MRtrix Documentation

Release 3.0

MRtrix contributors

May 05, 2017

Install

MRtrix provides a set of tools to perform diffusion-weighted MRI white matter tractography in the presence of crossing fibres, using Constrained Spherical Deconvolution (Tournier et al.. 2004; Tournier et al. 2007), and a probabilisitic streamlines algorithm (Tournier et al., 2012). These applications have been written from scratch in C++, using the functionality provided by Eigen, and Qt. The software is currently capable of handling DICOM, NIfTI and AnalyseAVW image formats, amongst others. The source code is distributed under the Mozilla Public License.

CHAPTER 1

Before you install

Acknowledging this work

If you wish to include results generated using the *MRtrix3* package in a publication, please include a line such as the following to acknowledge this work:

- Fibre-tracking was performed using the MRtrix package (J-D Tournier, Brain Research Institute, Melbourne, Australia, https://github.com/MRtrix3/mrtrix3) (Tournier et al. 2012)

Warning: This needs to be updated!

Any other relevant references will be listed on the specific application's page.

Warranty

The software described in this manual has no warranty, it is provided "as is". It is your responsibility to validate the behavior of the routines and their accuracy using the source code provided, or to purchase support and warranties from commercial redistributors. Consult the Mozilla Public License for further details.

License

CHAPTER 2

Linux installation

We outline the steps for installing *MRtrix3* on a Linux machine. Please consult the MRtrix3 forum if you encounter any issues with the configure, build or runtime operations of *MRtrix3*.

Check requirements

To install *MRtrix3*, you will need the following:

- a C++11 compliant compiler (GCC version >= 4.8, clang)
- Python version >= 2.7
- The zlib compression library
- Eigen version 3.2 *(do not install the beta version)*
- Qt version >= 4.7 *[GUI components only]*

Warning: To run the GUI components of *MRtrix3* (`mrview` & `shview`), you will also need:

- an OpenGL 3.3 compliant graphics card and corresponding software driver

Note that this implies you *cannot run the GUI components over a remote X11 connection*, since it can't support OpenGL 3.3+ rendering - see *Remote display issues* for details.

Install Dependencies

The installation procedure will depend on your system. Package names may changes between distributions, and between different releases of the same distribution. The commands below are suggestions based on what has been reported to work in the past, but may need to be amended. See below for hints on how to proceed in this case.

- Ubuntu Linux (and derivatives, e.g. Linux Mint):

```
sudo apt-get install git g++ python python-numpy libeigen3-dev zlib1g-dev libqt4-
↪opengl-dev libgl1-mesa-dev
```

- RPM-based distros (Fedora, CentOS):

```
sudo yum install git g++ python numpy eigen3-devel zlib-devel libqt4-devel libgl1-
↪mesa-dev
```

- Arch Linux:

```
sudo pacman -Syu git python python-numpy gcc zlib eigen qt5-svg
```

If this doesn't work

You may find that your package installer is unable to find the packages listed, or that the subsequent steps fail due to missing dependencies (particularly the `./configure` command). In this case, you will need to search the package database and find the correct names for these packages:

- git
- your compiler (gcc 4.8 or above, or clang)
- Python version >2.7
- NumPy
- the zlib compression library and its corresponding development header/include files
- the Eigen template library (only consists of development header/include files); note that *MRtrix3* may not run correctly with the beta release of Eigen, so we recommend download and installation of the latest stable release
- Qt version >4.7, its corresponding development header/include files, and the executables required to compile the code. Note this will most likely be broken up into several packages, depending on how your distribution has chosen to distribute this. You will need to get those that provide these Qt modules: Core, GUI, OpenGL, SVG, and the qmake, rcc & moc executables (note these will probably be included in one of the other packages).

Warning: The compiler included in Ubuntu 12.04 and other older distributions is no longer capable of compiling *MRtrix3*, as it now requires C++11 support. The solution is to use a newer compiler as provided by the Ubuntu toolchain PPA - follow the link for instructions on how to add the PPA. Once the PPA has been added, you'll need to issue a `sudo apt-get update`, followed by `sudo apt-get install g++-4.9`. You will probably also need to tell `./configure` to use this compiler (see `./configure -help` for further options):

```
CXX=g++-4.9 ./configure
```

If this *really* doesn't work

If for whatever reasons you need to install *MRtrix3* on a system with older dependencies, and you are unable to update the software (e.g. you want to run *MRtrix3* on a centrally-managed HPC cluster), you can as a last resort use the procedure described in *Standalone installation on Linux*.

Git setup

Set up your git environment as per the Git instructions page

Build *MRtrix3*

1. Clone the *MRtrix3* repository:

   ```
   git clone https://github.com/MRtrix3/mrtrix3.git
   ```

   or if you have set up your SSH keys (for collaborators):

   ```
   git clone git@github.com:MRtrix3/mrtrix3.git
   ```

2. Configure the *MRtrix3* install:

   ```
   cd mrtrix3
   ./configure
   ```

   If this does not work, examine the 'configure.log' file that is generated by this step, it may give clues as to what went wrong.

3. Build the binaries:

   ```
   ./build
   ```

Set up *MRtrix3*

1. Update the shell startup file, so that the locations of *MRtrix3* commands and scripts will be added to your `PATH` envionment variable.

   If you are not familiar or comfortable with modification of shell files, *MRtrix3* now provides a convenience script that will perform this setup for you (assuming that you are using `bash` or equivalent interpreter). From the top level *MRtrix3* directory, run the following:

   ```
   ./set_path
   ```

2. Close the terminal and start another one to ensure the startup file is read (or just type 'bash')
3. Type `mrview` to check that everything works
4. You may also want to have a look through the *Configuration file* and set anything you think might be required on your system.

Note: The above assumes that your shell will read the `~/.bashrc` file at startup time. This is not always guaranteed, depending on how your system is configured. If you find that the above doesn't work (e.g. typing `mrview` returns a 'command not found' error), try changing step 1 to instruct the `set_path` script to update `PATH` within a different file, for example `~/.bash_profile` or `~/.profile`, e.g. as follows:

```
./set_path ~/.bash_profile
```

Keeping *MRtrix3* up to date

1. You can update your installation at any time by opening a terminal in the *MRtrix3* folder, and typing:

```
git pull
./build
```

2. If this doesn't work immediately, it may be that you need to re-run the configure script:

```
./configure
```

and re-run step 1 again.

Standalone installation on Linux

In some cases, users need to install *MRtrix3* on systems running older distributions, over which they have little or no control, for example centrally-managed HPC clusters. In such cases, there genuinely is no way to install the dependencies required to compile and run *MRtrix3*. There are two ways to address this problem: *static executables*, and the *standalone packager*. With both approaches, you can compile *MRtrix3* on a modern distro (within a virtual machine for example), package it up, and install it on any Linux system without worrying about dependencies.

Important: setting the CPU architecture

By default, `configure` will cause the build script to generate code suitable to run on your current CPU (using the `-march=native` option). This means the executables will likely *not run* on a different CPUs with different instruction sets, resulting in 'illegal instruction' runtime errors. If you intend to run *MRtrix3* on a variety of different systems with a range of CPUs, or you have no idea what the CPU is on your target system, it is safest to specify a generic architecture when configuring *MRtrix3*, before invoking `./build`. This can be done by passing an empty string for the `ARCH` environment variable, for example:

```
ARCH= ./configure
./build
```

For more specific architectures, you can provide any values from the list of specifiers understood by the compiler, for example `ARCH='sandybridge' ./configure`

Static build

The simplest approach to this problem is to build so-called static executables, which have no run-time dependencies. This can be accomplished by generating a static configuration before building the software, as follows.

First, obtain the code and extract or clone it on a modern distribution (Arch, Ubuntu 14.04, Mint 17, ..., potentially with a virtual machine if required). Then, from the main *MRtrix3* folder:

```
./build clean
git pull
ARCH=x86-64 ./configure -static [-nogui]
./build
```

Note that this requires the availability of static versions of the required libraries. This is generally not a problem, most distributions will provide those by default, with the exception of Qt. If you require a static build of MRView, you will most likely need to build a static version of Qt beforehand. Use the `-nogui` option to skip installation of GUI components, which rely on Qt.

You can then copy the contents of the `release/bin/` folder onto target systems, make sure their location is listed in the `PATH`, and start using these commands.

If you also wish to be able to use the *MRtrix3* Python scripts, you can also copy the full contents of the `scripts` directory to the target system, and append their location to the `PATH` environment variable also. However, in order for certain functionalities of these scripts to work (for instance, controlling the command-line verbosity and multi-threading of invoked *MRtrix3* commands), the relative path between the scripts and binaries must be maintained; that is, the binaries must be located in `../release/bin` relative to `scripts`. Therefore, the recommended solution is:

1. Create an `mrtrix3` directory on the target system.
2. Place the contents of `release/bin` into `mrtrix3/release/bin` on the target system.
3. Place the contents of `scripts` (including all sub-directories) into `mrtrix3/scripts` on the target system.
4. Add `mrtrix3/release/bin` and `mrtrix3/scripts` to `PATH` on the target system.

Standalone packager

In the rare cases where the *static build* procedure above doesn't work for you, *MRtrix3* now includes the `package_mrtrix` script, which is designed to package an existing and fully-functional installation from one system, so that it can be installed as a self-contained standalone package on another system. What this means is that you can now compile *MRtrix3* on a modern distro (within a virtual machine for example), package it up, and install it on any Linux system without worrying about dependencies.

Note: this is *not* the recommended way to install *MRtrix3*, and may not work for your system. This is provided on a best-effort basis, as a convenience for users who genuinely have no alternative.

What it does

The `package_mrtrix` script is included in the top-level folder of the *MRtrix3* package (if you don't have it, use `git pull` to update). In essence, all it does is collate all the dynamic libraries necessary for runtime operation into a single folder, which you can then copy over and extract onto target systems. For a truly standalone installation, you need to add the `-standalone` option, which will also include any system libraries required for runtime operation from your current system, making them available on any target system.

Limitations

- **OpenGL support:** this approach cannot magically make your system run `mrview` if it doesn't already support OpenGL 3.3 and above. This is a hardware driver issue, and can only be addressed by upgrading the drivers for your system - something that may or may not be possible.
- **GUI support:** while this approach collates all the X11 libraries that are needed to launch the program, it is likely that these will then dynamically attempt to load further libraries that reside on your system. Unfortunately, this can introduce binary compatibility issues, and cause the GUI components to abort. This might happen even if your system does have OpenGL 3.3 support. There is unfortunately no simple solution to this.
- **Installation on remote systems:** bear in mind that running the GUI components over a remote X11 connection is not possible, since the GLX protocol does not support OpenGL 3 and above. You may be able to use an OpenGL-capable VNC connection, but if that is not possible, there is little point in installing the GUI components on the remote server. The recommendation would be to configure with the `-nogui` option to skip the GUI components. You should also be able to access your data over the network (e.g. using SAMBA or SSHFS), in which case you would be able to display the images by running `mrview` locally and loading the images over the shared network drives.

Instructions

First, obtain the code and extract or clone it on a modern distribution (Arch, Ubuntu 14.04, Mint 17, ..., potentially with a virtual machine if required). Then, from the main *MRtrix3* folder:

```
./build clean
git pull
ARCH='x86-64' ./configure [-nogui]
./build
./package_mrtrix -standalone
```

Then copy the resulting `_package/mrtrix3` folder to the desired location on the target system (maybe your own home folder). To make the *MRtrix3* command available on the command-line, the `bin/` folder needs to be added to your PATH (note this assumes that you're running the BASH shell):

```
export PATH=$(pwd)/mrtrix3/release/bin:$(pwd)/mrtrix3/scripts:$PATH
```

The above will only set your PATH for the current session. To make this the default for new sessions, you should add the relevant line to your `~/.bashrc` file:

```
echo export PATH=$(pwd)/mrtrix3/release/bin:$(pwd)/mrtrix3/scripts:\$PATH >> ~/.bashrc
```

CHAPTER 3

MacOS X installation

We outline the steps for installing *MRtrix3* on MacOS X. Please consult the MRtrix3 forum if you encounter any issues with the configure, build or runtime operations of *MRtrix3*.

Check requirements

To install *MRtrix3* , you will need the following:

- a C++11 compliant compiler (e.g. clang in Xcode)
- Python version >= 2.7 (already included in MacOS X)
- The zlib compression library (already included in MacOS X)
- Eigen version 3.2 *(do not install the beta version)*
- Qt version >= 5.1 *[GUI components only]* - important: versions prior to this will *not* work

Warning: To run the GUI components of *MRtrix3* (`mrview` & `shview`), you will also need:

- an OpenGL 3.3 compliant graphics card and corresponding software driver - thankfully OpenGL 3.3 is supported across the entire MacOS X range with OS versions >= 10.9.

Note: If you currently do not plan to contribute to the Mrtrix3 code, the most convenient way to install Mrtrix3 on MacOS X is to install it via homebrew.

- If you do not have homebrew installed, you can install it via: `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
- You need to add the MRtrix3 tap to homebrew: `brew tap MRtrix3/mrtrix3`
- You can now install the latest version of MRtrix3 with: `brew install mrtrix3`

This should be all you need to do. For all installation options type `brew info mrtrix3`. MRtrix3 will get upgraded when you upgrade all homebrew packages `brew update && brew upgrade`. If you want to avoid upgrading MRtrix3 the next time you upgrade homebrew you can do so with `brew pin mrtrix3`.

Install Dependencies

1. Update MacOS X to version 10.10 (Yosemite) or higher - OpenGL 3.3 will typically not work on older versions
2. Install XCode from the Apple Store
3. Install Eigen3 and Qt5.

   There are several alternatives to do this, depending on your current system setup. The most convenient is probably to use your favorite package manager (Homebrew or MacPorts), or install one of these if you haven't already.

   If you find your first attempt doesn't work, please resist the temptation to try one of the other options: in our experience, this only leads to further conflicts, which won't help installing MRtrix3 *and* will make things more difficult to fix later. Once you pick one of these options, we strongly recommend you stick with it, and consult the community forum if needed for advice and troubleshooting.

   - With Homebrew:
     - Install Eigen3: `brew install eigen`
     - Install Qt5: `brew install qt5`
     - Install pkg-config: `brew install pkg-config`
     - Add Qt's binaries to your path: `export PATH=`brew --prefix`/opt/qt5/bin:$PATH`
   - With MacPorts:
     - Install Eigen3: `port install eigen3`
     - Install Qt5: `port install qt5`
     - Install pkg-config: `port install pkgconfig`
     - Add Qt's binaries to your path: `export PATH=/opt/local/libexec/qt5/bin:$PATH`
   - As a last resort, you can manually install Eigen3 and Qt5: You can use this procedure if you have good reasons to avoid the other options, or if for some reason you cannot get either Homebrew or MacPorts to work.
     - **Install Eigen3: download and extract the source code from eigen.tuxfamily.org** Note that *MRtrix3* does not yet fully support the Eigen 3.3 beta, and installation of this version may result in runtime issues. If performing a manual installation, download the latest stable release of Eigen3.
     - **Install Qt5: download and install the latest version from http://download.qt.io/official_releases/qt/** You need to select the file labelled `qt-opensource-mac-x64-clang-5.X.X.dmg`. Note that you need to use at least Qt 5.1, since earlier versions don't support OpenGL 3.3. We advise you to use the latest version (5.5.0 as of the last update). You can choose to install it system-wide or just in your home folder, whichever suits - just remember where you installed it.
     - **Make sure Qt5 tools are in your PATH** (edit as appropriate) `export PATH=/path/to/Qt5/5.X.X/clang_64/bin:$PATH`
     - **Set the CFLAG variable for eigen** (edit as appropriate) `export EIGEN_CFLAGS="-isystem /where/you/extracted/eigen"` Make sure *not* to include the final `/Eigen` folder in the path name - use the folder in which it resides instead!

Git setup

Set up your git environment as per the Git instructions page

Build *MRtrix3*

1. Clone the *MRtrix3* repository:

```
git clone https://github.com/MRtrix3/mrtrix3.git
```

   or if you have set up your SSH keys (for collaborators):

```
git clone git@github.com:MRtrix3/mrtrix3.git
```

2. Configure the *MRtrix3* install:

```
cd mrtrix3
./configure
```

   If this does not work, examine the 'configure.log' file that is generated by this step, it may give clues as to what went wrong.

3. Build the binaries:

```
./build
```

Set up *MRtrix3*

1. Update the shell startup file, so that the locations of *MRtrix3* commands and scripts will be added to your `PATH` envionment variable.

   If you are not familiar or comfortable with modification of shell files, *MRtrix3* now provides a convenience script that will perform this setup for you (assuming that you are using `bash` or equivalent interpreter). From the top level *MRtrix3* directory, run the following:

```
./set_path
```

2. Close the terminal and start another one to ensure the startup file is read (or just type 'bash')
3. Type `mrview` to check that everything works
4. You may also want to have a look through the *List of MRtrix3 configuration file options* and set anything you think might be required on your system.

Note: The above assumes that your shell will read the `~/.bash_profile` file at startup time. This is not always guaranteed, depending on how your system is configured. If you find that the above doesn't work (e.g. typing `mrview` returns a 'command not found' error), try changing step 1 to instruct the `set_path` script to update `PATH` within a different file, for example `~/.profile` or `~/.bashrc`, e.g. as follows:

```
./set_path ~/.profile
```

Keeping *MRtrix3* up to date

1. You can update your installation at any time by opening a Git Bash terminal in the *MRtrix3* folder, and typing:

```
git pull
./build
```

2. If this doesn't work immediately, it may be that you need to re-run the configure script:

```
./configure
```

   and re-run step 1 again.

CHAPTER 4

Windows installation

We outline the steps for installing *MRtrix3* for Windows using MSYS2. Please consult the MRtrix3 forum if you encounter any issues with the configure, build or runtime operations of *MRtrix3*.

Warning: Some of the Python scripts provided with *MRtrix3* are dependent on external software tools (for instance FSL). If these packages are not available on Windows, then the corresponding *MRtrix3* scripts also cannot be run on Windows. A virtual machine may therefore be required in order to use these particular scripts; though *MRtrix3* may still be installed natively on Windows for other tasks.

Check requirements

To install *MRtrix3*, you will need the following:

- a C++11 compliant compiler
- Python version >= 2.7
- The zlib compression library
- Eigen version 3.2 *(do not install the beta version)*
- Qt version >= 4.7 *[GUI components only]*

Note: All of these dependencies are installed below by the MSYS2 package manager.

Warning: To run the GUI components of *MRtrix3* (`mrview` & `shview`), you will also need:

- an OpenGL 3.3 compliant graphics card and corresponding software driver

Warning: When following the instructions below, use the **'MinGW-w64 Win64 shell'**; 'MSYS2 shell' and 'MinGW-w64 Win32 shell' should be avoided.

Install and update MSYS2

1. Download and install the most recent 64-bit MSYS2 installer from http://msys2.github.io/ (msys2-x86_64-*.exe).
2. Run the program **'MinGW-w64 Win64 Shell'** from the start menu.
3. Update the system packages:

```
update-core
```

Note: Future versions of MSYS2 will drop `update-core`. If your version came without `update-core`, it is probably safe to skip this step.

Warning: At time of writing, this MSYS2 system update will give a number of instructions, including: terminating the terminal when the update is completed, and modifying the shortcuts for executing the shell(s). Although these instructions are not as prominent as they could be, it is *vital* that they are followed correctly!

4. Close the shell and start **'MinGW-w64 Win64 Shell'**
5. Update the other packages:

```
pacman -Su
```

Install *MRtrix3* dependencies

1. From the **'MinGW-w64 Win64 Shell'** run:

```
    pacman -S git python pkg-config mingw-w64-x86_64-gcc mingw-w64-x86_64-eigen3␣
↪mingw-w64-x86_64-qt5

Sometimes ``pacman`` may fail to find a particular package from any of
the available mirrors. If this occurs, you can download the relevant
package from `SourceForge <https://sourceforge.net/projects/msys2/files/REPOS/
↪MINGW/x86_64/>`__:
place both the package file and corresponding .sig file into the
``/var/cache/pacman/pkg`` directory, and repeat the ``pacman`` call above.

Sometimes ``pacman`` may refuse to install a particular package, claiming e.g.::

    error: failed to commit transaction (conflicting files)
    mingw-w64-x86_64-eigen3: /mingw64 exists in filesystem
    Errors occurred, no packages were upgraded.

Firstly, if the offending existing target is something trivial that can
be deleted, this is all that should be required. Otherwise, it is possible
```

```
that MSYS2 may mistake a _file_ existing on the filesystem as a
pre-existing _directory_; a good example is that quoted above, where
``pacman`` claims that directory ``/mingw64`` exists, but it is in fact the
two files ``/mingw64.exe`` and ``/mingw64.ini`` that cause the issue.
Temporarily renaming these two files, then changing their names back after
``pacman`` has completed the installation, should solve the problem.
```

Set up git and download *MRtrix3* sources

1. Configure global settings for Git in the **'MinGW-w64 Win64 Shell'**:

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
git config --global push.default upstream
```

2. Clone the *MRtrix3* repository:

```
git clone https://github.com/MRtrix3/mrtrix3.git
```

Build *MRtrix3*

1. Configure the *MRtrix3* install:

```
cd mrtrix3
./configure
```

   If this does not work, examine the 'configure.log' file that is generated by this step, it may give clues as to what went wrong.

2. Build the binaries:

```
./build
```

Set up *MRtrix3*

1. Update the shell startup file, so that the locations of *MRtrix3* commands and scripts will be added to your `PATH` envionment variable.

   If you are not familiar or comfortable with modification of shell files, *MRtrix3* now provides a convenience script that will perform this setup for you (assuming that you are using `bash` or equivalent interpreter). From the top level *MRtrix3* directory, run the following:

```
./set_path
```

2. Close the terminal and start another one to ensure the startup file is read (or just type 'bash')
3. Type `mrview` to check that everything works
4. You may also want to have a look through the *List of MRtrix3 configuration file options* and set anything you think might be required on your system.

Note: The above assumes that your shell will read the `~/.bashrc` file at startup time. This is not always guaranteed, depending on how your system is configured. If you find that the above doesn't work (e.g. typing `mrview` returns a 'command not found' error), try changing step 1 to instruct the `set_path` script to update `PATH` within a different file, for example `~/.bash_profile` or `~/.profile`, e.g. as follows:

```
./set_path ~/.bash_profile
```

Keeping *MRtrix3* up to date

1. You can update your installation at any time by typing:

   ```
   git pull
   ./build
   ```

2. If this doesn't work immediately, it may be that you need to re-run the configure script:

   ```
   ./configure
   ```

   and re-run step 1 again.

Compiling external projects with `msys2`

In `msys2`, the `ln -s` command actually creates a *copy* of the target, *not* a symbolic link. By doing so, the build script is unable to identify the location of the MRtrix libraries when trying to compile an external module.

The simplest way around this is simply to invoke the build script of the main *MRtrix3* install directly. For example, if compiling an external project called `myproject`, residing in a folder alongside the main `mrtrix3` folder, the build script can be invoked with:

```
# current working directory is 'myproject':
../mrtrix3/build
```

If you really want a symbolic link, one solution is to use a standard Windows command prompt, with Administrator privileges: In the file explorer, go to `C:\Windows\system32`, locate the file `cmd.exe`, right-click and select 'Run as administrator'. Within this prompt, use the `mklink` command (note that the argument order passed to `mklink` is reversed with respect to `ln -s`; i.e. provide the location of the link, *then* the target). Make sure that you provide the *full path* to both link and target, e.g.:

```
mklink C:\msys64\home\username\src\my_project\build␣
↪C:\msys64\home\username\src\MRtrix3\build
```

, and `msys64` should be able to interpret the softlink path correctly (confirm with `ls -la`).

I have also found recently that the build script will not correctly detect use of a softlink for compiling an external project when run under Python2, so Python3 must be used explicitly.

CHAPTER 5

HPC clusters installation

These instructions outline a few issues specific to high-performance computing (HPC) systems.

Installing *MRtrix3*

Most HPC clusters will run some flavour of GNU/Linux and hence a cluster administrator should be able to follow the steps outlined for a *Linux installation*. In particular, if your sysadmin is able to install the required dependencies (the preferred option), you should be able to subsequently *Build MRtrix3*.

However, it is not uncommon for HPC systems to run stable, and hence relatively old distributions, with outdated dependencies. This is particularly problematic since *MRtrix3* relies on recent technologies (C++11, OpenGL 3.3), which are only available on recent distributions. There is therefore a good chance these dependencies simply cannot be installed (certainly not without a huge amount of effort on the part of your sysadmin). In such cases, one can instead attempt a *Standalone installation on Linux*. Alternatively, if you (and your sysadmin) are comfortable with installation of dependencies from source within your home directory, you can try the instructions below.

Installation of *MRtrix3* and dependencies from source

The following instructions list the steps I used to compile *MRtrix3* natively on a local HPC cluster. Replicating these instructions line-for-line may not work on another system; I'm just providing these instructions here in case they help to point somebody in the right direction, or encourage users to try a native installation rather than resorting to transferring binaries compiled on another system.

- Installing a C++11-compliant g++ from source

  Note that during this process, there will be *three* `gcc` directories created: one is for the source code (including that of some prerequisites), one is for compilation objects, and one is the target of the final installation (since you almost certainly won't be able to install this version of `gcc` over the top of whatever is provided by the HPC sysadmin).

```
svn co svn://gcc.gnu.org/svn/gcc/branches/gcc-5-branch gcc_src/
```

(*Don't* checkout the trunk `gcc` code; *MRtrix3* will currently not compile with it)

The following `gcc` dependencies will be built as part of the `gcc` compilation, provided that they are placed in the correct location within the `gcc` source directory.

```
wget https://gmplib.org/download/gmp/gmp-6.1.1.tar.bz2
tar -xf gmp-6.1.1.tar.bz2
mv gmp-6.1.1/ gcc_src/gmp/
wget ftp://ftp.gnu.org/gnu/mpc/mpc-1.0.3.tar.gz
tar -xf mpc-1.0.3.tar.gz
mv mpc-1.0.3/ gcc_src/mpc/
wget http://www.mpfr.org/mpfr-current/mpfr-3.1.4.tar.gz
tar -xf mpfr-3.1.4.tar.gz
mv mpfr-3.1.4/ gcc_src/mpfr/
```

With the following, the `configure` script (which resides within the `gcc_src` directory in this example) must *not* be executed within that directory; rather, it must be executed from an alternative directory, which will form the target location for the compilation object files. The target installation directory (set using the `--prefix` option below) must be a location for which you have write access; most likely somewhere in your home directory.

```
mkdir gcc_obj; cd gcc_obj/
../gcc_src/configure --prefix=/path/to/installed/gcc --disable-multilib
make && make install
```

- Installing Python3 from source

  My local HPC cluster provided Python version 2.6.6, which was not adequate to successfully run the `configure` and `build` scripts in *MRtrix3*. Therefore this necessitated a manual Python install - a newer version of Python 2 would also work, but downloading Python 3 should result in less ambiguity about which version is being run.

```
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar -xf Python-3.5.2.tgz
mv Python-3.5.2/ python3/
cd python3/
./configure
./make
cd ../
```

- Installing Eigen3

```
wget http://bitbucket.org/eigen/eigen/get/3.2.8.tar.gz
tar -xf 3.2.8.tar.gz
mv eigen* eigen3/
```

- Installing *MRtrix3*

  Personally I prefer to install a no-GUI version of *MRtrix3* on high-performance computing systems, and transfer files to my local system if I need to view anything; so I use the `-nogui` flag for the `configure` script.

```
git clone https://github.com/MRtrix3/mrtrix3.git
cd mrtrix3/
export CXX=/path/to/installed/gcc/bin/g++
export EIGEN_CFLAGS="-isystem /path/to/eigen3/"
export LD_LIBRARY_PATH="/path/to/installed/gcc/lib64:$LD_LIBRARY_PATH"
```

```
../python3/python configure -nogui
../python3/python build
```

If you encounter issues when running *MRtrix3* commands that resemble the following:

```
mrconvert: /usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.9' not found
(required by mrconvert)
```

This indicates that the shared library of the compiler version installed on the cluster is being found before that of the C++11-compliant compiler installed manually. The `lib64/` directory of the manually-installed `gcc` version must appear *before* that of the version installed on the cluster in the `LD_LIBRARY_PATH` environment variable.

Remote display

Most people would expect to be able to run `mrview` on the server using X11 forwarding. Unfortunately, this will not work without some effort - please refer to *Remote display issues* for details.

Configuration

There are a number of parameters that can be set in the configuration file that are highly relevant in a HPC environment, particularly when the user's home folder is stored over a network-based filesystem (as is often the case). The *MRtrix3* configuration file is located either system-wide in `/etc/mrtrix.conf`, and/or in each user's home folder in `~/.mrtrix.conf`. Entries consist of `key: value` entries, one per line, stored as ASCII text.

- **NumberOfThreads** (default: hardware concurrency, as reported by the system): by default, *MRtrix3* will use as many threads as the system reports being able to run concurrently. You may want to change that number to a lower value, to prevent *MRtrix3* from taking over the system entirely. This is particularly true if you anticipate many users running many *MRtrix3* commands concurrently.
- **TmpFileDir** (default: '/tmp'): any image data passed from one *MRtrix3* command to the next using a Unix pipeline is actually stored in a temporary file, and its filename passed to the next command. While this is fine if the filesystem holding the temporary file is locally backed and large enough, it can cause significant slowdown and bottlenecks if it resides on a networked filesystems, as the temporary file will most likely need to be transferred in its entirety over the network and back again. Also, if the filesystem is too small, *MRtrix3* commands may abort when processing large files. In general, the `/tmp` folder is likely to be the most appropriate (especially if mounted as tmpfs). If however it is not locally mounted, or too small, you may want to set this folder to some other more suitable location.
- **TrackWriterBufferSize** (default: 16777216). When writing out track files, *MRtrix3* will buffer up the output and write out in chunks of 16MB, to limit the frequency of write() calls and the amount of IO requests. More importantly, when several instances of *MRtrix3* are generating tracks concurrently and writing to the same filesystem, frequent small writes will result in massive fragmentation of the output files. By setting a large buffer size, the chances of writes being concurrent is reduced drastically, and the output files are much less likely to be badly fragmented. Note that fragmentation can seriously affect the performance of subsequent commands that need to read affected data. Depending on the type of operations performed, it may be beneficial to use larger buffer sizes, for example 256MB. Note that larger numbers imply greater RAM usage to hold the data prior to write-out, so it is best to keep this much smaller than the total RAM capacity.

CHAPTER 6

Key features

While *MRtrix3* is primarily intended to be used for the analysis of diffusion MRI data, at its fundamental level it is designed as a general-purpose library for the analysis of *any* type of MRI data. As such, it provides a back-end to simplify a large number of operations, many of which will be invisible to the end-user. Specifically, *MRtrix3* features:

- a *consistent command-line interface*, with inline documentation for each command
- universal import/export capabilities when *accessing image data* across all *MRtrix3* applications.
- *Multi-file numbered image support* to load multiple images as a single multi-dimensional dataset
- efficient use of *Unix Pipelines* for complex workflows
- high performance on modern multi-core systems, with multi-threading used extensively throughout *MRtrix3*;
- available on all common modern operating systems (GNU/Linux, MacOSX, Windows);
- a consistent *Coordinate system* with most operations performed in scanner/world coordinates where possible.

CHAPTER 7

Configuration file

The behaviour of a number of aspects of *MRtrix3* can be controlled by the user via the *MRtrix3 configuration file*. Note, that this file is distinct from the build configuration file that is generated as part of the *MRtrix3* installation, but rather is used to specify default settings for a number of parameters, many of which relate to data visualisation when using `mrview`.

For all available configurable options, please refer to the configuration file options page.

Location

MRtrix3 applications will attempt to read configuration information from a two locations. The system-wide configuration file `/etc/mrtrix.conf` is read first if present, followed by the user-specific configuration `~/.mrtrix.conf`. If both system and user-specific configuration files exist, the parameters specified in the two configuration files will be aggregated, with user-specified configuration options taking precedence in the case of a conflict. In the case that a particular configuration parameter is not defined, *MRtrix3* will resort to hard-coded defaults.

Format

The configuration files are text files, with each line containing a key: value pair. For example

```
Analyse.LeftToRight: false
NumberOfThreads: 2
```

Note: Key names are case-sensitive.

The value entry may be interpreted by *MRtrix3* applications as either:

- `Boolean`: allowed values here are true or false
- `Integer`: any integer value

- `Floating-point`: any floating-point value
- `Text`: any text string, without any further interpretation

The list of all configuration file options can be found here.

CHAPTER 8

Images and other data

Format handling in *MRtrix3*

MRtrix3 provides a flexible data input/output back-end in the shared library, which is used across all applications. This means that all applications in *MRtrix3* can read or write images in all the supported formats - there is no need to explicitly convert the data to a given format prior to processing.

However, some specialised applications may expect additional information to be present in the input image. The MRtrix .mif/.mih formats are both capable of storing such additional information data in their header, and will hence always be supported for such applications. Most image formats however cannot carry additional information in their header (or at least, not easily) - this is in fact one of the main motivations for the development of the MRtrix image formats. In such cases, it would be necessary to use MRtrix format images. Alternatively, it may be necessary to provide the additional information using command-line arguments (this is the case particularly for the DW gradient table, when providing DWI data in NIfTI format for instance).

Image file formats are recognised by their file extension. One exception to this is DICOM: if the filename corresponds to a folder, it is assumed to contain DICOM data, and the entire folder will be scanned recursively for DICOM images.

It is also important to note that the name given as an argument will not necessarily correspond to an actual file name on disk: in many cases, images may be split over several files. What matters is that the text string provided as the *image specifier* is sufficient to unambiguously identify the full image.

Coordinate system

All *MRtrix3* applications will consistently use the same coordinate system, which is identical to the NIfTI standard. Note that this frame of reference differs from the DICOM standard (typically the x & y axis are reversed). The convention followed by *MRtrix3* applications is as follows:

dimensional	description
0 (x)	increasing from left to right
1 (y)	increasing from posterior to anterior
2 (z)	increasing from inferior to superior

All coordinates or vector components supplied to *MRtrix3* applications should be provided with reference to this coordinate system.

Multi-file numbered image support

It is possible to access a numbered series of images as a single multi-dimensional dataset, using a syntax specific to MRtrix. For example:

```
$ mrinfo MRI-volume-[].nii.gz
```

will collate all images that match the pattern `MRI-volume-<number>.nii.gz`, sort them in ascending numerical order, and access them as a single dataset with dimensionality one larger than that contained in the images. In other words, assuming there are 10 `MRI-volume-0.nii.gz` to `MRI-volume-9.nii.gz`, and each volume is a 3D image, the result will be a 4D dataset with 10 volumes.

Note that this isn't limited to one level of numbering:

```
$ mrconvert data-[]-[].nii combined.mif
```

will collate all images that match the `data-number-number.nii` pattern and generate a single dataset with dimensionality two larger than its constituents.

Finally, it is also possible to explicitly request specific numbers, using *Number sequences and floating-point lists* within the square brackets:

```
$ mrconvert data-[10:20].nii combined.mif
```

Data types

MRtrix3 applications can read and write data in any of the common data types. Many *MRtrix3* commands also support the `-datatype` option to specify the data type for the output image. For example:

```
$ mrconvert DICOM_images/ -datatype float32 output.nii
```

Note: Not all image formats support all possible datatypes. The MRtrix image file formats are designed to handle all of the possibilities listed below, while other image formats may only support a subset. When a data type is requested that isn't supported by the image format, a hopefully suitable alternative data type will be used instead.

Below is a list of the supported data types and their specifiers for use on the command-line. Note that *MRtrix* is not sensitive to the case of the specifier: `uint16le` will work just as well as `UInt16LE`.

Specifier	Description
Bit	bitwise data
Int8	signed 8-bit (char) integer
UInt8	unsigned 8-bit (char) integer
Int16	signed 16-bit (short) integer (native endian-ness)
UInt16	unsigned 16-bit (short) integer (native endian-ness)
Int16LE	signed 16-bit (short) integer (little-endian)
UInt16LE	unsigned 16-bit (short) integer (little-endian)
Int16BE	signed 16-bit (short) integer (big-endian)
UInt16BE	unsigned 16-bit (short) integer (big-endian)
Int32	signed 32-bit int (native endian-ness)
UInt32	unsigned 32-bit int (native endian-ness)
Int32LE	signed 32-bit int (little-endian)
UInt32LE	unsigned 32-bit int (little-endian)
Int32BE	signed 32-bit int (big-endian)
UInt32BE	unsigned 32-bit int (big-endian)
Float32	32-bit floating-point (native endian-ness)
Float32LE	32-bit floating-point (little-endian)
Float32BE	32-bit floating-point (big-endian)
Float64	64-bit (double) floating-point (native endian-ness)
Float64LE	64-bit (double) floating-point (little-endian)
Float64BE	64-bit (double) floating-point (big-endian)
CFloat32	complex 32-bit floating-point (native endian-ness)
CFloat32LE	complex 32-bit floating-point (little-endian)
CFloat32BE	complex 32-bit floating-point (big-endian)
CFloat64	complex 64-bit (double) floating-point (native endian-ness)
CFloat64LE	complex 64-bit (double) floating-point (little-endian)
CFloat64BE	complex 64-bit (double) floating-point (big-endian)

The image transfom

The orientation of the image with respect to the scanner axes is determined by the combination of the *image axes* and the *location of the corner voxel*. This information is encapsulated in the *transformation matrix*, commonly referred to simply as the *transform*. You can view the transform for any image using *mrinfo*, for example:

```
$ mrinfo dwi.mif
************************************************
Image:               "dwi.mif"
************************************************
  Dimensions:        104 x 104 x 54 x 167
  Voxel size:        2.30769 x 2.30769 x 2.3 x ?
  Data strides:      [ -1 -2 3 4 ]
  Format:            MRtrix
  Data type:         unsigned 16 bit integer (little endian)
  Intensity scaling: offset = 0, multiplier = 1
  Transform:               0.9999   6.887e-09    -0.01564      -116.1
                        -0.001242      0.9968    -0.07943      -89.44
                          0.01559     0.07944      0.9967      -64.27
  comments:          TOURNIER DONALD (BRI) [MR] diff60_b3000_2.3_iPat2+ADC
                     study: BRI_Temp_backup Donald
                     DOB: 09/03/1977
                     DOS: 03/10/2007 15:58:40
  dw_scheme:         [ 167 entries ]
```

The 'Transform' field above shows the first 3 rows of the transformation matrix (technically, this is a 4×4 matrix, but the last row is always set to `[ 0 0 0 1 ]`). The first 3 columns correspond to the *x*, *y* & *z* image axes respectively, while the last column corresponds to the location *in real (scanner/world) space* of the corner voxel (i.e. the voxel at index `[ 0 0 0 ]`).

In *MRtrix3*, the transform shown always corresponds to the transformation from image coordinates *in millimeters* to scanner coordinates *in millimeters* - the voxel size is not taken into account, and the image axes are always normalised to unit amplitude. This may differ from other packages.

Furthermore, *MRtrix3* will always present the transform that best matches the real space. If the transform of the image on file represents a large rotation, such that for example the image *x* axis is closer to the scanner's *z* axis, this transform will be modified by permutation or inversion of the axes to bring it in alignment with the expected coordinate system, so that the first axis genuinely can be interpreted as approximately left-right, etc. To achieve this, *MRtrix3* will also modify the image *Strides* to match.

Strides

A file is simply a linear array of values. Image data on the other hand are multidimensional arrays. The image values can therefore be ordered on file in many different ways. For example, we could start from the voxel at the left posterior inferior corner of the image, and store intensity values in order of traversal towards the *right*. Once the other end of the image is reached, we repeat the process for the row of values *anterior* to the last one, and repeat until the end of the slice. At this point, we store the slice *superior* to the last one, until the whole image has been stored. This ordering scheme is what is typically used in the NIfTI standard, and is commonly referred to as RAS (right anterior posterior), referring to the direction of traversal of each axis in turn. This scheme is also often referred to as *neurological*, although this term is in general much more ambiguous.

However, this is only a convention, and many other combinations are possible. For instance, it is possible to start from the *right* posterior inferior corner, and raster through along the *left* direction, then store the next row along the anterior direction, and finally the next slice in the superior direction. This scheme is what is normally used in the now deprecated Analyse format, and is commonly referred to as LAS or *radiological*.

Of course, there are many more possibilities. For instance, sagittal DICOM images will typically be stored using a PIR (posterior inferior right) order, since each sagittal slice is stored in order, etc. *MRtrix3* applications are agnostic to the order of storage, and can handle any such images provided the format is clear about what the order is.

In *MRtrix3*, the order of storage is defined by their *strides*. These refer to the number of voxels between a given voxel and the next voxel along a given dimension. For instance, in a 128×128×128 image stored using RAS ordering, the strides would be `1,128,16384`: the next voxel along the *x* axis is just one voxel away, while the next voxel along the *y* axis is 128 values away (i.e. a whole row of *x* values), and so on. In contrast, if stored in LAS order, the strides would be `-1,128,16384`, indicating that the next voxel along the *x* axis would actually be stored one value *before* the current one.

To simplify the specification of these strides, *MRtrix3* typically expects and provides *symbolic* strides. For example, the RAS strides above would be expressed as `1,2,3`, since this is sufficient to deduce the actual strides once the image dimensions are known. Likewise, LAS would correspond to strides of `-1,2,3`, PIR to `3,-1,-2`, etc. This has the advantage that the specification of the strides is then independent of the image dimensions.

Using strides to specify ordering also allows the specification to generalise to arbitrary dimensions. For example, it is fairly common for *MRtrix3* applications to request their output for 4D images to be written with strides `2,3,4,1` (if the image format supports it): this corresponds to a volume-contiguous order, whereby the values for all volumes of a given voxel are written next to each other on file; this often has performance advantages for applications that need to process all values for a given voxel concurrently (as is often the case in diffusion MRI), by allowing the hardware to make better use of resources (tractography is one such example).

Many *MRtrix3* commands accept the `-stride` option, which is used to specify the strides for the output image. For example, to generate a LAS (radiological) NIfTI image for use with FSL (along with the corresponding bvecs/bvals), you can use *mrconvert* along with the `-stride -1,2,3,4` option:

