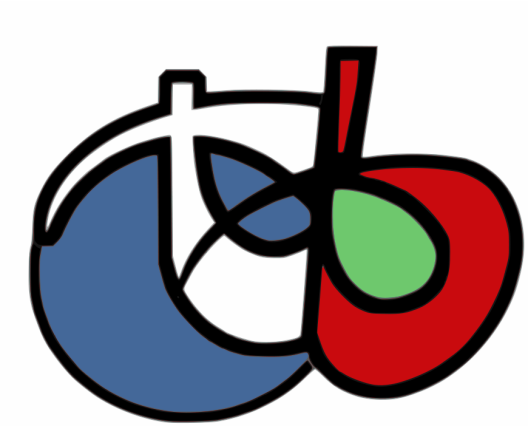
cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES

The ORFEO Tool Box Software Guide
Updated for OTB-6.6.1

OTB Development Team

November 29, 2018

http://www.orfeo-toolbox.org
e-mail: otb@cnes.fr



The ORFEO Toolbox is not a black box.

Ch.D.

FOREWORD

Beside the Pleiades (PHR) and Cosmo-Skymed (CSK) systems developments forming ORFEO, the dual and bilateral system (France - Italy) for Earth Observation, the ORFEO Accompaniment Program was set up, to prepare, accompany and promote the use and the exploitation of the images derived from these sensors.

The creation of a preparatory program[1] is needed because of:

- the new capabilities and performances of the ORFEO systems (optical and radar high resolution, access capability, data quality, possibility to acquire simultaneously in optic and radar),
- the implied need of new methodological developments : new processing methods, or adaptation of existing methods,
- the need to realise those new developments in very close cooperation with the final users for better integration of new products in their systems.

This program was initiated by CNES mid-2003 and will last until mid 2013. It consists in two parts, between which it is necessary to keep a strong interaction:

- A Thematic part,
- A Methodological part.

The Thematic part covers a large range of applications (civil and defence), and aims at specifying and validating value added products and services required by end users. This part includes consideration about products integration in the operational systems or processing chains. It also includes a careful thought on intermediary structures to be developed to help non-autonomous users. Lastly, this part aims at raising future users awareness, through practical demonstrations and validations.

[1] http://smsc.cnes.fr/PLEIADES/A_prog_accomp.htm

The Methodological part objective is the definition and the development of tools for the operational exploitation of the submetric optic and radar images (tridimensional aspects, changes detection, texture analysis, pattern matching, optic radar complementarities). It is mainly based on R&D studies and doctorate and post-doctorate researches.

In this context, CNES[2] decided to develop the *ORFEO ToolBox* (OTB), a set of algorithms encapsulated in a software library. The goals of the OTB is to capitalise a methological *savoir faire* in order to adopt an incremental development approach aiming to efficiently exploit the results obtained in the frame of methodological R&D studies.

All the developments are based on FLOSS (Free/Libre Open Source Software) or existing CNES developments. OTB is distributed under the permissive open source license Apache v2.0 - aka Apache Software License (ASL) v2.0:
`http://www.apache.org/licenses/LICENSE-2.0`

OTB is implemented in C++ and is mainly based on ITK[3] (Insight Toolkit).

[2] http://www.cnes.fr
[3] http://www.itk.org

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

Introduction

CHAPTER

ONE

WELCOME

Welcome to the *ORFEO ToolBox (OTB) Software Guide*.

This document presents the essential concepts used in OTB. It will guide you through the road of learning and using OTB. The Doxygen documentation for the OTB application programming interface is available on line at `https://www.orfeo-toolbox.org/doxygen`.

1.1 Organization

This software guide is divided into several parts, each of which is further divided into several chapters. Part I is a general introduction to OTB, with—in the next chapter—a description of how to install the ORFEO Toolbox on your computer. Part I also introduces basic system concepts such as an overview of the system architecture, and how to build applications in the C++ programming language. Part **??** is a short guide with gradual difficulty to get you start programming with OTB. Part **??** describes the system from the user point of view. Dozens of examples are used to illustrate important system features. Part **??** is for the OTB developer. It explains how to create your own classes and extend the system.

