BTK developers documentation

Fbrain ERC project: Computational Anatomy of Fetal Brain

August 28, 2013

Contents

1 Introduction

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

1.1 Copyright

[1] http://lsiit-miv.u-strasbg.fr/miv/index.php?contenu=erc

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

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

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

1.2 Installation

1.2.1 Dependencies

Baby Brain Toolkit (BTK) is written in C++ and needs the following packages[2]:

- CMake, Tclap, OpenMP, VTK, ANN (www.cs.umd.edu/~mount/ANN) and Doxygen. These libraries can be installed using the following command line:
  - for debian-based distributions: `apt-get install cmake cmake-curses-gui libtclap-dev libgomp1 libann-dev doxygen`
  - for MacOSX using macports[3]: `port install cmake tclap libANN doxygen`
- Install Git: this library can be installed using the following command line :
  - for debian-based distribution: `apt-get install git-core`
  - for MacOSX using macports: `port install git-core`
- Insight Toolkit (ITK) version 4.3.1
  - Download the tar.gz archive of ITK v4.3.1
  - Extract the archive
  - Then open a terminal and type :

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

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

```
BUILD_TESTING = OFF
ITK_USE_REVIEW = ON
ITK_BUILD_ALL_MODULES = ON
ITKV3_COMPATIBILITY = ON
BUILD_DOCUMENTATION = OFF
CMAKE_BUILD_TYPE = Release
```

Or for debugging and development purpose :

[2] BTK has been tested under Debian 5.0 and MacOSX 10.6.8

[3] When configuring in cmake, please use /opt/local/include for TCLAP_DIRECTORY

```
CMAKE_BUILD_TYPE = Debug
```

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

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

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

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

```
BUILD_TESTING = OFF
BUILD_EXAMPLE = OFF
CMAKE_BUILD_TYPE = Release
```

Or for debugging and developement purpose :

```
CMAKE_BUILD_TYPE = Debug
```

1.2.2 Download the BTK sources

- Get the BTK sources:
  - For the Latest stable version :

```
git clone https://github.com/rousseau/fbrain.git
git checkout Btk_1.0
```

  - For the current development version :

```
git clone https://github.com/rousseau/fbrain.git
```

- Then for compiling:

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

Make the following changes on the CMake configuration screen :

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

Or for debugging and developement purpose :

```
CMAKE_BUILD_TYPE = Debug
```

then :

```
make
```

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

```
BUILD_DOCUMENTATION = ON
```

Then for generating the documentation run :

```
make doc
```

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

2 Development

This part contain all informations needed to develop new programs in BTK.

2.1 BTK tree

BTK is build using a tree. The source code is contained into the *Code* folder. Programs are respectively into *Applications* and *Utilities*. *Documentation* and *Medias* contain the documentation and some pictures for the Website of the project