```
$ mrconvert dwi.mif -stride -1,2,3,4 -export_grad_fsl bvecs bvals dwi.nii
```

Likewise, if you need to ensure the orientation is neurological (RAS), you can specify strides `1,2,3,4` (or use the `1:4` shorthand). You can also specify other combinations if required: for example `-stride -2,-1,3,4` would correspond to a PLS coordinate system, `-stride 2,3,4,1` would correspond to volume-contiguous storage (with RAS for the spatial axes), etc.

The different formats supported by *MRtrix3* differ in the range of strides that they support. The *MRtrix image formats (.mih / .mif)* are the only formats to support arbitrary combinations.

Note: Not all image formats support all possible datatypes. The *MRtrix image formats (.mih / .mif)* are designed to handle arbitrary strides, while other image formats may only support a limited subset. When strides are requested that are not supported by the image format, a hopefully suitable alternative will be used instead.

Interaction between strides and transform

There is an interaction between the strides and the image transform: if the transform matrix corresponds to a 90° rotation, this can be viewed as changing the *strides* without affecting the transform. Such a large rotation has changed the order of storage relative to the anatomical labels typically used to refer to the ordering (e.g. RAS, LAS, etc). For example, if a RAS image is modified such that its transform rotates the image axes by 90° around the *y* axis, this in effect implies that voxels are now ordered IAR (i.e. *right* becomes *inferior*, *anterior* remains as-is, and *superior* becomes *right*).

The *MRtrix3* back-end will indeed interpret such large rotations as affecting the strides, so that if the strides are stated as `1,2,3`, the order of storage will always be left->right, posterior->anterior, inferior->superior *relative to the scanner axes*. Note that this also implies that the transform matrix will always be modified as necessary to bring it close to the standard coordinate system, so that the first image axis is close to the *x* axis, etc. This allows *MRtrix3* applications to operate on images in the knowledge that these axes are always anatomically as expected, without worrying about the details of *how* this information was actually stored on file.

It is important to bear this in mind when interpreting for output of *mrinfo* for example, since this produces the strides and transform *as interpreted by MRtrix3*, rather than those actually stored on file - although the two representations should be strictly equivalent. If you need to inspect the information as stored on file, use *mrinfo*'s `-norealign` option.

Supported formats

This lists the various file formats currently supported by *MRtrix3*.

MRtrix image formats (`.mih / .mif`)

These MRtrix-specific image formats are closely related. They consist of a text header, with data stored in binary format, either within the same file (.mif) or as one or more separate files (.mih). In both cases, the header structure is the same, as detailed below. These file formats were devised to address a number of limitations inherent in currently available formats. In particular:

- simplicity: as detailed below, the header format is deliberately kept very simple and human-readable, making it easy to debug and edit manually if needed.
- extendability: any information can be stored in the header, and will simply be ignored by the application if not recognised.

- arbitrary data organisation: voxel values can be stored in any order, making it simple to ensure for example that all FOD coefficients for a given voxel are stored contiguously on file.

Note that *MRtrix3* now includes *MatLab* functions to read and write MRtrix image files, and to load MRtrix tracks files. These are located in the `matlab` subfolder.

Compressed MRtrix image format (`.mif.gz`)

MRtrix3 also supports the compressed version of the single-file `.mif` format, both for reading and writing.

Note: While this can reduce file sizes, it does incur a runtime cost when reading or writing the image (a process that can often take longer than the operation to be performed), and will require the entire image to be loaded uncompressed into RAM (*MRtrix3* can otherwise make use of memory-mapping to keep RAM requirements to a minimum). For large files, these costs can become considerable; you may find that *MRtrix3* can process a large uncompressed image, yet run out of RAM when presented with the equivalent compressed version (in such cases, you can try using `gunzip` to uncompress the file manually before invoking the relevant *MRtrix3* command).

Header structure

The header is the first (and possibly only) data stored in the file, as ASCII-encoded text (although other encodings such as UTF8 may work equally well). Lines should be separated by Unix-style newlines (line-feed, ‘’, ASCII 0x0A), although MRtrix will also accept DOS-type newlines.

The first line should read only `mrtrix image` to indicate that this is an image in MRtrix format. The last line of the header should read only `END` to signal the end of the header, after which all data will be considered as binary.

All lines *between* these two entries must be represented as key-value pairs, as described below.

Header key-value pairs

All following lines are in the format `key:  value`, with the value entry extending up to the end of the line. All whitespace characters before and after the value entry are ignored. Some keys are required to read the images, others are optional (sensible defaults will be substituted if they are absent). Recognised keys are provided in the list below, along with the expected format of the corresponding values.

- **dim** [required]

  the image dimensions, supplied as a comma-separated list of integers. The number of entries specifies the dimensionality of the image. For example: `dim:  192,256,256` specifies a 192×256×256 image.

- **vox** [required]

  the voxel size along each dimension, as a comma-separated list of floating-point values. The number of entries should match that given in the dim entry. For example: `vox:  0.9,0.898438,0.898438`.

- **layout** [required]

  specifies the organisation of the data on file. In simplest terms, it provides a way of specifying the strides required to navigate the data file, in combination with the dim entry. It is given as a comma-separated list of signed integers, with the sign providing the direction of data traversal with respect to voxel coordinates, and the value providing a way of specifying the order of increasing stride.

  For example, assuming an image with `dim:  192,256,256`, the entry `layout:  +2,-0,-1` is interpreted as: the shortest stride is along the y-axis (second entry), then the z-axis (third entry), and then along the x-axis. Voxels are stored in the order left to right (positive stride) along the x-axis; anterior to posterior along the

y-axis (negative stride); and superior to inferior (negative stride) along the z-axis. Given the image dimensions, the final strides are therefore 256×256=65536 for adjacent voxels along the x-axis, -1 for the y-axis, and -256 for the z-axis. This also implies that the voxel at coordinate [0 0 0] is located 65536 voxel values into the data portion of the file.

- **datatype** [required]

  the datatype used to store individual voxel values. See the listing of valid *Data types*. For example: `datatype: UInt16LE`

- **file** [required]

  specifies where the binary image data are stored, in the format file: filename offset, with the offset provided in bytes from the beginning of the file. For example: `file: image.dat 0`.

  For the single-file format (.mif), the filename should consists of a single full-stop ('.') to indicate the current file, and the offset should correspond to a point in the file after the END statement of the header.

  For the separate header/data format (.mih), the filename should refer to an existing file in the same folder as the header (.mih) file. Multiple such entries can be supplied if the data are stored across several files.

- **transform** [optional]

  used to supply the 4×4 transformation matrix specifying the orientation of the axes with respect to real space. This is supplied as a comma-separated list of floating-point values, and only the first 12 such values will be used to fill the first 3 rows of the transform matrix. Multiple such entries can be provided to fill the matrix; for example, *MRtrix3* will normally produce 3 lines for the transform, with one row of 4 values per entry:

```
transform: 0.997986,-0.0541156,-0.033109,-74.0329
transform: 0.0540858,0.998535,-0.00179436,-100.645
transform: 0.0331575,2.34007e-08,0.99945,-125.84
```

- **scaling** [optional]

  used to specify how intensity values should be scaled, provided as an offset and scale. Voxel values will be read as value_returned = offset + scale * value_read. For example: `scaling: -1,2`. Default is `0,1` (no modification).

In addition to these keys, it is also possible to store additional key-value pairs within the header of these image files. If a particular key is not recognised by *MRtrix3*, it is simply ignored (but may be carried over to any outputs resulting from the command, depending on the particular command).

There are some keys that are utilized by particular *MRtrix3* commands in order to preserve important information as image data are passed between commands. A prominent example is `dw_scheme`, which is used to embed the diffusion gradient table within the *Image header*.

MRtrix sparse image formats (`.msh / .msf`)

These new image formats are designed for applications where the number of discrete elements within a voxel may vary between voxels. The most likely use case here is where each voxel contains some number of discrete fibre populations ('fixels'), and some information associated with each of these elements must be stored. Since only as many elements are as required for any particular voxel are actually stored, rather than having to store the maximum possible number for all voxels and padding with empty data, the format is referred to as 'sparse'.

Much like the standard MRtrix image formats (.mif and .mih), there are two different image file extensions available. One (.msh) separates the image header information and raw data into separate files, while the other (.msf) encodes all information relevant to the image into a single file.

However unlike these established formats, sparse images contain *two* separate raw data fields. The first of these behaves identically to standard images: a single intensity value for every image element. The second stores sparse

image data. For any particular image element, the intensity value within the standard image field defines a *pointer* to a location within the sparse image field, where the sparse data relevant for that image element can be found.

Additional image header features

These image formats have some features within the image header that differ from the standard MRtrix image formats:

- The 'magic number' that appears at the start of the file must read 'mrtrix sparse image'.
- Key:value pair 'sparse_data_name' defines the *name* of the class used in the sparse data field. This class name is typically not reader-friendly; the value that appears is that provided by the C++ call `typeid(XYZ).name()` for a class called XYZ. This is necessary to ensure that the data stored in the sparse field can be interpreted correctly.
- Key:value pair 'sparse_data_size' defines the size (in bytes) of the class used to store the sparse data.
- The 'datatype' field MUST be a 64-bit integer, with the same endianness as the system. A 64-bit integer type is required because the standard image data provides pointers to the sparse data in memory, while the endianness is tested to ensure that the sparse data can be interpreted correctly. Note that sparse images cannot be transferred and used between systems with different endianness.
- In addition to the 'file' key, a second key 'sparse_file' is also required, which provides the path to the beginning of the sparse image data. In the .msf format, this provides an offset from the start of the file to the start of the sparse data field; in the .msh format, a second associated data file with the extension .sdat is generated on image creation, and the path to this file is defined in the header.

Sparse data storage

Within the sparse data field, there is no delimiting information or identifying features; the image format relies on the integers stored in the standard image field to provide offset pointers to appropriate locations within the sparse field.

From the data position defined by such an offset, the first 4 bytes provide a 32-bit integer (with native endianness), which specifies the number of discrete elements stored. This is followed by data to fill precisely that number of instances of the sparse data class. Note that no endianness conversion can be performed on this data; data is read and written using a straight memory copy.

DICOM (folder or `.dcm`)

DICOM format is only supported for reading. *MRtrix3* applications will assume an image is in DICOM format if the image specifier provided corresponds to a folder or ends with the `.dcm` extension. For a folder, the application will scan the entire folder and its subfolders for DICOM files and generate a list of DICOM patients, studies and series. If a single series is found within the folder, this data set will be accessed with no further interaction required. Otherwise, the user will be prompted to select the series of interest. *MRtrix3* supports data from all major manufacturers, including Siemens mosaics and the newer single-file multi-frame format.

A separate application, *dcminfo*, is provided to view all DICOM header elements within a particular DICOM file, including Siemens' custom shadow attributes (CSA).

Note that no support is provided for reading the `DICOMDIR` entry due to case-sensitivity issues. DICOM data are typically stored on CD or DVD on a case-insensitive filesystem. However, Unix systems will typically not access these filesystems in a case-insensitive manner, and will fail to find the appropriate files if the case of filenames supplied in the DICOMDIR file does not match the case of the files found on the CD or DVD.

NIfTI (`.nii`)

This file format is supported both for reading and writing, and allows interoperation with other packages such as SPM or FSL.

Note: if both qform and sform orientation fields are present, the qform fields are ignored. Obviously, the qform fields will be used if they are present on their own.

Compressed NIfTI (`.nii.gz`)

MRtrix3 also supports compressed NIfTI images both for reading and writing.

Note: While this can reduce file sizes, it does incur a runtime cost when reading or writing the image (a process that can often take longer than the operation to be performed), and will require the entire image to be loaded uncompressed into RAM (*MRtrix3* can otherwise make use of memory-mapping to keep RAM requirements to a minimum). For large files, these costs can become considerable; you may find that *MRtrix3* can process a large uncompressed image, yet run out of RAM when presented with the equivalent compressed version (in such cases, you can try using `gunzip` to uncompress the file manually before invoking the relevant *MRtrix3* command).

FreeSurfer formats (`.mgh / .mgz`)

MRtrix3 supports both of these formats for reading and writing.

Analyse format (`.img / .hdr`)

This file format is supported both for reading and writing. However, when writing, the newer NIfTI standard will be used, since the Analyse format cannot store crucial information such as the image transform, and is hence deprecated. If these images are actually stored as NIfTI, they will be handled appropriately according to the standard.

Note: In order to specify an Analyse format image on the command line, type the name of its *data* file (`*.img`), *not* the header file.

Warning: By default, Analyse format images will be assumed to be stored using RAS (radiological) convention. This can modified in the *Configuration file* file, by setting the `Analyse.LeftToRight` entry to `true`.

Tracks file format (`.tck`)

The format for track files is similar to that for *MRtrix image formats (.mih / .mif)*. It consists of a text header in the same `key: value` format, ending with a single 'END' statement, and followed by binary data.

The first line of the header should read `mrtrix tracks` to indicate that the file contains tracks in MRtrix format. Further `key: value` pairs typically provide information about the parameters used to produce the tracks, and for the most part are not required to read the data. The only required keys are the following:

- **file**

  a `file: . offset` entry is required to specify the byte offset from the beginning of the file to the start of the binary track data. At this stage, only the single-file format is supported - in other words the filename part must be specified as '.' (see above for details).

- **datatype**

  specifies the datatype (and byte order). At this points only the Float32 data type is supported, either as little-endian (LE) or big-endian (BE).

  The binary track data themselves are stored as triplets of floating-point values (at this stage in 32 bit floating-point format), one per vertex along the track. Tracks are separated using a triplet of NaN values. Finally, a triplet of Inf values is used to indicate the end of the file.

CHAPTER 9

Command-line usage

MRtrix3 generally follows a relatively standard Unix syntax, namely:

```
$ command [options] argument1 argument2 ...
```

If you need to become familiar with using the command-line, there are plenty of tutorials online to get you started. There are however a few notable features specific to *MRtrix3*, which are outlined below.

Using short option names

Options do not need to be provided in full, as long as the initial part of the option provided is sufficient to unambiguously identify it. For example:

```
$ mrconvert -debug in.mif out.nii.gz
```

is the same as:

```
$ mrconvert -de in.mif out.nii.gz
```

but will conflict with the `-datatype` option if shortened any further:

```
$ mrconvert -d in.mif out.nii.gz
mrconvert: [ERROR] several matches possible for option "-d": "-datatype, "-debug"
```

Ordering of options on the command-line

Options can typically occur anywhere on the command-line, in any order - they do not usually need to precede the arguments. However, there are a few commands where the order does matter, particularly `mrcalc`.

Number sequences and floating-point lists

Options often expect arguments in the form of *number sequences* or *floating-point lists of numbers*. The former consists or a series of integers separated by commas or colons (no spaces), with colons indicating a range, optionally with an increment (if different from 1). For example:

- `1,4,8` becomes `[ 1 4 8 ]`
- `3,6:12,2` becomes `[ 3 6 7 8 9 10 11 12 2 ]`
- `1:3:10,8:2:0` becomes `[ 1 4 7 10 8 6 4 2 0 ]`

Note that the sign of the increment does not matter, it will always run in the direction required.

Likewise, floating-point lists consist of a comma-separated list of numbers, for example:

- `2.47,-8.2223,1.45e-3`

Unix Pipelines

The output of one program can be fed straight through to the input of another program via Unix pipes in a single command. The appropriate syntax is illustrated in this example:

```
$ dwi2tensor /data/DICOM_folder/ - | tensor2metric - -vector ev.mif
dwi2tensor: [done] scanning DICOM folder "/data/DICOM_folder/"
dwi2tensor: [100%] reading DICOM series "ep2d_diff"...
dwi2tensor: [100%] reformatting DICOM mosaic images...
dwi2tensor: [100%] loading data for image "ACME (hm) [MR] ep2d_diff"...
dwi2tensor: [100%] estimating tensor components...
tensor2metric: [100%] computing tensor metrics...
```

This command will execute the following actions:

1. `dwi2tensor` will load the input diffusion-weighted data in DICOM format from the folder `/data/DICOM_folder/` and compute the corresponding tensor components. The resulting data set is then fed into the pipe.
2. `tensor2metric` will access the data set from the pipe, generate an eigenvector map and store the resulting data set as `ev.mif`.

The two stages of the pipeline are separated by the `|` symbol, which indicates to the system that the output of the first command is to be used as input for the next command. The image that is to be fed to or from the pipeline is specified for each program using a single dash − where the image would normally be specified as an argument.

For this to work properly, it is important to know which arguments each program will interpret as input images, and which as output images. For example, this command will fail:

```
dwi2tensor - /data/DICOM_folder/ | tensor2metric - ev.mif
```

In this example, `dwi2tensor` will hang waiting for input data (its first argument should be the input DWI data set). This will also cause `tensor2metric` to hang while it waits for `dwi2tensor` to provide some input.

Advanced pipeline usage

Such pipelines are not limited to two programs. Complex operations can be performed in one line using this technique. Here is a longer example:

```
$ dwi2tensor /data/DICOM_folder/ - | tensor2metric - -vector - | mrcalc -
mask.nii -mult - | mrview -
dwi2tensor: [done] scanning DICOM folder "/data/DICOM_folder/"
dwi2tensor: [100%] reading DICOM series "ep2d_diff"...
dwi2tensor: [100%] reformatting DICOM mosaic images...
dwi2tensor: [100%] loading data for image "ACME (hm) [MR] ep2d_diff"...
dwi2tensor: [100%] estimating tensor components...
tensor2metric: [100%] computing tensor metrics...
mrcalc: [100%] computing: (/tmp/mrtrix-tmp-VihKrg.mif * mask.nii) ...
```

This command will execute the following actions:

1. `dwi2tensor` will load the input diffusion-weighted data in DICOM format from the folder /data/DICOM_folder/ and compute the corresponding tensor components. The resulting data set is then fed into the pipe.
2. `tensor2metric` will access the tensor data set from the pipe, generate an eigenvector map and feed the resulting data into the next stage of the pipeline.
3. `mrcalc` will access the eigenvector data set from the pipe, multiply it by the image mask.nii, and feed the resulting data into the next stage of the pipeline.
4. `mrview` will access the masked eigenvector data set from the pipe and display the resulting image.

How is it implemented?

The procedure used in *MRtrix3* to feed data sets down a pipeline is somewhat different from the more traditional use of pipes. Given the large amounts of data typically contained in a data set, the 'standard' practice of feeding the entire data set through the pipe would be prohibitively inefficient. *MRtrix3* applications access the data via memory-mapping (when this is possible), and do not need to explicitly copy the data into their own memory space. When using pipes, *MRtrix3* applications will simply generate a temporary file and feed its filename through to the next stage once their processing is done. The next program in the pipeline will then simply read this filename and access the corresponding file. The latter program is then responsible for deleting the temporary file once its processing is done.

This implies that any errors during processing may result in undeleted temporary files. By default, these will be created within the `/tmp` folder (on Unix, or the current folder on Windows) with a filename of the form `mrtrix-tmp-XXXXXX.xyz` (note this can be changed by specifying a custom `TmpFileDir` and `TmpFilePrefix` in the *Configuration file*). If a piped command has failed, and no other *MRtrix* programs are currently running, these can be safely deleted.

Really advanced pipeline usage

As implemented, *MRtrix3* commands treat image file names that start with the `TmpFilePrefix` (default is `mrtrix-tmp-`) as temporary. When reading the image name from the previous stage in the pipeline, the image file name will trivially match this. But this also means that it is possible to provide such a file as a normal *argument*, and it will be treated as a temporary *piped* image. For example:

```
$ mrconvert /data/DICOM/ -datatype float32 -
mrconvert: [done] scanning DICOM folder "/data/DICOM/"
mrconvert: [100%] reading DICOM series "ep2d_diff"...
mrconvert: [100%] reformatting DICOM mosaic images...
mrconvert: [100%] copying from "ACME (hm) [MR] ep2d_diff" to "/tmp/mrtrix-tmp-zcD1nr.
↪mif"...
/tmp/mrtrix-tmp-zcD1nr.mif
```

Notice that the name of the temporary file is now printed on the terminal, since the command's stdout has not be piped into another command, and we specified - as the second argument. You'll also see this file is now present in the `/tmp` folder. You can use this file by copy/pasting it as an *argument* to another *MRtrix* command (be careful though, it will be deleted once this command exits):

```
$ mrstats /tmp/mrtrix-tmp-zcD1nr.mif
        channel          mean       median    std. dev.          min          max      ␣
↪ count
         [ 0 ]       1053.47           96      1324.71            0         3827      ␣
↪ 506880
         [ 1 ]       173.526           84      140.645            0          549      ␣
↪ 506880
...
```

This allows for a non-linear arrangement of pipelines, whereby multiple pipelines can feed into a single command. This is achieved by using the shell's output capture feature to insert the temporary file name of one pipeline as an argument into a second pipeline. In BASH, output capture is achieved using the `$(commands)` syntax, or equivalently using backticks: `` `commands` ``. For example:

```
$ dwi2tensor /data/DICOM/ - | tensor2metric - -mask $(dwi2mask /data/DICOM/ - |␣
↪maskfilter - erode -npass 3 - ) -vec ev.mif -fa - | mrthreshold - -top 300 highFA.
↪mif
dwi2mask: [done] scanning DICOM folder "/data/DICOM/"
dwi2tensor: [done] scanning DICOM folder "/data/DICOM/"
dwi2mask: [100%] reading DICOM series "ep2d_diff"...
dwi2tensor: [100%] reading DICOM series "ep2d_diff"...
dwi2mask: [100%] reformatting DICOM mosaic images...
dwi2tensor: [100%] reformatting DICOM mosaic images...
dwi2mask: [100%] loading data for image "ACME (hm) [MR] ep2d_diff"...
dwi2tensor: [100%] loading data for image "ACME (hm) [MR] ep2d_diff"...
dwi2mask: [100%] finding min/max of "mean b=0 image"...
dwi2mask: [done] optimising threshold...
dwi2mask: [100%] thresholding...
dwi2tensor: [100%] estimating tensor components...
dwi2mask: [100%] finding min/max of "mean b=1000 image"...
dwi2mask: [done] optimising threshold...
dwi2mask: [100%] thresholding...
dwi2mask: [done] computing dwi brain mask...
maskfilter: [100%] applying erode filter to image -...
tensor2metric: [100%] computing tensor metrics...
mrthreshold: [100%] thresholding "/tmp/mrtrix-tmp-UHvhc2.mif" at 300th top voxel...
```

In this one command, we asked the system to perform this non-linear pipeline:

```
              dwi2tensor \
                         |--> tensor2metric  ---> mrthreshold
dwi2mask ---> maskfilter /
```

More specifically:

1. `dwi2tensor` will load the input diffusion-weighted data in DICOM format from the folder /data/DICOM/ and compute the corresponding tensor components. The resulting data set is then fed into the pipe.
   (a) meanwhile, `dwi2mask` will generate a brain mask from the DWI data, and feed the result into a second pipeline.
   (b) `maskfilter` will access the mask from this second pipeline, erode the mask by 3 voxels, and output the name of the temporary file for use as an *argument* by the next stage.

2. `tensor2metric` will access the tensor data set from the first pipe, generate eigenvector and FA maps within the mask provided as an *argument* by the second pipeline, store the eigenvector map in `ev.mif` and feed the FA map into the next stage of the pipeline.
3. `mrthreshold` will access the FA image from the pipe, identify the 300 highest-valued voxels, and produce a mask of these voxels, stored in `highFA.mif`.

CHAPTER 10

Troubleshooting

Remote display issues

The GUI components in *MRtrix3* (`mrview` & `shview`) use the OpenGL 3.3 API to make full use of modern graphics cards. Unfortunately, X11 forwarding is not supported for OpenGL >= 3. There are a number of reasons for this:

- OpenGL 1 & 2 used the OpenGL fixed function pipeline (now deprecated), whereas OpenGL >= 3 relies much more explicitly on shaders and buffer objects. Amongst other things, the use of buffer objects implies that potentially very large amounts of data be downloaded onto the GPU. In a X11 forwarding context, this would mean transferring these data over the network, which would probably end up being prohibitively slow in a sufficient number of situations that including support for it into the GLX was not thought to be worth the effort.
- X11 is unbelievably outdated, even according to the X.org developers themselves (as very clearly explained in this linux.conf.au talk). Current development efforts are going into its replacement, Wayland, which will start replacing X11 in earnest over the next few years (it's already available and usable on the latest distributions). Thankfully, remote display capability is planned for Wayland, and support for it has already been added.

So it is not possible to use `mrview` or `shview` over a standard remote X11 connection.

Why does *MRtrix3* use OpenGL 3.3 if it come with such limitations?

Because it's clearly the most future-proof option. The older OpenGL versions are deprecated, and not recommended for modern applications. The OpenGL 3.3 API is much closer to the way modern graphics hardware works, and can therefore provide better performance. Finally, as explained above, X11 will eventually be phased out anyway...

What can be done about this?

There are a number of options available to deal with this, each with their own idiosyncraties. The simplest is to render locally (option 1), the other options require a fair bit of setting up on the server, and potentially also on the clients.

1. Use MRView locally

This is the simplest option, and allows the use of the local graphics hardware (much like X11 forwarding would have). To use this relatively seamlessly, the simplest option is to access the remote data using a network filesystem, such as SSHFS, SMB or NFS, and run `mrview` locally, loading the data from the network share. While this may seem inefficient, bear in mind that *MRtrix3* will typically only load the data it needs to, so operation will probably not be slower than it would have been with the MRtrix 0.2.x version. Besides, the largest data files are likely to be track files (which will need to be loaded in their entirety); in the MRtrix 0.2.x version these needed to be streamed in whole over the network *for every screen update*.

Of the networked filesystems listed above, the simplest to use would probably be SSHFS, since it shouldn't require any additional setup on the server (assuming users already have an SSH account), and is readily available on all platforms (using Win-SSHFS on Windows, OSXFuse on MacOSX).

2. Use an OpenGL-capable VNC server

Using the VNC protocol, the server is responsible for doing all the rendering remotely, and sends the resulting screen updates over the network. With this approach, users are presented with a full-blown desktop environment running on the server. This may consume too many resources on the remote server, depending on the desktop environment used. Also, since rendering is performed on the remote server, it needs to be equipped with an OpenGL 3.3 capable graphics stack - this means decent hardware *and* an up to date driver. However, it has the advantage of being widely supported and readily available on all platforms, with many implementations available. The only tricky part here is ensuring the VNC server is OpenGL-capable. As far as I can tell, x11vnc can be used for this.

3. Use VirtualGL to allow OpenGL forwarding within X11

The VirtualGL project offers a means of rendering OpenGL graphics on the remote server, and sending the updated contents of the OpenGL window to the local display, alongside the normal X11 connection. This provides a means of running `mrview` in a potentially more familiar X11 over SSH session. As with the VNC solution, rendering needs to be performed on the remote server, meaning it needs to be equipped with an OpenGL 3.3 capable graphics stack - this means decent hardware *and* an up to date driver. Also, it requires the installation of additional software on the local system. Finally, for this to work, all OpenGL commands need to be prefixed with `vglrun` (not particularly problematic as this can be scripted or aliased). This has been reported to work well with *MRtrix3*.

OpenGL version 3.3 not supported

This will typically lead to `mrview` crashing with a message such as:

```
mrview: [ERROR] GLSL log [vertex shader]: ERROR: version '330' is not supported
```

There are three main reasons for this:

1. **Attempting to run MRView using X11 forwarding.** This will not work without some effort, see *Remote display issues* for details.
2. **Your installation genuinely does not support OpenGL 3.3.** In this case, the solution will involve figuring out:
   - whether your graphics hardware can support OpenGL 3.3 at all;
   - whether your Linux distribution provides any drivers for your graphics hardware that can support OpenGL 3.3;
   - if not, whether the manufacturer of your graphics hardware provides drivers for Linux that can be installed on your distribution;

- how to install these drivers - a process that is invariably distribution-specific, and beyond the scope of this document. If you're having serious issues with this, you should consider asking on the MRtrix3 community forum, you will often find others have come across similar issues and can provide useful advice. If you do, make sure you provide as much information as you can (at the very least, your exact distribution, including which version of it, the exact model of your graphics hardware, and what you've tried so far).

3. **Your installation does support OpenGL 3.3, but only provides access to the 3.3 functionality through the _compatibility_ profile, not through the (default) core profile.** To see whether this is the problem, you only need to add the line:

```
NeedOpenGLCoreProfile: 0
```

to your MRtrix configuration file (typically, `~/.mrtrix.conf`). If it doesn't work, you're probably stuck with reason 2.

MRView runs with visual artefacts or no display

If you find that MRView displays with visual glitches or a blank screen, particularly in volume render mode, and on ATI/AMD hardware, you may find that setting:

```
NeedOpenGLCoreProfile: 0
```

may resolve the problem.

Unusual symbols on terminal

When running *MRtrix3* commands on certain terminal emulators, you may see unusual characters appearing in the terminal output, that look something like the following:

```
$ mrinfo fa.mif -debug
mrinfo: ←[00;32m[INFO] opening image "fa.mif"...←[0m
mrinfo: ←[00;34m[DEBUG] reading key/value file "fa.mif"...←[0m
mrinfo: ←[01;31m[ERROR] failed to open key/value file "fa.mif": No such file or␣
↪directory←[0m
```

MRtrix3 uses VT100 terminal control codes to add colour to the terminal output, and to clear the terminal line of text when updating the text displayed during certain processes. Some terminal emulators may not have support for these codes, in which case unwanted characters and symbols may instead be displayed.

There are two possible solutions:

1. Use a different terminal emulator. In particular, earlier instructions for installing *MRtrix3* on Windows involved the use of the terminal provided with Git for Windows; this is known to not support VT100 codes. The current recommendation for *MRtrix3* Windows installation is based on MSYS2; the **'MinGW-w64 Win64 Shell'** provided in this installation is known to support VT100 codes.

2. Terminal colouring can be disabled using the MRtrix configuration file. Add the following line to either the system-wide or user config file to disable these advanced terminal features:

```
TerminalColor: 0
```

Hanging on network file system when writing images

When any *MRtrix3* command must read or write image data, there are two primary mechanisms by which this is performed:

1. Memory mapping: The operating system provides access to the contents of the file as though it were simply a block of data in memory, without needing to explicitly load all of the image data into RAM.

2. Preload / delayed write-back: When opening an existing image, the entire image contents are loaded into a block of RAM. If an image is modified, or a new image created, this occurs entirely within RAM, with the image contents written to disk storage only at completion of the command.

This design ensures that loading images for processing is as fast as possible and does not incur unnecessary RAM requirements, and writing files to disk is as efficient as possible as all data is written as a single contiguous block.

Memory mapping will be used wherever possible. However one circumstance where this should *not* be used is when *write access* is required for the target file, and it is stored on a *network file system*: in this case, the command typically slows to a crawl (e.g. progressbar stays at 0% indefinitely), as the memory-mapping implementation repeatedly performs small data writes and attempts to keep the entire image data synchronised.

MRtrix3 will now *test* the type of file system that a target image is stored on; and if it is a network-based system, it will *not* use memory-mapping for images that may be written to. *However*, if you experience the aforementioned slowdown in such a circumstance, it is possible that the particular configuration you are using is not being correctly detected or identified. If you are unfortunate enough to encounter this issue, please report to the developers the hardware configuration and file system type in use.

Linux: very slow performance when writing large images

This might be due to the Linux Disk Caching or the kernel's handling of _‘dirty pages <https://lonesysadmin.net/2013/12/22/better-linux-disk-caching-performance-vm-dirty_ratio/>‘__.

On Ubuntu, you can get your current dirty page handling settings with `sysctl -a | grep dirty`. Those settings can be modified in `/etc/sysctl.conf` by adding the following two lines to `/etc/sysctl.conf`:

```
vm.dirty_background_ratio = 60
vm.dirty_ratio = 80
```

`vm.dirty_background_ratio` is a percentage fraction of your RAM and should be larger than the image to be written. After changing `/etc/sysctl.conf`, execute `sysctl -p` to configure the new kernel parameters at runtime. Depending on your system, these changes might not be persistent after reboot.

Conflicts with previous versions of Qt

If previous versions of Qt were already installed on the system, they can sometimes conflict with the installation of *MRtrix3*. This can manifest in many ways, but the two most obvious one are:

- `./configure` reports using the older version, but `./build` completes without errors. However, MRView crashes, complaining about OpenGL version not being sufficient.
- `./configure` reports the correct version of Qt, but `./build` fails with various error messages (typically related to redefined macros, with previous definitions elsewhere in the code).

Compiler error during build

If you encounter an error during the build process that resembles the following:

```
ERROR: (#/#) [CC] release/cmd/command.o

/usr/bin/g++-4.8 -c -std=c++11 -pthread -fPIC -march=native -I/home/user/mrtrix3/
↪eigen -Wall -O2 -DNDEBUG -Isrc -Icmd -I./lib -Icmd cmd/command.cpp -o release/cmd/
↪command.o

failed with output

g++-4.8: internal compiler error: Killed (program cc1plus)
Please submit a full bug report,
with preprocessed source if appropriate.
See for instructions.
```

This is most typically caused by the compiler running out of RAM. This can be solved either through installing more RAM into your system, or by restricting the number of threads to be used during compilation:

```
NUMBER_OF_PROCESSORS=1 ./build
```

CHAPTER 11

Basic DWI processing

This tutorial will hopefully provide enough information for a novice user to get from the raw DW image data to performing some streamlines tractography. It may also be useful for experienced MRtrix users in terms of identifying some of the new command names.

For all *MRtrix3* scripts and commands, additional information on the command usage and available command-line options can be found by invoking the command with the `-help` option. Note that this tutorial includes commands and scripts for which there are relevant journal articles for citation; these are listed on the help pages also.

DWI geometric distortion correction

If the user has access to reversed phase-encode spin-echo image data, this can be used to correct the susceptibility-induced geometric distortions present in the diffusion images, as well as any eddy current-induced distortions and inter-volume subject motion. Procedures for this correct are not yet implemented in *MRtrix3*, though we do provide a script for interfacing with the relevant FSL tools:

```
dwipreproc <PE direction> <Input DWI series> <Output corrected DWI series>
[options]
```

For more details, see the *dwipreproc* help file. In particular, it is necessary to manually specify what type of reversed phase-encoding acquisition has taken place (if any), and provide the relevant input images.

DWI brain mask estimation

In previous versions of MRtrix, a heuristic was used to derive this mask; a dedicated command is now provided:

```
$ dwi2mask <Input DWI> <Output mask>
$ mrview <Input DWI> -roi.load <Output mask>
```

Note that if you are working with ex-vivo data, this command will likely not give the desired results. It can also give inconsistent results in cases of low SNR, strong B1 bias field, or even with good-quality images; it is recommended

that the output of this command should *always* be checked (and corrected if necessary) before proceeding with further processing.

Response function estimation

To perform spherical deconvolution, the DWI signal emanating from a single coherently-oriented fibre bundle must be estimated. We provide a script for doing this, which has *a range of algorithms and parameters*. This example will use fairly sensible defaults:

```
$ dwi2response tournier <Input DWI> <Output response text file>
$ shview <Output response text file>
```

Fibre Orientation Distribution estimation

This command performs Constrained Spherical Deconvolution (CSD) based on the response function estimated previously.

```
$ dwi2fod csd <Input DWI> <Input response text file> <Output FOD image> -mask <Input␣
↪DWI mask>
$ mrview <Input DWI> -odf.load_sh <Output FOD image>
```

Whole-brain streamlines tractography

For the sake of this tutorial, we will perform whole-brain streamlines tractography, using default reconstruction parameters.

```
$ tckgen <Input FOD image> <Output track file> -seed_image <Input DWI mask> -mask
↪<Input DWI mask> -number <Number of tracks>
$ mrview <Input DWI> -tractography.load <Output track file>
```

Note: Loading a very large number of tracks can inevitably make the `mrview` software run very slowly. When this occurs, it may be preferable to instead view only a subset of the generated tracks, e.g.:

```
$ tckedit <Track file> <Smaller track file> -number <Smaller number of tracks>
$ mrview <Input DWI> -tractography.load <Smaller track file>
```

Track Density Imaging (TDI)

TDI can be useful for visualising the results of tractography, particularly when a very large number of streamlines is generated.

```
$ tckmap <Input track file> <Output TDI> -vox <Voxel size in mm>
$ mrview <Output TDI>
```

CHAPTER 12

DWI denoising

MRtrix now includes a new command `dwidenoise` which implements DWI data denoising and noise map estimation by exploiting data redundancy in the PCA domain (*Veraart et al., 2016a, 2016b*). The method uses the prior knowledge that the eigenspectrum of random covariance matrices is described by the universal Marchenko Pastur distribution.

Recommended use

Image denoising must be performed as the first step of the image-processing pipeline. Interpolation or smoothing in other processing steps, such as motion and distortion correction, may alter the noise characteristics and thus violate the assumptions upon which MP-PCA is based.

Typical use will be:

```
dwidenoise dwi.mif out.mif -noise noise.mif
```

where `dwi.mif` contains the raw input DWI image, `out.mif` is the denoised DWI output, and `noise.mif` is the estimated spatially-varying noise level.

We always recommend eyeballing the residuals, i.e. out - in, as part of the quality control. The lack of anatomy in the residual maps is a marker of accuracy and signal-preservation during denoising. The residuals can be easily obtained with

```
mrcalc dwi.mif out.mif -subtract res.mif
mrview res.mif
```

The kernel size, default 5x5x5, can be chosen by the user (option: `-extent`). For maximal SNR gain we suggest to choose N>M for which M is typically the number of DW images in the data (single or multi-shell), where N is the number of kernel elements. However, in case of spatially varying noise, it might be beneficial to select smaller sliding kernels, e.g. N~M, to balance between precision, accuracy, and resolution of the noise map.

Note that this function does not correct for non-Gaussian noise biases yet.

References

1. J. Veraart, E. Fieremans, and D.S. Novikov *Diffusion MRI noise mapping using random matrix theory.* Magn. Res. Med., early view (2016), doi: 10.1002/mrm.26059

2. J. Veraart, D.S. Novikov, D. Christiaens, B. Ades-aron, J. Sijbers, and E. Fieremans *Denoising of diffusion MRI using random matrix theory.* NeuroImage, in press (2016), doi: 10.1016/j.neuroimage.2016.08.016

CHAPTER 13

Structural connectome for Human Connectome Project (HCP)

This document duplicates the information provided during the *MRtrix3* demonstration at ISMRM 2015 in Toronto. We will generate a structural connectome for quintessential Human Connectome Project subject 100307. Some of these instructions will be specific to HCP data, others will be more general recommendations.

Necessary files

To duplicate our methods and results, you will need to download the appropriate files, accessible through the following steps:

- https://db.humanconnectome.org/
- WU-Minn HCP Data - 900 Subjects + 7T
- Download Image Data: Single subject
- Session Type: 3T MRI
- Processing level: Preprocessed
- Package Type: MSM-Sulc + MSM-All
- add Structural Preprocessed and Diffusion Preprocessed to queue

The actual files within these compressed downloads that we will make use of are:

Diffusion preprocessed files

- bvals
- bvecs
- data.nii.gz
- nodif_brain_mask.nii.gz

Structural preprocessed files

- aparc+aseg.nii.gz
- T1w_acpc_dc_restore_brain.nii.gz

Structural image processing

1. Generate a tissue-segmented image appropriate for Anatomically-Constrained Tractography:

```
5ttgen fsl T1w_acpc_dc_restore_brain.nii.gz 5TT.mif -premasked
```

Note that it is *not necessary* to use a tissue-segmented image that has the same resolution as the diffusion images; *MRtrix3* will happily acquire interpolated values from each of them separately as tracking is performed. This allows ACT to exploit the higher spatial resolution of the tissue-segmented anatomical image, but still use the diffusion image information at its native resolution also.

2. Collapse the multi-tissue image into a 3D greyscale image for visualisation:

```
5tt2vis 5TT.mif vis.mif; mrview vis.mif
```

If the tissue segmentation image contains clearly erroneous tissue labels, you can delineate them manually using the ROI editor tool in *mrview*, then apply your corrections to the tissue data using the *5ttedit* command.

3. Modify the integer values in the parcellated image, such that the numbers in the image no longer correspond to entries in FreeSurfer's colour lookup table, but rows and columns of the connectome:

```
labelconvert aparc+aseg.nii.gz FreeSurferColorLUT.txt fs_default.txt nodes.mif
```

File `FreeSurferColorLUT.txt` is provided with FreeSurfer in its root directory. The target lookup table file (`fs_default.txt` in this case) is a handy text file that provides a structure name for every row / column of the connectome matrix: it is provided as part of *MRtrix3*, and located at `src/connectome/tables/fs_default.txt` within the *MRtrix3* folder.

4. Replace FreeSurfer's estimates of sub-cortical grey matter structures with estimates from FSL's FIRST tool:

```
labelsgmfix nodes.mif T1w_acpc_dc_restore_brain.nii.gz fs_default.txt
nodes_fixSGM.mif -premasked
```

Diffusion image processing

1. Convert the diffusion images into a non-compressed format (not strictly necessary, but will make subsequent processing faster), embed the diffusion gradient encoding information within the image header, re-arrange the data strides to make volume data contiguous in memory for each voxel, and convert to floating-point representation (makes data access faster in subsequent commands):

```
mrconvert data.nii.gz DWI.mif -fslgrad bvecs bvals -datatype float32 -stride
0,0,0,1
```

2. Generate a mean *b*=0 image (useful for visualisation):

```
dwiextract DWI.mif - -bzero | mrmath - mean meanb0.mif -axis 3
```

(If you are not familiar with the 'I' piping symbol, read more about it here)

3. Estimate the response function; note that here we are estimating *multi-shell*, *multi-tissue* response functions:

```
dwi2response msmt_5tt DWI.mif 5TT.mif RF_WM.txt RF_GM.txt RF_CSF.txt -voxels
RF_voxels.mif
```

```
mrview meanb0.mif -overlay.load RF_voxels.mif -overlay.opacity 0.5
```
(check appropriateness of response function voxel selections)

4. Perform Multi-Shell, Multi-Tissue Constrained Spherical Deconvolution:

```
dwi2fod msmt_csd DWI.mif RF_WM.txt WM_FODs.mif RF_GM.txt GM.mif RF_CSF.txt
CSF.mif -mask nodif_brain_mask.nii.gz
```

```
mrconvert WM_FODs.mif - -coord 3 0 | mrcat CSF.mif GM.mif - tissueRGB.mif
-axis 3
```

This generates a 4D image with 3 volumes, corresponding to the tissue densities of CSF, GM and WM, which will then be displayed in *mrview* as an RGB image with CSF as red, GM as green and WM as blue (as was presented in the MSMT CSD manuscript).

