muscle DTI Toolbox also comes a set of example data and example scripts. It is recommended to have a look at this scripts, and to run these scripts, before you run your own analysis.

2. Before you get started

The following software is required for DTI processing and needs to be downloaded and installed on your computer:

ITK-SNAP http://www.itksnap.org/pmwiki/pmwiki.php?n=Downloads.SNAP3
DSI studio http://dsi-studio.labsolver.org/dsi-studio-download
Convert3D http://www.itksnap.org/pmwiki/pmwiki.php?n=Downloads.C3D
MATLAB provided through UNSW or NeuRA

You will need to add DSI Studio and Convert3D to your 'environmental path', so that the commands 'dsi_studio' and 'c3d' will be recognised when called from the command line. This is how you can add programs to your path in Windows: http://www.howtogeek.com/118594/how-to-edit-your-system-path-for-easy-command-line-access/

In MATLAB, you will have to add the Muscle DTI Toolbox. This toolbox contains some downloaded and some custom-written MATLAB-functions that are needed for processing the data. Add the toolbox to your path in MATLAB by selecting from the menu-bar Home > Set Path > Add with subfolders… Then select the folder where you have saved the Muscle DTI Toolbox and press 'Save' and 'Close'.

3. Copy and rename MRI data

- Keep file naming consistent within one study.
- Each filename starts with the study code, followed by the subject number.
- Use simple, descriptive filenames to describe the scan type and the data it contains.
- Always keep an Excel spreadsheet with original filenames, new filenames, a short data description and comments on image quality.

The first step of processing the MRI and DTI data is to make a working directory for your study on your local drive or on a network drive. All raw data files and files used for processing will be stored in this working directory. In your working directory, make a subfolder for each subject named <study><subject number>. For example, create a subfolder MUTR001 for the data of subject 1 of study MUTR. After you've created this folder, copy the data from the server (/samurai) to this local directory. You will find the data on samurai in a folder specific to the study (e.g. MUTR) in a folder with a 3-digit (e.g. 003) code that identifies the scan session number. We will use the data stored in the subfolder 'NIFTI'.

The files on the server have sometimes long and not very descriptive filenames. After you've copied the files that you need (usually all anatomical scans and all DTI scans, but not survey scans) to your working directory, rename the files as follows

<study><subject number><scan type>.nii.gz, where:

- <study> is the study code (e.g. MUTR, CPDTI, STDTI)
- <subject number> is the subject number padded with zeros to form a 3-digit code (e.g. 001, 014 or 123 for subject 1, 14 and 123, respectively)
- <scan type> the name of the scan type. Choose a simple, descriptive name for each scan. For example, mdixon for mDixon scans, DTI_NSA2 for DTI scans obtained with NSA=2, etc. Within one study, use the same name for the same scan types for all subjects.

All data should be in zipped nifti format (.nii.gz). This is the default on samurai. If your data is not zipped (and thus has extension .nii), you can zip it using 7-zip or in MATLAB using the command `gzip(<filename>)`.

Each DTI scan has corresponding .bval and .bvec files, which will also be on samurai in the same folder as the DTI data. Copy these files to your working directory as well, because they are needed for the tensor

reconstruction. Give the .bvec and .bval-files the same name as the DTI scan (for example. MUTR001_DTI_NSA2.bval and MUTR001_DTI_NSA2.bvec).

In the main folder of your working directory, always keep an Excel-file named <study_code>_data_key.xls (e.g. MUTR_data_key.xls).This file has four columns with headers 'name on samurai', 'name in working directory', 'description' and 'data quality'. In the first column, write the original filename on Samurai (the exact filename including the folder). In the second column, give the new filename as used in your local working directory. Give a short, informative description of the scan in the third column, e.g. 'mDixon scan of joint position 1' or 'DTI scan with NSA=4 in joint position 3'. Use one excel-file for the whole study (not one per subject). In the fourth column you can comment on possible imaging artefacts present in the image (e.g. 'movement artefacts causing blurring in the images'.

Examples for file naming

MUTR001_mdixon.nii.gz could be the filename of the mDixon scan of subject 001 in the study MUTR

CPDTI023_DTI_NSA2.nii.gz could be the filename of the DTI scans (obtained with NSA=2) for subject 23 of study CPDTI. The bval and bvec file will be named CPDTI023_DTI_NSA2.bval and CPDTI023_DTI_NSA2.bvec, respectively.