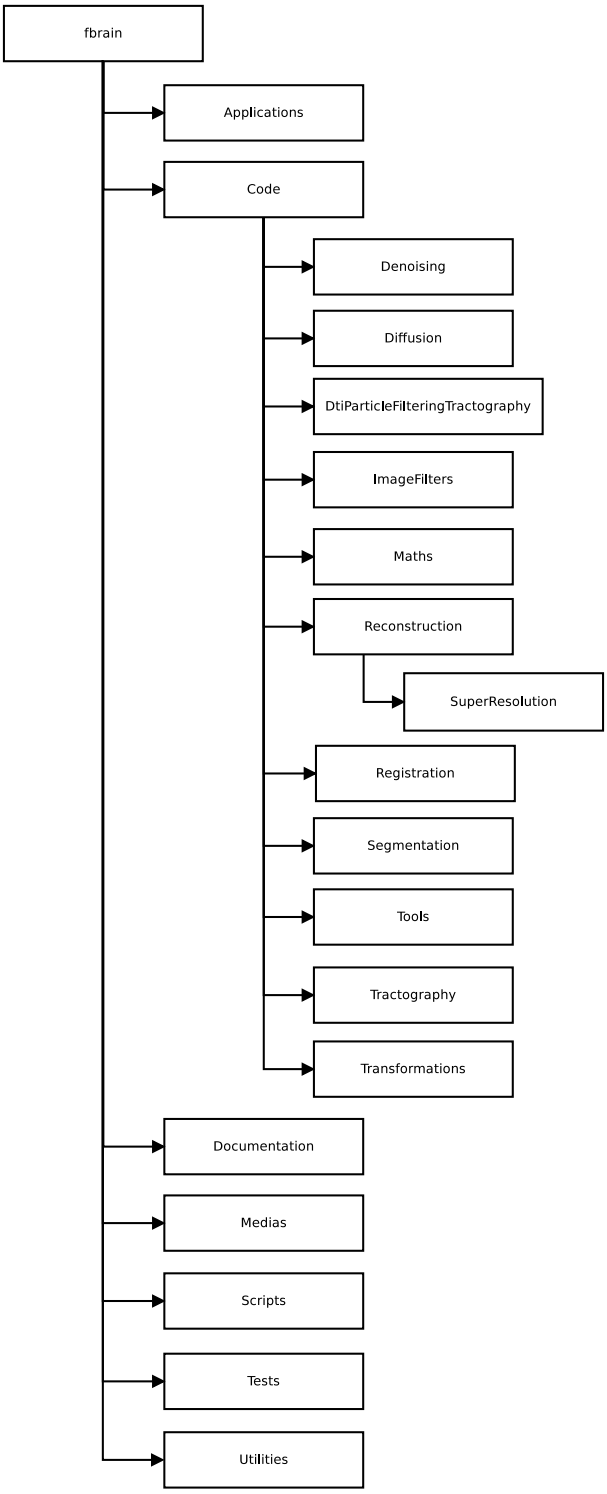

Figure 1: Tree of BTK.

2.2 Standards

Since BTK is written in C++, we use standards, most of time the standards come from ITK library.

- Name of Class :

Capital letter at the begining and then CamelCase[4] :

```
class MyClass
class MySuperClass
```

- Variable member of a Class :
  m_ at the beginin, then a Capital letter and the next in CamelCase :

```
int m_MyVariable
```

- Fonction member of a Class :
  Capital letter at the begining and then CamelCase :

```
void MyFonction()
void MySuperFonction()
```

- Conditions, loops, instructions... :
  Brackets at the line and aligned

```
for(int i = 0; i < 3; i++)
{
  //some stuff
}
void MySuperFonction()
{

}
```

- Constants, enum... :
  All letter of the word in capital letter, if composed word separated it by a _

```
enum AXES
{
    AXIAL =0,
    SAGITTAL,
    CORONAL
} ;

MY_CONSTANT TRUE = true;
```

Don't forget the C++ standards and be the more logical you can when programming !

2.3 Helper classes

In BTK we have some helper classes. As their names suggests,those classes are created for helping the programmer for usual fonctions.

[4] http://en.wikipedia.org/wiki/CamelCase

2.3.1 btkImageHelper

- File : Code/Tools/btkImageHelper.(h/txx)
- Using : Class of static fonctions for create, read, write a image or a vector of images
- example :

```
#include "btkImageHelper.h"

typedef itk::Image<short, 3> itkImage;

std::string inputFileName = "inputImage.nii.gz";
itkImage::Pointer image = btk::ImageHelper<itkImage>::ReadImage(inputFileName);

std::string outputFileName = "outputImage.nii.gz";
btk::ImageHelper<itkImage>::WriteImage(image, outputFileName);
```

2.3.2 btkIOImageHelper

- File : Code/Tools/btkIOImageHelper.(h/txx)
- Using : Class of static fonctions for check the type of the components of an image (before reading it)
- example :

```
#include "btkIOImageHelper.h"

std::string myFile;

if(IOImageHelper::GetComponentTypeOfImageFile(myFile) == IOImageHelper::FLOAT)
{
//Read the image in float
}
else if(IOImageHelper::GetComponentTypeOfImageFile(myFile) == IOImageHelper::USHORT)
{
//Read the image in unsigned short
}
```

2.3.3 btkFileHelper

- File : Code/Tools/btkFileHelper.(h/txx)
- Using : Class of static fonctions for check if a file exist or a vector of files
- example :

```
#include "btkFileHelper.h"

std::string filename("MyName");
bool fileExist = btk::FileHelper::FileExist(filename);
```

