
OTB CookBook Documentation

Release 5.8.0

OTB Team

November 07, 2016

CONTENTS

1	Welcome to OTB CookBook’s documentation!	1
2	Installation	3
2.1	Windows	3
2.2	Linux x86_64	4
2.3	FAQ for Packaging	4
2.4	MacOS X	5
2.5	Other packages	6
3	A brief tour of OTB Applications	9
3.1	Introduction	9
3.2	Running the applications	9
3.3	Advanced applications capabilities	17
4	Monteverdi	21
4.1	GUI : what does it look like ?	21
4.2	Examples	26
5	Recipes	37
5.1	From raw image to calibrated product	37
5.2	SAR processing	43
5.3	Residual registration	63
5.4	Image processing and information extraction	66
5.5	Classification	71
5.6	Feature extraction	99
5.7	Stereoscopic reconstruction from VHR optical images pair	106
5.8	BandMathImageFilterX (based on muParserX)	114
5.9	Numpy processing in OTB Applications	121
6	Applications	123
6.1	Image Manipulation	123
6.2	Vector Data Manipulation	146
6.3	Calibration	157
6.4	Geometry	173
6.5	Image Filtering	201
6.6	Feature Extraction	210
6.7	Stereo	236
6.8	Learning	252
6.9	Segmentation	298
6.10	Miscellaneous	316

WELCOME TO OTB COOKBOOK'S DOCUMENTATION!

Orfeo ToolBox (OTB) is an open-source project for state-of-the-art remote sensing. Built on the shoulders of the open-source geospatial community, it can process high resolution optical, multispectral and radar images at the terabyte scale. A wide variety of applications are available: from ortho-rectification or pansharpening, all the way to classification, SAR processing, and much more!

All of OTB's algorithms are accessible from Monteverdi, QGIS, Python, the command line or C++. Monteverdi is an easy to use visualization tool with an emphasis on hardware accelerated rendering for high resolution imagery (optical and SAR). With it, end-users can visualize huge raw imagery products and access all of the applications in the toolbox. From resource limited laptops to high performance MPI clusters, OTB is available on Windows, Linux and Mac. It is community driven, extensible and heavily documented. Orfeo ToolBox is not a black box!

This is the CookBook documentation for users. If you are new to OTB and Monteverdi, start here. It will go through how to install OTB on your system, how to start using Monteverdi and OTB applications to view and process your data, and recipes on how to accomplish typical remote sensing tasks. Finally, there is also documentation on every application shipped with OTB.

Get started now with the *Installation* chapter.

Get help, share your experience and contribute to the Orfeo-Toolbox project by joining [our community](#) and [users mailing list](#).

For other documentation, be sure to read:

- OTB's website: www.orfeo-toolbox.org
- *OTB Software Guide* for advanced users and developers. The software guide contains documented code examples, descriptions of the ITK pipeline model, multithreading and streaming functionalities, and an introduction to the C++ API.
- *Doxygen*, for exhaustive documentation of the C++ API.

INSTALLATION

We provide different standalone binary packages for OTB-Applications:

- for Windows platform (7 or higher)
- for 64bit Linux distribution
- for MacOS X

Other binaries can be available as packages (OSGeo packages, Debian/Ubuntu packages, OpenSuse packages), however be advised that they may not be up-to-date nor delivered with full features. If you want to build from source or if we don't provide packages for your system, information is available in the [Software Guide](#), in the section "Building from Source".

You can get latest binary packages from our [Download page](#).

2.1 Windows

Windows binary packages are available for Windows 7 or higher. They can be downloaded from [otb download page](#) .

Pick the correct version (32 bit or 64 bit) depending on your system.

Extract the archive and use one of the launchers, they contain all applications and their launchers (both command line and graphical launchers are provided):

- `monteverdi.bat` : A launcher script for Monteverdi
- `mapla.bat` : A launcher script for Mapla
- `otbenv.bat` : A script to initialize the environment for OTB executables
- `bin` : A folder containing application launchers (`otbcli.bat`, `otbgui.bat`) and the DLLs.
- `lib` : A folder containing application DLLs.

The applications can be launched from the Mapla launcher. If you want to use the `otbcli` and `otbgui` launchers, you can initialize a command prompt with `otbenv.bat`.

2.1.1 Notes:

- You must have "Visual C++ Redistributable for Visual Studio 2015" installed for using this package. It can be downloaded freely from [microsoft](#)

2.2 Linux x86_64

We provide a binary package for GNU/Linux x86_64. This package includes all of OTB applications along with commandline and graphical launchers. Download it from [OTB's download page](#).

This package is a self-extractible archive. You may uncompress it with a double-click on the file, or with the command line :

```
chmod +x OTB-5.8.0-Linux64.run
./OTB-5.8.0-Linux64.run
```

Please note that the resulting installation is not meant to be moved, you should uncompress the archive in its final location. Once the archive is extracted, the directory structure is made of:

- `monteverdi.sh` : A launcher script for Monteverdi
- `mapla.sh` : A launcher script for Mapla
- `otbenv.profile` : A script to initialize the environment for OTB executables
- `bin` : A folder containing application launchers (`otbcli.sh`, `otbgui.sh`), Monteverdi and Mapla.
- `lib` : A folder containing all shared libraries and OTB applications.
- `share` : A folder containing common resources and copyright mentions.

In order to run the command line launchers, this package doesn't require any special library that is not present in most modern Linux distributions. The graphical executable (`otbgui` launchers, Monteverdi and Mapla) use the X11 libraries, which are widely used in a lot of distributions :

```
libx11-6 libxext6 libxau6 libxxf86vm1 libxdmcp6 libdrm2
```

Monteverdi also requires the standard graphics libraries `libgl1` and `libglu1`. Make sure you have at least one version of them installed in your system.

The applications can be launched from the Mapla launcher. If you want to use the `otbcli` and `otbgui` launchers, you can initialize your environment with `source otbenv.profile`.

Notes:

- You must use `monteverdi` and `mapla` through `mapla.sh` and `monteverdi.sh` helper scripts in extracted directory.
- The helper scripts for `monteverdi` and `mapla` set required environment variables
- You might be tempted to move “OTB-5.8.0-Linux64” into another location say `/usr/local/` after extraction. But avoid this action!
- To have “OTB-5.8.0-Linux64” installed in `/usr/local` or `/opt` execute “OTB-5.8.0-Linux64.run” in that directory.
- Multiple installation of OTB can exists in same system without one conflicting the other!

2.3 FAQ for Packaging

2.3.1 Q: I am getting an error message...

```
Cannot mix incompatible Qt library (version 0x40806) with this library (version 0x40807)
Aborted
```

A: This is due to a conflict with system Qt4 (usually seen on KDE) and Qt4 + gtk libs in OTB package. The fix you need is to remove those libs from package.

```
cd /path/to/OTB-5.8.0-Linux64
rm -f lib/libQt* && rm -fr lib/gtk
```

2.3.2 Q: Monteverdi and Mapla applications look different from my other applications.

A: By default, Monteverdi, Mapla and otbapplication (otbgui_*) uses a default gtk theme (plastic) which is distributed with the OTB package!. We do this to be on safe side, like when a user does not have gtk libraries installed when using our package.

gtk libraries in the package is the reason why you are getting a default “plastic” look & feel that makes it look different from other GUI applications. To get same look and feel, you can “force” Monteverdi and Mapla to use system GTK.

```
export OTB_USE_SYSTEM_GTK=1
```

And now start `monteverdi.sh` or `mapla.sh` from OTB-5.8.0-Linux64 To get back default behaviour, unset `OTB_USE_SYSTEM_GTK=1` or set `OTB_USE_SYSTEM_GTK=0`

2.4 MacOS X

We provide for MacOS X through a standalone package. This package is a self-extractible archive, quite similar to the Linux one. You may uncompress it with the command line :

```
chmod +x OTB-5.8.0-Linux64.run
./OTB-5.8.0-Linux64.run
```

Once the archive is extracted, you can see OTB-5.8.0-Darwin64 directory in the same directory along with OTB-5.8.0-Darwin64.run

Contents of OTB-5.8.0-Darwin64 is briefly listed below:

- `Monteverdi.app` : A Mac OSX .app for Monteverdi
- `Mapla.app` : A Mac OSX .app for Mapla.
- `bin` : A folder containing application launchers (otbcli.sh, otbgui.sh), monteverdi and mapla binaries.
- `lib` : A folder containing all shared libraries and OTB applications.
- `share` : A folder containing common resources and copyright mentions.

Notes:

- If you want to use the otbcli and otbgui launchers, you must access them via a terminal prompt.
- The OSX .app are provided for monteverdi (viewer) and mapla (application browser).
- You must use monteverdi and mapla through their .app files only.
- You are allowed to move these .app files and refrain from moving or deleting OTB-5.8.0-Darwin64 after extraction. In case you need to have OTB installed in some other directory. Extract the .run file there.

2.5 Other packages

Warning! These packages may not be up-to-date with latest OTB releases. In addition, some features of the library may not be available on every platform. Some of these are not maintained by OTB-team. If you want to get involved in the packaging of OTB for your favourite platform, please contact us through the developer's mailing list: otb-developers@googlegroups.com.

2.5.1 Debian

There are OTB packages for Debian (unstable) since version 5.2.0. OTB Applications packages may be available as Debian packages through APT repositories:

- `otb-bin` for command line applications
- `otb-bin-qt` for Qt applications
- `python-otb` for python applications

Due to license issues, the OTB package built in Debian doesn't contain 6S. As a consequence, the package does not contain the `OpticalCalibration` application.

2.5.2 Ubuntu 12.04 and higher

For Ubuntu 12.04 and higher, OTB Applications packages may be available as Debian packages through APT repositories:

- `otb-bin` for command line applications
- `otb-bin-qt` for Qt applications
- `python-otb` for python applications

Since release 3.14.1, OTB Applications packages are available in the [ubuntugis-unstable](#) repository.

Since release 5.2.0, the Ubuntu packages derive from the Debian packages.

You can add it by using these command-lines:

```
sudo aptitude install add-apt-repository
sudo apt-add-repository ppa:ubuntugis/ubuntugis-unstable
```

After you can run:

```
sudo aptitude install otb-bin otb-bin-qt python-otb
```

If you are using *Synaptic*, you can add the repositories, update and install the packages through the graphical interface.

For further information about Ubuntu packages go to [ubuntugis-unstable](#) launchpad page and click on Read about installing.

`apt-add-repository` will try to retrieve the GPG keys of the repositories to certify the origin of the packages. If you are behind a http proxy, this step won't work and `apt-add-repository` will stall and eventually quit. You can temporarily ignore this error and proceed with the update step. Following this, `aptitude` update will issue a warning about a signature problem. This warning won't prevent you from installing the packages.

2.5.3 OpenSuse 12.X and higher

For OpenSuse 12.X and higher, OTB Applications packages are available through *zypper*.

First, you need to add the appropriate repositories with these command-lines (please replace 11.4 by your OpenSuse version):

```
sudo zypper ar
http://download.opensuse.org/repositories/games/openSUSE_11.4/ Games
sudo zypper ar
http://download.opensuse.org/repositories/Application:/Geo/openSUSE_11.4/ GEO
sudo zypper ar
http://download.opensuse.org/repositories/home:/tzotsos/openSUSE_11.4/ tzotsos
```

Now run:

```
sudo zypper refresh
sudo zypper install OrfeoToolbox
sudo zypper install OrfeoToolbox-python
```

Alternatively you can use the One-Click Installer from the [openSUSE Download page](#) or add the above repositories and install through Yast Package Management.

There is also support for the recently introduced 'rolling' openSUSE distribution named 'Tumbleweed'. For Tumbleweed you need to add the following repositories with these command-lines:

```
sudo zypper ar
http://download.opensuse.org/repositories/games/openSUSE_Tumbleweed/ Games
sudo zypper ar
http://download.opensuse.org/repositories/Application:/Geo/openSUSE_Tumbleweed/ GEO
sudo zypper ar
http://download.opensuse.org/repositories/home:/tzotsos/openSUSE_Tumbleweed/ tzotsos
```

and then add the OTB packages as shown above.

A BRIEF TOUR OF OTB APPLICATIONS

3.1 Introduction

OTB ships with more than 90 ready to use applications for remote sensing tasks. They usually expose existing processing functions from the underlying C++ library, or compose them into high level pipelines. OTB applications allow to:

- combine together two or more functions from the Orfeo Toolbox,
- provide a nice high level interface to handle: parameters, input data, output data and communication with the user.

OTB applications can be launched in different ways, and accessed from different entry points. The framework can be extended, but Orfeo Toolbox ships with the following:

- A command-line launcher, to call applications from the terminal,
- A graphical launcher, with an auto-generated QT interface, providing ergonomic parameters setting, display of documentation, and progress reporting,
- A SWIG interface, which means that any application can be loaded set-up and executed into a high-level language such as Python or Java for instance.
- QGIS plugin built on top of the SWIG/Python interface is available with seamless integration within QGIS.

The OTB Applications are now rich of more than 90 tools, which are listed in the the applications reference documentation, presented in chapter [chap:apprefdoc], page.

3.2 Running the applications

3.2.1 Common framework

All standard applications shared the same implementation and expose automatically generated interfaces. Thus, the command-line interface is prefixed by `otbcli_`, while the Qt interface is prefixed by `otbgui_`. For instance, calling `otbcli_Convert` will launch the command-line interface of the Convert application, while `otbgui_Convert` will launch its GUI.

3.2.2 Using the command-line launcher

The command-line application launcher allows to load an application plugin, to set its parameters, and execute it using the command line. Launching the `otbApplicationLauncherCommandLine` without argument results in the

following help to be displayed:

```
$ otbApplicationLauncherCommandLine
Usage : ./otbApplicationLauncherCommandLine module_name [MODULEPATH] [arguments]
```

The `module_name` parameter corresponds to the application name. The `[MODULEPATH]` argument is optional and allows to pass to the launcher a path where the shared library (or plugin) corresponding to `module_name` is.

It is also possible to set this path with the environment variable `OTB_APPLICATION_PATH`, making the `[MODULEPATH]` optional. This variable is checked by default when no `[MODULEPATH]` argument is given. When using multiple paths in `OTB_APPLICATION_PATH`, one must make sure to use the standard path separator of the target system, which is `:` on Unix, and `;` on Windows.

An error in the application name (i.e. in parameter `module_name`) will make the `otbApplicationLauncherCommandLine` lists the name of all applications found in the available path (either `[MODULEPATH]` and/or `OTB_APPLICATION_PATH`).

To ease the use of the applications, and try avoiding extensive environment customization, ready-to-use scripts are provided by the OTB installation to launch each application, and takes care of adding the standard application installation path to the `OTB_APPLICATION_PATH` environment variable.

These scripts are named `otbcli_<ApplicationName>` and do not need any path settings. For example you can start the Orthorectification application with the script called `otbcli_Orthorectification`.

Launching an application with no or incomplete parameters will make the launcher display a summary of the parameters, indicating the mandatory parameters missing to allow for application execution. Here is an example with the OrthoRectification application:

```
$ otbcli_OrthoRectification

ERROR: Waiting for at least one parameter...

===== HELP CONTEXT =====
NAME: OrthoRectification
DESCRIPTION: This application allows to ortho-rectify optical images from supported sensors.

EXAMPLE OF USE:
otbcli_OrthoRectification -io.in QB_TOULOUSE_MUL_Extract_500_500.tif -io.out QB_Toulouse_ortho.tif

DOCUMENTATION: http://www.orfeo-toolbox.org/Applications/OrthoRectification.html
===== PARAMETERS =====
      -progress                <boolean>          Report progress
MISSING -io.in                <string>          Input Image
MISSING -io.out               <string> [pixel]  Output Image [pixel=uint8/int8/uint16/int32]
      -map                    <string>          Output Map Projection [utm/lambert2/lambert94]
MISSING -map.utm.zone         <int32>         Zone number
      -map.utm.northhem       <boolean>       Northern Hemisphere
      -map.transmercator.falseeasting <float>       False easting
      -map.transmercator.falsenorthing <float>       False northing
      -map.transmercator.scale <float>       Scale factor
      -map.epsg.code         <int32>         EPSG Code
      -outputs.mode          <string>         Parameters estimation modes [auto/autosize]
MISSING -outputs.ulx         <float>         Upper Left X
MISSING -outputs.uly         <float>         Upper Left Y
MISSING -outputs.sizeX      <int32>         Size X
MISSING -outputs.sizeY      <int32>         Size Y
MISSING -outputs.spacingX   <float>         Pixel Size X
MISSING -outputs.spacingY   <float>         Pixel Size Y
      -outputs.isotropic     <boolean>       Force isotropic spacing by default
      -elev.dem              <string>       DEM directory
```

<code>-elev.geoid</code>	<code><string></code>	Geoid File
<code>-elev.default</code>	<code><float></code>	Average Elevation
<code>-interpolator</code>	<code><string></code>	Interpolation [nn/linear/bco]
<code>-interpolator.bco.radius</code>	<code><int32></code>	Radius for bicubic interpolation
<code>-opt.rpc</code>	<code><int32></code>	RPC modeling (points per axis)
<code>-opt.ram</code>	<code><int32></code>	Available memory for processing (in MB)
<code>-opt.gridspacing</code>	<code><float></code>	Resampling grid spacing

For a detailed description of the application behaviour and parameters, please check the application reference documentation presented chapter [chap:apprefdoc], page or follow the DOCUMENTATION hyperlink provided in `otbApplicationLauncherCommandLine` output. Parameters are passed to the application using the parameter key (which might include one or several `.` character), prefixed by a `-`. Command-line examples are provided in chapter [chap:apprefdoc], page .

3.2.3 Using the GUI launcher

The graphical interface for the applications provides a useful interactive user interface to set the parameters, choose files, and monitor the execution progress.

This interface can be activated through the CMake option `.`

This launcher needs the same two arguments as the command line launcher :

```
$ otbApplicationLauncherQt module_name [MODULEPATH]
```

The application paths can be set with the `OTB_APPLICATION_PATH` environment variable, as for the command line launcher. Also, as for the command-line application, a more simple script is generated and installed by OTB to ease the configuration of the module path : to launch the graphical user interface, one will start the `otbgui_Rescale` script.

The resulting graphical application displays a window with several tabs:

- Parameters is where you set the parameters and execute the application.
- Logs is where you see the output given by the application during its execution.
- Progress is where you see a progress bar of the execution (not available for all applications).
- Documentation is where you find a summary of the application documentation.

In this interface, every optional parameter has a check box that you have to tick if you want to set a value and use this parameter. The mandatory parameters cannot be unchecked.

The interface of the application is shown here as an example.

3.2.4 Using the Python interface

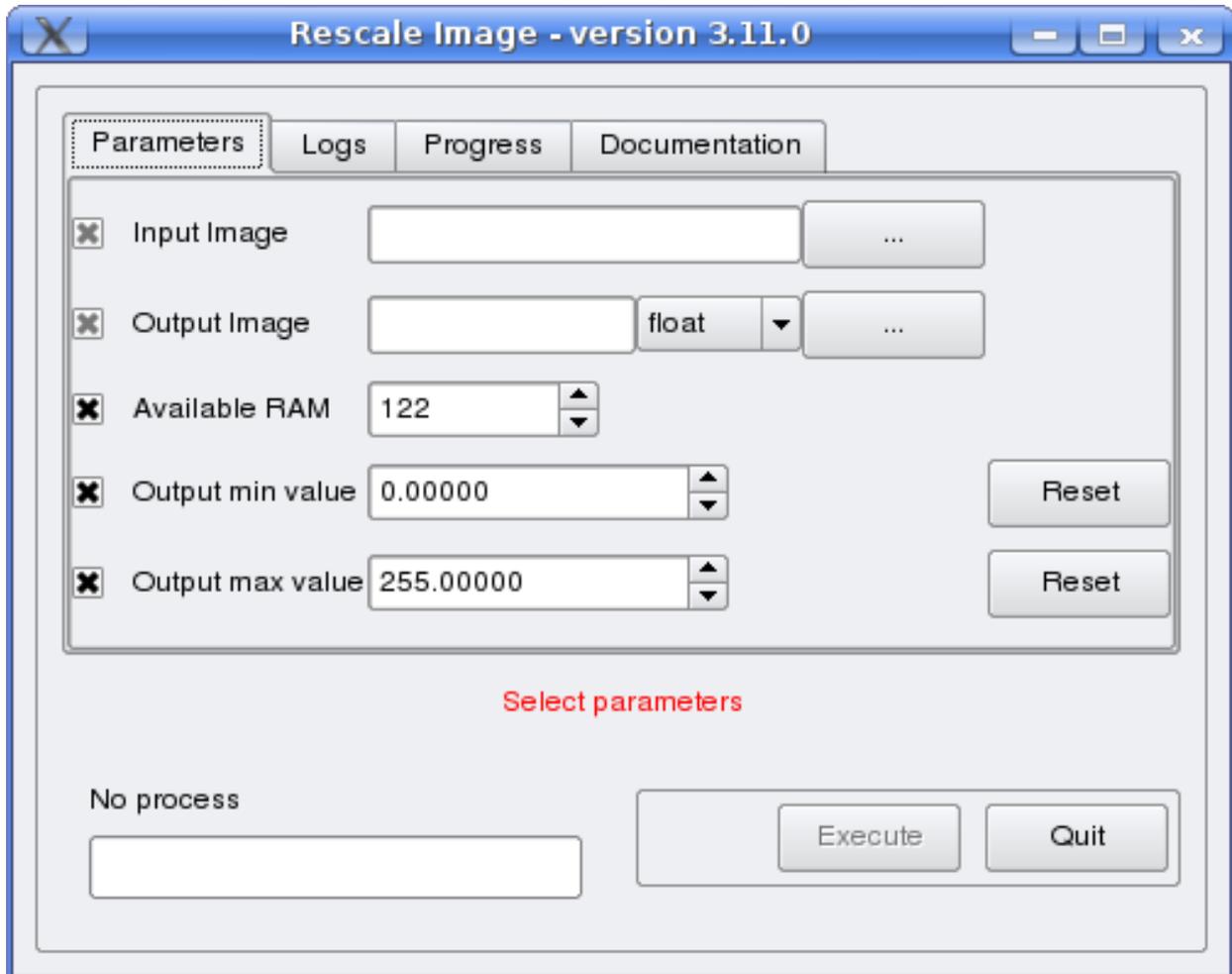
The applications can also be accessed from Python, through a module named `otbApplication`

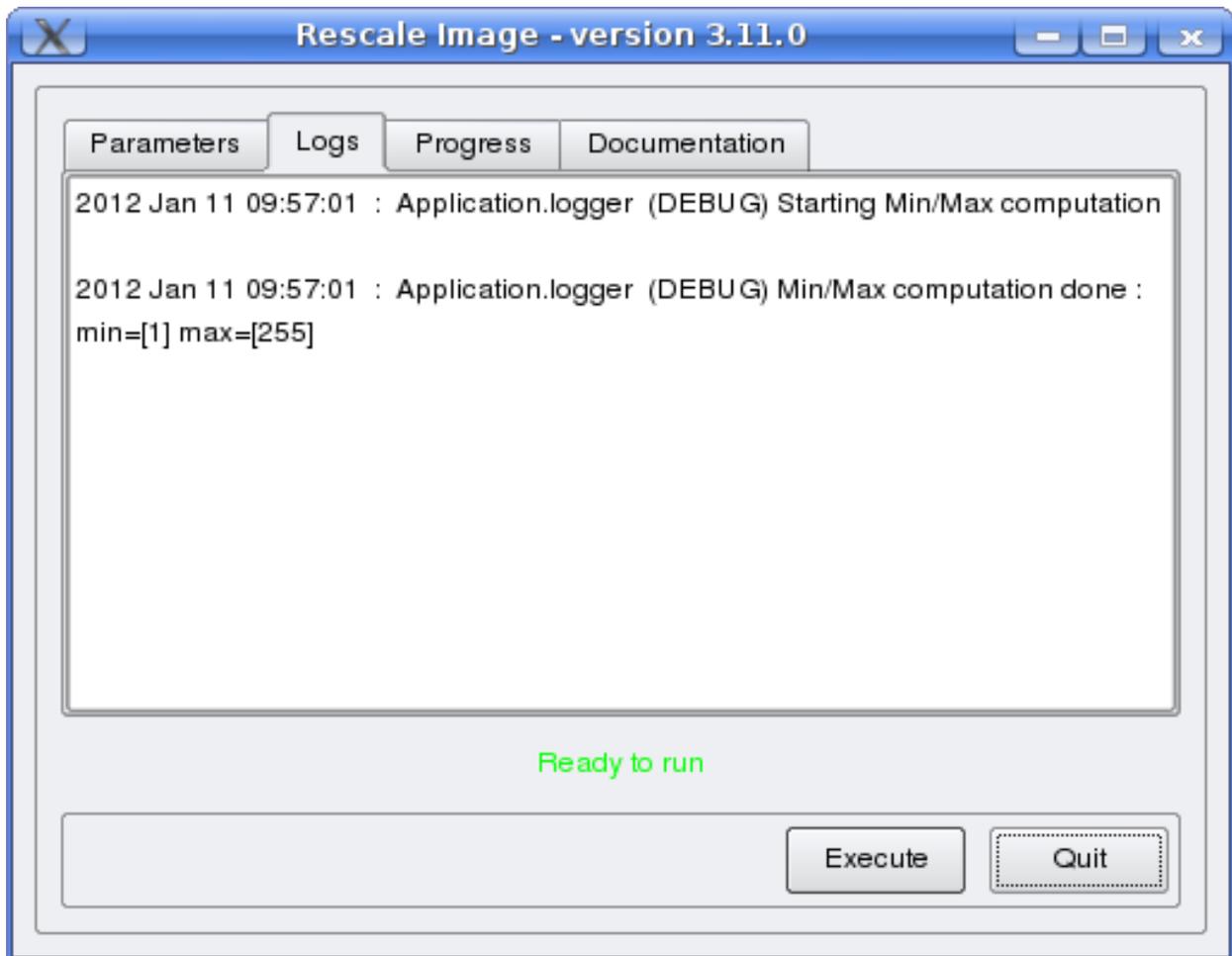
On Unix systems it is typically available in the `/usr/lib/otb/python` directory. You may need to configure the environment variable `PYTHONPATH` to include this directory so that the module becomes available from an Python shell.

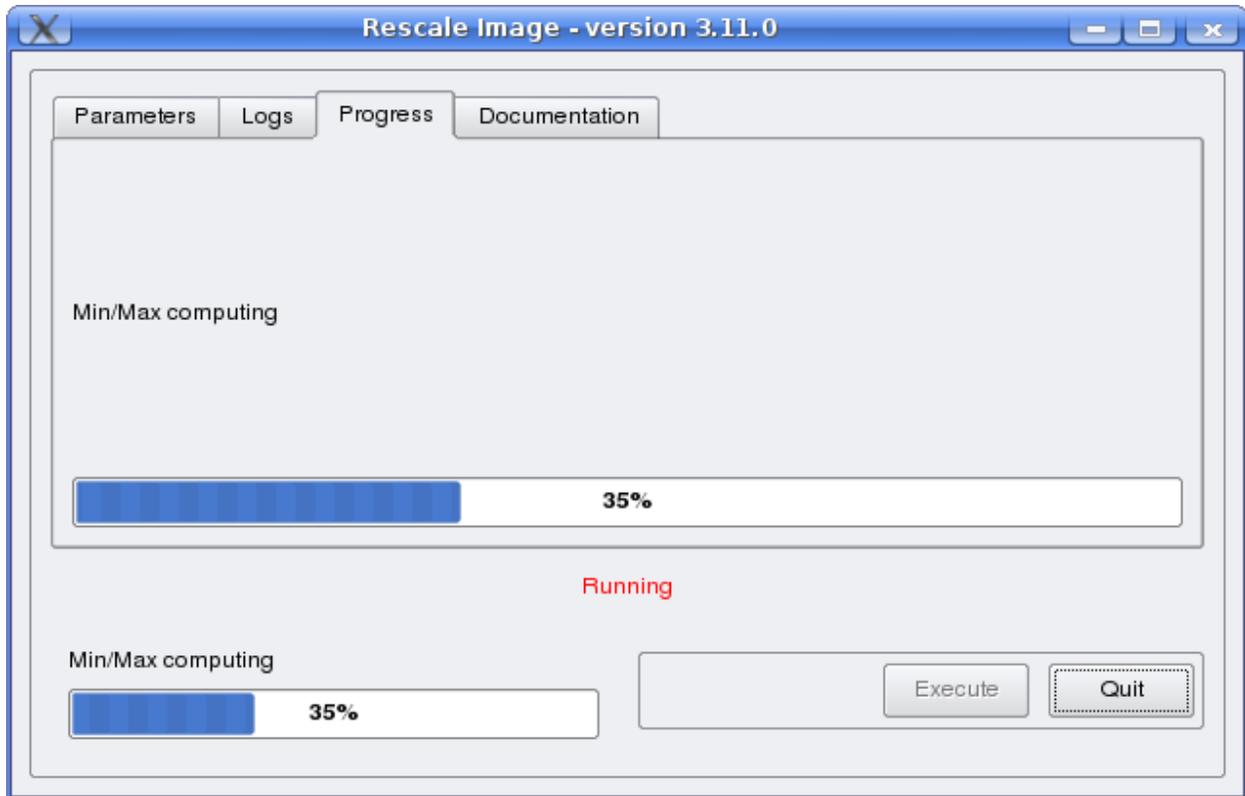
On Windows, you can install the `otb-python` package, and the module will be available from an OSGeo4W shell automatically.

In this module, two main classes can be manipulated :

- `Registry`, which provides access to the list of available applications, and can create applications







- Application, the base class for all applications. This allows to interact with an application instance created by the Registry

As for the command line and GUI launchers, the path to the application modules needs to be properly set with the `OTB_APPLICATION_PATH` environment variable. The standard location on Unix systems is `/usr/lib/otb/applications`. On Windows, the applications are available in the `otb-bin OSGeo4W` package, and the environment is configured automatically so you don't need to tweak `OTB_APPLICATION_PATH`.

Here is one example of how to use Python to run the Smoothing application, changing the algorithm at each iteration.

```
# Example on the use of the Smoothing application
#

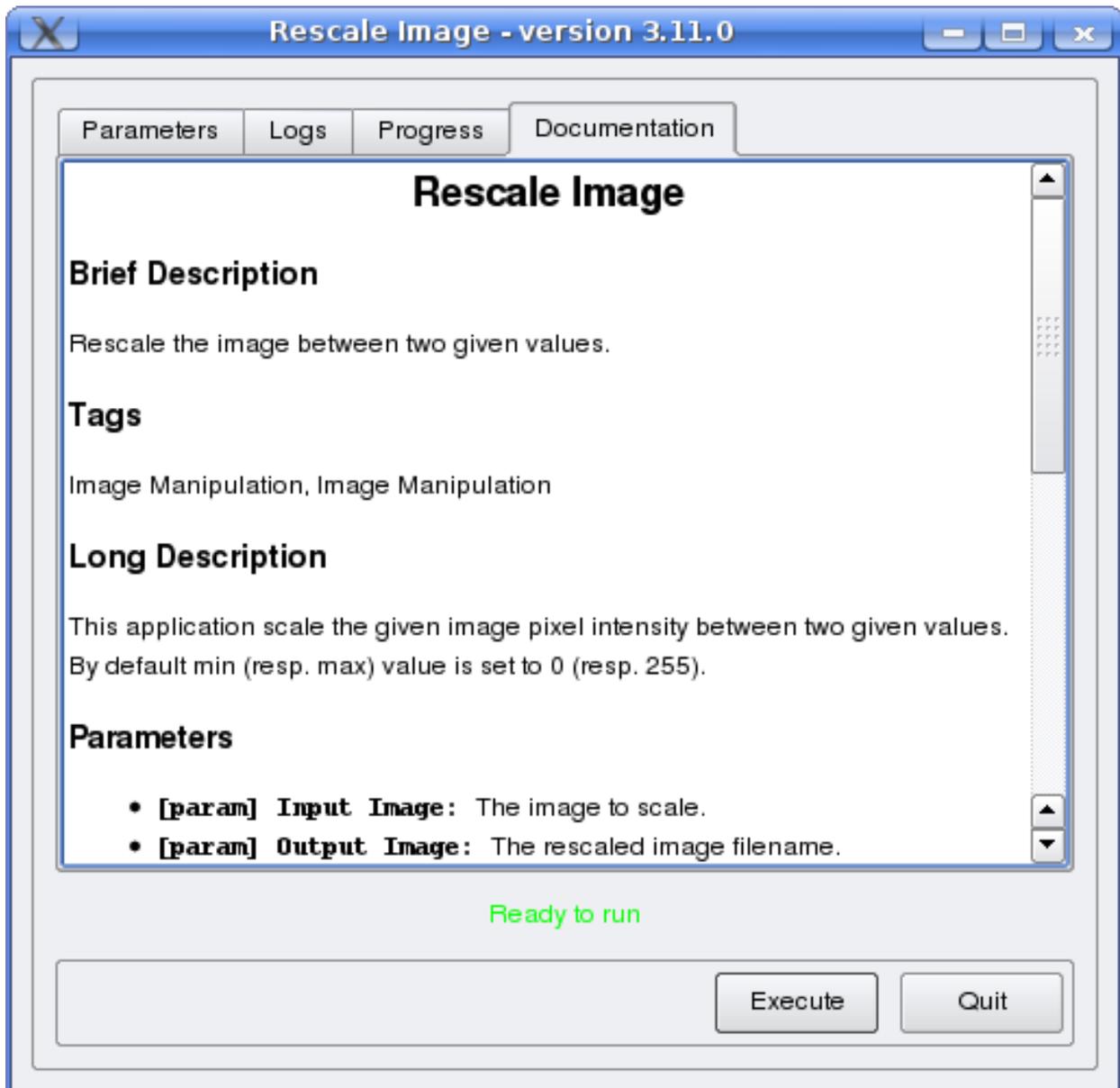
# We will use sys.argv to retrieve arguments from the command line.
# Here, the script will accept an image file as first argument,
# and the basename of the output files, without extension.
from sys import argv

# The python module providing access to OTB applications is otbApplication
import otbApplication

# otbApplication.Registry can tell you what application are available
print "Available applications : "
print str( otbApplication.Registry.GetAvailableApplications() )

# Let's create the application with codename "Smoothing"
app = otbApplication.Registry.CreateApplication("Smoothing")

# We print the keys of all its parameter
print app.GetParametersKeys()
```



```
# First, we set the input image filename
app.SetParameterString("in", argv[1])

# The smoothing algorithm can be set with the "type" parameter key
# and can take 3 values : 'mean', 'gaussian', 'anidif'
for type in ['mean', 'gaussian', 'anidif']:

    print 'Running with ' + type + ' smoothing type'

    # Here we configure the smoothing algorithm
    app.SetParameterString("type", type)

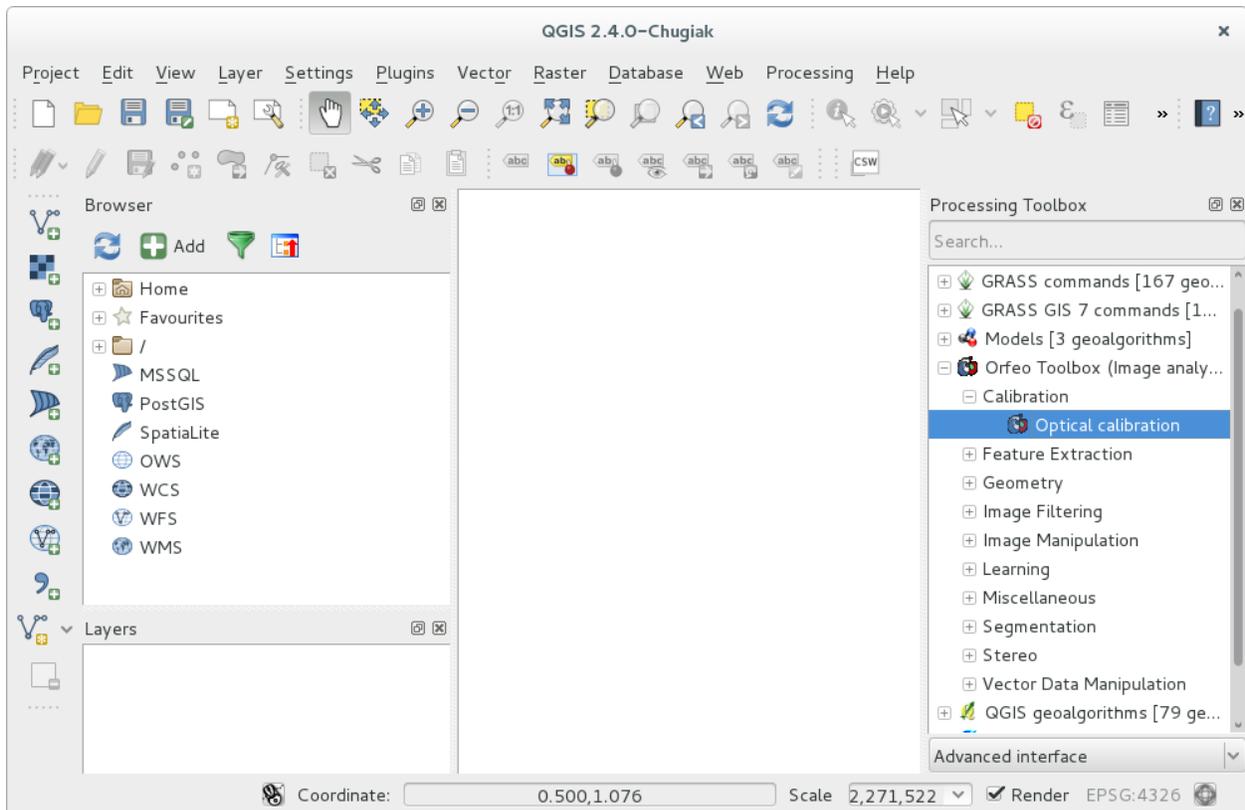
    # Set the output filename, using the algorithm to differentiate the outputs
    app.SetParameterString("out", argv[2] + type + ".tif")

    # This will execute the application and save the output file
    app.ExecuteAndWriteOutput()
```

3.2.5 Using OTB from QGIS

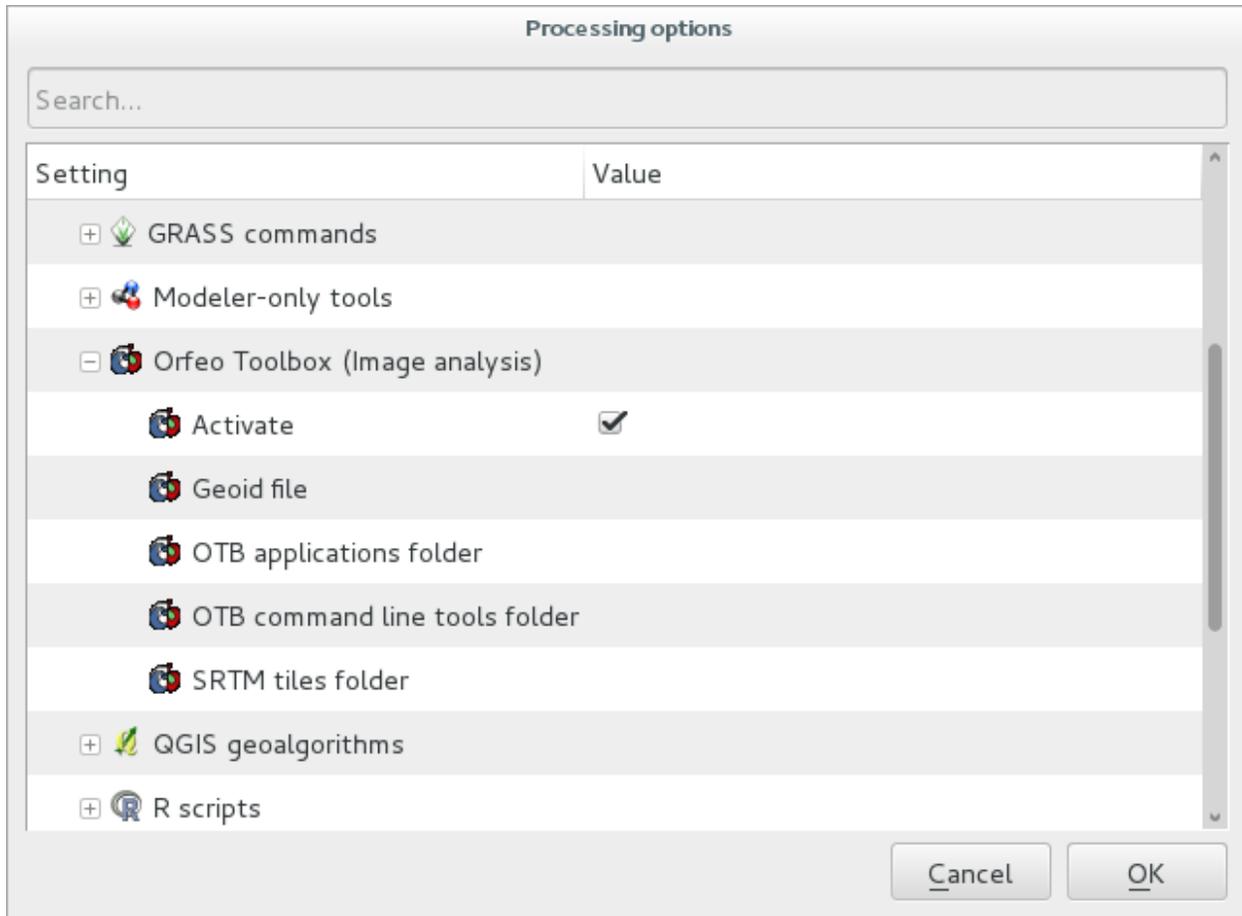
The processing toolbox

OTB applications are available from QGIS. Use them from the processing toolbox, which is accessible with Processing → Toolbox. Switch to “advanced interface” in the bottom of the application widget and OTB applications will be there.



Using a custom OTB

If QGIS cannot find OTB, the “applications folder” and “binaries folder” can be set from the settings in the Processing → Settings → “service provider”.



On some versions of QGIS, if an existing OTB installation is found, the textfield settings will not be shown. To use a custom OTB instead of the existing one, you will need to replace the otbcli, otbgui and library files in QGIS installation directly.

3.3 Advanced applications capabilities

3.3.1 Load/Save OTB-Applications parameters from/to file

Since OTB 3.20, OTB applications parameters can be export/import to/from an XML file using inxml/outxml parameters. Those parameters are available in all applications.

An example is worth a thousand words

```
otbcli_BandMath -il input_image_1 input_image_2
                -exp "abs(im1b1 - im2b1)"
                -out output_image
                -outxml saved_applications_parameters.xml
```

Then, you can run the applications with the same parameters using the output XML file previously saved. For this, you have to use the `inxml` parameter:

```
otbcli_BandMath -inxml saved_applications_parameters.xml
```

Note that you can also overload parameters from command line at the same time

```
otbcli_BandMath -inxml saved_applications_parameters.xml
                -exp "(im1b1 - im2b1)"
```

In this case it will use as mathematical expression `"(im1b1 - im2b1)"` instead of `"abs(im1b1 - im2b1)"`.

Finally, you can also launch applications directly from the command-line launcher executable using the `inxml` parameter without having to declare the application name. Use in this case:

```
otbApplicationLauncherCommandLine -inxml saved_applications_parameters.xml
```

It will retrieve the application name and related parameters from the input XML file and launch in this case the `BandMath` applications.

3.3.2 In-memory connection between applications

Applications are often use as parts of larger processing chains. Chaining applications currently requires to write/read back images between applications, resulting in heavy I/O operations and a significant amount of time dedicated to writing temporary files.

Since OTB 5.8, it is possible to connect an output image parameter from one application to the input image parameter of the next parameter. This results in the wiring of the internal ITK/OTB pipelines together, allowing to perform image streaming between the applications. There is therefore no more writing of temporary images. The last application of the processing chain is responsible for writing the final result images.

In-memory connection between applications is available both at the C++ API level and using the python bindings to the application presented in the [Using the Python interface](#) section.

Here is a Python code sample connecting several applications together:

```
import otbApplications as otb

app1 = otb.Registry.CreateApplication("Smoothing")
app2 = otb.Registry.CreateApplication("Smoothing")
app3 = otb.Registry.CreateApplication("Smoothing")
app4 = otb.Registry.CreateApplication("ConcatenateImages")

app1.IN = argv[1]
app1.Execute()

# Connection between app1.out and app2.in
app2.SetParameterInputImage("in", app1.GetParameterOutputImage("out"))

# Execute call is mandatory to wire the pipeline and expose the
# application output. It does not write image
app2.Execute()

app3.IN = argv[1]

# Execute call is mandatory to wire the pipeline and expose the
# application output. It does not write image
app3.Execute()
```

```
# Connection between app2.out, app3.out and app4.il using images list
app4.AddImageToParameterInputImageList("il", app2.GetParameterOutputImage("out"));
app4.AddImageToParameterInputImageList("il", app3.GetParameterOutputImage("out"));

app4.OUT = argv[2]

# Call to ExecuteAndWriteOutput() both wires the pipeline and
# actually writes the output, only necessary for last application of
# the chain.
app4.ExecuteAndWriteOutput()
```

Note: Streaming will only work properly if the application internal implementation does not break it, for instance by using an internal writer to write intermediate data. In this case, execution should still be correct, but some intermediate data will be read or written.

3.3.3 Parallel execution with MPI

Provided that Orfeo ToolBox has been built with MPI and SPTW modules activated, it is possible to use MPI for massive parallel computation and writing of an output image. A simple call to `mpirun` before the command-line activates this behaviour, with the following logic. MPI writing is only triggered if:

- OTB is built with MPI and SPTW,
- The number of MPI processes is greater than 1,
- The output filename is `.tif` or `.vrt`

In this case, the output image will be divided into several tiles according to the number of MPI processes specified to the `mpirun` command, and all tiles will be computed in parallel.

If the output filename extension is `.tif`, tiles will be written in parallel to a single Tiff file using SPTW (Simple Parallel Tiff Writer).

If the output filename extension is `.vrt`, each tile will be written to a separate Tiff file, and a global `VRT` file will be written.

Here is an example of MPI call on a cluster:

```
$ mpirun -np $nb_procs --hostfile $PBS_NODEFILE \
  otbcli_BundleToPerfectSensor \
  -inp $ROOT/IMG_PHR1A_P_001/IMG_PHR1A_P_201605260427149_ORT_1792732101-001_R1C1.JP2 \
  -inxs $ROOT/IMG_PHR1A_MS_002/IMG_PHR1A_MS_201605260427149_ORT_1792732101-002_R1C1.JP2 \
  -out $ROOT/pxs.tif uint16 -ram 1024

----- JOB INFO 1043196.tu-adm01 -----

JOBID           : 1043196.tu-adm01
USER            : michelj
GROUP           : ctsiap
JOB NAME        : OTB_mpi
SESSION         : 631249
RES REQUESTED  : mem=1575000mb,ncpus=560,place=free,walltime=04:00:00
RES USED        : cpupercent=1553,cput=00:56:12,mem=4784872kb,ncpus=560,vmem=18558416kb,
walltime=00:04:35
BILLING         : 42:46:40 (ncpus x walltime)
QUEUE          : t72h
ACCOUNT        : null
JOB EXIT CODE   : 0
```

```
----- END JOB INFO 1043196.tu-adm01 -----
```

One can see that the registration and pan-sharpening of the panchromatic and multi-spectral bands of a Pleiades image has been split among 560 cpus and took only 56 seconds.

Note that this MPI parallel invocation of applications is only available for command-line calls to OTB applications, and only for images output parameters.

MONTEVERDI

Monteverdi is a satellite image viewer. Its main features are:

- **Performance:** Navigate instantly in full size satellite images thanks to its hardware accelerated rendering engine. Compose tiles or compare multiple images in a stack with rapid cycling and shader effects.
- **Sensor geometry support:** View raw images directly in sensor geometry! Resampling is handled by the GPU through texture mapping. OTB automagically handles coordinates mapping between actors and viewport geometries.
- **Powerful:** Access to all processing application from OTB. Orthorectification, optical calibration, classification, SAR processing, and much more!

4.1 GUI : what does it look like ?



This is Monteverdi's main window where the different functionalities are reachable:

1. Main menu

2. Top toolbar
3. Image displaying
4. Right side dock
5. Stack layer

4.1.1 Main menu

The main menu is made up of four items. The main one is the File item, from which you can: open a image, load the otb applications, and finally quit. The Edit item lets the user change his/her preferences. The view item is intended to let the user display or hide different parts of the main window. Finally, the Help item lets the user know the 'About' information of the software, and also can display an useful keymap.

4.1.2 Top toolbar

The top toolbar is made up of ten icons; from left to right:

1. open one or more image(s)
2. zoom in
3. zoom out
4. zoom to full extent
5. zoom to layer extent
6. zoom to full resolution
7. gives/changes the current projection, used as reference of the view
8. selects the effect to be applied to the selected layer : chessboard, local contrast, local translucency, normal, spectral angle, swipe (horizontal and vertical)
9. a parameter used for the following effects : chessboard, local contrast, local translucency, spectral angle
10. a parameter used for the following effects : local contrast, spectral angle

4.1.3 Image displaying

This part of the main window is intended to display the images loaded by the user. There are many nice keyboard shortcuts or mouse tricks that let the user have a better experience in navigating throughout the loaded images. These shortcuts and tricks are given within the Help item of the main menu, by clicking Keymap; here is a short list of the most useful ones :

The classical ones:

- CTRL+O = Open file(s)
- CTRL+Q = Quit application

In the image displaying part:

- Mouse drag = Scroll view
- CTRL+Mouse drag = Quick scroll view (rending is done after releasing CTRL key)
- CTRL+Mouse wheel = Zoom in out
- - or - = Zoom in out

In the layer stack part:

- SHIFT+Page Up = Move layer to top of stack
- SHIFT+Page Down = Move layer to bottom of stack
- Delete = Delete selected layer
- SHIFT+Delete = Delete all layers

4.1.4 Right side dock

The dock on the right side is divided into four tabs :

- Quicklook : gives the user a degraded view of the whole extent, letting him/her easily select the area to be displayed
- Histogram : gives the user information about the value distribution of the selected channels. By clicking the mouse's left button, user can sample their values.
- Color Setup : lets the user map the image channels to the RGB channels. Also lets him/her set the alpha parameter (translucency).
- Color dynamics : lets the user change the displaying dynamics of a selected image. For each RGB channel (each mapped to an image channel), the user can decide how the pixel range of a selected image will be shortcut before being rescaled to 0-255 : either by setting the extremal values, or by setting the extremal quantiles.

Each tab is represented by the figures below ([fig:quickhisto] [fig:colorsetdyn]).

4.1.5 Layer stack

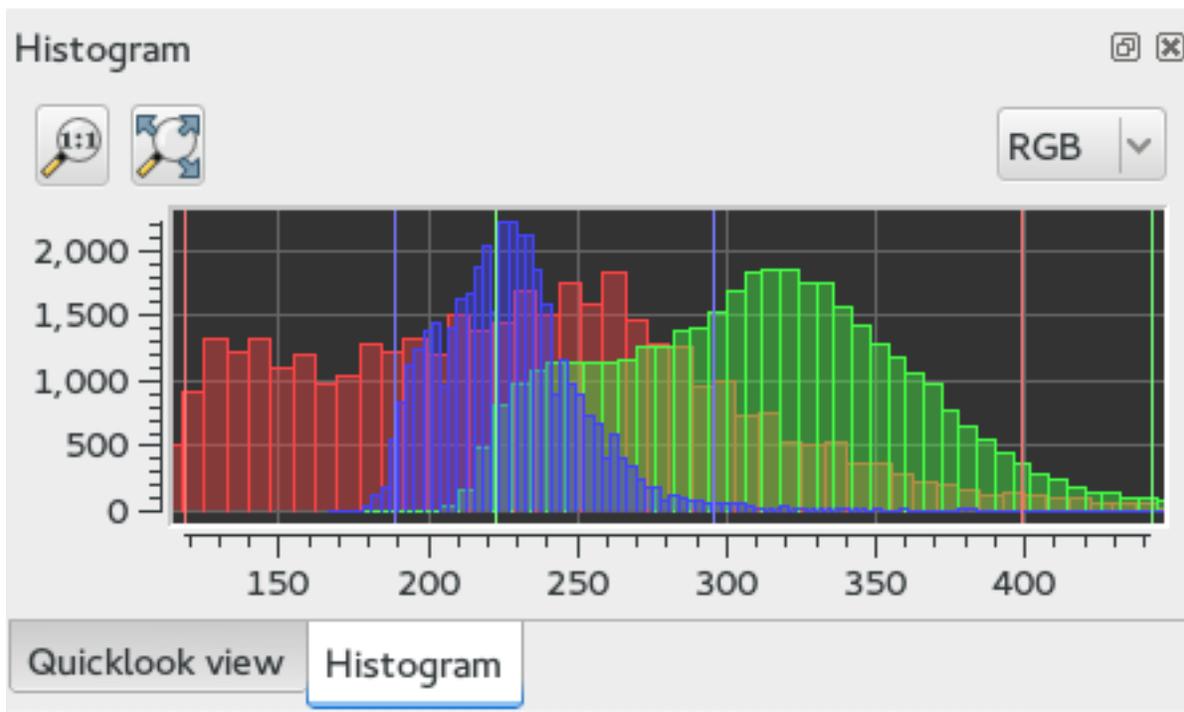
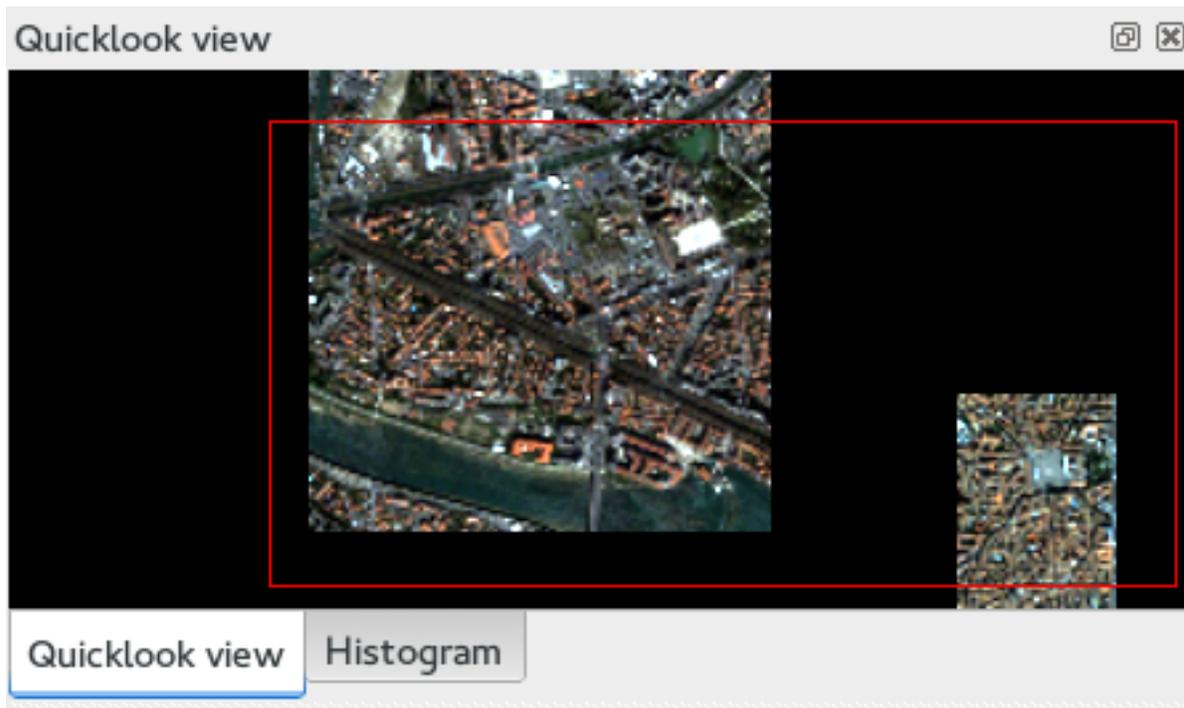
The layer stack is made up of one list of layers located beneath six icons. The list of layers gives the user some information about the loaded images: projection, resolution (if available), name, and effect applied to the images (see top toolbar subsection). If the user moves the mouse over the displayed images, they will get more information:

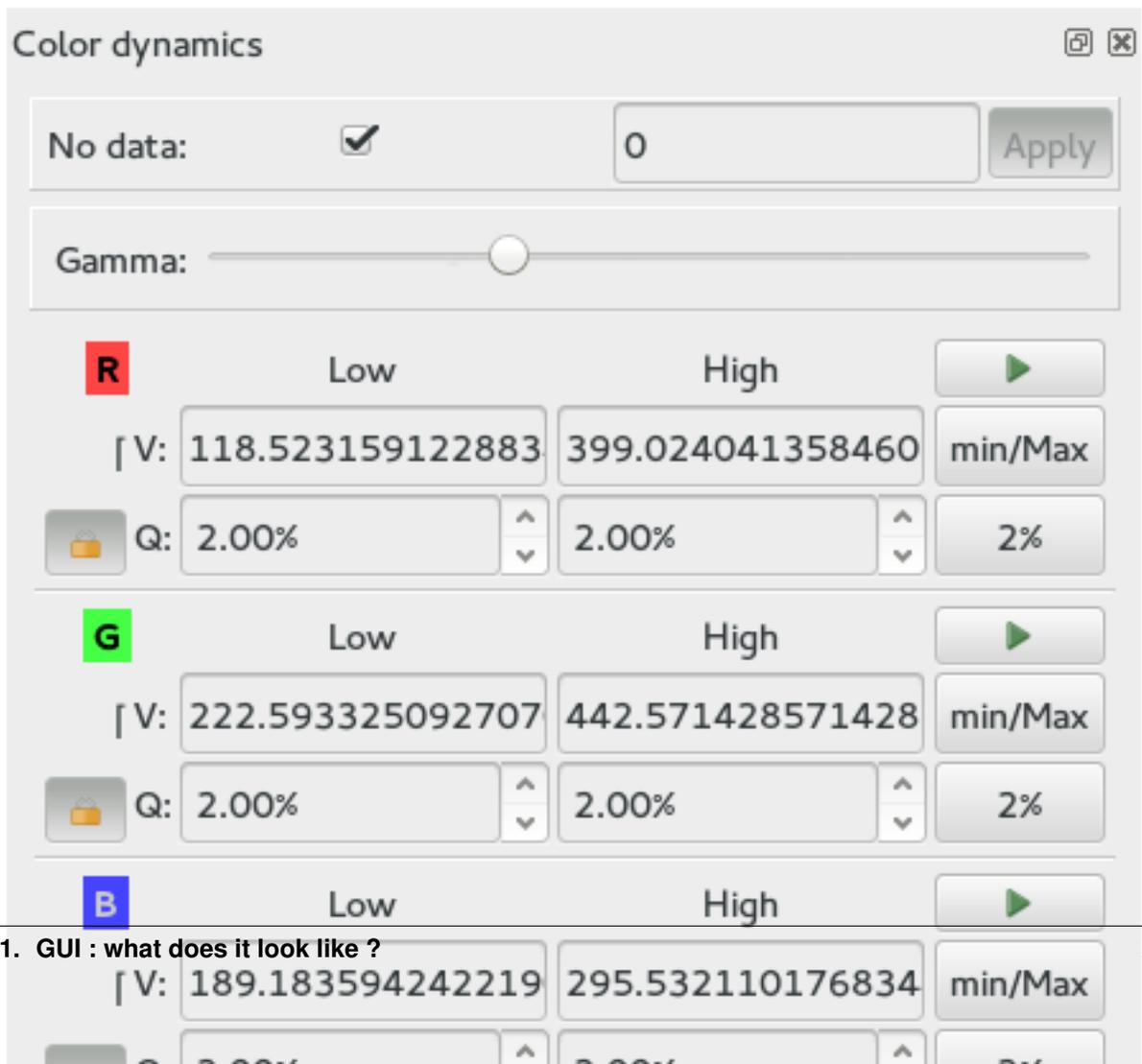
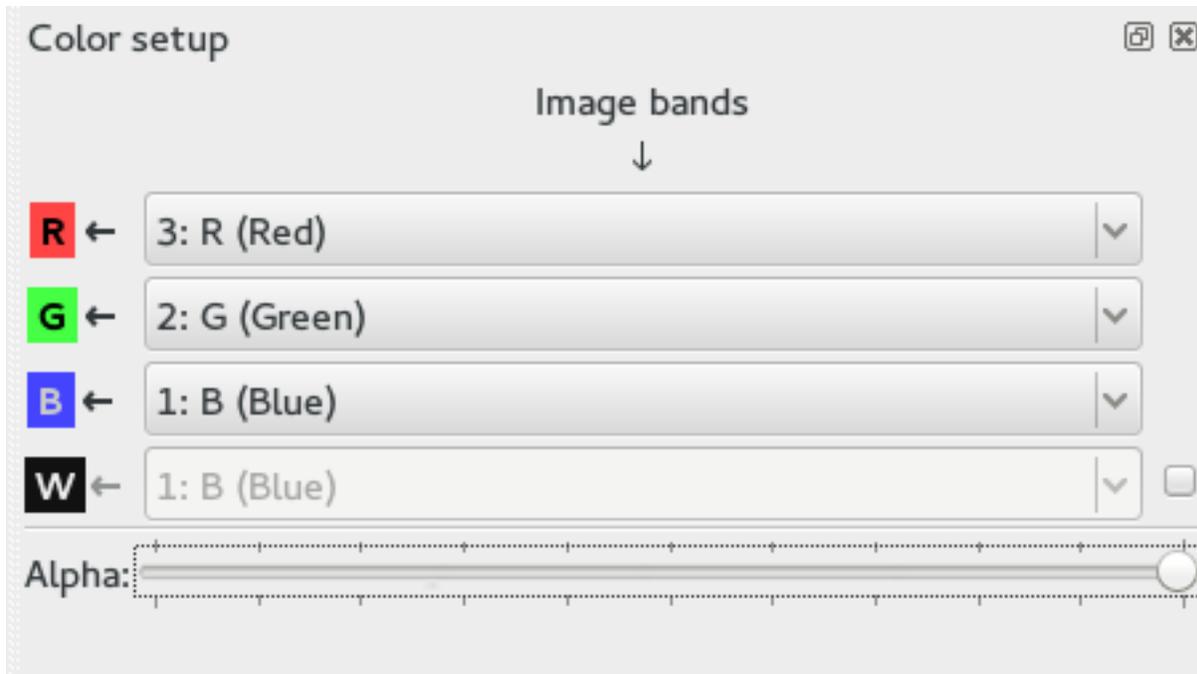
- (i,j) : pixel index
- (Red Green Blue) : original image pixel values from channel mapped to the RGB ones.
- (X,Y) : pixel position

Concerning the six icons, from left to right:

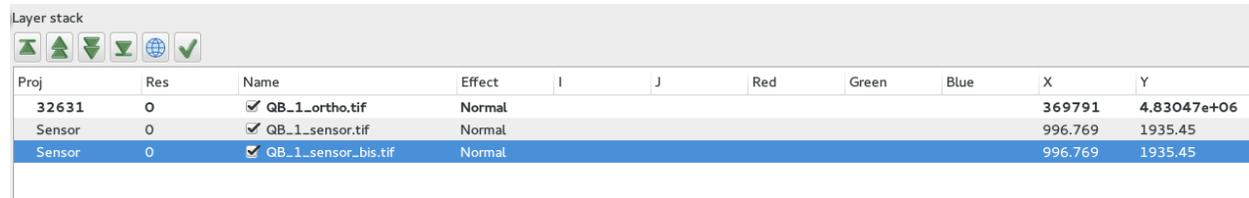
- 1st : moves the selected layer to the top of the stack
- 2nd : moves the selected layer up within the stack
- 3rd : moves the selected layer down within the stack
- 4th : moves the selected layer to the bottom of the stack
- 5th : use selected layer as projection reference
- 6th : applies all display settings (color-setup, color-dynamics, shader and so forth) of selected layer to all other layers

The layer stack is represented in the figure below ([fig:layerstack]) :





Layer stack



Proj	Res	Name	Effect	I	J	Red	Green	Blue	X	Y
32631	0	<input checked="" type="checkbox"/> QB_1_ortho.tif	Normal						369791	4.83047e+06
Sensor	0	<input checked="" type="checkbox"/> QB_1_sensor.tif	Normal						996.769	1935.45
Sensor	0	<input checked="" type="checkbox"/> QB_1_sensor_bis.tif	Normal						996.769	1935.45

4.2 Examples

With , it is also possible to interactively load otb-applications and use them to process images. For that purpose, the user just has to load otb-applications by clicking on the Main menu, File/Load OTB-Applications (or by simply using the shortcut CTRL+A). The figure below ([fig:applications]) represents the otb-applications loading window. The applications are arranged in thematic functionalities; the user can also quickly find the wanted application by typing its name in the dedicated field at the top of the loading window.

4.2.1 Optical calibration

In order to perform an optical calibration, launch the Optical calibration application (shortcut CTRL+A). We are going to use this application to perform a TOA (Top Of Atmosphere) conversion, which consists in converting the DN pixel values into spectral radiance (in W/m2/steradians/micrometers). Once the application is launched, the user must fill the required fields in (in, out, gainbias.txt -gain and bias values in a txt file-, solarillumination.txt -solar illumination values in watt/m2/micron for each band in a txt file-, and so on... refer to the documentation of the application).

- Note : if OTB (on which is based) is able to parse the metadata of the image to be calibrated, then some of the fields will be automatically filled in.

In the figure below ([fig:OC]), by taking a look at the layer stack, one can notice that the values of the calibrated image are now expressed in spectral radiance.

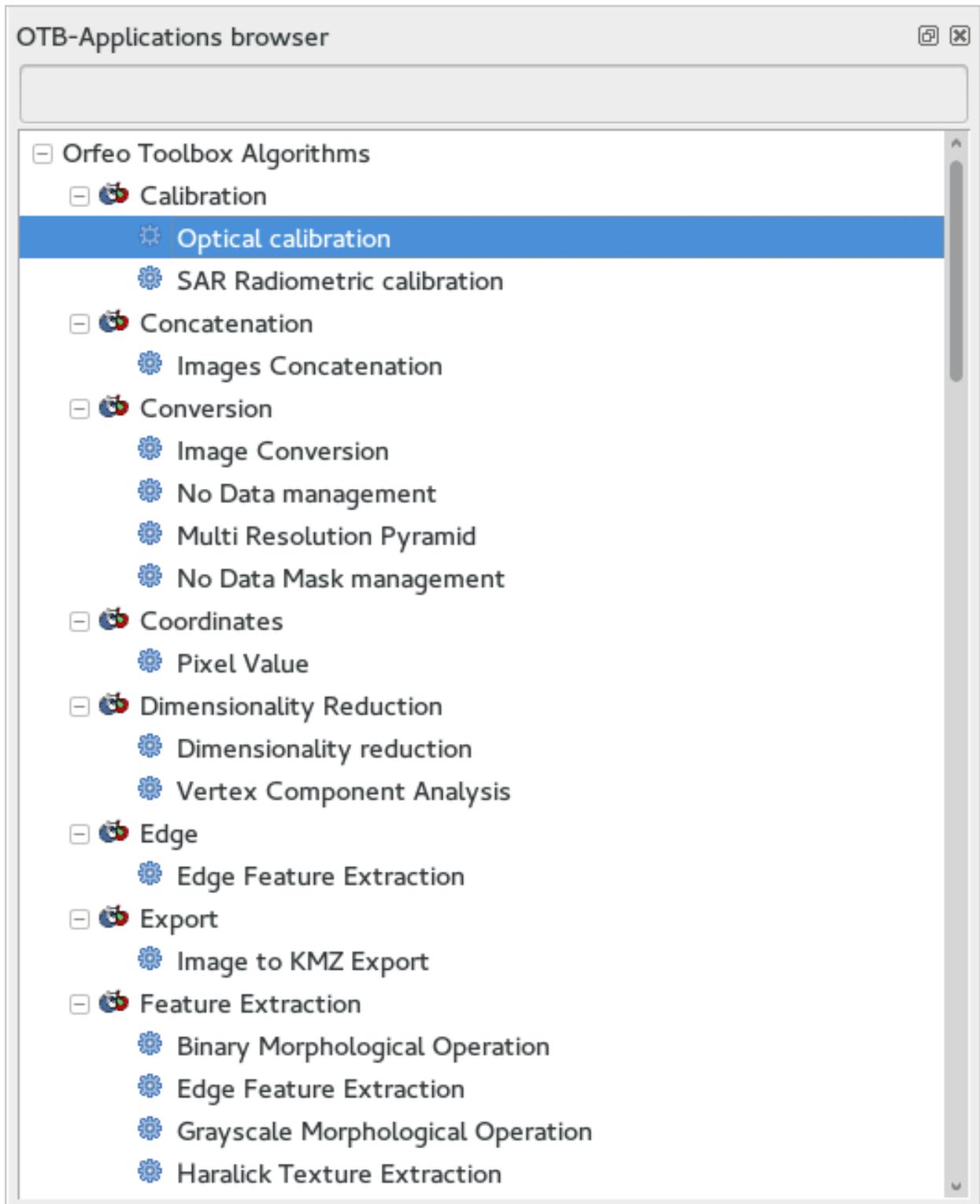
4.2.2 BandMath

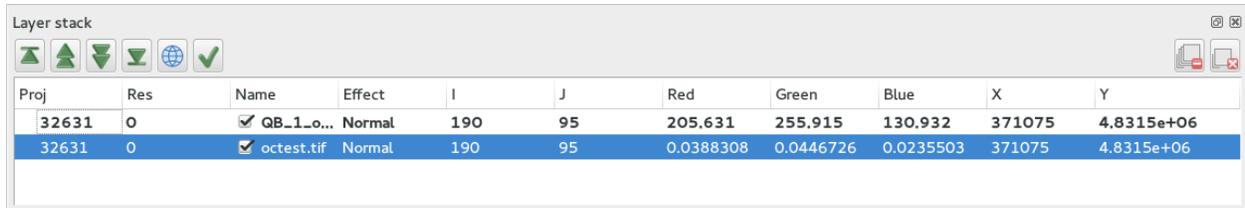
BandMath application is intended to apply mathematical operations on pixels (launch it with shortcut CTRL+A). In this example, we are going to use this application to change the dynamics of an image, and check the result by looking at histogram tab, in the right side dock. The formula used is the following : $im1b1 \times 1000$. In the figures below ([fig:BM]), one can notice that the mode of the distribution is located at position 356.0935, whereas in the transformed image, the mode is located at position 354737.1454, that's to say 1000 times farther away approximately (the cursors aren't placed exactly at the same position in the screenshots).

4.2.3 Segmentation

Now, let's use the segmentation application (launch it with shortcut CTRL+A). We let the user take a look at the application's documentation; let's simply say that as we wish we could display the segmentation with , we must tell the application to output the segmentation in raster format. Thus, the value of the mode option must be set to raster. The following figure ([fig:seg12]) shows the original image and the labels image.

Gray colors aren't very convenient for visualizing a segmentation. That's why we are going to use another application, the ColorMapping one (launch it with the shortcut CTRL+A as usual). There are many ways to use this application (see the documentation for more details). We wish we could colour the segmentation so that color difference between adjacent regions is maximized. For this purpose, we can use the method optimal (set the value of this option to optimal). The figure below ([fig:seg3]) shows the result of such colorization.





Now it should be nice to superimpose this colorization with the original image to assess the quality of the segmentation. provides the user a very simple way to do it. Once the two images are loaded in and that the original image is placed on the top of the stack, the user just has to select the translucency layer effect and set the size of the exploration circle to convenience. The figure below ([fig:seg4]) shows the result of such colorization. We encourage the reader to test the other layer effects.

4.2.4 Polarimetry

In this example, we are going to use three applications :

- the first one is SARDecompositions. This application is used to compute the HaA decomposition. It takes as inputs three complex channels from bands HH HV and VV.
- the second one is SplitImage. Indeed, the previous application had produced an output image made up of three channels, H a and A, and we wish to focus on the H parameter (entropy). So we let this application split this image into three one-band-images.
- the last one is ColorMapping. The entropy image has values ranging from 0 to 1, and they can be easily displayed by . But since we have a nice visualizing tool in hand, we wish we could go a little bit further. Here comes the application ColorMapping. It is going to be used with the following parameter settings:
 - method = continuous. This parameters tells the application to use a gradient of colors to represent the entropy image.
 - method.continuous.lut = hot. We specify here the kind of gradient to be used : low values in black, high ones in white, and intermediate ones in red/orange/yellow...
 - method.continuous.min = 0 and method.continuous.max = 1. Here, the gradient of colors must be adjusted to the dynamic of the entropy image (note: it is theoretically known that in HaA decomposition, H ranges from 0 to 1. Generally speaking, the histogram of can also be used for this purpose).

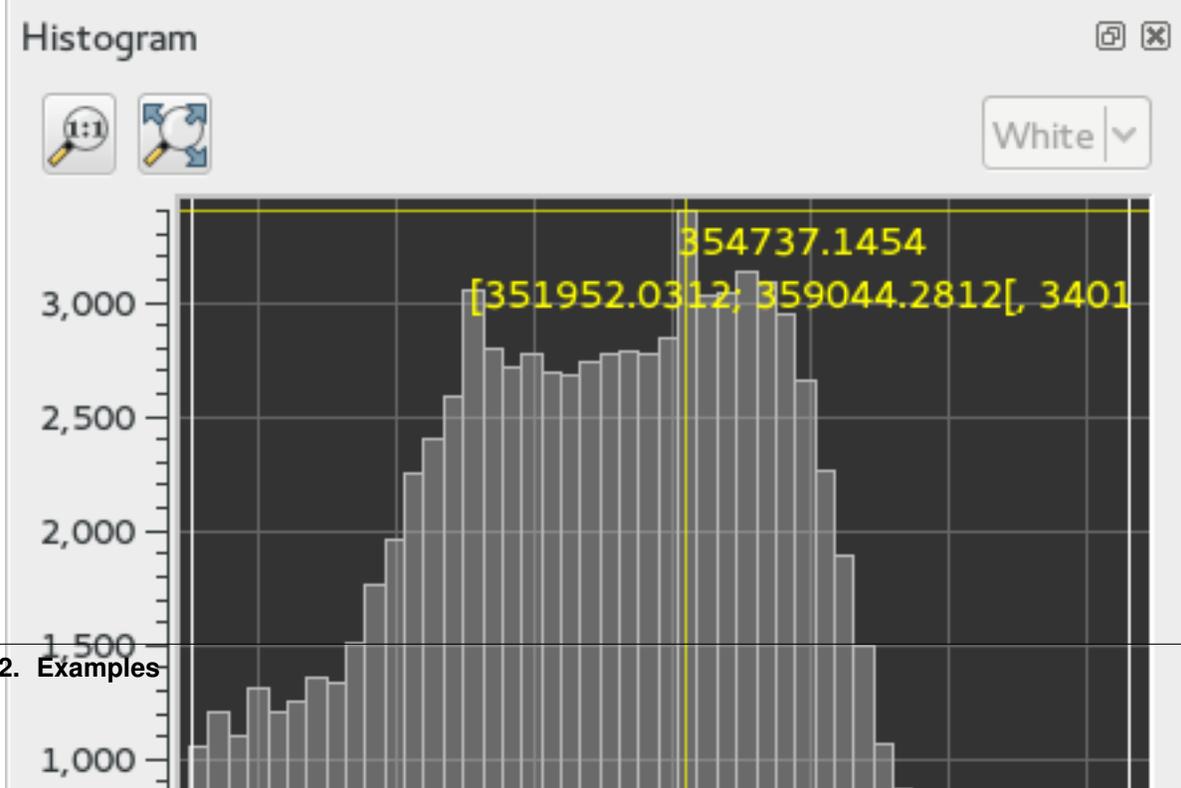
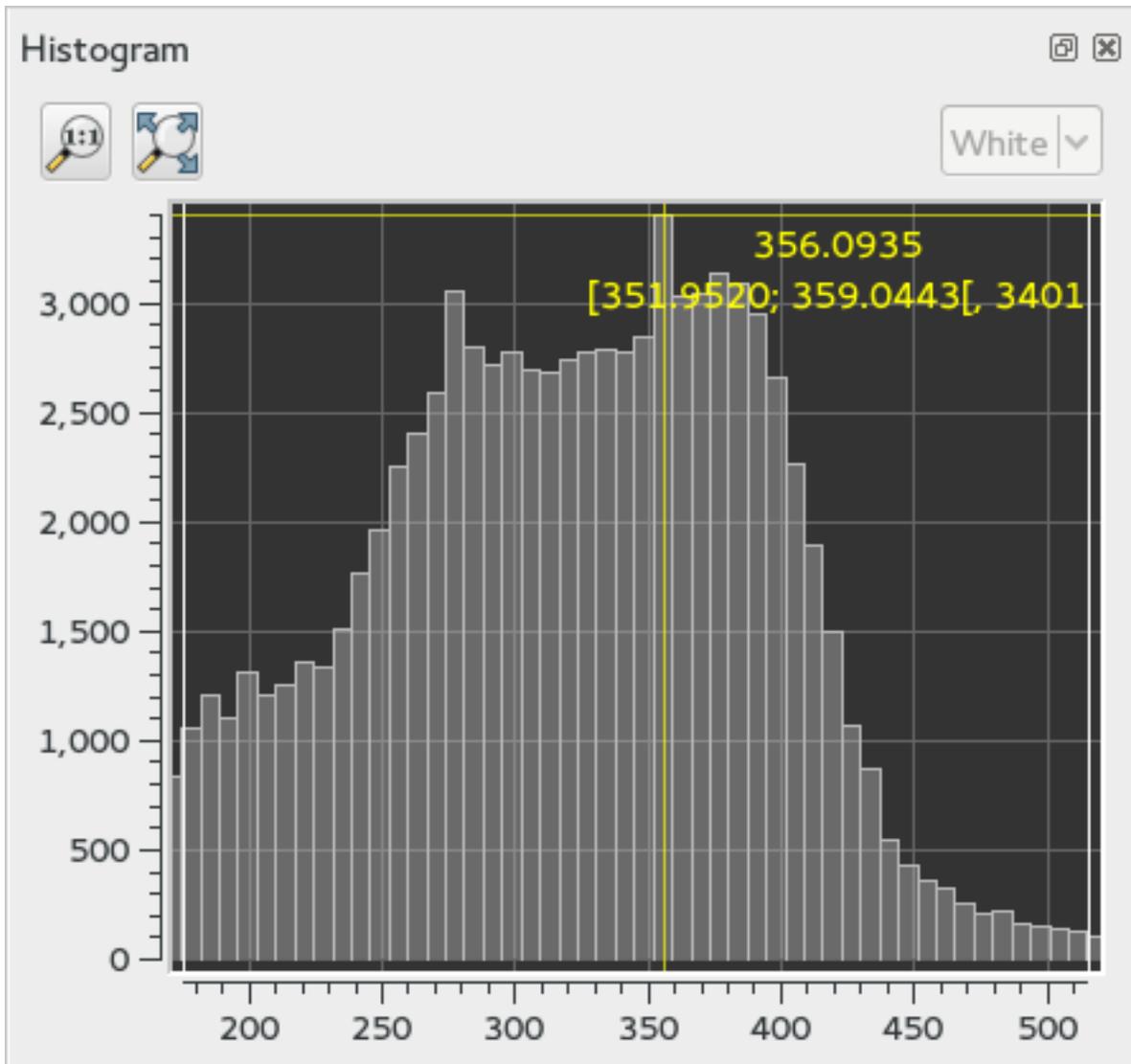
In the figure below ([fig:pol1]), we show the obtained result, with the local contrast layer effect.