```
mrview tissueRGB.mif -odf.load_sh WM_FODs.mif
```
(visually make sure that both the tissue segmentations and the white matter FODs are sensible)

Connectome generation

1. Generate the initial tractogram:

```
tckgen WM_FODs.mif 100M.tck -act 5TT.mif -backtrack -crop_at_gmwmi
-seed_dynamic WM_FODs.mif -maxlength 250 -number 100M -cutoff 0.06
```

Explicitly setting the maximum length is highly recommended for HCP data, as the default heuristic - 100 times the voxel size - would result in a maximum length of 125mm, which would preclude the reconstruction of some longer pathways.

We also suggest a reduced FOD amplitude cutoff threshold for tracking when using the MSMT CSD algorithm in conjunction with ACT; this allows streamlines to reach the GM-WM interface more reliably, and does not result in significant false positives since the MSMT algorithm does not produce many erroneous small FOD lobes.

2. Apply the Spherical-deconvolution Informed Filtering of Tractograms (SIFT) algorithm

This method reduces the overall streamline count, but provides more biologically meaningful estimates of structural connection density:

```
tcksift 100M.tck WM_FODs.mif 10M_SIFT.tck -act 5TT.mif -term_number 10M
```

If your system does not have adequate RAM to perform this process, the first recommendation is to reduce the spatial resolution of the FOD image and provide this alternative FOD image to SIFT (this should have little influence on the outcome of the algorithm, but will greatly reduce memory consumption):

```
mrresize WM_FODs.mif FOD_downsampled.mif -scale 0.5 -interp sinc
```

If this still does not adequately reduce RAM usage, you will need to reduce the number of input streamlines to a level where your processing hardware can successfully execute the *tcksift* command, e.g.:

```
tckedit 100M.tck 50M.tck -number 50M
```

Alternatively, if you're feeling brave, you can give SIFT2 a try...

3. Map streamlines to the parcellated image to produce a connectome:

```
tck2connectome 10M_SIFT.tck nodes_fixSGM.mif connectome.csv
```

```
mrview nodes_fixSGM.mif -connectome.init nodes_fixSGM.mif -connectome.load
connectome.csv
```

CHAPTER 14

Using the connectome visualisation tool

The connectome tool bar in *MRtrix3* has been designed from scratch, with the intention of providing a simple, data-driven mechanism for visually assessing individual connectomes as well as the results of network-based group statistics. The interface may therefore vary considerably from other connectome visualisation packages, and may be intimidating for new users who simply want to 'see the connectome'. I hope I can convince you in this tutorial that the design of this tool allows you, the user, to dictate exactly *how* you want to visualise the connectome, rather than being forced to conform to a particular prior expectation of how such things should be visualised.

Initialising the tool

My suspicion is that new users will load the tool, and immediately think: 'Where do I load my connectome?'. Well, let's take a step backwards. If you were to give the software a connectome matrix, with no other data, there would be no way to visualise that connectome in the space of an MR image: the software has no information about the spatial locations of the nodes upon which that connectome is based. So the first step is actually to load an image to provide the tool with this information, using the "Node image" button at the top of the toolbar. The desired image is the output of the `labelconvert` command, as detailed in the *Structural connectome construction* guide: the tool uses this image to localise each parcel in 3D space in preparation for visualisation. Alternatively, you can load the relevant parcellation image from the command-line when you first run `mrview`, using the `-connectome.init` option.

If you still do not see anything in the `mrview` main window, this is likely because you have not yet opened a primary image in `mrview`. This is currently necessary for `mrview` to correctly set up the camera positioning. The easiest solution is to open your parcellation image not only to initialise the connectome tool, but also as a standard image in `mrview`; then simply *hide* the main image using the 'View' menu.

With the basis parcellation image loaded, the tool will display the location of each node; note however that all of the nodes are exactly the same colour, and exactly the same size, and there are no connections shown between them - it's an entirely dis-connected network. This makes sense - we haven't actually provided the tool with any information regarding which connections are present and which are absent. We can also do the opposite: change the "Edge visualisation" - "Visibility" from 'None' to 'All', and now the software shows every edge in the connectome non-discriminantly.

Therefore, we need some mechanism of informing the software of which edges should be drawn, and which should not. Most logically, this could be achieved by loading a structural connectome, and perhaps applying some threshold. So

now, for the "Edge visualisation" - "Visibility" option, select "Matrix file", and load your connectome. The software now uses the data from this external file to threshold which edges are drawn and which are not, and also allows you to vary that threshold interactively. (You can also load a connectome matrix from the command line using the `-connectome.load` option.)

The connectome still however has a binary appearance; every edge in the connectome is either present or absent, and they all have the same size and the same colour. We know that our connectome contains weights distributed over a wide scale, and would like to be able to see this as part of our visualisation; for instance, we may decide that more dense connections should have a 'hot' colour appearance, whereas less dense connections should be darker. We can achieve this by changing the "Edge visualisation" - "Colour" from 'Fixed' to 'Matrix file', and selecting an appropriate matrix file (perhaps the same file as was used for the visibility threshold, perhaps not).

For most users, connectome data will be loaded using the 'open' button in the 'connectome matrices' section, or at the command-line when `mrview` is first run using the `-connectome.load` option.

Basis of connectome visualisation customisation

With the above steps completed, you should obtain a fairly rduimentary visualisation of the connectome you have loaded. The plethora of buttons and gadgets in the connectome tool user interface is however a clue regarding the scope of customisation available for precisely how the connectome data will be displayed.

As an example, consider the 'Edge visualisation - Colour' entry. These options control how the colour of each individual edge in the connectome will be determined, based on the data the tool is provided with. Clicking on the main combo box shows that there are a few options available:

- *Fixed*: Use the same fixed colour to display all visible edges.
- *By direction*: The XYZ spatial offset between the two nodes connected by

an edge is used to derive an RGB colour (much like the default streamlines colouring).

- *Connectome*: The colour of each edge will depend on the value for that

edge in the connectome you have loaded, based on some form of value -> colour mapping (a 'colour map').

- *Matrix file*: Operates similarly to the *connectome* option; except that

the value for each edge is drawn from a matrix file that is *not* the connectome matrix you have loaded (though it must be based on the same parcellation to have any meaning). So for instance: You could load a structural connectome file as your connectome matrix and show only those edges where the connection density is above a certain threshold, but then set the *colour* of each edge based on a *different matrix file* that contains functional connectivity values.

If the *Connectome* or *Matrix file* options are used, it is also possible to alter the colour map used, and modify the values at which the edges will reach the colours at either extreme of the colour map.

Hopefully, this simple demonstration will be enough to highlight the design principle of this tool, and therefore the frame of mind necessary to use it effectively:

What *data* do I want to determine a specific *visual property* of my connectome?

There is tremendous power in separating these two entities. For instance, consider a use case where I have performed network-based group statistics, and wish to visualise my result. I may choose to threshold the connectome edges based on statistical significance, but set the width of the connections based on the mean connection strength to get an idea of the density of connections in the detected network, but set the colour of each edge based on the effect size to see which components of the network are most affected. I can even automatically hide any nodes that are not involved in the detected network by selecting "Node visualisation" - "Visibility" - 'Degree >= 1'.

Importing detailed node information

When the parcellation image is first loaded, the software has no information regarding the designations of the underlying nodes, so it simply labels them as "Node 1", "Node 2" etc.. To show the anatomical name of each node in the list, you must load the connectome lookup table that was used as the target output in the `labelconvert` step during [structural connectome construction]. This file provides a list of node indices and their corresponding names, so is perfect for subsequent assessment of the resulting connectomes, whether using this tool or in other contexts (e.g. Matlab). Such a lookup table may also include a pre-defined colour for each node, which can then be used during visualisation by selecting "Node Visualisation -> Colour -> LUT".

Advanced visualisation

There are a couple of neat tricks that can be used to produce impressive-looking visualisations, but need some pre-processing or careful consideration in order to achieve them.

Visualising edges as streamlines / streamtubes

Rather than drawing a straight line between connected nodes to represent an edge, it is possible with tractography-based connectome construction to instead represent each connection based on the structural trajectory by which those nodes are inter-connected. This can be achieved as follows:

- When generating the connectome using *tck2connectome*, use the `-assignments` option. This will produce a text file where each line contains the indices of the two nodes to which that particular streamline was assigned.
- Use the *connectome2tck* command to produce a single track file, where every streamline represents the mean, or *exemplar*, trajectory between two nodes. This is achieved using two command-line options: `-exemplars` to instruct the command to generate the exemplar trajectory for each edge, rather than keeping all streamlines (you will need to provide your parcellation image); and `-files single` to instruct the command to place all computed exemplars into a single output file.
- In the `mrview` connectome toolbar, select "Edge visualisation" - "Geometry" - 'Streamlines / Streamtubes', and select the exemplar track file just generated.

Visualising nodes as triangulated meshes

Although the node parcellations are represented as volumetric segmentations, and we do not yet have support for importing mesh-based parcellations, it is still possible to visualise the conectome nodes using a mesh-based representation. This is done by explicitly converting the volume of each parcel to a triangulated mesh. The process is as follows:

- Compute a triangular mesh for each node, and store the results in a single file. The command is called *label2mesh*. Note that the output file *must* be in the `.obj` file format: this is the only format currently supported that is capable of storing multiple mesh objects in a single file.
- (Optional) Smooth the meshes to make them more aesthetically pleasing (the results of the conversion process used in `label2mesh` appear very 'blocky'). Apply the *meshfilter* command, using the `smooth` operator. Again, the output must be in the `.obj` format.
- In the `mrview` connectome toolbar, select "Node visualisation" - "Geometry" - 'Mesh', and select the mesh file just generated.

Using node selection to highlight features of interest

The table in the connectome toolbar that lists the node names and colours can also be used to select and highlight particular nodes. In most cases, this will simply be an additional 'toy' for navigating the data; however it's also possible that this capability will prove to be a powerful tool for demonstrating network features.

In any connectome visualisation software, when the user selects one or more particular nodes of interest, some modification must be applied to the visual features of the nodes in order to 'highlight' the nodes of interest. In many cases, this may be hard-wired to behave in a particular way. In the case of `mrview` in *MRtrix3*, this highlighting mechanism is entirely flexible: the user can control the visual modifications applied to both those network elements selected and those not selected. For instance, you may choose for nodes to become completely opaque when you select them, while other un-selected nodes remain transparent; or they may grow in size with respect to the rest of the connectome; or they may change in colour to highlight them; or those nodes not selected may disappear entirely. This flexibility is accessed via the "Selection visualisation settings" button, which will open a dialog window providing access to these settings.

As manual selection applies to nodes only, the behaviour for edges is as follows:

- When no nodes are selected, all edges are drawn according to their standard settings.
- If a single node is selected, all edges emanating from that node are considered to be 'selected', and the relevant visual modifiers will be applied.
- If two or more nodes are selected, only connections exclusively connecting between the nodes of interest are considered to be 'selected'.

Node visualisation using matrices

When using external data files to control the visual properties of the connectome, most commonly *vector files* will be used to determine visual properties of nodes, and *matrix files* will be used to determine visual properties of edges. These provide precisely one scalar value per connectome element, and therefore provide a static visual configuration.

It is however also possible to set any visual property of the connectome nodes based on a *matrix file*. In this scenario, the values to be drawn from the matrix - and hence their influence on the relevant visual property of the nodes - depends on the *current node selection*. That is: once you select a node of interest, the software extracts the relevant row from the matrix, and uses only that row to influence the node visual property to which it has been assigned. In the case where multiple nodes of interest are selected, an additional drop-down menu is provided, that allows you to prescribe how those multiple rows of matrix data are combined in order to produce a single scalar value per node, which can then be used to influence its relevant visual property.

CHAPTER 15

Structural connectome construction: Using `labelconvert`

The `labelconvert` (previously `labelconfig`) step in *Structural connectome construction* has proven to be a hurdle for many. It may be a 'unique' step in so far as that other software packages probably deal with this step implicitly, but in MRtrix we prefer things to be explicit and modular. So here I'll go through an example to demonstrate exactly what this command does.

Worked example

For this example, let's imagine that we're going to generate a structural connectome for Bert, the quintessential FreeSurfer subject. Also, we're going to generate the connectome based on the Desikan-Killiany atlas. The default FreeSurfer pipeline provides the volumetric image aparc+aseg.mgz; this is the file that will be used to define the nodes of our connectome.

Looking at the raw image itself, each node possesses a particular intensity, corresponding to a particular integer value. If we focus on the superior frontal gyrus in the right hemisphere, we can see that the image intensity is 2028 for this structure.

This immediately presents a problem for constructing a connectome: if any streamline encountering this region were written to row/column 2028, our connectome would be enormous, and consist mostly of zeroes (as most indices between 1 and 2028 do not correspond to any structure). Therefore, what we'd prefer is to map the unique integer index of this structure to a particular row/column index of the connectome; this should be done in such a way that all structures of interest have a unique integer value between 1 and *N*, where *N* is the number of nodes in the connectome.

Now looking at the file `FreeSurferColorLUT.txt` provided with FreeSurfer, we see the following:

```
...
2026    ctx-rh-rostralanteriorcingulate     80  20  140 0
2027    ctx-rh-rostralmiddlefrontal         75  50  125 0
2028    ctx-rh-superiorfrontal              20  220 160 0
2029    ctx-rh-superiorparietal             20  180 140 0
2030    ctx-rh-superiortemporal             140 220 220 0
...
```

This gives us a *meaningful name* for this structure based on the integer index. It also gives us some colour information, but let's not worry about that for now.

Our goal then is to determine a *new integer index* for this structure, that will determine the row/column of our connectome matrix that this structure corresponds to. This is dealt with by mapping the structure indices of this lookup table to a *new* lookup table. For this example, let's imagine that we're using the default MRtrix lookup table for the FreeSurfer Desikan-Killiany atlas segmentation: this is provided at `src/connectome/tables/fs_default.txt`.Examining this file in detail, we see the following:

```
...
74    R.RACG  ctx-rh-rostralanteriorcingulate 80  20  140 255
75    R.RMFG  ctx-rh-rostralmiddlefrontal     75  50  125 255
76    R.SFG   ctx-rh-superiorfrontal          20  220 160 255
77    R.SPG   ctx-rh-superiorparietal         20  180 140 255
78    R.STG   ctx-rh-superiortemporal         140 220 220 255
...
```

(This file is in a slightly different format to `FreeSurferColorLUT.txt`; don't worry about this for the time being)

This file contains the *same structure name* as the FreeSurfer look-up table, but it is assigned a *different integer index* (76)! What's going on?

The following is what the `labelconvert` command is actually going to do under the bonnet, using these two lookup table files:

1. Read the integer value at each voxel of the input image
2. Convert the integer value into a string, based on the *input lookup table file* (`FreeSurferColorLUT.txt`)
3. Find this string in the *output lookup table file* (`fs_default.txt`)
4. Write the integer index stored in the *output lookup table file* for this structure to the voxel in the output image

This is what the actual command call looks like:

```
labelconvert $FREESURFER_HOME/subjects/bert/mri/aparc+aseg.mgz $FREESURFER_HOME/
↪FreeSurferColorLUT.txt ~/mrtrix3/src/connectome/config/fs_default.txt bert_parcels.
↪mif
```

And this is what the resulting image looks like:

The integer labels of the underlying grey matter parcels have been *converted* from the input lookup table to the output lookup table (hence the name `labelconvert`). They now increase monotonically from 1 to the maximum index, with no 'gaps' (i.e. ununsed integer values) in between. Therefore, when you construct your connectome using `tck2connectome`, the connectome matrix will only be as big as it needs to be to store all of the node-node connectivity information.

Design rationale

Making this step of re-indexing parcels explicit in connectome construction has a few distinct advantages:

- You can use parcellations from any software / atlas: just provide the

structure index / name lookup table that comes with whatever software / atlas provides the parcellation, and define an appropriate target lookup table that defines which index you want each structure to map to.

- `tck2connectome` can be 'dumb and blind': it reads the integer indices

at either end of the streamline, and that's the row/column of the connectome matrix that needs to be incremented.

- You can have your grey matter parcels appear in any order in your

matrices: just define a new lookup table file. Doing this prior to connectome construction is less likely to lead to heartache than re-ordering the rows and columns in e.g. Matlab, where you may lose track of which matrices have been re-ordered and which have not.

- You can remove structures from the connectome, or merge multiple structures

into a single parcel, just by omitting or duplicating indices appropriately in the target lookup table file.

- Looking at your matrices and need to find out what structure corresponds to

a particular row/column? Just look at the config file!

Obviously if your parcellation image already has node indices that increase monotonically from 1, and you're happy enough with the numerical order of the nodes, you don't actually need to use the `labelconvert` step at all.

Custom design connectomes

Some notes for anybody that wishes to define their own configuration files (either for re-ordering nodes, changing selection of nodes, or using parcellations from alternative sources):

- If you wish to omit nodes from your connectome (e.g. the cerebellar hemispheres), you may be better off making these nodes the largest indices in your connectome, but then cropping them from the connectome matrices retrospectively, rather than omitting them from the parcellation image entirely: If you were to do the latter, streamlines that would otherwise be assigned to your unwanted nodes may instead be erroneously assigned to the nearest node that is part of your connectome (exactly what happens here will depend on the streamline-node assignment mechanism used).
- The command `labelconvert` is capable of reading in look-up tables in a number of formats. If you wish to define your own lookup table, you will need to conform to one of these formats in order for MRtrix commands to be able to import it. If you are using an atlas where the look-up table does not conform to any of these formats (and hence MRtrix refuses to import it), you can either manually manipulate it into a recognized format, or if it is likely that multiple users will be using that parcellation scheme, we may choose to add a parser to the MRtrix code: contact the developers directly if this is the case.

CHAPTER 16

Advanced debugging

On rare occasions, a user may encounter a critical error (e.g. "Segmentation fault") within an *MRtrix3* command that does not give sufficient information to identify the cause of the problem, and that the developers are unable to reproduce. In these cases, we will often ask to be provided with example data that can consistently reproduce the problem in order to localise the issue. An alternative is for the user to perform an initial debugging experiment, and provide us with the resulting information. The instructions for doing so are below.

1. If required, install `gdb`; the GNU Debugging Tool (specific instructions for this installation will depend on your operating system)
2. *Make sure* you are using the most up-to-date *MRtrix3* code! (`git pull`)
3. Configure and compile *MRtrix3* in debug mode:

   ```
   ./configure -debug -assert debug; ./build debug
   ```

   Note that this compilation will reside *alongside* your existing *MRtrix3* installation, but will not interfere with it in any way. Commands that are compiled in debug mode will reside in the `debug/bin` directory.
4. Execute the problematic command within `gdb`:

   ```
   gdb --args debug/bin/command (arguments) (-options) -debug
   ```

   Note that the version of the command that is compiled in debug mode resides in the `debug/bin` directory; you must provide this full path *explicitly* to ensure that this is the version of the command that is executed. The preceding `gdb --args` at the beginning of the line is simply the easiest way to execute the command within `gdb`. Include all of the file paths, options etc. that you used previously when the problem occurred. It is also recommended to use the *MRtrix3* `-debug` option so that *MRtrix3* produces more verbose information at the command-line.
5. If running on Windows, once `gdb` has loaded, type the following into the terminal:

   ```
   b abort
   b exit
   ```

   These 'breakpoints' must be set explicitly in order to prevent the command from being terminated completely on an error, which would otherwise preclude debugging once an error is actually encountered.

6. At the `gdb` terminal, type `r` and hit ENTER to run the command.
7. If an error is encountered, `gdb` will print an error, and then provide a terminal with `(gdb)` shown on the left hand side. Type `bt` and hit ENTER: This stands for 'backtrace', and will print details on the internal code that was running when the problem occurred.
8. Copy all of the raw text, from the command you ran in instruction 3 all the way down to the bottom of the backtrace details, and send it to us. The best place for these kind of reports is to make a new issue in the Issues tracker for the GitHub repository.
9. If `gdb` does not report any error, it is possible that a memory error is occurring, but even the debug version of the software is not performing the necessary checks to detect it. If this is the case, you can also try using Valgrind, which will perform a more exhaustive check for memory faults (and correspondingly, the command will run exceptionally slowly):

```
valgrind debug/bin/command (arguments) (-options)
```

We greatly appreciate any contribution that the community can make toward making *MRtrix3* as robust as possible, so please don't hesitate to report any issues you encounter.

CHAPTER 17

Warping images using warps generated from other packages

The `mrtransform` command applies warps in a deformation field format (i.e. where each voxel specifies the scanner space position in the corresponding image). However, you can also use `mrtransform` to apply warps generated from other packages that are in any format using the following steps.

1. **Generate an identity warp** using the input moving image (i.e. the image you wish to warp):

```
warpinit input_fod_image.mif identity_warp[].nii
```

2. **Compute a MRtrix compatible warp** by transforming the identity warp using your registration of choice. For example if you are using the ANTs registration package:

```
for i in {0..2};
do;
  WarpImageMultiTransform 3 identity_warp${i}.nii mrtrix_warp{i}.nii -R template.
↪nii ants_warp.nii ants_affine.txt;
done;
```

3. **Correct the mrtrix warp**. When transforming *identity_warp.nii* and producing the mrtrix_warp images, most registration packages will output 0.0 as the default value when the transformation maps outside the input image. This will result in many voxels in the output mrtrix_warp (which is a deformation field) pointing to the origin (0.0, 0.0, 0,0), particularly around the edge of the warp if an affine registration was performed. To correct this and convert all voxels with 0.0,0.0,0.0 to nan,nan,nan (suitable for *mrtransform*):

```
warpcorrect mrtrix_warp[].nii mrtrix_warp_corrected.mif
```

4. **Warp the image**. `mrtransform` can warp any 3D or 4D image, however if the number of volumes in the 4th dimension equals the number of coefficients in an antipodally symmetric spherical harmonic series (i.e. 6, 15, 28 etc), then it assumes the image to be an FOD image and reorientation is automatically applied. Also note that FOD modulation can be applied using the option *-modulation*. This preserves the total apparent fibre density across the width of each bundle before and after warping:

```
mrtransform input_fod_image.mif -warp mrtrix_warp_corrected.mif warped_fod_image.
↪mif
```

CHAPTER 18

Frequently Asked Questions (FAQ)

Processing of HCP data

We expect that a number of users will be wanting to use *MRtrix3* for the analysis of data from the Human Connectome Project (HCP). These data do however present some interesting challenges from a processing perspective. Here I will try to list a few ideas, as well as issues that do not yet have a robust solution; I hope that any users out there with experience with these data will also be able to contribute with ideas or suggestions.

Do my tracking parameters need to be changed for HCP data?

Probably. For instance, the default parameters for length criteria are currently set based on the voxel size rather than absolute values (so e.g. animal data will still get sensible defaults). With such high resolution data, these may not be appropriate. The default maximum length is 100 times the voxel size, or only 125mm at 1.25mm isotropic; this would preclude reconstruction of a number of long-range pathways in the brain, so should be overridden with something more sensible. The minimum length is more difficult, but in the absence of a better argument I'd probably stick with the default (5 x voxel size, or 2 x voxel size if ACT is used).

Also, the default step size for iFOD2 is 0.5 times the voxel size; this will make the track files slightly larger than normal, and will also make the tracks slightly more jittery, but actually disperse slightly less over distance, than standard resolution data. People are free to experiment with the relevant tracking parameters, but we don't yet have an answer for how these things should ideally behave.

Is it possible to use data from all shells in CSD?

The default CSD algorithm provided in the *dwi2fod* command is only compatible with a single b-value shell, and will by default select the shell with the largest b-value for processing.

The Multi-Shell Multi-Tissue (MSMT) CSD method has now been incorporated into *MRtrix3*, and is provided as the msdwi2fod command. There are also instructions for its use provided in the documentation.

The image data include information on gradient non-linearities. Can I make use of this?

Again, unfortunately not yet. Making CSD compatible with such data is more difficult than other diffusion models, due to the canonical response function assumption. To me, there are two possible ways that this could be handled:

- Use the acquired diffusion data to interpolate / extrapolate predicted data on a fixed b-value shell.
- Generate a representation of the response function that can be interpolated / extrapolated as a function of b-value, and therefore choose an appropriate response function per voxel.

Work is underway to solve these issues, but there's nothing available yet. For those wanting to pursue their own solution, bear in mind that the gradient non-linearities will affect both the effective b-value *and* the effective diffusion sensitisation directions in each voxel. Otherwise, the FODs look entirely reasonable without these corrections...

The anatomical tissue segmentation for ACT from 5ttgen fsl seems even worse than for 'normal' data...?

The combination of high spatial resolution and high receiver coil density results in a pretty high noise level in the middle of the brain. This in turn can trick an intensity-based segmentation like FSL's FAST into mislabeling things; it just doesn't have the prior information necessary to disentangle what's in there. I haven't looked into this in great detail, but I would very much like to hear if users have discovered more optimal parameters for FAST, or alternative segmentation software, for which they have been impressed by the results.

Why does SIFT crash on my system even though it's got heaps of RAM?

The main memory requirement for SIFT is that for every streamline, it must store a list of every fixel traversed, with an associated streamline length through each voxel. With a spatial resolution approximately double that of 'standard' DWI, the number of unique fixels traversed by each streamline will go up by a factor of around 3, with a corresponding increase in RAM usage. There is literally nothing I can do to reduce the RAM usage of SIFT; it's fully optimised.

One thing you can do however, is just down-scale the FOD image prior to running *tcksift*: `mrresize in.mif out.mif -scale 0.5 -interp sinc`. This will reduce the RAM usage to more manageable levels, and realistically probably won't have that much influence on the algorithm anyway. Importantly you can still use the high-resolution data for tracking (or indeed anything else); it's only the SIFT step that has the high RAM usage. And using `mrresize` rather than some other software to do the downsampling will ensure that the down-sampled image is still properly aligned with the high-resolution image in scanner space.

Generating Track-weighted Functional Connectivity (TW-FC) maps

This example demonstrates how these maps were derived, *precisely* as performed in the relevant NeuroImage paper. Assumes that you have a whole-brain tractogram named `tracks.tck`, and a 3D volume named `FC_map.mif` representing an extracted FC map with appropriate thresholding.

Initial TWI generation:

```
tckmap tracks.tck temp.mif <-template / -vox options> -contrast scalar_map -image FC_
↪map.mif -stat_vox mean -stat_tck sum
```

Deriving the mask (voxels with at least 5 streamlines with non-zero TW values):

```
tckmap tracks.tck - -template temp.mif -contrast scalar_map_count -image FC_map.mif |␣
↪mrcalc - 5 -ge mask.mif -datatype bit
```

Apply the mask:

```
mrcalc temp.mif mask.mif -mult TWFC.mif
```

Handling SIFT2 weights

With the original *tcksift* command, the output is a *new track file*, which can subsequently be used as input to any command independently of the fact that SIFT has been applied. SIFT2 is a little trickier: the output of the *tcksift2* command is a *text file*. This text file contains one line for every streamline, and each line contains a number; these are the weights of the individual streamlines. Importantly, the track file that was used as input to the *tcksift2* command is *unaffected* by the execution of that command.

There are therefore two important questions to arise from this:

How do I use the output from SIFT2?

Any *MRtrix3* command that receives a track file as input will also have a command-line option, `-tck_weights_in`. This option is used to pass the weights text file to the command. If this option is omitted, then processing will proceed as normal for the input track file, but without taking the weights into consideration.

Why not just add the weight information to the track data?

The `.tck` file format was developed quite a long time ago, and doesn't have the capability of storing such data. Therefore, combining per-streamline weighting data with the track data itself would require either modifying this format (which would break compatibility with MRtrix 0.2, and any other non-MRtrix code that uses this format), using some other existing format for track data (which, given our experiences with image formats, can be ill-devised), or creating a new format (which would need to support a lot more than just per-streamline weights in order to justify the effort, and would likely become a fairly lengthy endeavour).

Furthermore, writing to such a format would require duplicating all of the raw track data from the input file into a new output file. This is expensive in terms of time and HDD space; the original file could be deleted afterwards, but it would then be difficult to perform any operations on the track data where the streamline weight information should be ignored (sure, you could have a command-line option to ignore the weights, but is that any better than having a command-line option to input the weights?)

So, for now, it is best to think of the weights file provided by *tcksift2* as *accompanying* the track file, containing additional data that must be *explicitly* provided to any commands in order to be used. The track file can also be used *without* taking into account the streamline weights, simply by *not* providing the weights.

Making use of Python scripts library

In addition to the principal binary commands, *MRtrix3* also includes a number of Pyton scripts for performing common image processing tasks. These make use of a relatively simple set of library functions that provide a certain leven of convenience and consistency for building such scripts (e.g. common format help page; command-line parsing; creation, use and deletion of temporary working directory; control over command-line verbosity).

It is hoped that in addition to growing in complexity and capability over time, this library may also be of assistance to users when building their own processing scripts, rather than the use of e.g. Bash. The same syntax as that used in the provided scripts can be used. If however the user wishes to run a script that is based on this library, but is *not* located within the *MRtrix3* `scripts/` directory, it is necessary to explicitly inform Python of the location of those libraries; e.g.:

```
export PYTHONPATH=/home/user/mrtrix3/scripts:$PYTHONPATH
./my_script [arguments] (options)
```

(Replace the path to the *MRtrix3* scripts directory with the location of your own installation)

`tck2connectome` no longer has the `-contrast mean_scalar` option...?

The functionality previously provided by this command and option can now be achieved by instead splitting the operation into two independent steps:

```
tcksample tracks.tck scalar.mif mean_scalars.csv -stat_tck mean
tck2connectome tracks.tck nodes.mif connectome.csv -scale_file mean_scalars.csv -stat_
↪edge mean
```

The first step samples the image `scalar.mif` along each streamline, calculates the *mean sampled value along each streamline*, and stores these values into file `mean_scalars.csv` (one value for every streamline). The second step then assigns the value associated with each streamline during connectome construction to be the values from this file, and finally calculates the value of each edge to be the *mean of the values for the streamlines in that edge*.

Maximum spherical harmonic degree `lmax`

What determines `lmax` for my image data?

For any command or script operating on data in the spherical harmonic basis, it should be possible to manually set the maximum harmonic degree of the output using the `-lmax` command-line option. If this is *not* provided, then an appropriate value will be determined automatically.

The mechanisms by which this automatic determination of `lmax` occurs are as follows:

- Determine the maximum value for `lmax` that is supported by the number of DWI volumes in the shell being processed (or the total number of non-*b*=0 volumes in a single-shell acquisition). This is the number of coefficients required to store an anitipodally-symmetric spherical harmonic function:

lmax	Required volumes
2	6
4	15
6	28
8	45
10	66
12	91
...	...

- If `lmax` exceeds 8, reduce to 8. This is primarily based on the findings in this paper.
- Check the condition of the transformation between DWIs and spherical harmonics. If the transformation is ill-conditioned (usually indicating that the diffusion sensitisation gradient directions are not evenly distributed over the sphere or half-sphere), reduce `lmax` until the transformation is well-conditioned.

  As an example: concatenating two repeats of a 30 direction acquisition to produce 60 volumes will *not* support an ‘‘lmax‘‘=8 fit: the angular resolution of the data set is equivalent to 30 *unique* directions, and so ‘‘lmax‘‘=6 would be selected (and this would be accompanied by a command-line warning to the user).

- In the case of spherical deconvolution, the `lmax` selected for FOD estimation will also be reduced if `lmax` of the provided response function is less than that calculated as above.

Reduced `lmax` in particular subjects

If you find that certain subjects within a cohort have a reduced `lmax` compared to the rest of the cohort (usually spotted by checking the number of coefficients in the response function), the most likely cause is premature termination of the diffusion sequence during scanning of that subject, resulting in a reduced number of diffusion volumes and therefore a reduced `lmax` according to the table above.

Setting `lmax` in different applications

The range of permissible values for `lmax` depends on the particular command being used; e.g.:

- The way that response function estimation is currently implemented, it is impossible to set `lmax` to a value higher than that supported by the image data. The transformation from DWI data to spherical harmonics simply cannot be done in such a case, as the problem is under-determined. You can of course set `lmax` to a lower value than that supported by the data.
- In spherical deconvolution, it *is* possible to set a higher `lmax` than that supported by the data - so-called *super-resolved* spherical deconvolution. Here, additional information is provided by the non-negativity constraint to make estimation of additional spherical harmonic coefficients possible. However this is not guaranteed: sometimes the algorithm will fail in particular voxels, in cases where there are an insufficient number of directions in which the initial FOD estimate is negative, as the problem remains under-determined.
- If performing Track Orientation Density Imaging (TODI) using `tckgen -tod`, then the apodized point spread functions (aPSFs) can be generated at any value of `lmax`, since the angular resolution of the original image data is not a limiting factor here.

Visualising streamlines terminations

I am frequently asked about Figures 5-7 in the Anatomically-Constrained Tractography article, which demonstrate the effects that the ACT method has on the locations of streamlines terminations. There are two different techniques used in these figures, which I'll explain here in full.

- Figure 6 shows *streamlines termination density maps*: these are 3D maps where the intensity in each voxel reflects the number of streamlines terminations within that voxel. So they're a bit like Track Density Images (TDIs), except that it's only the streamlines termination points that contribute to the map, rather than the entire streamline. The easiest way to achieve this approach is with the `tckmap` command, using the `-ends_only` option.
- Figures 5 and 7 display large dots at the streamline endpoints lying within the displayed slab, in conjunction with the streamlines themselves and a background image. Unfortunately this functionality is not yet implemented within *MRtrix3*, so duplicating this type of visualisation requires a bit of manual manipulation and software gymnastics:
  - Use the new `tckresample` command, with the `-endpoints` option, to generate a new track file that contains only the two endpoints of each streamline.
  - Load this track file into the *old MRtrix 0.2 version of ''mrview''*. This software can be acquired here. Note that you will likely want to *not* run the installation component of the build for this software; that way you should not encounter issues with conflicting commmand names between MRtrix versions. This does however mean that you will need to provide the full path to the MRtrix 0.2 `mrview` executable in order to run it.

- Within the `mrview` tractography tool, enable the 'depth blend' option. This will display each streamline point as a dot, rather than drawing lines between the streamline points.
- Adjust the brightness / contrast of the background image so that it is completely black.
- Take a screenshot.
- Remove the streamline endpoints track file from the tractography tool, and disable the 'depth blend' option (it's best to disable the 'depth blend' option before opening any larger track file).
- Reset the windowing of the main image, and/or load the complete tracks file, and take an additional screenshot, making sure not to move the view focus or resize the `mrview` window (so that the two screenshots overlay on top of one another).
- The two screenshots are then combined using image editing software such as GIMP. The colors of the termination points can also be modified independently before they are combined with the second screenshot. One trick I used in this manuscript was to rotate the hue of the termination screenshot by 180 degrees: this provides a pseudo-random coloring of the termination points that contrasts well against the tracks.

CHAPTER 19

DWI Pre-processing for Quantitative Analysis

Introduction

This tutorial explains the required pre-processing steps for downstream applications that depend on FOD images for quantitative analysis (e.g. Fixel-Based Analysis of Apparent Fibre Density, as well as SIFT-based connectome analysis).

Most DWI models derive quantitative measures by using the ratio of the DW signal to the b=0 signal within each voxel. This voxel-wise b=0 normalisation implicitly removes intensity variations due to T2-weighting and RF inhomogeneity. However, unless all compartments within white matter are modelled accurately (e.g. intra- and extra-axonal space, myelin, cerebral spinal fluid (CSF) and grey matter partial volumes), the proportion of one compartment in a voxel may influence another. For example, if CSF partial volume at the border of white matter and the ventricles is not taken into account, then a voxel-wise normalisation performed by dividing by the b=0 (which has a long T2 and appears brighter in CSF than white matter in the T2-weighted b=0 image), will artificially reduce the DW signal from the white matter intra-axonal (restricted) compartment, ultimately changing the derived quantiative measures. Multi-compartment diffusion MRI models aspire to model multiple compartments to mitigate these issues. However, in practice current models are limited by restrictions/assumptions placed on the different compartments, and all require multiple b-value acquisitions and therefore longer scan times.

Apparent Fibre Density (AFD) is a Fibre Orientation Distribution (FOD)-derived measure that was developed to enable fibre-specific quantitative analysis using single-shell HARDI data. AFD is based on the assumption that under certain conditions (high b-value of ~3000 s/mm2, typical clinically achievable diffusion pulse duration, typical axon diameters in the range 1-4μm), the DW signal arising from the extra-axonal space (including CSF) is fully attenuated, and the remaining DW signal is proportional to the intra-axonal volume. Since the spherical deconvolution used to compute FODs is a linear transformation, the integral of the (unnormalised) FOD is proportional to the DW signal summed over all gradient directions (and therefore the intra-axonal space). FOD amplitude along a specific direction (the AFD) is therefore proportional to the intra-axonal volume of axons aligned with this direction. In recent work, we have opted to compute the AFD per fibre population by computing the FOD integral within each FOD lobe (see here for details).

To enable robust quantitative comparisons of AFD across subjects (or AFD-derived quantities such as SIFT-filtered tractograms) there are three steps required:

1. **Bias field correction** to eliminate low frequency intensity inhomogeneities across the image.

2. **Global intensity normalisation** by normalising the median CSF or WM b=0 intensity across all subjects (see below for more details). This avoids the above noted issues with voxel-wise b=0 normalisation, such as CSF partial volume influencing restricted white matter DW signal (and therefore the AFD).
3. **Use the same single fibre response function** in the spherical deconvolution for all subjects. This ensures differences in intra-axonal volume (and therefore DW signal) across subjects are detected as differences in the FOD amplitude (the AFD). See the AFD paper for more details.

Pre-processsing steps

Below is a list of recommended pre-processing steps for quantitative analysis using FOD images. Note that for all MRtrix scripts and commands, additional information on the command usage and available command-line options can be found by invoking the command with the `-help` option.

1. DWI denoising

The effective SNR of diffusion data can be improved considerably by exploiting the redundancy in the data to reduce the effects of thermal noise. This functionality is provided in the command `dwidenoise`:

```
dwidenoise <input_dwi> <output_dwi>
```

Note that this denoising step *must* be performed prior to any other image pre-processing: any form of image interpolation (e.g. re-gridding images following motion correction) will invalidate the statistical properties of the image data that are exploited by `dwidenoise`, and make the denoising process prone to errors. Therefore this process is applied as the very first step.

2. DWI general pre-processing

The `dwipreproc` script is provided for performing general pre-processing of diffusion image data - this includes eddy current-induced distortion correction, motion correction, and (possibly) susceptibility-induced distortion correction. Commands for performing this pre-processing are not yet implemented in *MRtrix3*; the `dwipreproc` script in its current form is in fact an interface to the relevant commands that are provided as part of the FSL package. Installation of FSL (including eddy) is therefore required to use this script, and citation of the relevant articles is also required (see the *dwipreproc* help page).

Usage of this script varies depending on the specific nature of the DWI acquisition with respect to EPI phase encoding - full details are available within the *dwipreproc* help file. Here, only a simple example is provided, where a single DWI series is acquired where all volumes have an anterior-posterior (A>>P) phase encoding direction:

```
dwipreproc AP <input_dwi> <output_dwi> -rpe_none
```

3. Estimate a brain mask

A whole-brain mask is required as input to the subsequent bias field correction step. This can be computed with:

```
dwi2mask <input_dwi> <output_mask>
```

4. Bias field correction

DWI bias field correction is perfomed by first estimating a correction field from the DWI b=0 image, then applying the field to correct all DW volumes. This can be done in a single step using the `dwibiascorrect` script in MRtrix. The script uses bias field correction algorthims available in ANTS or FSL. In our experience the N4 algorithm in ANTS gives superiour results. To install N4 install the ANTS package, then run perform bias field correction on DW images using:

```
dwibiascorrect -ants -mask <input_brain_mask> <input_dwi> <output_corrected_dwi>
```

5. Global intensity normalisation across subjects

The ideal approach is to normalise the median CSF b=0 intensity across all subjects (on the assumption that the CSF T2 is unlikely to be affected by pathology). However, in practice it is difficult to obtain a robust partial-volume-free estimate of the CSF intensity due to the typical low resolution of DW images. For participants less than 50 years old (with reasonably small ventricles), it can be difficult to identify pure CSF voxels at 2-2.5mm resolutions. We therefore recommend performing a global intensity normalisation using the median white matter b=0 intensity. While the white matter b=0 intensity may be influenced by pathology-induced changes in T2, our assumption is that such changes will be local to the pathology and therefore have little influence on the median b=0 value.

We have included the `dwiintensitynorm` script in MRtrix to perform an automatic global normalisation using the median white matter b=0 value. The script input requires two folders: a folder containing all DW images in the study (in .mif format) and a folder containing the corresponding whole brain mask images (with the same filename prefix). The script runs by first computing diffusion tensor Fractional Anisotropy (FA) maps, registering these to a study-specific template, then thresholding the template FA map to obtain an approximate white matter mask. The mask is then transformed back into the space of each subject image and used in the `dwinormalise` command to normalise the input DW images to have the same b=0 white matter median value. All intensity normalised data will be output in a single folder:

```
dwiintensitynorm <input_dwi_folder> <input_brain_mask_folder> <output_normalised_dwi_
↪folder> <output_fa_template> <output_template_wm_mask>
```

The dwiintensitynorm script also outputs the study-specific FA template and white matter mask. **It is recommended that you check that the white matter mask is appropriate** (i.e. does not contain CSF or voxels external to the brain. Note it only needs to be a rough WM mask). If you feel the white matter mask needs to be larger or smaller you can re-run `dwiintensitynorm` with a different `-fa_threshold` option. Note that if your input brain masks include CSF then this can cause spurious high FA values outside the brain which will may be included in the template white matter mask.

Keeping the FA template image and white matter mask is also handy if additional subjects are added to the study at a later date. New subjects can be intensity normalised in a single step by piping the following commands together:

```
dwi2tensor <input_dwi> -mask <input_brain_mask> - | tensor2metric - -fa - |␣
↪mrregister <fa_template> - -mask2 <input_brain_mask> -nl_scale 0.5,0.75,1.0 -nl_
↪niter 5,5,15 -nl_warp - tmp.mif | mrtransform <input_template_wm_mask> -template
↪<input_dwi> -warp - - | dwinormalise <input_dwi> - <output_normalised_dwi>; rm tmp.
↪mif
```

Note: The above command may also be useful if you wish to alter the mask then re-apply the intensity normalisation to all subjects in the study. For example you may wish to edit the mask using the ROI tool in `mrview` to remove white matter regions that you hypothesise are affected by the disease (e.g. removing the corticospinal tract in a study of motor neurone disease due to T2 hyperintensity). You also may wish to redefine the mask completely, for example in an elderly population (with larger ventricles) it may be appropriate to intensity normalise using the median b=0 CSF.

This could be performed by manually masking partial-volume-free CSF voxels, then running the above command with the CSF mask instead of the <input_template_wm_mask>.

Warning: We also strongly recommend you that you check the scale factors applied during intensity normalisation are not influenced by the variable of interest in your study. For example if one group contains global changes in white matter T2 then this may directly influence the intensity normalisation and therefore bias downstream results. To check this we recommend you perform an equivalence test to ensure mean scale factors are the same between groups. To output the scale factor applied for each subject use `mrinfo <output_normalised_dwi> -property dwi_norm_scale_factor`.

6. Computing a group average response function

As described here, using the same response function when estimating FOD images for all subjects enables differences in the intra-axonal volume (and therefore DW signal) across subjects to be detected as differences in the FOD amplitude (the AFD). At high b-values (~3000 s/mm2), the shape of the estimated white matter response function varies little across subjects and therefore choosing any single subjects' estimate response is OK. To estimate a response function from a single subject:

```
dwi2response tournier <Input DWI> <Output response text file>
```

Alternatively, to ensure the response function is representative of your study population, a group average response function can be computed by first estimating a response function per subject, then averaging with the script:

```
average_response <input_response_files (mulitple inputs accepted)> <output_group_
↪average_response>
```

CHAPTER 20

Fixel-Based Analysis (FBA)

Introduction

This tutorial explains how to perform fixel-based analysis using MRtrix commands. While the focus here is on the analysis of Apparent Fibre Density (AFD) derived from FODs, the majority of the steps in this tutorial are generic and therefore a similar process can be used to investigate other fixel-based measures derived from other diffusion MRI models.

Warning: This tutorial assumes you have already pre-processed your data to remove artefacts (e.g. eddy-current and magnetic suseptibility-induced distortions and subject motion). If performing analysis of AFD derived from FODs, then additional pre-processing is required and explained in this tutorial. Please post any questions or issues on the MRtrix community forum.

Fixel-based analysis steps

Warning: The following steps and commands are pre-release only. It is likely that some command and option names will change over the next few months, however the overall process will remain the same. We recommend you don't update MRtrix half way through a study, and look out for update announcements on the MRtrix community and blog pages.

Note that for all MRtrix scripts and commands, additional information on the command usage and available command-line options can be found by invoking the command with the `-help` option.

1. Upsampling DW images

Upsampling DWI data before computing FODs can increase anatomical contrast and improve downstream spatial normalisation and statistics. We recommend upsampling by a factor of two using bspline interpolation:

```
mrresize <input_dwi> -scale 2.0 <output_upsampled_dwi>
```

2. Compute upsampled brain mask images

Compute a whole brain mask from the upsampled DW images:

```
dwi2mask <input_upsampled_dwi> <output_upsampled_mask>
```

3. Fibre Orientation Distribution estimation

This command performs Constrained Spherical Deconvolution (CSD) using the group average response function estimated previously. Note that `dwi2fod csd` can be used, however here we use `dwi2fod msmt_csd` (even with single shell data) to benefit from the hard non-negativity constraint:

```
dwiextract <input_upsampled_dwi> - | dwi2fod msmt_csd - <group_average_response_text_
↪file> <output_fod_image> -mask <input_upsampled_mask>
```

4. Generate a study-specific unbiased FOD template

Population template creation is the most time consuming step in a fixel-based analysis. If you have a large number of subjects in your study, we recommend building the template from a subset of 20-40 individuals. Subjects should be chosen to ensure the generated template is representative of your population (i.e. equal number of patients and controls). To build a template, place all FOD images in a single folder. We also recommend placing a set of corresponding mask images (with the same prefix as the FOD images) in another folder. Using masks can speed up registration significantly. Run the population_template building script as follows:

```
population_template <input_folder_of_FOD_images> -mask_dir <input_mask_folder>
↪<output_fod_template_image>
```

5. Register all subject FOD images to the FOD template

Register the FOD image from all subjects to the FOD template image:

```
mrregister <input_fod_image> -mask1 <input_subject_mask> <input_fod_template_image> -
↪nl_warp <subject2template_warp> <template2subject_warp>
```

6. Compute the intersection of all subject masks in template space

Different subjects will have subtly different brain coverage. To ensure subsequent analysis is performed in voxels that contain data from all subjects, we warp all subject masks into template space and compute the mask intersection. For each subject:

```
mrtransform <input_upsampled_mask_image> -warp <subject2template_warp> -interp␣
↪nearest <output_warped_mask>
```

Compute the intersection of all warped masks:

```
mrmath <input_all_warped_masks_multiple_inputs> min <output_template_mask_
↪intersection>
```

7. Compute a white matter template analysis fixel mask

Here we perform a 2-step threshold to identify template white matter fixels to be included in the analysis. Fixels in the template fixel analysis mask are also used to identify the best fixel correspondence across all subjects (i.e. match fixels across subjects within a voxel).

Compute a template AFD peaks fixel image:

```
fod2fixel <input_fod_template_image> -mask <input_template_mask_intersection> -peak
↪<template_peaks_image.msf>
```

Note: Fixel images in MRtrix must be stored using the .msf (MRtrix sparse format) extension.

Next view the peaks file using the vector plot tool in mrview and identify an appropriate threshold that removes peaks from grey matter, yet does not introduce any 'holes' in your white matter (approximately 0.33).

Threshold the peaks fixel image:

```
fixelthreshold -crop <template_peaks_image.msf> 0.33 <analysis_fixel_mask.msf>
```

Generate an analysis voxel mask from the fixel mask. The median filter in this step should remove spurious voxels outside the brain, and fill in the holes in deep white matter where you have small peaks due to 3-fibre crossings:

```
fixel2voxel <analysis_fixel_mask.msf> count - | mrthreshold - - -abs 0.5 | mrfilter -␣
↪median <output_analysis_voxel_mask>
```

Recompute the fixel mask using the analysis voxel mask. Using the mask allows us to use a lower AFD threshold than possible in the steps above, to ensure we have included fixels with low AFD inside white matter:

```
fod2fixel -mask <input_analysis_voxel_mask> <input_fod_template_image> -peak <output_
↪temp.msf>
fixelthreshold <input_temp.msf> -crop 0.2 <output_analysis_fixel_mask.msf> -force
rm <temp.msf>
```

Note: We recommend having no more than 500,000 fixels in the analysis_fixel_mask (you can check this with `fixelstats`), otherwise downstream statistical analysis (using `fixelcfestats`) will run out of RAM). A mask with 500,000 fixels will require a PC with 128GB of RAM for the statistical analysis step.

8. Transform FOD images to template space

Note that here we transform FOD images into template space *without* FOD reorientation. Reorientation will be performed in a separate subsequent step:

```
mrtransform <input_subject_fod_image> -warp <subject2template_warp> -noreorientation
↪<output_warped_fod_image>
```

9. Segment FOD images to estimate fixels and their fibre density (FD)

Here we segment each FOD lobe to identify the number and orientation of fixels in each voxel. The output also contains the apparent fibre density (AFD) value per fixel estimated as the FOD lobe integral (see here for details on FOD segmentation). Note that in the following steps we will use a more generic shortened acronym - Fibre Density (FD) instead of AFD for consistency with our recent work (paper under review):