1.2 How to Learn OTB

There are two broad categories of users of OTB. First are class developers, those who create classes in C++. The second, users, employ existing C++ classes to build applications. Class developers must be proficient in C++, and if they are extending or modifying OTB, they must also be familiar with OTB's internal structures and design (material covered in Part **??**).

The key to learning how to use OTB is to become familiar with its palette of objects and the ways of combining them. We recommend that you learn the system by studying the examples and then, if you are a class developer, study the source code. Start by the first few tutorials in Part **??** to get

familiar with the build process and the general program organization, follow by reading Chapter **??**, which provides an overview of some of the key concepts in the system, and then review the examples in Part **??**. You may also wish to compile and run the dozens of examples distributed with the source code found in the directory `OTB/Examples`. (Please see the file `OTB/Examples/README.txt` for a description of the examples contained in the various subdirectories.) There are also several hundreds of tests found in the source distribution in `OTB/Testing/Code`, most of which are minimally documented testing code. However, they may be useful to see how classes are used together in OTB, especially since they are designed to exercise as much of the functionality of each class as possible.

1.3 Software Organization

The following sections describe the directory contents, summarize the software functionality in each directory, and locate the documentation and data.

1.3.1 Obtaining the Software

Periodic releases of the software are available on the OTB Website. These official releases are available a few times a year and announced on the ORFEO Web pages and mailing lists.

This software guide assumes that you are working with the latest official OTB release (available on the OTB Web site).

OTB can be downloaded without cost from the following web site:

`http://www.orfeo-toolbox.org/`

In order to track the kind of applications for which OTB is being used, you will be asked to complete a form prior to downloading the software. The information you provide in this form will help developers to get a better idea of the interests and skills of the toolkit users.

Once you fill out this form you will have access to the download page. This page can be book marked to facilitate subsequent visits to the download site without having to complete any form again.

Then choose the tarball that better fits your system. The options are `.zip` and `.tgz` files. The first type is better suited for MS-Windows while the second one is the preferred format for UNIX systems.

Once you unzip or untar the file, a directory called `OTB` will be created in your disk and you will be ready for starting the configuration process described in Section 2.

There are two other ways of getting the OTB source code:

- Clone the current release with Git from the OTB git server, (master branch)

- Clone the latest revision with Git from the OTB git server (develop branch).

These last two options need a proper Git installation. To get source code from Git, do:

```
git clone https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb.git
```

Using Git, you can easily navigate through the different versions. The master branch contains the latest stable version:

```
git checkout master
```

Specific versions are availables with tags:

```
git checkout 5.2.0
```

Finally, this brings you to the latest development version:

```
git checkout develop
```

There is also a mirror of OTB official repository on GitHub. You can find more information on the OTB git workflow in the wiki.

1.3.2 Directory Structure

To begin your OTB odyssey, you will first need to know something about OTB's software organization and directory structure. It is helpful to know enough to navigate through the code base to find examples, code, and documentation.

The `OTB` contains the following subdirectories:

- `OTB/Modules`—the heart of the software; the location of the majority of the source code.
- `OTB/CMake`—internal files used during the configuration process.
- `OTB/Copyright`—the copyright information of OTB and all the dependencies included in the OTB source tree.
- `OTB/Examples`—a suite of simple, well-documented examples used by this guide and to illustrate important OTB concepts.
- `OTB/Superbuild`—CMake scripts to automatically download, patch, build and install important dependencies of OTB (ITK, OSSIM, GDAL to name a few).
- `OTB/Utilities`—small programs used for the maintenance of OTB.

OTB is organized into different modules, each one covering different part of image processing. It is therefore important to understand the source code directory structure—found in `OTB/Modules`—.

- `OTB/Modules/Adapters`—Adapters for Boost, Curl, Gdal, OpenThreads and Ossim.
- `OTB/Modules/Applications`—a set of applications modules that can be launched in different ways (command-line, graphical interface, Python/Java), refer to the OTB Cookbook for more information.
- `OTB/Modules/Core`—core classes, macro definitions, typedefs, and other software constructs central to OTB.
- `OTB/Modules/Detection`—detection of clouds, roads.
- `OTB/Modules/Feature`—various local descriptors and features.
- `OTB/Modules/Filtering`—basic image processing filters.
- `OTB/Modules/Fusion`—image fusion algorithms, as for instance, pansharpening.
- `OTB/Modules/Hyperspectral`—hyperspectral images analysis.
- `OTB/Modules/IO`—classes that support the reading and writing of data.
- `OTB/Modules/Learning`—several functionalities for supervised learning and classification.
- `OTB/Modules/OBIA`—Object Based Image Analysis filters and data structures.
- `OTB/Modules/Radiometry`—classes allowing to compute vegetation indices and radiometric corrections.
- `OTB/Modules/Registration`—classes for registration of images or other data structures to each other.
- `OTB/Modules/Remote`—Functions to fetch remote modules.
- `OTB/Modules/Segmentation`—several functionalities for image segmentation.
- `OTB/Modules/ThirdParty`—Modules that import OTB's dependencies.
- `OTB/Modules/Wrappers`—Applications wrappers with several access points (command-line, QT Gui, SWIG...).

See also chapter **??** for more information about how to write modules.

1.3.3 Documentation

Besides this text, there are other documentation resources that you should be aware of.

Doxygen Documentation. The Doxygen documentation is an essential resource when working with OTB. These extensive Web pages describe in detail every class and method in the system. The documentation also contains inheritance and collaboration diagrams, listing of event invocations, and data members. The documentation is heavily hyper-linked to other classes and to the source code. The Doxygen documentation is available on-line at `http://www.orfeo-toolbox.org/doxygen/`.

Header Files. Each OTB class is implemented with a .h and .cxx/.txx file (.txx file for templated classes). All methods found in the .h header files are documented and provide a quick way to find documentation for a particular method. (Indeed, Doxygen uses the header documentation to produces its output.)

1.3.4 Data

The OTB Toolkit was designed to support the ORFEO Acompaniment Program and its associated data. This data is available at `http://smsc.cnes.fr/PLEIADES/index.htm`.

1.4 The OTB Community and Support

1.4.1 Join the Mailing List

It is strongly recommended that you join the users mailing list. This is one of the primary resources for guidance and help regarding the use of the toolkit. You can subscribe to the users list online at

`http://groups.google.com/group/otb-users`

The otb-users mailing list is also the best mechanism for expressing your opinions about the toolbox and to let developers know about features that you find useful, desirable or even unnecessary. OTB developers are committed to creating a self-sustaining open-source OTB community. Feedback from users is fundamental to achieving this goal.

1.4.2 Community

OTB was created from its inception as a collaborative, community effort. Research, teaching, and commercial uses of the toolkit are expected. If you would like to participate in the community, there are a number of possibilities.

- Users may actively report bugs, defects in the system API, and/or submit feature requests. Currently the best way to do this is through the OTB users mailing list.
- Developers may contribute classes or improve existing classes. If you are a developer, you may request permission to join the OTB developers mailing list. Please do so by sending email to otb "at" cnes.fr. To become a developer you need to demonstrate both a level of competence as well as trustworthiness. You may wish to begin by submitting fixes to the OTB users mailing list.
- Research partnerships with members of the ORFEO Acompaniment Program are encouraged. CNES will encourage the use of OTB in proposed work and research projects.
- Educators may wish to use OTB in courses. Materials are being developed for this purpose, e.g., a one-day, conference course and semester-long graduate courses. Watch the OTB web pages or check in the `OTB-Documents/CourseWare` directory for more information.

Orfeo ToolBox is currently in the incubation stage of being part of the OSGeo[1] foundation.Within the ORFEO ToolBox community we act respectfully toward others in line with the OSGeo Code of Conduct[2].

1.5 A Brief History of OTB

Beside the Pleiades (PHR) and Cosmo-Skymed (CSK) systems developments forming ORFEO, the dual and bilateral system (France - Italy) for Earth Observation, the ORFEO Accompaniment Program was set up, to prepare, accompany and promote the use and the exploitation of the images derived from these sensors.

The creation of a preparatory program[3] is needed because of :

- the new capabilities and performances of the ORFEO systems (optical and radar high resolution, access capability, data quality, possibility to acquire simultaneously in optic and radar),
- the implied need of new methodological developments : new processing methods, or adaptation of existing methods,
- the need to realize those new developments in very close cooperation with the final users for better integration of new products in their systems.

This program was initiated by CNES mid-2003 and will last until 2010 at least It consists in two parts, between which it is necessary to keep a strong interaction :

[1] http://www.osgeo.org/
[2] http://www.osgeo.org/code_of_conduct
[3] http://smsc.cnes.fr/PLEIADES/A_prog_accomp.htm

- A Thematic part
- A Methodological part.

The Thematic part covers a large range of applications (civil and defence ones), and aims at specifying and validating value added products and services required by end users. This part includes consideration about products integration in the operational systems or processing lines. It also includes a careful thought on intermediary structures to be developed to help non-autonomous users. Lastly, this part aims at raising future users awareness, through practical demonstrations and validations.

The Methodological part objective is the definition and the development of tools for the operational exploitation of the future submetric optic and radar images (tridimensional aspects, change detection, texture analysis, pattern matching, optic radar complementarities). It is mainly based on R&D studies and doctorate and post-doctorate research.

In this context, CNES[4] decided to develop the *ORFEO ToolBox* (OTB), a set of algorithms encapsulated in a software library. The goals of the OTB is to capitalize a methological *savoir faire* in order to adopt an incremental development approach aiming to efficiently exploit the results obtained in the frame of methodological R&D studies.

All the developments are based on FLOSS (Free/Libre Open Source Software) or existing CNES developments.

OTB is implemented in C++ and is mainly based on ITK[5] (Insight Toolkit):

- ITK is used as the core element of OTB
- OTB classes inherit from ITK classes
- The software development procedure of OTB is strongly inspired from ITK's (Extreme Programming, test-based coding, Generic Programming, etc.)
- The documentation production procedure is the same as for ITK
- Several chapters of the Software Guide are literally copied from ITK's Software Guide (with permission).
- Many examples are taken from ITK.

1.5.1 ITK's history

In 1999 the US National Library of Medicine of the National Institutes of Health awarded six three-year contracts to develop an open-source registration and segmentation toolkit, that eventually came to be known as the Insight Toolkit (ITK) and formed the basis of the Insight Software

[4] http://www.cnes.fr
[5] http://www.itk.org

Consortium. ITK's NIH/NLM Project Manager was Dr. Terry Yoo, who coordinated the six prime contractors composing the Insight consortium. These consortium members included three commercial partners—GE Corporate R&D, Kitware, Inc., and MathSoft (the company name is now Insightful)—and three academic partners—University of North Carolina (UNC), University of Tennessee (UT) (Ross Whitaker subsequently moved to University of Utah), and University of Pennsylvania (UPenn). The Principle Investigators for these partners were, respectively, Bill Lorensen at GE CRD, Will Schroeder at Kitware, Vikram Chalana at Insightful, Stephen Aylward with Luis Ibáñez at UNC (Luis is now at Kitware), Ross Whitaker with Josh Cates at UT (both now at Utah), and Dimitri Metaxas at UPenn (now at Rutgers). In addition, several subcontractors rounded out the consortium including Peter Raitu at Brigham & Women's Hospital, Celina Imielinska and Pat Molholt at Columbia University, Jim Gee at UPenn's Grasp Lab, and George Stetten at the University of Pittsburgh.

In 2002 the first official public release of ITK was made available.

CHAPTER

TWO

COMPILING OTB FROM SOURCE

There are two ways to install OTB on your system: installing from a binary distribution or compiling from sources. You can find information about the installation of binary packages for OTB and Monteverdi in the OTB-Cookbook.

This chapter covers the compilation of OTB from source. Note that it also includes the compilation of Monteverdi which is integrated as an OTB module since version 5.8.

OTB has been developed and tested across different combinations of operating systems, compilers, and hardware platforms including Windows, GNU/Linux and macOS. It is known to work with the following compilers in 32/64 bit:

- Visual Studio 2015 on Windows
- GCC 4.x,5.x or CLang 3.x on GNU/Linux
- AppleClang on macOS (10.8 or higher)

Since release version 6.2.0, OTB is compiled using the C++14 standard by default.

The challenge of supporting OTB across platforms has been solved through the use of CMake, a cross-platform, open-source build system. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice. CMake is quite sophisticated: it supports complex environments requiring system configuration, compiler feature testing, and code generation.

CMake supports several generators to produce the compilation scripts, dependending on the platform and compiler. It can use:

- Makefiles for Unix systems
- Visual Studio workspaces for Windows