If you have multiple conditions, for example if you have scanned different joint angles, include the condition in the filename as well. For example, MUTR001_mdixon_j01.nii.gz will be the mDixon scan of joint position 1, for subject 1 in the study MUTR, and MUTR001_mdixon_j02.nii.gz will be the mDixon scan of joint position 2.

The bottom line for file naming is: (1) always include study code and subject number in the filename, (2) use filenames that describe which scan type and condition the file contains and (3) use consistent filenames within your study.

4. DTI preprocessing and tensor reconstruction

All the following steps will be done by the MATLAB-function `Preprocessing_and_DTI_recon`. By running this script with the appropriate inputs, the data is filtered and .src and .fib file are created, which can be used by DSI Studio for tractography. The following description explains which steps are performed when you use this function.

The first step of preprocessing the DTI data is filtering the diffusion weighted data. We use the Local Principal Component Analysis (LPCA) filter described in:

Manjon JV, Coupe P, Concha L, Buades A, Collins DL, and Robles M. Diffusion Weighted Image Denoising Using Overcomplete Local PCA. PLoS ONE 8: 2013.

We use a MATLAB-implementation of this filter that was made available by the authors.

The NIFTI-files for a DTI scan contain all the diffusion-weighted data for all gradient directions. From these 4D data and the b-value and gradient directions (b-vectors), the diffusion tensor can be reconstructed. From the tensor, the primary, secondary and tertiary diffusion directions can then be extracted, as well as maps of fractional anisotropy, ADC, etc. The tensor reconstruction and subsequent calculations are done by DSI studio, but DSI Studio is called from MATLAB so that you don't actually have to open DSI Studio. In order for MATLAB to call DSI studio, you will need to have installed DSI studio and have added it to your path (see section 'Software'). Prior to tensor reconstruction, the b-vectors need to be reoriented to account for

differences in scanner coordinate system and image coordinate system. This is also done using MATLAB with the function correct_bvec.m.

DTI filtering, b-vector correction and tensor reconstruction can be done automatically from MATLAB using the function `Preprocessing_and_DTI_recon.m`. This will create the DSI studio source file (with extension .src.gz, containing all the DTI data in DSI studio format) and the fibre-file (with extension .fib.gz, containing primary eigenvector data, FA data as well as the full tensor for each voxel). You can also give the DTI file name and bvec- and bval-files as input to the function.

How to use the function Preprocessing_and_DTI_recon?

For full information on usage, go to MATLAB and type `help Preprocessing_and_DTI_recon` in the command window. An example script `example_preprocessing.m` that does the preprocessing of the example data is provided in the main folder of the Muscle DTI Toolbox.

You can run `Preprocessing_and_DTI_recon` in MATLAB without any input arguments. MATLAB will then ask you to manually select the raw DTI data and the corresponding bval- and bvec-files. It will also ask you if you want to filter the data or not and if you want to correct the bvec-file.

For automated processing, the function accepts the following syntax:

```
Preprocessing_and_DTI_recon('DTI',<raw_dti_filename>,'bval',<bval_filename>,'bvec',<bvec_filename>,'filter',<true/false>)
```

with <raw_dti_filename> the filename of the DTI data (including extension .nii.gz) and <bval_filename> and <bvec_filename> the filename of the corresponding .bval- and .bvec-file, respectively. If you don't want your data to be filtered, you can optionally provide the input argument `'filter',false` (the default is 'true').

The DSI studio source and fibre-files will have extension .src.gz and .fib.gz, respectively, and the same filename as the DTI data with "_LPCA" appended to the filename when the LPCA filter has been applied. For example, the file MUTR023_DTI_scan2_LPCA.fib.gz contains the fibre-file, obtained from the filtered DTI data for subject 23 of study MUTR.

5. Muscle segmentation

Muscle segmentation (= outlining muscle boundaries) is preferably done in ITK-SNAP, an open source program dedicated to medical image segmentation. The details of how to segment is beyond the scope of this manual, but here are a few tips and tricks:

- A useful short course on segmentation with ITK-SNAP can be found on the following website: http://www.itksnap.org/pmwiki/pmwiki.php?n=Train.RSNA2016
- Save the segmentation regularly by pressing Ctrl + S. ITK-snap often crashes and when it does, you will lose the unsaved changes to your segmentation. The more often you save, the less work you have to redo when ITK-SNAP crashes. (It will not take long before you get this point.)
- Use all views (axial/sagittal/coronal) if unsure about the boundary of the object you are segmenting. Muscle boundaries are sometimes not obvious in the axial slices, but sometimes become clear when looking at it from the sagittal or coronal view.
- Familiarise yourself with the shortcut-keys such as the numbers 1-5 to switch between the different segmentation options and zoom modes. I find it helpful to have one hand on the mouse and the other hand on the keyboard during segmentation for rapid switching between the zoom (2), polygon

tracing (3) and paint (4) option. Note that you can make the segmentation more or less transparent by pressing 'd' and 'a', respectively.

- You can overlay a color-coded map of the primary eigenvector data on the anatomical data by clicking File > Add another image… in the menu bar of ITK-SNAP and selecting the corresponding eigenvector-map data (the .nii.gz with '_EV1' at the end of the filename).

Once you are happy with your segmentation, save the segmentation file as a zipped NIFTI (.nii.gz) in a subfolder 'masks' of the folder in which your MRI data lives. In the masks folder, make a label-info file with exactly the same filename as the segmentation data file but with extension .lab. For example, if your segmentation file is called segmentation.nii.gz, the corresponding label info file is called segmentation.lab. The .lab file is a simple text-file with as many lines as there are labels in the segmentation file. The .lab file should look like this:

```
<label_number1>, <short name1>, <long name1>
<label_number2>, <short name2>, <long name2>
<label_number3>, <short name3>, <long name3>
etc.
```

The label_number is the number of the label in the segmentation data file. The short name is an abbreviated name for the object and the long name is a more descriptive name (for example the full muscle name). Separate the number, short and long name by a comma. This file can be created in any text editor or even in MATLAB. An example of a label file can be found in the folder *example_data/masks*.

Example
This if what the label-info file looks like for a segmentation file in which the medial gastrocnemius has been given label 1, the lateral gastrocnemius label 3 and the tibialis anterior label 7:

```
1, MG, medial gastrocnemius
3, LG, lateral gastrocnemius
7, TA, tibialis anterior
```

6. DTI masks and muscle surface generation

All the following steps will be done by the MATLAB-scripts `MakeSurfaceAndMasks` and `MakeFAmask`. These functions will create the masks required for DTI tractography and the muscle surface required for tract truncation and extrapolation.

For fibre tractography in a muscle the following masks are required:

- A *seed mask* which contains only voxels inside the muscle that are at least two voxels distant from the boundary. This is to avoid seeding at the boundary of the muscle where the DTI signal is often unreliable.
- A *boundary mask* which contains only voxels at the boundary of the muscle. This mask is used to terminate tractography to prevent fibre tracts from continuing outside the muscle.
- A *fractional anisotropy mask* which contains only voxels outside the plausible FA values for muscles (typically 0.1<FA<0.5). This mask stops fibre tracts from going through regions with FA values outside the range that is feasible for muscle tissue.

The tractography masks have to be at the resolution of the DTI data and will be created by downsampling the segmentation made on the anatomical data to the resolution of the DTI scan using Convert3D. In order for this function to work, you need to have installed Convert3D on your computer and have it added to your

path, so that c3d is recognised as an external command (see section 'Software'). In addition to tractography masks, a triangulated mesh of the muscle surface (surface model) is created from the segmentation. The surface model is created using the MATLAB toolbox iso2mesh and will be saved as an STL-file. This surface will be used in subsequent analyses to truncate tracts with endpoints outside the muscle surface, and to extrapolate tracts so that their endpoints are located exactly on the muscle surface.

The FA mask is created using the function `MakeFAmask.m`.

How to use the function MakeSurfaceAndMasks?

For full information on usage, go to MATLAB and type `help MakeSurfaceAndMasks` in the command window.

