OTB: An introduction for not-so-dummies

Francesco P. Lovergine

francesco@lovergine.com

INTRODUCTION

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get started
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get started
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get started
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

Tutorial goals

Introducing developers about philosophy and mechanisms of CNES Orfeo Toolbox.

What you will find here:

- Architecture and design
- Main objects and methods
- Main filters and functors
- Examples to get start
- Suggestions

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What is OTB

Orfeo Toolbox is a CNES project (mid 2003) aimed to provide an advanced FOSS toolbox for remote sensing and earth observation applications.

At the current date (3.10) the main components available are:

- The true core: the OTB C++ library.
- The revamped command-line OTB Apps set.
- Multiple bindings (*wrapping*) for Python and Java.
- Monteverdi (toy) GUI.
- An *impressive* developer's guide.

Last but not least, its license is the french *CeCILL* one (GPL-like).

IMHO

As usually, a *we–do–all* syndrome is evident, even if mitigated by full re-use of third parties components.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, Fltk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What's in OTB (the library)

Orfeo Toolbox predates Insight Toolkit (ITK) a library for medical imaging and adapt it to EO imaging. It fully re-uses other third-parties components to accomplish its ambitious goals. On those respects it is an archetypal FOSS product.

- CMake configuration/build system (>= 2.6)
- ITK 3.10 (*custom*) and VNL for numerics
- Ossim library
- Gdal
- optionally FFTW, cURL, FItk libraries, etc.
- KML library
- SVM library

Warning

You are fully adviced to use *embedded* libraries for its dependencies, instead of system ones, at least currently.

What is the OTB approach

OTB C++ library is fully inspired to the ITK library and adopts the following basic ideas:

- Fully OO and templated
- Pattern based and Generic Programming
- Agile development

Both provided examples and tests are of big help in exploitation of the library features.

What is the OTB approach

OTB C++ library is fully inspired to the ITK library and adopts the following basic ideas:

- Fully OO and templated
- Pattern based and Generic Programming
- Agile development

Both provided examples and tests are of big help in exploitation of the library features.

What is the OTB approach

OTB C++ library is fully inspired to the ITK library and adopts the following basic ideas:

- Fully OO and templated
- Pattern based and Generic Programming
- Agile development

Both provided examples and tests are of big help in exploitation of the library features.

What is the OTB approach

OTB C++ library is fully inspired to the ITK library and adopts the following basic ideas:

- Fully OO and templated
- Pattern based and Generic Programming
- Agile development

Both provided examples and tests are of big help in exploitation of the library features.

What is the OTB approach

OTB C++ library is fully inspired to the ITK library and adopts the following basic ideas:

- Fully OO and templated
- Pattern based and Generic Programming
- Agile development

Both provided examples and tests are of big help in exploitation of the library features.

GETTING START

Installing

```
$ tar xvf OrfeoToolbox-3.10.0.tgz
$ mkdir build && cd build
$ ccmake ../OrfeoToolbox-3.10.0
$ make
$ make install # destinazione: /usr/local/*/otb
```

At least under wheezy/sid at the date there are not big functional problems and all work flawlessly.

Installing

```
$ tar xvf OrfeoToolbox-3.10.0.tgz
$ mkdir build && cd build
$ ccmake ../OrfeoToolbox-3.10.0
$ make
$ make install # destinazione: /usr/local/*/otb
```

At least under wheezy/sid at the date there are not big functional problems and all work flawlessly.

Installing

```
CMAKE_BACKWARDS_COMPATIBILITY *2.4
GDALCONFIG_EXECUTABLE         */usr/bin/gdal-config
GEOTIFF_INCLUDE_DIRS          */usr/include/geotiff
GETTEXT_LIBRARY               */usr/lib/libgettextlib
GETTEXT_MSGFMT_EXECUTABLE     */usr/bin/msgfmt
GETTEXT_MSGMERGE_EXECUTABLE   */usr/bin/msgmerge
ITK_USE_KWSTYLE               *OFF
JPEG_INCLUDE_DIRS             */usr/include
OGR_INCLUDE_DIRS              */usr/include/gdal
OPENTHREADS_INCLUDE_DIR       */usr/include
OPENTHREADS_LIBRARY           */usr/lib/libOpenThread
OPENTHREADS_LIBRARY_DEBUG     */usr/lib/libOpenThread
OTB_LANG                      *it_IT.UTF-8
OTB_USE_EXTERNAL_FLTK         *OFF
OTB_USE_EXTERNAL_ITK          *OFF
```

Installing

```
OTB_USE_GETTEXT          *ON
OTB_USE_JPEG2000         *ON
TIFF_INCLUDE_DIRS        */usr/include
USE_OPENGL               *ON
BUILD_EXAMPLES            OFF
BUILD_SHARED_LIBS         ON
BUILD_TESTING             OFF
CMAKE_BUILD_TYPE
CMAKE_INSTALL_PREFIX      /usr/local
GDAL_CONFIG               /usr/bin/gdal-config
GDAL_INCLUDE_DIR          /usr/include/gdal
GDAL_LIBRARY              /usr/lib/libgdal.so <-- 1.8+
OTB_BUILD_PEDANTIC        OFF
```