```
fod2fixel <input_warped_fod_image> -mask <input_analysis_voxel_mask> -afd <output_fd_
↪not_reoriented.msf>
```

Note: If you would like to perform fixel-based analysis of metrics derived from other diffusion MRI models (e.g. CHARMED), replace steps 8 & 9. For example, in step 8 you can warp preprocessed DW images (also without any reorientation). In step 9 you could then estimate your DWI model of choice.

10. Reorient fixel orientations

Here we reorient the direction of all fixels based on the Jacobian matrix (local affine transformation) at each voxel in the warp:

```
fixelreorient <input_fd_not_reoriented.msf> <subject2template_warp> <output_fd_
↪reoriented.msf>
```

11. Assign subject fixels to template fixels

In step 8 we obtained spatial correspondence between subject and template. In step 10 we corrected the fixel orientations to ensure angular correspondence of the segmented peaks of subject and template. Here, for each fixel in the template fixel analysis mask, we identify the corresponding fixel in each voxel of the subject image and assign the FD value of the subject fixel to the corresponding fixel in template space. If no fixel exists in the subject that corresponds to the template fixel then it is assigned a value of zero. See this paper for more information:

```
fixelcorrespondence <input_fd_reoriented.msf> <input_analysis_fixel_mask.msf> <output_
↪fd.msf>
```

12. Compute fibre cross-section (FC) metric

Apparent fibre density, and other related measures that are influenced by the quantity of restricted water, only permit the investigation of group differences in the number of axons that manifest as a change to *within-voxel* density. However, depending on the disease type and stage, changes to the number of axons may also manifest as macroscopic differences in brain morphology. This step computes a fixel-based metric related to morphological differences in fibre cross-section, where information is derived entirely from the warps generated during registration (paper under review):

```
warp2metric <subject2template_warp> -fc <input_analysis_fixel_mask.msf> <output_fc.
↪msf>
```

The FC files will be used in the next step. However, for group statistical analysis of FC we recommend taking the log (FC) to ensure data are centred about zero and normally distributed:

```
fixellog <input_fc.msf> <output_log_fc.msf>
```

13. Compute a combined measure of fibre density and cross-section (FDC)

To account for changes to both within-voxel fibre density and macroscopic atrophy, fibre density and fibre cross-section must be combined (a measure we call fibre density & cross-section, FDC). This enables a more complete picture of group differences in white matter. Note that as discussed in our future work (under review), group differences in FD or FC alone must be interpreted with care in crossing-fibre regions. However group differences in FDC are more directly interpretable. To generate the combined measure we 'modulate' the FD by FC:

```
fixelcalc <input_fd.msf> mult <input_fc.msf> <output_fdc.msf>
```

14. Perform whole-brain fibre tractography on the FOD template

Statistical analysis using connectivity-based fixel enhancement exploits connectivity information derived from probabilistic fibre tractography. To generate a whole-brain tractogram from the FOD template:

```
tckgen -angle 22.5 -maxlen 250 -minlen 10 -power 1.0 <input_fod_template_image> -seed_
↪image <input_analysis_voxel_mask> -mask <input_analysis_voxel_mask> -number␣
↪20000000 <output_tracks_20_million.tck>
```

15. Reduce biases in tractogram densities

Perform SIFT to reduce tractography biases in the whole-brain tractogram:

```
tcksift <input_tracks_20_million.tck> <input_fod_template_image> <output_tracks_2_
↪million_sift.tck> -term_number 2000000
```

16. Perform statistical analysis of FD, FC, and FDC

You will need to perform a separate analysis for FD, FC and FDC. Statistics is performed using connectivity-based fixel enhancement as follows:

```
fixelcfestats <input_files> <input_analysis_fixel_mask.msf> <input_design_
↪matrix.txt> <output_contrast_matrix.txt> <input_tracks_2_million_sift.tck>
↪<output_prefix>
```

Where the input files.txt is a text file containing the file path and name of each input fixel file on a separate line. The line ordering should correspond to the lines in the design_matrix.txt. Note that for correlation analysis, a column of 1's will not be automatically included (as per FSL randomise). Note that fixelcfestats currently only accepts a single contrast. However if the opposite (negative) contrast is also required (i.e. a two-tailed test), then use the `-neg` option. Several output files will generated all starting with the supplied prefix.

17. Visualise the results

To view the results load the population FOD template image in `mrview`, and overlay the fixel images using the vector plot tool. Note that p-value images are saved as 1-p-value. Therefore to visualise all p-values < 0.05, threshold the fixels using the vector plot tool at 0.95.

CHAPTER 21

Anatomically-Constrained Tractography (ACT)

This page describes the recommended processing steps for taking advantage of the Anatomically-Constrained Tractography (ACT) framework, the image format used, and the commands available for manipulating these data. There are also instructions for anyone looking to make use of alternative tissue segmentation approaches.

References

For full details on ACT, please refer to the following journal article:

> Smith, R. E., Tournier, J.-D., Calamante, F., & Connelly, A. (2012). Anatomically-constrained tractography: Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 62(3), 1924–1938. doi:10.1016/j.neuroimage.2012.06.005

If you use ACT in your research, please cite the article above in your manuscripts.

Pre-processing steps

DWI distortion correction

For the anatomical information to be incorporated accurately during the tractography reconstruction process, any geometric distortions present in the diffusion images must be corrected. The FSL 5.0 commands `topup` and `eddy` are effective in performing this correction based on a reversed phase-encode acquisition, though their interfaces can be difficult to figure out.

A common strategy is to acquire a pair of *b=0* images, the first with phase encode A>>P and the second P>>A, followed by the DWI acquisition with phase encode A>>P; the provided script `dwipreproc` using the `-rpe_pair` option interfaces with FSL to perform the distortion correction in this particular case. For alternative acquisitions, see the help page of the `dwipreproc` script.

Image registration

My personal preference is to register the T1-contrast anatomical image to the diffusion image series before any further processing of the T1 image is performed. By registering the T1 image to the diffusion series rather than the other way around, reorientation of the diffusion gradient table is not necessary; and by doing this registration before subsequent T1 processing, any subsequent images derived from the T1 are inherently aligned with the diffusion image series. This registration should be rigid-body only; if the DWI distortion correction is effective, a higher-order registration is likely to only introduce errors.

DWI pre-processing

Because the anatomical image is used to limit the spatial extent of streamlines propagation rather than a binary mask derived from the diffusion image series, I highly recommend dilating the DWI brain mask prior to computing FODs; this is to make sure that any errors in derivation of the DWI mask do not leave gaps in the FOD data within the brain white matter, and therefore result in erroneous streamlines termination.

Tissue segmentation

So far I have had success with using FSL tools to also perform the anatomical image segmentation; FAST is not perfect, but in most cases it's good enough, and most alternative software I tried provided binary mask images only, which is not ideal. The `5ttgen` script using the `fsl` algorithm interfaces with FSL to generate the necessary image data from the raw T1 image, using BET, FAST and FIRST. Note that this script also crops the resulting image so that it contains no more than the extracted brain (as this reduces the file size and therefore improves memory access performance during tractography); if you want the output image to possess precisely the same dimensions as the input T1 image, you can use the `-nocrop` option.

Using ACT

Once the necessary pre-processing steps are completed, using ACT is simple: just provide the tissue-segmented image to the `tckgen` command using the `-act` option.

In addition, since the propagation and termination of streamlines is primarily handled by the 5TT image, it is no longer necessary to provide a mask using the `-mask` option. In fact, for whole-brain tractography, it is recommend that you _not_ provide such an image when using ACT: depending on the accuracy of the DWI brain mask, its inclusion may only cause erroneous termination of streamlines inside the white matter due to exiting this mask. If the mask encompasses all of the white matter, then its inclusion does not provide any additional information to the tracking algorithm.

The 5TT format

When the ACT framework is invoked, it expects the tissue information to be provided in a particular format; this is referred to as the 'five-tissue-type (5TT)' format. This is a 4D, 32-bit floating-point image, where the dimension of the fourth axis is 5; that is, there are five 3D volumes in the image. These five volumes correspond to the different tissue types. In all brain voxels, the sum of these five volumes should be 1.0, and outside the brain it should be zero. The tissue type volumes must appear in the following order for the anatomical priors to be applied correctly during tractography:

0. Cortical grey matter
1. Sub-cortical grey matter

2. White matter
3. CSF
4. Pathological tissue

The first four of these are described in the ACT NeuroImage paper. The fifth can be optionally used to manually delineate regions of the brain where the architecture of the tissue present is unclear, and therefore the type of anatomical priors to be applied are also unknown. For any streamline entering such a region, *no anatomical priors are applied* until the streamline either exists that region, or stops due to some other streamlines termination criterion.

The following binaries are provided for working with the 5TT format:

- `5tt2gmwmi`: Produces a mask image suitable for seeding streamlines from the grey matter - white matter interface (GMWMI). The resulting image should then be provided to the `tckgen` command using the `-seed_gmwmi` option.
- `5tt2vis`: Produces a 3D greyscale image suitable for visualisation purposes.
- `5ttedit`: Allows the user to edit the tissue segmentations. Useful for manually correcting tissue segmentations that are known to be erroneous (e.g. dark blobs in the white matter being labelled as grey matter); see the command's help page for more details.

Alternative tissue segmentation software

Users who wish to experiment with using tissue segmentations from different software sources are encouraged to do so; if a particular approach is shown to be effective we can add an appropriate script to MRtrix. The `5ttgen` script has a second algorithm, `freesurfer`, which demonstrates how the output of different software can be manipulated to provide the tissue segmentations in the appropriate format. It is however not recommended to actually use this alternative algorithm for patient studies; many midbrain structures are not segmented by FreeSurfer, so the tracking may not behave as desired.

Users who wish to try manipulating the tissue segmentations from some alternative software into the 5TT format may find it most convenient to make a copy of one of the existing algorithms within the `scripts/src/_5ttgen//` directory, and modify accordingly. The `5ttgen` script will automatically detect the presence of the new algorithm, and make it available at the command-line.

CHAPTER 22

Spherical-deconvolution Informed Filtering of Tractograms (SIFT)

SIFT, or 'Spherical-deconvolution Informed Filtering of Tractograms', is a novel approach for improving the quantitative nature of whole-brain streamlines reconstructions. By producing a reconstruction where the streamlines densities are proportional to the fibre densities as estimated by spherical deconvolution throughout the white matter, the number of streamlines connecting two regions becomes a proportional estimate of the cross-sectional area of the fibres connecting those two regions. We therefore hope that this method will attract usage in a range of streamlines tractography applications.

The actual usage of SIFT can be found in the help page of the `tcksift` command. In this page I'll outline some issues that are worth thinking about if you are looking to apply this method.

References

For full details on SIFT, please refer to the following journal article:

> Smith, R. E., Tournier, J.-D., Calamante, F., & Connelly, A. (2013). SIFT: Spherical-deconvolution informed filtering of tractograms. NeuroImage, 67, 298–312. doi:10.1016/j.neuroimage.2012.11.049

If you use SIFT in your research, please cite the article above in your manuscripts.

DWI bias field correction

DWI volumes often have a non-negligible *B1* bias field, mostly due to high-density receiver coils. If left uncorrected, SIFT will incorrectly interpret this as a spatially-varying fibre density. Therefore bias field correction is highly recommended. We generally estimate the bias field based on the mean *b=0* image, and apply the estimated field to all DWI volumes. This can currently be achieved using the `dwibiascorrect` script, which can employ either the FAST tool in FSL or the N4 algorithm in ANTS to perform the field estimate.

Number of streamlines pre / post SIFT

In diffusion MRI streamlines tractography, we generate discrete samples from a continuous fibre orientation field. The more streamlines we generate, the better our reconstruction of that field. Furthermore, the greater number of streamlines we generate, the less influence the discrete quantification of connectivity has on the connectome (e.g. would rather be comparing 1,000 v.s. 2,000 streamlines to 1 v.s. 2; it's less likely to be an artefact of random / discrete sampling). So the more streamlines the better, at the cost of execution speed & hard drive consumption.

However we also have the added confound of SIFT. The larger the number of streamlines that can be fed to SIFT the better, as it can make better choices regarding which streamlines to keep/remove; but it also introduces a memory constraint. SIFT can deal with approximately 4-8 million streamlines per GB of RAM (depending on the seeding mechanism used and the spatial resolution of your diffusion images), so ideally you'll want access to dedicated high-performance computing hardware. On top of this, there's the issue of how many streamlines to have remaining in the reconstruction after SIFT; the more streamlines that SIFT removes, the better the streamlines reconstruction will fit the image data, but the more likely you are to run into quantisation issues with the resulting tractogram.

So when you design your image processing pipeline, you need to consider the compromise between these factors:

- Initially generating a larger number of streamlines is beneficial for both the quality and the density of the filtered reconstruction, at the expense of longer computation time (both in generating the streamlines, and running SIFT), and a higher RAM requirement for running SIFT.
- Filtering a greater number of streamlines will always produce a superior fit to the image data, at the expense of having a lower-density reconstruction to work with afterwards, and a slightly longer computation time.

Unfortunately there's no single answer of how many streamlines are required, as it will depend on the diffusion model, tractography algorithm, and spatial extent of your target regions / connectome parcellation granularity. There are a couple of papers / abstracts on the topic if you look hard enough, but nothing definitive, and nothing involving SIFT. I would recommend testing using your own data to find numbers that are both adequate in terms of test-retest variability, and computationally reasonable.

Personally I have been using a FreeSurfer parcellation (84 nodes), generating 100 million streamlines and filtering to 10 million using SIFT (I'm a physicist; I like orders of magnitude). In retrospect, I would say that when using white matter seeding, filtering by a factor of 10 is inadequate (i.e. the fit of the reconstruction to the data is not good enough); and with grey matter - white matter interface seeding, a final number of 10 million is inadequate (the streamlines are mostly very short, so the appearance of the reconstruction is quite sparse). Another alternative is 'dynamic seeding', which uses the SIFT model during tractogram generation to only seed streamlines in pathways that are poorly reconstructed (see the `-seed_dynamic` option in `tckgen`); this provides a better initial estimate, so the percentage of streamlines that need to be removed in order to achieve a good fit is reduced. I will leave it to the end user to choose numbers that they deem appropriate (unless we do a paper on the topic, in which case you will use our published values without question).

Normalising connection density between subjects

An ongoing issue with our Apparent Fibre Density (AFD) work is how to guarantee that a smaller FOD in a subject actually corresponds to a reduced density of fibres. Structural connectome studies have a similar issue with regards to streamline counts; Even if SIFT is applied, this only guarantees correct proportionality between different connection pathways within a subject, not necessarily between subjects. The simplest and most common solution is simply to use an identical number of streamlines for every subject in connectome construction; however this isn't perfect:

- The distribution of streamlines lengths may vary between subjects, such that the reconstructed streamlines 'density' differs.
- A subject may have decreased fibre density throughout the brain, but be morphologically normal; if the same number of streamlines are generated, this difference won't be reflected in the tractogram post-SIFT.

- If the white matter volume varies between subjects, but the actual number of fibres within a given volume is consistent, then the subject with a larger brain may have an elevated total number of fibre connections; this would also be missed if the number of streamlines were fixed between subjects.

It's also possible to scale by the total white matter volume of each subject; this would however fail to take into account any differences in the density of fibres within a fixed volume between subjects.

An alternative approach is to try to achieve normalisation of FOD amplitudes across subjects, as is done using AFD. This requires a couple of extra processing steps, namely inter-subject intensity normalisation and use of a group average response function, which are also far from error-free. But if this can be achieved, it means that a fixed density of streamlines should be used to reconstruct a given FOD amplitude between subjects, and then the cross-sectional area of fibres represented by each streamline is also identical between subjects; this can be achieved by terminating SIFT at a given value of the proportionality coefficient using the `-term_mu` option. One potential disadvantage of this approach (in addition to the issues associated with intensity normalisation) is that using a group average response function instead of the individual subject response may result in spurious peaks or incorrect relative volume fractions in the FODs, which could influence the tracking results.

Ideally, a diffusion model would provide the absolute partial volume of each fibre population, rather than a proportional quantity: this could then be used directly in SIFT. However the diffusion models that do provide such information tend to get the crossing fibre geometry wrong in the first place...

If anyone has any ideas on how to solve this pickle, let us know.

No DWI distortion correction available

SIFT should ideally be used in conjunction with ACT; by passing the ACT 5TT image to `tcksift` using the `-act` option, the command will automatically derive a processing mask that will limit the contribution of non-pure-white-matter voxels toward the model. Without this information, non-pure-white-matter voxels adversely affect both streamlines tractography, and the construction of the SIFT model.

If you are looking to apply SIFT without correction of DWI geometric distortions (and therefore without reliable high-resolution co-registered anatomical image data), these are some points that you may wish to consider:

- The spatial extent of the DWI mask may have a large influence on your streamlines tractography results. Therefore greater care should perhaps be taken to validate this mask, including manual editing if necessary.
- It is possible to manually provide a processing mask to `tcksift` using the `-proc_mask` option. If users are capable of heuristically generating an approximate white matter partial volume image from the DWI data alone, this may be appropriate information to provide to the SIFT model.

Use of SIFT for quantifying pathways of interest

In some circumstances, researchers may be interested in the connection density of one or two specific pathways of interest, rather than that of the whole brain. SIFT is still applicable in this scenario; however the SIFT algorithm itself is only applicable to whole-brain fibre-tracking data. Therefore, the workflow in this scenario should be: * Generate a whole-brain tractogram; * Apply SIFT; * Extract the pathway(s) of interest using `tckedit`. * Get the streamline count using `tckinfo`.

The SIFT algorithm is *not directly applicable to targeted tracking data*. The underlying biophysical model in SIFT assumes that the estimated density of each fibre population in every voxel of the image should be proportionally reconstructed by streamlines; if only a subset of pathways in the brain are permitted to be reconstructed by the tractography algorithm, this will clearly not be the case, so appplication of SIFT in this instance will provide erroneous results.

CHAPTER 23

Structural connectome construction

Included in this new version of MRtrix are some useful tools for generating structural connectomes based on streamlines tractography. Here I will describe the steps taken to produce a connectome, and some issues that should be taken into consideration. Note that I will **not** be going into appropriate parcellations or network measures or anything like that; once you've generated your connectomes, you're on your own.

Preparing a parcellation image for connectome generation

Parcellations are typically provided as an integer image, where each integer corresponds to a particular node, and voxels where there is no parcellation node have a value of 0. However, for all of the parcellation schemes I've looked at thus far, the values used for the nodes do not increase monotonically from 1, but rather have some non-linear distribution; a text file (or 'lookup table') is then provided that links node indices to structure names. This is however undesirable for connectome construction; it would be preferable for the node indices to increase monotonically from 1, so that each integer value corresponds to a row/column position in the connectome matrix.

This functionality is provided in the command `labelconvert`. It takes as its input a parcellation image that has been provided by some other software package, and converts the label indices; this is done so that the code that actually generates the connectome can be 'dumb and blind', i.e. the integer values at the streamline endpoints correspond to the row & column of the connectome matrix that should be incremented. In addition, this processing chain design provides flexibility in terms of both the source of the parcellation data, and the way in which the user wishes to customise the layout of their connectome.

Please consult the tutorial *Structural connectome construction: Using labelconvert* for a guide on how to use the `labelconvert` command.

Generating the connectome

The command `tck2connectome` is responsible for converting the tractogram into a connectome matrix, based on the provided parcellation image. By default, the streamline count is used as the connectivity metric; run `tck2connectome -help` to see alternative heuristics / measures.

A factor in structural connectome production commonly overlooked or not reported in the literature is the mechanism used to assign streamlines to grey matter parcels. If done incorrectly this can have a large influence on the resulting connectomes. This is one aspect where *Anatomically-Constrained Tractography (ACT)* really shines; because streamlines can only terminate precisely at the grey matter - white matter interface, within sub-cortical grey matter, or at the inferior edge of the image, this assignment becomes relatively trivial. The default assignment mechanism is a radial search outwards from the streamline termination point, out to a maximum radius of 2mm; and the streamline endpoint is only assigned to the first non-zero node index. If you do not have the image data necessary to use the ACT framework, see the 'No DWI distortion correction available' section below.

SIFT and the structural connectome

If you are generating structural connectomes, you should be using *Spherical-deconvolution Informed Filtering of Tractograms (SIFT)*.

Extracting pathways of interest from a connectome

The command `connectome2tck` can be used to extract specific connections of interest from a connectome for further interrogation or visualisation. Note that since the resulting connectome matrix does not encode precisely which parcellation node pair each streamline was assigned to, the streamlines are re-assigned to parcellation nodes as part of this command. Run `connectome2tck -help` to see the various ways in which streamlines may be selected from the connectome.

Also: Beware of running this command on systems with distributed network file storage. This particular command uses an un-buffered file output when writing the streamlines files, which re-opens the output file and writes data for individual streamlines at a time (necessary as many files may be generated at once); such systems tend to be optimised for large-throughput writes, so this command may cause performance issues.

No DWI distortion correction available

If you can't perform DWI susceptibility distortion correction, it severely limits how accurately you can estimate the structural connectome. If this is the case for you, below is a few points that are worth considering.

Non-linear registration

Rather than actually correcting the DWI geometric distortions, some people try to do a non-linear registration between DWI and T1 images. In general I'm against this: the registration is fairly ill-posed due to the differing contrasts, and an off-the-shelf non-linear registration will have too many degrees of freedom. Pursue at your own risk.

Grey matter parcellation

With good spatial alignment, parcellations that highlight only the cortial ribbon (e.g. FreeSurfer) are highly accurate and effective, and the assignment of streamlines to those parcellations will also be robust if ACT is used. But without these, residual registration errors may have a large influence, and assigning streamlines to parcellations only as thick as the cortex may also be erroneous (streamlines may terminate prior to the parcel, or travel through and extend well beyond it). A parcellation with large-volume nodes that is based on atlas registration (e.g. AAL) is likely more appropriate in this case.

Assignment of streamlines to parcellation nodes

Without ACT, streamlines will terminate pretty much anywhere within the DWI brain mask. Not only this, but they may traverse multiple parcellation nodes, turn around within a node and traverse elsewhere, terminate just prior to entering a node, all sorts of weirdness. I have provided a few assignment mechanisms that you can experiment with - run `tck2connectome -help` to see the list and parameters for each. Alternatively if anyone has a better idea for how this could potentially be done, I'd love to hear it.

CHAPTER 24

Global tractography

Introduction

Global tractography is the process of finding the full track configuration that best explains the measured DWI data. As opposed to streamline tracking, global tractography is less sensitive to noise, and the density of the resulting tractogram is directly related to the data at hand.

As of version 3.0, MRtrix supports global tractography using a multi-tissue spherical convolution model, as introduced in *Christiaens et al. (2015)*. This method extends the method of *Reisert et al. (2011)* to multi-shell response functions, estimated from the data, and adopts the multi-tissue model presented in *Jeurissen et al. (2014)* to account for partial voluming.

User guide

The most common use will be:

```
tckglobal dwi.mif wmr.txt -riso csfr.txt -riso gmr.txt -mask mask.mif -niter 1e8 -fod␣
↪fod.mif -fiso fiso.mif tracks.tck
```

In this example, `dwi.mif` is the input dataset, including the gradient table, and `tracks.tck` is the output tractogram. `wmr.txt`, `gmr.txt` and `csfr.txt` are tissue response functions (cf. next section). Optional output images fod.mif and fiso.mif contain the predicted WM fODF and isotropic tissue fractions of CSF and GM respectively, estimated as part of the global optimization and thus affected by spatial regularization.

Input response functions

Input response functions for (single fibre) white matter, grey matter, and CSF can be estimated from the data in prior tissue segmentations, as described in *Jeurissen et al. (2014)* and *Christiaens et al. (2015)*.

Obtaining good segmentations of WM, GM and CSF will typically require T1 data. While MRtrix doesn't implement segmentation methods itself, it does provide a script that calls the relevant FSL or Freesurfer tools to obtain a tissue segmentation in the appropriate format, for example:

```
5ttgen fsl T1.mif 5tt.mif
```

Note that the T1 image must be aligned with (e.g. registered to) the DWI data. See this page for more information.

Response functions for single-fibre WM, GM, and CSF, can then be estimated using:

```
dwi2response msmt_5tt dwi.mif 5tt.mif wm.txt gm.txt csf.txt
```

For a detailed explanation of different strategies for response function estimation, have a look at this page.

Parameters

`-niter`: The number of iterations in the optimization. Although the default value is deliberately kept low, a full brain reconstruction will require at least 100 million iterations.

`-lmax`: Maximal order of the spherical harmonics basis.

`-length`: Length of each track segment (particle), which determines the resolution of the reconstruction.

`-weight`: Weight of each particle. Decreasing its value by a factor of two will roughly double the number of reconstructed tracks, albeit at increased computation time.

Particle potential `-ppot`: The particle potential essentially associates a *cost* to each particle, relative to its weight. As such, we are in fact trying to reconstruct the data as well as possible, with as few particles as needed. This ensures that there is sufficient *proof* for each individual particle, and hence avoids that a bit of noise in the data spurs generation of new (random) particles. Think of it as a parameter that balances sensitivity versus specificity. A higher particle potential requires more *proof* in the data and therefore leads to higher specificity; a smaller value increases sensitivity.

Connection potential `-cpot`: The connection potential is the driving force for connecting segments and hence building tracks. Higher values increase connectivity, at the cost of increased invalid connections.

Ancillary outputs

`-fod`: Outputs the predicted fibre orientation distribution function (fODF) as an image of spherical harmonics coefficients. This fODF is estimated as part of the global track optimization, and therefore incorporates the spatial regularization that it imposes. Internally, the fODF is represented as a discrete sum of apodized point spread functions (aPSF) oriented along the directions of all particles in the voxel, akin to track orientation distribution imaging (TODI, *Dhollander et al., 2014*). This internal representation is used to predict the DWI signal upon every change to the particle configuration.

`-fiso`: Outputs the estimated density of all isotropic tissue components, as multiple volumes in one 4-D image in the same order as their respective `-riso` kernels were provided.

`-eext`: Outputs the residual data energy image, including the L1-penalty imposed by the particle potential.

References

1. D. Christiaens, M. Reisert, T. Dhollander, S. Sunaert, P. Suetens, and F. Maes. *Global tractography of multi-shell diffusion-weighted imaging data using a multi-tissue model.* NeuroImage, 123 (2015) pp. 89–101 [SD link]

2. M. Reisert, I. Mader, C. Anastasopoulos, M. Weigel, S. Schnell, and V. Kiselev. *Global fiber reconstruction becomes practical.* NeuroImage, 54 (2011) pp. 955–962 [SD link]

3. B. Jeurissen, J.D. Tournier, T. Dhollander, A. Connelly, and J. Sijbers. *Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data.* NeuroImage, 103 (2014), pp. 411–426 [SD link]

4. T. Dhollander, L. Emsell, W. Van Hecke, F. Maes, S. Sunaert, and P. Suetens. *Track Orientation Density Imaging (TODI) and Track Orientation Distribution (TOD) based tractography.* NeuroImage, 94 (2014), pp. 312–336 [SD link]

CHAPTER 25

Multi-tissue constrained spherical deconvolution

Introduction

Multi-tissue constrained spherical deconvolution (CSD) of multi-shell data exploits the unique b-value dependencies of the different macroscopic tissue types (WM/GM/CSF) to estimate a multi-tissue orientation distribution function (ODF) as explained in *Jeurissen et al. (2014)*. As it includes separate compartments for each tissue type, it can produce a map of the WM/GM/CSF signal contributions directly from the DW data. In addition, the more complete modelling of the DW signal results in more accurate apparent fiber density (AFD) measures and more precise fibre orientation estimates at the tissue interfaces.

User guide

Multi-tissue CSD can be performed as:

```
dwi2fod msmt_csd dwi.mif wm.txt wm.mif gm.txt gm.mif csf.txt csf.mif
```

where

- `dwi.mif` is the dwi data set (input)
- `<tissue>.txt` is the tissue-specific response function (input)
- `<tissue>.mif` is the tissue-specific ODF (output)

Note that input response functions and their corresponding output ODFs need to be specified in pairs.

Typically, you will also want to use the `-mask` to avoid calculations in non-brain voxels:

```
dwi2fod msmt_csd -mask mask.mif dwi.mif wm.txt wm.mif gm.txt gm.mif csf.txt csf.mif
```

RGB tissue signal contribution maps can be obtained as follows:

```
mrconvert -coord 3 0 wm.mif - | mrcat csf.mif gm.mif - vf.mif
```

The resulting WM fODFs can be displayed together with the tissue signal contribution map as:

```
mrview vf.mif -odf.load_sh wm.mif
```

Per tissue response function estimation

Input response functions for CSF, GM and single fibre WM can be estimated from the data using prior tissue segmentations, similarly to that described in *Jeurissen et al. (2014)* using the `dwi2response msmt_5tt` command:

```
dwi2response msmt_5tt dwi.mif 5tt.mif wm.txt gm.txt csf.txt
```

where

- `dwi.mif` is the same dwi data set as used above (input)
- `5tt.mif` is a tissue type segmentation of a coregistered T1 data set from the same subject (input)
- `<tissue>.txt` is the tissue-specific response function as used above (output)

Prior tissue type segmentation can be obtained from a structural T1 scan using the *5ttgen* script:

```
5ttgen fsl T1.mif 5tt.mif
```

where

- `T1.mif` is a coregistered T1 data set from the same subject (input)
- `5tt.mif` is the tissue type segmentation used above (output)

The difference between the default behaviour of `dwi2response msmt_5tt` and the method described in *Jeurissen et al. (2014)* is that instead of selecting WM single-fibre voxels using an FA threshold, the `dwi2response tournier` algorithm is instead used.

Note that this process is dependent on accurate correction of EPI geometric distortions, and rigid-body registration between the DWI and T1 modalities, such that the T1 image can be reliably used to select pure-tissue voxels in the DWI volumes. Failure to achieve these may result in inappropriate voxels being used for response function estimation, with concomitant errors in tissue estimates.

References

1. B. Jeurissen, J.D. Tournier, T. Dhollander, A. Connelly, and J. Sijbers. *Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data.* NeuroImage, 103 (2014), pp. 411–426 [SD link]

CHAPTER 26

Diffusion gradient scheme handling

An essential piece of information for DWI processing is the diffusion-weighted (DW) gradient scheme, also known as the "*DW gradient table*", the "*DW encoding*", the "*b-vectors*", the "*bvecs*", and other variations on the theme. This table provides information about the diffusion sensitisation gradients applied during acquisition of each imaging volume in a DWI dataset, usually in the form of the *b*-value and the (unit) vector for the DW gradient direction. In this page we will describe the details of how this information is typically stored / represented, and how *MRtrix3* handles / manipulates this data.

Gradient table storage

MRtrix3 allows the DW gradient table to be read directly from, or written to, the image headers for specific image formats; notably *DICOM (folder or .dcm)* (read-only) and the *MRtrix image formats (.mih / .mif)* (read/write). *MRtrix3* applications will automatically make use of this information when it is available for the input dataset through storage of the table within the *image header*, without requiring explicit intervention from the user. In addition, *MRtrix3* commands can also import or export this information from/to two different external file formats: typically referred to as the *MRtrix format* and the *FSL format*. These differ in a number of respects, as outlined below.

MRtrix format

This format consists of a single ASCII text file, with no restrictions on the filename. It consists of one row per entry (i.e. per DWI volume), with each row consisting of 4 space-separated floating-point values; these correspond to `[ x y z b ]`, where `[ x y z ]` are the components of the gradient vector, and `b` is the *b*-value in units of s/mm^2. A typical *MRtrix* format DW gradient table file might look like this:

Listing 26.1: **grad.b:**

```
         0          0          0          0
         0          0          0          0
-0.0509541  0.0617551   -0.99679       3000
  0.011907   0.955047   0.296216       3000
 -0.525115   0.839985   0.136671       3000
```

```
 -0.785445     -0.6111  -0.0981447        3000
  0.060862   -0.456701    0.887536        3000
  0.398325    0.667699      0.6289        3000
 -0.680604    0.689645   -0.247324        3000
  0.237399    0.969995   0.0524565        3000
  0.697302    0.541873   -0.469195        3000
 -0.868811    0.407442     0.28135        3000

...
```

It is important to note that in this format, the direction vectors are assumed to be provided with respect to *real* or *scanner* coordinates. This is the same convention as is used in the DICOM format. Also note that the file does not need to have the file type extension `.b` (or any other particular suffix); this is simply a historical convention.

Image header

When using the *MRtrix image formats (.mih / .mif)*, *MRtrix3* has the capability of *embedding* the diffusion gradient table *within the header of the image file*. This provides significant advantages when performing image processing:

- The table accompanies the image data at all times, which means that the user is not responsible for tracking which diffusion gradient table corresponds to which image file, or whether or not a particular gradient table file reflects some manipulation that has been applied to an image.
- In *MRtrix3* commands that require a diffusion gradient table, and/or make modifications to the image data that require corresponding modifications to the diffusion gradient table, these data will be utilised (and/or modified) *automatically*, without requiring explicit intervention from the user.

For these reasons, the general recommendation of the *MRtrix3* team is to make use of the *MRtrix image formats (.mih / .mif)* whenever possible.

This embedding is achieved by writing an entry into the Image *Header key-value pairs*, using the key `dw_scheme`. The value of this entry is the complete diffusion gradient table, stored in the *MRtrix format*. However, this entry should generally *not be accessed or manipulated directly* by users; instead, users should rely on the internal handling of these data as performed by *MRtrix3* commands, or where relevant, use the command-line options provided as part of specific *MRtrix3 commands*, as detailed later.

FSL format

This format consists of a pair of ASCII text files, typically named `bvecs` & `bvals` (or variations thereof). The `bvals` file consists of a single row of space-separated floating-point values, all in one row, with one value per volume in the DWI dataset. The `bvecs` file consists of 3 rows of space-separated floating-point values, with the first row corresponding to the *x*-component of the DW gradient vectors, one value per volume in the dataset; the second row corresponding to the *y*-component, and the third row to the *z*-component. A typical pair of FSL format DW gradient files might look like:

Listing 26.2: **bvecs:**

```
0 0 -4.30812931665e-05 -0.00028279245503 -0.528846962834659 -0.781281266220383 0.
↪014299684287952  0.36785999072309 -0.66507232482745  0.237350171404029  0.
↪721877079467007 -0.880754419294581 0 -0.870185851757858 ...
0 0 -0.002606397951389 -0.97091525561761 -0.846605326714759  0.615840299891175 0.
↪403330065122241 -0.70377676751476 -0.67378508548543 -0.971399047063277 -0.
↪513131073140676 -0.423391107245363 0 -0.416501756655988 ...
0 0 -0.999996760803023  0.23942421337746  0.059831733802001 -0.101684552642539 0.
↪914942902775223  0.60776414747636 -0.32201498900359  0.007004078617919 -0.
↪464317089148873  0.212157919445896 0 -0.263255013300656 ...
```

Listing 26.3: **bvals:**

```
0 0 3000 3000 3000 3000 3000 3000 3000 3000 3000 3000 ...
```

It is important to note that in this format, the gradient vectors are provided *with respect to the image axes*, **not** in real or scanner coordinates (actually, it's a little bit more complicated than that, refer to the FSL wiki for details). This is a rich source of confusion, since seemingly innocuous changes to the image can introduce inconsistencies in the *b*-vectors. For example, simply reformatting the image from sagittal to axial will effectively rotate the *b*-vectors, since this operation changes the image axes. It is also important to remember that a particular `bvals/bvecs` pair is only valid for the particular image that it corresponds to.

Using the DW gradient table in *MRtrix3* applications

Querying the DW gradient table

As mentioned above, *MRtrix3* will use the DW gradient table from the image headers when it is available. Currently, only the *DICOM (folder or .dcm)* and *MRtrix image formats (.mih / .mif)* support this. The DW gradient table can be queried for any particular image using the *mrinfo* command in combination with the `-dwgrad` option. For example:

```
$ mrinfo DICOM/ -dwgrad
mrinfo: [done] scanning DICOM folder "DICOM/"
mrinfo: [100%] reading DICOM series "BRI 64 directions ep2d_diff_3scan_trace_p2"
          0           0           0           0
  -0.999994  0.00167109  0.00300897        3000
         -0    0.999996  0.00299996        3000
  0.0261389     0.65148   -0.758215        3000
  -0.590138   -0.767763   -0.249553        3000
   0.236087   -0.527069   -0.816371        3000
   0.893005   -0.261931    -0.36597        3000
  -0.797405    0.126351   -0.590068        3000
  -0.233751    0.930868   -0.280794        3000
  -0.936406    0.141569   -0.321095        3000
  -0.505355   -0.845584     0.17206        3000
  -0.346203   -0.848909     0.39937        3000
  -0.457204   -0.633042    0.624678        3000
    0.48716   -0.391994   -0.780395        3000
   0.617871    0.674589   -0.403938        3000
   0.577709   -0.102522    0.809779        3000
   0.825818   -0.523076   -0.210752        3000

...
```

Exporting the DW gradient table

This information can also be exported from the image headers using the `-export_grad_mrtrix` option (for the *MRtrix format*) or `-export_grad_fsl` option (for the *FSL format*) in commands that support it. For example:

```
$ mrinfo dwi.mif -export_grad_mrtrix grad.b
```

results in a `grad.b` file in *MRtrix format*, while:

```
$ mrconvert DICOM/ dwi.nii.gz -export_grad_fsl bvecs bvals
mrconvert: [done] scanning DICOM folder "DICOM/"
mrconvert: [100%] reading DICOM series "BRI 64 directions ep2d_diff_3scan_trace_p2"
mrconvert: [100%] reformatting DICOM mosaic images
mrconvert: [100%] copying from "DICOM data...ns ep2d_diff_3scan_trace_p2" to "dwi.nii.
↪gz"
mrconvert: [100%] compressing image "dwi.nii.gz"
```

converts the DWI data in the `DICOM/` folder to *Compressed NIfTI (.nii.gz)*, and exports the DW gradient table to *FSL format* if found in the DICOM headers, resulting in a pair of `bvecs` & `bvals` files.

Importing the DW gradient table

If the *image header* already contain the DW information, then no further action is required - the *MRtrix3* application will be able to find it and use it directly. If this is not the case (e.g. the image format does not support including it in the header), or the information contained is not correct, *MRtrix3* applications also allow the DW gradient table to be imported using the `-grad` option (for the *MRtrix format*) or the `-fslgrad` option (for the *FSL format*). Note that this will override the information found in the image headers if it was there. This can be used during conversion using `mrconvert`, or at the point of use. For example:

```
$ mrconvert dwi.nii -fslgrad dwi_bvecs dwi_bvals dwi.mif
```

will convert the `dwi.nii` from *NIfTI (.nii)* to *MRtrix image formats (.mih / .mif)*, embedding the DW gradient table information found in the `dwi_bvecs` & `dwi_bvals` files (in *FSL format*) directly into the output image header. As another example:

```
$ dwi2tensor DICOM/ -grad encoding.b tensor.nii
```

will process the DWI dataset found in the `DICOM/` folder (in *DICOM (folder or .dcm)* format), but *override* any DW gradient information in the DICOM data with the table stored in the *MRtrix format* file `encoding.b`.

Operations performed by *MRtrix3* when handling DW gradient tables

MRtrix3 applications will perform a number of sanity checks and modifications to the information in the DW gradient table, depending on the nature of the operation, and its original format.

When using the FSL format

In this format, the gradient vectors are provided relative to the image axes (as detailed in the FSL wiki). To convert them to the internal representation used in *MRtrix3* (and in the *MRtrix format* gradient table), these vectors need to be transformed into the real / scanner coordinate system. To do this requires knowledge of the DWI dataset these vectors correspond to, in particular the image transform. In essence, this consists of rotating the gradient vectors according to the rotation part of the transform (i.e. the top-left 3×3 part of the matrix). This will introduce differences between the components of the gradient vectors when stored in *MRtrix format* compared to the *FSL format*, particularly for images not acquired in a pure axial orientation (i.e. images where the rotation part of the image transform is identity). Indeed, as mentioned earlier, there is an additional confound related to the handed-ness of the coordinate system; see the FSL wiki for details.

Warning: **Never** perform a manual conversion between MRtrix and FSL gradient table formats using a text editor or basic shell script. This poses a risk of introducing an unwanted rotation / reflection of the gradient directions, with concomitant errors in later processing.

Note that in this operation, what matters is the transform as stored in the NIfTI headers (i.e. the `sform` / `qform`); the transform as reported by *mrinfo* can differ substantially from this (while still being consistent with the data), as the *MRtrix3* image loading backend will try to provide the image transform in a near-axial orientation (by inverting / exchanging columns of the transform, and adjusting the *Strides* to match - see *The image transfom* for details). To find out the actual transform that was stored in the NIfTI header, use *mrinfo* with the `-norealign` option.

When copying or converting

Applications like `mrconvert` that don't actually need to interpret the DW gradient table will simply pass the information through to the output unmodified. If the DW gradient table was found in the input image header, it will be written to the output image header if the image format supports it (i.e. if the output is in *MRtrix image formats (.mih / .mif)* - DICOM is not supported for writing). If the DW gradient table is imported via the `-grad` or `-fslgrad` option, it will also be passed through as-is (although including the modifications mentioned above in the *When using the FSL format* section). If the output image format does not allow storing the DW gradient table in the image header, the `-export_grad_mrtrix` or `-export_grad_fsl` options can be used to write it out to separate files, ready for use with third-party applications, or directly within *MRtrix3* if users prefer to keep their data organised in this way.

When using the information for processing

Applications that actually need to make use of the DW gradient information (e.g. `dwi2tensor`, `dwi2fod`, `dwiextract`, ...) will perform additional sanity checks and modifications of these data, beyond those described above. These include:

- verifying that the number of volumes in the DWI dataset matches the number of entries in the DW gradient table;
- where relevant, verifying that the DW gradient tables contains the data in a shell structure, by clustering similar *b*-values together (see *mrinfo*'s `-shell` and `-shellcount` options);
- normalising the gradient vectors to unit amplitude;
- scaling the *b*-values by the square of the gradient vector amplitude - see *b-value scaling* for details.

Note: *mrinfo* will also perform most of these checks. While there is no technical reason for it to interpret the DW gradient information, in practice it is generally helpful to view the information as it would be interpreted by other *MRtrix3* applications. If this is not desired, you can add the `-raw_dwgrad` option to *mrinfo* to disable these modifications when querying the DW gradient table.

b-value scaling

On MRI scanners that do not explicitly allow for multi-shell datasets, a common workaround is to set the scanning protocol according to the largest desired *b*-value, but use gradient vector directions that have *less than unit norm*. This results in diffusion sensitisation gradients with reduced strength, and hence images with lower *b*-values.

For example, if this was the desired gradient table:

```
0    0    0    0
1    0    0  700
1    0    0 2800
```

This could be achieved on some systems by supplying this custom diffusion vectors file, now nominally containing only $b = 0$ and $b = 2800$ s/mm^2:

```
0    0    0    0
0.5  0    0 2800
1    0    0 2800
```

By default, *MRtrix3* applications will **automatically** scale the *b*-values by the squared amplitude of the gradient vectors (so that the stored gradient table is equivalent to the first example), in order to more sensibly reflect the nature of the image data.

While this scaling allows such datasets to be processed seamlessly, it will introduce minor variations in the *b*-values for other datasets, due to minor rounding errors in the components of the direction vectors. These are benign, and have no consequence on the correct operation of *MRtrix3* applications, since the deviations are typically very small, and the strategy used to group *b*-values into shells is robust to such variations. If however this becomes a problem (e.g. for third-party applications), this feature can be disabled using the `-bvalue_scaling` option for those applications that support it.

CHAPTER 27

Orthonormal Spherical Harmonic basis

An important change between the old and new versions of MRtrix is a modification to the Spherical Harmonic (SH) basis functions. This change has important consequences in terms of data that were generated prior to the user changing to the new version, or any data that may be used interchangeably between the two versions.

Important: note that although it is possible to use and display FODs generated using MRtrix 0.2.x in the newer *MRtrix3* applications (and vice-versa), the FODs will *NOT* be correct. Moreover, it is very difficult to tell the difference on simple visual inspection - the FODs will still *look* reasonable, but will give incorrect results if used for tractography or in quantitative analyses. To ensure your images are correct, you should use the *shbasis* application included in *MRtrix3*, as described below.

The problem

For Spherical Deconvolution (SD) as implemented in MRtrix, processing is done in the Spherical Harmonic (SH) basis; this mathematical formulation provides a smooth representation of data distributed on the sphere. When we do SD, the resulting Fibre Orientation Distributions (FODs) are written to an image. These FOD images contain coefficients in this SH basis, that when interpreted correctly, produce the FOD butterflies we all know and love. If you've ever looked at the raw image volumes from an FOD image, you'll know that all but the first one are basically not interpretable.

Here's where it gets tricky. In all previous versions of MRtrix, there was a 'bug' in the SH basis functions. Mathematically, the basis was 'non-orthonormal'; you don't necessarily need to know what this means, just appreciate that the formulation of this mathematical basis was not optimal.

Now this 'bug' didn't actually cause any problems; the previous version of MRtrix was self-consistent in its handling of the issue throughout the code. It was annoying for any users transferring data between MRtrix and other packages though. For the release of the new *MRtrix3*, we have decided to correct the underlying error in the SH basis once and for all, as there are various mathematical operations that are greatly simplified when the basis is orthonormal. This does however introduce a problem for anyone that has done prior image processing using the old MRtrix 0.2 and wants to be able to use that data with *MRtrix3*: if you have image data that was generated using the *old* SH basis, but read it using MRtrix code that was compiled using the *new* SH basis, the data will *not be interpreted correctly*.

The solution

There is a solution, but it takes a bit of manual labour on your part. We have provided a new command called `shbasis`. This command will read your image data, and tell you which SH basis it thinks your image data are stored in (or if it's unable to make this decision).

Furthermore, it includes a command-line option for *changing* the SH basis of the underlying image data: `-convert`. The most important choice for this option is `-convert native`. This option identifies the SH basis that *MRtrix3* is compiled for (this is the new orthonormal basis by default); and if the image data is not currently stored in this basis, it *modifies the image data in-place* so that it conforms to the correct basis.

Any data that you generate after this update has occurred will automatically be produced in the new SH basis, and therefore will not need to be converted using `shbasis`. However if you are uncertain whether or not a particular image does or does not need to be converted, `shbasis` can always be used to verify whether or not the image data are in the correct SH basis; and if you provide the `-convert native` option despite the image data already being in the new SH basis, no modification of the image data will take place.

My recommendation is therefore as follows. When you commit to using the new version of MRtrix, you should go through *all* of your diffusion image data on *all* systems that you use, and run `shbasis -convert native` on all images that contain spherical harmonic data (only FOD images; raw DWIs / response functions / TDIs / etc. do not need to be converted).

Also: Remember I said that data previously generated will not be interpreted correctly by *MRtrix3* commands without the SH basis conversion? The same applies in the other direction. So if you load FOD images that have either been generated using *MRtrix*, or have been previously converted using `shbasis`, commands from the previous version of MRtrix (0.2) won't interpret them correctly. We hope that once we have feature completeness in *MRtrix3*, the old version will no longer be necessary, and therefore this will not be a problem.

Problematic data

In some circumstances, the `shbasis` command will give an error something like this:

```
shbasis [WARNING]: Cannot make unambiguous decision on SH basis of image csd.
mif (power ratio regressed to l=0 is 1.58446)
```

`shbasis` uses a data-driven approach to automatically determine the SH basis that the image data are currently stored in; however a number of issues can arise that lead to a breakdown of the numerical assumption that it is based on, and it can no longer make this decision.

If this occurs, but you are confident that your image data are in the old non-orthonormal basis and need to be converted to the new orthonormal basis, you can run: `shbasis <image> -convert force_oldtonew`. This will inform `shbasis` that even though it's unable to determine the current SH basis, you're confident that you do know it, and therefore it should perform the conversion anyway. It will give you a couple of loud warnings just to make sure you appreciate the danger in what you're doing, so you should only ever use this setting for problematic data; for the vast majority of conversions, `-convert native` is much better.

CHAPTER 28

Dixels and Fixels

So internally we have created a couple of new terms that we find invaluable when discussing diffusion MRI processing methods and statistics. We'd like to share these with our user base in the hope that others will gain advantages from using the same terminology, and also so that we all know what everyone else is talking about! Anyone using *MRtrix3* to develop their own software may also see these terms scattered throughout the library code, so will need to know what they represent.

All MRtrix users should be familiar with the terms 'pixel' and 'voxel'; these correspond to 'picture element' and 'volume element' respectively. However in Diffusion MRI we also deal with orientation information within each image volume element, so we wanted terminology to allow us to convey the types of discrete elements that we deal with on a daily basis.

We have settled on the following terms; note that this may conflict with presentations that we have done in the past, but this is now what we are sticking to.

'Dixel': *Directional Element*

Imagine a single image voxel, the data for which is in fact a function on the sphere (i.e. varies with orientation). We now take samples of that function along a set of pre-defined directions on the unit sphere. Each of those samples is referred to as a *dixel*: a directional element within a specific voxel. Each dixel is described by the voxel in which it resides, the direction along which the relevant spherical function was sampled, and the intensity of the function in that direction.

Importantly, it is the *combination* of the voxel location and sampling direction that describe the dixel. If a different direction were used to sample the spherical function, that would be a different dixel with a different value; likewise, if the spherical function in an adjacent voxel were sampled along the same direction, that would also be a different dixel with a different value. Each dixel is a unique sample of a spatially-varying spherical function.

Most commonly, the term dixel is used to refer to the situation where a set of directions on the unit sphere has been used to sample a Fibre Orientation Distribution (FOD) that is otherwise continuous as expressed in the Spherical Harmonic basis. However, by the definition of the term, 'dixel' could also be used to describe a single voxel within a particular image volume in a HARDI experiment; if the HARDI signal in a single voxel is considered to be discrete samples of the orientation dependence of the diffusion signal in that voxel, then each of those samples could be labelled a dixel.

Although we find this term useful in our internal discussions, and the original Apparent Fibre Density (AFD) statistical method was based around this concept, it is not a term that we expect to be adopted by others, as its applicability for the end user is limited.

'Fixel': *Fibre bundle element*

It will be more common to hear use of the term *fixel*; this refers to a specific fibre bundle within a specific voxel. Each fixel is therefore parametrized by the voxel in which it resides, the estimated mean direction of the underlying fibres attributed to that bundle, a fibre density (or partial volume fraction), and potentially other metrics.

At this point it is important to distinguish between 'dixel' and 'fixel'. A 'dixel' is typically assumed to represent a sample of a spherical function along some pre-determined direction, where that direction belongs to some basis set of equally-distributed unit directions that has been used to sample an otherwise continuous spherical function. 'Fixel', on the other hand, is used to describe a set of fibres within a voxel that are sufficiently similar in orientation that they are indistinguishable from one another, and therefore form a fibre 'bundle' within that voxel.

In reality, fixels have been used in the field of Diffusion MRI for a long time: multi-tensor fitting, ball-and-sticks, any diffusion model that is capable of fitting multiple anisotropic elements to each image voxel, can be considered as providing fixels. We've just resorted to long-winded explanations to describe what we're on about. With MRtrix we are historically more accustomed to dealing with FODs that are continuous functions on the sphere, and are utilised as such during processing; however, if the FOD is *segmented* in any way (either through peak-finding, the segmentation approach as described in the appendices of the SIFT NeuroImage paper, or more advanced methods), each discrete feature of a particular FOD can be labelled a fixel, as each represents a set of fibres within that voxel that form a coherent bundle in orientation space.

The term 'fixel' has now appeared in the literature with the publication of our new statistical method, Connectivity-based Fixel Enhancement, which allows for the inference of group differences not just at the voxel level, but the *fixel* level; that is, if only one fibre bundle within a crossing-fibre voxel is affected in a cohort, we hope to both identify the bundle affected, and quantify the group effect that is specific to that bundle.

CHAPTER 29

Motivation for `afdconnectivity`

Due to the interest in the *afdconnectivity* command, I thought I'd explain the reasoning behind the approach, the rationale behind the improvements made in commit 40ccdb62, and the argument for why we recommend the use of *Spherical-deconvolution Informed Filtering of Tractograms (SIFT)* as an alternative if possible.

The `afdconnectivity` command was originally written as a 'hack' for a colleague who wanted to obtain quantitative measures of 'connectivity' in the absence of EPI distortion correction. Without EPI distortion correction *Anatomically-Constrained Tractography (ACT)* cannot be applied, and consequently streamlines may terminate within white matter. Streamline count (as a measure of connectivity) between two grey matter regions will therefore not include those streamlines that terminate in white matter (and therefore the estimated connectivity may not be accurate).

The afdconnectivity command attempts to get around this issue by estimating a measure of 'connectivity' as follows:

- The integral of a discrete lobe of an FOD (*fixel*) is proportional to the volume of the MR-visible tissue (intracellular at high *b*-value) aligned in that direction.
- By taking a set of streamlines corresponding to a pathway of interest, and summing the integrals of all FOD lobes traversed by the bundle, you obtain an estimate of the total fibre volume of the pathway of interest.
- If you then divide by the length of the bundle (taken as the mean streamline length), you get an estimate of the cross-sectional area of the bundle, which is a measure of 'connectivity' independent of fibre length.

The major problem with this approach is the assumption that *all* of the fibre volume in each fixel traversed by the streamlines of interest belong to the bundle of interest; clearly not the case in various circumstances. The changes I have made to `afdconnectivity` are aimed at improving the behaviour in the presence of partial volume and erroneous streamlines.

The default behaviour is as before: determine a fixel mask using some bundle of streamlines, sum the apparent fibre density (a volume) of the fixels within the mask, and divide by mean streamline length (to get an estimate of cross-sectional area of the pathway).

Now, you can optionally provide a whole-brain fibre-tracking data set using the `-wbft` option (your bundle .tck file should then be a subset of this tractogram). In this case, the program determines the total streamlines density attributed to each fixel, and for those fixels traversed by the streamlines of interest, some fraction of the fibre volume of that fixel is contributed to the result. This fraction is determined for each fixel by the ratio of streamlines density from the bundle of interest, to the total streamlines density from the tractogram. The fibre volume of each fixel is therefore divided 'fairly' between the bundle of interest and the rest of the tractogram.

Although this may be an improvement in many circumstances, it's still not our recommended method. Effectively what's happening in this scenario is that for each streamline, a fibre volume is determined, based on its 'fair share' of each fixel it traverses. However this means that the effective cross-sectional area of that streamline is *allowed to vary drastically along its length*; this is clearly not physically realistic. Furthermore, due to the relative over- or under-reconstruction of different pathways in whole-brain fibre-tracking, there's no guarantee that this proportional 'sharing' of fibre volume between streamlines is biologically accurate.

Now consider the alternative: filtering a tractogram using *Spherical-deconvolution Informed Filtering of Tractograms (SIFT)*, then selecting a subset of the remaining streamlines corresponding to your pathway of interest. By the model underlying SIFT, each streamline represents a constant cross-sectional area of fibres; so the *streamline count* becomes your estimate of bundle cross-sectional area and therefore 'connectivity' (with the SIFT proportionality coefficient providing the conversion between streamline count and AFD if you so choose).

This argument also holds if you are looking to use the image output from `afdconnectivity`, which provides the estimated fibre volume of the pathway of interest within each voxel. I have already stated why this is a poor interpretation with the default `afdconnectivity` behaviour; it's improved with use of the `-wbft` option, but is noisy in regions where fixels are traversed by very few streamlines, and still may not share the fibre volume of each fixel appropriately. Again, SIFT provides the better alternative: an equivalent map can be produced by selecting your streamlines of interest post-SIFT, and running `tckmap -precise` (sums streamline lengths within each voxel rather than counting streamlines). Remember: a product of cross-sectional area and length gives a volume!

This is also an important message for interpretation of AFD results, both in this context and others. FOD amplitude (in any guise) is *in no way* a measure of "tissue integrity", no matter how many quotation marks you use; it's a measure of *density*. This is the reasoning behind the modulation step in AFD, and is the entire premise behind the SIFT method.

Anyways, rant over. We are considering writing a technical note that will discuss this issue, so we are trusting the *MRtrix3* beta user base not to do anything scientifically unethical with this information / command until we can create the relevant article for citation.

CHAPTER 30

Response function estimation

A compulsory step in spherical deconvolution is deriving the 'response function (RF)', which is used as the kernel during the deconvolution step. For the white matter, this is the signal expected for a voxel containing a single, coherently-oriented fibre bundle. While some groups prefer to define this function using some *ad-hoc* template function (e.g. a diffusion tensor with empirical diffusivities), the MRtrix contributors are in preference of deriving this function directly from the image data, typically by averaging the diffusion signal from a set of empirically-determined 'single-fibre (SF)' voxels.

The process of estimating this function from the data is however non-trivial; there is no single unambiguous way in which this should be done. Earlier in the beta version of MRtrix3, we provided a command `dwi2response` that advertised automated determination of the response function, based on a published method with a few additional enhancements. Unfortunately user testing showed that this algorithm would not produce the desired result in a number of circumstances, and the available command-line options for altering its behaviour were not intuitive.

As a result, we are now instead providing *dwi2response* as a *script*. This was done for a few reasons. Firstly, it means that we can provide multiple different mechanisms / algorithms for response function estimation, all accessible within the one script, allowing users to experiment with different approaches. Secondly, because these Python scripts are more accessible to most users than C++ code, the algorithms themselves can be modified, or some may even choose to try devising their own heiristics for response function estimation. Thirdly, it reinforces the fact that there is unfortunately *not* a black-box, one-size-fits-all solution to this problem.

Here I will discuss some of the technical aspects of response function estimation, and describe the mechanisms by which the currently provided algorithms work. If however you are not interested in the nitty-gritty of this process, feel free to scroll to the bottom of the page.

Necessary steps

Looking at the process of response function estimation in full detail, there are four crucial steps. For each of these, I will also briefly mention the typical process used.

1. Select those image voxels that are to be used when determining the response function - the 'single-fibre mask'. *Typical*: Varies.

2. Estimate the direction of the underlying fibres in each voxel. *Typical*: Often the diffusion tensor fit is still used for this purpose; though CSD itself can also be used as long as an initial response function estimate is available.
3. Rotate the signal measured in each single-fibre voxel in such a way that the estimated fibre direction coincides with the z-axis. *Typical*: This may be done by rotating the diffusion gradient table according to the estimated fibre direction; or if the diffusion signal is converted to spherical harmonics, then a spherical convolution can be used.
4. Combine these signals to produce a single response function. *Typical*: The `m=0` terms of the spherical harmonic series (which are rotationally symmetric about the z-axis) are simply averaged across single-fibre voxels.

Of these steps, the first is the one that has caused the greatest difficulty, and is also the principle mechanism where the provided response function estimation algorithms vary. It will therefore be the primary focus of this document, though note that the other aspects of this process may also change with ongoing development.

`dwi2response` algorithms

`fa`

In the previous version of MRtrix ('0.2'), the following heuristic was suggested in the documentation for deriving the response function:

- Erode a brain mask by a few voxels, to omit any voxels near the edge of the brain;
- Select those voxels within the mask that have a Fractional Anisotropy (FA) of 0.7 or greater;
- The `estimate_response` command would then be used to generate a response function, which would internally perform diffusion tensor estimation to get the fibre directions as well as the gradient reorientation.

Rather than this series of commands, `dwi2response` now provides a similar heuristic in-built as the `fa` algorithm. The primary difference is that by default, it will instead select the 300 voxels with the highest FA (though this can be modified at the command-line).

This algorithm is provided partly for nostalgic purposes, but it also highlights the range of possibilities for single-fbre voxel selection. One of the problems associated with this approach (over and above the feeling of uncleanliness from using the tensor model) is that in white matter regions close to CSF, Gibbs ringing can make the signal in *b=0* images erroneously low, which causes an artificial increase in FA, and therefore such voxels get included in the single-fibre mask.

`manual`

This algorithm is provided for cases where none of the available algorithms give adequate results, for deriving multi-shell multi-tissue response functions in cases where the voxel mask for each tissue must be defined manually, or for anyone who may find it useful if trying to devise their own mechanism for response function estimation. It requires manual definition of both the single-fibre voxel mask (or just a voxel mask for isotropic tissues); the fibre directions can also be provided manually if necessary (otherwise a tensor fit will be used).

`msmt_5tt`

This algorithm is intended for deriving multi-shell, multi-tissue response functions that are compatible with the new Multi-Shell Multi-Tissue (MSMT) CSD algorithm. The response function estimation algorithm is identical to that described in the manuscript: As long as EPI inhomogeneity field correction has been performed, and a tissue-segmented anatomical image (prepared in the 5TT format for *ACT*) is provided with good prior rigid-body alignment to the diffusion images, then these high-resolution tissue segmentations can be used to identify single-tissue voxels in the

diffusion images. This algorithm is hard-wired to provide response functions for the most typical use case for MSMT CSD: An isotropic grey matter response, an anisotropic white matter response, and an isotropic CSF response; the output response functions are provided in the format expected by the *dwi2fod* command. Those wishing to experiment with different multi-tissue response function configurations will need to use the `manual` algorithm (which will provide a multi-shell response function if the input DWI contains such data).

For reference, this algorithm operates as follows:

1. Resample the 5TT segmented image to diffusion image space.
2. For each of the three tissues (WM, GM, CSF), select those voxels that obey the following criteria:

- The tissue partial volume fraction must be at least 0.95.
- For GM and CSF, the FA must be no larger than 0.2.

3. For WM, use the mask derived from step 2 as the initialisation to the `tournier` algorithm, to select single-fibre voxels.
4. Derive a multi-shell response for each tissue for each of these three tissues. For GM and CSF, use *lmax=0* for all shells.

`tax`

This algorithm is a fairly accurate reimplementation of the approach proposed by Tax et al.. The operation of the algorithm can be summarized as follows:

1. Initialise the response function using a relatively 'fat' profile, and the single-fibre mask using all brain voxels.
2. Perform CSD in all single-fibre voxels.
3. Exclude from the single-fibre voxel mask those voxels where the resulting FOD detects more than one discrete fibre population, e.g. using the ratio of the amplitudes of the first and second tallest peaks.
4. Re-calculate the response function using the updated single-fibre voxel mask.
5. Return to step 2, repeating until some termination criterion is achieved.

The following are the differences between the implementation in `dwi2response` and this manuscript:

- Deriving the initial response function. In the manuscript, this is done using a tensor model with a low FA. I wasn't fussed on this approach myself, in part because it's difficult to get the correct intensity sscaling. Instead, the script examines the mean and standard deviation of the raw DWI volumes, and derives an initial *lmax=4* response function based on these.
- The mechanism used to identify the peaks of the FOD. In `dwi2response`, the FOD segmentation algorithm described in the SIFT paper (Appendix 2) is used to locate the FOD peaks. The alternative is to use the *sh2peaks* command, which uses a Newton search from 60 pre-defined directions to locate these peaks. In my experience, the latter is slower, and may fail to identify some FOD peaks because the seeding directions are not sufficiently dense.

For the sake of completeness, the following are further modifications that were made to the algorithm as part of the earlier `dwi2response` *binary*, but have been removed from the script as it is now provided:

- Rather than using the ratio of amplitudes between the tallest and second-tallest peaks, this command instead looked at the ratio of the AFD of the largest FOD lobe, and the sum of the AFD of all other (positive) lobes in the voxel. Although this in some way makes more sense from a physical perspective (comparing the volume occupied by the primary fibre bundle to the volume of 'everything else'), it's possible that due to the noisy nature of the FODs at small amplitudes, this may have only introduced variance into the single-fibre voxel identification process. Therefore the script has reverted to the original & simpler peak amplitude ratio calculation.

- A second, more stringent pass of SF voxel exclusion was performed, which introduced two more criteria that single-fibre voxels had to satisfy:
- Dispersion: A measure of dispersion of an FOD lobe can be derived as the ratio between the integral (fibre volume) and the peak amplitude. As fibre dispersion increases, the FOD peak amplitude decreases, but the fibre volume is unaffected; therefore this ratio increases. The goal here was to explicitly exclude voxels from the single-fibre mask if significant orientation dispersion was observed; this can be taken into account somewhat by using the FOD peak amplitudes (as orientation dispersion will decrease the amplitude of the tallest peak), but from my initial experimentation I wanted something more stringent. However as before, given the difficulties that many users experienced with the `dwi2response` command, this algorithm in the new script errs on the side of simplicity, so this test is not performed.
- Integral: By testing only the ratio of the tallest to second-tallest FOD peak amplitude, the absolute value of the peak amplitude is effectively ignored. This may or may not be considered problematic, for either small or large FOD amplitudes. If the peak amplitude / AFD is smaller than that of other voxels, it's possible that this voxel experiences partial volume with CSF: this may satisfy the peak ratio requirement, but using such a voxel is not ideal in response function estimation as its noise level will be higher and the Rician noise bias will be different. Conversely, both in certain regions of the brain and in some pathologies, some voxels can appear where the AFD is much higher due to T2 shine-through; it may seem appealing to use such voxels in response function estimation as the SNR is higher, but as for the low-signal case, the Rician noise bias will be different to that in the rest of the brain. The previous `dwi2response` binary attempted to exclude such voxels by looking at the mean and standard deviation of AFD within the single-fibre mask, and excluding voxels above or below a certain threshold. As before, while this heuristic may or may not seem appropriate depending on your point of view, it has been excluded from the new `dwi2response` script to keep things as simple as possible.

tournier

Independently and in parallel, Donald also developed a newer method for response function estimation based on CSD itself; it was used in this manuscript. It bears some resemblance to the `tax` algorithm, but relies on a threshold on the number of voxels in the single-fibre mask, rather than the ratio between tallest and second-tallest peaks. The operation is as follows:

1. Define an initial response function that is as sharp as possible (ideally a flat disk, but will be fatter due to spherical harmonic truncation). Limit this initial function to *lmax=4*, as this makes the FODs less noisy in the first iteration.
2. Run CSD for all voxels within the mask (initially, this is the whole brain).
3. Select the 300 'best' single-fibre voxels. This is not precisely the ratio between tallest and second-tallest peaks; instead, the following equation is used, which also biases toward selection of voxels where the tallest FOD peak is larger: `sqrt(|peak1|) * (1 - |peak2| / |peak1|)^2`. Use these voxels to generate a new response fuction.
4. Test to see if the selection of single-fibre voxels has changed; if not, the script is completed.
5. Derive a mask of voxels to test in the next iteration. This is the top 3,000 voxels according to the equation above, and dilated by one voxel.
6. Go back to step 2.

This approach appears to be giving reasonable results for the datasets on which it has been tested. However if you are involved in the processing of non-human brain images in particular, you may need to experiment with the number of single-fibre voxels as the white matter is typically smaller.

TL;DR

If this document appears far too long for your liking, or you're not particularly interested in the details and just want to know what option to use so that you can continue with your processing, the following are our 'cautious' recommendations. However we emphasize that this script may not work flawlessly for all data; if it did, we wouldn't be providing a script with so many options! Furthermore, these recommendations may change over time, so check in every now and then, and make sure to sign up to the new community forum.

- If you're processing single-shell data, the `tournier` algorithm appears to be fairly robust.
- If you're processing multi-shell data, and are able to perform EPI inhomogeneity distortion correction, `msmt_5tt` is currently the only fully-automated method for getting multi-shell multi-tissue response functions.

CHAPTER 31

List of MRtrix3 commands

5tt2gmwmi

Synopsis

```
5tt2gmwmi [ options ]  5tt_in mask_out
```

- *5tt_in*: the input 5TT segmented anatomical image
- *mask_out*: the output mask image

Description

Generate a mask image appropriate for seeding streamlines on the grey matter - white matter interface

Options

- **-mask_in image** Filter an input mask image according to those voxels that lie upon the grey matter - white matter boundary. If no input mask is provided, the output will be a whole-brain mask image calculated using the anatomical image only.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. Anatomically-constrained tractography:Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 2012, 62, 1924-1938

Author: Robert E. Smith (robert.smith@florey.edu.au)

5tt2vis

Synopsis

```
5tt2vis [ options ]  input output
```

- *input*: the input 4D tissue-segmented image
- *output*: the output 3D image for visualisation

Description

generate an image for visualisation purposes from an ACT 5TT segmented anatomical image

Options

- **-bg value** image intensity of background (default: 0)
- **-cgm value** image intensity of cortical grey matter (default: 0.5)
- **-sgm value** image intensity of sub-cortical grey matter (default: 0.75)
- **-wm value** image intensity of white matter (default: 1)
- **-csf value** image intensity of CSF (default: 0.15)
- **-path value** image intensity of pathological tissue (default: 2)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

5ttedit

Synopsis

```
5ttedit [ options ]  input output
```

- *input*: the 5TT image to be modified
- *output*: the output modified 5TT image

Description

manually set the partial volume fractions in an ACT five-tissue-type (5TT) image using mask images

Options

- **-cgm image** provide a mask of voxels that should be set to cortical grey matter
- **-sgm image** provide a mask of voxels that should be set to sub-cortical grey matter
- **-wm image** provide a mask of voxels that should be set to white matter
- **-csf image** provide a mask of voxels that should be set to CSF
- **-path image** provide a mask of voxels that should be set to pathological tissue

- **-none image** provide a mask of voxels that should be cleared (i.e. are non-brain); note that this will supersede all other provided masks

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

afdconnectivity

Synopsis

```
afdconnectivity [ options ]  image tracks
```

- *image*: the input FOD image.
- *tracks*: the input track file defining the bundle of interest.

Description

obtain an estimate of fibre connectivity between two regions using AFD and streamlines tractography

This estimate is obtained by determining a fibre volume (AFD) occupied by the pathway of interest, and dividing by the streamline length.

If only the streamlines belonging to the pathway of interest are provided, then ALL of the fibre volume within each fixel selected will contribute to the result. If the -wbft option is used to provide whole-brain fibre-tracking (of which the pathway of interest should contain a subset), only the fraction of the fibre volume in each fixel estimated to belong to the pathway of interest will contribute to the result.

Use -quiet to suppress progress messages and output fibre connectivity value only.

For valid comparisons of AFD connectivity across scans, images MUST be intensity normalised and bias field corrected, and a common response function for all subjects must be used.

Note that the sum of the AFD is normalised by streamline length to account for subject differences in fibre bundle length. This normalisation results in a measure that is more related to the cross-sectional volume of the tract (and therefore 'connectivity'). Note that SIFT-ed tract count is a superior measure because it is unaffected by tangential yet unrelated fibres. However, AFD connectivity may be used as a substitute when Anatomically Constrained Tractography is not possible due to uncorrectable EPI distortions, and SIFT may therefore not be as effective.

Options

- **-wbft tracks** provide a whole-brain fibre-tracking data set (of which the input track file should be a subset), to improve the estimate of fibre bundle volume in the presence of partial volume
- **-afd_map image** output a 3D image containing the AFD estimated for each voxel.
- **-all_fixels** if whole-brain fibre-tracking is NOT provided, then if multiple fixels within a voxel are traversed by the pathway of interest, by default the fixel with the greatest streamlines density is selected to contribute to the AFD in that voxel. If this option is provided, then ALL fixels with non-zero streamlines density will contribute to the result, even if multiple fixels per voxel are selected.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au) and Robert E. Smith (robert.smith@florey.edu.au)

amp2sh

Synopsis

```
amp2sh [ options ]  amp SH
```

- *amp*: the input amplitude image.
- *SH*: the output spherical harmonics coefficients image.

Description

convert a set of amplitudes (defined along a set of corresponding directions) to their spherical harmonic representation. The spherical harmonic decomposition is calculated by least-squares linear fitting.

The directions can be defined either as a DW gradient scheme (for example to compute the SH representation of the DW signal) or a set of [az el] pairs as output by the dirgen command. The DW gradient scheme or direction set can be supplied within the input image header or using the -gradient or -directions option. Note that if a direction set and DW gradient scheme can be found, the direction set will be used by default.

Note that this program makes use of implied symmetries in the diffusion profile. First, the fact the signal attenuation profile is real implies that it has conjugate symmetry, i.e. Y(l,-m) = Y(l,m)* (where * denotes the complex conjugate). Second, the diffusion profile should be antipodally symmetric (i.e. S(x) = S(-x)), implying that all odd l components should be zero. Therefore, this program only computes the even elements.

Note that the spherical harmonics equations used here differ slightly from those conventionally used, in that the (-1)^m factor has been omitted. This should be taken into account in all subsequent calculations.

The spherical harmonic coefficients are stored as follows. First, since the signal attenuation profile is real, it has conjugate symmetry, i.e. Y(l,-m) = Y(l,m)* (where * denotes the complex conjugate). Second, the diffusion profile should be antipodally symmetric (i.e. S(x) = S(-x)), implying that all odd l components should be zero. Therefore, only the even elements are computed. Note that the spherical harmonics equations used here differ slightly from those conventionally used, in that the (-1)^m factor has been omitted. This should be taken into account in all subsequent calculations. Each volume in the output image corresponds to a different spherical harmonic component. Each volume will correspond to the following: volume 0: l = 0, m = 0 ; volume 1: l = 2, m = -2 (imaginary part of m=2 SH) ; volume 2: l = 2, m = -1 (imaginary part of m=1 SH) ; volume 3: l = 2, m = 0 ; volume 4: l = 2, m = 1 (real part of m=1 SH) ; volume 5: l = 2, m = 2 (real part of m=2 SH) ; etc...

Options

- **-lmax order** set the maximum harmonic order for the output series. By default, the program will use the highest possible lmax given the number of diffusion-weighted images, up to a maximum of 8.
- **-normalise** normalise the DW signal to the b=0 image
- **-directions file** the directions corresponding to the input amplitude image used to sample AFD. By default this option is not required providing the direction set is supplied in the amplitude image. This should be supplied as a list of directions [az el], as generated using the dirgen command
- **-rician noise** correct for Rician noise induced bias, using noise map supplied

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with

each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.

- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

DW Shell selection options

- **-shell list** specify one or more diffusion-weighted gradient shells to use during processing, as a comma-separated list of the desired approximate b-values. Note that some commands are incompatible with multiple shells, and will throw an error if more than one b-value is provided.

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

connectome2tck

Synopsis

```
connectome2tck [ options ]  tracks_in assignments_in prefix_out
```

- *tracks_in*: the input track file
- *assignments_in*: text file containing the node assignments for each streamline
- *prefix_out*: the output file / prefix

Description

extract streamlines from a tractogram based on their assignment to parcellated nodes

Options

Options for determining the content / format of output files

- **-nodes list** only select tracks that involve a set of nodes of interest (provide as a comma-separated list of integers)
- **-exclusive** only select tracks that exclusively connect nodes from within the list of nodes of interest
- **-files option** select how the resulting streamlines will be grouped in output files. Options are: per_edge, per_node, single (default: per_edge)
- **-exemplars image** generate a mean connection exemplar per edge, rather than keeping all streamlines (the parcellation node image must be provided in order to constrain the exemplar endpoints)
- **-keep_unassigned** by default, the program discards those streamlines that are not successfully assigned to a node. Set this option to generate corresponding outputs containing these streamlines (labelled as node index 0)
- **-keep_self** by default, the program will not output streamlines that connect to the same node at both ends. Set this option to instead keep these self-connections.

Options for importing / exporting streamline weights

- **-tck_weights_in path** specify a text scalar file containing the streamline weights
- **-prefix_tck_weights_out prefix** provide a prefix for outputting a text file corresponding to each output file, each containing only the streamline weights relevant for that track file

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)

- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

dcmedit

Synopsis

```
dcmedit [ options ]  file
```

- *file*: the DICOM file to be edited.

Description

Edit DICOM file in-place. Note that this simply replaces the existing values without modifying the DICOM structure in any way. Replacement text will be truncated if it is too long to fit inside the existing tag.

WARNING: this command will modify existing data! It is recommended to run this command on a copy of the original data set to avoid loss of data.

Options

- **-anonymise** remove any identifiable information, by replacing the following tags: - any tag with Value Representation PN will be replaced with 'anonymous' - tag (0010,0030) PatientBirthDate will be replaced with an empty stringWARNING: there is no guarantee that this command will remove all identiable information, since such information may be contained in any number of private vendor-specific tags. You will need to double-check the results independently if you need to ensure anonymity.
- **-id text** replace all ID tags with string supplied. This consists of tags (0010, 0020) PatientID and (0010, 1000) OtherPatientIDs
- **-tag group element newvalue** replace specific tag.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.

- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dcminfo

Synopsis

```
dcminfo [ options ]  file
```

- *file*: the DICOM file to be scanned.

Description

output DICOM fields in human-readable format.

Options

- **-all** print all DICOM fields.
- **-csa** print all Siemens CSA fields
- **-tag group element** print field specified by the group & element tags supplied. Tags should be supplied as Hexadecimal (i.e. as they appear in the -all listing).

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dirflip

Synopsis

```
dirflip [ options ]  in out
```

- *in*: the input files for the directions.
- *out*: the output files for the directions.

Description

optimise the polarity of the directions in a scheme with respect to a unipolar electrostatic repulsion model, by inversion of individual directions. The orientations themselves are not affected, only their polarity. This is necessary to ensure near-optimal distribution of DW directions for eddy-current correction.

Options

- **-permutations num** number of permutations to try.
- **-cartesian** Output the directions in Cartesian coordinates [x y z] instead of [az el].

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)

- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dirgen

Synopsis

```
dirgen [ options ]  ndir dirs
```

- *ndir*: the number of directions to generate.
- *dirs*: the text file to write the directions to, as [az el] pairs.

Description

generate a set of uniformly distributed directions using a bipolar electrostatic repulsion model.

Options

- **-power exp** specify exponent to use for repulsion power law (default: 2). This must be a power of 2 (i.e. 2, 4, 8, 16, ...).
- **-niter num** specify the maximum number of iterations to perform (default: 10000).
- **-unipolar** optimise assuming a unipolar electrostatic repulsion model rather than the bipolar model normally assumed in DWI
- **-cartesian** Output the directions in Cartesian coordinates [x y z] instead of [az el].

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Jones, D.; Horsfield, M. & Simmons, A. Optimal strategies for measuring diffusion in anisotropic systems by magnetic resonance imaging. Magnetic Resonance in Medicine, 1999, 42: 515-525

Papadakis, N. G.; Murrills, C. D.; Hall, L. D.; Huang, C. L.-H. & Adrian Carpenter, T. Minimal gradient encoding for robust estimation of diffusion anisotropy. Magnetic Resonance Imaging, 2000, 18: 671-679

Author: J-Donald Tournier (jdtournier@gmail.com)

dirmerge

Synopsis

```
dirmerge [ options ]  subsets bvalue files [ bvalue files ... ] out
```

- *subsets*: the number of subsets (phase-encode directions) per b-value
- *bvalue files*: the b-value and sets of corresponding files, in order
- *out*: the output directions file, with each row listing the X Y Z gradient directions, the b-value, and an index representing the phase encode direction

Description

splice or merge sets of directions over multiple shells into a single set, in such a way as to maintain near-optimality upon truncation.

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.

- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dirorder

Synopsis

```
dirorder [ options ]  input output
```

- *input*: the input directions file
- *output*: the output directions file

Description

reorder a set of directions to ensure near-uniformity upon truncation - i.e. if the scan is terminated early, the acquired directions are still close to optimal

Options

- **-cartesian** Output the directions in Cartesian coordinates [x y z] instead of [az el].

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dirsplit

Synopsis

```
dirsplit [ options ]  dirs out [ out ... ]
```

- *dirs*: the text file containing the directions.
- *out*: the output partitioned directions

Description

split a set of evenly distributed directions (as generated by dirgen) into approximately uniformly distributed subsets.

Options

- **-permutations num** number of permutations to try
- **-cartesian** Output the directions in Cartesian coordinates [x y z] instead of [az el].

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.

- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dirstat

Synopsis

```
dirstat [ options ]  dirs
```

- *dirs*: the text file containing the directions.

Description

report statistics on a direction set

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

MRtrix is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

For more details, see www.mrtrix.org

dwi2adc

Synopsis

```
dwi2adc [ options ]  input output
```

- *input*: the input image.
- *output*: the output image.

Description

convert mean dwi (trace-weighted) images to mean adc maps

Options

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dwi2fod

Synopsis

```
dwi2fod [ options ]  algorithm dwi response odf [ response odf ... ]
```

- *algorithm*: the algorithm to use for FOD estimation. (options are: csd,msmt_csd)
- *dwi*: the input diffusion-weighted image
- *response odf*: pairs of input tissue response and output ODF images

Description

estimate fibre orientation distributions from diffusion data using spherical deconvolution.

The spherical harmonic coefficients are stored as follows. First, since the signal attenuation profile is real, it has conjugate symmetry, i.e. Y(l,-m) = Y(l,m)* (where * denotes the complex conjugate). Second, the diffusion profile should be antipodally symmetric (i.e. S(x) = S(-x)), implying that all odd l components should be zero. Therefore, only the even elements are computed. Note that the spherical harmonics equations used here differ slightly from those conventionally used, in that the (-1)^m factor has been omitted. This should be taken into account in all subsequent calculations. Each volume in the output image corresponds to a different spherical harmonic component. Each volume will correspond to the following: volume 0: l = 0, m = 0 ; volume 1: l = 2, m = -2 (imaginary part of m=2 SH) ; volume 2: l = 2, m = -1 (imaginary part of m=1 SH) ; volume 3: l = 2, m = 0 ; volume 4: l = 2, m = 1 (real part of m=1 SH) ; volume 5: l = 2, m = 2 (real part of m=2 SH) ; etc...

Options

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.

- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

DW Shell selection options

- **-shell list** specify one or more diffusion-weighted gradient shells to use during processing, as a comma-separated list of the desired approximate b-values. Note that some commands are incompatible with multiple shells, and will throw an error if more than one b-value is provided.

Options common to more than one algorithm

- **-directions file** specify the directions over which to apply the non-negativity constraint (by default, the built-in 300 direction set is used). These should be supplied as a text file containing [az el] pairs for the directions.
- **-lmax order** the maximum spherical harmonic order for the output FOD(s).For algorithms with multiple outputs, this should be provided as a comma-separated list of integers, one for each output image; for single-output algorithms, only a single integer should be provided. If omitted, the command will use the highest possible lmax given the diffusion gradient table, up to a maximum of 8.
- **-mask image** only perform computation within the specified binary brain mask image.

Options for the Constrained Spherical Deconvolution algorithm

- **-filter spec** the linear frequency filtering parameters used for the initial linear spherical deconvolution step (default = [1 1 1 0 0]). These should be supplied as a text file containing the filtering coefficients for each even harmonic order.
- **-neg_lambda value** the regularisation parameter lambda that controls the strength of the non-negativity constraint (default = 1).
- **-norm_lambda value** the regularisation parameter lambda that controls the strength of the constraint on the norm of the solution (default = 1).
- **-threshold value** the threshold below which the amplitude of the FOD is assumed to be zero, expressed as an absolute amplitude (default = 0).
- **-niter number** the maximum number of iterations to perform for each voxel (default = 50). Use '-niter 0' for a linear unconstrained spherical deconvolution.

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.

- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- If using csd algorithm:Tournier, J.-D.; Calamante, F. & Connelly, A. Robust determination of the fibre orientation distribution in diffusion MRI: Non-negativity constrained super-resolved spherical deconvolution. NeuroImage, 2007, 35, 1459-1472
- If using msmt_csd algorithm:Jeurissen, B; Tournier, J-D; Dhollander, T; Connelly, A & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data NeuroImage, 2014, 103, 411-426

Tournier, J.-D.; Calamante, F., Gadian, D.G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution.NeuroImage, 2004, 23, 1176-1185

Author: J-Donald Tournier (jdtournier@gmail.com) and Ben Jeurissen (ben.jeurissen@uantwerpen.be)

dwi2mask

Synopsis

```
dwi2mask [ options ]  image image
```

- *image*: the input DWI image containing volumes that are both diffusion weighted and b=0
- *image*: the output whole brain mask image

Description

Generates a whole brain mask from a DWI image. All diffusion weighted and b=0 volumes are used to obtain a mask that includes both brain tissue and CSF. In a second step peninsula-like extensions, where the peninsula itself is wider than the bridge connecting it to the mask, are removed. This may help removing artefacts and non-brain parts, e.g. eyes, from the mask.

Options

- **-clean_scale value** the maximum scale used to cut bridges. A certain maximum scale cuts bridges up to a width (in voxels) of 2x the provided scale. Setting this to 0 disables the mask cleaning step. (Default: 2)

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Dhollander T, Raffelt D, Connelly A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5.

Author: David Raffelt (david.raffelt@florey.edu.au), Thijs Dhollander (thijs.dhollander@gmail.com) and Ben Jeurissen (ben.jeurissen@uantwerpen.be)

dwi2noise

Synopsis

```
dwi2noise [ options ]  dwi noise
```

- *dwi*: the input diffusion-weighted image.
- *noise*: the output noise map

Description

estimate noise level voxel-wise using residuals from a truncated SH fit

WARNING: This command is deprecated and may be removed in future releases. Try using the dwidenoise command with the -noise option instead.

Options

- **-lmax order** set the maximum harmonic order for the output series. By default, the program will use the highest possible lmax given the number of diffusion-weighted images, up to a maximum of 8.

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

DW Shell selection options

- **-shell list** specify one or more diffusion-weighted gradient shells to use during processing, as a comma-separated list of the desired approximate b-values. Note that some commands are incompatible with multiple shells, and will throw an error if more than one b-value is provided.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

dwi2tensor

Synopsis

```
dwi2tensor [ options ]  dwi dt
```

- *dwi*: the input dwi image.
- *dt*: the output dt image.

Description

Diffusion (kurtosis) tensor estimation using iteratively reweighted linear least squares estimator.

The tensor coefficients are stored in the output image as follows: volumes 0-5: D11, D22, D33, D12, D13, D23 ; If diffusion kurtosis is estimated using the -dkt option, these are stored as follows: volumes 0-2: W1111, W2222, W3333 ; volumes 3-8: W1112, W1113, W1222, W1333, W2223, W2333 ; volumes 9-11: W1122, W1133, W2233 ; volumes 12-14: W1123, W1223, W1233 ;

Options

- **-mask image** only perform computation within the specified binary brain mask image.
- **-b0 image** the output b0 image.
- **-dkt image** the output dkt image.
- **-iter integer** number of iterative reweightings (default: 2); set to 0 for ordinary linear least squares.
- **-predicted_signal image** the predicted dwi image.

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Veraart, J.; Sijbers, J.; Sunaert, S.; Leemans, A. & Jeurissen, B. Weighted linear least squares estimation of diffusion MRI parameters: strengths, limitations, and pitfalls. NeuroImage, 2013, 81, 335-346

Author: Ben Jeurissen (ben.jeurissen@uantwerpen.be)

dwidenoise

Synopsis

```
dwidenoise [ options ]  dwi out
```

- *dwi*: the input diffusion-weighted image.
- *out*: the output denoised DWI image.

Description

Denoise DWI data and estimate the noise level based on the optimal threshold for PCA.

DWI data denoising and noise map estimation by exploiting data redundancy in the PCA domain using the prior knowledge that the eigenspectrum of random covariance matrices is described by the universal Marchenko Pastur distribution.

Important note: image denoising must be performed as the first step of the image processing pipeline. The routine will fail if interpolation or smoothing has been applied to the data prior to denoising.

Note that this function does not correct for non-Gaussian noise biases.

Options

- **-mask image** only perform computation within the specified binary brain mask image.
- **-extent window** set the window size of the denoising filter. (default = 5,5,5)
- **-noise level** the output noise map.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Veraart, J.; Novikov, D.S.; Christiaens, D.; Ades-aron, B.; Sijbers, J. & Fieremans, E. Denoising of diffusion MRI using random matrix theory. NeuroImage, 2016, in press, doi: 10.1016/j.neuroimage.2016.08.016

Veraart, J.; Fieremans, E. & Novikov, D.S. Diffusion MRI noise mapping using random matrix theory. Magn. Res. Med., 2016, early view, doi: 10.1002/mrm.26059

Author: Daan Christiaens (daan.christiaens@kuleuven.be) & Jelle Veraart (jelle.veraart@nyumc.org) & J-Donald Tournier (jdtournier@gmail.com)

dwiextract

Synopsis

```
dwiextract [ options ]  input output
```

- *input*: the input DW image.
- *output*: the output image (diffusion-weighted volumes by default.

Description

Extract either diffusion-weighted volumes or b=0 volumes from an image containing both

Options

- **-bzero** output b=0 volumes instead of the diffusion weighted volumes.

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

DW Shell selection options

- **-shell list** specify one or more diffusion-weighted gradient shells to use during processing, as a comma-separated list of the desired approximate b-values. Note that some commands are incompatible with multiple shells, and will throw an error if more than one b-value is provided.

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

dwinormalise

Synopsis

```
dwinormalise [ options ]  input mask output
```

- *input*: the input DWI image containing volumes that are both diffusion weighted and b=0
- *mask*: the input mask image used to normalise the intensity
- *output*: the output DWI intensity normalised image

Description

Intensity normalise the b=0 signal within a supplied white matter mask

Options

- **-intensity value** normalise the b=0 signal to the specified value (Default: 1e+003)
- **-percentile value** define the percentile of the mask intensties used for normalisation. If this option is not supplied then the median value (50th percentile) will be normalised to the desired intensity value.

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fixel2sh

Synopsis

```
fixel2sh [ options ]  fixel_in sh_out
```

- *fixel_in*: the input sparse fixel image.
- *sh_out*: the output sh image.

Description

convert a fixel-based sparse-data image into an SH image that can be visually evaluated using MRview

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

MRtrix is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

For more details, see www.mrtrix.org

fixel2tsf

Synopsis