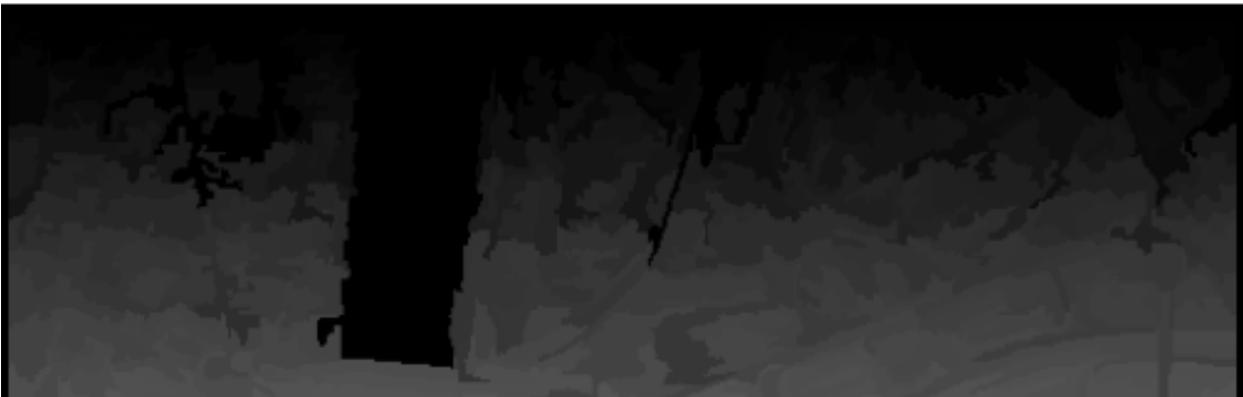
4.2.5 Pansharpening

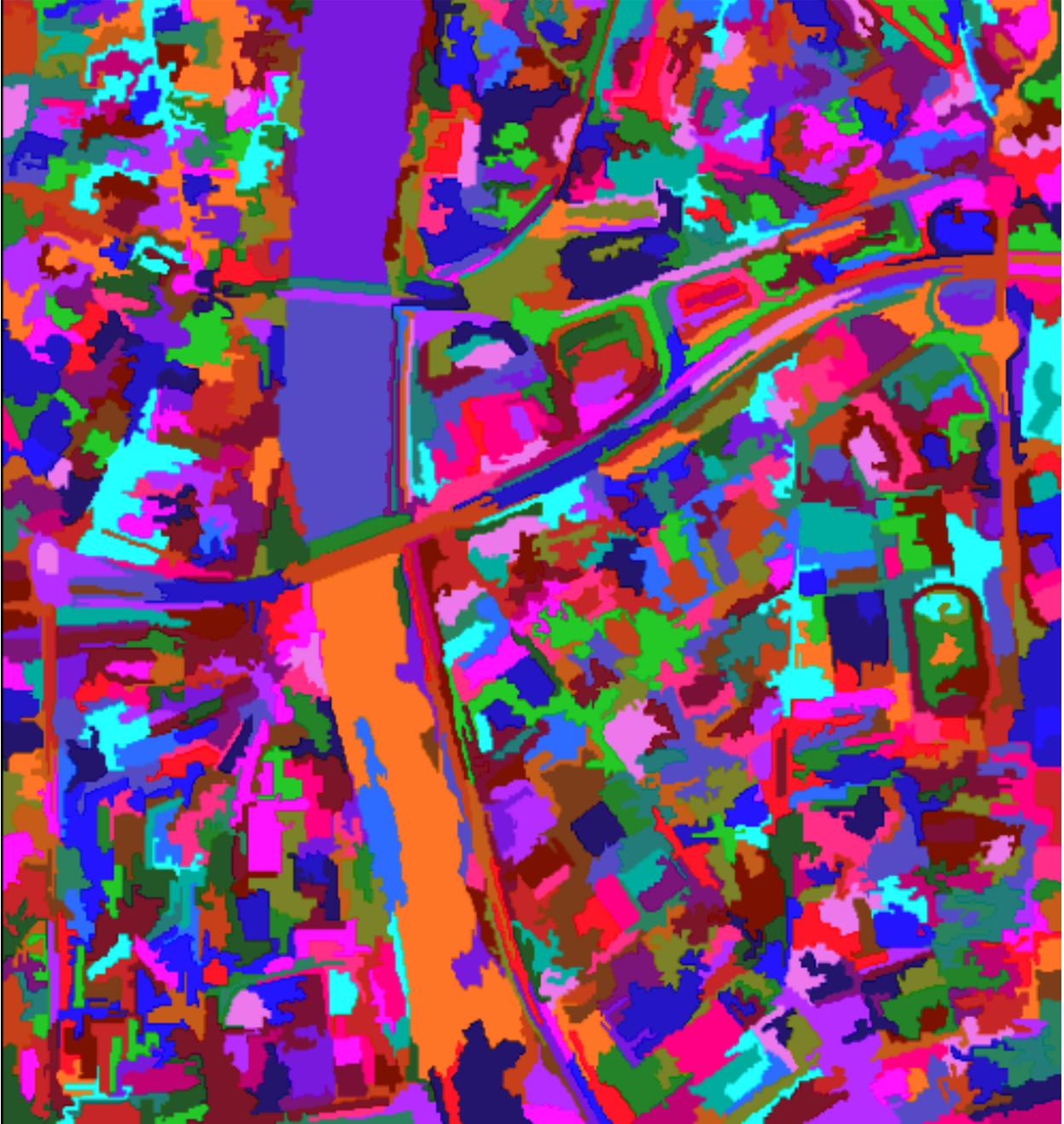
Finally, let's try a last example with the Pansharpening application (launch it with shortcut CTRL+A). The fields are quite easy to fill in : this application needs a panchromatic image, a XS image, and an output image. These images are represented in the figures below ([fig:ps12] and [fig:ps3]):

Now, in order to inspect the result properly, these three images are loaded in . The pansharpened image is placed to the top of the stack layer, and different layer effects are applied to it :

- in figure [fig:ps4] : chessboard effect, to compare the result with the XS image.
- in figure [fig:ps5] : translucency effect, to compare the result with the panchromatic image.







Layer stack

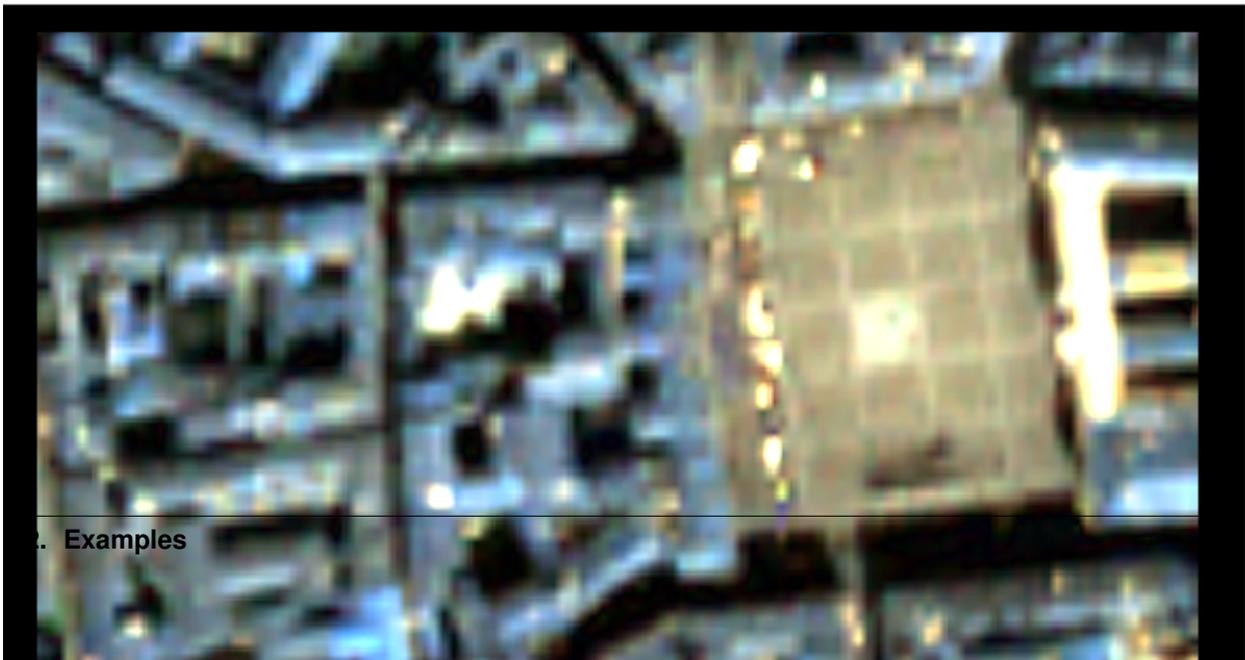
Proj	Res	Name	Effect	I	J	Red	Green	Blue	X	Y
32631	0	GB_1... Local transl...	Local transl...	167	148	219.611	289.671	143.615	371015	4.83136e+06
32631	0	cmtest.tif	Normal	167	148	106	90	205	371015	4.83136e+06

Position: 167, 148 [N 43.624; E 1.40126; 0] [R: 219.611; G: 289.671; B: 143.615] Zoom Level: 1.32215:1

Layer stack

Proj	Res	Name	Effect	I	J	Red	Green	Blue	X	Y
Unknown	0	haasplitt... Local contrast	Local contrast	75	120	255	131	0	76.197	121.204

Position: 75, 120 [R: 255; G: 131; B: 0] Zoom Level: 3.655:1



4. Examples



File Edit View Help
 Proj 32631 (GB_Tou... Layer FX Chessboard Size: 150

Layer stack

Proj	Res	Name	Effect	I	J	Red	Green	Blue	X	Y
32631	0	psitest.tif	Chessboard			373984			4.82917e+06	
32631	0	GB_Toulouse_Ortho_PAN.tif	Normal			373984			4.82917e+06	
32631	0	GB_Toulouse_Ortho_XS.tif	Normal			373984			4.82917e+06	

Position [N 43.6048 ; E 1.43856 ; 0] Zoom Level 1.462:1

File Edit View Help
 Proj 32631 (GB_Tou... Layer FX Local translucency Size: 80

Layer stack

Proj	Res	Name	Effect	I	J	Red	Green	Blue	X	Y
32631	0	psitest.tif	Local translucency	374	111	283	409	301	374375	4.82912e+06
32631	0	GB_Toulouse_Ortho_PAN.tif	Normal	374	111	339	339	339	374375	4.82912e+06
32631	0	GB_Toulouse_Ortho_XS.tif	Normal	374	111	275	398	293	374375	4.82912e+06

Position [374, 111] [N 43.6044 ; E 1.44341 ; 0] [R: 283 ; G: 409 ; B: 301] Zoom Level 1.462:1

4.2.6 Conclusion

The images used in this documentation can be found in the OTB-Data repository (<https://git.orfeo-toolbox.org/otb-data.git>):

- in OTB-Data/Input :
 - QB_TOULOUSE_MUL_Extract_500_500.tif and QB_Toulouse_Ortho_XS_ROI_170x230.tif (GUI presentation)
 - RSAT_imagery_HH.tif RSAT_imagery_HV.tif RSAT_imagery_VV.tif (polarimetry example)
 - QB_Toulouse_Ortho_PAN.tif QB_Toulouse_Ortho_XS.tif (pansharpening example)
- in OTB-Data/Input/mv2-test : QB_1_ortho.tif

RECIPES

This chapter presents guideline to perform various remote sensing and image processing tasks with either , or both. Its goal is not to be exhaustive, but rather to help the non-developer user to get familiar with these two packages, so that he can use and explore them for his future needs.

5.1 From raw image to calibrated product

This section presents various pre-processing tasks that are presented in a classical order to obtain a calibrated, pan-sharpened image.

5.1.1 Optical radiometric calibration

In remote sensing imagery, pixel values are called DN (for Digital Numbers) and can not be physically interpreted and compared: they are influenced by various factors such as the amount of light flowing trough the sensor, the gain of the detectors and the analogic to numeric converter.

Depending on the season, the light and atmospheric conditions, the position of the sun or the sensor internal parameters, these DN can drastically change for a given pixel (apart from any ground change effects). Moreover, these effects are not uniform over the spectrum: for instance aerosol amount and type has usually more impact on the blue channel.

Therefore, it is necessary to calibrate the pixel values before any physical interpretation is made out of them. In particular, this processing is mandatory before any comparison of pixel spectrum between several images (from the same sensor), and to train a classifier without dependence to the atmospheric conditions at the acquisition time.

Calibrated values are called surface reflectivity, which is a ratio denoting the fraction of light that is reflected by the underlying surface in the given spectral range. As such, its values lie in the range $[0, 1]$. For convenience, images are often stored in thousandth of reflectivity, so that they can be encoded with an integer type. Two levels of calibration are usually distinguished:

- The first level is called *Top Of Atmosphere (TOA)* reflectivity. It takes into account the sensor gain, sensor spectral response and the solar illumination.
- The second level is called *Top Of Canopy (TOC)* reflectivity. In addition to sensor gain and solar illumination, it takes into account the optical thickness of the atmosphere, the atmospheric pressure, the water vapor amount, the ozone amount, as well as the composition and amount of aerosol gasses.

This transformation can be done either with **OTB Applications** or with **Monteverdi** . Sensor-related parameters such as gain, date, spectral sensitivity and sensor position are seamlessly read from the image metadata. Atmospheric parameters can be tuned by the user. Supported sensors are :

- Pleiades

- SPOT5
- QuickBird
- Ikonos
- WorldView-1
- WorldView-2
- Formosat

The *OpticalCalibration* application allows to perform optical calibration. The mandatory parameters are the input and output images. All other parameters are optional. By default the level of calibration is set to TOA (Top Of Atmosphere). The output images are expressed in thousandth of reflectivity using a 16 bits unsigned integer type.

A basic TOA calibration task can be performed with the following command:

```
otbcli_OpticalCalibration -in input_image -out output_image
```

A basic TOC calibration task can be performed with the following command:

```
otbcli_OpticalCalibration -in input_image -out output_image -level toc
```

5.1.2 Pan-sharpening

Because of physical constrains on the sensor design, it is difficult to achieve high spatial and spectral resolution at the same time : a better spatial resolution means a smaller detector, which in turns means lesser optical flow on the detector surface. On the contrary, spectral bands are obtained through filters applied on the detector surface, that lowers the optical flow, so that it is necessary to increase the detector size to achieve an acceptable signal to noise ratio.

For these reasons, many high resolution satellite payload are composed of two sets of detectors, which in turns delivers two different kind of images :

- The multi-spectral (XS) image, composed of 3 to 8 spectral bands containing usually blue, green, red and near infra-red bands at a given resolution (usually from 2.8 meters to 2 meters).
- The panchromatic (PAN) image, which is a grayscale image acquired by a detector covering a wider part of the light spectrum, which allows to increase the optical flow and thus to reduce pixel size. Therefore, resolution of the panchromatic image is usually around 4 times lower than the resolution of the multi-spectral image (from 46 centimeters to 70 centimeters).

It is very frequent that those two images are delivered side by side by data providers. Such a dataset is called a bundle. A very common remote sensing processing is to fuse the panchromatic image with the multi-spectral one so as to get an image combining the spatial resolution of the panchromatic image with the spectral richness of the multi-spectral image. This operation is called pan-sharpening.

This fusion operation requires two different steps :

1. The multi-spectral (XS) image is zoomed and registered to the panchromatic image,
2. A pixel-by-pixel fusion operator is applied to the co-registered pixels of the multi-spectral and panchromatic image to obtain the fused pixels.

Using either **OTB Applications** or modules from **Monteverdi** , it is possible to perform both steps in a row, or step-by-step fusion, as described in the above sections.

The *BundleToPerfectSensor* application allows to perform both steps in a row. Seamless sensor modelling is used to perform zooming and registration of the multi-spectral image on the panchromatic image. In the case of a Pléiades bundle, a different approach is used : an affine transform is used to zoom the multi-spectral image and apply a residual translation. This translation is computed based on metadata about the geometric processing of the bundle.

This zooming and registration of the multi-spectral image over the panchromatic image can also be performed by the *Superimpose* application.

After the registration step, a simple pan-sharpening is applied, according to the following formula:

$$PXS(i, j) = \frac{PAN(i, j)}{PAN_{smooth}(i, j)} \cdot XS(i, j)$$

Where i and j are pixels indices, PAN is the panchromatic image, XS is the multi-spectral image and PAN_{smooth} is the panchromatic image smoothed with a kernel to fit the multi-spectral image scale.

Here is a simple example of how to use the *BundleToPerfectSensor* application:

```
otbcli_BundleToPerfectSensor -inp pan_image -inxs xs_image -out output_image
```

There are also optional parameters that can be useful for this tool:

- The `-elev` option allows to specify the elevation, either with a DEM formatted for OTB (`-elev.dem` option, see section [ssec:dem]) or with an average elevation (`-elev.default` option). Since registration and zooming of the multi-spectral image is performed using sensor-models, it may happen that the registration is not perfect in case of landscape with high elevation variation. Using a DEM in this case allows to get better registration.
- The `-lmSpacing` option allows to specify the step of the registration grid between the multi-spectral image and panchromatic image. This is expressed in amount of panchromatic pixels. A lower value gives a more precise registration but implies more computation with the sensor models, and thus increase the computation time. Default value is 10 pixels, which gives sufficient precision in most of the cases.
- The `-mode` option allows to select the registration mode for the multi-spectral image. The `default` mode uses the sensor model of each image to create a generic “MS to Pan” transform. The `phr` mode uses a simple affine transform (which doesn’t need an elevation source nor a registration grid).

Pan-sharpening is a quite heavy processing requiring a lot of system resource. The `-ram` option allows you to limit the amount of memory available for the computation, and to avoid overloading your computer. Increasing the available amount of RAM may also result in better computation time, seems it optimises the use of the system resources. Default value is 256 Mb.

Figure 5 : Pan-sharpened image using Orfeo ToolBox.

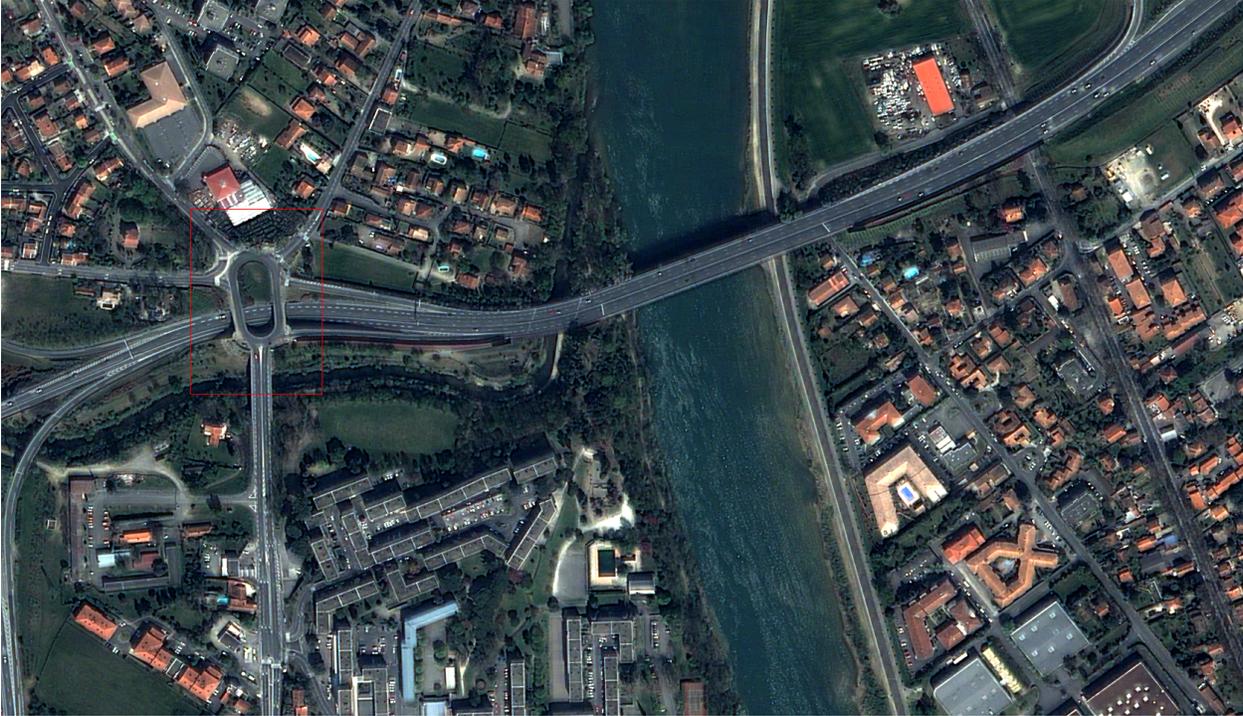
Please also note that since registration and zooming of the multi-spectral image with the panchromatic image relies on sensor modelling, this tool will work only for images whose sensor models is available in **Orfeo Toolbox** (see *Ortho-rectification and map projections* for a detailed list). It will also work with ortho-ready products in cartographic projection.

5.1.3 Digital Elevation Model management

A Digital Elevation Model (DEM) is a georeferenced image (or collection of images) where each pixel corresponds to a local elevation. DEM are useful for tasks involving sensor to ground and ground to sensor coordinate transforms, like during ortho-rectification (see *Ortho-rectification and map projections*). These transforms need to find the intersection between the line of sight of the sensor and the earth geoid. If a simple spheroid is used as the earth model, potentially high localisation errors can be made in areas where elevation is high or perturbed. Of course, DEM accuracy and resolution have a great impact on the precision of these transforms.

Two main available DEM, free of charges, and with worldwide cover, are both delivered as 1 degree by 1 degree tiles:

- The **Shuttle Radar topographic Mission (SRTM)** is a 90 meters resolution DEM, obtained by radar interferometry during a campaign of the Endeavour space shuttle from NASA in 2000.



- The **Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER)** is a 30 meters resolution DEM obtained by stereoscopic processing of the archive of the ASTER instrument.

The **Orfeo Toolbox** relies on **OSSIM** capabilities for sensor modelling and DEM handling. Tiles of a given DEM are supposed to be located within a single directory. General elevation support is also supported from GeoTIFF files.

Whenever an application or **Monteverdi** module requires a DEM, the option **elev.dem** allows set the DEM directory. This directory must contains the DEM tiles, either in DTED or SRTM format, either as GeoTIFF files. Subdirectories are not supported.

Depending on the reference of the elevation, you also need to use a geoid to manage elevation accurately. For this, you need to specify a path to a file which contains the geoid. **Geoid** corresponds to the equipotential surface that would coincide with the mean ocean surface of the Earth .

We provide one geoid in the **OTB-Data** repository.

In all applications, the option **elev.geoid** allows to manage the path to the geoid. Finally, it is also possible to use an average elevation in case no DEM is available by using the **elev.default** option.

5.1.4 Ortho-rectification and map projections

There are several level of products available on the remote sensing imagery market. The most basic level often provide the geometry of acquisition (sometimes called the raw geometry). In this case, pixel coordinates can not be directly used as geographical positions. For most sensors (but not for all), the different lines corresponds to different acquisition times and thus different sensor positions, and different rows correspond to different cells of the detector.

The mapping of a raw image so as to be registered to a cartographic grid is called ortho-rectification, and consist in inverting the following effects (at least):

- In most cases, lines are orthogonal to the sensor trajectory, which is not exactly (and in some case not at all) following a north-south axis,

- Depending on the sensor, the line of sight may be different from a Nadir (ground position of the sensor), and thus a projective warping may appear,
- The variation of height in the landscape may result in severe warping of the image.

Moreover, depending on the area of the world the image has been acquired on, different map projections should be used.

The ortho-rectification process is as follows: once an appropriate map projection has been defined, a localisation grid is computed to map pixels from the raw image to the ortho-rectified one. Pixels from the raw image are then interpolated according to this grid in order to fill the ortho-rectified pixels.

Ortho-rectification can be performed either with **OTB Applications** or **Monteverdi**. Sensor parameters and image meta-data are seamlessly read from the image files without needing any user interaction, provided that all auxiliary files are available. The sensor for which **Orfeo Toolbox** supports ortho-rectification of raw products are the following:

- Pleiades
- SPOT5
- Ikonos
- Quickbird
- GeoEye
- WorldView

In addition, GeoTiff and other file format with geographical information are seamlessly read by **Orfeo Toolbox**, and the ortho-rectification tools can be used to re-sample these images in another map projection.

Beware of “ortho-ready” products

There are some image products, called “ortho-ready”, that should be processed carefully. They are actual products in raw geometry, but their metadata also contains projection data :

- a map projection
- a physical origin
- a physical spacing
- and sometimes an orientation angle

The purpose of this projection information is to give an approximate map projection to a raw product. It allows you to display the raw image in a GIS viewer at the (almost) right location, without having to reproject it. Obviously, this map projection is not as accurate as the sensor parameters of the raw geometry. In addition, the impact of the elevation model can't be observed if the map projection is used. In order to perform an ortho-rectification on this type of product, the map projection has to be hidden from **Orfeo Toolbox**.

You can see if a product is an “ortho-ready” product by using tools such as `gdalinfo` or `ReadImageInfo`, and check if the product verifies the 2 following conditions :

- The product is in raw geometry : you should expect the presence of RPC coefficients and a non-empty OSSIM keywordlist.
- The product has a map projection : you should see a projection name with physical origin and spacing.

In that case, you can hide the map projection from the **Orfeo Toolbox** by using *extended* filenames. Instead of using the plain input image path, you append a specific key at the end :

```
"path_to_image?&skipcarto=true"
```

The double quote can be necessary for a successful parsing. More details about the extended filenames can be found in the [wiki page](#) , and also in the [OTB Software Guide](#) .

Ortho-rectification with OTB Applications

The *OrthoRectification* application allows to perform ortho-rectification and map re-projection. The simplest way to use it is the following command:

```
otbcli_OrthoRectification -io.in input_image -io.out output_image
```

In this case, the tool will automatically estimates all the necessary parameters:

- The map projection is set to UTM (a worldwide map projection) and the UTM zone is automatically estimated,
- The ground sampling distance of the output image is computed to fit the image resolution,
- The region of interest (upper-left corner and size of the image) is estimated so as to contain the whole input image extent.

In order to use a Digital Elevation Model (see *Digital Elevation Model management.*) for better localisation performances, one can pass the directory containing the DEM tiles to the application:

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
```

If one wants to use a different map projection, the *-map* option may be used (example with *lambert93* map projection):

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
                          -map lambert93
```

Map projections handled by the application are the following (please note that the ellipsoid is always WGS84):

- UTM : `-map utm` | The UTM zone and hemisphere can be set by the options `-map.utm.zone` and `-map.utm.northhem`.
- Lambert 2 etendu: `-map lambert2`
- Lambert 93: `-map lambert93`
- TransMercator: `-map transmercator` | The related parameters (false easting, false northing and scale factor) can be set by the options `-map.transmercator.falseeasting`, `-map.transmercator.falsenorthing` and `-map.transmercator.scale`
- WGS : `-map wgs`
- Any map projection system with an EPSG code : `-map epsg` | The EPSG code is set with the option `-map.epsg.code`

The group `outputs` contains parameters to set the origin, size and spacing of the output image. For instance, the ground spacing can be specified as follows:

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
                          -map lambert93
                          -outputs.spacingx spx
                          -outputs.spacingy spy
```

Please note that since the y axis of the image is bottom oriented, the y spacing should be negative to avoid switching north and south direction.

A user-defined region of interest to ortho-rectify can be specified as follows:

```
otbcli_OrthoRectification -io.in input_image
                        -io.out output_image
                        -elev.dem dem_dir
                        -map lambert93
                        -outputs.spacingx spx
                        -outputs.spacingy spy
                        -outputs.ulx ul_x_coord
                        -outputs.uly ul_y_coord
                        -outputs.size_x x_size
                        -outputs.size_y y_size
```

Where the `-outputs.ulx` and `-outputs.uly` options allow to specify the coordinates of the upper-left corner of the output image. The `-outputs.size_x` and `-outputs.size_y` options allow to specify the size of the output image.

A few more interesting options are available:

- The `-opt.rpc` option allows to use an estimated RPC model instead of the rigorous SPOT5 model, which speeds-up the processing,
- The `-opt.gridspacing` option allows to define the spacing of the localisation grid used for ortho-rectification. A coarser grid results in speeding-up the processing, but with potential loss of accuracy. A standard value would be 10 times the ground spacing of the output image.
- The `-interpolator` option allows to change the interpolation algorithm between nearest neighbor, linear and bicubic. Default is nearest neighbor interpolation, but bicubic should be fine in most cases.
- The `-opt.ram` option allows to specify the amount of memory available for the processing (in Mb). Default is 256 Mb. Increasing this value to fit the available memory on your computer might speed-up the processing.

5.2 SAR processing

This section describes how to use the applications related to SAR processing.

5.2.1 Calibration

The application `SarRadiometricCalibration` can deal with the calibration of data from four radar sensors: RadarSat2, Sentinel1, COSMO-SkyMed and TerraSAR-X.

Examples :

If `SARimg.tif` is a TerraSAR-X or a COSMO-SkyMed image :

```
otbcli_SarRadiometricCalibration -in SARimg.tif
                                -out SARimg-calibrated.tif
```

If `SARimg.tif` is a RadarSat2 or a Sentinel1 image, it 's possible to specify the look-up table (automatically found in the metadata provided with such image) :

```
otbcli_SarRadiometricCalibration -in SARimg.tif
                                -lut gamma
                                -out SARimg-calibrated.tif
```

For TerraSAR-X (and soon for RadarSat2 and Sentinel1), it is also possible to use a noise LUT to derive calibrated noise profiles :

```
otbcli_SarRadiometricCalibration -in SARimg.tif
                                  -lut gamma -noise 1
                                  -out SARimg-calibrated.tif
```

5.2.2 Despeckle

SAR images are generally corrupted by speckle noise. To suppress speckle and improve the radar image analysis lots of filtering techniques have been proposed. The module implements to well-known despeckle methods: Frost, Lee, Gamma-MAP and Kuan.

Figure ([fig:S1VVdespeckledextract]) shows an extract of a SLC Sentinel1 image, band VV, taken over Cape Verde and the result of the Gamma filter. The following commands were used to produce the despeckled extract :

First, the original image is converted into an intensity one (real part corresponds to band 1, and imaginary part to band 2):

```
otbcli_BandMath -il S1-VV-extract.tif
                -exp im1b1^2+im1b2^2
                -out S1-VV-extract-int.tif
```

Then the intensity image is despeckled with the Gamma-MAP filter :

```
otbcli_Despeckle -in S1-VV-extract-int.tif
                 -filter.gammapad.rad 5
                 -filter.gammapad.nblocks 1
                 -out S1-VV-despeckled-extract.tif
```

The produced images were then rescaled to intensities ranging from 0 to 255 in order to be displayed.

5.2.3 Polarimetry

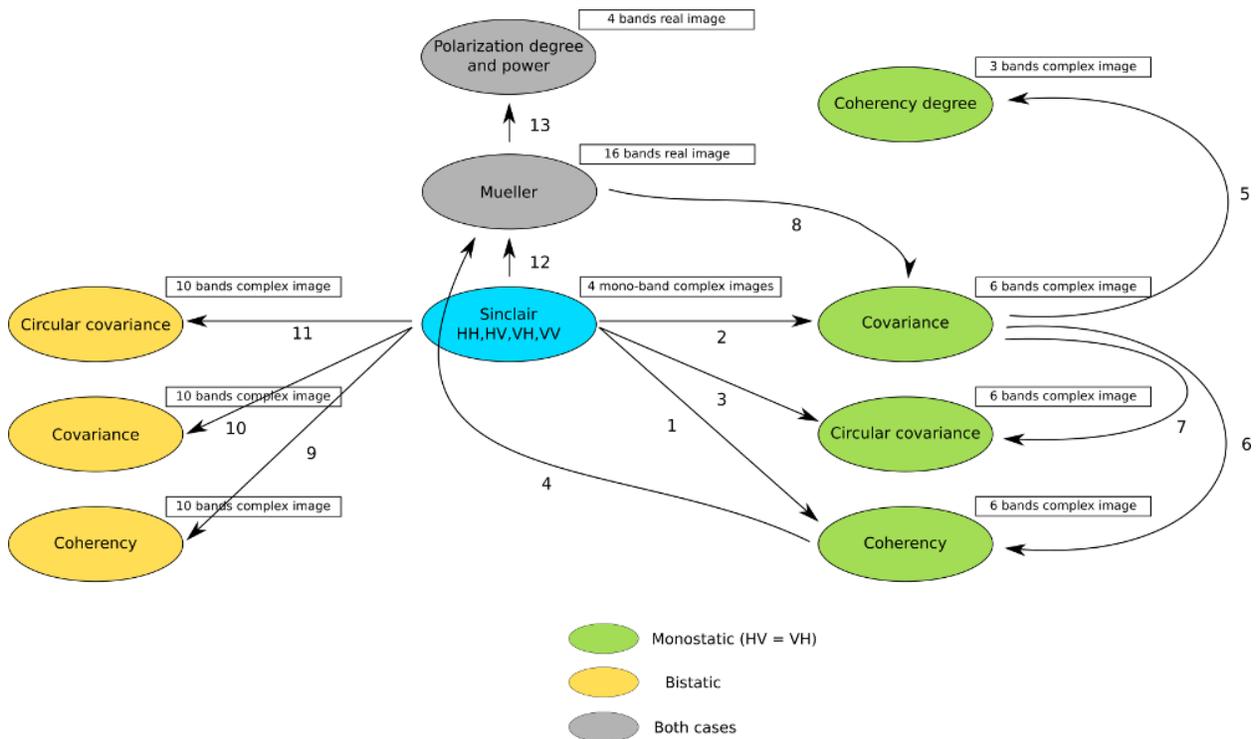
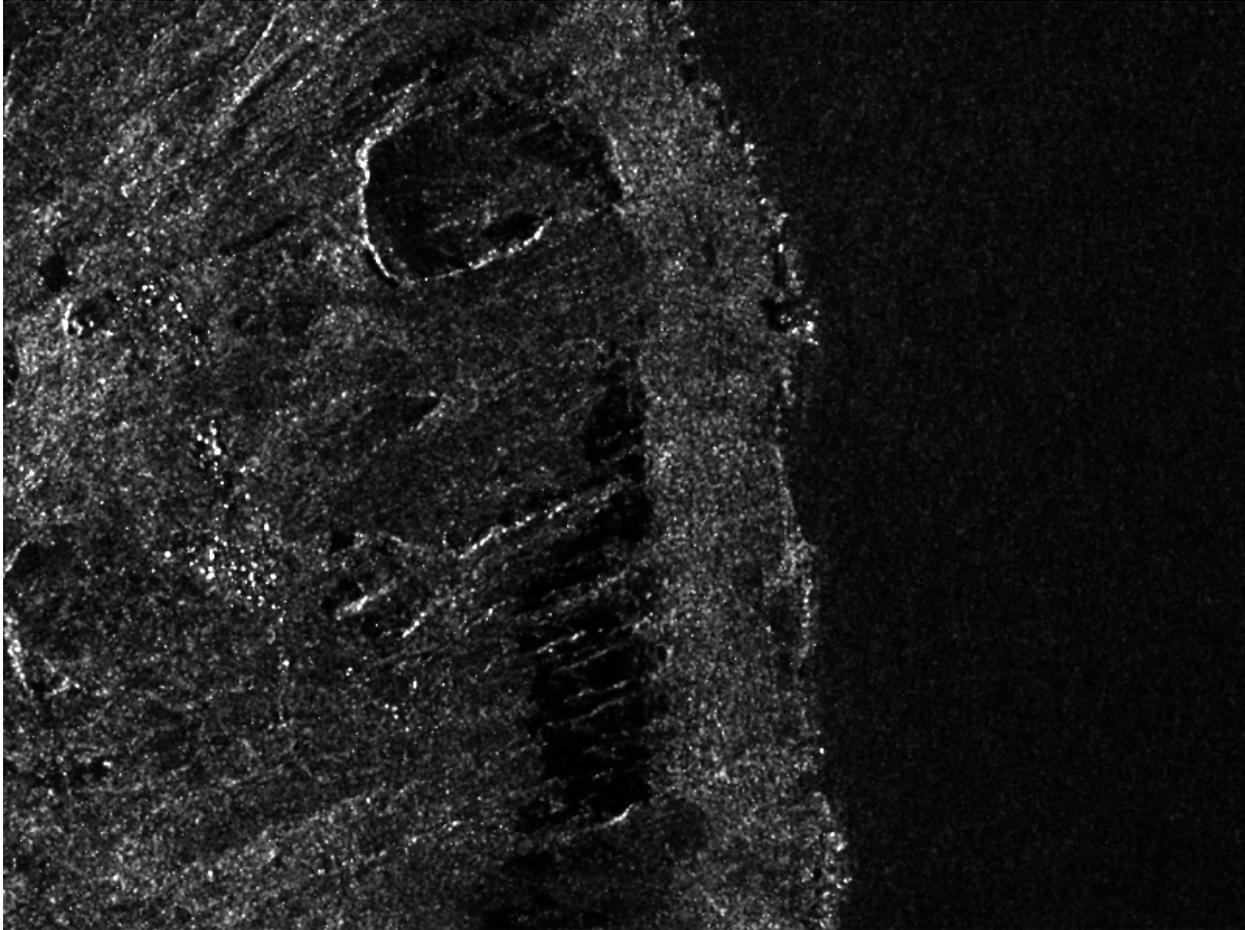
In conventional imaging radar the measurement is a scalar which is proportional to the received back-scattered power at a particular combination of linear polarization (HH, HV, VH or VV). Polarimetry is the measurement and interpretation of the polarization of this measurement which allows to measure various optical properties of a material. In polarimetry the basic measurement is a 2×2 complex scattering matrix yielding an eight dimensional measurement space (Sinclair matrix). For reciprocal targets where $HV = VH$, this space is compressed to five dimensions: three amplitudes ($|HH|$, $|HV|$, and $|VV|$); and two phase measurements, (co-pol: HH-VV, and cross-pol: HH-HV). (see [grss-ieee](#)).

Matrix conversions

This applications allows converting classical polarimetric matrices to each other. For instance, it is possible to get the coherency matrix from the Sinclair one, or the Mueller matrix from the coherency one. The figure below ([fig:polconv]) shows the workflow used in this application.

The filters used in this application never handle matrices, but images where each band is related to their elements. As most of the time SAR polarimetry handles symmetric matrices, only the relevant elements are stored, so that the images representing them have a minimal number of bands. For instance, the coherency matrix size is 3×3 in the monostatic case, and 4×4 in the bistatic case : it will thus be stored in a 6-band or a 10-band complex image (the diagonal and the upper elements of the matrix).

The Sinclair matrix is a special case : it is always represented as 3 or 4 one-band complex images (for mono- or bistatic case).



There are 13 available conversions, each one being related to the following parameters:

1. msinclairtocoherency
2. msinclairtocovariance
3. msinclairtocircovariance
4. mcoherencytomueller
5. mcovariancetocoherencydegree
6. mcovariancetocoherency
7. mlinearcovariancetocircularcovariance
8. muellertomcovariance
9. bsinclairtocoherency
10. bsinclairtocovariance
11. bsinclairtocircovariance
12. sinclairtomueller
13. muellertopoldegandpower

For each option parameter, the list below gives the formula used.

— Monostatic case —

1. msinclairtocoherency (SinclairToReciprocalCoherencyMatrixFuncor)
 - (a) $0.5 \cdot (S_{hh} + S_{vv}) \cdot (S_{hh} + S_{vv})^*$
 - (b) $0.5 \cdot (S_{hh} + S_{vv}) \cdot (S_{hh} - S_{vv})^*$
 - (c) $0.5 \cdot (S_{hh} + S_{vv}) \cdot (2S_{hv})^*$
 - (d) $0.5 \cdot (S_{hh} - S_{vv}) \cdot (S_{hh} - S_{vv})^*$
 - (e) $0.5 \cdot (S_{hh} - S_{vv}) \cdot (2S_{hv})^*$
 - (f) $0.5 \cdot (2S_{hv}) \cdot (2S_{hv})^*$
2. msinclairtocovariance (SinclairToReciprocalCovarianceMatrixFuncor)
 - (a) $S_{hh} \cdot S_{hh}^*$
 - (b) $\sqrt{2} \cdot S_{hh} \cdot S_{hv}^*$
 - (c) $S_{hh} \cdot S_{vv}^*$
 - (d) $2 \cdot S_{hv} \cdot S_{hv}^*$
 - (e) $\sqrt{2} \cdot S_{hv} \cdot S_{vv}^*$
 - (f) $S_{vv} \cdot S_{vv}^*$
3. msinclairtocircovariance (SinclairToReciprocalCircularCovarianceMatrixFuncor)
 - (a) $S_{ll} \cdot S_{ll}^*$
 - (b) $S_{ll} \cdot S_{lr}^*$
 - (c) $S_{ll} \cdot S_{rr}^*$
 - (d) $S_{lr} \cdot S_{lr}^*$
 - (e) $S_{lr} \cdot S_{rr}^*$

(f) $S_{rr} \cdot S_{rr}^*$

With:

- $S_{ll} = 0.5(S_{hh} + 2jS_{hv} - S_{vv})$
- $S_{lr} = 0.5(jS_{hh} + jS_{vv})$
- $S_{rr} = 0.5(-S_{hh} + 2jS_{hv} + S_{vv})$

4. mcoherencytomueller (ReciprocalCoherencyToReciprocalMuellerFunctor)

(a) $0.5 * (C_{11} + C_{22} + C_{33})$

(b) $Re(C_{12}) + Im(C_{22})$

(c) $Re(C_{13})$

(d) $Im(C_{23})$

(e) $Re(C_{12})$

(f) $0.5 * (C_{11} + C_{22} - C_{33})$

(g) $Re(C_{23})$

(h) $Im(C_{13})$

(i) $-Re(C_{13})$

(j) $-Re(C_{23})$

(k) $0.5.Re(VAL1)$

(l) $0.5.Im(VAL0)$

(m) $Im(C_{23})$

(n) $Im(C_{13})$

(o) $0.5.Im(VAL1^*)$

(p) $0.5.Re(VAL0)$

With:

- $VAL0 = C_{33} + C_{12} - C_{11} - (C_{12} - C_{22})^*$
- $VAL1 = -C_{33} + C_{12} - C_{11} - (C_{12} - C_{22})^*$

Where C_{ij} are related to the elements of the reciprocal coherence matrix.

5. mcovariancetocoherencydegree (ReciprocalCovarianceToCoherencyDegreeFunctor)

(a) $abs(S_{hh} \cdot S_{vv}^*) / sqrt(S_{hh} \cdot S_{hh}^*) / sqrt(S_{vv} \cdot S_{vv}^*)$

(b) $abs(S_{hv} \cdot S_{vv}^*) / sqrt(S_{hv} \cdot S_{hv}^*) / sqrt(S_{vv} \cdot S_{vv}^*)$

(c) $abs(S_{hh} \cdot S_{hv}^*) / sqrt(S_{hh} \cdot S_{hh}^*) / sqrt(S_{hv} \cdot S_{hv}^*)$

6. mcovariancetocoherency (ReciprocalCovarianceToReciprocalCoherencyFunctor)

(a) $0.5 \cdot (C_{33} + C_{13} + C_{13}^* + C_{11})$

(b) $0.5 \cdot (-C_{33} - C_{13} + C_{13}^* + C_{11})$

(c) $0.5 \cdot (\sqrt{2} \cdot C_{12} + \sqrt{2} \cdot C_{23}^*)$

(d) $0.5 \cdot (C_{33} - C_{13} - C_{13}^* + C_{11})$

(e) $0.5 \cdot (\sqrt{2} \cdot C_{12} - \sqrt{2} \cdot C_{23}^*)$

(f) $0.5.(2.C_{22})$

Where C_{ij} are related to the elements of the reciprocal linear covariance matrix.

7. `mlinearcovariancetocircularcovariance (ReciprocalLinearCovarianceToReciprocalCircularCovarianceFuncor)`

(a) $0.25.(C_{33} - i.\sqrt{2}.C_{23} - C_{13} + i.\sqrt{2}.C_{23}^* - C_{13}^* + 2.C_{22} - i.\sqrt{2}.C_{12} + i.\sqrt{2}.C_{12}^* + C_{11})$

(b) $0.25.(i.\sqrt{2}.C_{33} + 2.C_{23} - i.\sqrt{2}.C_{13} + i.\sqrt{2}.C_{13}^* + 2.C_{12}^* - i.\sqrt{2}.C_{11})$

(c) $0.25.(-C_{33} + i.\sqrt{2}.C_{23} + C_{13} + i.\sqrt{2}.C_{23}^* + C_{13}^* + 2.C_{22} - i.\sqrt{2}.C_{12} - i.\sqrt{2}.C_{12}^* - C_{11})$

(d) $0.25.(2.C_{33} + 2.C_{13} + 2.C_{13}^* + 2.C_{11})$

(e) $0.25.(i.\sqrt{2}.C_{33} + i.\sqrt{2}.C_{13} + 2.C_{23}^* - i.\sqrt{2}.C_{13}^* + 2.C_{12} - i.\sqrt{2}.C_{11})$

(f) $0.25.(C_{33} + i.\sqrt{2}.C_{23} - C_{13} - i.\sqrt{2}.C_{23}^* - C_{13}^* + 2.C_{22} + i.\sqrt{2}.C_{12} - i.\sqrt{2}.C_{12}^* + C_{11})$

Where C_{ij} are related to the elements of the reciprocal linear covariance matrix.

8. `muellertomcovariance (MuellerToReciprocalCovarianceFuncor)`

(a) $0.5.(M_{11} + M_{22} + 2.M_{12})$

(b) $0.5.\sqrt{2}.[(M_{13} + M_{23}) + j.(M_{14} + M_{24})]$

(c) $-0.5.(M_{33} + M_{44}) - j.M_{34}$

(d) $M_{11} - M_{22}$

(e) $0.5.\sqrt{2}.[(M_{13} - M_{23}) + j.(M_{14} - M_{24})]$

(f) $0.5.(M_{11} + M_{22} - 2.M_{12})$

— Bistatic case —

1. `bsinclairtocoherency (SinclairToCoherencyMatrixFuncor)`

(a) $(S_{hh} + S_{vv}).(S_{hh} + S_{vv})^*$

(b) $(S_{hh} + S_{vv}).(S_{hh} - S_{vv})^*$

(c) $(S_{hh} + S_{vv}).(S_{hv} + S_{vh})^*$

(d) $(S_{hh} + S_{vv}).(j(S_{hv} - S_{vh}))^*$

(e) $(S_{hh} - S_{vv}).(S_{hh} - S_{vv})^*$

(f) $(S_{hh} - S_{vv}).(S_{hv} + S_{vh})^*$

(g) $(S_{hh} - S_{vv}).(j(S_{hv} - S_{vh}))^*$

(h) $(S_{hv} + S_{vh}).(S_{hv} + S_{vh})^*$

(i) $(S_{hv} + S_{vh}).(j(S_{hv} - S_{vh}))^*$

(j) $j(S_{hv} - S_{vh}).(j(S_{hv} - S_{vh}))^*$

2. `bsinclairtocoherence (SinclairToCovarianceMatrixFuncor)`

(a) $S_{hh}.S_{hh}^*$

(b) $S_{hh}.S_{hv}^*$

(c) $S_{hh}.S_{vh}^*$

(d) $S_{hh}.S_{vv}^*$

(e) $S_{hv}.S_{hv}^*$

(f) $S_{hv}.S_{vh}^*$

- (g) $S_{hv} \cdot S_{vv}^*$
- (h) $S_{vh} \cdot S_{vh}^*$
- (i) $S_{vh} \cdot S_{vv}^*$
- (j) $S_{vv} \cdot S_{vv}^*$

3. `bsinclairtocircovariance` (`SinclairToCircularCovarianceMatrixFunctor`)

- (a) $S_{ll} \cdot S_{ll}^*$
- (b) $S_{ll} \cdot S_{lr}^*$
- (c) $S_{ll} \cdot S_{rl}^*$
- (d) $S_{ll} \cdot S_{rr}^*$
- (e) $S_{lr} \cdot S_{lr}^*$
- (f) $S_{lr} \cdot S_{rl}^*$
- (g) $S_{lr} \cdot S_{rr}^*$
- (h) $S_{rl} \cdot S_{rl}^*$
- (i) $S_{rl} \cdot S_{rr}^*$
- (j) $S_{rr} \cdot S_{rr}^*$

With:

- $S_{ll} = 0.5(S_{hh} + jS_{hv} + jS_{vh} - S_{vv})$
- $S_{lr} = 0.5(jS_{hh} + S_{hv} - S_{vh} + jS_{vv})$
- $S_{rl} = 0.5(jS_{hh} - S_{hv} + S_{vh} + jS_{vv})$
- $S_{rr} = 0.5(-S_{hh} + jS_{hv} + jS_{vh} + S_{vv})$

— Both cases —

4. `sinclairtomueller` (`SinclairToMueller`)

- (a) $0.5Re(T_{xx} \cdot T_{xx}^* + T_{xy} \cdot T_{xy}^* + T_{yx} \cdot T_{yx}^* + T_{yy} \cdot T_{yy}^*)$
- (b) $0.5Re(T_{xx} \cdot T_{xx}^* - T_{xy} \cdot T_{xy}^* + T_{yx} \cdot T_{yx}^* - T_{yy} \cdot T_{yy}^*)$
- (c) $Re(T_{xx} \cdot T_{xy}^* + T_{yx} \cdot T_{yy}^*)$
- (d) $Im(T_{xx} \cdot T_{xy}^* + T_{yx} \cdot T_{yy}^*)$
- (e) $0.5Re(T_{xx} \cdot T_{xx}^* + T_{xy} \cdot T_{xy}^* - T_{yx} \cdot T_{yx}^* - T_{yy} \cdot T_{yy}^*)$
- (f) $0.5Re(T_{xx} \cdot T_{xx}^* - T_{xy} \cdot T_{xy}^* - T_{yx} \cdot T_{yx}^* + T_{yy} \cdot T_{yy}^*)$
- (g) $Re(T_{xx} \cdot T_{xy}^* - T_{yx} \cdot T_{yy}^*)$
- (h) $Im(T_{xx} \cdot T_{xy}^* - T_{yx} \cdot T_{yy}^*)$
- (i) $Re(T_{xx} \cdot T_{yx}^* + T_{xy} \cdot T_{yy}^*)$
- (j) $Im(T_{xx} \cdot T_{yx}^* - T_{xy} \cdot T_{yy}^*)$
- (k) $Re(T_{xx} \cdot T_{yy}^* + T_{xy} \cdot T_{yx}^*)$
- (l) $Im(T_{xx} \cdot T_{yy}^* - T_{xy} \cdot T_{yx}^*)$
- (m) $Re(T_{xx} \cdot T_{yx}^* + T_{xy} \cdot T_{yy}^*)$
- (n) $Im(T_{xx} \cdot T_{yx}^* - T_{xy} \cdot T_{yy}^*)$

(o) $Re(T_{xx} \cdot T_{yy}^* + T_{xy} \cdot T_{yx}^*)$

(p) $Im(T_{xx} \cdot T_{yy}^* - T_{xy} \cdot T_{yx}^*)$

With :

- $T_{xx} = -S_{hh}$
- $T_{xy} = -S_{hv}$
- $T_{yx} = S_{vh}$
- $T_{yy} = S_{vv}$

5. muellertopoldegandpower (MuellerToPolarisationDegreeAndPowerFuncor)

(a) P_{min}

(b) P_{max}

(c) $DegP_{min}$

(d) $DegP_{max}$

Examples :

1. `otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
-inhv imageryC_HV.tif
-invv imageryC_VV.tif
-conv msinclairtocoherency
-outc coherency.tif`
2. `otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
-inhv imageryC_HV.tif
-invv imageryC_VV.tif
-conv msinclairtocovariance
-outc covariance.tif`
3. `otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
-inhv imageryC_HV.tif
-invv imageryC_VV.tif
-conv msinclairtocircovariance
-outc circ_covariance.tif`
4. `otbcli_SARPolarMatrixConvert -inc coherency.tif
-conv mcoherencytomueller
-outf mueller.tif`
5. `otbcli_SARPolarMatrixConvert -inc covariance.tif
-conv mcovariancetocoherencydegree
-outc coherency_degree.tif`
6. `otbcli_SARPolarMatrixConvert -inc covariance.tif
-conv mcovariancetocoherency
-outc coherency.tif`
7. `otbcli_SARPolarMatrixConvert -inc covariance.tif
-conv mlinearcovariancetocircularcovariance
-outc circ_covariance.tif`
8. `otbcli_SARPolarMatrixConvert -inf mueller.tif
-conv muellertomcovariance
-outc covariance.tif`

- ```

9. otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
 -inhv imageryC_HV.tif
 -invh imageryC_VH.tif
 -invv imageryC_VV.tif
 -conv bsinclairtocoherency
 -outc bcoherency.tif

10. otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
 -inhv imageryC_HV.tif
 -invh imageryC_VH.tif
 -invv imageryC_VV.tif
 -conv bsinclairtocovariance
 -outc bcovariance.tif

11. otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
 -inhv imageryC_HV.tif
 -invh imageryC_VH.tif
 -invv imageryC_VV.tif
 -conv bsinclairtocircovariance
 -outc circ_bcovariance.tif

12. otbcli_SARPolarMatrixConvert -inhh imageryC_HH.tif
 -inhv imageryC_HV.tif
 -invh imageryC_VH.tif
 -invv imageryC_VV.tif
 -conv sinclairtomueller
 -outf mueller.tif

13. otbcli_SARPolarMatrixConvert -inf mueller.tif
 -conv muellertopoldegandpower
 -outf degreepower.tif

```

## Polarimetric decompositions

From one-band complex images (HH, HV, VH, VV), returns the selected decomposition. The H-alpha-A decomposition is currently the only one available; it is implemented for the monostatic case (transmitter and receiver are co-located). User must provide three one-band complex images HH, HV or VH, and VV (HV = VH in monostatic case). The H-alpha-A decomposition consists in averaging 3x3 complex coherency matrices (incoherent analysis) : The user must provide the size of the averaging window, thanks to the parameter `inco.kernelsize`. The applications returns a float vector image, made of three channels : H(entropy), Alpha, A(Anisotropy).

Here are the formula used (refer to the previous section about how the coherence matrix is obtained from the Sinclair one):

1.  $entropy = - \sum_{i=0}^2 \frac{p[i] \cdot \log p[i]}{\log 3}$
2.  $\alpha = \sum_{i=0}^2 p[i] \cdot \alpha_i$
3.  $anisotropy = \frac{SortedEigenValues[1] - SortedEigenValues[2]}{SortedEigenValues[1] + SortedEigenValues[2]}$

Where:

- $p[i] = \max(SortedEigenValues[i], 0) / \sum_{i=0}^{2, SortedEigenValues[i] > 0} SortedEigenValues[i]$
- $\alpha_i = |SortedEigenVector[i]| * \frac{180}{\pi}$

Example :

We first extract a ROI from the original image (not required). Here `imagery_HH.tif` represents the element HH of the Sinclair matrix (and so forth).

- `otbcli_ExtractROI -in imagery_HH.tif -out imagery_HH_extract.tif -startx 0 -starty 0 -sizeX 1000 -sizeY 1000`
- `otbcli_ExtractROI -in imagery_HV.tif -out imagery_HV_extract.tif -startx 0 -starty 0 -sizeX 1000 -sizeY 1000`
- `otbcli_ExtractROI -in imagery_VV.tif -out imagery_VV_extract.tif -startx 0 -starty 0 -sizeX 1000 -sizeY 1000`

Next we apply the H-alpha-A decomposition:

```
otbcli_SARDecompositions -inhh imagery_HH_extract.tif
 -inhv imagery_HV_extract.tif
 -invv imagery_VV_extract.tif
 -decomp haa -inco.kernelsize 5
 -out haa_extract.tif
```

The result has three bands : entropy (0..1) - alpha (0..90) - anisotropy (0..1). It is split into 3 mono-band images thanks to following command :

```
otbcli_SplitImage -in haa_extract.tif -out haa_extract_splitted.tif
```

Each image is then colored thanks to a color look-up table 'hot'. Notice how minimum and maximum values are provided for each polarimetric variable.

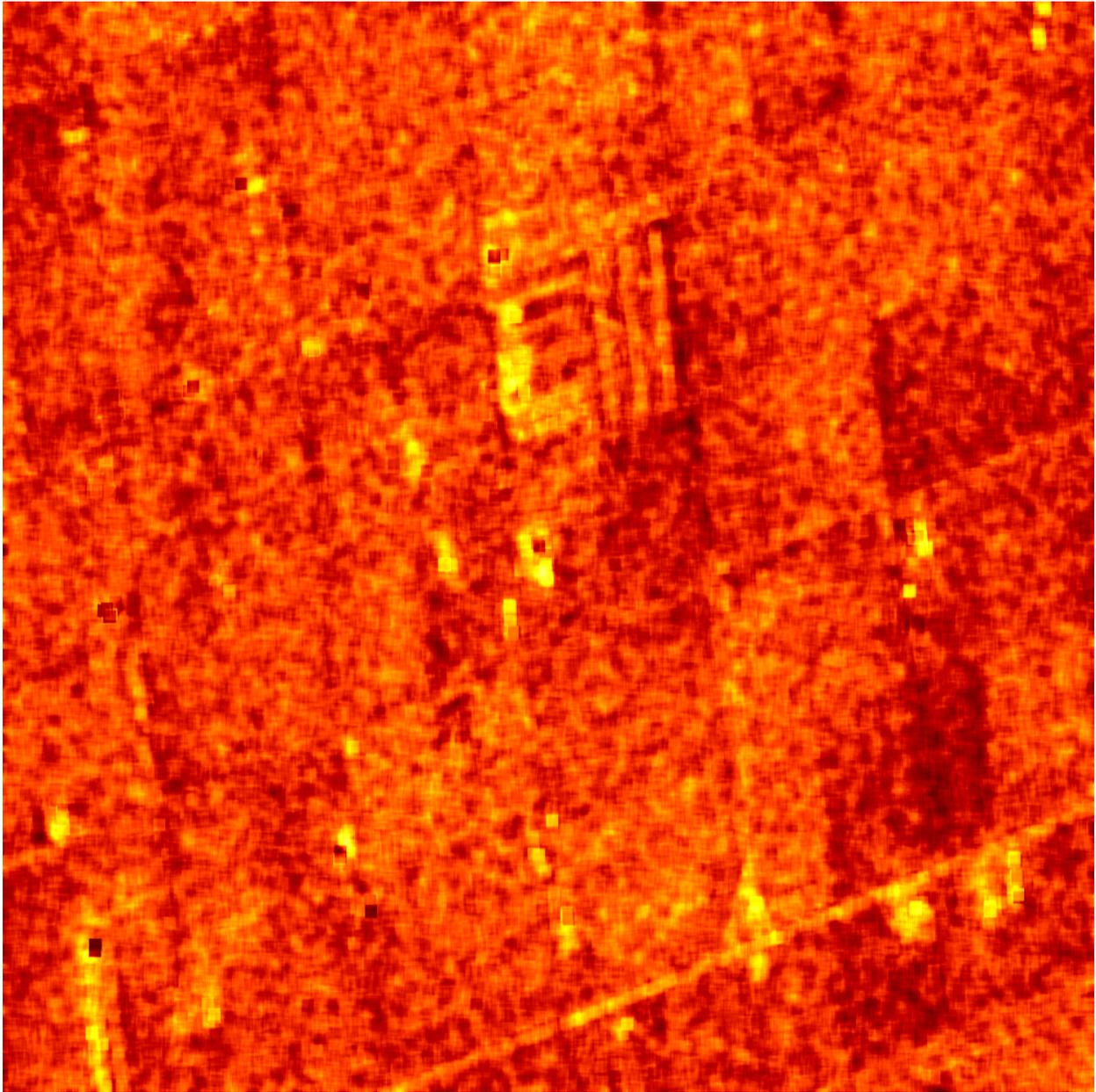
- `otbcli_ColorMapping -in haa_extract_splitted_0.tif -method continuous -method.continuous.lut hot -method.continuous.min 0 -method.continuous.max 1 -out entropy_hot.tif uint8`
- `otbcli_ColorMapping -in haa_extract_splitted_1.tif -method continuous -method.continuous.lut hot -method.continuous.min 0 -method.continuous.max 90 -out alpha_hot.tif uint8`
- `otbcli_ColorMapping -in haa_extract_splitted_2.tif -method continuous -method.continuous.lut hot -method.continuous.min 0 -method.continuous.max 1 -out anisotropy_hot.tif uint8`

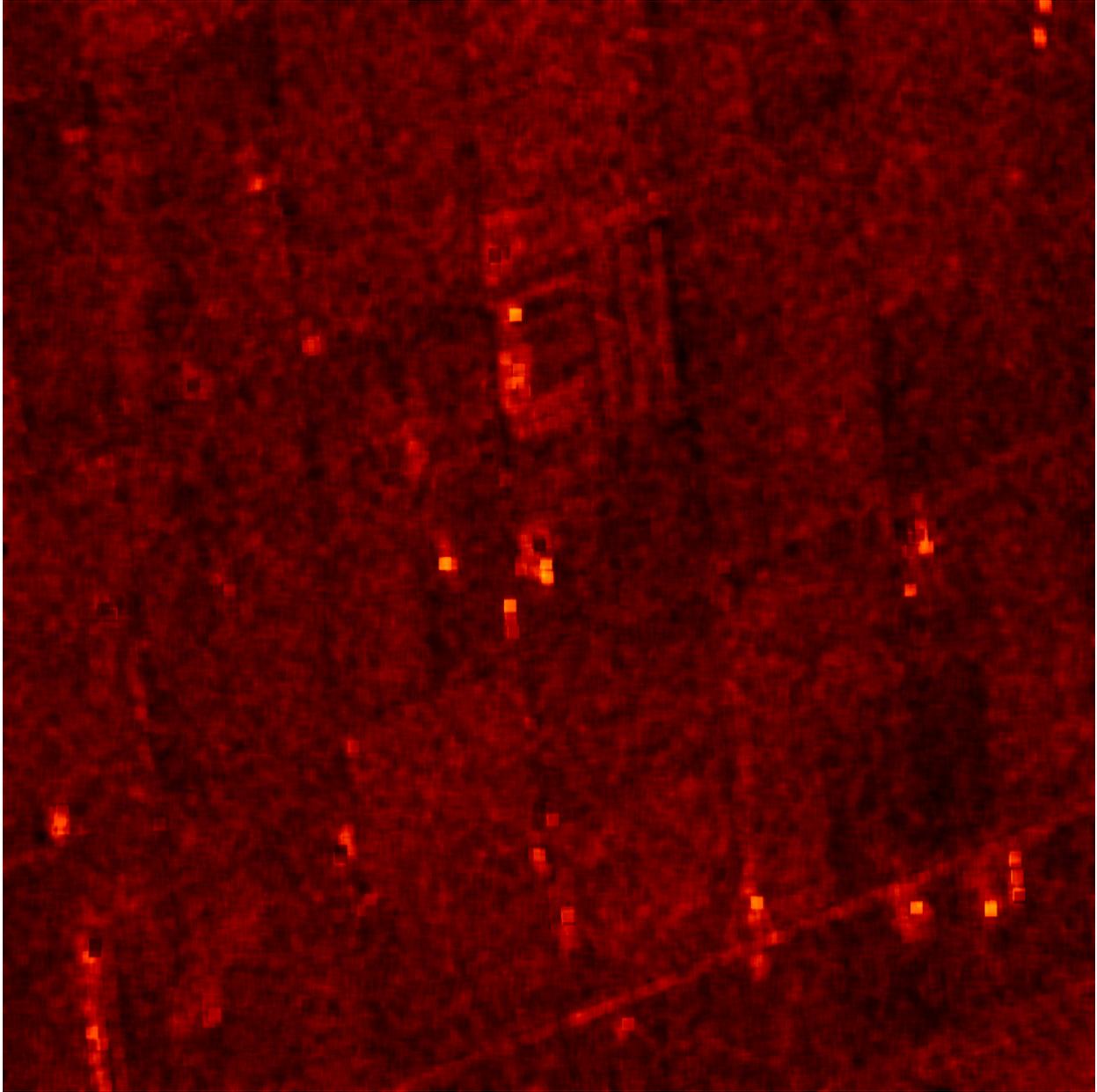
The results are shown in the figures below ([fig:entropyimage] , [fig:alphaimage] and [fig:anisotropyimage]).

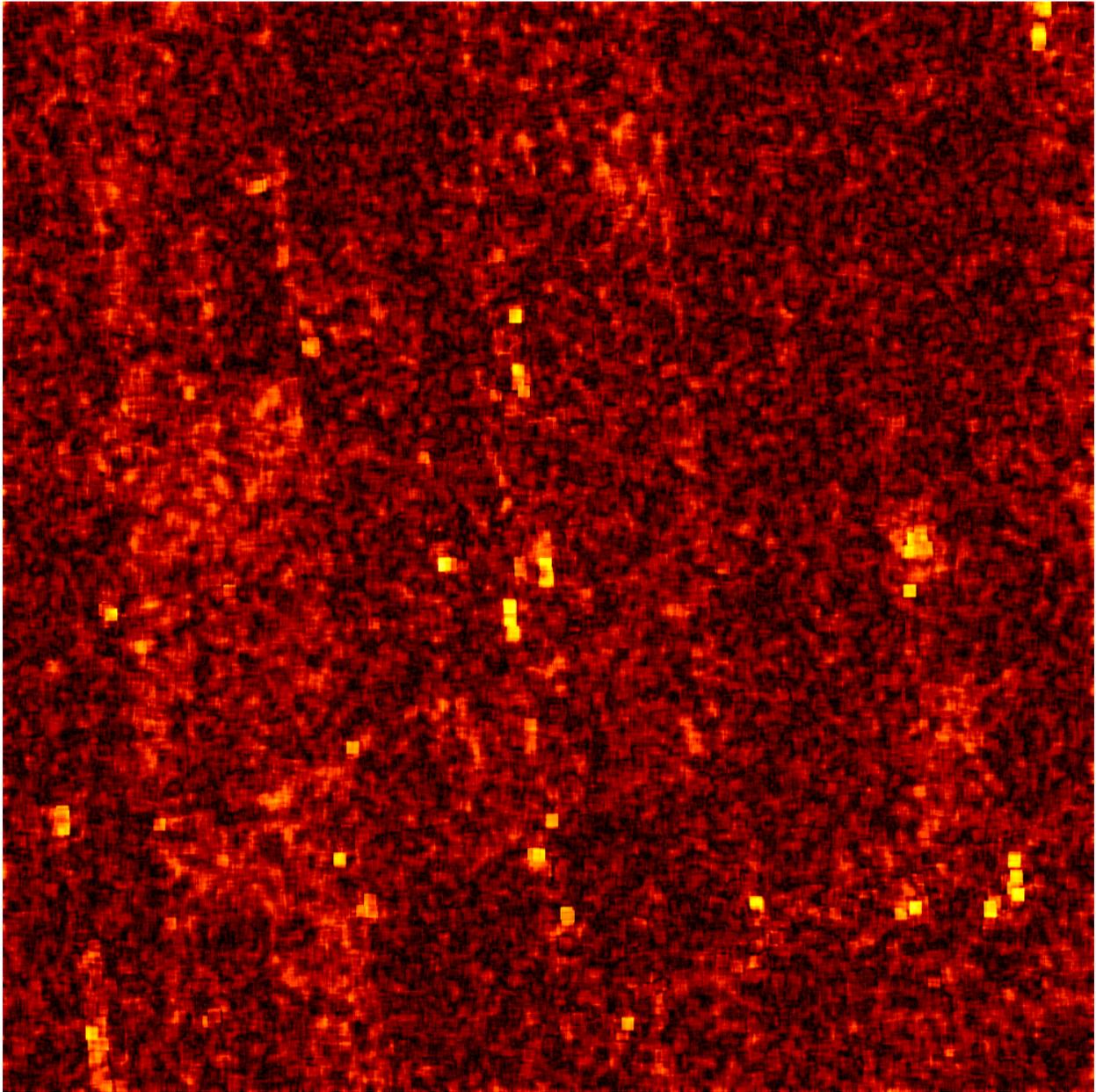
## Polarimetric synthetis

This application gives, for each pixel, the power that would have been received by a SAR system with a basis different from the classical (H,V) one (polarimetric synthetis). The new basis are indicated through two Jones vectors, defined by the user thanks to orientation ( $\psi$ ) and ellipticity ( $\kappa$ ) parameters. These parameters are namely  $\psi_{ii}$ ,  $\kappa_{ii}$ ,  $\psi_{ir}$  and  $\kappa_{ir}$ . The suffixes (i) and (r) refer to the transmitting antenna and the receiving antenna respectively. Orientations and ellipticity are given in degrees, and are between -90/90 degrees and -45/45 degrees respectively.

Four polarization architectures can be processed :







1. HH\_HV\_VH\_VV : full polarization, general bistatic case.
2. HH\_HV\_VV or HH\_VH\_VV : full polarization, monostatic case (transmitter and receiver are co-located).
3. HH\_HV : dual polarization.
4. VH\_VV : dual polarization.

The application takes a complex vector image as input, where each band correspond to a particular emission/reception polarization scheme. User must comply with the band order given above, since the bands are used to build the Sinclair matrix.

In order to determine the architecture, the application first relies on the number of bands of the input image.

1. Architecture HH\_HV\_VH\_VV is the only one with four bands, there is no possible confusion.
2. Concerning HH\_HV\_VV and HH\_VH\_VV architectures, both correspond to a three channels image. But they are processed in the same way, as the Sinclair matrix is symmetric in the monostatic case.
3. Finally, the two last architectures (dual-polarization), can't be distinguished only by the number of bands of the input image. User must then use the parameters `emissionh` and `emissionv` to indicate the architecture of the system : `emissionh=1` and `emissionv=0` for HH\_HV, `emissionh=0` and `emissionv=1` for VH\_VV.

Note : if the architecture is HH\_HV, `khii` and `psii` are automatically set to 0/0 degrees; if the architecture is VH\_VV, `khii` and `psii` are automatically set to 0/90 degrees.

It is also possible to force the calculation to co-polar or cross-polar modes. In the co-polar case, values for `psir` and `khir` will be ignored and forced to `psii` and `khii`; same as the cross-polar mode, where `khir` and `psir` will be forced to `psii + 90` degrees and `-khii`.

Finally, the result of the polarimetric synthesis is expressed in the power domain, through a one-band scalar image.

The final formula is thus :  $P = |B^T \cdot [S] \cdot A|^2$ , where A and B are two Jones vectors and S is a Sinclair matrix.

The two figures below ([fig:polsynthll] and [fig:polsynthlr]) show the two images obtained with the basis LL and LR (L for left circular polarization and R for right polarization), from a Radarsat-2 image taken over Vancouver, Canada. Once the four two-band images `imagery_HH` `imagery_HV` `imagery_VH` `imagery_VV` were merged into a single four complex band image `imageryC_HH_HV_VH_VV.tif`, the following commands were used to produce the LL and LR images :

```
otbcli_SARPolarSynth -in imageryC_HH_HV_VH_VV.tif
 -psii 0 -khii 45 -mode co
 -out test-LL.tif

otbcli_SARPolarSynth -in imageryC_HH_HV_VH_VV.tif
 -psii 0 -khii 45 -mode cross
 -out test-LR.tif
```

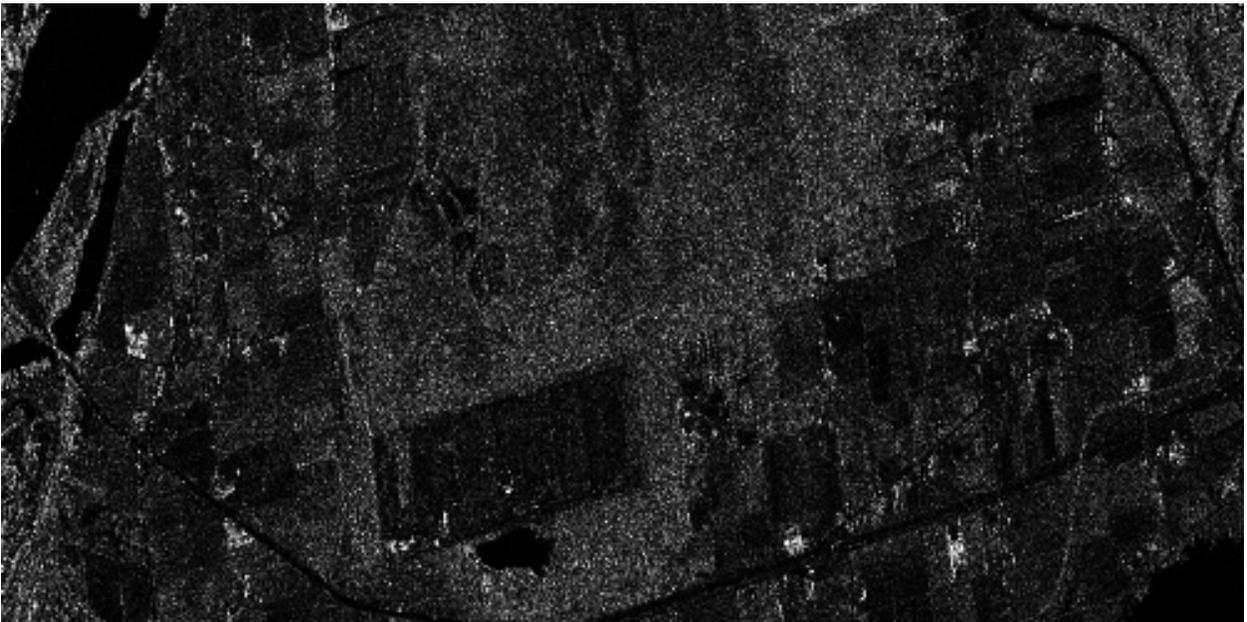
The produced images were then rescaled to intensities ranging from 0 to 255 in order to be displayed.

## Polarimetric data visualization

Finally, let's talk about polarimetric data visualization. There is a strong link between polarimetric data visualization and the way they can be decomposed into significant physical processes. Indeed, by setting the results (or combinations) of such decompositions to RGB channels that help in interpreting SAR polarimetric images.

There is no specific dedicated application yet, but it is possible to use a combination of different applications as a replacement. Let's do it with a RADARSAT-2 acquisition over the famous place of the Golden Gate Bridge, San Francisco, California.

We first make an extract from the original image (not mandatory).



- `otbcli_ExtractROI -in imagery_HH.tif -out imagery_HH_extract.tif -startx 0 -starty 6300 -sizeX 2790 -sizeY 2400`
- `otbcli_ExtractROI -in imagery_HV.tif -out imagery_HV_extract.tif -startx 0 -starty 6300 -sizeX 2790 -sizeY 2400`
- `otbcli_ExtractROI -in imagery_VV.tif -out imagery_VV_extract.tif -startx 0 -starty 6300 -sizeX 2790 -sizeY 2400`

Then we compute the amplitude of each band using the **BandMath** application:

- `otbcli_BandMath -il imagery_HH_extract.tif -out HH.tif -exp "sqrt(im1b1^2+im1b2^2) "`
- `otbcli_BandMath -il imagery_HV_extract.tif -out HV.tif -exp "sqrt(im1b1^2+im1b2^2) "`
- `otbcli_BandMath -il imagery_VV_extract.tif -out VV.tif -exp "sqrt(im1b1^2+im1b2^2) "`

Note that **BandMath** application interprets the image 'imagery\_XX\_extract.tif' as an image made of two bands, where the first one is related to the real part of the signal, and where the second one is related to the imaginary part (that's why the modulus is obtained by the expressions  $im1b1^2 + im1b2^2$ ).

Then, we rescale the produced images to intensities ranging from 0 to 255:

- `otbcli_Rescale -in HH.tif -out HH_res.png uint8`
- `otbcli_Rescale -in HV.tif -out HV_res.png uint8`
- `otbcli_Rescale -in VV.tif -out VV_res.png uint8`

Figures below ([fig:hhfrisco] , [fig:hvfrisco] and [fig:vvfrisco]) show the images obtained :

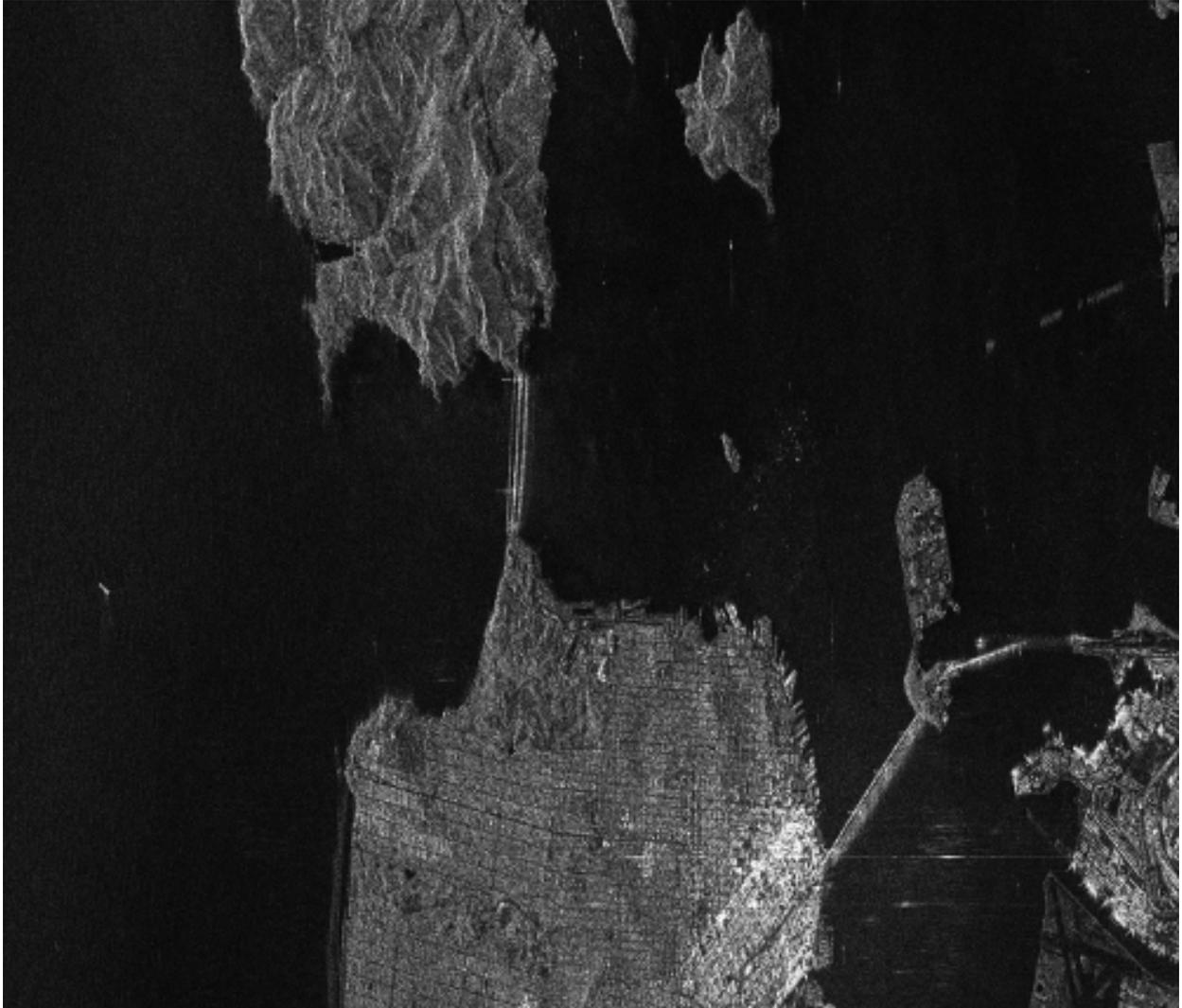
Now the most interesting step. In order to get a friendly coloration of these data, we are going to use the Pauli decomposition, defined as follows :

- $a = \frac{|S_{HH}-S_{VV}|}{\sqrt{2}}$
- $b = \sqrt{2} \cdot |S_{HV}|$
- $c = \frac{|S_{HH}+S_{VV}|}{\sqrt{2}}$

We use the **BandMath** application again:

- `otbcli_BandMath -il imagery_HH_extract.tif imagery_HV_extract.tif imagery_VV_extract.tif -out Channel1.tif -exp "sqrt(((im1b1-im3b1)^2+(im1b2-im3b2)^2)) "`
- `otbcli_BandMath -il imagery_HH_extract.tif imagery_HV_extract.tif imagery_VV_extract.tif -out Channel2.tif -exp "sqrt(im2b1^2+im2b2^2) "`







- `otbcli_BandMath -il imagery_HH_extract.tif imagery_HV_extract.tif  
imagery_VV_extract.tif  
-out Channel3.tif  
-exp "sqrt(((im1b1+im3b1)^2+(im1b2+im3b2)^2))"`

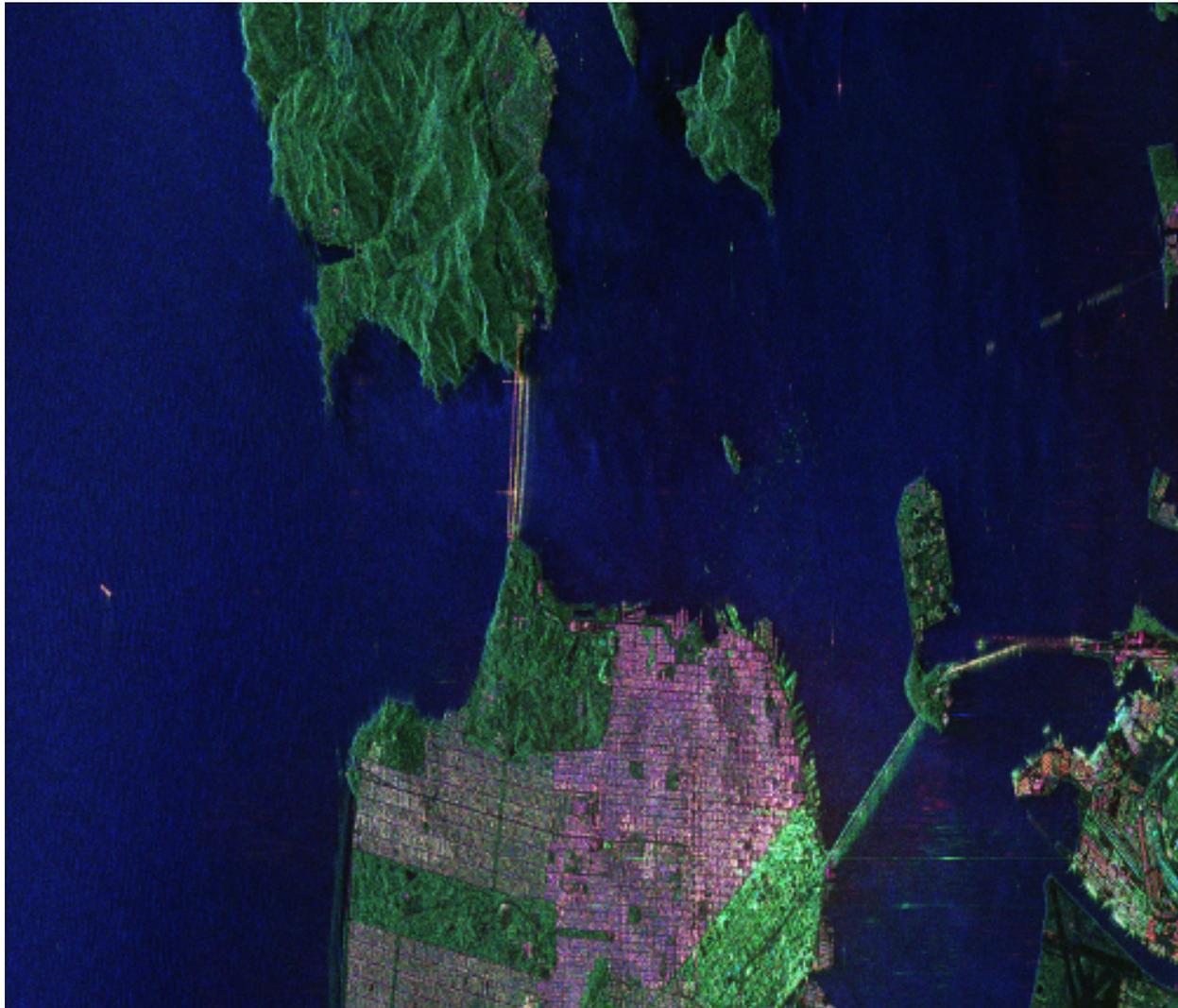
Note that  $\sqrt{2}$  factors have been omitted purposely, since their effects will be canceled by the rescaling step. Then, we rescale the produced images to intensities ranging from 0 to 255 :

- `otbcli_Rescale -in Channel1.tif -out Channel1_res.tif uint8`
- `otbcli_Rescale -in Channel2.tif -out Channel2_res.tif uint8`
- `otbcli_Rescale -in Channel3.tif -out Channel3_res.tif uint8`

And finally, we merge the three bands into a single RGB image.

```
otbcli_ConcatenateImages -il Channel1_res.tif Channel2_res.tif Channel3_res.tif
-out visuPauli.png
```

The result is shown in the figure below ([fig:colorfrisco]).



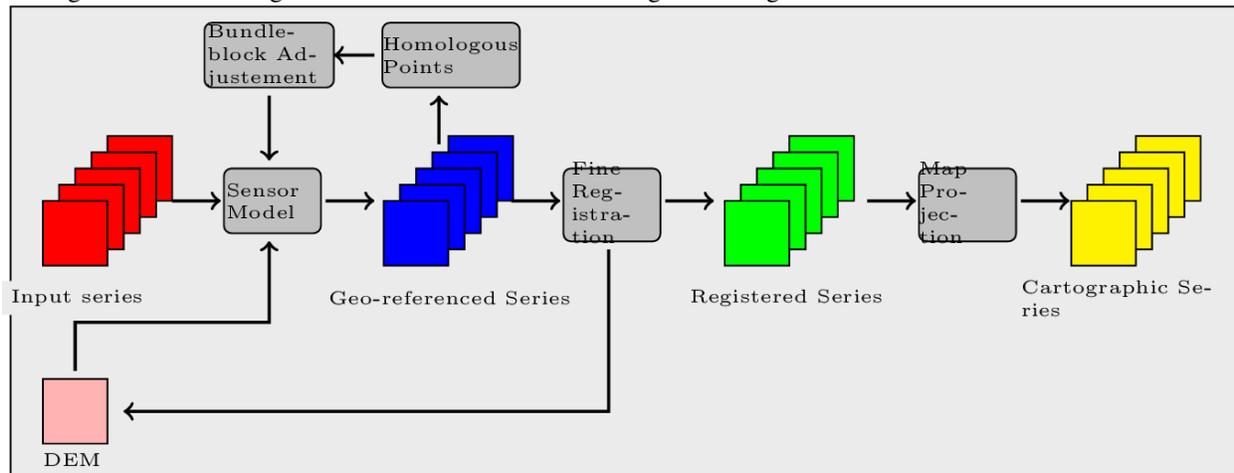
## 5.3 Residual registration

Image registration is a fundamental problem in image processing. The aim is to align two or more images of the same scene often taken at different times, from different viewpoints, or by different sensors. It is a basic step for orthorectification, image stitching, image fusion, change detection, and others. But this process is also critical for stereo reconstruction process to be able to obtain an accurate estimation of epipolar geometry.

Sensor model is generally not sufficient to provide image registrations. Indeed, several sources of geometric distortion can be contained in optical remote sensing images including earth rotation, platform movement, non linearity, etc.

They result in geometric errors on scene level, image level and pixel level. It is critical to rectify the errors before a thematic map is generated, especially when the remote sensing data need to be integrated together with other GIS data.

This figure illustrates the generic workflow in the case of image series registration:



We will now illustrate this process by applying this workflow to register two images. This process can be easily extended to perform image series registration.

The aim of this example is to describe how to register a Level 1 QuickBird image over an orthorectified Pleiades image over the area of Toulouse, France.

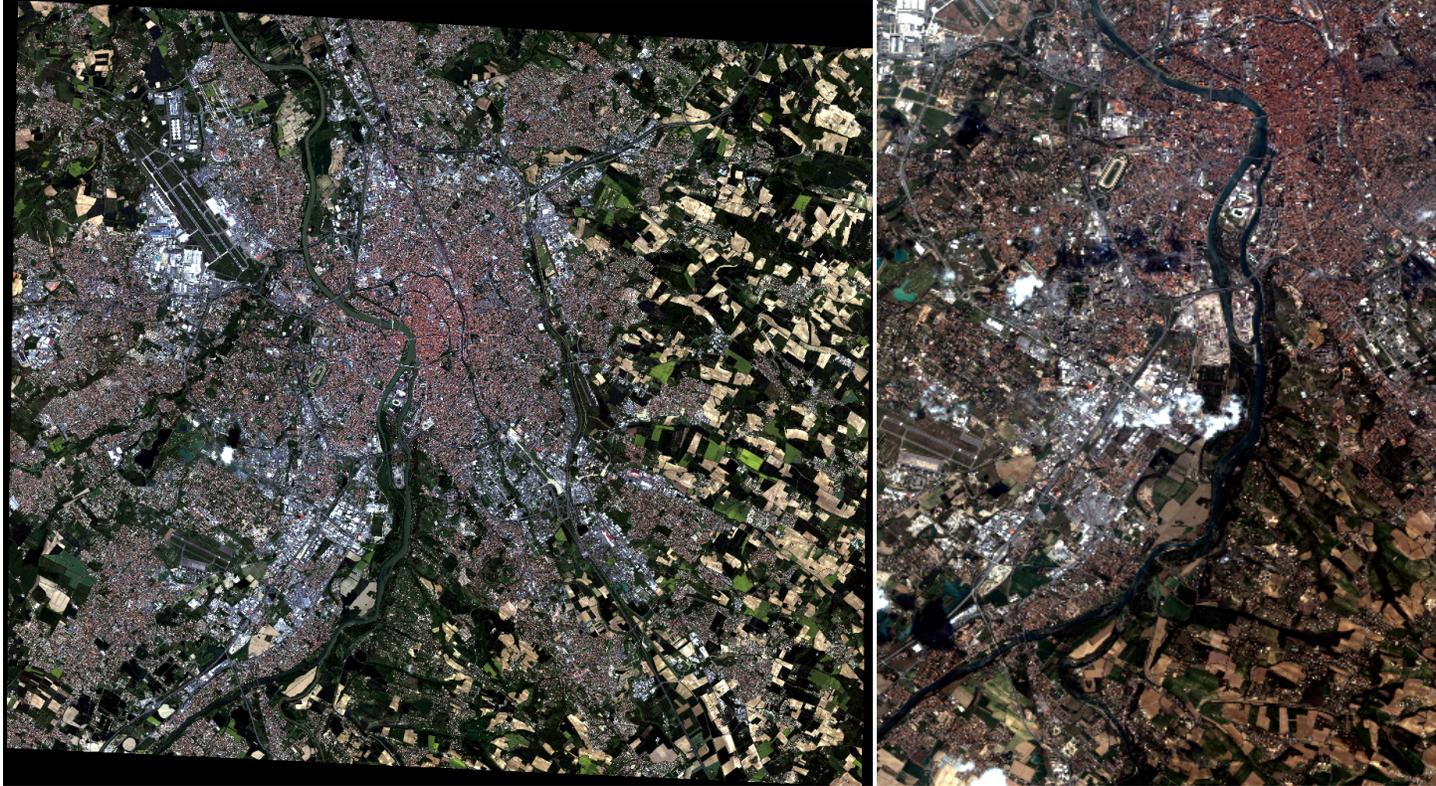


Figure 4.10: From left to right: Pleiades ortho-image, and original QuickBird image over Toulouse

### 5.3.1 Extract metadata from the image reference

We first dump geometry metadata of the image we want to refine in a text file. In OTB, we use the extension *.geom* for this type of file. As you will see the application which will estimate a refine geometry only needs as input this metadata and a set of homologous points. The refinement application will create a new *.geom* file containing refined geometry parameters which can be used after for reprojection for example.

The use of external *.geom* file is available in OTB since release 3.16. See [here](#) for more information.

```
otbcli_ReadImageInfo -in slave_image
 -outkwl TheGeom.geom
```

### 5.3.2 Extract homologous points from images

The main idea of the residual registration is to estimate a second transformation (after the application of sensors model).

The homologous point application use interest point detection method to get a set of point which match in both images.

The basic idea is to use this set of homologous points and estimate with them a residual transformation between the two images.

There is a wide variety of keypoint detector in the literature. They allow to detect and describe local features in images. These algorithms provide for each interesting point a “feature description”. This descriptor has the property to be invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. keypoints. Features extracted from the input images are then matched against each other. These correspondences are then used to create the homologous points.

**SIFT** or **SURF** keypoints can be computed in the application. The band on which keypoints are computed can be set independently for both images.

The application offers two modes :

- the first is the full mode where keypoints are extracted from the full extent of both images (please note that in this mode large image file are not supported).
- The second mode, called *geobins*, allows to set-up spatial binning so as to get fewer points spread across the entire image. In this mode, the corresponding spatial bin in the second image is estimated using geographical transform or sensor modeling, and is padded according to the user defined precision.

Moreover, in both modes the application can filter matches whose co-localization in the first image exceed this precision. Last, the elevation parameters allow to deal more precisely with sensor modelling in case of sensor geometry data. The *outvector* option allows to create a vector file with segments corresponding to the localization error between the matches.

Finally, with the *2wgs84* option, you can match two sensor geometry images or a sensor geometry image with an ortho-rectified reference. In all cases, you get a list of ground control points spread all over your image.

```
otbcli_HomologousPointsExtraction -in1 slave_image
 -in2 reference_image
 -algorithm surf
 -mode geobins
 -mode.geobins.binstep 512
 -mode.geobins.binsize 512
 -mfilter 1
 -precision 20
 -2wgs84 1
 -out homologous_points.txt
 -outvector points.shp
 -elev.dem dem_path/SRTM4-HGT/
 -elev.geoid OTB-Data/Input/DEM/egm96.grd
```

Note that for a proper use of the application, elevation must be correctly set (including DEM and geoid file).

### 5.3.3 Geometry refinement using homologous points

Now that we can use this set of tie points to estimate a residual transformation. For this we use the dedicated application called **RefineSensorModel**. This application make use of OSSIM capabilities to align the sensor model.

It reads the input geometry metadata file (*.geom*) which contains the sensor model information that we want to refine and the text file (*homologous\_points.txt*) containing the list of ground control point. It performs a least-square fit of the sensor model adjustable parameters to these tie points and produces an updated geometry file as output (the extension which is always use is *.geom*)

The application can provide as well an optional ground control points based statistics file and a vector file containing residues that you can display in a GIS software.

Please note again that for a proper use of the application, elevation must be correctly set (including DEM and geoid file). The map parameters allows to choose a map projection in which the accuracy will be estimated (in meters).

Accuracy values are provided as output of the application (computed using tie points location) and allow also to control the precision of the estimated model.

```
otbcli_RefineSensorModel -elev.dem dem_path/SRTM4-HGT/
-elev.geoid OTB-Data/Input/DEM/egm96.grd
-inggeom slave_image.geom
-outgeom refined_slave_image.geom
-inpoints homologous_points.txt
-outstat stats.txt
-outvector refined_slave_image.shp
```

### 5.3.4 Orthorectify image using the affine geometry

Now we will show how we can use this new sensor model. In our case we'll use this sensor model to orthorectify the image over the Pléiades reference. **Orfeo Toolbox** offers since version 3.16 the possibility to use <http://wiki.orfeo-toolbox.org/index.php/ExtendedFileNameextend> image path to use different metadata file as input. That's what we are going to use there to orthorectify the QuickBird image using the *.geom* file obtained by the **RefineSensorModel** applications. over the first one using for the second image estimated sensor model which take into account the original sensor model of the slave and which also fit to the set of tie points.

```
otbcli_OrthoRectification -io.in slave_image?&geom=TheRefinedGeom.geom
-io.out ortho_slave_image
-elev.dem dem_path/SRTM4-HGT/
-elev.geoid OTB-Data/Input/DEM/egm96.grd
```

As a result, if you've got enough homologous points in images and control that the residual error between the set of tie points and the estimated sensor model is small, you must achieve a good registration now between the 2 rectified images. Normally far better than 'only' performing separate orthorectification over the 2 images.

This methodology can be adapt and apply in several cases, for example :

- register stereo pair of images and estimate accurate epipolar geometry
- registration prior to change detection

## 5.4 Image processing and information extraction

### 5.4.1 Simple calculus with channels

The *BandMath* application provides a simple and efficient way to perform band operations. The command line application and the corresponding Monteverdi module (shown in the section [Band:sub:math module]) are based on the same standards. It computes a band wise operation according to a user defined mathematical expression. The following code computes the absolute difference between first bands of two images.

```
otbcli_BandMath -il input_image_1 input_image_2
-exp "abs(im1b1 - im2b1)"
-out output_image
```

The naming convention "im[x]b[y]" designates the yth band of the xth input image.

The *BandMath* application embeds built-in operators and functions listed in [muparser documentation](#) thus allowing a vast choice of possible operations.

## 5.4.2 Images with no-data values

Image files can contain a no-data value in their metadata. It represents a special pixel value that should be treated as “no data available for this pixel”. For instance, SRTM tiles use a particular no-data value of -32768 (usually found on sea areas).

On multiband images, the no-data values are handled independently for each band. The case of an image with no-data values defined only for a subset of its bands is supported.

This metadata is now handled by OTB image readers and writer (using the GDAL driver). The no-data value can be read from an image files and stored in the image metadata dictionary. It can also be exported by image writers. The OTB filters that produce a no-data value are able to export this value so that the output file will store it.

An application has been created to manage the no-data value. The application has the following features :

- Build a mask corresponding to the no-data pixels in the input image : it gives you a binary image of the no-data pixels in your input image.
- Change the no-data value of the input image : it will change all pixels that carry the old no-data value to the new one and update the metadata
- Apply an external mask to the input image as no-data : all the pixels that corresponds have a null mask value are flagged as no-data in the output image.

For instance, the following command converts the no-data value of the input image to the default value for DEM (which is -32768) :

```
otbcli_ManageNoData -in input_image.tif
 -out output_image.tif
 -mode changevalue
 -mode.changevalue.newv -32768
```

The third mode “apply” can be useful if you apply a formula to the entire image. This will likely change the values of pixels flagged as no-data, but the no-data value in the image metadata doesn’t change. If you want to fix all no-data pixels to their original value, you can extract the mask of the original image and apply it on the output image. For instance:

```
otbcli_ManageNoData -in input_image.tif
 -out mask.tif
 -mode buildmask

otbcli_BandMath -il input_image.tif
 -out filtered_image.tif
 -exp "2*im1b1-4"

otbcli_ManageNoData -in filtered_image.tif
 -out output_image.tif
 -mode apply
 -mode.apply.mask mask.tif
```

You can also use this “apply” mode with an additional parameter “mode.apply.ndval”. This parameter allow to set the output nodata value applying according to your input mask.

## 5.4.3 Segmentation

Segmenting objects across a very high resolution scene and with a controlled quality is a difficult task for which no method has reached a sufficient level of performance to be considered as operational.

Even if we leave aside the question of segmentation quality and consider that we have a method performing reasonably well on our data and objects of interest, the task of scaling up segmentation to real very high resolution data is itself

challenging. First, we can not load the whole data into memory, and there is a need for on the flow processing which does not cope well with traditional segmentation algorithms. Second, the result of the segmentation process itself is difficult to represent and manipulate efficiently.

The experience of segmenting large remote sensing images is packed into a single *Segmentation* in **OTB Applications**.

You can find more information about this application [here](#).

## 5.4.4 Large-Scale Mean-Shift (LSMS) segmentation

LSMS is a segmentation workflow which allows to perform tile-wise segmentation of very large image with theoretical guarantees of getting identical results to those without tiling.

It has been developed by David Youssefi and Julien Michel during David internship at CNES.

For more a complete description of the LSMS method, please refer to the following publication, *J. Michel, D. Youssefi and M. Grizonnet, "Stable Mean-Shift Algorithm and Its Application to the Segmentation of Arbitrarily Large Remote Sensing Images," in IEEE Transactions on Geoscience and Remote Sensing, vol. 53, no. 2, pp. 952-964, Feb. 2015.* The workflow consists in chaining 3 or 4 dedicated applications and produces a GIS vector file with artifact-free polygons corresponding to the segmented image, as well as mean and variance of the radiometry of each band for each polygon.

### Step 1: Mean-Shift Smoothing

The first step of the workflow is to perform Mean-Shift smoothing with the *MeanShiftSmoothing* application:

```
otbcli_MeanShiftSmoothing -in input_image
 -fout filtered_range.tif
 -foutpos filtered_spat.tif
 -ranger 30
 -spatialr 5
 -maxiter 10
 -modesearch 0
```

Note that the *modesearch* option should be disabled, and that the *foutpos* parameter is optional: it can be activated if you want to perform the segmentation based on both spatial and range modes.

This application will smooth large images by streaming them, and deactivating the *modesearch* will guarantee that the results will not depend on the streaming scheme. Please also note that the *maxiter* is used to set the margin to ensure these identical results, and as such increasing the *maxiter* may have an additional impact on processing time.

### Step 2: Segmentation

The next step is to produce an initial segmentation based on the smoothed images produced by the *MeanShiftSmoothing* application. To do so, the *LSMSSegmentation* will process them by tiles whose dimensions are defined by the *tilsizex* and *tilsizey* parameters, and by writing intermediate images to disk, thus keeping the memory consumption very low throughout the process. The segmentation will group together adjacent pixels whose range distance is below the *ranger* parameter and (optionally) spatial distance is below the *spatialr* parameter.

```
otbcli_LSMSSegmentation -in filtered_range.tif
 -inpos filtered_spatial.tif
 -out segmentation.tif uint32
 -ranger 30
```

```
-spatialr 5
-minsize 0
-tilesizex 256
-tilesizey 256
```

Note that the final segmentation image may contain a very large number of segments, and the *uint32* image type should therefore be used to ensure that there will be enough labels to index those segments. The *minsize* parameter will filter segments whose size in pixels is below its value, and their labels will be set to 0 (nodata).

Please note that the output segmented image may look patchy, as if there were tiling artifacts: this is because segments are numbered sequentially with respect to the order in which tiles are processed. You will see after the result of the vectorization step that there are no artifacts in the results.

The *LSMSSegmentation* application will write as many intermediate files as tiles needed during processing. As such, it may require twice as free disk space as the final size of the final image. The *cleanup* option (active by default) will clear the intermediate files during the processing as soon as they are not needed anymore. By default, files will be written to the current directory. The *tmpdir* option allows to specify a different directory for these intermediate files.

### Step 3 (optional): Merging small regions

The *LSMSSegmentation* application allows to filter out small segments. In the output segmented image, those segments will be removed and replaced by the background label (0). Another solution to deal with the small regions is to merge them with the closest big enough adjacent region in terms of radiometry. This is handled by the *LSMSSmallRegionsMerging* application, which will output a segmented image where small regions have been merged. Again, the *uint32* image type is advised for this output image.

```
otbcli_LSMSSmallRegionsMerging -in filtered_range.tif
 -inseg segmentation.tif
 -out segmentation_merged.tif uint32
 -minsize 10
 -tilesizex 256
 -tilesizey 256
```

The *minsize* parameter allows to specify the threshold on the size of the regions to be merged. Like the *LSMSSegmentation* application, this application will process the input images tile-wise to keep resources usage low, with the guarantee of identical results. You can set the tile size using the *tilesizex* and *tilesizey* parameters. However unlike the *LSMSSegmentation* application, it does not require to write any temporary file to disk.

### Step 4: Vectorization

The last step of the LSMS workflow consists in the vectorization of the segmented image into a GIS vector file. This vector file will contain one polygon per segment, and each of these polygons will hold additional attributes denoting the label of the original segment, the size of the segment in pixels, and the mean and variance of each band over the segment. The projection of the output GIS vector file will be the same as the projection from the input image (if input image has no projection, so does the output GIS file).

```
otbcli_LSMSVectorization -in input_image
 -inseg segmentation_merged.tif
 -out segmentation_merged.shp
 -tilesizex 256
 -tilesizey 256
```

This application will process the input images tile-wise to keep resources usage low, with the guarantee of identical results. You can set the tile size using the *tilesizex* and *tilesizey* parameters. However unlike the *LSMSSegmentation* application, it does not require to write any temporary file to disk.

## 5.4.5 Dempster Shafer based Classifier Fusion

This framework is dedicated to perform cartographic validation starting from the result of a detection (for example a road extraction), enhance the results reliability by using a classifier fusion algorithm. Using a set of descriptor, the processing chain validates or invalidates the input geometrical features.

### Fuzzy Model (requisite)

The *DSFuzzyModelEstimation* application performs the fuzzy model estimation (once by use case: descriptor set / Belief support / Plausibility support). It has the following input parameters :

- `-psin` a vector data of positive samples enriched according to the “Compute Descriptors” part
- `-nsin` a vector data of negative samples enriched according to the “Compute Descriptors” part
- `-belsup` a support for the Belief computation
- `-plasup` a support for the Plausibility computation
- `-desclist` an initialization model (xml file) or a descriptor name list (listing the descriptors to be included in the model)

The application can be used like this:

```
otbcli_DSfuzzyModelEstimation -psin PosSamples.shp
 -nsin NegSamples.shp
 -belsup "ROADSA"
 -plasup "NONDVI" "ROADSA" "NOBUIL"
 -desclist "NONDVI" "ROADSA" "NOBUIL"
 -out FuzzyModel.xml
```

The output file `FuzzyModel.xml` contains the optimal model to perform information fusion.

### First Step: Compute Descriptors

The first step in the classifier fusion based validation is to compute, for each studied polyline, the chosen descriptors. In this context, the *ComputePolylineFeatureFromImage* application can be used for a large range of descriptors. It has the following inputs :

- `-in` an image (of the studied scene) corresponding to the chosen descriptor (NDVI, building Mask...)
- `-vd` a vector data containing polyline of interest
- `-expr` a formula (“`b1 > 0.4`”, “`b1 == 0`”) where `b1` is the standard name of input image first band
- `-field` a field name corresponding to the descriptor codename (NONDVI, ROADSA...)

The output is a vector data containing polylines with a new field containing the descriptor value. In order to add the “NONDVI” descriptor to an input vector data (“`inVD.shp`”) corresponding to the percentage of pixels along a polyline that verifies the formula “`NDVI > 0.4`” :

```
otbcli_ComputePolylineFeatureFromImage -in NDVI.TIF
 -vd inVD.shp
 -expr "b1 > 0.4"
 -field "NONDVI"
 -out VD_NONDVI.shp
```

`NDVI.TIF` is the NDVI mono band image of the studied scene. This step must be repeated for each chosen descriptor:

```

otbcli_ComputePolylineFeatureFromImage -in roadSpectralAngle.TIF
 -vd VD_NONDVI.shp
 -expr "b1 > 0.24"
 -field "ROADSA"
 -out VD_NONDVI_ROADSA.shp

otbcli_ComputePolylineFeatureFromImage -in Buildings.TIF
 -vd VD_NONDVI_ROADSA.shp
 -expr "b1 == 0"
 -field "NOBUILDING"
 -out VD_NONDVI_ROADSA_NOBUIL.shp

```

Both `NDVI.TIF` and `roadSpectralAngle.TIF` can be produced using **Monteverdi** feature extraction capabilities, and `Buildings.TIF` can be generated using **Monteverdi** rasterization module. From now on, `VD_NONDVI_ROADSA_NOBUIL.shp` contains three descriptor fields. It will be used in the following part.

## Second Step: Feature Validation

The final application (*VectorDataDSValidation*) will validate or unvalidate the studied samples using the [Dempster-Shafer theory](#). Its inputs are :

- `-in` an enriched vector data “`VD_NONDVI_ROADSA_NOBUIL.shp`”
- `-belsup` a support for the Belief computation
- `-plasup` a support for the Plausibility computation
- `-descmod` a fuzzy model `FuzzyModel.xml`

The output is a vector data containing only the validated samples.

```

otbcli_VectorDataDSValidation -in extractedRoads_enriched.shp
 -descmod FuzzyModel.xml
 -out validatedSamples.shp

```

## 5.5 Classification

### 5.5.1 Pixel based classification

Orfeo ToolBox ships with a set of application to perform supervised pixel-based image classification. This framework allows to learn from multiple images, and using several machine learning method such as SVM, Bayes, KNN, Random Forests, Artificial Neural Network, and others...(see application help of `TrainImagesClassifier` and `TrainVectorClassifier` for further details about all the available classifiers). Here is an overview of the complete workflow:

1. Compute samples statistics for each image
2. Compute sampling rates for each image (only if more than one input image)
3. Select samples positions for each image
4. Extract samples measurements for each image
5. Compute images statistics
6. Train machine learning model from samples

## Samples statistics estimation

The first step of the framework is to know how many samples are available for each class in your image. The `PolygonClassStatistics` will do this job for you. This application processes a set of training geometries and an image and outputs statistics about available samples (i.e. pixel covered by the image and out of a no-data mask if provided), in the form of a XML file:

- number of samples per class
- number of samples per geometry

Supported geometries are polygons, lines and points. Depending on the geometry type, this application behaves differently:

- polygon: select pixels whose center falls inside the polygon
- lines: select pixels intersecting the line
- points: select closest pixel to the provided point

The application will require the input image, but it is only used to define the footprint in which samples will be selected. The user can also provide a raster mask, that will be used to discard pixel positions, using parameter `-mask`.

A simple use of the application `PolygonClassStatistics` could be as follows :

```
otbcli_PolygonClassStatistics -in LANDSAT_MultiTempIm_clip_GapF_20140309.tif
 -vec training.shp
 -field CODE
 -out classes.xml
```

The `-field` parameter is the name of the field that corresponds to class labels in the input geometries.

The output XML file will look like this:

```
<?xml version="1.0" ?>
<GeneralStatistics>
 <Statistic name="samplesPerClass">
 <StatisticMap key="11" value="56774" />
 <StatisticMap key="12" value="59347" />
 <StatisticMap key="211" value="25317" />
 <StatisticMap key="221" value="2087" />
 <StatisticMap key="222" value="2080" />
 <StatisticMap key="31" value="8149" />
 <StatisticMap key="32" value="1029" />
 <StatisticMap key="34" value="3770" />
 <StatisticMap key="36" value="941" />
 <StatisticMap key="41" value="2630" />
 <StatisticMap key="51" value="11221" />
 </Statistic>
 <Statistic name="samplesPerVector">
 <StatisticMap key="0" value="3" />
 <StatisticMap key="1" value="2" />
 <StatisticMap key="10" value="86" />
 <StatisticMap key="100" value="21" />
 <StatisticMap key="1000" value="3" />
 <StatisticMap key="1001" value="27" />
 <StatisticMap key="1002" value="7" />
 ...
 </Statistic>
</GeneralStatistics>
```

## Samples selection

Now, we know exactly how many samples are available in the image for each class and each geometry in the training set. From this statistics, we can now compute the sampling rates to apply for each classes, and perform the sample selection. This will be done by the `SampleSelection` application.

There are several strategies to compute those sampling rates:

- **Constant strategy:** All classes will be sampled with the same number of samples, which is user-defined.
- **Smallest class strategy:** The class with the least number of samples will be fully sampled. All other classes will be sampled with the same number of samples.
- **Percent strategy:** Each class will be sampled with a user-defined percentage (same value for all classes) of samples available in this class.
- **Total strategy:** A global number of samples to generate is divided proportionally among each class (classes proportions are enforced).
- **Take all strategy:** Take all the available samples
- **By class strategy:** Set a target number of samples for each class. The number of samples for each class is read from a CSV file.

To actually select the sample positions, there are two available sampler:

- **Random:** Randomly select samples while respecting the sampling rate
- **Periodic:** Sample periodically using the sampling rate

The application will make sure that samples spans the whole training set extent by adjusting the sampling rate. Depending on the strategy to determine the sampling rate, some geometries of the training set might not be sampled.

The application will accept as input the input image and training geometries, as well class statistics XML file computed during previous step. It will output a vector file containing point geometries which indicate the location of the samples.

```
otbcli_SampleSelection -in LANDSAT_MultiTempIm_clip_GapF_20140309.tif
 -vec training.shp
 -instats classes.xml
 -field CODE
 -strategy smallest
 -outrates rates.csv
 -out samples.sqlite
```

The csv file written by the optional `-outrates` parameter sums-up what has been done during samples selection:

```
#className requiredSamples totalSamples rate
11 941 56774 0.0165745
12 941 59347 0.0158559
211 941 25317 0.0371687
221 941 2087 0.450886
222 941 2080 0.452404
31 941 8149 0.115474
32 941 1029 0.91448
34 941 3770 0.249602
36 941 941 1
41 941 2630 0.357795
51 941 11221 0.0838606
```



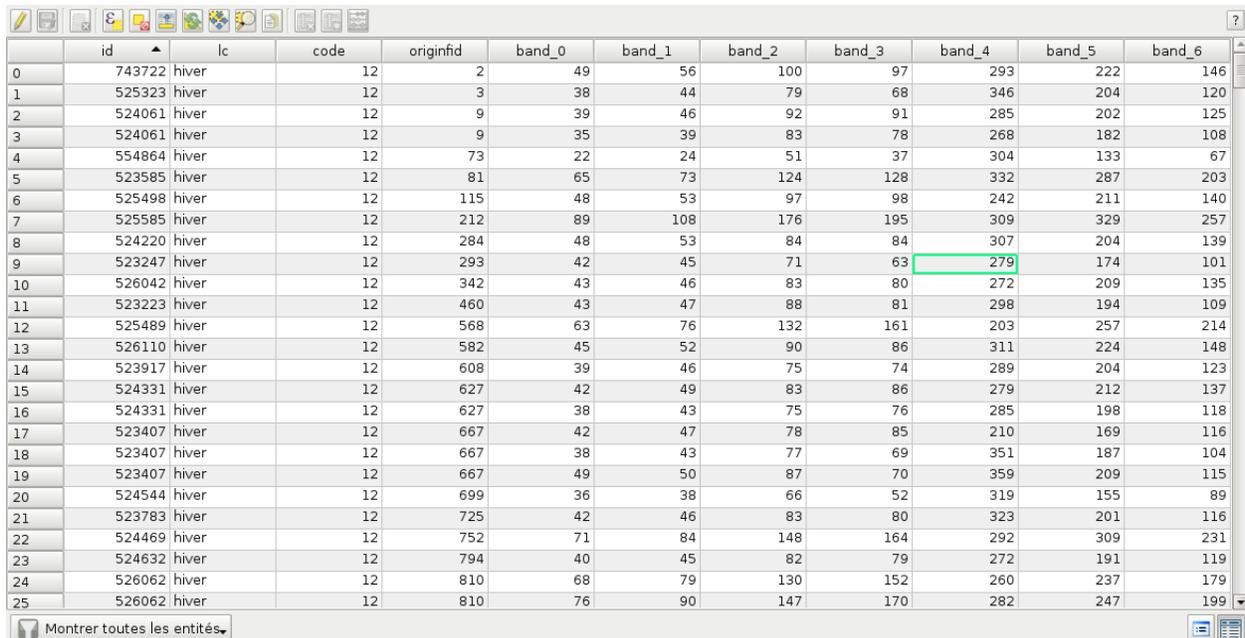
Figure 5.1: This image shows the polygons of the training with a color corresponding to their class. The red dot shows the samples that have been selected.

## Samples extraction

Now that we selected the location of the samples, we will attach measurement to them. This is the purpose of the `SampleExtraction` application. It will walk through the list of samples and extract the underlying pixel values. If no `-out` parameter is given, the `SampleExtraction` application can work in update mode, thus allowing to extract features from multiple images of the same location.

Features will be stored in fields attached to each sample. Field name can be generated from a prefix a sequence of numbers (i.e. if prefix is `feature_` then features will be named `feature_0`, `feature_1`, ...). This can be achieved with the `-outfield` prefix option. Alternatively, one can set explicit names for all features using the `-outfield` list option.

```
otbcli_SampleExtraction -in LANDSAT_MultiTempIm_clip_GapF_20140309.tif
 -vec samples.sqlite
 -outfield prefix
 -outfield.prefix.name band_
 -field CODE
```



	id	lc	code	originfid	band_0	band_1	band_2	band_3	band_4	band_5	band_6
0	743722	hiver	12	2	49	56	100	97	293	222	146
1	525323	hiver	12	3	38	44	79	68	346	204	120
2	524061	hiver	12	9	39	46	92	91	285	202	125
3	524061	hiver	12	9	35	39	83	78	268	182	108
4	554864	hiver	12	73	22	24	51	37	304	133	67
5	523585	hiver	12	81	65	73	124	128	332	287	203
6	525498	hiver	12	115	48	53	97	98	242	211	140
7	525585	hiver	12	212	89	108	176	195	309	329	257
8	524220	hiver	12	284	48	53	84	84	307	204	139
9	523247	hiver	12	293	42	45	71	63	279	174	101
10	526042	hiver	12	342	43	46	83	80	272	209	135
11	523223	hiver	12	460	43	47	88	81	298	194	109
12	525489	hiver	12	568	63	76	132	161	203	257	214
13	526110	hiver	12	582	45	52	90	86	311	224	148
14	523917	hiver	12	608	39	46	75	74	289	204	123
15	524331	hiver	12	627	42	49	83	86	279	212	137
16	524331	hiver	12	627	38	43	75	76	285	198	118
17	523407	hiver	12	667	42	47	78	85	210	169	116
18	523407	hiver	12	667	38	43	77	69	351	187	104
19	523407	hiver	12	667	49	50	87	70	359	209	115
20	524544	hiver	12	699	36	38	66	52	319	155	89
21	523783	hiver	12	725	42	46	83	80	323	201	116
22	524469	hiver	12	752	71	84	148	164	292	309	231
23	524632	hiver	12	794	40	45	82	79	272	191	119
24	526062	hiver	12	810	68	79	130	152	260	237	179
25	526062	hiver	12	810	76	90	147	170	282	247	199

Figure 5.2: Attributes table of the updated samples file.

## Working with several images

If the training set spans several images, the `MultiImageSamplingRate` application allows to compute the appropriate sampling rates per image and per class, in order to get samples that spans the whole images coverage.

It is first required to run the `PolygonClassStatistics` application on each image of the set separately. The `MultiImageSamplingRate` application will then read all the produced statistics XML files and derive the sampling rates according the sampling strategy. For more information, please refer to the [Samples statistics estimation](#) section.

There are 3 modes for the sampling rates estimation from multiple images:

- **Proportional mode:** For each class, the requested number of samples is divided proportionally among the images.

- **Equal mode:** For each class, the requested number of samples is divided equally among the images.
- **Custom mode:** The user indicates the target number of samples for each image.

The different behaviors for each mode and strategy are described as follows.

$T_i(c)$  and  $N_i(c)$  refers resp. to the total number and needed number of samples in image  $i$  for class  $c$ . Let's call  $L$  the total number of image.

- **Strategy = all**
  - Same behavior for all modes proportional, equal, custom : take all samples
- **Strategy = constant** (let's call  $M$  the global number of samples per class required)
  - *Mode = proportional:* For each image  $i$  and each class  $c$ ,  $N_i(c) = \frac{M * T_i(c)}{\sum_k (T_k(c))}$
  - *Mode = equal:* For each image  $i$  and each class  $c$ ,  $N_i(c) = \frac{M}{L}$
  - *Mode = custom:* For each image  $i$  and each class  $c$ ,  $N_i(c) = M_i$  where  $M_i$  is the custom requested number of samples for image  $i$
- **Strategy = byClass** (let's call  $M(c)$  the global number of samples for class  $c$ )
  - *Mode = proportional:* For each image  $i$  and each class  $c$ ,  $N_i(c) = M(c) * \frac{T_i(c)}{\sum_k (T_k(c))}$
  - *Mode = equal:* For each image  $i$  and each class  $c$ ,  $N_i(c) = \frac{M(c)}{L}$
  - *Mode = custom:* For each image  $i$  and each class  $c$ ,  $N_i(c) = M_i(c)$  where  $M_i(c)$  is the custom requested number of samples for each image  $i$  and each class  $c$
- **Strategy = percent**
  - *Mode = proportional:* For each image  $i$  and each class  $c$ ,  $N_i(c) = p * T_i(c)$  where  $p$  is the user-defined percentage
  - *Mode = equal:* For each image  $i$  and each class  $c$ ,  $N_i(c) = p * \frac{\sum_k (T_k(c))}{L}$  where  $p$  is the user-defined percentage
  - *Mode = custom:* For each image  $i$  and each class  $c$ ,  $N_i(c) = p(i) * T_i(c)$  where  $p(i)$  is the user-defined percentage for image  $i$
- **Strategy = total**
  - *Mode = proportional:* For each image  $i$  and each class  $c$ ,  $N_i(c) = total * (\frac{\sum_k (T_i(k))}{\sum_k (T_l(k))}) * (\frac{T_i(c)}{\sum_k (T_i(k))})$  where  $total$  is the total number of samples specified
  - *Mode = equal:* For each image  $i$  and each class  $c$ ,  $N_i(c) = (total/L) * (\frac{T_i(c)}{\sum_k (T_i(k))})$  where  $total$  is the total number of samples specified
  - *Mode = custom:* For each image  $i$  and each class  $c$ ,  $N_i(c) = total(i) * (\frac{T_i(c)}{\sum_k (T_i(k))})$  where  $total(i)$  is the total number of samples specified for image  $i$
- **Strategy = smallest class**
  - *Mode = proportional:* the smallest class is computed globally, then this smallest size is used for the strategy constant+proportional
  - *Mode = equal:* the smallest class is computed globally, then this smallest size is used for the strategy constant+equal
  - *Mode = custom:* the smallest class is computed and used for each image separately

The `MultiImageSamplingRate` application can be used as follows:

```
otbcli_MultiImageSamplingRate -il stats1.xml stats2.xml stats3.xml
 -out rates.csv
 -strategy smallest
 -mim proportional
```

The output filename from `-out` parameter will be used to generate as many filenames as necessary (e.g. one per input filename), called `rates_1.csv`, `rates_2.csv` ...

Once rates are computed for each image, sample selection can be performed on each corresponding image using the by class strategy:

```
otbcli_SampleSelection -in img1.tif
 -vec training.shp
 -instats stats1.xml
 -field CODE
 -strategy byclass
 -strategy.byclass.in rates_1.csv
 -out samples1.sqlite
```

Samples extraction can then be performed on each image by following the [Samples extraction](#) step. The learning application can process several samples files.

### Images statistics estimation

Some machine learning algorithms converge faster if the range of features is  $[-1, 1]$  or  $[0, 1]$ . Other will be sensitive to relative ranges between feature, e.g. a feature with a larger range might have more weight in the final decision. This is for instance the case for machine learning algorithm using euclidean distance at some point to compare features. In those cases, it is advised to normalize all features to the range  $[-1, 1]$  before performing the learning. For this purpose, the `ComputeImageStatistics` application allows to compute and output to an XML file the mean and standard deviation based on pooled variance of each band for one or several images.

```
otbcli_ComputeImageStatistics -il im1.tif im2.tif im3.tif
 -out images_statistics.xml
```

The output statistics file can then be fed to the training and classification applications.

### Training the model

Now that the training samples are ready, we can perform the learning using the `TrainVectorClassifier` application.

```
otbcli_TrainVectorClassifier -io.vd samples.sqlite
 -cfield CODE
 -io.out model.rf
 -classifier rf
 -feat band_0 band_1 band_2 band_3 band_4 band_5 band_6
```

The `-classifier` parameter allows to choose which machine learning model algorithm to train. Please refer to the `TrainVectorClassifier` application reference documentation.

In case of multiple samples files, you can add them to the `-io.vd` parameter (see [Working with several images](#) section).

The feature to be used for training must be explicitly listed using the `-feat` parameter. Order of the list matters.

If you want to use a statistic file for features normalization, you can pass it using the `-io.stats` parameter. Make sure that the order of feature statistics in the statistics file matches the order of feature passed to the `-feat` option.

The field in vector data allowing to specify the label of each sample can be set using the `-cfield` option.

By default, the application will estimate the trained classifier performances on the same set of samples that has been used for training. The `-io.vd` parameter allows to specify a different samples file for this purpose, for a more fair estimation of the performances. Note that this performances estimation scheme can also be estimated afterward (see [Validating the classification model](#) section).

### Using the classification model

Once the classifier has been trained, one can apply the model to classify pixel inside defined classes on a new image using the *ImageClassifier* application:

```
otbcli_ImageClassifier -in image.tif
 -model model.rf
 -out labeled_image.tif
```

You can set an input mask to limit the classification to the mask area with value `>0`.

```
-imstat images_statistics.xml
```

### Validating the classification model

The performance of the model generated by the *TrainImagesClassifier* application is directly estimated by the application itself, which displays the precision, recall and F-score of each class, and can generate the global confusion matrix as an output \*.CSV file.

With the *ComputeConfusionMatrix* application, it is also possible to estimate the performance of a model from a classification map generated with the *ImageClassifier* application. This labeled image is compared to positive reference samples (either represented as a raster labeled image or as a vector data containing the reference classes). It will compute the confusion matrix and precision, recall and F-score of each class too, based on the *ConfusionMatrixCalculator* class.

```
otbcli_ComputeConfusionMatrix -in labeled_image.tif
 -ref vector
 -ref.vector.in vectordata.shp
 -ref.vector.field Class (name_of_label_field)
 -out confusion_matrix.csv
```

### Fancy classification results

Color mapping can be used to apply color transformations on the final gray level label image. It allows to get an RGB classification map by re-mapping the image values to be suitable for display purposes. One can use the *ColorMapping* application. This tool will replace each label with an 8-bits RGB color specified in a mapping file. The mapping file should look like this :

```
Lines beginning with a # are ignored
1 255 0 0
```

In the previous example, 1 is the label and 255 0 0 is a RGB color (this one will be rendered as red). To use the mapping tool, enter the following :

```
otbcli_ColorMapping -in labeled_image.tif
 -method custom
 -method.custom.lut lut_mapping_file.txt
 -out RGB_color_image.tif
```

Other look-up tables (LUT) are available : standard continuous LUT, optimal LUT, and LUT computed over a support image.

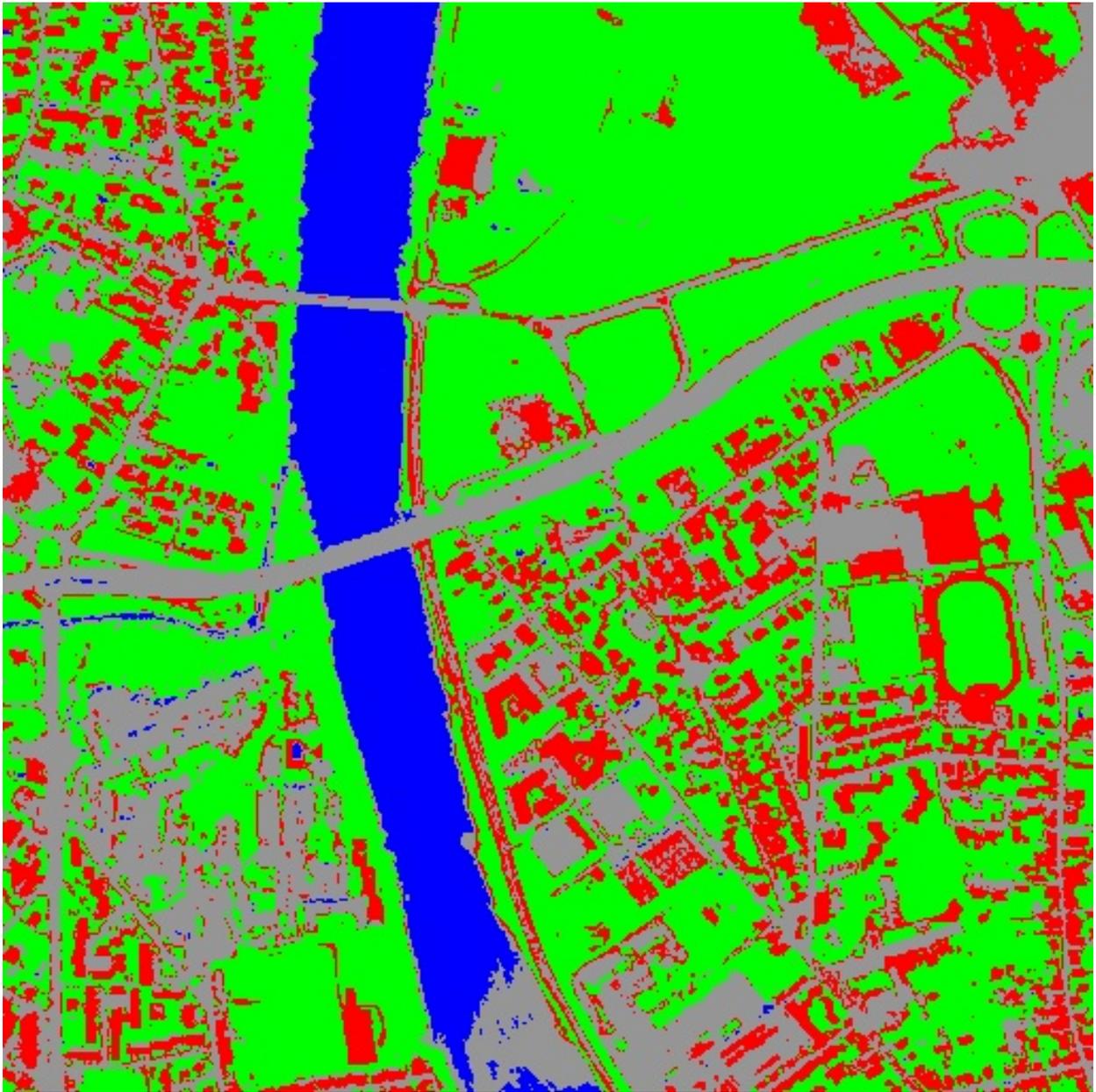
### Example

We consider 4 classes: water, roads, vegetation and buildings with red roofs. Data is available in the OTB-Data repository and this image is produced with the commands inside this file .



Figure 2: From left to right: Original image, result image with fusion (with monteverdi viewer) of original image and fancy classification and input image with fancy color classification from labeled image.





## 5.5.2 Fusion of classification maps

After having processed several classifications of the same input image but from different models or methods (SVM, KNN, Random Forest,...), it is possible to make a fusion of these classification maps with the *FusionOfClassifications* application which uses either majority voting or the Dempster-Shafer framework to handle this fusion. The Fusion of Classifications generates a single more robust and precise classification map which combines the information extracted from the input list of labeled images.

The *FusionOfClassifications* application has the following input parameters :

- `-il` list of input labeled classification images to fuse
- `-out` the output labeled image resulting from the fusion of the input classification images
- `-method` the fusion method (either by majority voting or by Dempster Shafer)
- `-nodatalabel` label for the no data class (default value = 0)
- `-undecidedlabel` label for the undecided class (default value = 0)

The input pixels with the no-data class label are simply ignored by the fusion process. Moreover, the output pixels for which the fusion process does not result in a unique class label, are set to the undecided value.

### Majority voting for the fusion of classifications

In the Majority Voting method implemented in the *FusionOfClassifications* application, the value of each output pixel is equal to the more frequent class label of the same pixel in the input classification maps. However, it may happen that the more frequent class labels are not unique in individual pixels. In that case, the undecided label is attributed to the output pixels.

The application can be used like this:

```
otbcli_FusionOfClassifications -il cmap1.tif cmap2.tif cmap3.tif
 -method majorityvoting
 -nodatalabel 0
 -undecidedlabel 10
 -out MVFusedClassificationMap.tif
```

Let us consider 6 independent classification maps of the same input image (Cf. left image in *Figure 1*) generated from 6 different SVM models. The *Figure 2* represents them after a color mapping by the same LUT. Thus, 4 classes (water: blue, roads: gray,vegetation: green, buildings with red roofs: red) are observable on each of them.

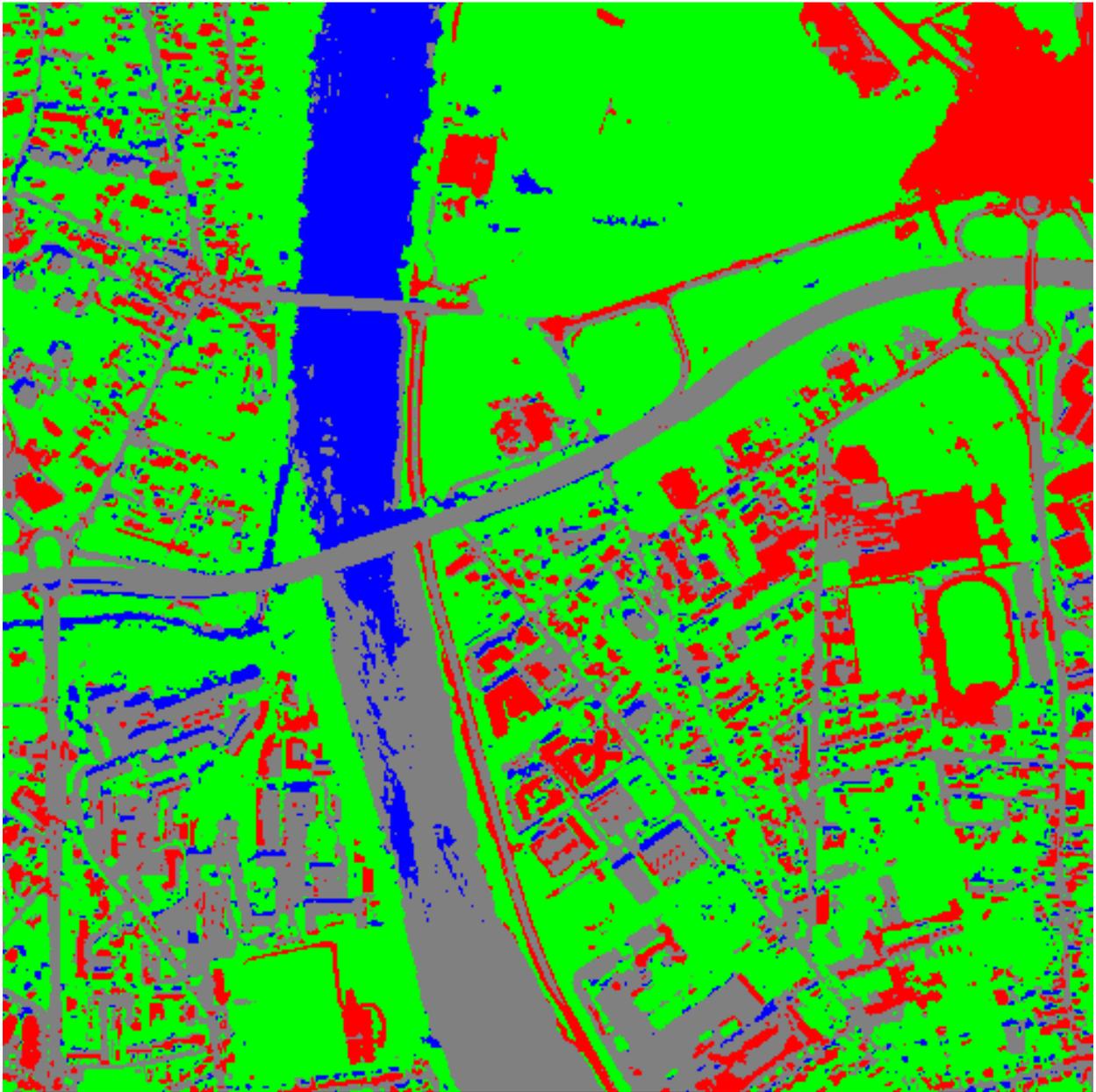
Figure 3: Six fancy colored classified images to be fused, generated from 6 different SVM models.

As an example of the *FusionOfClassifications* application by *majority voting*, the fusion of the six input classification maps represented in *Figure 3* leads to the classification map illustrated on the right in *Figure 4*. Thus, it appears that this fusion highlights the more relevant classes among the six different input classifications. The white parts of the fused image correspond to the undecided class labels, i.e. to pixels for which there is not a unique majority voting.

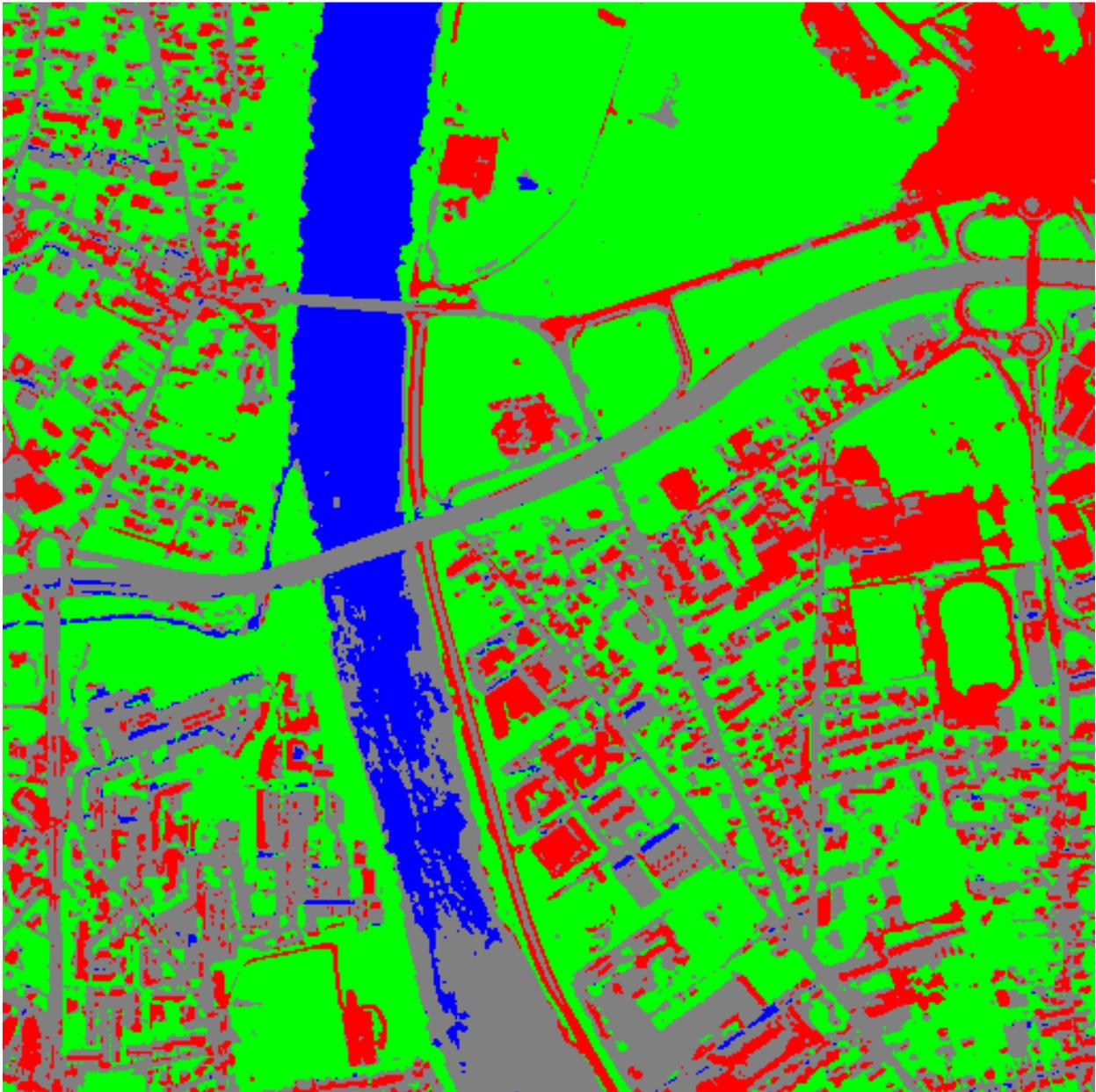
Figure 4: From left to right: Original image, and fancy colored classified image obtained by a majority voting fusion of the 6 classification maps represented in Fig. 4.13 (water: blue, roads: gray, vegetation: green, buildings with red roofs: red, undecided: white)

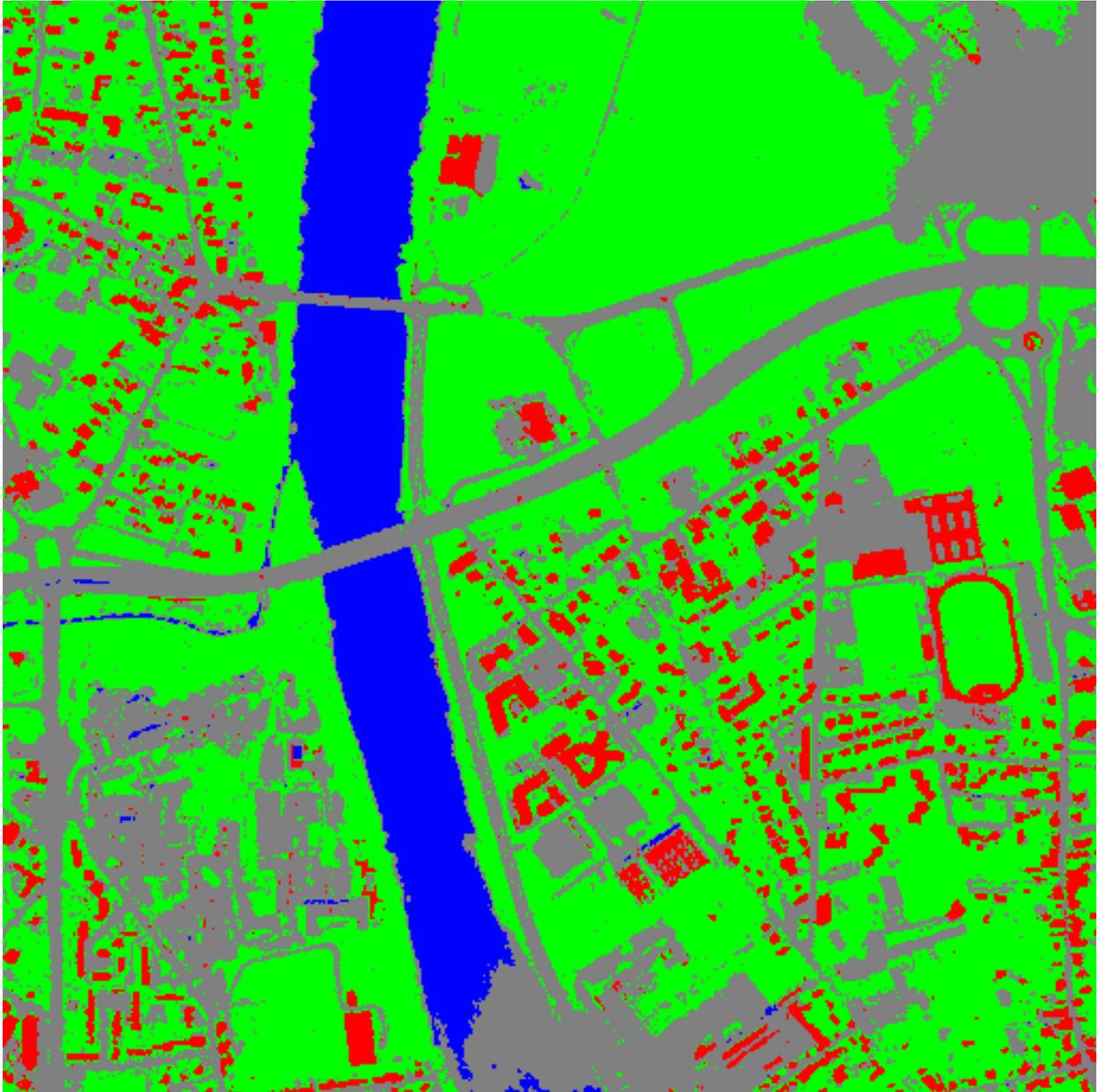
### Dempster Shafer framework for the fusion of classifications

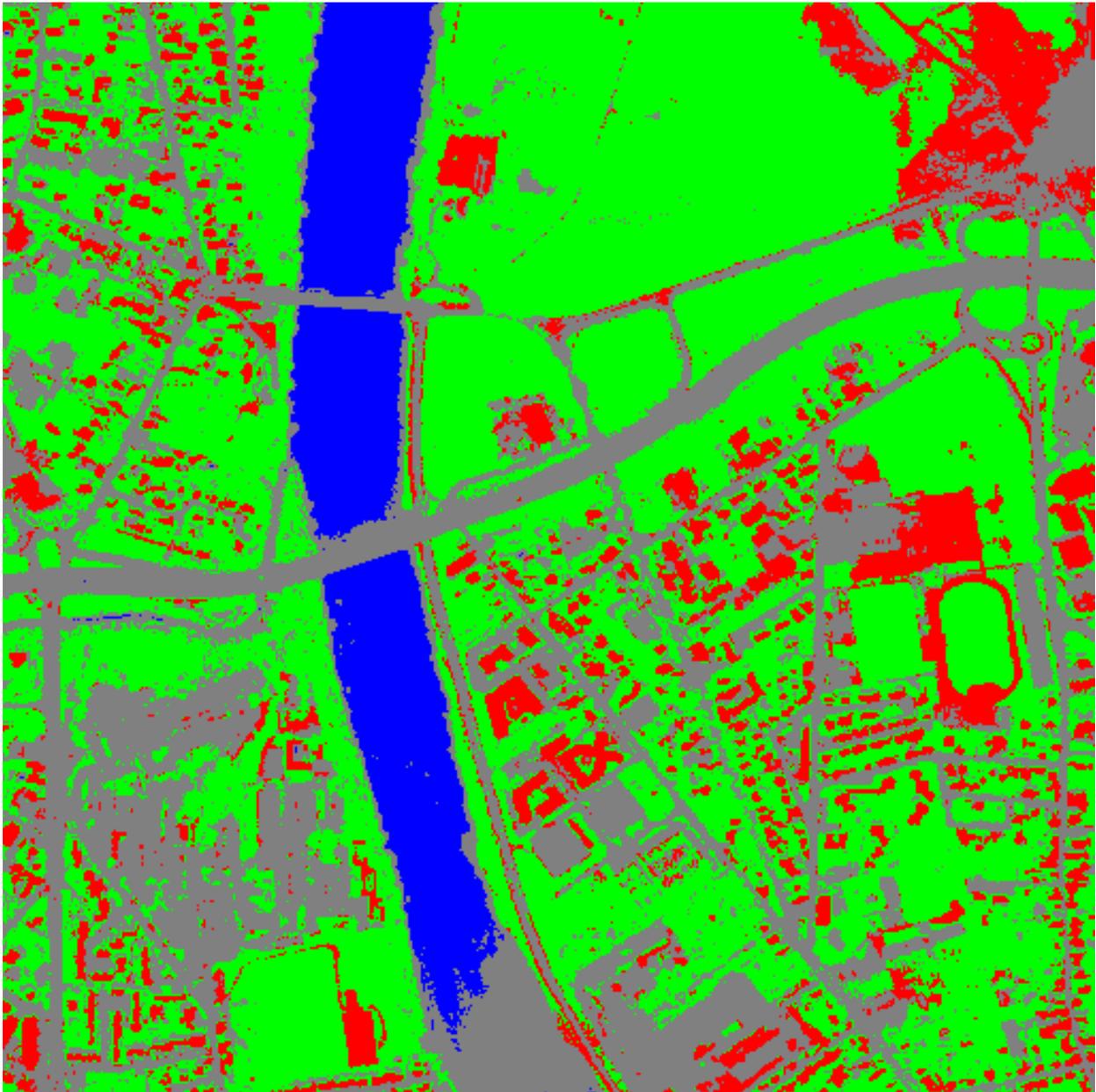
The *FusionOfClassifications* application, handles another method to compute the fusion: the Dempster Shafer framework. In the [Dempster-Shafer theory](#) , the performance of each classifier resulting in the classification maps to fuse are evaluated with the help of the so-called *belief function* of each class label, which measures the degree of belief that the corresponding label is correctly assigned to a pixel. For each classifier, and for each class label, these belief functions

















are estimated from another parameter called the *mass of belief* of each class label, which measures the confidence that the user can have in each classifier according to the resulting labels.

In the Dempster Shafer framework for the fusion of classification maps, the fused class label for each pixel is the one with the maximal belief function. In case of multiple class labels maximizing the belief functions, the output fused pixels are set to the undecided value.

In order to estimate the confidence level in each classification map, each of them should be confronted with a ground truth. For this purpose, the masses of belief of the class labels resulting from a classifier are estimated from its confusion matrix, which is itself exported as a \*.CSV file with the help of the *ComputeConfusionMatrix* application. Thus, using the Dempster-Shafer method to fuse classification maps needs an additional input list of such \*.CSV files corresponding to their respective confusion matrices.

The application can be used like this:

```
otbcli_FusionOfClassifications -il cmap1.tif cmap2.tif cmap3.tif
 -method dempstershafer
 -method.dempstershafer.cmfl
 cmat1.csv cmat2.csv cmat3.csv
 -nodatalabel 0
 -undecidedlabel 10
 -out DSFusedClassificationMap.tif
```

As an example of the *FusionOfClassifications* application by *Dempster Shafer*, the fusion of the six input classification maps represented in *Figure 3* leads to the classification map illustrated on the right in *Figure 5* [fig:ClassificationMapFusionApplicationDS]. Thus, it appears that this fusion gives access to a more precise and robust classification map based on the confidence level in each classifier.

Figure 5: From left to right: Original image, and fancy colored classified image obtained by a Dempster-Shafer fusion of the 6 classification maps represented in Fig. 4.13 (water: blue, roads: gray, vegetation: green, buildings with red roofs: red, undecided: white).

## Recommendations to properly use the fusion of classification maps

In order to properly use the *FusionOfClassifications* application, some points should be considered. First, the `list_of_input_images` and `OutputFusedClassificationImage` are single band labeled images, which means that the value of each pixel corresponds to the class label it belongs to, and labels in each classification map must represent the same class. Secondly, the undecided label value must be different from existing labels in the input images in order to avoid any ambiguity in the interpretation of the `OutputFusedClassificationImage`.

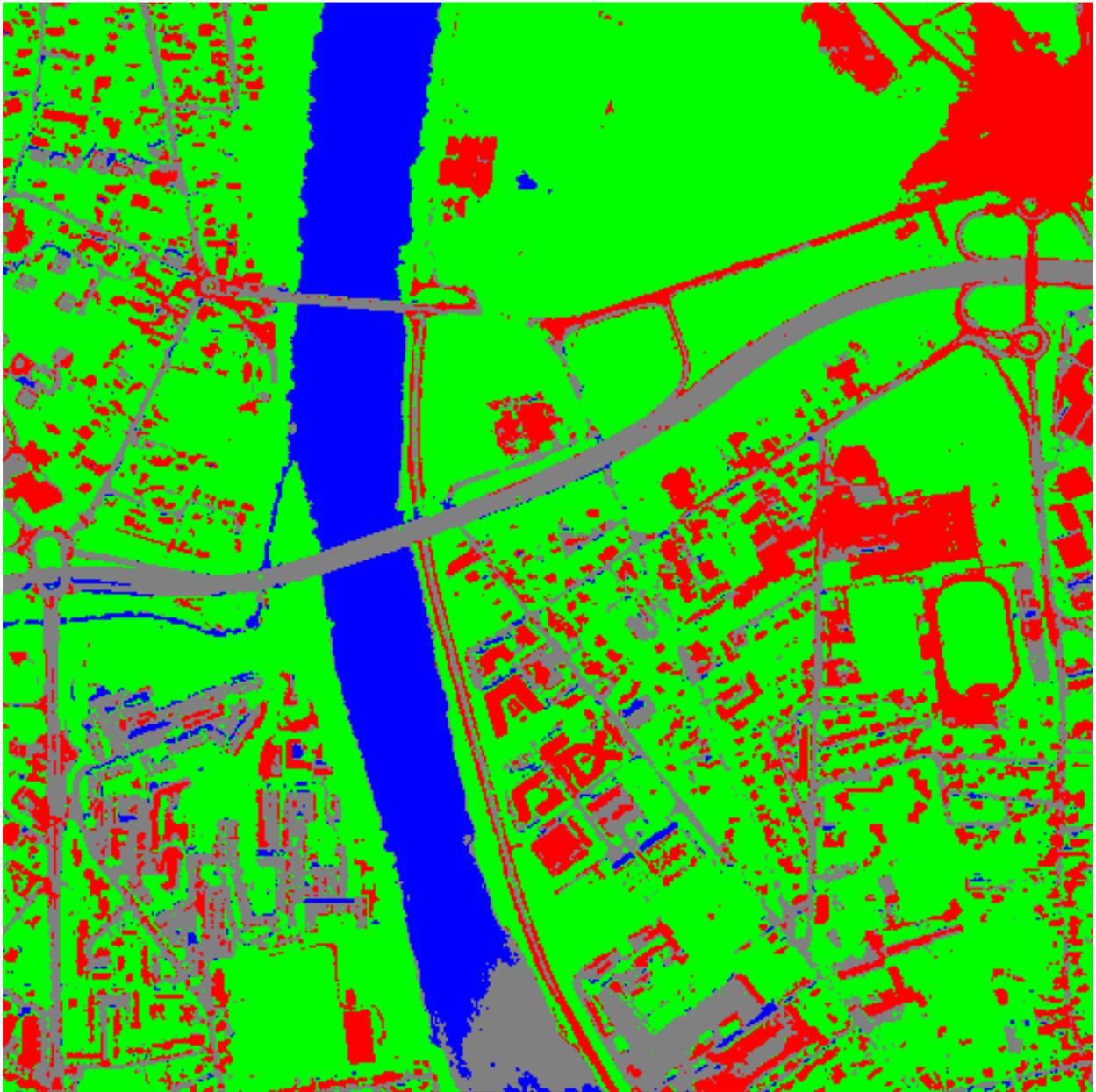
### 5.5.3 Majority voting based classification map regularization

Resulting classification maps can be regularized in order to smooth irregular classes. Such a regularization process improves classification results by making more homogeneous areas which are easier to handle.

#### Majority voting for the classification map regularization

The *ClassificationMapRegularization* application performs a regularization of a labeled input image based on the Majority Voting method in a specified ball shaped neighborhood. For each center pixel, Majority Voting takes the more representative value of all the pixels identified by the structuring element and then sets the output center pixel to this majority label value. The ball shaped neighborhood is identified by its radius expressed in pixels.





## Handling ambiguity and not classified pixels in the majority voting based regularization

Since, the Majority Voting regularization may lead to not unique majority labels in the neighborhood, it is important to define which behaviour the filter must have in this case. For this purpose, a Boolean parameter (called `ip.suvbool`) is used in the *ClassificationMapRegularization* application to choose whether pixels with more than one majority class are set to Undecided (true), or to their Original labels (false = default value).

Moreover, it may happen that pixels in the input image do not belong to any of the considered class. Such pixels are assumed to belong to the NoData class, the label of which is specified as an input parameter for the regularization. Therefore, those NoData input pixels are invariant and keep their NoData label in the output regularized image.

The *ClassificationMapRegularization* application has the following input parameters :

- `-io.in` labeled input image resulting from a previous classification process
- `-io.out` output labeled image corresponding to the regularization of the input image
- `-ip.radius` integer corresponding to the radius of the ball shaped structuring element (default value = 1 pixel)
- `-ip.suvbool` boolean parameter used to choose whether pixels with more than one majority class are set to Undecided (true), or to their Original labels (false = default value). Please note that the Undecided value must be different from existing labels in the input image
- `-ip.nodatalabel` label for the NoData class. Such input pixels keep their NoData label in the output image (default value = 0)
- `-ip.undecidedlabel` label for the Undecided class (default value = 0).

The application can be used like this:

```
otbcli_ClassificationMapRegularization -io.in labeled_image.tif
 -ip.radius 3
 -ip.suvbool true
 -ip.nodatalabel 10
 -ip.undecidedlabel 7
 -io.out regularized.tif
```

## Recommendations to properly use the majority voting based regularization

In order to properly use the *ClassificationMapRegularization* application, some points should be considered. First, both `InputLabeledImage` and `OutputLabeledImage` are single band labeled images, which means that the value of each pixel corresponds to the class label it belongs to. The `InputLabeledImage` is commonly an image generated with a classification algorithm such as the SVM classification. Remark: both `InputLabeledImage` and `OutputLabeledImage` are not necessarily of the same type. Secondly, if `ip.suvbool == true`, the Undecided label value must be different from existing labels in the input labeled image in order to avoid any ambiguity in the interpretation of the regularized `OutputLabeledImage`. Finally, the structuring element radius must have a minimum value equal to 1 pixel, which is its default value. Both NoData and Undecided labels have a default value equal to 0.

### Example

Resulting from the application presented in section *Fancy classification results* and illustrated in Fig. [fig:MeanShiftVectorImageFilter], the Fig.[fig:ClassificationMapRegularizationApplication] shows a regularization of a classification map composed of 4 classes: water, roads, vegetation and buildings with red roofs. The radius of the ball shaped structuring element is equal to 3 pixels, which corresponds to a ball included in a 7 x 7 pixels square. Pixels with more than one majority class keep their original labels.

**!image! !image!** [fig:ClassificationMapRegularizationApplication]

## 5.5.4 Regression

The machine learning models in OpenCV and LibSVM also support a regression mode : they can be used to predict a numeric value (i.e. not a class index) from an input predictor. The workflow is the same as classification. First, the regression model is trained, then it can be used to predict output values. The applications to do that are and .



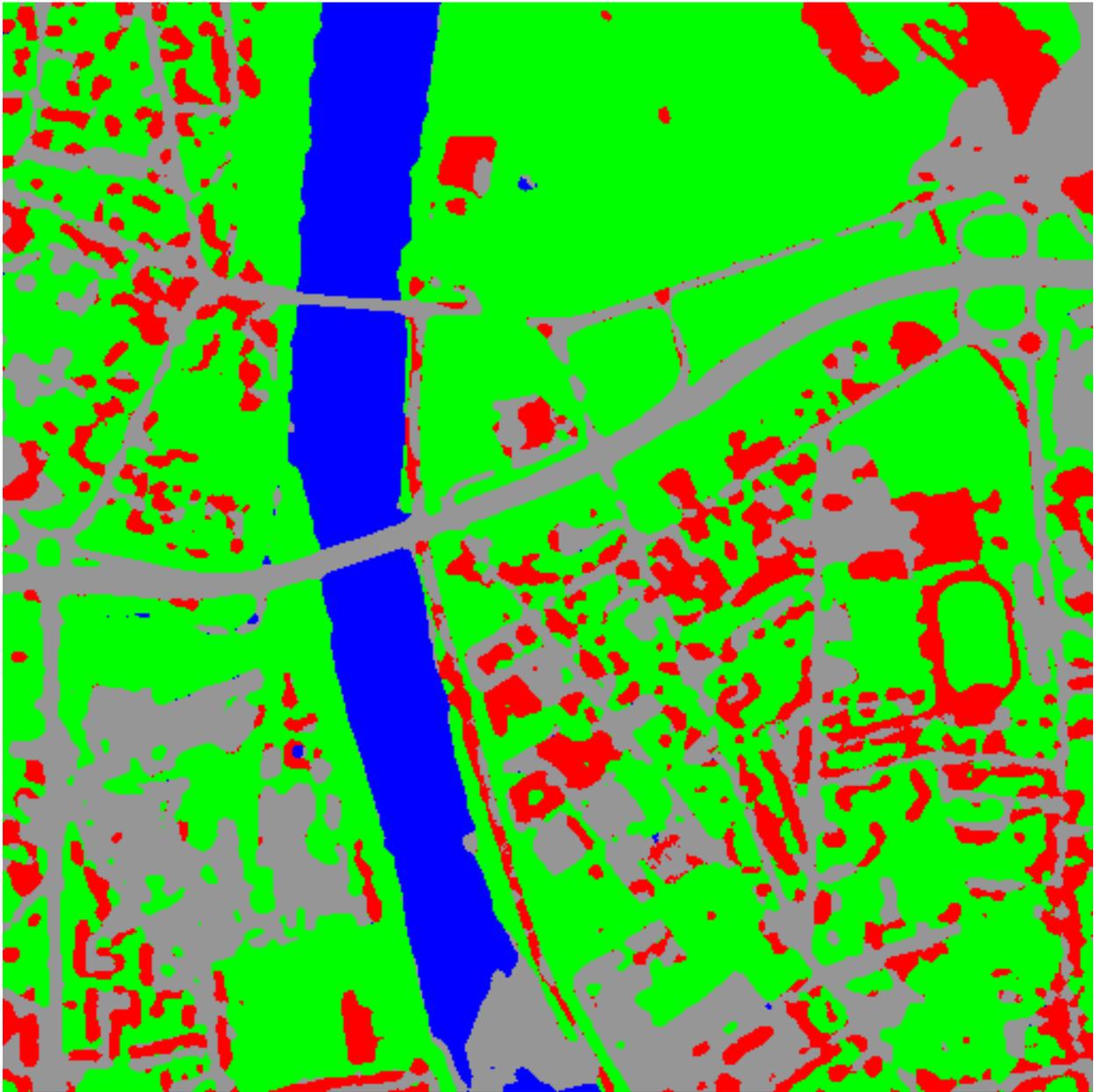
Figure 6: From left to right: Original image, fancy colored classified image and regularized classification map with radius equal to 3 pixels.

The input data set for training must have the following structure :

- $n$  components for the input predictors
- one component for the corresponding output value

The application supports 2 input formats :





- An image list : each image should have components matching the structure detailed earlier ( $n$  feature components + 1 output value)
- A CSV file : the first  $n$  columns are the feature components and the last one is the output value

If you have separate images for predictors and output values, you can use the application.