2.3.4 btkIOTransformHelper

- File : Code/Tools/btkIOTransformHelper.(h/txx)
- Using : Class of static fonctions for reading and writing transforms, this class is currently under developpment !
- example :

```
#include "btkIOTransformHelper.h"

typedef itk::AffineTransform<double,3> Transform;
Transform::Pointer transform = Transform::New();
transform->SetIdentity();

std string filename("MyName");

btk::IOTransformHelper<Transform>::WriteTransform(transform,filename);
```

2.3.5 btkMacro

- File : Code/Tools/btkMacro.h
- Using : File with some macro for Set/Get variable into classes, print error or warning, and throw exceptions
- example :

```
#include "btkMacro.h"

namespace btk
{
class MyClass:
{
public:
// the macro remove the m_ at the begining of the variable
btkSetMacro(MyVariable,int);
btkGetMacro(MyVariable,int);

private:
int m_MyVariable;

};
}

...

btkWaring("This is a usefull warning !");


...

btkCerr("This print an error ! ");


...
```

```
btkException("This is going to throw an itk::Exception (don't forget to catch ! ) ")
```

2.4 Doxygen

In order to generate a clear documentation, we using doxygen, so for your classes you should add comments in doxygen spirit. Here is a dummy example :

```
#ifndef __MYCLASS_H__
#define __MYCLASS_H__

/*!
 * \file MyClass.h
 * \brief MyClass class
 * \author Luke Skywalker
 * \version 0.1
 */


/*! \namespace btk
 *
 * namespace to contain all btk components
 */
namespace btk
{
  /*! \class MyClass
   * \brief class for doing some magical things
   *
   *  Description of the class
   */
  class MyClass
  {

  public:
    /*!
     *  \brief Constructor
     *
     *  Constructor of the MyClass class
     *
     *
     */
    MyClass();

    /*!
     *  \brief Destructor
     *
     *  Destructor of the MyClass class
     */
    virtual ~MyClass();

  protected :
  /*!
   *  \brief DoSomething
   *
```

```
   *  This method do something
   * \param value : integer value for doing something
   */
  void DoSomething(int value);

  /*!
   *  \brief DoAnotherThing
   *
   *  This method do something
   * \param value : integer value for doing something
   * \return bool : return true if value > 5 else return false
   */
  bool DoAnotherThing(int value);

  private:

  int m_MyVariable;/*!< My Variable for doing things*/

  };
};
```

2.5 Create a new program

First of all, you should know that in Btk we have two kind of programs, the first is the *Application* programs which contain the importants programs (for the research project) and the second is *Utilities* which contain tools and Helpful programs.
Creating a new executable program is separated into two steps. In first you should create a file.cxx which will contained your main Next you should add your program (and the nessesary includes) in the corresponding CMake file.

2.5.1 Create a main

You must create a new file with name of your choice for example btkImageReconstruction.cxx . If your program take arguments (ie: input/output), you should use TCLAP[5] library.

2.5.2 Add the programm into CMake

When your main program is written, you must add it into the CMake file. Note that there are two CMake files , one for the *Application* and the other for *Utilities*.

Here is a example :

```
ADD_EXECUTABLE(btkTensorStreamlineTractography
       btkTensorStreamlineTractography.cxx
       ${fbrain_SOURCE_DIR}/Code/Tractography/btkCartesianCoordinates.cxx
       ${fbrain_SOURCE_DIR}/Code/Tractography/btkSphericalCoordinates.cxx
       ${fbrain_SOURCE_DIR}/Code/Tractography/btkVector.cxx
       ${fbrain_SOURCE_DIR}/Code/Tractography/btkDirection.cxx
       ${fbrain_SOURCE_DIR}/Code/Tractography/btkPoint.cxx
       ${fbrain_SOURCE_DIR}/Code/DtiParticleFilteringTractography/btkDTFPSignalExtractor.cxx
       ${fbrain_SOURCE_DIR}/Code/DtiParticleFilteringTractography/btkDTFPSignal.cxx
       ${fbrain_SOURCE_DIR}/Code/Tractography/btkSphericalHarmonics.cxx
```

[5] See the documentation http://tclap.sourceforge.net/html/index.html

```
)

TARGET_LINK_LIBRARIES(btkTensorStreamlineTractography ${ITK_LIBRARIES} vtkHybrid)

INSTALL(TARGETS btkTensorStreamlineTractography
        DESTINATION bin)
```

Acknowledgment

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