```
fixel2tsf [ options ]  fixel_in tracks tsf
```

- *fixel_in*: the input fixel image
- *tracks*: the input track file
- *tsf*: the output track scalar file

Description

Map fixel values to a track scalar file based on an input tractogram. This is useful for visualising the output from fixelcfestats in 3D.

Options

- **-angle value** the max anglular threshold for computing correspondence between a fixel direction and track tangent (default = 30 degrees)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

For more details, see www.mrtrix.org

fixel2voxel

Synopsis

```
fixel2voxel [ options ]  fixel_in operation image_out
```

- *fixel_in*: the input sparse fixel image.
- *operation*: the operation to apply, one of: mean, sum, product, rms, var, std, min, max, absmax, magmax, count, complexity, sf, dec_unit, dec_scaled, split_size, split_value, split_dir.
- *image_out*: the output scalar image.

Description

convert a fixel-based sparse-data image into some form of scalar image. This could be: - Some statistic computed across all fixel values within a voxel: mean, sum, product, rms, var, std, min, max, absmax, magmax- The number of fixels in each voxel: count- Some measure of crossing-fibre organisation: complexity, sf ('single-fibre')- A 4D directionally-encoded colour image: dec_unit, dec_scaled- A 4D scalar image with one 3D volume per fixel: split_size, split_value- A 4D image with three 3D volumes per fixel direction: split_dir

Options

- **-weighted** weight the contribution of each fixel to the per-voxel result according to its volume (note that this option is not applicable for all operations, and should be avoided if the value stored in the fixel image is itself the estimated fibre volume)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- Reference for 'complexity' operation:Riffert, T. W.; Schreiber, J.; Anwander, A. & Knosche, T. R. Beyond Fractional Anisotropy: Extraction of bundle-specific structural metrics from crossing fibre models. NeuroImage, 2014 (in press)

Author: Robert E. Smith (robert.smith@florey.edu.au)

fixelcalc

Synopsis

```
fixelcalc [ options ]  input1 operation input2 output
```

- *input1*: the input fixel image.
- *operation*: the type of operation to be applied (either add, sub, mult or divide)
- *input2*: the input fixel image.
- *output*: the output fixel image.

Description

Perform basic calculations (add, subtract, multiply, divide) between two fixel images

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fixelcfestats

Synopsis

```
fixelcfestats [ options ]  input template design contrast tracks output
```

- *input*: a text file listing the file names of the input fixel images
- *template*: the fixel mask used to define fixels of interest. This can be generated by thresholding the group average AFD fixel image.
- *design*: the design matrix. Note that a column of 1's will need to be added for correlations.
- *contrast*: the contrast vector, specified as a single row of weights
- *tracks*: the tracks used to determine fixel-fixel connectivity
- *output*: the filename prefix for all output.

Description

Fixel-based analysis using connectivity-based fixel enhancement and non-parametric permutation testing.

Options

- **-notest** don't perform permutation testing and only output population statistics (effect size, stdev etc)
- **-negative** automatically test the negative (opposite) contrast. By computing the opposite contrast simultaneously the computation time is reduced.
- **-nperms num** the number of permutations (default: 5000).
- **-cfe_dh value** the height increment used in the cfe integration (default: 0.1)
- **-cfe_e value** cfe extent exponent (default: 2)
- **-cfe_h value** cfe height exponent (default: 3)
- **-cfe_c value** cfe connectivity exponent (default: 0.5)
- **-angle value** the max angle threshold for computing inter-subject fixel correspondence (Default: 45 degrees)
- **-connectivity threshold** a threshold to define the required fraction of shared connections to be included in the neighbourhood (default: 0.01)
- **-smooth FWHM** smooth the fixel value along the fibre tracts using a Gaussian kernel with the supplied FWHM (default: 10mm)
- **-nonstationary** do adjustment for non-stationarity

- **-nperms_nonstationary num** the number of permutations used when precomputing the empirical statistic image for nonstationary correction (Default: 5000)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Raffelt, D.; Smith, RE.; Ridgway, GR.; Tournier, JD.; Vaughan, DN.; Rose, S.; Henderson, R.; Connelly, A.Connectivity-based fixel enhancement: Whole-brain statistical analysis of diffusion MRI measures in the presence of crossing fibres. Neuroimage, 2015, 15(117):40-55

- If using the -nonstationary option: Salimi-Khorshidi, G. Smith, S.M. Nichols, T.E. Adjusting the effect of nonstationarity in cluster-based and TFCE inference. NeuroImage, 2011, 54(3), 2006-19

Author: David Raffelt (david.raffelt@florey.edu.au)

fixelcorrespondence

Synopsis

```
fixelcorrespondence [ options ]  subject template output
```

- *subject*: the input subject fixel image.
- *template*: the input template fixel image.
- *output*: the output fixel image.

Description

Obtain angular correpondence by mapping subject fixels to a template fixel mask. It is assumed that the subject image has already been spatially normalised and is aligned with the template.

Options

- **-angle value** the max angle threshold for computing inter-subject fixel correspondence (Default: 45 degrees)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fixellog

Synopsis

```
fixellog [ options ]  input output
```

- *input*: the input fixel image.
- *output*: the output fixel image.

Description

compute the natural logarithm of all values in a fixel image

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fixelreorient

Synopsis

```
fixelreorient [ options ]  input warp output
```

- *input*: the input fixel image.
- *warp*: a 4D deformation field used to perform reorientation. Reorientation is performed by applying the Jacobian affine transform in each voxel in the warp, then re-normalising the vector representing the fixel direction
- *output*: the output fixel image.

Description

Reorient fixel directions using the local affine transformation (Jacobian matrix) of an input warp.

Options

Standard options

- **-info** display information messages.

- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fixelstats

Synopsis

```
fixelstats [ options ]  input
```

- *input*: the input fixel image.

Description

Compute fixel image statistics

Options

Statistics options

- **-output field** output only the field specified. Multiple such options can be supplied if required. Choices are: mean, median, std, min, max, count. Useful for use in scripts.
- **-mask image** only perform computation within the specified binary mask image.
- **-histogram file** generate histogram of intensities and store in specified text file. Note that the first line of the histogram gives the centre of the bins.
- **-bins num** the number of bins to use to generate the histogram (default = 100).
- **-dump file** dump the voxel intensities to a text file.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fixelthreshold

Synopsis

```
fixelthreshold [ options ]  fixel_in threshold fixel_out
```

- *fixel_in*: the input fixel image.
- *threshold*: the input threshold
- *fixel_out*: the output fixel image

Description

Threshold the values in a fixel image

Options

- **-crop** remove fixels that fall below threshold (instead of assigning their value to zero or one)
- **-invert** invert the output image (i.e. below threshold fixels are included instead)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

fod2dec

Synopsis

```
fod2dec [ options ]  input output
```

- *input*: The input FOD image (spherical harmonic coefficients).
- *output*: The output DEC image (weighted RGB triplets).

Description

Generate FOD-based DEC maps, with optional panchromatic sharpening and/or luminance/perception correction.

By default, the FOD-based DEC is weighted by the integral of the FOD. To weight by another scalar map, use the outputmap option. This option can also be used for panchromatic sharpening, e.g., by supplying a T1 (or other sensible) anatomical volume with a higher spatial resolution.

Options

- **-mask image** Only perform DEC computation within the specified mask image.
- **-threshold value** FOD amplitudes below the threshold value are considered zero.

- **-outputmap image** Weight the computed DEC map by a provided outputmap. If the outputmap has a different grid, the DEC map is first resliced and renormalised. To achieve panchromatic sharpening, provide an image with a higher spatial resolution than the input FOD image; e.g., a T1 anatomical volume. Only the DEC is subject to the mask, so as to allow for partial colouring of the outputmap. Default when this option is *not* provided: integral of input FOD, subject to the same mask/threshold as used for DEC computation.
- **-no-weight** Do not weight the DEC map (reslicing and renormalising still possible by explicitly providing the outputmap option as a template).
- **-lum** Correct for luminance/perception, using default values Cr,Cg,Cb = 0.3,0.5,0.2 and gamma = 2.2 (*not* correcting is the theoretical equivalent of Cr,Cg,Cb = 1,1,1 and gamma = 2).
- **-lum-coefs values** The coefficients Cr,Cg,Cb to correct for luminance/perception. Note: this implicitly switches on luminance/perception correction, using a default gamma = 2.2 unless specified otherwise.
- **-lum-gamma value** The gamma value to correct for luminance/perception. Note: this implicitly switches on luminance/perception correction, using a default Cr,Cg,Cb = 0.3,0.5,0.2 unless specified otherwise.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Dhollander T, Smith RE, Tournier JD, Jeurissen B, Connelly A. Time to move on: an FOD-based DEC map to replace DTI's trademark DEC FA. Proc Intl Soc Mag Reson Med, 2015, 23, 1027.

Dhollander T, Raffelt D, Smith RE, Connelly A. Panchromatic sharpening of FOD-based DEC maps by structural T1 information. Proc Intl Soc Mag Reson Med, 2015, 23, 566.

Author: Thijs Dhollander (thijs.dhollander@gmail.com)

fod2fixel

Synopsis

```
fod2fixel [ options ]  fod
```

- *fod*: the input fod image.

Description

use a fast-marching level-set method to segment fibre orientation distributions, and save parameters of interest as fixel images

Options

- **-mask image** only perform computation within the specified binary brain mask image.

Metric values for fixel-based sparse output images

- **-afd image** store the total Apparent Fibre Density per fixel (integral of FOD lobe)
- **-peak image** store the peak FOD amplitude per fixel
- **-disp image** store a measure of dispersion per fixel as the ratio between FOD lobe integral and peak amplitude

FOD FMLS segmenter options

- **-fmls_ratio_integral_to_neg value** threshold the ratio between the integral of a positive FOD lobe, and the integral of the largest negative lobe. Any lobe that fails to exceed the integral dictated by this ratio will be discarded. Default: 0.
- **-fmls_ratio_peak_to_mean_neg value** threshold the ratio between the peak amplitude of a positive FOD lobe, and the mean peak amplitude of all negative lobes. Any lobe that fails to exceed the peak amplitude dictated by this ratio will be discarded. Default: 1.
- **-fmls_peak_value value** threshold the raw peak amplitude of positive FOD lobes. Any lobe for which the peak amplitude is smaller than this threshold will be discarded. Default: 0.1.
- **-fmls_no_thresholds** disable all FOD lobe thresholding; every lobe with a positive FOD amplitude will be retained.
- **-fmls_peak_ratio_to_merge value** specify the amplitude ratio between a sample and the smallest peak amplitude of the adjoining lobes, above which the lobes will be merged. This is the relative amplitude between the smallest of two adjoining lobes, and the 'bridge' between the two lobes. A value of 1.0 will never merge two peaks into a single lobe; a value of 0.0 will always merge lobes unless they are bisected by a zero crossing. Default: 1.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.

- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- Reference for the FOD segmentation method:Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. SIFT: Spherical-deconvolution informed filtering of tractograms. NeuroImage, 2013, 67, 298-312 (Appendix 2)
- Reference for Apparent Fibre Density:Raffelt, D.; Tournier, J.-D.; Rose, S.; Ridgway, G.R.; Henderson, R.; Crozier, S.; Salvado, O.; Connelly, A. Apparent Fibre Density: a novel measure for the analysis of diffusion-weighted magnetic resonance images.Neuroimage, 2012, 15;59(4), 3976-94.

Author: Robert E. Smith (robert.smith@florey.edu.au)

label2colour

Synopsis

```
label2colour [ options ]  nodes_in colour_out
```

- *nodes_in*: the input node parcellation image
- *colour_out*: the output colour image

Description

convert a parcellated image (where values are node indices) into a colour image (many software packages handle this colouring internally within their viewer program; this binary explicitly converts a parcellation image into a colour image that should be viewable in any software)

Options

- **-lut file** Provide the relevant colour lookup table (if not provided, nodes will be coloured randomly)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

label2mesh

Synopsis

```
label2mesh [ options ]  nodes_in mesh_out
```

- *nodes_in*: the input node parcellation image
- *mesh_out*: the output mesh file

Description

generate meshes from a label image.

Options

- **-blocky** generate 'blocky' meshes with precise delineation of voxel edges, rather than the default Marching Cubes approach

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

labelconvert

Synopsis

```
labelconvert [ options ]  path_in lut_in lut_out image_out
```

- *path_in*: the input image
- *lut_in*: the connectome lookup table for the input image
- *lut_out*: the target connectome lookup table for the output image
- *image_out*: the output image

Description

Convert a connectome node image from one lookup table to another. Typical usage is to convert a parcellation image provided by some other software, based on the lookup table provided by that software, to conform to a new lookup table, particularly one where the node indices increment from 1, in preparation for connectome construction; examples of such target lookup table files are provided in src//connectome//tables//

Options

- **-spine image** provide a manually-defined segmentation of the base of the spine where the streamlines terminate, so that this can become a node in the connection matrix.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

maskfilter

Synopsis

```
maskfilter [ options ]  input filter output
```

- *input*: the input image.
- *filter*: the type of filter to be applied (clean, connect, dilate, erode, median)
- *output*: the output image.

Description

Perform filtering operations on 3D / 4D mask images.

The available filters are: clean, connect, dilate, erode, median.

Each filter has its own unique set of optional parameters.

Options

Options for mask cleaning filter

- **-scale value** the maximum scale used to cut bridges. A certain maximum scale cuts bridges up to a width (in voxels) of 2x the provided scale. (Default: 2)

Options for connected-component filter

- **-axes axes** specify which axes should be included in the connected components. By default only the first 3 axes are included. The axes should be provided as a comma-separated list of values.
- **-largest** only retain the largest connected component
- **-connectivity** use 26-voxel-neighbourhood connectivity (Default: 6)

Options for dilate / erode filters

- **-npass value** the number of times to repeatedly apply the filter

Options for median filter

- **-extent voxels** specify the extent (width) of kernel size in voxels. This can be specified either as a single value to be used for all axes, or as a comma-separated list of the extent for each axis. The default is 3x3x3.

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au), David Raffelt (david.raffelt@florey.edu.au), Thijs Dhollander (thijs.dhollander@gmail.com) and J-Donald Tournier (jdtournier@gmail.com)

mesh2pve

Synopsis

```
mesh2pve [ options ]  source template output
```

- *source*: the mesh file; note vertices must be defined in realspace coordinates
- *template*: the template image
- *output*: the output image

Description

convert a mesh surface to a partial volume estimation image.

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. Anatomically-constrained tractography: Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 2012, 62, 1924-1938

Author: Robert E. Smith (robert.smith@florey.edu.au)

meshconvert

Synopsis

```
meshconvert [ options ]  input output
```

- *input*: the input mesh file
- *output*: the output mesh file

Description

convert meshes between different formats, and apply transformations.

Options

- **-binary** write the output file in binary format

Options for applying spatial transformations to vertices

- **-transform_first2real image** transform vertices from FSL FIRST's native corrdinate space to real space
- **-transform_real2first image** transform vertices from FSL real space to FIRST's native corrdinate space
- **-transform_voxel2real image** transform vertices from voxel space to real space
- **-transform_real2voxel image** transform vertices from real space to voxel space

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

For more details, see www.mrtrix.org

meshfilter

Synopsis

```
meshfilter [ options ]  input filter output
```

- *input*: the input mesh file
- *filter*: the filter to apply.Options are: smooth
- *output*: the output mesh file

Description

apply filter operations to meshes.

Options

Options for mesh smoothing filter

- **-smooth_spatial value** spatial extent of smoothing (default: 10mm)
- **-smooth_influence value** influence factor for smoothing (default: 10)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

mraverageheader

Synopsis

```
mraverageheader [ options ]  input [ input ... ] output
```

- *input*: the input image(s).
- *output*: the output image

Description

This command calculates the average (unbiased) coordinate space of all input images

Options

- **-padding value** boundary box padding in voxels. Default: 0
- **-resolution type** subsampling of template compared to smallest voxel size in any input image. Valid options are 'mean': unbiased but loss of resolution for individual images possible, and 'max': smallest voxel size of any input image defines the resolution. Default: mean
- **-fill** set the intensity in the first volume of the average space to 1

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Maximilian Pietsch (maximilian.pietsch@kcl.ac.uk)

mrcalc

Synopsis

```
mrcalc [ options ]  operand [ operand ... ]
```

- *operand*: an input image, intensity value, or the special keywords 'rand' (random number between 0 and 1) or 'randn' (random number from unit std.dev. normal distribution).

Description

apply generic voxel-wise mathematical operations to images.

This command will only compute per-voxel operations. Use 'mrmath' to compute summary statistics across images or along image axes.

This command uses a stack-based syntax, with operators (specified using options) operating on the top-most entries (i.e. images or values) in the stack. Operands (values or images) are pushed onto the stack in the order they appear (as arguments) on the command-line, and operators (specified as options) operate on and consume the top-most entries in the stack, and push their output as a new entry on the stack. For example:

    $ mrcalc a.mif 2 -mult r.mif

performs the operation r = 2*a for every voxel a,r in images a.mif and r.mif respectively. Similarly:

    $ mrcalc a.mif -neg b.mif -div -exp 9.3 -mult r.mif

performs the operation r = 9.3*exp(-a/b), and:

    $ mrcalc a.mif b.mif -add c.mif d.mif -mult 4.2 -add -div r.mif

performs r = (a+b)/(c*d+4.2).

As an additional feature, this command will allow images with different dimensions to be processed, provided they satisfy the following conditions: for each axis, the dimensions match if they are the same size, or one of them has size one. In the latter case, the entire image will be replicated along that axis. This allows for example a 4D image of size [X Y Z N] to be added to a 3D image of size [X Y Z], as if it consisted of N copies of the 3D image along the 4th axis (the missing dimension is assumed to have size 1). Another example would a single-voxel 4D image of size [1 1 1 N], multiplied by a 3D image of size [X Y Z], which would allow the creation of a 4D image where each volume consists of the 3D image scaled by the corresponding value for that volume in the single-voxel image.

Options

Unary operators

- **-abs** absolute value
- **-neg** negative value

- **-sqrt** square root
- **-exp** exponential function
- **-log** natural logarithm
- **-log10** common logarithm
- **-cos** cosine
- **-sin** sine
- **-tan** tangent
- **-cosh** hyperbolic cosine
- **-sinh** hyperbolic sine
- **-tanh** hyperbolic tangent
- **-acos** inverse cosine
- **-asin** inverse sine
- **-atan** inverse tangent
- **-acosh** inverse hyperbolic cosine
- **-asinh** inverse hyperbolic sine
- **-atanh** inverse hyperbolic tangent
- **-round** round to nearest integer
- **-ceil** round up to nearest integer
- **-floor** round down to nearest integer
- **-isnan** true (1) is operand is not-a-number (NaN)
- **-isinf** true (1) is operand is infinite (Inf)
- **-finite** true (1) is operand is finite (i.e. not NaN or Inf)
- **-real** real part of complex number
- **-imag** imaginary part of complex number
- **-phase** phase of complex number
- **-conj** complex conjugate

Binary operators

- **-add** add values
- **-subtract** subtract nth operand from (n-1)th
- **-multiply** multiply values
- **-divide** divide (n-1)th operand by nth
- **-pow** raise (n-1)th operand to nth power
- **-min** smallest of last two operands
- **-max** greatest of last two operands
- **-lt** less-than operator (true=1, false=0)

- **-gt** greater-than operator (true=1, false=0)
- **-le** less-than-or-equal-to operator (true=1, false=0)
- **-ge** greater-than-or-equal-to operator (true=1, false=0)
- **-eq** equal-to operator (true=1, false=0)
- **-neq** not-equal-to operator (true=1, false=0)
- **-complex** create complex number using the last two operands as real,imaginary components

Ternary operators

- **-if** if first operand is true (non-zero), return second operand, otherwise return third operand

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

mrcat

Synopsis

```
mrcat [ options ]  image1 image2 [ image2 ... ] output
```

- *image1*: the first input image.
- *image2*: additional input image(s).
- *output*: the output image.

Description

concatenate several images into one

Options

- **-axis axis** specify axis along which concatenation should be performed. By default, the program will use the last non-singleton, non-spatial axis of any of the input images - in other words axis 3 or whichever axis (greater than 3) of the input images has size greater than one.

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

MRtrix is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

For more details, see www.mrtrix.org

mrcheckerboardmask

Synopsis

```
mrcheckerboardmask [ options ]  input output
```

- *input*: the input image to be .
- *output*: the output binary image mask.

Description

create bitwise checkerboard image

Options

- **-tiles value** specify the number of tiles in any direction
- **-invert** invert output binary mask.
- **-nan** use NaN as the output zero value.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Max Pietsch (maximilian.pietsch@kcl.ac.uk)

For more details, see www.mrtrix.org

mrclusterstats

Synopsis

```
mrclusterstats [ options ]  input design contrast mask output
```

- *input*: a text file containing the file names of the input images, one file per line
- *design*: the design matrix, rows should correspond with images in the input image text file
- *contrast*: the contrast matrix, only specify one contrast as it will automatically compute the opposite contrast.
- *mask*: a mask used to define voxels included in the analysis.
- *output*: the filename prefix for all output.

Description

Voxel-based analysis using permutation testing and threshold-free cluster enhancement.

Options

- **-notest** don't perform permutation testing and only output population statistics (effect size, stdev etc)
- **-negative** automatically test the negative (opposite) contrast. By computing the opposite contrast simultaneously the computation time is reduced.
- **-nperms num** the number of permutations (default = 5000).
- **-threshold value** the cluster-forming threshold to use for a standard cluster-based analysis. This disables TFCE, which is the default otherwise.
- **-tfce_dh value** the height increment used in the TFCE integration (default = 0.1)
- **-tfce_e value** TFCE extent parameter (default = 0.5)
- **-tfce_h value** TFCE height parameter (default = 2)
- **-connectivity** use 26-voxel-neighbourhood connectivity (Default: 6)
- **-nonstationary** perform non-stationarity correction (currently only implemented with tfce)
- **-nperms_nonstationary num** the number of permutations used when precomputing the empirical statistic image for nonstationary correction (Default: 5000)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- If not using the -threshold command-line option:Smith, S. M. & Nichols, T. E. Threshold-free cluster enhancement: Addressing problems of smoothing, threshold dependence and localisation in cluster inference. NeuroImage, 2009, 44, 83-98
- If using the -nonstationary option:Salimi-Khorshidi, G. Smith, S.M. Nichols, T.E. Adjusting the effect of non-stationarity in cluster-based and TFCE inference. Neuroimage, 2011, 54(3), 2006-19

Author: David Raffelt (david.raffelt@florey.edu.au)

mrconvert

Synopsis

```
mrconvert [ options ]  input output
```

- *input*: the input image.
- *output*: the output image.

Description

perform conversion between different file types and optionally extract a subset of the input image.

If used correctly, this program can be a very useful workhorse. In addition to converting images between different formats, it can be used to extract specific studies from a data set, extract a specific region of interest, or flip the images.

Options

- **-coord axis coord** extract data from the input image only at the coordinates specified.
- **-vox sizes** change the voxel dimensions of the output image. The new sizes should be provided as a comma-separated list of values. Only those values specified will be changed. For example: 1,,3.5 will change the voxel size along the x & z axes, and leave the y-axis voxel size unchanged.

- **-axes axes** specify the axes from the input image that will be used to form the output image. This allows the permutation, ommission, or addition of axes into the output image. The axes should be supplied as a comma-separated list of axes. Any ommitted axes must have dimension 1. Axes can be inserted by supplying -1 at the corresponding position in the list.
- **-scaling values** specify the data scaling parameters used to rescale the intensity values. These take the form of a comma-separated 2-vector of floating-point values, corresponding to offset & scale, with final intensity values being given by offset + scale * stored_value. By default, the values in the input image header are passed through to the output image header when writing to an integer image, and reset to 0,1 (no scaling) for floating-point and binary images. Note that his option has no effect for floating-point and binary images.

Modify generic header entries

- **-clear_property key** remove the specified key from the image header altogether.
- **-set_property key value** set the value of the specified key in the image header.
- **-append_property key value** append the given value to the specified key in the image header (this adds the value specified as a new line in the header value).

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.

DW gradient table export options

- **-export_grad_mrtrix path** export the diffusion-weighted gradient table to file in MRtrix format
- **-export_grad_fsl bvecs_path bvals_path** export the diffusion-weighted gradient table to files in FSL (bvecs / bvals) format

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

mrcrop

Synopsis

```
mrcrop [ options ]  image_in image_out
```

- *image_in*: the image to be cropped
- *image_out*: the output path for the resulting cropped image

Description

Crop an image series to a reduced field of view, using either manual setting of axis dimensions, or a computed mask image corresponding to the brain.

If using a mask, a gap of 1 voxel will be left at all 6 edges of the image such that trilinear interpolation upon the resulting images is still valid.

This is useful for axially-acquired brain images, where the image size can be reduced by a factor of 2 by removing the empty space on either side of the brain.

Options

- **-mask image** crop the input image according to the spatial extent of a mask image
- **-axis index start end** crop the input image in the provided axis

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

mrfilter

Synopsis

```
mrfilter [ options ]  input filter output
```

- *input*: the input image.
- *filter*: the type of filter to be applied
- *output*: the output image.

Description

Perform filtering operations on 3D / 4D MR images. For 4D images, each 3D volume is processed independently.

The available filters are: fft, gradient, median, smooth, normalise.

Each filter has its own unique set of optional parameters.

Options

Options for FFT filter

- **-axes list** the axes along which to apply the Fourier Transform. By default, the transform is applied along the three spatial axes. Provide as a comma-separate list of axis indices.
- **-inverse** apply the inverse FFT
- **-magnitude** output a magnitude image rather than a complex-valued image
- **-centre_zero** re-arrange the FFT results so that the zero-frequency component appears in the centre of the image, rather than at the edges

Options for gradient filter

- **-stdev sigma** the standard deviation of the Gaussian kernel used to smooth the input image (in mm). The image is smoothed to reduced large spurious gradients caused by noise. Use this option to override the default stdev of 1 voxel. This can be specified either as a single value to be used for all 3 axes, or as a comma-separated list of 3 values, one for each axis.
- **-magnitude** output the gradient magnitude, rather than the default x,y,z components
- **-scanner** define the gradient with respect to the scanner coordinate frame of reference.

Options for median filter

- **-extent size** specify extent of median filtering neighbourhood in voxels. This can be specified either as a single value to be used for all 3 axes, or as a comma-separated list of 3 values, one for each axis (default: 3x3x3).

Options for normalisation filter

- **-extent size** specify extent of normalisation filtering neighbourhood in voxels. This can be specified either as a single value to be used for all 3 axes, or as a comma-separated list of 3 values, one for each axis (default: 3x3x3).

Options for smooth filter

- **-stdev mm** apply Gaussian smoothing with the specified standard deviation. The standard deviation is defined in mm (Default 1 voxel). This can be specified either as a single value to be used for all axes, or as a comma-separated list of the stdev for each axis.
- **-fwhm mm** apply Gaussian smoothing with the specified full-width half maximum. The FWHM is defined in mm (Default 1 voxel * 2.3548). This can be specified either as a single value to be used for all axes, or as a comma-separated list of the FWHM for each axis.
- **-extent voxels** specify the extent (width) of kernel size in voxels. This can be specified either as a single value to be used for all axes, or as a comma-separated list of the extent for each axis. The default extent is 2 * ceil(2.5 * stdev / voxel_size) - 1.

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au), David Raffelt (david.raffelt@florey.edu.au) and J-Donald Tournier (jdtournier@gmail.com)

mrinfo

Synopsis

```
mrinfo [ options ]  image [ image ... ]
```

- *image*: the input image(s).

Description

display header information, or extract specific information from the header.

By default, all information contained in each image header will be printed to the console in a reader-friendly format.

Alternatively, command-line options may be used to extract specific details from the header(s); these are printed to the console in a format more appropriate for scripting purposes or piping to file. If multiple options and/or images are provided, the requested header fields will be printed in the order in which they appear in the help page, with all requested details from each input image in sequence printed before the next image is processed.

The command can also write the diffusion gradient table from a single input image to file; either in the MRtrix or FSL format (bvecs/bvals file pair; includes appropriate diffusion gradient vector reorientation)

Options

- **-format** image file format
- **-ndim** number of image dimensions
- **-size** image size along each axis
- **-vox** voxel size along each image dimension
- **-datatype** data type used for image data storage
- **-stride** data strides i.e. order and direction of axes data layout
- **-offset** image intensity offset
- **-multiplier** image intensity multiplier
- **-transform** the voxel to image transformation
- **-norealign** do not realign transform to near-default RAS coordinate system (the default behaviour on image load). This is useful to inspect the transform and strides as they are actually stored in the header, rather than as MRtrix interprets them.
- **-property key** any text properties embedded in the image header under the specified key (use 'all' to list all keys found)

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).
- **-raw_dwgrad** do not modify the gradient table from what was found in the image headers. This skips the validation steps normally performed within MRtrix applications (i.e. do not verify that the number of entries in the gradient table matches the number of volumes in the image, do not scale b-values by gradient norms, do not normalise gradient vectors)

DW gradient table export options

- **-export_grad_mrtrix path** export the diffusion-weighted gradient table to file in MRtrix format
- **-export_grad_fsl bvecs_path bvals_path** export the diffusion-weighted gradient table to files in FSL (bvecs / bvals) format
- **-dwgrad** the diffusion-weighting gradient table, as stored in the header (i.e. without any interpretation, scaling of b-values, or normalisation of gradient vectors)
- **-shells** list the average b-value of each shell
- **-shellcounts** list the number of volumes in each shell

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (d.tournier@brain.org.au) and Robert E. Smith (robert.smith@florey.edu.au)

mrmath

Synopsis

```
mrmath [ options ]  input [ input ... ] operation output
```

- *input*: the input image(s).
- *operation*: the operation to apply, one of: mean, median, sum, product, rms, norm, var, std, min, max, absmax, magmax.
- *output*: the output image.

Description

compute summary statistic on image intensities either across images, or along a specified axis for a single image. Supported operations are:

mean, median, sum, product, rms (root-mean-square value), norm (vector 2-norm), var (unbiased variance), std (unbiased standard deviation), min, max, absmax (maximum absolute value), magmax (value with maximum absolute value, preserving its sign).

See also ‘mrcalc’ to compute per-voxel operations.

Options

- **-axis index** perform operation along a specified axis of a single input image

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

mrmesh

Synopsis

```
mrmesh [ options ]  input output
```

- *input*: the input image.
- *output*: the output mesh file.

Description

Generate a mesh file from an image.

Options

- **-blocky** generate a 'blocky' mesh that precisely represents the voxel edges
- **-threshold value** manually set the intensity threshold at which the mesh will be generated (if omitted, a threshold will be determined automatically)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

mrmetric

Synopsis

```
mrmetric [ options ]  image1 image2
```

- *image1*: the first input image.
- *image2*: the second input image.

Description

computes a dissimilarity metric between two images. Currently only the mean squared difference is implemented

Options

- **-space iteration method** voxel (default): per voxel image1: scanner space of image 1 image2: scanner space of image 2 average: scanner space of the average affine transformation of image 1 and 2
- **-interp method** set the interpolation method to use when reslicing (choices: nearest, linear, cubic, sinc. Default: linear).
- **-metric method** define the dissimilarity metric used to calculate the cost. Choices: diff (squared differences), cc (negative cross correlation). Default: diff). cc is only implemented for -space average and -interp linear.
- **-mask1 image** mask for image 1
- **-mask2 image** mask for image 2
- **-nonormalisation** do not normalise the dissimilarity metric to the number of voxels.
- **-overlap** output number of voxels that were used.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au) and Max Pietsch (maximilian.pietsch@kcl.ac.uk)

mrpad

Synopsis

```
mrpad [ options ]  image_in image_out
```

- *image_in*: the image to be padded
- *image_out*: the output path for the resulting padded image

Description

Pad an image to increase the FOV

Options

- **-uniform number** pad the input image by a uniform number of voxels on all sides (in 3D)
- **-axis index lower upper** pad the input image along the provided axis (defined by index). Lower and upper define the number of voxels to add to the lower and upper bounds of the axis

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

mrregister

Synopsis

```
mrregister [ options ]  image1 image2
```

- *image1*: input image 1 ('moving')
- *image2*: input image 2 ('template')

Description

Register two images together using a rigid, affine or a non-linear transformation model.

By default this application will perform an affine, followed by non-linear registration.

FOD registration (with apodised point spread reorientation) will be performed by default if the number of volumes in the 4th dimension equals the number of coefficients in an antipodally symmetric spherical harmonic series (e.g. 6, 15, 28 etc). The -no_reorientation option can be used to force reorientation off if required.

Non-linear registration computes warps to map from both image1->image2 and image2->image1. Similar to Avants (2008) Med Image Anal. 12(1): 26–41, registration is performed by matching both the image1 and image2 in a 'midway space'. Warps can be saved as two deformation fields that map directly between image1->image2 and image2->image1, or if using -nl_warp_full as a single 5D file that stores all 4 warps image1->mid->image2, and image2->mid->image1. The 5D warp format stores x,y,z deformations in the 4th dimension, and uses the 5th dimension to index the 4 warps. The affine transforms estimated (to midway space) are also stored as comments in the image header. The 5D warp file can be used to reinitialise subsequent registrations, in addition to transforming images to midway space (e.g. for intra-subject alignment in a 2-time-point longitudinal analysis).

Options

- **-type choice** the registration type. Valid choices are: rigid, affine, nonlinear, rigid_affine, rigid_nonlinear, affine_nonlinear, rigid_affine_nonlinear (Default: affine_nonlinear)
- **-transformed image** image1 after registration transformed to the space of image2
- **-transformed_midway image1_transformed image2_transformed** image1 and image2 after registration transformed to the midway space
- **-mask1 filename** a mask to define the region of image1 to use for optimisation.
- **-mask2 filename** a mask to define the region of image2 to use for optimisation.

Rigid registration options

- **-rigid file** the output text file containing the rigid transformation as a 4x4 matrix
- **-rigid_1tomidway file** the output text file containing the rigid transformation that aligns image1 to image2 in their common midway space as a 4x4 matrix
- **-rigid_2tomidway file** the output text file containing the rigid transformation that aligns image2 to image1 in their common midway space as a 4x4 matrix
- **-rigid_init_translation type** initialise the translation and centre of rotation Valid choices are: mass (aligns the centers of mass of both images, default), geometric (aligns geometric image centres) and none.
- **-rigid_init_rotation type** initialise the rotation Valid choices are: search (search for the best rotation using mean squared residuals), moments (rotation based on directions of intensity variance with respect to centre of mass), none (default).
- **-rigid_init_matrix file** initialise either the rigid, affine, or syn registration with the supplied rigid transformation (as a 4x4 matrix in scanner coordinates). Note that this overrides rigid_init_translation and rigid_init_rotation initialisation
- **-rigid_scale factor** use a multi-resolution scheme by defining a scale factor for each level using comma separated values (Default: 0.25,0.5,1.0)
- **-rigid_niter num** the maximum number of iterations. This can be specified either as a single number for all multi-resolution levels, or a single value for each level. (Default: 1000)

- **-rigid_metric type** valid choices are: diff (intensity differences), Default: diff
- **-rigid_metric.diff.estimator type** Valid choices are: l1 (least absolute: |x|), l2 (ordinary least squares), lp (least powers: |x|^1.2), Default: l2
- **-rigid_lmax num** explicitly set the lmax to be used per scale factor in rigid FOD registration. By default FOD registration will use lmax 0,2,4 with default scale factors 0.25,0.5,1.0 respectively. Note that no reorientation will be performed with lmax = 0.
- **-rigid_log file** write gradient descent parameter evolution to log file

Affine registration options

- **-affine file** the output text file containing the affine transformation as a 4x4 matrix
- **-affine_1tomidway file** the output text file containing the affine transformation that aligns image1 to image2 in their common midway space as a 4x4 matrix
- **-affine_2tomidway file** the output text file containing the affine transformation that aligns image2 to image1 in their common midway space as a 4x4 matrix
- **-affine_init_translation type** initialise the translation and centre of rotation Valid choices are: mass (aligns the centers of mass of both images), geometric (aligns geometric image centres) and none. (Default: mass)
- **-affine_init_rotation type** initialise the rotation Valid choices are: search (search for the best rotation using mean squared residuals), moments (rotation based on directions of intensity variance with respect to centre of mass), none (Default: none).
- **-affine_init_matrix file** initialise either the affine, or syn registration with the supplied affine transformation (as a 4x4 matrix in scanner coordinates). Note that this overrides affine_init_translation and affine_init_rotation initialisation
- **-affine_scale factor** use a multi-resolution scheme by defining a scale factor for each level using comma separated values (Default: 0.25,0.5,1.0)
- **-affine_niter num** the maximum number of iterations. This can be specified either as a single number for all multi-resolution levels, or a single value for each level. (Default: 1000)
- **-affine_metric type** valid choices are: diff (intensity differences), Default: diff
- **-affine_metric.diff.estimator type** Valid choices are: l1 (least absolute: |x|), l2 (ordinary least squares), lp (least powers: |x|^1.2), Default: l2
- **-affine_lmax num** explicitly set the lmax to be used per scale factor in affine FOD registration. By default FOD registration will use lmax 0,2,4 with default scale factors 0.25,0.5,1.0 respectively. Note that no reorientation will be performed with lmax = 0.
- **-affine_log file** write gradient descent parameter evolution to log file

Advanced linear transformation initialisation options

- **-init_translation.unmasked1** disregard mask1 for the translation initialisation (affects 'mass')
- **-init_translation.unmasked2** disregard mask2 for the translation initialisation (affects 'mass')
- **-init_rotation.unmasked1** disregard mask1 for the rotation initialisation (affects 'search' and 'moments')
- **-init_rotation.unmasked2** disregard mask2 for the rotation initialisation (affects 'search' and 'moments')
- **-init_rotation.search.angles angles** rotation angles for the local search in degrees between 0 and 180. (Default: 2,5,10,15,20)

- **-init_rotation.search.scale scale** relative size of the images used for the rotation search. (Default: 0.15)
- **-init_rotation.search.directions num** number of rotation axis for local search. (Default: 250)
- **-init_rotation.search.run_global** perform a global search. (Default: local)
- **-init_rotation.search.global.iterations num** number of rotations to investigate (Default: 10000)

Non-linear registration options

- **-nl_warp warp1 warp2** the non-linear warp output defined as two deformation fields, where warp1 can be used to transform image1->image2 and warp2 to transform image2->image1. The deformation fields also encapsulate any linear transformation estimated prior to non-linear registration.
- **-nl_warp_full image** output all warps used during registration. This saves four different warps that map each image to a midway space and their inverses in a single 5D image file. The 4th image dimension indexes the x,y,z component of the deformation vector and the 5th dimension indexes the field in this order: image1->midway, midway->image1, image2->midway, midway->image2. Where image1->midway defines the field that maps image1 onto the midway space using the reverse convention When linear registration is performed first, the estimated linear transform will be included in the comments of the image header, and therefore the entire linear and non-linear transform can be applied (in either direction) using this output warp file with mrtransform
- **-nl_init image** initialise the non-linear registration with the supplied warp image. The supplied warp must be in the same format as output using the -nl_warp_full option (i.e. have 4 deformation fields with the linear transforms in the image header)
- **-nl_scale factor** use a multi-resolution scheme by defining a scale factor for each level using comma separated values (Default: 0.25,0.5,1.0)
- **-nl_niter num** the maximum number of iterations. This can be specified either as a single number for all multi-resolution levels, or a single value for each level. (Default: 50)
- **-nl_update_smooth stdev** regularise the gradient update field with Gaussian smoothing (standard deviation in voxel units, Default 2.0 x voxel_size)
- **-nl_disp_smooth stdev** regularise the displacement field with Gaussian smoothing (standard deviation in voxel units, Default 1.0 x voxel_size)
- **-nl_grad_step num** the gradient step size for non-linear registration (Default: 0.5)
- **-nl_lmax num** explicitly set the lmax to be used per scale factor in non-linear FOD registration. By default FOD registration will use lmax 0,2,4 with default scale factors 0.25,0.5,1.0 respectively. Note that no reorientation will be performed with lmax = 0.

FOD registration options

- **-directions file** the directions used for FOD reorienation using apodised point spread functions (Default: 60 directions)
- **-noreorientation** turn off FOD reorientation. Reorientation is on by default if the number of volumes in the 4th dimension corresponds to the number of coefficients in an antipodally symmetric spherical harmonic series (i.e. 6, 15, 28, 45, 66 etc

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be,

uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- If FOD registration is being performed:Raffelt, D.; Tournier, J.-D.; Fripp, J; Crozier, S.; Connelly, A. & Salvado, O. Symmetric diffeomorphic registration of fibre orientation distributions. NeuroImage, 2011, 56(3), 1171-1180

Raffelt, D.; Tournier, J.-D.; Crozier, S.; Connelly, A. & Salvado, O. Reorientation of fiber orientation distributions using apodized point spread functions. Magnetic Resonance in Medicine, 2012, 67, 844-855

Author: David Raffelt (david.raffelt@florey.edu.au) & Max Pietsch (maximilian.pietsch@kcl.ac.uk)

mrresize

Synopsis

```
mrresize [ options ]  input output
```

- *input*: input image to be resized.
- *output*: the output image.

Description

Resize an image by defining the new image resolution, voxel size or a scale factor.

Note that if the image is 4D, then only the first 3 dimensions can be resized.

Also note that if the image is down-sampled, the appropriate smoothing is automatically applied using Gaussian smoothing.

Options

- **-size dims** define the new image size for the output image. This should be specified as a comma-separated list.
- **-voxel size** define the new voxel size for the output image. This can be specified either as a single value to be used for all dimensions, or as a comma-separated list of the size for each voxel dimension.
- **-scale factor** scale the image resolution by the supplied factor. This can be specified either as a single value to be used for all dimensions, or as a comma-separated list of scale factors for each dimension.
- **-interp method** set the interpolation method to use when resizing (choices: nearest, linear, cubic, sinc. Default: cubic).

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

mrstats

Synopsis

```
mrstats [ options ]  image
```

- *image*: the input image from which statistics will be computed.

Description

compute images statistics.

Options

Statistics options

- **-output field** output only the field specified. Multiple such options can be supplied if required. Choices are: mean, median, std, min, max, count. Useful for use in scripts.
- **-mask image** only perform computation within the specified binary mask image.
- **-histogram file** generate histogram of intensities and store in specified text file. Note that the first line of the histogram gives the centre of the bins.
- **-bins num** the number of bins to use to generate the histogram (default = 100).
- **-dump file** dump the voxel intensities to a text file.
- **-voxel pos** only perform computation within the specified voxel, supplied as a comma-separated vector of 3 integer values (multiple voxels can be included).
- **-position file** dump the position of the voxels in the mask to a text file.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

mrthreshold

Synopsis

```
mrthreshold [ options ]  input output
```

- *input*: the input image to be thresholded.
- *output*: the output binary image mask.

Description

create bitwise image by thresholding image intensity. By default, an optimal threshold is determined using a parameter-free method. Alternatively the threshold can be defined manually by the user or using a histogram-based analysis to cut out the background.

Options

- **-abs value** specify threshold value as absolute intensity.
- **-histogram** define the threshold by a histogram analysis to cut out the background. Note that only the first study is used for thresholding.
- **-percentile value** threshold the image at the ith percentile.
- **-top N** provide a mask of the N top-valued voxels
- **-bottom N** provide a mask of the N bottom-valued voxels
- **-invert** invert output binary mask.
- **-toppercent N** provide a mask of the N%% top-valued voxels
- **-bottompercent N** provide a mask of the N%% bottom-valued voxels
- **-nan** use NaN as the output zero value.
- **-ignorezero** ignore zero-valued input voxels.
- **-mask image** compute the optimal threshold based on voxels within a mask.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.

- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- If not using the -abs option:Ridgway, G. R.; Omar, R.; Ourselin, S.; Hill, D. L.; Warren, J. D. & Fox, N. C. Issues with threshold masking in voxel-based morphometry of atrophied brains. NeuroImage, 2009, 44, 99-111

Author: J-Donald Tournier (jdtournier@gmail.com)

mrtransform

Synopsis

```
mrtransform [ options ]  input output
```

- *input*: input image to be transformed.
- *output*: the output image.

Description

apply spatial transformations to an image.

If a linear transform is applied without a template image the command will modify the image header transform matrix

FOD reorientation (with apodised point spread functions) will be performed by default if the number of volumes in the 4th dimension equals the number of coefficients in an antipodally symmetric spherical harmonic series (e.g. 6, 15, 28 etc). The -no_reorientation option can be used to force reorientation off if required.

If a DW scheme is contained in the header (or specified separately), and the number of directions matches the number of volumes in the images, any transformation applied using the -linear option will be also be applied to the directions.

Options

Affine transformation options

- **-linear transform** specify a linear transform to apply, in the form of a 3x4 or 4x4 ascii file. Note the standard 'reverse' convention is used, where the transform maps points in the template image to the moving image. Note that the reverse convention is still assumed even if no -template image is supplied
- **-flip axes** flip the specified axes, provided as a comma-separated list of indices (0:x, 1:y, 2:z).
- **-inverse** apply the inverse transformation
- **-half** apply the matrix square root of the transformation. This can be combined with the inverse option.
- **-replace file** replace the linear transform of the original image by that specified, rather than applying it to the original image. The specified transform can be either a template image, or a 3x4 or 4x4 ascii file.
- **-identity** set the header transform of the image to the identity matrix

Regridding options

- **-template image** reslice the input image to match the specified template image grid.
- **-midway_space** reslice the input image to the midway space. Requires either the -template or -warp option. If used with -template and -linear option the input image will be resliced onto the grid halfway between the input and template. If used with the -warp option the input will be warped to the midway space defined by the grid of the input warp (i.e. half way between image1 and image2)
- **-interp method** set the interpolation method to use when reslicing (choices: nearest, linear, cubic, sinc. Default: cubic).

Non-linear transformation options

- **-warp image** apply a non-linear 4D deformation field to warp the input image. Each voxel in the deformation field must define the scanner space position that will be used to interpolate the input image during warping (i.e. pull-back/reverse warp convention). If the -template image is also supplied the deformation field will be resliced first to the template image grid. If no -template option is supplied then the output image will have the same image grid as the deformation field. This option can be used in combination with the -affine option, in which case the affine will be applied first)
- **-warp_full image** warp the input image using a 5D warp file output from mrregister. Any linear transforms in the warp image header will also be applied. The -warp_full option must be used in combination with either the -template option or the -midway_space option. If a -template image is supplied then the full warp will be used. By default the image1->image2 transform will be applied, however the -from 2 option can be used to apply the image2->image1 transform. Use the -midway_space option to warp the input image to the midway space. The -from option can also be used to define which warp to use when transforming to midway space
- **-from image** used to define which space the input image is when using the -warp_mid option. Use -from 1 to warp from image1 or -from 2 to warp from image2

Fibre orientation distribution handling options

- **-modulate** modulate FODs during reorientation to preserve the apparent fibre density across fibre bundle widths before and after the transformation
- **-directions file** directions defining the number and orientation of the apodised point spread functions used in FOD reorientation (Default: 300 directions)

- **-noreorientation** turn off FOD reorientation. Reorientation is on by default if the number of volumes in the 4th dimension corresponds to the number of coefficients in an antipodally symmetric spherical harmonic series (i.e. 6, 15, 28, 45, 66 etc

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.
- **-nan** Use NaN as the out of bounds value (Default: 0.0)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- If FOD reorientation is being performed:Raffelt, D.; Tournier, J.-D.; Crozier, S.; Connelly, A. & Salvado, O. Reorientation of fiber orientation distributions using apodized point spread functions. Magnetic Resonance in Medicine, 2012, 67, 844-855
- If FOD modulation is being performed:Raffelt, D.; Tournier, J.-D.; Rose, S.; Ridgway, G.R.; Henderson, R.; Crozier, S.; Salvado, O.; Connelly, A.; Apparent Fibre Density: a novel measure for the analysis of diffusion-weighted magnetic resonance images. NeuroImage, 2012, 15;59(4), 3976-94.

Author: J-Donald Tournier (jdtournier@gmail.com) and David Raffelt (david.raffelt@florey.edu.au) and Max Pietsch (maximilian.pietsch@kcl.ac.uk)

mrview

Synopsis

```
mrview [ options ] [ image ... ]
```

- *image*: An image to be loaded.

Description

The MRtrix image viewer.

Any images listed as arguments will be loaded and available through the image menu, with the first listed displayed initially. Any subsequent command-line options will be processed as if the corresponding action had been performed through the GUI.

Note that because images loaded as arguments (i.e. simply listed on the command-line) are opened before the GUI is shown, subsequent actions to be performed via the various command-line options must appear after the last argument. This is to avoid confusion about which option will apply to which image. If you need fine control over this, please use the -load or -select_image options. For example:

$ mrview -load image1.mif -interpolation 0 -load image2.mif -interpolation 0

or

$ mrview image1.mif image2.mif -interpolation 0 -select_image 2 -interpolation 0

Options

View options

- **-mode index** Switch to view mode specified by the integer index, as per the view menu.
- **-load image** Load image specified and make it current.
- **-reset** Reset the view according to current image. This resets the FOV, projection and focus.
- **-fov value** Set the field of view, in mm.
- **-focus x,y,z or boolean** Either set the position of the crosshairs in scanner coordinates, with the new position supplied as a comma-separated list of floating-point values or show or hide the focus cross hair using a boolean value as argument.

- **-voxel x,y,z** Set the position of the crosshairs in voxel coordinates, relative the image currently displayed. The new position should be supplied as a comma-separated list of floating-point values.
- **-volume idx** Set the volume index for the image displayed, as a comma-separated list of integers.
- **-plane index** Set the viewing plane, according to the mappping 0: sagittal; 1: coronal; 2: axial.
- **-lock yesno** Set whether view is locked to image axes (0: no, 1: yes).
- **-select_image index** Switch to image number specified, with reference to the list of currently loaded images.
- **-autoscale** Reset the image scaling to automatically determined range.
- **-interpolation boolean** Enable or disable image interpolation in main image.
- **-colourmap index** Switch the image colourmap to that specified, as per the colourmap menu.
- **-noannotations** Hide all image annotation overlays
- **-comments boolean** Show or hide image comments overlay.
- **-voxelinfo boolean** Show or hide voxel information overlay.
- **-orientationlabel boolean** Show or hide orientation label overlay.
- **-colourbar boolean** Show or hide colourbar overlay.
- **-imagevisible boolean** Show or hide the main image.
- **-intensity_range min,max** Set the image intensity range to that specified.

Window management options

- **-size width,height** Set the size of the view area, in pixel units.
- **-position x,y** Set the position of the main window, in pixel units.
- **-fullscreen** Start fullscreen.
- **-exit** Quit MRView.

Debugging options

- **-fps** Display frames per second, averaged over the last 10 frames. The maximum over the last 3 seconds is also displayed.

Overlay tool options

- **-overlay.load image** Loads the specified image on the overlay tool.
- **-overlay.opacity value** Sets the overlay opacity to floating value [0-1].
- **-overlay.interpolation_on** Enables overlay image interpolation.
- **-overlay.interpolation_off** Disables overlay image interpolation.
- **-overlay.colourmap index** Sets the colourmap of the overlay as indexed in the colourmap dropdown menu.

ROI editor tool options

- **-roi.load image** Loads the specified image on the ROI editor tool.
- **-roi.opacity value** Sets the overlay opacity to floating value [0-1].

Tractography tool options

- **-tractography.load tracks** Load the specified tracks file into the tractography tool.
- **-tractography.thickness value** Line thickness of tractography display, [-1.0, 1.0], default is 0.0.
- **-tractography.opacity value** Opacity of tractography display, [0.0, 1.0], default is 1.0.

ODF tool options

- **-odf.load_sh image** Loads the specified SH-based ODF image on the ODF tool.
- **-odf.load_tensor image** Loads the specified tensor image on the ODF tool.
- **-odf.load_dixel image** Loads the specified dixel-based image on the ODF tool.

Vector plot tool options

- **-vector.load image** Load the specified MRtrix sparse image file (.msf) into the fixel tool.

Connectome tool options

- **-connectome.init image** Initialise the connectome tool using a parcellation image.
- **-connectome.load path** Load a matrix file into the connectome tool.

Screen Capture tool options

- **-capture.folder path** Set the output folder for the screen capture tool.
- **-capture.prefix string** Set the output file prefix for the screen capture tool.
- **-capture.grab** Start the screen capture process.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.

- **-version** display version information and exit.

References

Tournier, J.-D.; Calamante, F. & Connelly, A. MRtrix: Diffusion tractography in crossing fiber regions. Int. J. Imaging Syst. Technol., 2012, 22, 53-66

Author: J-Donald Tournier (d.tournier@brain.org.au), Dave Raffelt (david.raffelt@florey.edu.au) and Robert E. Smith (robert.smith@florey.edu.au)

peaks2amp

Synopsis

```
peaks2amp [ options ]  directions amplitudes
```

- *directions*: the input directions image. Each volume corresponds to the x, y & z component of each direction vector in turn.
- *amplitudes*: the output amplitudes image.

Description

convert peak directions image to amplitudes.

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.

- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

sh2amp

Synopsis

```
sh2amp [ options ]  input directions output
```

- *input*: the input image consisting of spherical harmonic (SH) coefficients.
- *directions*: the list of directions along which the SH functions will be sampled, generated using the dirgen command
- *output*: the output image consisting of the amplitude of the SH functions along the specified directions.

Description

Evaluate the amplitude of an image of spherical harmonic functions along the specified directions

Options

- **-gradient** assume input directions are supplied as a gradient encoding file
- **-nonnegative** cap all negative amplitudes to zero

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Data type options

- **-datatype spec** specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

sh2peaks

Synopsis

```
sh2peaks [ options ]  SH output
```

- *SH*: the input image of SH coefficients.
- *output*: the output image. Each volume corresponds to the x, y & z component of each peak direction vector in turn.

Description

extract the peaks of a spherical harmonic function at each voxel, by commencing a Newton search along a set of specified directions

Options

- **-num peaks** the number of peaks to extract (default: 3).
- **-direction phi theta** the direction of a peak to estimate. The algorithm will attempt to find the same number of peaks as have been specified using this option.
- **-peaks image** the program will try to find the peaks that most closely match those in the image provided.

- **-threshold value** only peak amplitudes greater than the threshold will be considered.
- **-seeds file** specify a set of directions from which to start the multiple restarts of the optimisation (by default, the built-in 60 direction set is used)
- **-mask image** only perform computation within the specified binary brain mask image.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

sh2power

Synopsis

```
sh2power [ options ]  SH power
```

- *SH*: the input spherical harmonics coefficients image.
- *power*: the output power image.

Description

compute the total power of a spherical harmonics image.

This command computes the sum of squared SH coefficients, which equals the mean-squared amplitude of the spherical function it represents.

Options

- **-spectrum** output the power spectrum, i.e., the power contained within each harmonic degree (l=0, 2, 4, ...) as a 4-D image.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

sh2response

Synopsis

```
sh2response [ options ]  SH mask directions response
```

- *SH*: the spherical harmonic decomposition of the diffusion-weighted images
- *mask*: the mask containing the voxels from which to estimate the response function
- *directions*: a 4D image containing the direction vectors along which to estimate the response function
- *response*: the output axially-symmetric spherical harmonic coefficients

Description

generate an appropriate response function from the image data for spherical deconvolution

Options

- **-lmax value** specify the maximum harmonic degree of the response function to estimate

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

shbasis

Synopsis

```
shbasis [ options ]  SH [ SH ... ]
```

- *SH*: the input image(s) of SH coefficients.

Description

examine the values in spherical harmonic images to estimate (and optionally change) the SH basis used.

In previous versions of MRtrix, the convention used for storing spherical harmonic coefficients was a non-orthonormal basis (the m!=0 coefficients were a factor of sqrt(2) too large). This error has been rectified in the new MRtrix (assuming that compilation was performed without the USE_NON_ORTHONORMAL_SH_BASIS symbol defined), but will cause issues if processing SH data that was generated using an older version of MRtrix (or vice-versa).

This command provides a mechanism for testing the basis used in storage of image data representing a spherical harmonic series per voxel, and allows the user to forcibly modify the raw image data to conform to the desired basis.

Options

- **-convert mode** convert the image data in-place to the desired basis (if necessary). Options are: old, new, native (whichever basis MRtrix is compiled for; most likely the new orthonormal basis), force_oldtonew, force_newtoold. Note that for the "force_*" choices should ideally only be used in cases where the command is unable to automatically determine the SH basis using the existing image data.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

shconv

Synopsis

```
shconv [ options ]  SH_in response SH_out
```

- *SH_in*: the input spherical harmonics coefficients image.
- *response*: the convolution kernel (response function)
- *SH_out*: the output spherical harmonics coefficients image.

Description

perform a spherical convolution

Options

- **-mask image** only perform computation within the specified binary brain mask image.

Stride options

- **-stride spec** specify the strides of the output data in memory, as a comma-separated list. The actual strides produced will depend on whether the output image format can support it.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

shview

Synopsis

```
shview [ options ] [ coefs ]
```

- *coefs*: a text file containing the even order spherical harmonics coefficients to display.

Description

view spherical harmonics surface plots.

Options

- **-response** assume SH coefficients file only contains m=0 terms (zonal harmonics). Used to display the response function as produced by estimate_response

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

tck2connectome

Synopsis

```
tck2connectome [ options ]  tracks_in nodes_in connectome_out
```

- *tracks_in*: the input track file
- *nodes_in*: the input node parcellation image
- *connectome_out*: the output .csv file containing edge weights

Description

generate a connectome matrix from a streamlines file and a node parcellation image

Options

Structural connectome streamline assignment option

- **-assignment_end_voxels** use a simple voxel lookup value at each streamline endpoint
- **-assignment_radial_search radius** perform a radial search from each streamline endpoint to locate the nearest node.Argument is the maximum radius in mm; if no node is found within this radius, the streamline endpoint is not assigned to any node. Default search distance is 2mm.
- **-assignment_reverse_search max_dist** traverse from each streamline endpoint inwards along the streamline, in search of the last node traversed by the streamline. Argument is the maximum traversal length in mm (set to 0 to allow search to continue to the streamline midpoint).
- **-assignment_forward_search max_dist** project the streamline forwards from the endpoint in search of a parcellation node voxel. Argument is the maximum traversal length in mm.
- **-assignment_all_voxels** assign the streamline to all nodes it intersects along its length (note that this means a streamline may be assigned to more than two nodes, or indeed none at all)

Structural connectome metric options

- **-scale_length** scale each contribution to the connectome edge by the length of the streamline
- **-scale_invlength** scale each contribution to the connectome edge by the inverse of the streamline length
- **-scale_invnodevol** scale each contribution to the connectome edge by the inverse of the two node volumes
- **-scale_file path** scale each contribution to the connectome edge according to the values in a vector file

Other options for tck2connectome

- **-stat_edge statistic** statistic for combining the values from all streamlines in an edge into a single scale value for that edge (options are: sum,mean,min,max; default=sum)
- **-tck_weights_in path** specify a text scalar file containing the streamline weights
- **-keep_unassigned** By default, the program discards the information regarding those streamlines that are not successfully assigned to a node pair. Set this option to keep these values (will be the first row/column in the output matrix)
- **-out_assignments path** output the node assignments of each streamline to a file
- **-zero_diagonal** set all diagonal entries in the matrix to zero (these represent streamlines that connect to the same node at both ends)
- **-vector** output a vector representing connectivities from a given seed point to target nodes, rather than a matrix of node-node connectivities

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.

- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

tckconvert

Synopsis

```
tckconvert [ options ]  input output
```

- *input*: the input track file.
- *output*: the output track file.

Description

Convert between different track file formats.

The program currently supports MRtrix .tck files (input/output), ascii text files (input/output), and VTK polydata files (output only).

Note that ascii files will be stored with one streamline per numbered file. To support this, the command will use the multi-file numbering syntax, where square brackets denote the position of the numbering for the files, for example:

$ tckconvert input.tck output-[].txt

will produce files named output-0000.txt, output-0001.txt, output-0002.txt, ...

Options

- **-scanner2voxel reference** if specified, the properties of this image will be used to convert track point positions from real (scanner) coordinates into voxel coordinates.
- **-scanner2image reference** if specified, the properties of this image will be used to convert track point positions from real (scanner) coordinates into image coordinates (in mm).
- **-voxel2scanner reference** if specified, the properties of this image will be used to convert track point positions from voxel coordinates into real (scanner) coordinates.
- **-image2scanner reference** if specified, the properties of this image will be used to convert track point positions from image coordinates (in mm) into real (scanner) coordinates.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Daan Christiaens (daan.christiaens@gmail.com), J-Donald Tournier (jdtournier@gmail.com), Philip Broser (philip.broser@me.com).

tckedit

Synopsis

```
tckedit [ options ]  tracks_in [ tracks_in ... ] tracks_out
```

- *tracks_in*: the input track file(s)
- *tracks_out*: the output track file

Description

perform various editing operations on track files.

Options

Region Of Interest processing options

- **-include spec** specify an inclusion region of interest, as either a binary mask image, or as a sphere using 4 comma-separared values (x,y,z,radius). Streamlines must traverse ALL inclusion regions to be accepted.
- **-exclude spec** specify an exclusion region of interest, as either a binary mask image, or as a sphere using 4 comma-separared values (x,y,z,radius). Streamlines that enter ANY exclude region will be discarded.

- **-mask spec** specify a masking region of interest, as either a binary mask image, or as a sphere using 4 comma-separared values (x,y,z,radius). If defined, streamlines exiting the mask will be truncated.

Streamline length threshold options

- **-maxlength value** set the maximum length of any streamline in mm
- **-minlength value** set the minimum length of any streamline in mm

Streamline count truncation options

- **-number count** set the desired number of selected streamlines to be propagated to the output file
- **-skip count** omit this number of selected streamlines before commencing writing to the output file

Thresholds pertaining to per-streamline weighting

- **-maxweight value** set the maximum weight of any streamline
- **-minweight value** set the minimum weight of any streamline

Other options specific to tckedit

- **-inverse** output the inverse selection of streamlines based on the criteria provided, i.e. only those streamlines that fail at least one criterion will be written to file.
- **-ends_only** only test the ends of each streamline against the provided include/exclude ROIs

Options for handling streamline weights

- **-tck_weights_in path** specify a text scalar file containing the streamline weights
- **-tck_weights_out path** specify the path for an output text scalar file containing streamline weights

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

tckgen

Synopsis

```
tckgen [ options ]  source tracks
```

- *source*: the image containing the source data. The type of data depends on the algorithm used:- FACT: the directions file (each triplet of volumes is the X,Y,Z direction of a fibre population).- iFOD1/2, Nulldist2 & SD_Stream: the SH image resulting from CSD.- Nulldist1 & SeedTest: any image (will not be used).- Tensor_Det / Tensor_Prob: the DWI image.
- *tracks*: the output file containing the tracks generated.

Description

perform streamlines tractography.

Options

- **-algorithm name** specify the tractography algorithm to use. Valid choices are: FACT, iFOD1, iFOD2, Nulldist1, Nulldist2, SD_Stream, Seedtest, Tensor_Det, Tensor_Prob (default: iFOD2).

Region Of Interest processing options

- **-include spec** specify an inclusion region of interest, as either a binary mask image, or as a sphere using 4 comma-separared values (x,y,z,radius). Streamlines must traverse ALL inclusion regions to be accepted.
- **-exclude spec** specify an exclusion region of interest, as either a binary mask image, or as a sphere using 4 comma-separared values (x,y,z,radius). Streamlines that enter ANY exclude region will be discarded.
- **-mask spec** specify a masking region of interest, as either a binary mask image, or as a sphere using 4 comma-separared values (x,y,z,radius). If defined, streamlines exiting the mask will be truncated.

Streamlines tractography options

- **-step size** set the step size of the algorithm in mm (default is 0.1 x voxelsize; for iFOD2: 0.5 x voxelsize).
- **-angle theta** set the maximum angle between successive steps (default is 90deg x stepsize / voxelsize).

- **-number tracks** set the desired number of tracks. The program will continue to generate tracks until this number of tracks have been selected and written to the output file; set to 0 to ignore limit.
- **-maxnum tracks** set the maximum number of tracks to generate. The program will not generate more tracks than this number, even if the desired number of tracks hasn't yet been reached (default is 100 x number); set to 0 to ignore limit.
- **-maxlength value** set the maximum length of any track in mm (default is 100 x voxelsize).
- **-minlength value** set the minimum length of any track in mm (default is 5 x voxelsize without ACT, 2 x voxelsize with ACT).
- **-cutoff value** set the FA or FOD amplitude cutoff for terminating tracks (default is 0.1).
- **-initcutoff value** set the minimum FA or FOD amplitude for initiating tracks (default is the same as the normal cutoff).
- **-trials number** set the maximum number of sampling trials at each point (only used for probabilistic tracking).
- **-unidirectional** track from the seed point in one direction only (default is to track in both directions).
- **-initdirection dir** specify an initial direction for the tracking (this should be supplied as a vector of 3 comma-separated values.
- **-noprecomputed** do NOT pre-compute legendre polynomial values. Warning: this will slow down the algorithm by a factor of approximately 4.
- **-power value** raise the FOD to the power specified (default is 1/nsamples).
- **-samples number** set the number of FOD samples to take per step for the 2nd order (iFOD2) method (Default: 4).
- **-rk4** use 4th-order Runge-Kutta integration (slower, but eliminates curvature overshoot in 1st-order deterministic methods)
- **-stop** stop propagating a streamline once it has traversed all include regions
- **-downsample factor** downsample the generated streamlines to reduce output file size (default is (samples-1) for iFOD2, no downsampling for all other algorithms)

Anatomically-Constrained Tractography options

- **-act image** use the Anatomically-Constrained Tractography framework during tracking;provided image must be in the 5TT (five-tissue-type) format
- **-backtrack** allow tracks to be truncated and re-tracked if a poor structural termination is encountered
- **-crop_at_gmwmi** crop streamline endpoints more precisely as they cross the GM-WM interface

Tractography seeding options

- **-seed_sphere spec** spherical seed as four comma-separated values (XYZ position and radius)
- **-seed_image image** seed streamlines entirely at random within a mask image (this is the same behaviour as the streamline seeding in MRtrix 0.2)
- **-seed_random_per_voxel image num_per_voxel** seed a fixed number of streamlines per voxel in a mask image; random placement of seeds in each voxel
- **-seed_grid_per_voxel image grid_size** seed a fixed number of streamlines per voxel in a mask image; place seeds on a 3D mesh grid (grid_size argument is per axis; so a grid_size of 3 results in 27 seeds per voxel)

- **-seed_rejection image** seed from an image using rejection sampling (higher values = more probable to seed from)
- **-seed_gmwmi image** seed from the grey matter - white matter interface (only valid if using ACT framework). Input image should be a 3D seeding volume; seeds drawn within this image will be optimised to the interface using the 5TT image provided using the -act option.
- **-seed_dynamic fod_image** determine seed points dynamically using the SIFT model (must not provide any other seeding mechanism). Note that while this seeding mechanism improves the distribution of reconstructed streamlines density, it should NOT be used as a substitute for the SIFT method itself.
- **-max_seed_attempts count** set the maximum number of times that the tracking algorithm should attempt to find an appropriate tracking direction from a given seed point
- **-output_seeds path** output the seed location of all successful streamlines to a file

DW gradient table import options

- **-grad encoding** specify the diffusion-weighted gradient scheme used in the acquisition. The program will normally attempt to use the encoding stored in the image header. This should be supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units of s/mm^2.
- **-fslgrad bvecs bvals** specify the diffusion-weighted gradient scheme used in the acquisition in FSL bvecs/bvals format.
- **-bvalue_scaling mode** specifies whether the b-values should be scaled by the square of the corresponding DW gradient norm, as often required for multi-shell or DSI DW acquisition schemes. The default action can also be set in the MRtrix config file, under the BValueScaling entry. Valid choices are yes/no, true/false, 0/1 (default: true).

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

References based on streamlines algorithm used:

- FACT:Mori, S.; Crain, B. J.; Chacko, V. P. & van Zijl, P. C. M. Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging. Annals of Neurology, 1999, 45, 265-269
- iFOD1 or SD_STREAM:Tournier, J.-D.; Calamante, F. & Connelly, A. MRtrix: Diffusion tractography in crossing fiber regions. Int. J. Imaging Syst. Technol., 2012, 22, 53-66

- iFOD2:Tournier, J.-D.; Calamante, F. & Connelly, A. Improved probabilistic streamlines tractography by 2nd order integration over fibre orientation distributions. Proceedings of the International Society for Magnetic Resonance in Medicine, 2010, 1670
- Nulldist1 / Nulldist2:Morris, D. M.; Embleton, K. V. & Parker, G. J. Probabilistic fibre tracking: Differentiation of connections from chance events. NeuroImage, 2008, 42, 1329-1339
- Tensor_Det:Basser, P. J.; Pajevic, S.; Pierpaoli, C.; Duda, J. & Aldroubi, A. In vivo fiber tractography using DT-MRI data. Magnetic Resonance in Medicine, 2000, 44, 625-632
- Tensor_Prob:Jones, D. Tractography Gone Wild: Probabilistic Fibre Tracking Using the Wild Bootstrap With Diffusion Tensor MRI. IEEE Transactions on Medical Imaging, 2008, 27, 1268-1274

References based on command-line options:

- -rk4:Basser, P. J.; Pajevic, S.; Pierpaoli, C.; Duda, J. & Aldroubi, A. In vivo fiber tractography using DT-MRI data. Magnetic Resonance in Medicine, 2000, 44, 625-632
- -act, -backtrack, -seed_gmwmi:Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. Anatomically-constrained tractography: Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 2012, 62, 1924-1938
- -seed_dynamic:Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. SIFT2: Enabling dense quantitative assessment of brain white matter connectivity using streamlines tractography. NeuroImage, 2015, 119, 338-351

Author: J-Donald Tournier (jdtournier@gmail.com) and Robert E. Smith (robert.smith@florey.edu.au)

tckglobal

Synopsis

```
tckglobal [ options ]  source response tracks
```

- *source*: the image containing the raw DWI data.
- *response*: the response of a track segment on the DWI signal.
- *tracks*: the output file containing the tracks generated.

Description

Multi-Shell Multi-Tissue Global Tractography.

This command will reconstruct the global white matter fibre tractogram that best explains the input DWI data, using a multi-tissue spherical convolution model.

Example use:

$ tckglobal dwi.mif wmr.txt -riso csfr.txt -riso gmr.txt -mask mask.mif -niter 1e8 -fod fod.mif -fiso fiso.mif tracks.tck

in which dwi.mif is the input image, wmr.txt is an anisotropic, multi-shell response function for WM, and csfr.txt and gmr.txt are isotropic response functions for CSF and GM. The output tractogram is saved to tracks.tck. Optional output images fod.mif and fiso.mif contain the predicted WM fODF and isotropic tissue fractions of CSF and GM respectively, estimated as part of the global optimization and thus affected by spatial regularization.

Options

Input options

- **-grad scheme** specify the diffusion encoding scheme (required if not supplied in the header).
- **-mask image** only reconstruct the tractogram within the specified brain mask image.
- **-riso response** set one or more isotropic response functions. (multiple allowed)

Parameters

- **-lmax order** set the maximum harmonic order for the output series. (default = 8)
- **-length size** set the length of the particles (fibre segments). (default = 1mm)
- **-weight w** set the weight by which particles contribute to the model. (default = 0.1)
- **-ppot u** set the particle potential, i.e., the cost of adding one segment, relative to the particle weight. (default = 0.05)
- **-cpot v** set the connection potential, i.e., the energy term that drives two segments together. (default = 0.5)
- **-t0 start** set the initial temperature of the metropolis hastings optimizer. (default = 0.1)
- **-t1 end** set the final temperature of the metropolis hastings optimizer. (default = 0.001)
- **-niter n** set the number of iterations of the metropolis hastings optimizer. (default = 10M)

Output options

- **-fod odf** Predicted fibre orientation distribution function (fODF).This fODF is estimated as part of the global track optimization, and therefore incorporates the spatial regularization that it imposes. Internally, the fODF is represented as a discrete sum of apodized point spread functions (aPSF) oriented along the directions of all particles in the voxel, used to predict the DWI signal from the particle configuration.
- **-noapo** disable spherical convolution of fODF with apodized PSF, to output a sum of delta functions rather than a sum of aPSFs.
- **-fiso iso** Predicted isotropic fractions of the tissues for which response functions were provided with -riso. Typically, these are CSF and GM.
- **-eext eext** Residual external energy in every voxel.
- **-etrend stats** internal and external energy trend and cooling statistics.

Advanced parameters, if you really know what you're doing

- **-balance b** balance internal and external energy. (default = 0)Negative values give more weight to the internal energy, positive to the external energy.
- **-density lambda** set the desired density of the free Poisson process. (default = 1)
- **-prob prob** set the probabilities of generating birth, death, randshift, optshift and connect proposals respectively. (default = 0.25,0.05,0.25,0.1,0.35)
- **-beta b** set the width of the Hanning interpolation window. (in [0, 1], default = 0)If used, a mask is required, and this mask must keep at least one voxel distance to the image bounding box.
- **-lambda lam** set the weight of the internal energy directly. (default = 1)If provided, any value of -balance will be ignored.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Christiaens, D.; Reisert, M.; Dhollander, T.; Sunaert, S.; Suetens, P. & Maes, F. Global tractography of multi-shell diffusion-weighted imaging data using a multi-tissue model. NeuroImage, 2015, 123, 89-101

Author: Daan Christiaens (daan.christiaens@kuleuven.be)

tckinfo

Synopsis

```
tckinfo [ options ]  tracks [ tracks ... ]
```

- *tracks*: the input track file.

Description

print out information about track file

Options

- **-count** count number of tracks in file explicitly, ignoring the header

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

tckmap

Synopsis

```
tckmap [ options ]  tracks output
```

- *tracks*: the input track file.
- *output*: the output track-weighted image

Description

Use track data as a form of contrast for producing a high-resolution image.

Note: if you run into limitations with RAM usage, make sure you output the results to a .mif file or .mih / .dat file pair - this will avoid the allocation of an additional buffer to store the output for write-out.

Options

Options for the header of the output image

- **-template image** an image file to be used as a template for the output (the output image will have the same transform and field of view).
- **-vox size** provide either an isotropic voxel size (in mm), or comma-separated list of 3 voxel dimensions.
- **-datatype spec** specify output image data type.

Options for the dimensionality of the output image

- **-dec** perform track mapping in directionally-encoded colour (DEC) space
- **-dixel path** map streamlines to dixels within each voxel; requires either a number of dixels (references an internal direction set), or a path to a text file containing a set of directions stored as azimuth/elevation pairs
- **-tod lmax** generate a Track Orientation Distribution (TOD) in each voxel; need to specify the maximum spherical harmonic degree lmax to use when generating Apodised Point Spread Functions

Options for the TWI image contrast properties

- **-contrast type** define the desired form of contrast for the output imageOptions are: tdi, length, invlength, scalar_map, scalar_map_count, fod_amp, curvature, vector_file (default: tdi)
- **-image image** provide the scalar image map for generating images with 'scalar_map' / 'scalar_map_count' contrast, or the spherical harmonics image for 'fod_amp' contrast
- **-vector_file path** provide the vector data file for generating images with 'vector_file' contrast
- **-stat_vox type** define the statistic for choosing the final voxel intensities for a given contrast type given the individual values from the tracks passing through each voxel. Options are: sum, min, mean, max (default: sum)
- **-stat_tck type** define the statistic for choosing the contribution to be made by each streamline as a function of the samples taken along their lengths. Only has an effect for 'scalar_map', 'fod_amp' and 'curvature' contrast types. Options are: sum, min, mean, max, median, mean_nonzero, gaussian, ends_min, ends_mean, ends_max, ends_prod (default: mean)
- **-fwhm_tck value** when using gaussian-smoothed per-track statistic, specify the desired full-width half-maximum of the Gaussian smoothing kernel (in mm)
- **-map_zero** if a streamline has zero contribution based on the contrast & statistic, typically it is not mapped; use this option to still contribute to the map even if this is the case (these non-contributing voxels can then influence the mean value in each voxel of the map)

Options for the streamline-to-voxel mapping mechanism

- **-upsample factor** upsample the tracks by some ratio using Hermite interpolation before mappping(If omitted, an appropriate ratio will be determined automatically)
- **-precise** use a more precise streamline mapping strategy, that accurately quantifies the length through each voxel (these lengths are then taken into account during TWI calculation)
- **-ends_only** only map the streamline endpoints to the image
- **-tck_weights_in path** specify a text scalar file containing the streamline weights

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

- For TDI or DEC TDI:Calamante, F.; Tournier, J.-D.; Jackson, G. D. & Connelly, A. Track-density imaging (TDI): Super-resolution white matter imaging using whole-brain track-density mapping. NeuroImage, 2010, 53, 1233-1243
- If using -contrast length and -stat_vox mean:Pannek, K.; Mathias, J. L.; Bigler, E. D.; Brown, G.; Taylor, J. D. & Rose, S. E. The average pathlength map: A diffusion MRI tractography-derived index for studying brain pathology. NeuroImage, 2011, 55, 133-141
- If using -dixel option with TDI contrast only:Smith, R.E., Tournier, J-D., Calamante, F., Connelly, A. A novel paradigm for automated segmentation of very large whole-brain probabilistic tractography data sets. In proc. ISMRM, 2011, 19, 673
- If using -dixel option with any other contrast:Pannek, K., Raffelt, D., Salvado, O., Rose, S. Incorporating directional information in diffusion tractography derived maps: angular track imaging (ATI). In Proc. ISMRM, 2012, 20, 1912
- If using -tod option:Dhollander, T., Emsell, L., Van Hecke, W., Maes, F., Sunaert, S., Suetens, P. Track Orientation Density Imaging (TODI) and Track Orientation Distribution (TOD) based tractography. NeuroImage, 2014, 94, 312-336
- If using other contrasts / statistics:Calamante, F.; Tournier, J.-D.; Smith, R. E. & Connelly, A. A generalised framework for super-resolution track-weighted imaging. NeuroImage, 2012, 59, 2494-2503
- If using -precise mapping option:Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. SIFT: Spherical-deconvolution informed filtering of tractograms. NeuroImage, 2013, 67, 298-312 (Appendix 3)

Author: Robert E. Smith (robert.smith@florey.edu.au) and J-Donald Tournier (jdtournier@gmail.com)

tcknormalise

Synopsis

```
tcknormalise [ options ]  tracks transform output
```

- *tracks*: the input track file.
- *transform*: the image containing the transform.
- *output*: the output track file

Description

apply a normalisation map to a tracks file.

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

tckresample

Synopsis

```
tckresample [ options ]  in_tracks out_tracks
```

- *in_tracks*: the input track file
- *out_tracks*: the output resampled tracks

Description

Resample each streamline to a new set of vertices.

This may be either increasing or decreasing the number of samples along each streamline, or changing the positions of the samples according to some specified trajectory.

Options

Streamline resampling options

- **-upsample ratio** increase the density of points along the length of each streamline by some factor (may improve mapping streamlines to ROIs, and/or visualisation)
- **-downsample ratio** increase the density of points along the length of each streamline by some factor (decreases required storage space)
- **-step_size value** re-sample the streamlines to a desired step size (in mm)
- **-num_points count** re-sample each streamline to a fixed number of points
- **-endpoints** only output the two endpoints of each streamline
- **-line num start end** resample tracks at 'num' equidistant locations along a line between 'start' and 'end' (specified as comma-separated 3-vectors in scanner coordinates)
- **-arc num start mid end** resample tracks at 'num' equidistant locations along a circular arc specified by points 'start', 'mid' and 'end' (specified as comma-separated 3-vectors in scanner coordinates)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au) and J-Donald Tournier (jdtournier@gmail.com)

tcksample

Synopsis

```
tcksample [ options ]  tracks image values
```

- *tracks*: the input track file
- *image*: the image to be sampled
- *values*: the output sampled values

Description

sample values of an associated image along tracks

By default, the value of the underlying image at each point along the track is written to either an ASCII file (with all values for each track on the same line), or a track scalar file (.tsf). Alternatively, some statistic can be taken from the values along each streamline and written to a vector file.

Options

- **-stat_tck statistic** compute some statistic from the values along each streamline (options are: mean,median,min,max)
- **-precise** use the precise mechanism for mapping streamlines to voxels (obviates the need for trilinear interpolation) (only applicable if some per-streamline statistic is requested)
- **-use_tdi_fraction** each streamline is assigned a fraction of the image intensity in each voxel based on the fraction of the track density contributed by that streamline (this is only appropriate for processing a whole-brain tractogram, and images for which the quantiative parameter is additive)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.

- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

tcksift

Synopsis

```
tcksift [ options ]  in_tracks in_fod out_tracks
```

- *in_tracks*: the input track file
- *in_fod*: input image containing the spherical harmonics of the fibre orientation distributions
- *out_tracks*: the output filtered tracks file

Description

filter a whole-brain fibre-tracking data set such that the streamline densities match the FOD lobe integrals.

Options

- **-nofilter** do NOT perform track filtering - just construct the model in order to provide output debugging images
- **-output_at_counts counts** output filtered track files (and optionally debugging images if -output_debug is specified) at specific numbers of remaining streamlines; provide as comma-separated list of integers

Options for setting the processing mask for the SIFT fixel-streamlines comparison model

- **-proc_mask image** provide an image containing the processing mask weights for the model; image spatial dimensions must match the fixel image
- **-act image** use an ACT five-tissue-type segmented anatomical image to derive the processing mask

Options affecting the SIFT model

- **-fd_scale_gm** provide this option (in conjunction with -act) to heuristically downsize the fibre density estimates based on the presence of GM in the voxel. This can assist in reducing tissue interface effects when using a single-tissue deconvolution algorithm
- **-no_dilate_lut** do NOT dilate FOD lobe lookup tables; only map streamlines to FOD lobes if the precise tangent lies within the angular spread of that lobe
- **-make_null_lobes** add an additional FOD lobe to each voxel, with zero integral, that covers all directions with zero / negative FOD amplitudes
- **-remove_untracked** remove FOD lobes that do not have any streamline density attributed to them; this improves filtering slightly, at the expense of longer computation time (and you can no longer do quantitative comparisons between reconstructions if this is enabled)
- **-fd_thresh value** fibre density threshold; exclude an FOD lobe from filtering processing if its integral is less than this amount (streamlines will still be mapped to it, but it will not contribute to the cost function or the filtering)

Options to make SIFT provide additional output files

- **-csv file** output statistics of execution per iteration to a .csv file
- **-output_debug** provide various output images for assessing & debugging performace etc.
- **-out_selection path** output a text file containing the binary selection of streamlines

Options to control when SIFT terminates filtering

- **-term_number value** number of streamlines - continue filtering until this number of streamlines remain
- **-term_ratio value** termination ratio - defined as the ratio between reduction in cost function, and reduction in density of streamlines.Smaller values result in more streamlines being filtered out.
- **-term_mu value** terminate filtering once the SIFT proportionality coefficient reaches a given value

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. SIFT: Spherical-deconvolution informed filtering of tractograms. NeuroImage, 2013, 67, 298-312

Author: Robert E. Smith (robert.smith@florey.edu.au)

tcksift2

Synopsis

```
tcksift2 [ options ]  in_tracks in_fod out_weights
```

- *in_tracks*: the input track file
- *in_fod*: input image containing the spherical harmonics of the fibre orientation distributions
- *out_weights*: output text file containing the weighting factor for each streamline

Description

successor to the SIFT method; instead of removing streamlines, use an EM framework to find an appropriate cross-section multiplier for each streamline

Options

Options for setting the processing mask for the SIFT fixel-streamlines comparison model

- **-proc_mask image** provide an image containing the processing mask weights for the model; image spatial dimensions must match the fixel image
- **-act image** use an ACT five-tissue-type segmented anatomical image to derive the processing mask

Options affecting the SIFT model

- **-fd_scale_gm** provide this option (in conjunction with -act) to heuristically downsize the fibre density estimates based on the presence of GM in the voxel. This can assist in reducing tissue interface effects when using a single-tissue deconvolution algorithm
- **-no_dilate_lut** do NOT dilate FOD lobe lookup tables; only map streamlines to FOD lobes if the precise tangent lies within the angular spread of that lobe

- **-make_null_lobes** add an additional FOD lobe to each voxel, with zero integral, that covers all directions with zero / negative FOD amplitudes
- **-remove_untracked** remove FOD lobes that do not have any streamline density attributed to them; this improves filtering slightly, at the expense of longer computation time (and you can no longer do quantitative comparisons between reconstructions if this is enabled)
- **-fd_thresh value** fibre density threshold; exclude an FOD lobe from filtering processing if its integral is less than this amount (streamlines will still be mapped to it, but it will not contribute to the cost function or the filtering)

Options to make SIFT provide additional output files

- **-csv file** output statistics of execution per iteration to a .csv file
- **-output_debug** provide various output images for assessing & debugging performace etc.
- **-out_coeffs path** output text file containing the weighting coefficient for each streamline

Regularisation options for SIFT2

- **-reg_tikhonov value** provide coefficient for regularising streamline weighting coefficients (Tikhonov regularisation) (default: 0)
- **-reg_tv value** provide coefficient for regularising variance of streamline weighting coefficient to fixels along its length (Total Variation regularisation) (default: 0.1)

Options for controlling the SIFT2 optimisation algorithm

- **-min_td_frac fraction** minimum fraction of the FOD integral reconstructed by streamlines; if the reconstructed streamline density is below this fraction, the fixel is excluded from optimisation (default: 0.1)
- **-min_iters count** minimum number of iterations to run before testing for convergence; this can prevent premature termination at early iterations if the cost function increases slightly (default: 10)
- **-max_iters count** maximum number of iterations to run before terminating program
- **-min_factor factor** minimum weighting factor for an individual streamline; if the factor falls below this number the streamline will be rejected entirely (factor set to zero) (default: 0)
- **-min_coeff coeff** minimum weighting coefficient for an individual streamline; similar to the '-min_factor' option, but using the exponential coefficient basis of the SIFT2 model; these parameters are related as: factor = e^(coeff). Note that the -min_factor and -min_coeff options are mutually exclusive - you can only provide one. (default: -inf)
- **-max_factor factor** maximum weighting factor that can be assigned to any one streamline (default: inf)
- **-max_coeff coeff** maximum weighting coefficient for an individual streamline; similar to the '-max_factor' option, but using the exponential coefficient basis of the SIFT2 model; these parameters are related as: factor = e^(coeff). Note that the -max_factor and -max_coeff options are mutually exclusive - you can only provide one. (default: inf)
- **-max_coeff_step step** maximum change to a streamline's weighting coefficient in a single iteration (default: 1)
- **-min_cf_decrease frac** minimum decrease in the cost function (as a fraction of the initial value) that must occur each iteration for the algorithm to continue (default: 2.5e-005)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. SIFT2: Enabling dense quantitative assessment of brain white matter connectivity using streamlines tractography. NeuroImage, 2015, 119, 338-351

Author: Robert E. Smith (robert.smith@florey.edu.au)

tckstats

Synopsis

```
tckstats [ options ]  tracks_in
```

- *tracks_in*: the input track file

Description

calculate statistics on streamlines length.

Options

- **-histogram path** output a histogram of streamline lengths
- **-dump path** dump the streamlines lengths to a text file
- **-tck_weights_in path** specify a text scalar file containing the streamline weights

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Robert E. Smith (robert.smith@florey.edu.au)

tensor2metric

Synopsis

```
tensor2metric [ options ]  tensor
```

- *tensor*: the input tensor image.

Description

Generate maps of tensor-derived parameters.

Options

- **-adc image** compute the mean apparent diffusion coefficient (ADC) of the diffusion tensor. (sometimes also referred to as the mean diffusivity (MD))
- **-fa image** compute the fractional anisotropy (FA) of the diffusion tensor.
- **-ad image** compute the axial diffusivity (AD) of the diffusion tensor. (equivalent to the principal eigenvalue)
- **-rd image** compute the radial diffusivity (RD) of the diffusion tensor. (equivalent to the mean of the two non-principal eigenvalues)
- **-cl image** compute the linearity metric of the diffusion tensor. (one of the three Westin shape metrics)

- **-cp image** compute the planarity metric of the diffusion tensor. (one of the three Westin shape metrics)
- **-cs image** compute the sphericity metric of the diffusion tensor. (one of the three Westin shape metrics)
- **-value image** compute the selected eigenvalue(s) of the diffusion tensor.
- **-vector image** compute the selected eigenvector(s) of the diffusion tensor.
- **-num sequence** specify the desired eigenvalue/eigenvector(s). Note that several eigenvalues can be specified as a number sequence. For example, '1,3' specifies the principal (1) and minor (3) eigenvalues/eigenvectors (default = 1).
- **-modulate choice** specify how to modulate the magnitude of the eigenvectors. Valid choices are: none, FA, eigval (default = FA).
- **-mask image** only perform computation within the specified binary brain mask image.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

References

Basser, P. J.; Mattiello, J. & Lebihan, D. MR diffusion tensor spectroscopy and imaging. Biophysical Journal, 1994, 66, 259-267

Westin, C. F.; Peled, S.; Gudbjartsson, H.; Kikinis, R. & Jolesz, F. A. Geometrical diffusion measures for MRI from tensor basis analysis. Proc Intl Soc Mag Reson Med, 1997, 5, 1742

Author: Thijs Dhollander (thijs.dhollander@gmail.com) & Ben Jeurissen (ben.jeurissen@uantwerpen.be) & J-Donald Tournier (jdtournier@gmail.com)

transformcalc

Synopsis