The function requires at least two inputs. The first input is the full filename of the segmentation file (with extension .nii.gz). It requires a label-info file with the same name as the segmentation file (but with extension .lab) to exist (see section 'Muscle segmentation').The second input is the name of the DTI file to which the masks will be resampled to. If no extra input arguments are given, masks will be created for all labels present in the label-info file. The following command will create masks and surfaces for all labels:

```
fname_segm = 'C:/Bart/data/masks/segmentation.nii.gz';
fname_DTI  = 'C:/Bart/data/DTI_scan.nii.gz'
MakeSurfaceAndMasks(fname_segm,fname_DTI)
```

If the segmentation file contains several labels but you only want to create files, for example, for label 3 and 5, use the following syntax:

```
MakeSurfaceAndMasks(fname_segm,fname_DTI,'LabelNumbers',[3 5])
```

Instead of label numbers, you can also give a list (cell string) of label names to create surfaces and masks for:

```
MakeSurfaceAndMasks(fname_segm,fname_DTI,'LabelNames',{'MG','TA'})
```

The label names correspond to the short names in the label-info file.

If you give an output argument, a MATLAB structure with filenames will be returned for each label:

```
mask_filenames = MakeSurfaceAndMasks(fname_segm,fname_DTI)
```

How to use the function MakeFAmask?

For full information on usage, go to MATLAB and type `help MakeFAmask` in the command window.

To create an FA mask, two inputs are required: the filename of the .fib.gz file and a 2-element vector with the lower and upper FA threshold.

Example

```
fib_fname = 'C:/Bart/data/DTI_NSA2.fib.gz';
FA_threshold = [0.1 0.5];
FA_mask_filename = MakeFAmask(fib_fname,FA_threshold)
```

7. Muscle fibre tracking

Now the DTI data have been preprocessed, the tensor has been reconstructed and the tractography masks have been created, you are ready for fibre tracking (tractography). Tractography is done in DSI Studio, but DSI studio is called from MATLAB so that you don't actually have to open the program. Again, this requires

DSI Studio to be installed on your computer and added to your path so that dsi_studio is recognised as an external command. The MATLAB-function `TrackFibres` will be used to call DSI Studio to do the fibre tracking.

How to use the function TrackFibres ?

For full information on usage, go to MATLAB and type `help TrackFibres` in the command window.

The function requires two MATLAB structures as input. The first structure contains filenames of the fibre-file (in the field 'Source'), the seed mask (in the field 'Seed'), the boundary mask (in the field 'TER') and the FA mask (in the field 'ROA'). The second structure provides tractography settings such as step size and stopping criteria. The first output argument of the function is a structure `DTItracts` with the tracts and some extra information on fibre tracking. Type `help TrackFibres` for more info on the meaning of the fields in the input and output structures. Also, see the script `example_fibretracking.m` for an example of tractography and subsequent analyses on the fibre tracts.

Example:

```
% Fibre track filenames
TrackFileNames.FIB      = 'C:/Bart/data/DTI_NSA2.fib.gz';
TrackFileNames.Tracts   = 'C:/Bart/data/tracts/example_tracts.mat';
TrackFileNames.Seed     = 'C:/Bart/data/masks/MG_seed.nii.gz';
TrackFileNames.TER      = 'C:/Bart/data/masks/MG_boundary.nii.gz';
TrackFileNames.ROA      = 'C:/Bart/data/masks/FA_mask.nii.gz';

% Fibre track settings
TrackSettings.MinLength    = 20;
TrackSettings.MaxLength    = 200;
TrackSettings.FA_threshold = 0.10;
TrackSettings.Stepsize     = 1;
TrackSettings.FiberCount   = 1000;
TrackSettings.SeedCount    = [];
TrackSettings.MaxAngle     = 10;
TrackSettings.Smoothing    = 0;

[DTItracts,StopFlag] = TrackFibres(TrackFileNames,TrackSettings);
```

Step 8. Fascicle reconstruction and muscle architecture measurements

The function `TrackFibres` will return the raw fibre tracts which can now be truncated and extrapolated to have endpoints on the muscle surface. Truncation simply throws out parts of fibres that are outside the muscle surface. Extrapolation means fitting a polynomial in 3D to the raw fibre tracts, and then linearly extending the polynomial at its endpoints until the muscle surface is intersected. Measurements of muscle architecture are made on the truncated, polynomial fitted and extrapolated 'fascicles'.

Tract truncation, extrapolation and architecture measurements are done by the MATLAB function `CalcArchitecture`. This function will call other four other functions (`TruncateTracts`, `ExtrapolateTracts`, `CalcPenAngle` and `CalcCurvature`) to perform the different steps.

How to use the function CalcArchitecture?

For full information on usage, go to MATLAB and type `help CalcArchitecture` in the command window. Also check out the help for subfunctions `TruncateTracts`, `ExtrapolateTracts`, `CalcPenAngle` and `CalcCurvature` for more information.

The function expects two or three inputs. The first input is the structure DTItracts (output of function `TrackFibres`). The second input is the muscle surface model, presented as a structure with fields `vertices` and `faces`. This variable can be created by loading the surface from the previously created STL-file using `stlread`. The optional third input is the order of the polynomial fitted on the raw fibre tracts. If only two inputs are provided, the default order of 3 is used.

Example

The following lines will load the surface into the workspace in the structure `surf_model`

```
stl_filename = 'C:/Bart/data/masks/MG_surf.stl';
surf_model = stlread(stl_filename)
```

And the following line will perform the tract truncation, extrapolation and architecture measurements with the default polynomial order of 3:

```
DTItracts = CalcArchitecture (DTItracts, surf_model);
```

Example for architecture measurements with a polynomial order of 2:

```
DTItracts = CalcArchitecture (DTItracts, surf_model,2);
```

The structure array `DTItracts` now only exists in the workspace of MATLAB. To save as a MAT-file that can be loaded into MATLAB for later use, use the following command:

```
save(<tract_filename>,'-struct','DTItracts')
```

where `tract_filename` is the full filename of the MAT-file with the DTI tracts (i.e.

You can load the data again as a structure array `DTItracts` as follows:

```
DTItracts = load(<tract_filename>)
```

Step 9. Tract analysis of DTI indices

The DTI indices like fractional anisotropy and mean diffusivity were previously calculated for all voxels and stored in the .fib.gz file. To calculate the value of a DTI index of a tract (or fibre), the maps can be interpolated at the location of that tract. The interpolation of the maps with the DTI indices is done using the function `CalcDTI_indices`. By running this function with the appropriate inputs, the fractional anisotropy, mean diffusivity and the three eigenvalues are calculated for all the tracts in the data.

How to use the function CalcDTI_indices?

For full information on usage, go to MATLAB and type `help CalcDTI_indices` in the command window.

The function expects two inputs. The first input is the structure `DTItracts` containing at least the fields `tracts` and `fibindex_trunc`. These fields should be present in `DTItracts` after you have run `CalcArchitecture`. The second input is the full file name of the fibre-file.

Example

`DTItracts` should already be loaded into the workspace, or can be loaded using:

```
DTItracts = load(<tract_filename>);
```

```
fibre_filename = DTI_NSA2.fib.gz;
DTItracts = CalcDTI_indices(DTItracts,fibre_filename);
```

This adds the fields `fa`, `md`, `lambda1`, `lambda2` and `lambda3` to `DTItracts`. Save the data again using the following command:

```
save(<tract_filename>,'-struct','DTItracts')
```

10. Inspect the results

You can inspect the results of fibre tractography and subsequent analyses using the function `InspectTracts`. This will create a figure with a 3D representation of the fibre tracts and distribution of architectural parameters and DTI indices for all tracts.

How to use the function InspectTracts?

For full information on usage, go to MATLAB and type `help InspectTracts` in the command window.

You can run the function without any input arguments by simply typing `InspectTracts` in the command window. This will open a dialog box to select a tract file. Alternatively, you can give the filename of the tract file or the structure `DTItracts` (if loaded into the workspace) as the first input. If you want to display the surface model as well, add `'SurfModel',<stl_filename>` to the command.

Example

To plot both the 3D fibre tracts and the distribution of architectural parameters/DTI indices for all tracts:

```
InspectTracts(DTItracts)
```

To add the surface model to the 3D plot, use the optional input argument `'SurfModel'`:

```
surf_model = stlread('example_data/masks/surface.stl')
InspectTracts(DTItracts,'SurfModel',surf_model)
```

You can also just give the filename of the surface model as input argument:

```
InspectTracts(DTItracts,'SurfModel', 'example_data/masks/surface.stl')
```

The first input (`DTItracts`) can also be the filename of the tract file (instead of the structure `DTItracts`).

```
InspectTracts('example_data/tracts/example_tracts.mat')
```