```
otbcli_ConcatenateImages -il features.tif output_value.tif
 -out training_set.tif
```

## Statistics estimation

As in classification, a statistics estimation step can be performed before training. It allows to normalize the dynamic of the input predictors to a standard one : zero mean, unit standard deviation. The main difference with the classification case is that with regression, the dynamic of output values can also be reduced.

The statistics file format is identical to the output file from application, for instance :

```
<?xml version="1.0" ?>
<FeatureStatistics>
 <Statistic name="mean">
 <StatisticVector value="198.796" />
 <StatisticVector value="283.117" />
 <StatisticVector value="169.878" />
 <StatisticVector value="376.514" />
 </Statistic>
 <Statistic name="stddev">
 <StatisticVector value="22.6234" />
 <StatisticVector value="41.4086" />
 <StatisticVector value="40.6766" />
 <StatisticVector value="110.956" />
 </Statistic>
</FeatureStatistics>
```

In the application, normalization of input predictors and output values is optional. There are 3 options :

- No statistic file : normalization disabled
- Statistic file with  $n$  components : normalization enabled for input predictors only
- Statistic file with  $n+1$  components : normalization enabled for input predictors and output values

If you use an image list as training set, you can run application. It will produce a statistics file suitable for input and output normalization (third option).

```
otbcli_ComputeImagesStatistics -il training_set.tif
 -out stats.xml
```

## Training

Initially, the machine learning models in OTB only used classification. But since they come from external libraries (OpenCV and LibSVM), the regression mode was already implemented in these external libraries. So the integration of these models in OTB has been improved in order to allow the usage of regression mode. As a consequence , the machine learning models have nearly the same set of parameters for classification and regression mode.

- Decision Trees
- Gradient Boosted Trees
- Neural Network

- Random Forests
- K-Nearest Neighbors

The behavior of application is very similar to `.train`. From the input data set, a portion of the samples is used for training, whereas the other part is used for validation. The user may also set the model to train and its parameters. Once the training is done, the model is stored in an output file.

```
otbcli_TrainRegression -io.il training_set.tif
 -io.imstat stats.xml
 -io.out model.txt
 -sample.vtr 0.5
 -classifier knn
 -classifier.knn.k 5
 -classifier.knn.rule median
```

## Prediction

Once the model is trained, it can be used in application to perform the prediction on an entire image containing input predictors (i.e. an image with only  $n$  feature components). If the model was trained with normalization, the same statistic file must be used for prediction. The behavior of `otbcli_PredictRegression` with respect to statistic file is identical to :

- no statistic file : normalization off
- $n$  components : input only
- $n+1$  components : input and output

The model to use is read from file (the one produced during training).

```
otbcli_PredictRegression -in features_bis.tif
 -model model.txt
 -imstat stats.xml
 -out prediction.tif
```

## 5.6 Feature extraction

As described in the OTB Software Guide, the term *Feature Extraction* refers to techniques aiming at extracting added value information from images. These extracted items named *features* can be local statistical moments, edges, radiometric indices, morphological and textural properties. For example, such features can be used as input data for other image processing methods like *Segmentation* and *Classification*.

### 5.6.1 Local statistics extraction

This application computes the 4 local statistical moments on every pixel in the selected channel of the input image, over a specified neighborhood. The output image is multi band with one statistical moment (feature) per band. Thus, the 4 output features are the Mean, the Variance, the Skewness and the Kurtosis. They are provided in this exact order in the output image.

The *LocalStatisticExtraction* application has the following input parameters:

- `--in` the input image to compute the features on
- `--channel` the selected channel index in the input image to be processed (default value is 1)
- `--radius` the computational window radius (default value is 3 pixels)

--out the output image containing the local statistical moments

The application can be used like this:

```
otbcli_LocalStatisticExtraction -in InputImage
 -channel 1
 -radius 3
 -out OutputImage
```

## 5.6.2 Edge extraction

This application Computes edge features on every pixel in the selected channel of the input image.

The *EdgeExtraction* application has the following input parameters:

--in the input image to compute the features on

--channel the selected channel index in the input image to be processed (default value is 1)

- -filter the choice of edge detection method (gradient/sobel/touzi) (default value is gradient)
  - (-filter.touzi.xradius) the X Radius of the Touzi processing neighborhood (only if filter==touzi) (default value is 1 pixel) \_\_
  - (-filter.touzi.yradius) the Y Radius of the Touzi processing neighborhood (only if filter==touzi) (default value is 1 pixel)

--out the output mono band image containing the edge features

The application can be used like this:

```
otbcli_EdgeExtraction -in InputImage
 -channel 1
 -filter sobel
 -out OutputImage
```

or like this if filter==touzi:

```
otbcli_EdgeExtraction -in InputImage
 -channel 1
 -filter touzi
 -filter.touzi.xradius 2
 -filter.touzi.yradius 2
 -out OutputImage
```

## 5.6.3 Radiometric indices extraction

This application computes radiometric indices using the channels of the input image. The output is a multi band image into which each channel is one of the selected indices.

The *RadiometricIndices* application has the following input parameters:

--in the input image to compute the features on

--out the output image containing the radiometric indices

--channels .blue the Blue channel index in the input image (default value is 1)

--channels .green the Green channel index in the input image (default value is 1)

--channels .red the Red channel index in the input image (default value is 1)

**--channels.nir** the Near Infrared channel index in the input image (default value is 1)

**--channels.mir** the Mid-Infrared channel index in the input image (default value is 1)

**--list** the list of available radiometric indices (default value is Vegetation:NDVI)

The available radiometric indices to be listed into -list with their relevant channels in brackets are:

Vegetation:NDVI - Normalized difference vegetation index (Red, NIR)  
 Vegetation:TNDVI - Transformed normalized difference vegetation index (Red, NIR)  
 Vegetation:RVI - Ratio vegetation index (Red, NIR)  
 Vegetation:SAVI - Soil adjusted vegetation index (Red, NIR)  
 Vegetation:TSAVI - Transformed soil adjusted vegetation index (Red, NIR)  
 Vegetation:MSAVI - Modified soil adjusted vegetation index (Red, NIR)  
 Vegetation:MSAVI2 - Modified soil adjusted vegetation index 2 (Red, NIR)  
 Vegetation:GEMI - Global environment monitoring index (Red, NIR)  
 Vegetation:IPVI - Infrared percentage vegetation index (Red, NIR)

Water:NDWI - Normalized difference water index (Gao 1996) (NIR, MIR)  
 Water:NDWI2 - Normalized difference water index (Mc Feeters 1996) (Green, NIR)  
 Water:MNDWI - Modified normalized difference water index (Xu 2006) (Green, MIR)  
 Water:NDPI - Normalized difference pond index (Lacaux et al.) (MIR, Green)  
 Water:NDTI - Normalized difference turbidity index (Lacaux et al.) (Red, Green)

Soil:RI - Redness index (Red, Green)  
 Soil:CI - Color index (Red, Green)  
 Soil:BI - Brightness index (Red, Green)  
 Soil:BI2 - Brightness index 2 (NIR, Red, Green)

The application can be used like this, which leads to an output image with 3 bands, respectively with the Vegetation:NDVI, Vegetation:RVI and Vegetation:IPVI radiometric indices in this exact order:

```
otbcli_RadiometricIndices -in InputImage
 -out OutputImage
 -channels.red 3
 -channels.green 2
 -channels.nir 4
 -list Vegetation:NDVI Vegetation:RVI
 Vegetation:IPVI
```

or like this, which leads to a single band output image with the Water:NDWI2 radiometric indice:

```
otbcli_RadiometricIndices -in InputImage
 -out OutputImage
 -channels.red 3
 -channels.green 2
 -channels.nir 4
 -list Water:NDWI2
```

## 5.6.4 Morphological features extraction

Morphological features can be highlighted by using image filters based on mathematical morphology either on binary or gray scale images.

### Binary morphological operations

This application performs binary morphological operations (dilation, erosion, opening and closing) on a mono band image with a specific structuring element (a ball or a cross) having one radius along X and another one along Y. NB:

the cross shaped structuring element has a fixed radius equal to 1 pixel in both X and Y directions.

The *BinaryMorphologicalOperation* application has the following input parameters:

- in the input image to be filtered
- channel the selected channel index in the input image to be processed (default value is 1)
- structype the choice of the structuring element type (ball/cross) (default value is ball)
- (-structype.ball.xradius) the ball structuring element X Radius (only if structype==ball) (default value is 5 pixels)
- (-structype.ball.yradius) the ball structuring element Y Radius (only if structype==ball) (default value is 5 pixels)
- filter the choice of the morphological operation (dilate/erode/opening/closing) (default value is dilate)
- (-filter.dilate.foreval) the foreground value for the dilation (idem for filter.erode/opening/closing) (default value is 1)
- (-filter.dilate.backval) the background value for the dilation (idem for filter.erode/opening/closing) (default value is 0)
- out the output filtered image

The application can be used like this:

```
otbcli_BinaryMorphologicalOperation -in InputImage
 -channel 1
 -structype ball
 -structype.ball.xradius 10
 -structype.ball.yradius 5
 -filter opening
 -filter.opening.foreval 1.0
 -filter.opening.backval 0.0
 -out OutputImage
```

## Gray scale morphological operations

This application performs morphological operations (dilation, erosion, opening and closing) on a gray scale mono band image with a specific structuring element (a ball or a cross) having one radius along X and another one along Y. NB: the cross shaped structuring element has a fixed radius equal to 1 pixel in both X and Y directions.

The *GrayScaleMorphologicalOperation* application has the following input parameters:

- in the input image to be filtered
- channel the selected channel index in the input image to be processed (default value is 1)
- structype the choice of the structuring element type (ball/cross) (default value is ball)
- (-structype.ball.xradius) the ball structuring element X Radius (only if structype==ball) (default value is 5 pixels)
- (-structype.ball.yradius) the ball structuring element Y Radius (only if structype==ball) (default value is 5 pixels)
- filter the choice of the morphological operation (dilate/erode/opening/closing) (default value is dilate)
- out the output filtered image

The application can be used like this:

```
otbcli_GrayScaleMorphologicalOperation -in InputImage
 -channel 1
 -structype ball
 -structype.ball.xradius 10
 -structype.ball.yradius 5
 -filter opening
 -out OutputImage
```

## 5.6.5 Textural features extraction

Texture features can be extracted with the help of image filters based on texture analysis methods like Haralick and structural feature set (SFS).

### Haralick texture features

This application computes Haralick, advanced and higher order texture features on every pixel in the selected channel of the input image. The output image is multi band with a feature per band.

The *HaralickTextureExtraction* application has the following input parameters:

```
--in the input image to compute the features on
--channel the selected channel index in the input image to be processed (default value is 1)
--texture the texture set selection [simple/advanced/higher] (default value is simple)
--parameters.min the input image minimum (default value is 0)
--parameters.max the input image maximum (default value is 255)
--parameters.xrad the X Radius of the processing neighborhood (default value is 2 pixels)
--parameters.yrad the Y Radius of the processing neighborhood (default value is 2 pixels)
--parameters.xoff the ΔX Offset for the co-occurrence computation (default value is 1 pixel)
--parameters.yoff the ΔY Offset for the co-occurrence computation (default value is 1 pixel)
--parameters.nbbin the number of bin per axis for histogram generation (default value is 8)
--out the output multi band image containing the selected texture features (one feature per band)
```

The available values for -texture with their relevant features are:

**--texture=simple**: In this case, 8 local Haralick textures features will be processed. The 8 output image channels are: Energy, Entropy, Correlation, Inverse Difference Moment, Inertia, Cluster Shade, Cluster Prominence and Haralick Correlation. They are provided in this exact order in the output image. Thus, this application computes the following Haralick textures over a neighborhood with user defined radius. To improve the speed of computation, a variant of Grey Level Co-occurrence Matrix (GLCM) called Grey Level Co-occurrence Indexed List (GLCIL) is used. Given below is the mathematical explanation on the computation of each textures. Here  $g(i, j)$  is the frequency of element in the GLCIL whose index is  $i, j$ . GLCIL stores a pair of frequency of two pixels taken from the given offset and the cell index  $(i, j)$  of the pixel in the neighborhood window.  $g(i, j)$  is the frequency value of the pair having index is  $i, j$ .

$$\text{“Energy”} = f_1 = \sum_{i,j} g(i, j)^2$$

$$\text{“Entropy”} = f_2 = - \sum_{i,j} g(i, j) \log_2 g(i, j), \text{ or } 0 \text{ if } g(i, j) = 0$$

$$\text{“Correlation”} = f_3 = \sum_{i,j} \frac{(i-\mu)(j-\mu)g(i,j)}{\sigma^2}$$

$$\text{“Inverse Difference Moment”} = f_4 = \sum_{i,j} \frac{1}{1+(i-j)^2} g(i, j)$$

$$\text{“Inertia”} = f_5 = \sum_{i,j} (i - j)^2 g(i, j) \text{ (sometimes called “contrast”)}$$

$$\text{“Cluster Shade”} = f_6 = \sum_{i,j} ((i - \mu) + (j - \mu))^3 g(i, j)$$

$$\text{“Cluster Prominence”} = f_7 = \sum_{i,j} ((i - \mu) + (j - \mu))^4 g(i, j)$$

“Haralick’s Correlation” =  $f_8 = \frac{\sum_{i,j} (i,j)g(i,j) - \mu_i^2}{\sigma_i^2}$  where  $\mu_t$  and  $\sigma_t$  are the mean and standard deviation of the row (or column, due to symmetry) sums. Above,  $\mu =$  (weighted pixel average) =  $\sum_{i,j} i \cdot g(i, j) = \sum_{i,j} j \cdot g(i, j)$  (due to matrix symmetry), and  $\sigma =$  (weighted pixel variance) =  $\sum_{i,j} (i - \mu)^2 \cdot g(i, j) = \sum_{i,j} (j - \mu)^2 \cdot g(i, j)$  (due to matrix symmetry).

**--texture=advanced:** In this case, 10 advanced texture features will be processed. The 10 output image channels are: Mean, Variance, Dissimilarity, Sum Average, Sum Variance, Sum Entropy, Difference of Entropies, Difference of Variances, IC1 and IC2. They are provided in this exact order in the output image. The textures are computed over a sliding window with user defined radius.

To improve the speed of computation, a variant of Grey Level Co-occurrence Matrix (GLCM) called Grey Level Co-occurrence Indexed List (GLCIL) is used. Given below is the mathematical explanation on the computation of each textures. Here  $g(i, j)$  is the frequency of element in the GLCIL whose index is  $i, j$ . GLCIL stores a pair of frequency of two pixels taken from the given offset and the cell index  $(i, j)$  of the pixel in the neighborhood window. (where each element in GLCIL is a pair of pixel index and it’s frequency,  $g(i, j)$  is the frequency value of the pair having index is  $i, j$ ).

$$\text{“Mean”} = \sum_{i,j} i g(i, j)$$

$$\text{“Sum of squares: Variance”} = f_4 = \sum_{i,j} (i - \mu)^2 g(i, j)$$

$$\text{“Dissimilarity”} = f_5 = \sum_{i,j} (i - j) g(i, j)^2$$

$$\text{“Sum average”} = f_6 = - \sum_i i g_{x+y}(i)$$

$$\text{“Sum Variance”} = f_7 = \sum_i (i - f_8)^2 g_{x+y}(i)$$

$$\text{“Sum Entropy”} = f_8 = - \sum_i g_{x+y}(i) \log(g_{x+y}(i))$$

$$\text{“Difference variance”} = f_{10} = \text{variance of } g_{x-y}(i)$$

$$\text{“Difference entropy”} = f_{11} = - \sum_i g_{x-y}(i) \log(g_{x-y}(i))$$

$$\text{“Information Measures of Correlation IC1”} = f_{12} = \frac{f_9 - HXY1}{H}$$

$$\text{“Information Measures of Correlation IC2”} = f_{13} = \sqrt{1 - \exp -2|HXY2 - f_9|}$$

Above,  $\mu =$  (weighted pixel average) =  $\sum_{i,j} i \cdot g(i, j) = \sum_{i,j} j \cdot g(i, j)$  (due to matrix symmetry), and

$$g_{x+y}(k) = \sum_i \sum_j g(i) \text{ where } i + j = k \text{ and } k = 2, 3, \dots, 2N_g \text{ and}$$

$$g_{x-y}(k) = \sum_i \sum_j g(i) \text{ where } i - j = k \text{ and } k = 0, 1, \dots, N_g - 1$$

**--texture=higher:** In this case, 11 local higher order statistics texture coefficients based on the grey level run-length matrix will be processed. The 11 output image channels are: Short Run Emphasis, Long Run Emphasis, Grey-Level Nonuniformity, Run Length Nonuniformity, Run Percentage, Low Grey-Level Run Emphasis, High Grey-Level Run Emphasis, Short Run Low Grey-Level Emphasis, Short Run High Grey-Level Emphasis, Long Run Low Grey-Level Emphasis and Long Run High Grey-Level Emphasis. They are provided in this exact order in the output image. Thus, this application computes the following Haralick textures over a sliding window with user defined radius: (where  $p(i, j)$  is the element in cell  $i, j$  of a normalized Run Length Matrix,  $n_r$  is the total number of runs and  $n_p$  is the total number of pixels):

$$\text{“Short Run Emphasis”} = SRE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j)}{j^2}$$

$$\text{“Long Run Emphasis”} = LRE = \frac{1}{n_r} \sum_{i,j} p(i, j) * j^2$$

$$\text{“Grey-Level Nonuniformity”} = GLN = \frac{1}{n_r} \sum_i \left( \sum_j p(i, j) \right)^2$$

$$\text{“Run Length Nonuniformity”} = RLN = \frac{1}{n_r} \sum_j \left( \sum_i p(i, j) \right)^2$$

$$\text{“Run Percentage”} = RP = \frac{n_r}{n_p}$$

$$\text{“Low Grey-Level Run Emphasis”} = LGRE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j)}{i^2}$$

$$\text{“High Grey-Level Run Emphasis”} = HGRE = \frac{1}{n_r} \sum_{i,j} p(i, j) * i^2$$

$$\text{“Short Run Low Grey-Level Emphasis”} = SRLGE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j)}{i^2 j^2}$$

$$\text{“Short Run High Grey-Level Emphasis”} = SRHGE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j) * i^2}{j^2}$$

$$\text{“Long Run Low Grey-Level Emphasis”} = LRLGE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j) * j^2}{i^2}$$

$$\text{“Long Run High Grey-Level Emphasis”} = LRHGE = \frac{1}{n_r} \sum_{i,j} p(i, j) i^2 j^2$$

The application can be used like this:

```
otbcli_HaralickTextureExtraction -in Input Image
 -channel 1
 -texture simple
 -parameters.min 0
 -parameters.max 255
 -out Output Image
```

## SFS texture extraction

This application computes Structural Feature Set textures on every pixel in the selected channel of the input image. The output image is multi band with a feature per band. The 6 output texture features are SFS’Length, SFS’Width, SFS’PSI, SFS’W-Mean, SFS’Ratio and SFS’SD. They are provided in this exact order in the output image.

It is based on line direction estimation and described in the following publication. Please refer to Xin Huang, Liangpei Zhang and Pingxiang Li publication, Classification and Extraction of Spatial Features in Urban Areas Using High-Resolution Multispectral Imagery. IEEE Geoscience and Remote Sensing Letters, vol. 4, n. 2, 2007, pp 260-264.

The texture is computed for each pixel using its neighborhood. User can set the spatial threshold that is the max line length, the spectral threshold that is the max difference authorized between a pixel of the line and the center pixel of the current neighborhood. The adjustment constant alpha and the ratio Maximum Consideration Number, which describes the shape contour around the central pixel, are used to compute the *w - mean* value.

The *SFSTextureExtraction* application has the following input parameters:

```
--in the input image to compute the features on
--channel the selected channel index in the input image to be processed (default value is 1)
--parameters.spethre the spectral threshold (default value is 50)
--parameters.spthre the spatial threshold (default value is 100 pixels)
--parameters.nbdir the number of directions (default value is 20)
--parameters.alpha the alpha value (default value is 1)
--parameters.maxcons the ratio Maximum Consideration Number (default value is 5)
--out the output multi band image containing the selected texture features (one feature per band)
```

The application can be used like this:

```
otbcli_SFSTextureExtraction -in InputImage
 -channel 1
 -out OutputImage
```

## 5.7 Stereoscopic reconstruction from VHR optical images pair

This section describes how to convert pair of stereo images into elevation information.

The standard problem of terrain reconstruction with available **OTB Applications** contains the following steps:

- Estimation of displacements grids for epipolar geometry transformation
- Epipolar resampling of the image pair using those grids
- Dense disparity map estimation
- Projection of the disparities on a Digital Surface Model (DSM)

Let's go to the third dimension!

### 5.7.1 Estimate epipolar geometry transformation

The aim of this application is to generate resampled grids to transform images in epipolar geometry. **Epipolar geometry** is the geometry of stereo vision. The operation of stereo rectification determines transformations to apply to each image such that pairs of conjugate epipolar lines become collinear, parallel to one of the image axes and aligned. In this geometry, the objects present on a given row of the left image are also located on the same line in the right image.

Applying this transformation reduces the problem of elevation (or stereo correspondences determination) to a 1-D problem. We have two images `image1` and `image2` over the same area (the stereo pair) and we assume that we know the localization functions (forward and inverse) associated for each of these images.

The forward function allows to go from the image referential to the geographic referential:

$$(long, lat) = f_{image1}^{forward}(i, j, h)$$

where  $h$  is the elevation hypothesis,  $(i, j)$  are the pixel coordinates in `image1` and  $(long, lat)$  are geographic coordinates. As you can imagine, the inverse function allows to go from geographic coordinates to the image geometry.

For the second image, in that case, the expression of the inverse function is:

$$(long, lat, h) = f_{image2}^{inverse}(i, j)$$

Using jointly the forward and inverse functions from the image pair, we can construct a co-localization function  $f_{image1 \rightarrow image2}$  between the position of a pixel in the first and its position in the second one:

$$(i_{image2}, j_{image2}) = f_{image1 \rightarrow image2}(i_{image1}, j_{image1}, h)$$

The expression of this function is:

$$f_{image1 \rightarrow image2}(i_{image1}, j_{image1}, h) = f_{image2}^{inverse} f_{image1}^{forward}((i_{image1}, j_{image1}), h)$$

The expression is not really important, what we need to understand is that if we are able to determine for a given pixel in image1 the corresponding pixel in image2, as we know the expression of the co-localization function between both images, we can determine by identification the information about the elevation (variable  $h$  in the equation)!

We now have the mathematical basis to understand how 3-D information can be extracted by examination of the relative positions of objects in the two 2-D epipolar images.

The construction of the two epipolar grids is a little bit more complicated in the case of VHR optical images. That is because most of passive remote sensing from space use a push-broom sensor, which corresponds to a line of sensors arranged perpendicularly to the flight direction of the spacecraft. This acquisition configuration implies a slightly different strategy for stereo-rectification (see [here](#)).

We will now explain how to use the *StereoRectificationGridGenerator* application to produce two images which are **deformation grids** to resample the two images in epipolar geometry.

```
otbcli_StereoRectificationGridGenerator -io.inleft image1.tif
 -io.inright image2.tif
 -epi.elevation.avg.value 50
 -epi.step 5
 -io.outimage1 outimage1_grid.tif
 -io.outright outimage1_grid.tif
```

The application estimates the displacement to apply to each pixel in both input images to obtain epipolar geometry. The application accept a 'step' parameter to estimate displacements on a coarser grid. Here we estimate the displacements every 10 pixels. This is because in most cases with a pair of VHR and a small angle between the two images, this grid is very smooth. Moreover, the implementation is not *streamable* and uses potentially a lot of memory. Therefore it is generally a good idea to estimate the displacement grid at a coarser resolution.

The application outputs the size of the output images in epipolar geometry. **Note these values**, we will use them in the next step to resample the two images in epipolar geometry.

In our case, we have:

```
Output parameters value:
epi.rectsizeX: 4462
epi.rectsizeY: 2951
epi.baseline: 0.2094
```

The `epi.baseline` parameter provides the mean value (in *pixels.meters<sup>-1</sup>*) of the baseline to sensor altitude ratio. It can be used to convert disparities to physical elevation, since a disparity of this value will correspond to an elevation offset of one meter with respect to the mean elevation.

we can now move forward to the resampling in epipolar geometry.

## 5.7.2 Resample images in epipolar geometry

The former application generates two grids of displacements. The *GridBasedImageResampling* allows to resample the two input images in the epipolar geometry using these grids. These grids are intermediary results not really useful on their own in most cases. This second step *only* consists in applying the transformation and resample both images. This application can obviously be used in lot of other contexts.

The two commands to generate epipolar images are:

```
otbcli_GridBasedImageResampling -io.in image1.tif
 -io.out image1_epipolar.tif
 -grid.in outimage1_grid.tif
 -out.sizeX 4462
 -out.sizeY 2951
```

```
otbcli_GridBasedImageResampling -io.in image2.tif
 -io.out image2_epipolar.tif
 -grid.in outimage2_grid.tif
 -out.size_x 4462
 -out.size_y 2951
```

As you can see, we set *size\_x* and *size\_y* parameters using output values given by the *StereoRectificationGridGenerator* application to set the size of the output epipolar images.



Figure 1: Extract of resample image1 and image2 in epipolar geometry over Pyramids of Cheops. ©CNES 2012

We obtain two images in epipolar geometry, as shown in *Figure 1*. Note that the application allows to resample only a part of the image using the *-out.ulx* and *-out.uly* parameters.

### 5.7.3 Disparity estimation: Block matching along epipolar lines

Finally, we can begin the stereo correspondences lookup process!

Things are becoming a little bit more complex but do not worry. First, we will describe the power of the *BlockMatching* application.

The resampling of our images in epipolar geometry allows us to constrain the search along a 1-dimensional line as opposed to both dimensions, but what is even more important is that the disparities along the lines, i.e. the offset along the lines measured by the block-matching process can be directly linked to the local elevation

An almost complete spectrum of stereo correspondence algorithms has been published and it is still augmented at a significant rate! See for example . The **Orfeo Toolbox** implements different strategies for block matching:



- Sum of Square Distances block-matching (SSD)
- Normalized Cross-Correlation (NCC)
- Lp pseudo-norm (LP)

Another important parameter (mandatory in the application!) is the range of disparities. In theory, the block matching can perform a blind exploration and search for an infinite range of disparities between the stereo pair. We need now to evaluate a range of disparities where the block matching will be performed (in the general case from the deepest point on Earth, [the Challenger Deep](#) . to the Everest summit!)

We deliberately exaggerated but you can imagine that without a smaller range the block matching algorithm can take a lot of time. That is why these parameters are mandatory for the application and as a consequence we need to estimate them manually. This is pretty simple using the two epipolar images.

In our case, we take one point on a *flat* area. The image coordinate in  $image_1$  is [1970, 1525] and in  $image_2$  is [1970, 1526]. We then select a second point on a higher region (in our case a point near the top of the Pyramid of Cheops!). The image coordinate of this pixel in  $image_1$  is [1661, 1299] and in  $image_2$  is [1633, 1300]. So you see for the horizontal exploration, we must set the minimum value lower than  $-30$  (the convention for the sign of the disparity range is from image1 to image2).

Note that this estimation can be simplified using an external DEM in the *StereoRectificationGridGenerator* application. Regarding the vertical disparity, in the first step we said that we reduced the problem of 3-D extraction to a 1-D problem, but this is not completely true in general cases. There might be small disparities in the vertical direction which are due to parallax errors (i.e. epipolar lines exhibit a small shift in the vertical direction, around 1 pixel). In fact, the exploration is typically smaller along the vertical direction of disparities than along the horizontal one. You can also estimate them on the epipolar pair (in our case we use a range of  $-1$  to  $1$ ).

One more time, take care of the sign of this minimum and this maximum for disparities (always from image1 to

image2).

The command line for the *BlockMatching* application is :

```
otbcli_BlockMatching -io.inleft image1_epipolar.tif
 -io.inright image2_epipolar.tif
 -io.out disparity_map_ncc.tif
 -bm.minhd -45
 -bm.maxhd 5
 -bm.minvd 1
 -bm.maxvd 1
 -mask.inleft image1_epipolar_mask.tif
 -mask.inright image2_epipolar_mask.tif
 -io.outmetric 1
 -bm.metric ncc
 -bm.subpixel dichotomy
 -bm.medianfilter.radius 5
 -bm.medianfilter.incoherence 2.0
```

The application creates by default a two bands image : the horizontal and vertical disparities.

The *BlockMatching* application gives access to a lot of other powerful functionalities to improve the quality of the output disparity map.

Here are a few of these functionalities:

- `-io.outmetric`: if the optimal metric values image is activated, it will be concatenated to the output image (which will then have three bands: horizontal disparity, vertical disparity and metric value)
- `-bm.subpixel`: Perform sub-pixel estimation of disparities
- `-mask.inleft` and `-mask.inright`: you can specify a no-data value which will discard pixels with this value (for example the epipolar geometry can generate large part of images with black pixels) This mask can be easily generated using the *BandMath* application:

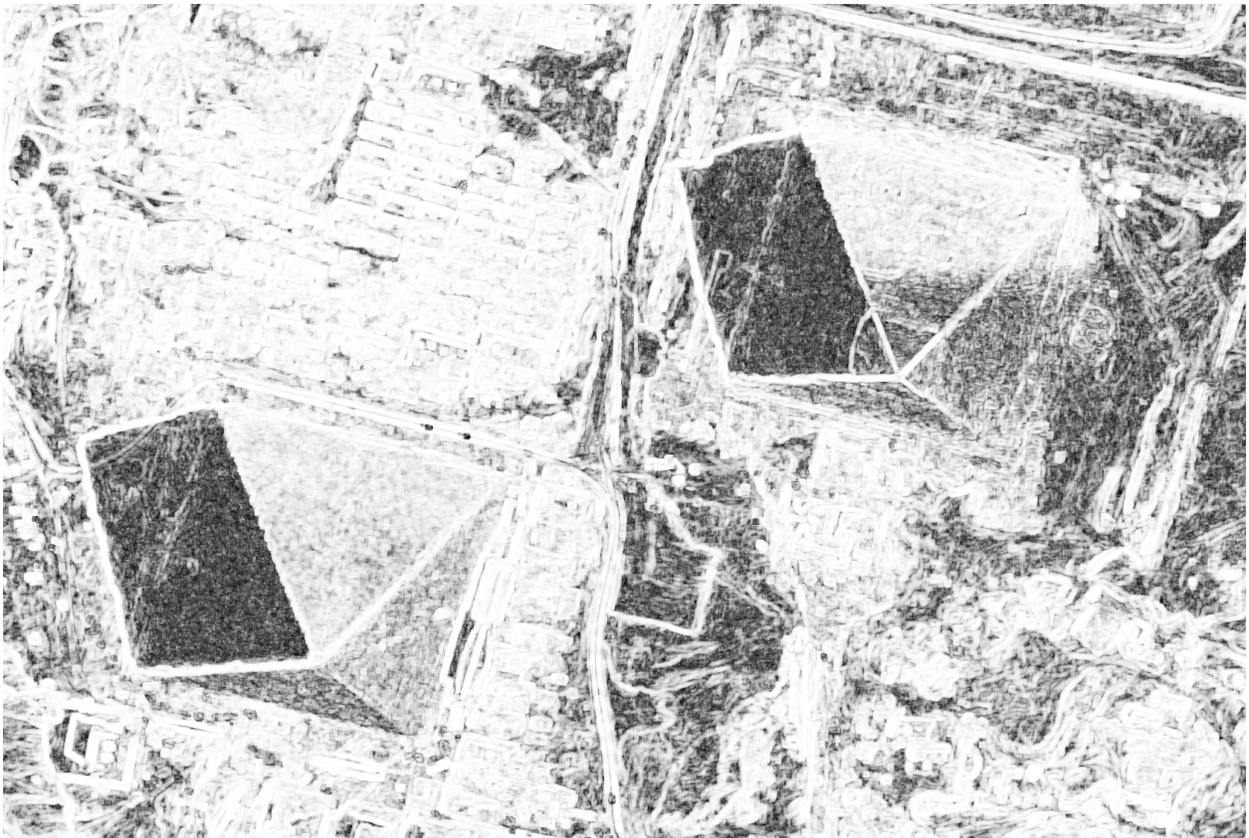
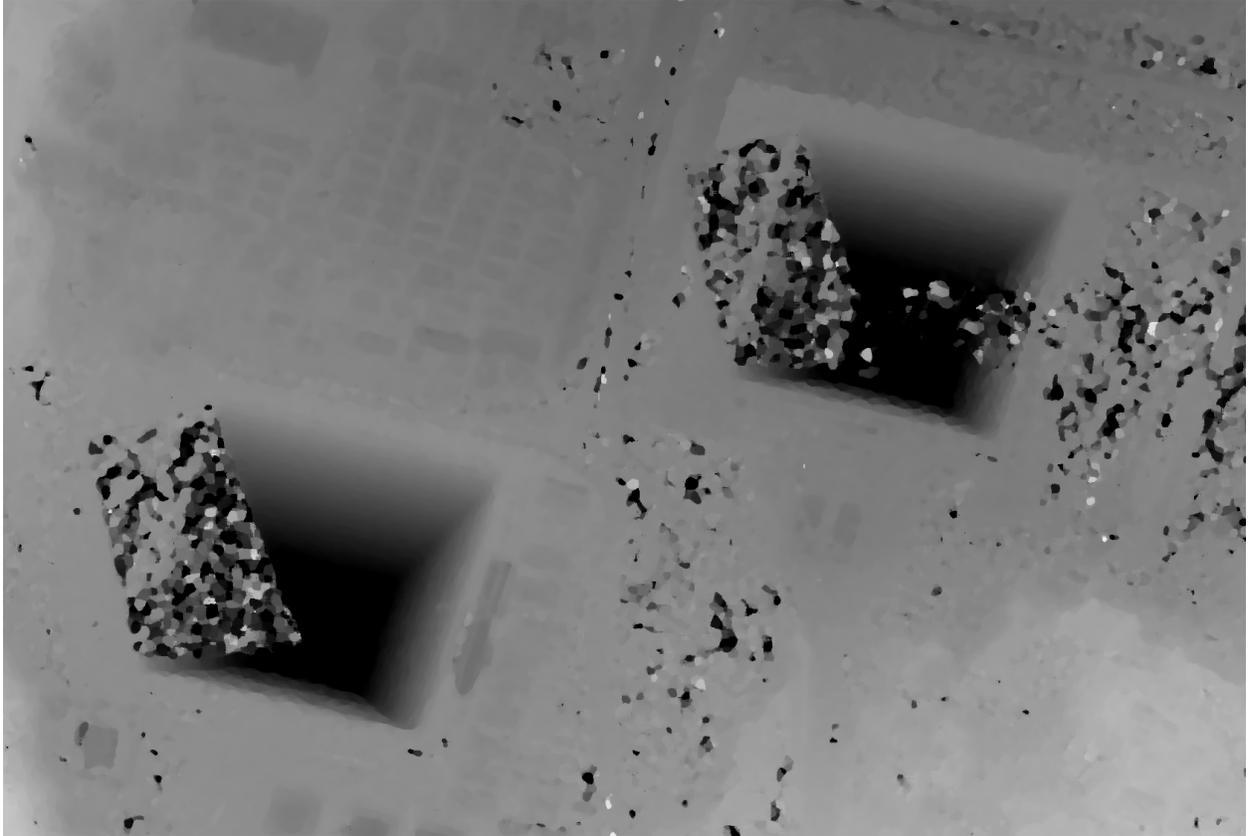
```
otbcli_BandMath -il image1_epipolar.tif
 -out image1_epipolar_mask.tif
 -exp "if(im1b1<=0,0,255) "
```

```
otbcli_BandMath -il image2_epipolar.tif
 -out image2_epipolar_mask.tif
 -exp "if(im1b1<=0,0,255) "
```

- `-mask.variancet` : The block matching algorithm has difficulties to find matches on uniform areas. We can use the variance threshold to discard those regions and speed-up computation time.
- `-bm.medianfilter.radius 5` and `-bm.medianfilter.incoherence 2.0`: Applies a median filter to the disparity map. The median filter belongs to the family of nonlinear filters. It is used to smooth an image without being biased by outliers or shot noise. The radius corresponds to the neighbourhood where the median value is computed. A detection of incoherence between the input disparity map and the median-filtered one is performed (a pixel corresponds to an incoherence if the absolute value of the difference between the pixel value in the disparity map and in the median image is higher than the incoherence threshold, whose default value is 1). Both parameters must be defined in the application to activate the filter.

Of course all these parameters can be combined to improve the disparity map.

Figure 2: Horizontal disparity and optimal metric map



5.7. Stereoscopic reconstruction from VHR optical images pair

## 5.7.4 From disparity to Digital Surface Model

Using the previous application, we evaluated disparities between images. The next (and last!) step is now to transform the disparity map into an elevation information to produce an elevation map. It uses as input the disparity maps (horizontal and vertical) to produce a Digital Surface Model (DSM) with a regular sampling. The elevation values is computed from the triangulation of the “left-right” pairs of matched pixels. When several elevations are available on a DSM cell, the highest one is kept.

First, an important point is that it is often a good idea to rework the disparity map given by the *BlockMatching* application to only keep relevant disparities. For this purpose, we can use the output optimal metric image and filter disparities with respect to this value.

For example, if we used Normalized Cross-Correlation (NCC), we can keep only disparities where optimal metric value is superior to 0.9. Disparities below this value can be consider as inaccurate and will not be used to compute elevation information (the *-io.mask* parameter can be used for this purpose).

This filtering can be easily done with **OTB Applications** .

We first use the *BandMath* application to filter disparities according to their optimal metric value:

```
otbcli_BandMath -il disparity_map_ncc.tif
 -out thres_hdisparity.tif uint8
 -exp "if(im1b3>0.9,255,0) "
```

Then, we concatenate thresholded disparities using the *ConcatenateImages* :

```
otbcli_ConcatenateImages -il thres_hdisparity.tif thres_vdisparity.tif
 -out thres_hvdisparity.tif
```

Now, we can use the *DisparityMapToElevationMap* application to compute the elevation map from the filtered disparity maps.