First steps

Once installed under a suitable tree OTB allows using CMake to build new stuff, by means of very simple `CMakeLists.txt` build files such as:

HelloWorld: build script

```
PROJECT(HelloWorld)
CMAKE_MINIMUM_REQUIRED(VERSION 2.6)
FIND_PACKAGE(OTB REQUIRED)
INCLUDE(${OTB_USE_FILE})
ADD_EXECUTABLE(HelloWorld HelloWorld.cxx)
TARGET_LINK_LIBRARIES(HelloWorld OTBCommon OTBIO \
        ${OTB_IO_UTILITIES_DEPENDENT_LIBRARIES})
```

Configure and build

```
$ cmake .
$ make
```

Once installed under a suitable tree OTB allows using CMake to build new stuff, by means of very simple `CMakeLists.txt` build files such as:

HelloWorld: build script

```
PROJECT(HelloWorld)
CMAKE_MINIMUM_REQUIRED(VERSION 2.6)
FIND_PACKAGE(OTB REQUIRED)
INCLUDE(${OTB_USE_FILE})
ADD_EXECUTABLE(HelloWorld HelloWorld.cxx)
TARGET_LINK_LIBRARIES(HelloWorld OTBCommon OTBIO \
        ${OTB_IO_UTILITIES_DEPENDENT_LIBRARIES})
```

Configure and build

```
$ cmake .
$ make
```

Once installed under a suitable tree OTB allows using CMake to build new stuff, by means of very simple `CMakeLists.txt` build files such as:

HelloWorld: build script

```
PROJECT(HelloWorld)
CMAKE_MINIMUM_REQUIRED(VERSION 2.6)
FIND_PACKAGE(OTB REQUIRED)
INCLUDE(${OTB_USE_FILE})
ADD_EXECUTABLE(HelloWorld HelloWorld.cxx)
TARGET_LINK_LIBRARIES(HelloWorld OTBCommon OTBIO \
        ${OTB_IO_UTILITIES_DEPENDENT_LIBRARIES})
```

Configure and build

```
$ cmake .
$ make
```

HelloWorld: the code

```
#include "otbImage.h"
#include <iostream>

int main()
{
  typedef otb::Image<unsigned short,2> ImageType;

  ImageType::Pointer image = ImageType::New();

  std::cout << "Hello World and RIP Dennis!" \
            << std::endl;

  return EXIT_SUCCESS;
}
```

Concepts behind OTB

The two main classes `otb::Image`, `itk::Mesh` and itk::PointSet are used within one or more pipelines of filters for data processing. This is the main workflow of a generic OTB program.

The I/O is done in terms of *sources* (readers) and *mappers* (writers or viewers even) that are input/output components at the begin and end of the pipeline.

OTB objects use the object factory pattern to instantiate classes. This is taken from `ITK` and its `itk::ObjectFactoryBase` and allow a more flexible creation of dynamic objects. So, conventional C++ `new` methods are not used in any part of the library.

Both ITK and OTB are heavily multi-threadable and stream oriented to allow managing big chuncks of data.

Concepts behind OTB

The two main classes `otb::Image`, `itk::Mesh` and itk::PointSet are used within one or more pipelines of filters for data processing. This is the main workflow of a generic OTB program.

The I/O is done in terms of *sources* (readers) and *mappers* (writers or viewers even) that are input/output components at the begin and end of the pipeline.

OTB objects use the object factory pattern to instantiate classes. This is taken from `ITK` and its `itk::ObjectFactoryBase` and allow a more flexible creation of dynamic objects. So, conventional C++ `new` methods are not used in any part of the library.

Both ITK and OTB are heavily multi-threadable and stream oriented to allow managing big chuncks of data.

Concepts behind OTB

The two main classes `otb::Image`, `itk::Mesh` and itk::PointSet are used within one or more pipelines of filters for data processing. This is the main workflow of a generic OTB program.

The I/O is done in terms of *sources* (readers) and *mappers* (writers or viewers even) that are input/output components at the begin and end of the pipeline.

OTB objects use the object factory pattern to instantiate classes. This is taken from `ITK` and its `itk::ObjectFactoryBase` and allow a more flexible creation of dynamic objects. So, conventional C++ `new` methods are not used in any part of the library.

Both ITK and OTB are heavily multi-threadable and stream oriented to allow managing big chuncks of data.

Concepts behind OTB

The two main classes `otb::Image`, `itk::Mesh` and itk::PointSet are used within one or more pipelines of filters for data processing. This is the main workflow of a generic OTB program.

The I/O is done in terms of *sources* (readers) and *mappers* (writers or viewers even) that are input/output components at the begin and end of the pipeline.

OTB objects use the object factory pattern to instantiate classes. This is taken from `ITK` and its `itk::ObjectFactoryBase` and allow a more flexible creation of dynamic objects. So, conventional C++ `new` methods are not used in any part of the library.

Both ITK and OTB are heavily multi-threadable and stream oriented to allow managing big chuncks of data.

Copyright

Copyright (C) 2011 Francesco P. Lovergine
<francesco@lovergine.com>

This work is distributed under a Creative Commons License Attributions-Noncommercial-NoDerivs 3.0.

See the full text of the license on
http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode

