Orfeo ToolBox
Open source processing of remote sensing images (updated for 5.4)

OTB Team CNES
Things to know about OTB...

Orfeo ToolBox is:

- An image processing library for remote sensing
- Free and open source software under CeCILL-v2 license (equivalent to GPL)
- Funded and developed by CNES (French Space Agency) in the frame of the Orfeo Pléiades program (and beyond)
- Written in C++ on top of ITK (medical image processing)
- Built on the shoulders of giants (GDAL, OSSIM, OpenCV...)
- Big Data capable, thanks to built-in streaming and multithreading

orfeo-toolbox.org
Why open source?

Maximum reach
OTB is dedicated to every user of satellite images. Its wide dissemination contributes to the missions success (Pléiades, Sentinels...)

Quality and efficiency
OTB covers a vast panel of applications and thematic fields. Openness should:

▶ Facilitate appropriation and validation for users
▶ Encourage contributions and bug reports
▶ Available on multiple platforms
▶ “The Cathedral & the Bazaar”¹: the more widely available the source code is for public testing experimentation, the more rapidly all forms of bugs will be discovered

Reproducible research
OTB capitalizes a part of the CNES R&D in IP, open source contributes to transparent, reproducible and trans-disciplinary research.

¹http://www.catb.org/esr/writings/cathedral-bazaar/
Outline

Functions and algorithms

Key characteristics

How to use OTB?

What's new in OTB?

Conclusion
Incomplete list of OTB functions

Pre-processing

- Radiometric calibration, orthorectification, resampling (raster and vector), pan-sharpening, stereo rectification...
- Sensor supported: Sentinels, Pléiades, SPOT6, SPOT5, Digital Globe satellites
- Geometric models (thanks to OSSIM), support for DEM (SRTM or GeoTIFF)

Images and vector manipulation

- Formats supported by GDAL (raster and vector), conversion raster/vector
- Region of interest extraction, of spectral bands, concatenation or splitting...
- Band math, color mapping, contrast enhancement
- Linear filtering, Mathematical morphology
(Incomplete) List of OTB functions

Feature extraction
- Edge detection, scale-invariant feature transform, lines, corners
- Radiometric indices, textures (Haralick, SFS, PanTex)
- Local statistics (Flusser moments, Histogram of Oriented Gradient)
- Keypoints matching (SIFT, SURF...)

Change detection
- Classic methods with image metrics comparison
- Multivariate Alteration Detector

Dimensionality reduction, hyperspectral processing
- PCA, NAPCA, ICA, MAF...
- Dimension estimation, endmembers extraction, Vertex Component Analysis(VCA)
Incomplete list of OTB functions

Segmentation

- Segmentation algorithms: Connected Components, MeanShift, Watershed...
- Methods to apply those algorithms on large dataset
- Vector or raster representation which allow Object Based Image Analysis

Classification

- 9 supervised methods available (including SVM and Random Forests)
- Fusion and regularization of classifications
- K-Means clustering or Kohonen maps
- Object classification (from a segmentation)
Outline

Functions and algorithms

Key characteristics

How to use OTB?

What's new in OTB?

Conclusion
Motivations

- Interfaces seamlessly with other image processing and remote sensing open-source software
- Increase the number of functions
- Combine tools to create hybrid data pipeline

OTB backbone

- ITK: data processing pipeline
- GDAL: read and write raster and vector data
- OSSIM: sensor modelling and metadata support
- OpenCV and LibSVM: machine learning algorithms
- MuParser and MuParserX: powerful parsing of mathematical expression (band math)
Compatible (and available) on multiple platforms

Goal

- Compile with recent versions of:
  - GCC
  - Clang
  - MinGW
  - Visual Studio...

- Binary packages available:
  - UbuntuGIS repository (GIS and IP software for Ubuntu)
  - Experimental Debian packages
  - Available in OSGeo4W (OSGeo tools on Windows)
  - Binary installers, Port and Brew formula for Mac OS X...

Number of OTB downloads on Sourceforge per Operating System
Flexibility, scalability: Pipeline, Streaming and multithreading

**Pipeline data model**

**Streaming**

Behind the scene
State of the art

- Try to keep track of up-to-date information about the latest developments, exchanging ideas, identifying future trends, and making networking
- Reference implementation of algorithms based on publications
  - e.g.: morphological profil, MeanShift segmentation, Haralick textures, SURF keypoints...
- Reference implementation contributes by authors with their publications. e.g.: Large Scale MeanShift, object detection ...
How is OTB developed?

- Distributed version control: Git (migration from Mercurial in July 2015)
- C++ and CMake (CTest, CDash)
- Test driven development (TDD)
- Agile (scrum)
- Continuous integration and packaging

Every day, almost 3000 tests are compiled, launched on 16 different configurations.
Outline

Functions and algorithms

Key characteristics

How to use OTB?

What’s new in OTB?

Conclusion
How to use OTB?

**Visualization**
Parameters tuning ➞ Monteverdi

**Processing chains**
High level tasks ➞ OTB Applications – CLI, QT or Python API

**Fast prototyping**
Fast development ➞ OTB Core - C++ API

**Third parties**
ITK, GDAL, OSSIM, MuParser, ...

**End users software**
Monteverdi

**High level tools**
OTB Applications – CLI, QT or Python API

**Full API**
OTB Core - C++ API

**Third parties**
ITK, GDAL, OSSIM, MuParser, ...

**Write your own code**
Flexible, access to full API, requires C++ knowledge

**Use the applications**
High level functions (e.g. segmentation), callable from CLI, Qt, Python, can be extended

**Use Monteverdi**
Visualization, data management, Access to all applications
Show me the code!

```c++
#include "otbImage.h"
#include "otbImageFileReader.h"
#include "otbImageFileWriter.h"
#include "itkCannyEdgeDetectionImageFilter.h"
#include "itkRescaleIntensityImageFilter.h"

int main(int argc, char * argv[])
{
    typedef double PixelType;
    typedef otb::Image<PixelType> ImageType;

    typedef unsigned char OutputPixelType;
    typedef otb::Image<OutputPixelType> OutputImageType;

    typedef otb::ImageFileReader<ImageType> ReaderType;
    ReaderType::Pointer reader = ReaderType::New();
    reader->SetFileName(argv[1]);

    typedef itk::CannyEdgeDetectionImageFilter<ImageType, ImageType> FilterType;
    FilterType::Pointer filter = FilterType::New();
    filter->SetInput(reader->GetOutput());

    typedef otb::ImageFileWriter<OutputImageType> WriterType;
    WriterType::Pointer writer = WriterType::New();
    writer->SetFileName(argv[2]);
    writer->SetInput(filter->GetOutput());
    writer->Update();
}
```
The applications: write it once, use everywhere

- 87 applications are shipped with OTB
- 1 application = 1 dynamic library (plugin)
- Applications are auto-descriptive and auto-documented
- Applications can be extended outside of OTB
- Several plugins players:
  - Command-line
  - Qt auto-generated
  - Python
- Applications are meant for integration in external systems
Applications: command-line invocation

```
$ otbcli_OrthoRectification
```

**ERROR:** Waiting for at least one parameter...

This is the OrthoRectification application, version 5.2.1

This application allows to ortho-rectify optical images from supported sensors.

Complete documentation: http://www.orfeo-toolbox.org/Applications/OrthoRectification.html

**Parameters:**

- `-progress` `<boolean>` Report progress
- `MISSING -io.in` `<string>` Input Image (mandatory)
- `MISSING -io.out` `<string>` [pixel] Output Image [pixel=uint8/uint16/uint32/int32/float/double] (default value is float) (mandatory)
- `-map` `<string>` Output Cartographic Map Projection [utm/lambert2/lambert93/wgs/epsg] (mandatory)
- `-map.utm.zone` `<int32>` Zone number (mandatory, default value is 31)
- `-map.utm.northhem` `<boolean>` Northern Hemisphere (optional, off by default)
- `-map.epsg.code` `<int32>` EPSG Code (mandatory, default value is 4326)
- `-outputs.mode` `<string>` Parameters estimation modes [auto/autosize/autospacing/outputroi/orthofit] (mandatory)
- `MISSING -outputs.ulx` `<float>` Upper Left X (mandatory)
- `MISSING -outputs.uly` `<float>` Upper Left Y (mandatory)
- `MISSING -outputs.sizex` `<int32>` Size X (mandatory)
- `MISSING -outputs.sizey` `<int32>` Size Y (mandatory)
- `MISSING -outputs.spacingx` `<float>` Pixel Size X (mandatory)
- `MISSING -outputs.spacingy` `<float>` Pixel Size Y (mandatory)
- `-outputs.lr.x` `<float>` Lower right X (optional, off by default)
- `-outputs.lr.y` `<float>` Lower right Y (optional, off by default)
- `-outputs.ortho` `<string>` Model ortho-image (optional, off by default)
- `-outputs.isotropic` `<boolean>` Force isotropic spacing by default (optional, on by default)
- `-outputs.default` `<float>` Default pixel value (optional, on by default, default value is 0)
- `-elev.dem` `<string>` DEM directory (optional, off by default)
- `-elev.geoid` `<string>` Geoid File (optional, off by default)
- `-elev.default` `<float>` Default elevation (mandatory, default value is 0)
- `-interpolator` `<string>` Interpolation [bco/nn/linear] (mandatory, default value is bco)
Applications: Graphical interface
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the OrthoRectification application
OrthoRectification = otb.Registry.CreateApplication("OrthoRectification")

# The following lines set all the application parameters:
OrthoRectification.IO.IN = "QB_TOULOUSE_MUL_Extract_500_500.tif"
OrthoRectification.IO.OUT = "QB_Toulouse_ortho.tif"

app.MAP = 'epsg'
app.MAP.EPSG.CODE = 32768

# The following line execute the application
OrthoRectification.ExecuteAndWriteOutput()
Monteverdi (access to OTB applications)
QGIS (access to OTB applications)
Outline

- Functions and algorithms
- Key characteristics
- How to use OTB?
- What's new in OTB?
- Conclusion
5.0 (Mai 2015)

Modularity

▶ Better code layout, coherent modules (124 modules and 16 groups) with source, test and applications.
▶ Dependency management
▶ External contributions: https://www.orfeo-toolbox.org/external-projects/

SuperBuild

▶ No more third party software in OTB!
▶ The Superbuild downloads, configures, builds and installs dependencies
▶ Offline mode for compiling OTB without network access (e.g. airplane)
Open governance: Project Steering Committee

Genèse du PSC

- Until 2015: OTB is open-source software
- In March 2015: OTB became free software, with CNES as the first PSC

A club of developers, not managers

- High level project steering, roadmaps, communication and planning
- Vote RFCs: all members’ votes have the same value (±1, ±0)
- Seats do not expire. Exits are by resignation or vote of expulsion
- The PSC is not a legal entity and has no funding

Numbers

- 5 members from 4 different organisations
- 2 releases under a PSC (5.2, 5.4)
- 3 online meetings (with public logs)
5.2 (December 2015)

OTB

- New SAR processing applications (polarimetry, radiometry, speckle)
- Support for Sentinel-1 products
- Better Python bindings
- Better GDAL 2.0 compatibility and support Sentinel-2 images
- ...

Monteverdi 3.0

- Display an image mosaic or multi-temporal dataset
- Efficient visualization tools (local contrast, gradient...)
- Access to OTB applications
5.4 (May 2016)

OTB

- Switched to a fixed release schedule
- Merged Ice into OTB
- External build of external modules
- New SAR decompositions: Barnes, Huynen, Pauli

Monteverdi 3.2

- Screenshot feature
- Generate GDAL overviews
- Support for GDAL subdatasets
- Added to the SuperBuild
How many users?

Hard to tell...

- ≈ 600 members on the otb-users list
- Between 100 and 150 mails by months
- ≈ 100 members on the developers list
- ≈ 118 user accounts on the bug tracker
- ≈ 50 contributors in the documentation
- ≈ 3400 downloads for OTB 5.0 on SourceForge (released June 1, 2015).
Success stories

- OTB has been useful for (some) ORFEO users!
- Several training courses (3/5-day courses) given in France, Belgium, Madagascar, UNESCO, Hawaii, Finland...
- OTB has successfully processed 619 Pléiades images on RTU web site
- OTB provides many useful RS functions in one single tool
- OTB is/was the only open-source supporting PHR images (thanks to OpenJPEG)
- OTB equals or beats state-of-the-art tools (open source and maybe $$) on some points:
  - band calculator
  - tile-wise segmentation of full imagery
  - full scene classification with a range of machine learning algorithms
  - bridges between RS and SIG...
- Beyond Orfeo, OTB is already used in several projects and software
  - OSGeo incubation
Projects and software using OTB

- OTB applications are available through QGIS processing framework
- OTB is a component of Sentinel-2 and Venus ground segment (CNES and ESA)
- Terr’Image: Educational software for satellite image analysis
- Use to prototype THEIA products from the Scientific Expertise Centres
- ESA Sentinel-2 for Agriculture
- Gnorasi Software (National Technical University of Athens)
- Vahine project (hyperspectral processing of astrophysics), IPAG
- Geosud project (IRSTEA)
- TCM research program (ETS Quebec)
Support/Help/Contribute

General resources
  Site web  orfeo-toolbox.org
  Wiki     wiki.orfeo-toolbox.org
  Blog     blog.orfeo-toolbox.org

Documentation and help
  Guides   Software Guide and CookBook (remote sensing recipes)
  Doxygen  doxygen

Users mailing list  otb-users@googlegroups.com
Developers mailing list  otb-developers@googlegroups.com

Follow-up
  Look at the code?  git.orfeo-toolbox.org
    Find a bug?    bugs.orfeo-toolbox.org
    Agile?         scrum.orfeo-toolbox.org
    Weather?       dash.orfeo-toolbox.org
Thank you! Any questions?