```
transformcalc [ options ]  input [ input ... ] operation output
```

- *input*: the input for the specified operation
- *operation*: the operation to perform, one of: invert, half, rigid, header, average, interpolate, decompose.invert: invert the input transformation:matrix_in invert outputhalf: calculate the matrix square root of the input transformation:matrix_in half outputrigid: calculate the rigid transformation of the affine input transformation:matrix_in rigid outputheader: calculate the transformation matrix from an original image and an image with modified header:mov mapmovhdr header outputaverage: calculate the average affine matrix of all input matrices:input ... average outputinterpolate: create interpolated transformation matrix between input (t=0) and input2 (t=1). Based on matrix decomposition with linear interpolation of translation, rotation and stretch described in Shoemake, K., Hill, M., & Duff, T. (1992). Matrix Animation and Polar Decomposition. Matrix, 92, 258-264. doi:10.1.1.56.1336input input2 interpolate outputdecompose: decompose transformation matrix M into translation, rotation and stretch and shear (M = T * R * S). The output is a key-value text file scaling: vector of 3 scaling factors in x, y, z direction, shear: list of shear factors for xy, xz, yz axes, angles: list of Euler angles about static x, y, z axes in radians in the range [0:pi]x[-pi:pi]x[-pi:pi], angle_axis: angle in radians and rotation axis, translation : translation vector along x, y, z axes in mm, R: composed roation matrix (R = rot_x * rot_y * rot_z), S: composed scaling and shear matrix.matrix_in decompose output
- *output*: the output transformation matrix.

Description

This command's function is to process linear transformation matrices.

It allows to perform affine matrix operations or to convert the transformation matrix provided by FSL's flirt command to a format usable in MRtrix

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Max Pietsch (maximilian.pietsch@kcl.ac.uk)

transformconvert

Synopsis

```
transformconvert [ options ]  input [ input ... ] operation output
```

- *input*: the input for the specified operation
- *operation*: the operation to perform, one of:flirt_import, itk_import.flirt_import: Convert a transformation matrix produced by FSL's flirt command into a format usable by MRtrix. You'll need to provide as additional arguments the NIfTI images that were passed to flirt with the -in and -ref options:matrix_in in ref flirt_import outputitk_import: Convert a plain text transformation matrix file produced by ITK's (ANTS, Slicer) affine registration into a format usable by MRtrix.
- *output*: the output transformation matrix.

Description

This command's function is to convert linear transformation matrices.

It allows to convert the transformation matrix provided by FSL's flirt command

and ITK's linear transformation format to a format usable in MRtrix.

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: Max Pietsch (maximilian.pietsch@kcl.ac.uk)

tsfdivide

Synopsis

```
tsfdivide [ options ]  input input output
```

- *input*: the input track scalar file.
- *input*: the input track scalar file.
- *output*: the output track scalar file

Description

an application to divide corresponding values in track scalar files

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

tsfinfo

Synopsis

```
tsfinfo [ options ]  tracks [ tracks ... ]
```

- *tracks*: the input track scalar file.

Description

print out information about track scalar file

Options

- **-count** count number of tracks in file explicitly, ignoring the header
- **-ascii prefix** save values of each track scalar file in individual ascii files, with the specified prefix.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

tsfmult

Synopsis

```
tsfmult [ options ]  input input output
```

- *input*: the input track scalar file.
- *input*: the input track scalar file.
- *output*: the output track scalar file

Description

an application to multiply corresponding values in track scalar files

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

tsfsmooth

Synopsis

```
tsfsmooth [ options ]  input output
```

- *input*: the input track scalar file.
- *output*: the output track scalar file

Description

Gaussian filter a track scalar file

Options

- **-stdev sigma** apply Gaussian smoothing with the specified standard deviation. The standard deviation is defined in units of track points (default: 4)

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

tsfthreshold

Synopsis

```
tsfthreshold [ options ]  input N output
```

- *input*: the input track scalar file.
- *N*: the desired threshold
- *output*: the binary output track scalar file

Description

an application to threshold and invert track scalar files

Options

- **-invert** invert the output mask

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

voxel2fixel

Synopsis

```
voxel2fixel [ options ]  image_in fixel_in fixel_out
```

- *image_in*: the input image.
- *fixel_in*: the input fixel image.
- *fixel_out*: the output fixel image.

Description

map the scalar value in each voxel to all fixels within that voxel. This could be used to enable CFE-based statistical analysis to be performed on voxel-wise measures

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

warp2metric

Synopsis

```
warp2metric [ options ]  in
```

- *in*: the input deformation field

Description

compute fixel or voxel-wise metrics from a 4D deformation field

Options

- **-fc template_input output** use an input template fixel image to define fibre orientations and output a fixel image describing the change in fibre cross-section (FC) in the perpendicular plane to the fixel orientation
- **-jmat output** output a Jacobian matrix image stored in column-major order along the 4th dimension.Note the output jacobian describes the warp gradient w.r.t the scanner space coordinate system
- **-jdet output** output the Jacobian determinant instead of the full matrix

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

warpconvert

Synopsis

```
warpconvert [ options ]  in out
```

- *in*: the input warp image.
- *out*: the output warp image.

Description

convert between different representations of a non-linear warp. A deformation field is defined as an image where each voxel defines the corresponding position in the other image (in scanner space coordinates). A displacement field stores the displacements (in mm) to the other image from the each voxel's position (in scanner space). The warpfull file is the 5D format output from mrregister -nl_warp_full, which contains linear transforms, warps and their inverses that map each image to a midway space.

Options

- **-type choice** the conversion type required. Valid choices are: deformation2displacement, displacement2deformation, warpfull2deformation, warpfull2displacement (Default: deformation2displacement)
- **-template image** define a template image when converting a warpfull file (which is defined on a grid in the midway space between image 1 & 2). For example to generate the deformation field that maps image1 to image2, then supply image2 as the template image
- **-midway_space** to be used only with warpfull2deformation and warpfull2displacement conversion types. The output will only contain the non-linear warp to map an input image to the midway space (defined by the warpfull grid). If a linear transform exists in the warpfull file header then it will be composed and included in the output.
- **-from image** to be used only with warpfull2deformation and warpfull2displacement conversion types. Used to define the direction of the desired output field.Use -from 1 to obtain the image1->image2 field and from 2 for image2->image1. Can be used in combination with the -midway_space option to produce a field that only maps to midway space.

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

warpcorrect

Synopsis

```
warpcorrect [ options ]  in out
```

- *in*: the input warp image.
- *out*: the output warp image.

Description

replaces voxels in a deformation field that point to 0,0,0 with nan,nan,nan. This can be used in conjunction with the warpinit command to compute a MRtrix compatible deformation field from non-linear transformations generated by any other registration package.

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: David Raffelt (david.raffelt@florey.edu.au)

warpinit

Synopsis

```
warpinit [ options ]  template warp
```

- *template*: the input template image.
- *warp*: the output warp image.

Description

create an initial warp image, representing an identity transformation. This is useful to obtain the warp fields from other normalisation applications, by applying the transformation of interest to the warp field generated by this program.

The image generated is a 4D image with the same spatial characteristics as the input template image. It contains 3 volumes, with each voxel containing its own x,y,z coordinates.

Note that this command can be used to create 3 separate X,Y,Z images directly (which may be useful to create images suitable for use in the registration program) using the following syntax:

$ warpinit template.mif warp-[].nii

Options

Standard options

- **-info** display information messages.
- **-quiet** do not display information messages or progress status.
- **-debug** display debugging messages.
- **-force** force overwrite of output files. Caution: Using the same file as input and output might cause unexpected behaviour.
- **-nthreads number** use this number of threads in multi-threaded applications (set to 0 to disable multi-threading)
- **-failonwarn** terminate program if a warning is produced
- **-help** display this information page and exit.
- **-version** display version information and exit.

Author: J-Donald Tournier (jdtournier@gmail.com)

CHAPTER 32

Python scripts provided with MRtrix3

5ttgen freesurfer

Synopsis

```
5ttgen freesurfer [ options ] input output
```

- *input*: The input FreeSurfer parcellation image (any image containing 'aseg' in its name)
- *output*: The output 5TT image

Description

Generate the 5TT image based on a FreeSurfer parcellation image

Options

Options specific to the 'freesurfer' algorithm

- **-lut** Manually provide path to the lookup table on which the input parcellation image is based (e.g. FreeSurferColorLUT.txt)

Options common to all 5ttgen algorithms

- **-nocrop** Do NOT crop the resulting 5TT image to reduce its size (keep the same dimensions as the input image)
- **-sgm_amyg_hipp** Represent the amygdalae and hippocampi as sub-cortical grey matter in the 5TT image

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. Anatomically-constrained tractography: Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 2012, 62, 1924-1938
- If using 'fsl' algorithm: Smith, S. M. Fast robust automated brain extraction. Human Brain Mapping, 2002, 17, 143-155
- If using 'fsl' algorithm: Zhang, Y.; Brady, M. & Smith, S. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE Transactions on Medical Imaging, 2001, 20, 45-57
- If using 'fsl' algorithm: Patenaude, B.; Smith, S. M.; Kennedy, D. N. & Jenkinson, M. A Bayesian model of shape and appearance for subcortical brain segmentation. NeuroImage, 2011, 56, 907-922
- If using 'fsl' algorithm: Smith, S. M.; Jenkinson, M.; Woolrich, M. W.; Beckmann, C. F.; Behrens, T. E.; Johansen-Berg, H.; Bannister, P. R.; De Luca, M.; Drobnjak, I.; Flitney, D. E.; Niazy, R. K.; Saunders, J.; Vickers, J.; Zhang, Y.; De Stefano, N.; Brady, J. M. & Matthews, P. M. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage, 2004, 23, S208-S219

Author: Robert E. Smith (robert.smith@florey.edu.au)

5ttgen fsl

Synopsis

```
5ttgen fsl [ options ] input output
```

- *input*: The input T1-weighted image
- *output*: The output 5TT image

Description

Use FSL commands to generate the 5TT image based on a T1-weighted image

Options

Options specific to the 'fsl' algorithm

- **-t2 <T2 image>** Provide a T2-weighted image in addition to the default T1-weighted image; this will be used as a second input to FSL FAST
- **-mask** Manually provide a brain mask, rather than deriving one in the script
- **-premasked** Indicate that brain masking has already been applied to the input image

Options common to all 5ttgen algorithms

- **-nocrop** Do NOT crop the resulting 5TT image to reduce its size (keep the same dimensions as the input image)
- **-sgm_amyg_hipp** Represent the amygdalae and hippocampi as sub-cortical grey matter in the 5TT image

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. Anatomically-constrained tractography: Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 2012, 62, 1924-1938
- If using 'fsl' algorithm: Smith, S. M. Fast robust automated brain extraction. Human Brain Mapping, 2002, 17, 143-155
- If using 'fsl' algorithm: Zhang, Y.; Brady, M. & Smith, S. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE Transactions on Medical Imaging, 2001, 20, 45-57
- If using 'fsl' algorithm: Patenaude, B.; Smith, S. M.; Kennedy, D. N. & Jenkinson, M. A Bayesian model of shape and appearance for subcortical brain segmentation. NeuroImage, 2011, 56, 907-922
- If using 'fsl' algorithm: Smith, S. M.; Jenkinson, M.; Woolrich, M. W.; Beckmann, C. F.; Behrens, T. E.; Johansen-Berg, H.; Bannister, P. R.; De Luca, M.; Drobnjak, I.; Flitney, D. E.; Niazy, R. K.; Saunders, J.; Vickers, J.; Zhang, Y.; De Stefano, N.; Brady, J. M. & Matthews, P. M. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage, 2004, 23, S208-S219

Author: Robert E. Smith (robert.smith@florey.edu.au)

5ttgen

Synopsis

```
5ttgen [ options ] algorithm ...
```

- *algorithm*: Select the software / algorithm to be used to derive the 5TT image; additional details and options become available once an algorithm is nominated. Options are: freesurfer, fsl

Description

Generate a 5TT image suitable for ACT

Options

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file

- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. Anatomically-constrained tractography: Improved diffusion MRI streamlines tractography through effective use of anatomical information. NeuroImage, 2012, 62, 1924-1938
- If using 'fsl' algorithm: Smith, S. M. Fast robust automated brain extraction. Human Brain Mapping, 2002, 17, 143-155
- If using 'fsl' algorithm: Zhang, Y.; Brady, M. & Smith, S. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE Transactions on Medical Imaging, 2001, 20, 45-57
- If using 'fsl' algorithm: Patenaude, B.; Smith, S. M.; Kennedy, D. N. & Jenkinson, M. A Bayesian model of shape and appearance for subcortical brain segmentation. NeuroImage, 2011, 56, 907-922
- If using 'fsl' algorithm: Smith, S. M.; Jenkinson, M.; Woolrich, M. W.; Beckmann, C. F.; Behrens, T. E.; Johansen-Berg, H.; Bannister, P. R.; De Luca, M.; Drobnjak, I.; Flitney, D. E.; Niazy, R. K.; Saunders, J.; Vickers, J.; Zhang, Y.; De Stefano, N.; Brady, J. M. & Matthews, P. M. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage, 2004, 23, S208-S219

Author: Robert E. Smith (robert.smith@florey.edu.au)

dwi2response dhollander

Synopsis

```
dwi2response dhollander [ options ] input out_sfwm out_gm out_csf
```

- *input*: The input DWI
- *out_sfwm*: Output single-fibre WM response text file

- *out_gm*: Output GM response text file
- *out_csf*: Output CSF response text file

Description

Unsupervised estimation of WM, GM and CSF response functions. Does not require a T1 image (or segmentation thereof).

Options

Options specific to the 'dhollander' algorithm

- **-erode** Number of erosion passes to apply to initial (whole brain) mask. (default: 3)
- **-fa** FA threshold for crude WM versus GM-CSF separation. (default: 0.2)
- **-sfwm** Number of single-fibre WM voxels to select, as a percentage of refined WM. (default: 0.5 per cent)
- **-gm** Number of GM voxels to select, as a percentage of refined GM. (default: 2 per cent)
- **-csf** Number of CSF voxels to select, as a percentage of refined CSF. (default: 10 per cent)

Options common to all dwi2response algorithms

- **-shell** The b-value shell(s) to use in response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-lmax** The maximum harmonic degree(s) of response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-mask** Provide an initial mask for response voxel selection
- **-voxels** Output an image showing the final voxel selection(s)
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5
- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185
- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80
- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwi2response fa

Synopsis

```
dwi2response fa [ options ] input output
```

- *input*: The input DWI
- *output*: The output response function text file

Description

Use the old FA-threshold heuristic for single-fibre voxel selection and response function estimation

Options

Options specific to the 'fa' algorithm

- **-erode** Number of brain mask erosion steps to apply prior to threshold (not used if mask is provided manually)

- **-number** The number of highest-FA voxels to use
- **-threshold** Apply a hard FA threshold, rather than selecting the top voxels

Options common to all dwi2response algorithms

- **-shell** The b-value shell(s) to use in response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-lmax** The maximum harmonic degree(s) of response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-mask** Provide an initial mask for response voxel selection
- **-voxels** Output an image showing the final voxel selection(s)
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5
- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185
- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80

- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwi2response manual

Synopsis

```
dwi2response manual [ options ] input in_voxels output
```

- *input*: The input DWI
- *in_voxels*: Input voxel selection mask
- *output*: Output response function text file

Description

Derive a response function using an input mask image alone (i.e. pre-selected voxels)

Options

Options specific to the 'manual' algorithm

- **-dirs** Manually provide the fibre direction in each voxel (a tensor fit will be used otherwise)

Options common to all dwi2response algorithms

- **-shell** The b-value shell(s) to use in response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-lmax** The maximum harmonic degree(s) of response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-mask** Provide an initial mask for response voxel selection
- **-voxels** Output an image showing the final voxel selection(s)
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5
- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185
- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80
- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwi2response msmt_5tt

Synopsis

```
dwi2response msmt_5tt [ options ] input in_5tt out_wm out_gm out_csf
```

- *input*: The input DWI
- *in_5tt*: Input co-registered 5TT image
- *out_wm*: Output WM response text file
- *out_gm*: Output GM response text file
- *out_csf*: Output CSF response text file

Description

Derive MSMT-CSD tissue response functions based on a co-registered five-tissue-type (5TT) image

Options

Options specific to the 'msmt_5tt' algorithm

- **-dirs** Manually provide the fibre direction in each voxel (a tensor fit will be used otherwise)
- **-fa** Upper fractional anisotropy threshold for GM and CSF voxel selection
- **-pvf** Partial volume fraction threshold for tissue voxel selection
- **-wm_algo algorithm** dwi2response algorithm to use for WM single-fibre voxel selection

Options common to all dwi2response algorithms

- **-shell** The b-value shell(s) to use in response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-lmax** The maximum harmonic degree(s) of response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-mask** Provide an initial mask for response voxel selection
- **-voxels** Output an image showing the final voxel selection(s)
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script

- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5
- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185
- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80
- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwi2response tax

Synopsis

```
dwi2response tax [ options ] input output
```

- *input*: The input DWI
- *output*: The output response function text file

Description

Use the Tax et al. (2014) recursive calibration algorithm for single-fibre voxel selection and response function estimation

Options

Options specific to the 'tax' algorithm

- **-peak_ratio** Second-to-first-peak amplitude ratio threshold
- **-max_iters** Maximum number of iterations
- **-convergence** Percentile change in any RF coefficient required to continue iterating

Options common to all dwi2response algorithms

- **-shell** The b-value shell(s) to use in response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-lmax** The maximum harmonic degree(s) of response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-mask** Provide an initial mask for response voxel selection
- **-voxels** Output an image showing the final voxel selection(s)
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5

- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185
- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80
- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwi2response tournier

Synopsis

```
dwi2response tournier [ options ] input output
```

- *input*: The input DWI
- *output*: The output response function text file

Description

Use the Tournier et al. (2013) iterative algorithm for single-fibre voxel selection and response function estimation

Options

Options specific to the 'tournier' algorithm

- **-iter_voxels** Number of single-fibre voxels to select when preparing for the next iteration
- **-sf_voxels** Number of single-fibre voxels to use when calculating response function
- **-dilate** Number of mask dilation steps to apply when deriving voxel mask to test in the next iteration
- **-max_iters** Maximum number of iterations

Options common to all dwi2response algorithms

- **-shell** The b-value shell(s) to use in response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-lmax** The maximum harmonic degree(s) of response function estimation (single value for single-shell response, comma-separated list for multi-shell response)
- **-mask** Provide an initial mask for response voxel selection
- **-voxels** Output an image showing the final voxel selection(s)
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5
- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185
- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80
- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwi2response

Synopsis

```
dwi2response [ options ] algorithm ...
```

- *algorithm*: Select the algorithm to be used to derive the response function; additional details and options become available once an algorithm is nominated. Options are: dhollander, fa, manual, msmt_5tt, tax, tournier

Description

Estimate response function(s) for spherical deconvolution

Options

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using 'dhollander' algorithm: Dhollander, T.; Raffelt, D. & Connelly, A. Unsupervised 3-tissue response function estimation from single-shell or multi-shell diffusion MR data without a co-registered T1 image. ISMRM Workshop on Breaking the Barriers of Diffusion MRI, 2016, 5
- If using 'fa' algorithm: Tournier, J.-D.; Calamante, F.; Gadian, D. G. & Connelly, A. Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution. NeuroImage, 2004, 23, 1176-1185

- If using 'msmt_csd' algorithm: Jeurissen, B.; Tournier, J.-D.; Dhollander, T.; Connelly, A. & Sijbers, J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage, 2014, 103, 411-426
- If using 'tax' algorithm: Tax, C. M.; Jeurissen, B.; Vos, S. B.; Viergever, M. A. & Leemans, A. Recursive calibration of the fiber response function for spherical deconvolution of diffusion MRI data. NeuroImage, 2014, 86, 67-80
- If using 'tournier' algorithm: Tournier, J.-D.; Calamante, F. & Connelly, A. Determination of the appropriate b value and number of gradient directions for high-angular-resolution diffusion-weighted imaging. NMR Biomedicine, 2013, 26, 1775-1786

Author: Robert E. Smith (robert.smith@florey.edu.au) and Thijs Dhollander (thijs.dhollander@gmail.com)

dwibiascorrect

Synopsis

```
dwibiascorrect [ options ] input output
```

- *input*: The input image series to be corrected
- *output*: The output corrected image series

Description

Perform B1 field inhomogeneity correction for a DWI volume series

Options

Options for the dwibiascorrect script

- **-mask** Manually provide a mask image for bias field estimation
- **-bias** Output the estimated bias field
- **-ants** Use ANTS N4 to estimate the inhomogeneity field
- **-fsl** Use FSL FAST to estimate the inhomogeneity field
- **-grad** Pass the diffusion gradient table in MRtrix format
- **-fslgrad bvecs bvals** Pass the diffusion gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- If using -fast option: Zhang, Y.; Brady, M. & Smith, S. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE Transactions on Medical Imaging, 2001, 20, 45-57
- If using -fast option: Smith, S. M.; Jenkinson, M.; Woolrich, M. W.; Beckmann, C. F.; Behrens, T. E.; Johansen-Berg, H.; Bannister, P. R.; De Luca, M.; Drobnjak, I.; Flitney, D. E.; Niazy, R. K.; Saunders, J.; Vickers, J.; Zhang, Y.; De Stefano, N.; Brady, J. M. & Matthews, P. M. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage, 2004, 23, S208-S219
- If using -ants option: Tustison, N.; Avants, B.; Cook, P.; Zheng, Y.; Egan, A.; Yushkevich, P. & Gee, J. N4ITK: Improved N3 Bias Correction. IEEE Transactions on Medical Imaging, 2010, 29, 1310-1320

Author: Robert E. Smith (robert.smith@florey.edu.au)

dwiintensitynorm

Synopsis

```
dwiintensitynorm [ options ] input_dir mask_dir output_dir fa_template wm_mask
```

- *input_dir*: The input directory containing all DWI images
- *mask_dir*: Input directory containing brain masks, corresponding to one per input image (with the same file name prefix)

- *output_dir*: The output directory containing all of the intensity normalised DWI images
- *fa_template*: The output population specific FA template, which is threshold to estimate a white matter mask
- *wm_mask*: The output white matter mask (in template space), used to estimate the median b=0 white matter value for normalisation

Description

Performs a global DWI intensity normalisation on a group of subjects using the median b=0 white matter value as the reference. The white matter mask is estimated from a population average FA template then warped back to each subject to perform the intensity normalisation. Note that bias field correction should be performed prior to this step.

Options

Options for the dwiintensitynorm script

- **-fa_threshold** The threshold applied to the Fractional Anisotropy group template used to derive an approximate white matter mask

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

---

Author: David Raffelt (david.raffelt@florey.edu.au)

dwipreproc

Synopsis

```
dwipreproc [ options ] pe_dir input output
```

- *pe_dir*: The phase encode direction; can be a signed axis number (e.g. -0, 1, +2) or a code (e.g. AP, LR, IS)
- *input*: The input DWI series to be corrected
- *output*: The output corrected image series

Description

Perform diffusion image pre-processing using FSL's eddy tool; including inhomogeneity distortion correction using FSL's topup tool if possible

Options

Options for passing reversed phase-encode data; one of these options MUST be provided

- **-rpe_none** Specify explicitly that no reversed phase-encoding image data is provided; eddy will perform eddy current and motion correction only
- **-rpe_pair forward reverse** Provide a pair of images to use for inhomogeneity field estimation; note that the FIRST of these two images must have the same phase-encode direction as the input DWIs
- **-rpe_all input_revpe** Provide a second DWI series identical to the input series, that has the opposite phase encoding; these will be combined in the output image

Options for the dwipreproc script

- **-cuda** Use the CUDA version of eddy
- **-grad** Provide a gradient table in MRtrix format
- **-fslgrad bvecs bvals** Provide a gradient table in FSL bvecs/bvals format
- **-export_grad_mrtrix grad** Export the final gradient table in MRtrix format
- **-export_grad_fsl bvecs bvals** Export the final gradient table in FSL bvecs/bvals format

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)

- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- Andersson, J. L. & Sotiropoulos, S. N. An integrated approach to correction for off-resonance effects and subject movement in diffusion MR imaging. NeuroImage, 2015, 125, 1063-1078
- Smith, S. M.; Jenkinson, M.; Woolrich, M. W.; Beckmann, C. F.; Behrens, T. E.; Johansen-Berg, H.; Bannister, P. R.; De Luca, M.; Drobnjak, I.; Flitney, D. E.; Niazy, R. K.; Saunders, J.; Vickers, J.; Zhang, Y.; De Stefano, N.; Brady, J. M. & Matthews, P. M. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage, 2004, 23, S208-S219
- If using -rpe_all option: Skare, S. & Bammer, R. Jacobian weighting of distortion corrected EPI data. Proceedings of the International Society for Magnetic Resonance in Medicine, 2010, 5063
- If using -rpe_pair or -rpe_all options: Andersson, J. L.; Skare, S. & Ashburner, J. How to correct susceptibility distortions in spin-echo echo-planar images: application to diffusion tensor imaging. NeuroImage, 2003, 20, 870-888

Author: Robert E. Smith (robert.smith@florey.edu.au)

labelsgmfix

Synopsis

```
labelsgmfix [ options ] parc t1 lut output
```

- *parc*: The input FreeSurfer parcellation image
- *t1*: The T1 image to be provided to FIRST
- *lut*: The lookup table file that the parcellated image is based on
- *output*: The output parcellation image

Description

In a FreeSurfer parcellation image, replace the sub-cortical grey matter structure delineations using FSL FIRST

Options

Options for the labelsgmfix script

- **-premasked** Indicate that brain masking has been applied to the T1 input image
- **-sgm_amyg_hipp** Consider the amygdalae and hippocampi as sub-cortical grey matter structures, and also replace their estimates with those from FIRST

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

References

- Patenaude, B.; Smith, S. M.; Kennedy, D. N. & Jenkinson, M. A Bayesian model of shape and appearance for subcortical brain segmentation. NeuroImage, 2011, 56, 907-922
- Smith, S. M.; Jenkinson, M.; Woolrich, M. W.; Beckmann, C. F.; Behrens, T. E.; Johansen-Berg, H.; Bannister, P. R.; De Luca, M.; Drobnjak, I.; Flitney, D. E.; Niazy, R. K.; Saunders, J.; Vickers, J.; Zhang, Y.; De Stefano, N.; Brady, J. M. & Matthews, P. M. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage, 2004, 23, S208-S219
- Smith, R. E.; Tournier, J.-D.; Calamante, F. & Connelly, A. The effects of SIFT on the reproducibility and biological accuracy of the structural connectome. NeuroImage, 2015, 104, 253-265

Author: Robert E. Smith (robert.smith@florey.edu.au)

population_template

Synopsis

```
population_template [ options ] input_dir template
```

- *input_dir*: The input directory containing all images used to build the template
- *template*: The output template image

Description

Generates an unbiased group-average template from a series of images. First a template is optimised with linear registration (rigid or affine, affine is default), then non-linear registration is used to optimise the template further.

Options

Options for the population_template script

- **-mask_dir** Optionally input a set of masks inside a single directory, one per input image (with the same file name prefix). Using masks will speed up registration significantly
- **-warp_dir** Output a directory containing warps from each input to the template. If the folder does not exist it will be created
- **-transformed_dir** Output a directory containing the input images transformed to the template. If the folder does not exist it will be created
- **-linear_transformations_dir** Output a directory containing the linear transformations used to generate the template. If the folder does not exist it will be created
- **-template_mask** Output a template mask. Only works in -mask_dir has been input. The template mask is computed as the intersection of all subject masks in template space.
- **-rigid** perform rigid registration instead of affine. This should be used for intra-subject registration in longitudinal analysis
- **-linear_no_pause** Do not pause the script if a linear registration seems implausible
- **-linear_scale** Specifiy the multi-resolution pyramid used to build the rigid or affine template, in the form of a list of scale factors (default: 0.3,0.4,0.6,0.8,1.0,1.0). This implicitly defines the number of template levels
- **-linear_lmax** Specifiy the lmax used for rigid or affine registration for each scale factor, in the form of a list of integers (default: 2,2,2,4,4,4). The list must be the same length as the linear_scale factor list
- **-linear_niter** Specifiy the number of registration iterations used within each level before updating the template, in the form of a list of integers (default:500 for each scale). The must be a single number or a list of same length as the linear_scale factor list
- **-linear_estimator** Choose estimator for intensity difference metric. Valid choices are: l1 (least absolute: |x|), l2 (ordinary least squares), lp (least powers: |x|^1.2), Default: l2
- **-nl_scale** Specifiy the multi-resolution pyramid used to build the non-linear template, in the form of a list of scale factors (default: 0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0). This implicitly defines the number of template levels

- **-nl_lmax** Specifiy the lmax used for non-linear registration for each scale factor, in the form of a list of integers (default: 2,2,2,2,2,2,2,4,4,4,4,4,4,4). The list must be the same length as the nl_scale factor list
- **-nl_niter** Specifiy the number of registration iterations used within each level before updating the template, in the form of a list of integers (default: 5,5,5,5,5,5,5,5,5,5,5,5,5,5,5). The list must be the same length as the nl_scale factor list
- **-nl_update_smooth** Regularise the gradient update field with Gaussian smoothing (standard deviation in voxel units, Default 2.0 x voxel_size)
- **-nl_disp_smooth** Regularise the displacement field with Gaussian smoothing (standard deviation in voxel units, Default 1.0 x voxel_size)
- **-nl_grad_step** The gradient step size for non-linear registration (Default: 0.5)
- **-noreorientation** Turn off FOD reorientation in mrregister. Reorientation is on by default if the number of volumes in the 4th dimension corresponds to the number of coefficients in an antipodally symmetric spherical harmonic series (i.e. 6, 15, 28, 45, 66 etc
- **-initial_alignment** Method of alignment to form the initial template. Options are "mass" (default), "geometric" and "none".

Standard options

- **-continue <TempDir> <LastFile>** Continue the script from a previous execution; must provide the temporary directory path, and the name of the last successfully-generated file
- **-force** Force overwrite of output files if pre-existing
- **-help** Display help information for the script
- **-nocleanup** Do not delete temporary files during script, or temporary directory at script completion
- **-nthreads number** Use this number of threads in MRtrix multi-threaded applications (0 disables multi-threading)
- **-tempdir /path/to/tmp/** Manually specify the path in which to generate the temporary directory
- **-quiet** Suppress all console output during script execution
- **-verbose** Display additional information for every command invoked

---

Author: David Raffelt (david.raffelt@florey.edu.au) & Max Pietsch (maximilian.pietsch@kcl.ac.uk) & Thijs Dhollander thijs.dhollander@gmail.com)

CHAPTER 33

List of MRtrix3 configuration file options

- **AmbientIntensity** *default: 0.6*

  The default intensity for the ambient light in OpenGL renders.

- **AnalyseLeftToRight** *default: 0 (false)*

  A boolean value to indicate whether images in Analyse format should be assumed to be in LAS orientation (default) or RAS (when this is option is turned on).

- **BValueScaling** *default: 1 (true)*

  Specifies whether the b-values should be scaled by the squared norm of the gradient vectors when loading a DW gradient scheme. This is commonly required to correctly interpret images acquired on scanners that nominally only allow a single b-value, as the common workaround is to scale the gradient vectors to modulate the actual b-value.

- **BZeroThreshold** *default: 10.0*

  Specifies the b-value threshold for determining those image volumes that correspond to b=0.

- **BackgroundColor** *default: 1,1,1 (white)*

  The default colour to use for the background in OpenGL panels, notably the SH viewer.

- **ConnectomeEdgeAssociatedAlphaMultiplier** *default: 1.0*

  The multiplicative factor to apply to the transparency of edges connected to one selected node.

- **ConnectomeEdgeAssociatedColour** *default: 0.0,0.0,0.0*

  The colour mixed in to edges connected to one currently selected node.

- **ConnectomeEdgeAssociatedColourFade** *default: 0.5*

  The fraction of the colour of an edge connected to one selected node determined by the fixed colour.

- **ConnectomeEdgeAssociatedSizeMultiplier** *default: 1.0*

  The multiplicative factor to apply to the size of edges connected to one selected node.

- **ConnectomeEdgeOtherAlphaMultiplier** *default: 1.0*

  The multiplicative factor to apply to the transparency of edges not connected to any selected node.

- **ConnectomeEdgeOtherColour** *default: 0.0,0.0,0.0*

  The colour mixed in to edges not connected to any currently selected node.

- **ConnectomeEdgeOtherColourFade** *default: 0.75*

  The fraction of the colour of an edge not connected to any selected node determined by the fixed colour.

- **ConnectomeEdgeOtherSizeMultiplier** *default: 1.0*

  The multiplicative factor to apply to the size of edges not connected to any selected node.

- **ConnectomeEdgeOtherVisibilityOverride** *default: true*

  Whether or not to force invisibility of edges not connected to any selected node.

- **ConnectomeEdgeSelectedAlphaMultiplier** *default: 1.0*

  The multiplicative factor to apply to the transparency of edges connected to two selected nodes.

- **ConnectomeEdgeSelectedColour** *default: 0.9,0.9,1.0*

  The colour used to highlight the edges connected to two currently selected nodes.

- **ConnectomeEdgeSelectedColourFade** *default: 0.5*

  The fraction of the colour of an edge connected to two selected nodes determined by the fixed selection highlight colour.

- **ConnectomeEdgeSelectedSizeMultiplier** *default: 1.0*

  The multiplicative factor to apply to the size of edges connected to two selected nodes.

- **ConnectomeEdgeSelectedVisibilityOverride** *default: false*

  Whether or not to force visibility of edges connected to two selected nodes.

- **ConnectomeNodeAssociatedAlphaMultiplier** *default: 1.0*

  The multiplicative factor to apply to the transparency of nodes associated with a selected node.

- **ConnectomeNodeAssociatedColour** *default: 0.0,0.0,0.0*

  The colour mixed in to those nodes associated with any selected node.

- **ConnectomeNodeAssociatedColourFade** *default: 0.5*

  The fraction of the colour of an associated node determined by the fixed associated highlight colour.

- **ConnectomeNodeAssociatedSizeMultiplier** *default: 1.0*

  The multiplicative factor to apply to the size of nodes associated with a selected node.

- **ConnectomeNodeOtherAlphaMultiplier** *default: 1.0*

  The multiplicative factor to apply to the transparency of nodes not currently selected nor associated with a selected node.

- **ConnectomeNodeOtherColour** *default: 0.0,0.0,0.0*

  The colour mixed in to those nodes currently not selected nor associated with any selected node.

- **ConnectomeNodeOtherColourFade** *default: 0.75*

The fraction of the colour of an unselected, non-associated node determined by the fixed not-selected highlight colour.

- **ConnectomeNodeOtherSizeMultiplier** *default: 1.0*

  The multiplicative factor to apply to the size of nodes not currently selected nor associated with a selected node.

- **ConnectomeNodeOtherVisibilityOverride** *default: false*

  Whether or not nodes are forced to be invisible when not selected or associated with any selected node.

- **ConnectomeNodeSelectedAlphaMultiplier** *default: 1.0*

  The multiplicative factor to apply to the transparency of selected nodes.

- **ConnectomeNodeSelectedColour** *default: 1.0,1.0,1.0*

  The colour used to highlight those nodes currently selected.

- **ConnectomeNodeSelectedColourFade** *default: 0.75*

  The fraction of the colour of a selected node determined by the fixed selection highlight colour.

- **ConnectomeNodeSelectedSizeMultiplier** *default: 1.0*

  The multiplicative factor to apply to the size of selected nodes.

- **ConnectomeNodeSelectedVisibilityOverride** *default: true*

  Whether or not nodes are forced to be visible when selected.

- **DiffuseIntensity** *default: 0.3*

  The default intensity for the diffuse light in OpenGL renders.

- **FailOnWarn** *default: 0 (false)*

  A boolean value specifying whether MRtrix applications should abort as soon as any (otherwise non-fatal) warning is issued.

- **HelpCommand** *default: less*

  The command to use to display each command's help page (leave empty to send directly to the terminal).

- **IconSize** *default: 24*

  The size of the icons in the main MRView toolbar.

- **ImageInterpolation** *default: true*

  Define default interplation setting for image and image overlay.

- **ImageInterpolation** *default: true*

  Interpolation switched on in the main image

- **InitialToolBarPosition** *default: top*

  The starting position of the MRView toolbar. Valid values are: top, bottom, left, right.

- **LightPosition** *default: 1,1,3*

  The default position vector to use for the light in OpenGL renders.

- **MRViewColourBarHeight** *default: 100*

  The height of the colourbar in MRView, in pixels.

- **MRViewColourBarInset** *default: 20*

  How far away from the edge of the main window to place the colourbar in MRView, in pixels.

- **MRViewColourBarPosition** *default: bottomright*

  The position of the colourbar within the main window in MRView. Valid values are: bottomleft, bottomright, topleft, topright.

- **MRViewColourBarTextOffset** *default: 10*

  How far away from the colourbar to place the associated text, in pixels.

- **MRViewColourBarWidth** *default: 20*

  The width of the colourbar in MRView, in pixels.

- **MRViewColourHorizontalPadding** *default: 100*

  The width in pixels between horizontally adjacent colour bars.

- **MRViewDockFloating** *default: 0 (false)*

  Whether MRView tools should start docked in the main window, or floating (detached from the main window).

- **MRViewFocusModifierKey** *default: meta (cmd on MacOSX)*

  Modifier key to select focus mode in MRView. Valid choices include shift, alt, ctrl, meta (on MacOSX: shift, alt, ctrl, cmd).

- **MRViewImageBackgroundColour** *default: 0,0,0 (black)*

  The default image background colour in the main MRView window.

- **MRViewMaxNumColourmapRows** *default: 3*

  The maximal number of rows used to layout a collection of rendered colourbars Note, that all tool-specific colourbars will form a single collection.

- **MRViewMoveModifierKey** *default: shift*

  Modifier key to select move mode in MRView. Valid choices include shift, alt, ctrl, meta (on MacOSX: shift, alt, ctrl, cmd).

- **MRViewRotateModifierKey** *default: ctrl*

  Modifier key to select rotate mode in MRView. Valid choices include shift, alt, ctrl, meta (on MacOSX: shift, alt, ctrl, cmd).

- **MRViewShowColourbar** *default: true*

  Colourbar shown in main image overlay

- **MRViewShowComments** *default: true*

  Comments shown in main image overlay

- **MRViewShowFocus** *default: true*

  Focus cross hair shown in main image

- **MRViewShowOrientationLabel** *default: true*

  Anatomical orientation information shown in main image overlay

- **MRViewShowVoxelInformation** *default: true*

  Voxel information shown in main image overlay

- **MRViewToolFontSize** *default: 2 points less than the standard system font*

  The point size for the font to use in MRView tools.

- **MRViewToolsColourBarPosition** *default: topright*

  The position of all visible tool colourbars within the main window in MRView. Valid values are: bottomleft, bottomright, topleft, topright.

- **MSAA** *default: 0 (false)*

  How many samples to use for multi-sample anti-aliasing (to improve display quality).

- **NIFTI.AllowBitwise** *default: 0 (false)*

  A boolean value to indicate whether bitwise storage of binary data is permitted (most 3rd party software packages don't support bitwise data). If false (the default), data will be stored using more widely supported unsigned 8-bit integers.

- **NeedOpenGLCoreProfile** *default: 1 (true)*

  Whether the creation of an OpenGL 3.3 context requires it to be a core profile (needed on newer versions of the ATI drivers on Linux, for instance).

- **NumberOfThreads** *default: number of threads provided by hardware*

  Set the default number of CPU threads to use for multi-threading.

- **NumberOfUndos** *default: 16*

  The number of undo operations permitted in the MRView ROI editor tool.

- **ObjectColor** *default: 1,1,0 (yellow)*

  The default colour to use for objects (i.e. SH glyphs) when not colouring by direction.

- **SparseDataInitialSize** *default: 16777216*

  Initial buffer size for data in MRtrix sparse image format file (in bytes).

- **SpecularExponent** *default: 1*

  The default exponent for the specular light in OpenGL renders.

- **SpecularIntensity** *default: 0.4*

  The default intensity for the specular light in OpenGL renders.

- **TerminalColor** *default: 1 (true)*

  A boolean value to indicate whether colours should be used in the terminal.

- **TmpFileDir** *default: '/tmp' (on Unix), '.' (on Windows)*

  The prefix for temporary files (as used in pipelines). By default, these files get written to the current folder, which may cause performance issues when operating over distributed file systems. In this case, it may be better to specify */tmp/* here.

- **TmpFilePrefix** *default: 'mrtrix-tmp-'*

  The prefix to use for the basename of temporary files. This will be used to generate a unique filename for the temporary file, by adding random characters to this prefix, followed by a suitable suffix (depending on file type). Note that this prefix can also be manipulated using the *MRTRIX_TMPFILE_PREFIX* environment variable, without editing the config file.

- **ToolbarStyle** *default: 2*

The style of the main toolbar buttons in MRView. See Qt's documentation for Qt::ToolButtonStyle.

- **TrackWriterBufferSize** *default: 16777216*

  The size of the write-back buffer (in bytes) to use when writing track files. MRtrix will store the output tracks in a relatively large buffer to limit the number of write() calls, avoid associated issues such as file fragmentation.

- **VSync** *default: 0 (false)*

  Whether the screen update should synchronise with the monitor's vertical refresh (to avoid tearing artefacts).

- **reg_analyse_descent** *default: 0 (false)*

  Linear registration: write comma separated gradient descent parameters and gradients to stdout and verbose gradient descent output to stderr

- **reg_bbgd** *default: 1 (true)*

  Linear registration: use Barzilai Borwein gradient descent

- **reg_coherence_len** *default: 3.0*

  Linear registration: estimated spatial coherence length in voxel

- **reg_gdweight_matrix** *default: 0.0003*

  Linear registration: weight for optimisation of linear (3x3) matrix parameters

- **reg_gdweight_translation** *default: 1*

  Linear registration: weight for optimisation of translation parameters

- **reg_stop_len** *default: 0.0001*

  Linear registration: smallest step in fraction of voxel at which to stop registration

CHAPTER 34

MRtrix 0.2 equivalent commands

For those users moving to *MRtrix3* from the old MRtrix 0.2.x software, this list provides the equivalent command names for the functionalities that they are accustomed to from the older version of the software. The new command naming scheme was carefully designed, so we hope people agree that it makes sense, and allows users to easily find the command functionalities available that are relevant for the data they are processing.

Further information can be found on these commands either through the documentation, or by typing the binary name at the command-line with no arguments to access the help file for that command.

MRtrix 0.2.x	MRtrix3	Comments
`average`	*mrmath*	Use `mean` statistic as second argument, and `-axis` option
`cat_tracks`	*tckedit*	Simply provide multiple input track files to the command
`cleanup_ANTS_warp`	*no equivalent*	
`csdeconv`	*dwi2fod*	
`dicom_filename`	*no equivalent*	See *dcminfo*
`dir2amp`	*peaks2amp*	
`disp_profile`	*shview*	
`dwi2SH`	*amp2sh*	
`dwi2tensor`	*dwi2tensor*	
`erode`	*maskfilter*	Specify `erode` or `dilate` algorithm as second argument
`estimate_response`	*dwi2response* manual	The *dwi2response* script also has a number of algorithms for automatically selec
`filter_tracks`	*tckedit*	
`find_SH_peaks`	*sh2peaks*	
`gen_ROI`	*no equivalent*	
`gen_WM_mask`	*no equivalent*	Use of this command was discouraged and so it has been discontinued
`gen_unit_warp`	*warpinit*	
`gendir`	*dirgen*	The electrostatic repulsion algorithm now only uses an exponent of 2 by default,
`import_tracks`	*tckconvert*	
`median3D`	*mrfilter*	Specify `median` algorithm as second argument
`mrabs`	*mrcalc*	Use `-abs` operator
`mradd`	*mrcalc* or *mrmath*	E.g. `mrcalc A.mif B.mif -add out.mif` or `mrmath A.mif B.mi`
`mrcat`	*mrcat*	

Table 34.1 – co

MRtrix 0.2.x	MRtrix3	Comments
`mrconvert`	*mrconvert*	
`mrinfo`	*mrinfo*	
`mrmult`	*mrcalc* or *mrmath*	E.g. `mrcalc A.mif B.mif -mult out.mif` or `mrmath A.mif B.mi`
`mrstats`	*mrstats*	
`mrtransform`	*mrtransform*	
`mrview`	*mrview*	
`normalise_tracks`	*tcknormalise*	
`read_dicom`	*dcminfo*	
`read_ximg`	*no equivalent*	
`resample_tracks`	*tcksample*	
`sample_tracks`	*tcksample*	
`sdeconv`	*dwi2fod*	
`select_tracks`	*tckedit*	
`streamtrack`	*tckgen*	
`tensor2ADC`	*tensor2metric*	Use `-adc` output option
`tensor2FA`	*tensor2metric*	Use `-fa` output option
`tensor2vector`	*tensor2metric*	Use `-vector` output option
`tensor_metric`	*tensor2metric*	
`threshold`	*mrthreshold*	Note that automatic threshold parameter determination (i.e. if you don't explicitl
`track_info`	*tckinfo*	
`tracks2prob`	*tckmap*	
`tracks2vtk`	*tckconvert*	
`truncate_tracks`	*tckedit*	