```
otbcli_DisparityMapToElevationMap -io.in disparity_map_ncc.tif
 -io.left image1.tif
 -io.right image2.tif
 -io.lgrid outimage1_pyramid.tif
 -io.rgrid outimage2_pyramid.tif
 -io.mask thres_hdisparity.tif
 -io.out disparity_map_ssd_to_elevation.tif
 -hmin 10
 -hmax 400
 -elev.default 50
```

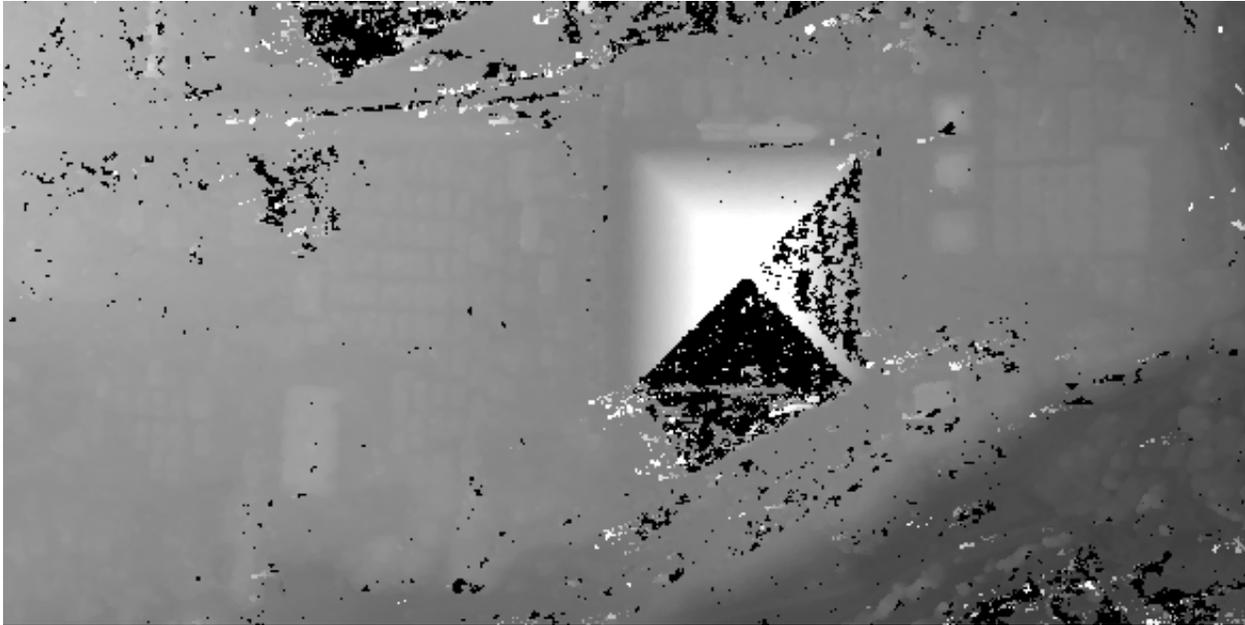
It produces the elevation map projected in WGS84 (EPSG code:4326) over the ground area covered by the stereo pair. Pixels values are expressed in meters.

Figure 3: Extract of the elevation map over Pyramids of Cheops.

This is it *Figure 3* shows the output DEM from the Cheops pair.

## 5.7.5 One application to rule them all in multi stereo framework scheme

An application has been added to fuse one or multiple stereo reconstruction(s) using all in one approach : *StereoFramework* . It computes the DSM from one or several stereo pair. First of all the user have to choose his input data and defines stereo couples using *-input.co* string parameter. This parameter use the following formatting convention “ *index<sub>0</sub> index<sub>1</sub>, index<sub>2</sub> index<sub>3</sub>, ...*”, which will create a first couple with image *index<sub>0</sub>* and *index<sub>1</sub>*, a second with image *index<sub>1</sub>* and *index<sub>2</sub>*, and so on. If left blank images are processed by pairs (which is equivalent as using “0 1,2 3,4 5” ...). In addition to the usual elevation and projection parameters, main parameters have been split in groups detailed below:



**Output :** output parameters : DSM resolution, NoData value, Cell Fusion method,

- : output projection map selection.
- : Spatial Sampling Distance of the output DSM in meters
- : DSM empty cells are filled with this float value (-32768 by default)
- : Choice of fusion strategy in each DSM cell (max, min, mean, acc)
- : Output DSM
- : Output DSM extent choice

**Stereorect :** Direct and inverse stereorectification grid subsampling parameters

- : Step of the direct deformation grid (in pixels)
- : Sub-sampling of the inverse epipolar grid

**BM :** Block Matching parameters.

- : Block-matching metric choice (robust SSD, SSD, NCC, Lp Norm)
- : Radius of blocks for matching filter (in pixels, 2 by default)
- : Minimum altitude below the selected elevation source (in meters, -20.0 by default)
- : Maximum altitude above the selected elevation source (in meters, 20.0 by default)

**Postproc :** Post-Processing parameters

- : use bijection consistency. Right to Left correlation is computed to validate Left to Right disparities. If bijection is not found pixel is rejected
- : use median disparities filtering (disabled by default)
- : use block matching metric output to discard pixels with low correlation value (disabled by default, float value)");

**Mask :** Compute optional intermediate masks.

- : Mask for left input image (must have the same size for all couples)

- : Mask for right input image (must have the same size for all couples)
- : This parameter allows to discard pixels whose local variance is too small. The size of the neighborhood is given by the radius parameter. (disabled by default)

### 5.7.6 Stereo reconstruction good practices

The parameters and are used inside the application to derive the minimum and maximum horizontal disparity exploration, so they have a critical impact on computation time. It is advised to choose an elevation source that is not too far from the DSM you want to produce (for instance, an SRTM elevation model). Therefore, the altitude from your elevation source will be already taken into account in the epipolar geometry and the disparities will reveal the elevation offsets (such as buildings). It allows you to use a smaller exploration range along the elevation axis, causing a smaller exploration along horizontal disparities and faster computation.

and have also a deep impact in time consumption, thus they have to be carefully chosen in case of large image processing.

To reduce time consumption it would be useful to crop all sensor images to the same extent. The easiest way to do that is to choose an image as reference, and then apply *ExtractROI* application on the other sensor images using the fit mode option.

### 5.7.7 Algorithm outline

The following algorithms are used in the application: For each sensor pair

- Compute the epipolar deformation grids from the stereo pair (direct and inverse)
- Resample into epipolar geometry with BCO interpolator
- Create masks for each epipolar image : remove black borders and resample input masks
- Compute horizontal disparities with a block matching algorithm
- Refine Disparities to sub-pixel precision with a dichotomy algorithm
- Apply an optional Median filter
- Filter disparities based on the correlation score (optional) and exploration bounds
- Translate disparities in sensor geometry
- Convert disparity map to 3D map

Then fuse all 3D maps to produce DSM with desired geographic or cartographic projection and parametrizable extent.

## 5.8 BandMathImageFilterX (based on muParserX)

This section describes how to use the BandMathImageFilterX.

### 5.8.1 Fundamentals: headers, declaration and instantiation

A simple example is given below:

```

#include "otbBandMathImageFilterX.h"
#include "otbVectorImage.h"

int otbBandMathImageFilterXNew(int itkNotUsed(argc), char* itkNotUsed(argv) [])
{
 typedef double PixelType;
 typedef otb::VectorImage<PixelType, 2> ImageType;
 typedef otb::BandMathImageFilterX<ImageType> FilterType;

 FilterType::Pointer filter = FilterType::New();

 return EXIT_SUCCESS;
}

```

As we can see, the new band math filter works with the class `otb::VectorImage`.

## 5.8.2 Syntax : first elements

The default prefix name for variables related to the  $i$ th input is  $im(i+1)$  (note the indexing from 1 to N, for N inputs). The user has the possibility to change this default behaviour by setting its own prefix.

```

// All variables related to image1 (input 0) will have the prefix im1
filter->SetNthInput(0, image1);

```

```

// All variables related to image2 (input 1) will have the prefix toulouse
filter->SetNthInput(1, image2, "toulouse");

```

```

// All variables related to anotherImage (input 2) will have the prefix im3
filter->SetNthInput(2, anotherImage);

```

In this document, we will keep the default convention. Following list summaries the available variables for input #0 (and so on for every input).

Variables and their descriptions:

Variables	Description	Type
<code>im1</code>	a pixel from first input, made of n components/bands (first image is indexed by 1)	Vector
<code>im1bj</code>	jth component of a pixel from first input (first band is indexed by 1)	Scalar
<code>im1bjNkxp</code>	a neighbourhood ("N") of pixels of the jth component from first input, of size kxp	Matrix
<code>im1bjMini</code>	global statistic : minimum of the jth band from first input	Scalar
<code>im1bjMaxi</code>	global statistic : maximum of the jth band from first input	Scalar
<code>im1bjMean</code>	global statistic : mean of the jth band from first input	Scalar
<code>im1bjSum</code>	global statistic : sum of the jth band from first input	Scalar
<code>im1bjVar</code>	global statistic : variance of the jth band from first input	Scalar
<code>im1PhyX</code> and <code>im1PhyY</code>	spacing of first input in X and Y directions	Scalar

[variables]

Moreover, we also have the generic variables `idxX` and `idxY` that represent the indices of the current pixel (scalars).

Note that the use of a global statistics will automatically make the filter (or the application) request the largest possible regions from the concerned input images, without user intervention.

For instance, the following formula (addition of two pixels)

$$im1 + im2$$

[firstequation]

is correct only if the two first inputs have the same number of bands. In addition, the following formula is not consistent even if *im1* represents a pixel of an image made of only one band:

$$im1 + 1$$

A scalar can't be added to a vector. The right formula is instead (one can notice the way that *muParserX* allows to define vectors on the fly):

$$im1 + \{1\}$$

or

$$im1 + \{1, 1, 1, \dots, 1\}$$

if *im1* is made of *n* components.

On the other hand, the variable *im1b1* for instance is represented as a scalar; so we have the following different possibilities:

Correct / incorrect expressions:

Expression	Status
<i>im1b1</i> + 1	correct
{ <i>im1b1</i> } + {1}	correct
<i>im1b1</i> + {1}	incorrect
{ <i>im1b1</i> } + 1	incorrect
<i>im1</i> + { <i>im2b1, im2b2</i> }	correct if <i>im1</i> represents a pixel of two components (equivalent to <i>im1</i> + <i>im2</i> )

Similar remarks can be made for the multiplication/division; for instance, the following formula is incorrect:

$$\{im2b1, im2b2\} * \{1, 2\}$$

whereas this one is correct:

$$\{im2b1, im2b2\} * \{1, 2\}'$$

or in more simple terms (and only if *im2* contains two components):

$$im2 * \{1, 2\}'$$

Concerning division, this operation is not originally defined between two vectors (see next section “New operators and functions” -[ssec:operators]-).

Now, let's go back to the first formula: this one specifies the addition of two images band to band. With *muParserX* lib, we can now define such operation with only one formula, instead of many formulas (as many as the number of bands). We call this new functionality the **batch mode**, which directly arises from the introduction of vectors within *muParserX* framework.

Finally, let's say a few words about neighbourhood variables. These variables are defined for each particular input, and for each particular band. The two last numbers, *kxp*, indicate the size of the neighbourhood. All neighbourhoods are centred: this means that *k* and *p* can only be odd numbers. Moreover, *k* represents the dimension in the *x* direction

(number of columns), and  $p$  the dimension in the  $y$  direction (number of rows). For instance, `im1b3N3x5` represents the following neighbourhood:

.	.	.
.	.	.
.	.	.
.	.	.
.	.	.

[correctness]

Fundamentally, a neighbourhood is represented as a matrix inside the `muParserX` framework; so the remark about mathematically well-defined formulas still stands.

### 5.8.3 New operators and functions

New operators and functions have been implemented within `BandMathImageFilterX`. These ones can be divided into two categories.

- adaptation of existing operators/functions, that were not originally defined for vectors and matrices (for instance `cos`, `sin`, ...). These new operators/ functions keep the original names to which we add the prefix “v” for vector (`vcos`, `vsin`, ...).
- truly new operators/functions.

Concerning the last category, here is a list of implemented operators or functions (they are all implemented in `otb-ParserXPlugins.h/cxx` files -OTB/Code/Common-):

**Operators `div` and `dv`** The first operator allows the definition of an element-wise division of two vectors (and even matrices), provided that they have the same dimensions. The second one allows the definition of the division of a vector/matrix by a scalar (components are divided by the same unique value). For instance:

$$im1 \text{ div } im2$$

$$im1 \text{ dv } 2.0$$

**Operators `mult` and `mlt`** These operators are the duals of the previous ones. For instance:

$$im1 \text{ mult } im2$$

$$im1 \text{ mlt } 2.0$$

Note that the operator `'**'` could have been used instead of `'pw'` one. But `'pw'` is a little bit more permissive, and can tolerate one-dimensional vector as right element.

**Operators `pow` and `pw`** The first operator allows the definition of an element-wise exponentiation of two vectors (and even matrices), provided that they have the same dimensions. The second one allows the definition of the division of a vector/matrix by a scalar (components are exponentiated by the same unique value). For instance:

$$im1 \text{ pow } im2$$

*im1 pw 2.0*

**Function bands** This function allows to select specific bands from an image, and/or to rearrange them in a new vector; for instance:

*bands(im1, {1, 2, 1, 1})*

produces a vector of 4 components made of band 1, band 2, band 1 and band 1 values from the first input. Note that curly brackets must be used in order to select the desired band indices.

**\*\* Function dotpr \*\*** This function allows the dot product between two vectors or matrices (actually in our case, a kernel and a neighbourhood of pixels):

$$\sum_{(i,j)} m_1(i, j) * m_2(i, j)$$

For instance:

*dotpr(kernel1, im1b1N3x5)*

is correct provided that kernel1 and im1b1N3x5 have the same dimensions. The function can take as many neighbourhoods as needed in inputs.

**Function mean** This function allows to compute the mean value of a given vector or neighborhood (the function can take as many inputs as needed; one mean value is computed per input). For instance:

*mean(im1b1N3x3, im1b2N3x3, im1b3N3x3, im1b4N3x3)*

Note: a limitation coming from muparserX itself makes impossible to pass all those neighborhoods with a unique variable.

**Function var** This function allows to compute the variance of a given vector or neighborhood (the function can take as many inputs as needed; one var value is computed per input). For instance:

*var(im1b1N3x3)*

**Function median** This function allows to compute the median value of a given vector or neighborhood (the function can take as many inputs as needed; one median value is computed per input). For instance:

*median(im1b1N3x3)*

**Function corr** This function allows to compute the correlation between two vectors or matrices of the same dimensions (the function takes two inputs). For instance:

*corr(im1b1N3x3, im1b2N3x3)*

**Function maj** This function allows to compute the most represented element within a vector or a matrix (the function can take as many inputs as needed; one maj element value is computed per input). For instance:

*maj(im1b1N3x3, im1b2N3x3)*

**Function vmin and vmax** These functions allow to compute the min or max value of a given vector or neighborhood (only one input). For instance:

$$(vmax(im3b1N3x5) + vmin(im3b1N3x5)) \text{ div } \{2.0\}$$

**Function cat** This function allows to concatenate the results of several expressions into a multidimensional vector, whatever their respective dimensions (the function can take as many inputs as needed). For instance:

$$cat(im3b1, vmin(im3b1N3x5), median(im3b1N3x5), vmax(im3b1N3x5))$$

Note: the user should prefer the use of semi-colons (;) when setting expressions, instead of directly use this function. The filter or the application will call the function 'cat' automatically. For instance:

$$filter -> SetExpression("im3b1;vmin(im3b1N3x5);median(im3b1N3x5);vmax(im3b1N3x5)");$$

Please, also refer to the next section "Application Programming Interface" ([ssec:API]).

**Function ndvi** This function implements the classical normalized difference vegetation index; it takes two inputs. For instance:

$$ndvi(im1b1, im1b4)$$

First argument is related to the visible red band, and the second one to the near-infrareds band.

The table below summarises the different functions and operators.

Functions and operators summary:

Variables	Remark
ndvi	two inputs
bands	two inputs; length of second vector input gives the dimension of the output
dotptr	many inputs
cat	many inputs
mean	many inputs
var	many inputs
median	many inputs
maj	many inputs
corr	two inputs
div and dv	operators
mult and mlt	operators
pow and pw	operators
vnorm	adapation of an existing function to vectors : one input
vabs	adapation of an existing function to vectors : one input
vmin	adapation of an existing function to vectors : one input
vmax	adapation of an existing function to vectors : one input
vcos	adapation of an existing function to vectors : one input
vsin	adapation of an existing function to vectors : one input
vtan	adapation of an existing function to vectors : one input
vtanh	adapation of an existing function to vectors : one input
vsinh	adapation of an existing function to vectors : one input
vcosh	adapation of an existing function to vectors : one input
vlog	adapation of an existing function to vectors : one input
vlog10	adapation of an existing function to vectors : one input
vexp	adapation of an existing function to vectors : one input
vsqrt	adapation of an existing function to vectors : one input

[variables]

## 5.8.4 Application Programming Interface (API)

In this section, we make some comments about the public member functions of the new band math filter.

```
/** Set the nth filter input with or without a specified associated variable name */
void SetNthInput(unsigned int idx, const ImageType * image);
void SetNthInput(unsigned int idx, const ImageType * image, const std::string& varName);

/** Return a pointer on the nth filter input */
ImageType * GetNthInput(unsigned int idx);
```

Refer to the section “Syntax : first elements” ([ssec:syntax]) where the two first functions have already been commented. The function `GetNthInput` is quite clear to understand.

```
/** Set an expression to be parsed */
void SetExpression(const std::string& expression);
```

Each time the function `SetExpression` is called, a new expression is pushed inside the filter. **There are as many outputs as there are expressions. The dimensions of the outputs (number of bands) are totally dependent on the dimensions of the related expressions (see also last remark of the section “Syntax : first element” -[ssec:syntax]-).** Thus, the filter always performs a pre-evaluation of each expression, in order to guess how to allocate the outputs.

The concatenation of the results of many expressions (whose results can have different dimensions) into one unique output is possible. For that purpose, semi-colons (“;”) are used as separating characters. For instance:

```
filter->SetExpression("im1 + im2; im1b1 * im2b1");
```

will produce a unique output (one expression) of many bands (actually, number of bands of  $im1 + 1$ ).

```
/** Return the nth expression to be parsed */
std::string GetExpression(int) const;
```

This function allows the user to get any expression by its ID number.

```
/** Set a matrix (or a vector) */
void SetMatrix(const std::string& name, const std::string& definition);
```

This function allows the user to set new vectors or matrices. This is particularly useful when the user wants to use the `dotpr` function (see previous section). First argument is related to the name of the variable, and the second one to the definition of the vector/matrix. The definition is done by a string, where first and last elements must be curly brackets (“{” and “}”). Different elements of a row are separated by commas (“,”), and different rows are separated by semi-colons (“;”). For instance:

```
filter->SetMatrix("kernel1", "{ 0.1 , 0.2 , 0.3 ; 0.4 , 0.5 , 0.6 ; \
0.7 , 0.8 , 0.9 ; 1.0 , 1.1 , 1.2 ; 1.3 , 1.4 , 1.5 }");
```

defines the `kernel1`, whose elements are given as follows:

<b>0,1</b>	<b>0,2</b>	<b>0,3</b>
0,4	0,5	0,6
0,7	0,8	0,9
1,0	1,1	1,2
1,3	1,4	1,5

Definition of `kernel1`.

[correctness]

```
/** Set a constant */
void SetConstant(const std::string& name, double value);
```

This function allows the user to set new constants.

```
/** Return the variable and constant names */
std::vector<std::string> GetVarNames() const;
```

This function allows the user to get the list of the variable and constant names, in the form of a `std::vector` of strings.

```
/** Import constants and expressions from a given filename */
void ImportContext(const std::string& filename);
```

This function allows the user to define new constants and/or expressions (context) by using a txt file with a specific syntax. For the definition of constants, the following pattern must be observed: `#type name value`. For instance:

```
#F expo 1.1 #M kernel1 { 0.1 , 0.2 , 0.3 ; 0.4 , 0.5 , 0.6 ; 0.7 , 0.8 , 0.9 ; 1 , 1.1 , 1.2 ; 1.3 , 1.4 , 1.5 }
```

As we can see, `#I/#F` allows the definition of an integer/float constant, whereas `#M` allows the definition of a vector/matrix. It is also possible to define expressions within the same txt file, with the pattern `#E expr`. For instance:

```
#F expo 1.1 #M kernel1 0.1 , 0.2 , 0.3 ; 0.4 , 0.5 , 0.6 ; 0.7 , 0.8 , 0.9 ; 1 , 1.1 , 1.2 ; 1.3 , 1.4 , 1.5 #E
dotpr(kernel1,im1b1N3x5)
```

```
/** Export constants and expressions to a given filename */
void ExportContext(const std::string& filename);
```

This function allows the user to export a txt file that saves its favorite constant or expression definitions. Such a file will be reusable by the `ImportContext` function (see above).

Please, also refer to the section dedicated to application.

## 5.9 Numpy processing in OTB Applications

Input and output images to any OTB application in the form of numpy array is now possible in OTB python wrapping. The python wrapping only exposes OTB `ApplicationEngine` module which allow to access existing C++ applications. Due to blissful nature of `ApplicationEngine`'s loading mechanism no specific wrapping is required for each application.

Numpy extension to Python wrapping allows data exchange to application as an array rather than a disk file. Of course, it is possible to load an image from file and then convert to numpy array or just provide a file as earlier via `Application.SetParameterString(...)`.

This bridge that completes numpy and OTB makes it easy to plug OTB into any image processing chain via python code that uses GIS/Image processing tools such as GDAL, GRASS GIS, OSSIM that can deal with numpy.

Below code reads an input image using python pillow (PIL) and convert it to numpy array. This numpy array is used an input to the application set `SetImageFromNumpyArray(...)` method. The application used in this example is `ExtractROI`. After extracting a small area the output image is taken as numpy array with `GetImageFromNumpyArray(...)` method

```
import sys
import os
import numpy as np
import otbApplication
from PIL import Image as PILImage

pilimage = PILImage.open('poupees.jpg')
```

```
npimage = np.asarray(pilimage)
imshow(pilimage)

ExtractROI = otbApplication.Registry.CreateApplication('ExtractROI')
ExtractROI.SetImageFromNumpyArray('in', npimage)
ExtractROI.SetParameterInt('startx', 140)
ExtractROI.SetParameterInt('starty', 120)
ExtractROI.SetParameterInt('sizex', 150)
ExtractROI.SetParameterInt('sizey', 150)
ExtractROI.Execute()

ExtractOutput = ExtractROI.GetImageAsNumpyArray('out')
output_pil_image = PILImage.fromarray(np.uint8(ExtractOutput))
imshow(output_pil_image)
```

# APPLICATIONS

## 6.1 Image Manipulation

### 6.1.1 Color Mapping

Maps an input label image to 8-bits RGB using look-up tables.

#### Detailed description

**This application allows one to map a label image to a 8-bits RGB image (in both ways) using different methods.**

-The custom method allows one to use a custom look-up table. The look-up table is loaded from a text file where each line describes an entry. The typical use of this method is to colorise a classification map. -The continuous method allows mapping a range of values in a scalar input image to a colored image using continuous look-up table, in order to enhance image interpretation. Several look-up tables can be chosen with different color ranges.

**-The optimal method computes an optimal look-up table. When processing a segmentation label image (label to color), the color**

- The support image method uses a color support image to associate an average color to each region.

#### Parameters

This section describes in details the parameters available for this application. Table <sup>1</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ColorMapping*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
op	Choices	Choices
op labeltcolor	<i>Choice</i>	Label to color
op colortolabel	<i>Choice</i>	Color to label
op.colortolabel.notfound	Int	Int
method	Choices	Choices
method custom	<i>Choice</i>	Color mapping with custom labeled look-up table

Continued on next page

<sup>1</sup> Table: Parameters table for Color Mapping.

Table 6.1 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
method continuous	<i>Choice</i>	Color mapping with continuous look-up table
method optimal	<i>Choice</i>	Compute an optimized look-up table
method image	<i>Choice</i>	Color mapping with look-up table calculated on support image
method.custom.lut	Input File name	Input File name
method.continuous.lut	Choices	Choices
method.continuous.lut red	<i>Choice</i>	Red
method.continuous.lut green	<i>Choice</i>	Green
method.continuous.lut blue	<i>Choice</i>	Blue
method.continuous.lut grey	<i>Choice</i>	Grey
method.continuous.lut hot	<i>Choice</i>	Hot
method.continuous.lut cool	<i>Choice</i>	Cool
method.continuous.lut spring	<i>Choice</i>	Spring
method.continuous.lut summer	<i>Choice</i>	Summer
method.continuous.lut autumn	<i>Choice</i>	Autumn
method.continuous.lut winter	<i>Choice</i>	Winter
method.continuous.lut copper	<i>Choice</i>	Copper
method.continuous.lut jet	<i>Choice</i>	Jet
method.continuous.lut hsv	<i>Choice</i>	HSV
method.continuous.lut overunder	<i>Choice</i>	OverUnder
method.continuous.lut relief	<i>Choice</i>	Relief
method.continuous.min	Float	Float
method.continuous.max	Float	Float
method.optimal.background	Int	Int
method.image.in	Input image	Input image
method.image.nodatavalue	Float	Float
method.image.low	Int	Int
method.image.up	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image filename.

**Output Image:** Output image filename.

**Operation:** Selection of the operation to execute (default is : label to color). Available choices are:

- **Label to color**
- **Color to label**
- **Not Found Label:** Label to use for unknown colors.

**Color mapping method:** Selection of color mapping methods and their parameters. Available choices are:

- **Color mapping with custom labeled look-up table:** Apply a user-defined look-up table to a labeled image. Look-up table is loaded from a text file.
- **Look-up table file:** An ASCII file containing the look-up table with one color per line (for instance the line ‘1 255 0 0’ means that all pixels with label 1 will be replaced by RGB color 255 0 0) Lines beginning with a # are ignored.
- **Color mapping with continuous look-up table:** Apply a continuous look-up table to a range of input values.
  - **Look-up tables:** Available look-up tables. Available choices are:

- **Red**
- **Green**
- **Blue**
- **Grey**
- **Hot**
- **Cool**
- **Spring**
- **Summer**
- **Autumn**
- **Winter**
- **Copper**
- **Jet**
- **HSV**
- **OverUnder**
- **Relief**
- **Mapping range lower value:** Set the lower input value of the mapping range.
- **Mapping range higher value:** Set the higher input value of the mapping range.
- **Compute an optimized look-up table:** [label to color] Compute an optimal look-up table such that neighboring labels in a segmentation are mapped to highly contrasted colors. [color to label] Searching all the colors present in the image to compute a continuous label list.
- **Background label:** Value of the background label.
- **Color mapping with look-up table calculated on support image**
- **Support Image:** Support image filename. For each label, the LUT is calculated from the mean pixel value in the support image, over the corresponding labeled areas. First of all, the support image is normalized with extrema rejection.
- **NoData value:** NoData value for each channel of the support image, which will not be handled in the LUT estimation. If NOT checked, ALL the pixel values of the support image will be handled in the LUT estimation.
- **lower quantile:** lower quantile for image normalization.
- **upper quantile:** upper quantile for image normalization.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ColorMapping -in ROI_QB_MUL_1_SVN_CLASS_MULTI.png -method custom -method.custom.lut ROI_QB_MU
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ColorMapping application
ColorMapping = otbApplication.Registry.CreateApplication("ColorMapping")

The following lines set all the application parameters:
ColorMapping.SetParameterString("in", "ROI_QB_MUL_1_SVN_CLASS_MULTI.png")

ColorMapping.SetParameterString("method", "custom")

ColorMapping.SetParameterString("method.custom.lut", "ROI_QB_MUL_1_SVN_CLASS_MULTI_PNG_ColorTable.txt")

ColorMapping.SetParameterString("out", "Colorized_ROI_QB_MUL_1_SVN_CLASS_MULTI.tif")

The following line execute the application
ColorMapping.ExecuteAndWriteOutput()
```

## Limitations

**The segmentation optimal method does not support streaming, and thus large images. The operation color to label is not implemented.** ColorMapping using support image is not threaded.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

[ImageSVMClassifier](#)

## 6.1.2 Images Concatenation

Concatenate a list of images of the same size into a single multi-channel one.

### Detailed description

This application performs images channels concatenation. It will walk the input image list (single or multi-channel) and generates a single multi-channel image. The channel order is the one of the list.

### Parameters

This section describes in details the parameters available for this application. Table <sup>2</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConcatenateImages* .

---

<sup>2</sup> Table: Parameters table for Images Concatenation.

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
out	Output image	Output image
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input images list:** The list of images to concatenate.
- **Output Image:** The concatenated output image.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ConcatenateImages -il GomaAvant.png GomaApres.png -out otbConcatenateImages.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ConcatenateImages application
ConcatenateImages = otbApplication.Registry.CreateApplication("ConcatenateImages")

The following lines set all the application parameters:
ConcatenateImages.SetParameterStringList("il", ['GomaAvant.png', 'GomaApres.png'])

ConcatenateImages.SetParameterString("out", "otbConcatenateImages.tif")

The following line execute the application
ConcatenateImages.ExecuteAndWriteOutput()
```

## Limitations

All input images must have the same size.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

Rescale application, Convert

### 6.1.3 Image Conversion

Convert an image to a different format, eventually rescaling the data and/or changing the pixel type.

#### Detailed description

**This application performs an image pixel type conversion (short, ushort, uchar, int, uint, float and double types are handled). The**

The conversion can include a rescale using the image 2 percent minimum and maximum values. The rescale can be linear or log2.

#### Parameters

This section describes in details the parameters available for this application. Table <sup>3</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Convert* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
type	Choices	Choices
type none	<i>Choice</i>	None
type linear	<i>Choice</i>	Linear
type log2	<i>Choice</i>	Log2
type.linear.gamma	Float	Float
mask	Input image	Input image
hcp	Group	Group
hcp.high	Float	Float
hcp.low	Float	Float
out	Output image	Output image
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input image:** Input image.

**Rescale type:** Transfer function for the rescaling. Available choices are:

- **None**
- **Linear**
- **Gamma correction factor:** Gamma correction factor.
- **Log2**

**Input mask:** The masked pixels won't be used to adapt the dynamic (the mask must have the same dimensions as the input image).

**[Histogram Cutting Parameters]:** Parameters to cut the histogram edges before rescaling.

- **High Cut Quantile:** Quantiles to cut from histogram high values before computing min/max rescaling (in percent, 2 by default).
- **Low Cut Quantile:** Quantiles to cut from histogram low values before computing min/max rescaling (in percent, 2 by default).

---

<sup>3</sup> Table: Parameters table for Image Conversion.

**Output Image:** Output image.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_Convert -in QB_Toulouse_Ortho_XS.tif -out otbConvertWithScalingOutput.png uint8 --type linear
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Convert application
Convert = otbApplication.Registry.CreateApplication("Convert")

The following lines set all the application parameters:
Convert.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

Convert.SetParameterString("out", "otbConvertWithScalingOutput.png")
Convert.SetParameterOutputImagePixelFormat("out", 1)

Convert.SetParameterString("type", "linear")

The following line execute the application
Convert.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

Rescale

## 6.1.4 DEM Conversion

Converts a geo-referenced DEM image into a general raster file compatible with OTB DEM handling.

## Detailed description

In order to be understood by the Orfeo ToolBox and the underlying OSSIM library, a geo-referenced Digital Elevation Model image can be converted into a general raster image, which consists in 3 files with the following extensions: .ras, .geom and .omd. Once converted, you have to place these files in a separate directory, and you can then use this directory to set the “DEM Directory” parameter of a DEM based OTB application or filter.

## Parameters

This section describes in details the parameters available for this application. Table <sup>4</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DEMConvert*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output File name	Output File name
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input geo-referenced DEM:** Input geo-referenced DEM to convert to general raster format.
- **Prefix of the output files:** will be used to get the prefix (name without extensions) of the files to write. Three files - prefix.geom, prefix.omd and prefix.ras - will be generated.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_DEMConvert -in QB_Toulouse_Ortho_Elev.tif -out outputDEM
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the DEMConvert application
DEMConvert = otbApplication.Registry.CreateApplication("DEMConvert")

The following lines set all the application parameters:
DEMConvert.SetParameterString("in", "QB_Toulouse_Ortho_Elev.tif")

DEMConvert.SetParameterString("out", "outputDEM")

The following line execute the application
DEMConvert.ExecuteAndWriteOutput()
```

## Limitations

None

---

<sup>4</sup> Table: Parameters table for DEM Conversion.

## Authors

This application has been written by OTB-Team.

## 6.1.5 Download or list SRTM tiles related to a set of images

Download or list SRTM tiles related to a set of images

### Detailed description

This application allows selecting the appropriate SRTM tiles that covers a list of images. It builds a list of the required tiles. Two modes are available: the first one download those tiles from the USGS SRTM3 website ([http://dds.cr.usgs.gov/srtm/version2\\_1/SRTM3/](http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/)), the second one list those tiles in a local directory. In both cases, you need to indicate the directory in which directory tiles will be download or the location of local SRTM files.

### Parameters

This section describes in details the parameters available for this application. Table <sup>5</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DownloadSRTMTiles* .

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
mode	Choices	Choices
mode download	<i>Choice</i>	Download
mode list	<i>Choice</i>	List tiles
mode.download.outdir	Directory	Directory
mode.list.indir	Directory	Directory
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input images list:** The list of images on which you want to determine corresponding SRTM tiles.

**Download/List corresponding SRTM tiles.** Available choices are:

- **Download:** Download corresponding tiles on USGE server.
- **Output directory:** Directory where zipped tiles will be save. You'll need to unzip all tile files before using them in your application.
- **List tiles:** List tiles in an existing local directory.
- **Input directory:** Input directory where SRTM tiles can are located.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_DownloadSRTMTiles -il QB_Toulouse_Ortho_XS.tif -mode list -mode.list.indir /home/user/srtm_dir
```

<sup>5</sup> Table: Parameters table for Download or list SRTM tiles related to a set of images.

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the DownloadSRTMTiles application
DownloadSRTMTiles = otbApplication.Registry.CreateApplication("DownloadSRTMTiles")

The following lines set all the application parameters:
DownloadSRTMTiles.SetParameterStringList("il", ['QB_Toulouse_Ortho_XS.tif'])

DownloadSRTMTiles.SetParameterString("mode", "list")

DownloadSRTMTiles.SetParameterString("mode.list.indir", "/home/user/srtm_dir/")

The following line execute the application
DownloadSRTMTiles.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.1.6 Extract ROI

Extract a ROI defined by the user.

### Detailed description

This application extracts a Region Of Interest with user defined size, or reference image.

### Parameters

This section describes in details the parameters available for this application. Table <sup>6</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ExtractROI*.

---

<sup>6</sup> Table: Parameters table for Extract ROI.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
mode	Choices	Choices
mode standard	<i>Choice</i>	Standard
mode fit	<i>Choice</i>	Fit
mode.fit.ref	Input image	Input image
mode.fit.elev	Group	Group
mode.fit.elev.dem	Directory	Directory
mode.fit.elev.geoid	Input File name	Input File name
mode.fit.elev.default	Float	Float
startx	Int	Int
starty	Int	Int
sizeX	Int	Int
sizeY	Int	Int
cl	List	List
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image.

**Output Image:** Output image.

**Available RAM (Mb):** Available memory for processing (in MB).

**Extraction mode** Available choices are:

- **Standard:** In standard mode, extract is done according the coordinates entered by the user.
- **Fit:** In fit mode, extract is made to best fit a reference image.
  - **Reference image:** Reference image to define the ROI.
  - **Elevation management:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.
  - **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
  - **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
  - **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Start X:** ROI start x position.

**Start Y:** ROI start y position.

**Size X:** size along x in pixels.

**Size Y:** size along y in pixels.

**Output Image channels:** Channels to write in the output image.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ExtractROI -in VegetationIndex.hd -startx 40 -starty 250 -sizex 150 -sizey 150 -out ExtractROI.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ExtractROI application
ExtractROI = otbApplication.Registry.CreateApplication("ExtractROI")

The following lines set all the application parameters:
ExtractROI.SetParameterString("in", "VegetationIndex.hd")

ExtractROI.SetParameterInt("startx", 40)

ExtractROI.SetParameterInt("starty", 250)

ExtractROI.SetParameterInt("sizex", 150)

ExtractROI.SetParameterInt("sizey", 150)

ExtractROI.SetParameterString("out", "ExtractROI.tif")

The following line execute the application
ExtractROI.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.1.7 No Data management

Manage No-Data

### Detailed description

This application has two modes. The first allows building a mask of no-data pixels from the no-data flags read from the image file. The second allows updating the change the no-data value of an image (pixels value and metadata). This last mode also allows replacing NaN in images with a proper no-data value. To do so, one should activate the NaN is no-data option.

## Parameters

This section describes in details the parameters available for this application. Table <sup>7</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ManageNoData* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
usenan	Boolean	Boolean
mode	Choices	Choices
mode buildmask	<i>Choice</i>	Build a no-data Mask
mode changevalue	<i>Choice</i>	Change the no-data value
mode apply	<i>Choice</i>	Apply a mask as no-data
mode.buildmask.inv	Float	Float
mode.buildmask.outv	Float	Float
mode.changevalue.newv	Float	Float
mode.apply.mask	Input image	Input image
mode.apply.ndval	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input image:** Input image.

**Output Image:** Output image.

**Consider NaN as no-data:** If active, the application will consider NaN as no-data values as well.

**No-data handling mode:** Allows choosing between different no-data handling options. Available choices are:

- **Build a no-data Mask**
- **Inside Value:** Value given in the output mask to pixels that are not no data pixels.
- **Outside Value:** Value given in the output mask to pixels that are no data pixels.
- **Change the no-data value**
- **The new no-data value:** The new no-data value.
- **Apply a mask as no-data:** Apply an external mask to an image using the no-data value of the input image.
- **Mask image:** Mask to be applied on input image (valid pixels have non null values).
- **Nodata value used:** No Data value used according to the mask image.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ManageNoData -in QB_Toulouse_Ortho_XS.tif -out QB_Toulouse_Ortho_XS_nodatamask.tif uint8 -mod
```

<sup>7</sup> Table: Parameters table for No Data management.

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ManageNoData application
ManageNoData = otbApplication.Registry.CreateApplication("ManageNoData")

The following lines set all the application parameters:
ManageNoData.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

ManageNoData.SetParameterString("out", "QB_Toulouse_Ortho_XS_nodatamask.tif")
ManageNoData.SetParameterOutputImagePixelFormat("out", 1)

ManageNoData.SetParameterFloat("mode.buildmask.inv", 255)

ManageNoData.SetParameterFloat("mode.buildmask.outv", 0)

The following line execute the application
ManageNoData.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

BanMath

## 6.1.8 Multi Resolution Pyramid

Build a multi-resolution pyramid of the image.

### Detailed description

This application builds a multi-resolution pyramid of the input image. User can specified the number of levels of the pyramid and the subsampling factor. To speed up the process, you can use the fast scheme option

### Parameters

This section describes in details the parameters available for this application. Table <sup>8</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is

---

<sup>8</sup> Table: Parameters table for Multi Resolution Pyramid.

*MultiResolutionPyramid* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
level	Int	Int
sfactor	Int	Int
vfactor	Float	Float
fast	Boolean	Boolean
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image**
- **Output Image:** will be used to get the prefix and the extension of the images to write.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Number Of Levels:** Number of levels in the pyramid (default is 1).
- **Subsampling factor:** Subsampling factor between each level of the pyramid (default is 2).
- **Variance factor:** Variance factor use in smoothing. It is multiplied by the subsampling factor of each level in the pyramid (default is 0.6).
- **Use Fast Scheme:** If used, this option allows one to speed-up computation by iteratively subsampling previous level of pyramid instead of processing the full input.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_MultiResolutionPyramid -in QB_Toulouse_Ortho_XS.tif -out multiResolutionImage.tif -level 1 -s
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the MultiResolutionPyramid application
MultiResolutionPyramid = otbApplication.Registry.CreateApplication("MultiResolutionPyramid")

The following lines set all the application parameters:
MultiResolutionPyramid.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

MultiResolutionPyramid.SetParameterString("out", "multiResolutionImage.tif")

MultiResolutionPyramid.SetParameterInt("level", 1)

MultiResolutionPyramid.SetParameterInt("sfactor", 2)

MultiResolutionPyramid.SetParameterFloat("vfactor", 0.6)
```

```
MultiResolutionPyramid.SetParameterString("fast","1")

The following line execute the application
MultiResolutionPyramid.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.1.9 Quick Look

Generates a subsampled version of an image extract

### Detailed description

**Generates a subsampled version of an extract of an image defined by ROIStart and ROISize.** This extract is subsampled using the ratio OR the output image Size.

### Parameters

This section describes in details the parameters available for this application. Table <sup>9</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Quicklook*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
cl	List	List
rox	Int	Int
roy	Int	Int
rsx	Int	Int
rsy	Int	Int
sr	Int	Int
sx	Int	Int
sy	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The image to read.
- **Output Image:** The subsampled image.
- **Channel List:** Selected channels.
- **ROI Origin X:** first point of ROI in x-direction.
- **ROI Origin Y:** first point of ROI in y-direction.

---

<sup>9</sup> Table: Parameters table for Quick Look.

- **ROI Size X**: size of ROI in x-direction.
- **ROI Size Y**: size of ROI in y-direction.
- **Sampling ratio**: Sampling Ratio, default is 2.
- **Size X**: quicklook size in x-direction (used if no sampling ration is given).
- **Size Y**: quicklook size in y-direction (used if no sampling ration is given).
- **Load otb application from xml file**: Load otb application from xml file.
- **Save otb application to xml file**: Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_Quicklook -in qb_RoadExtract.tif -out quicklookImage.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Quicklook application
Quicklook = otbApplication.Registry.CreateApplication("Quicklook")

The following lines set all the application parameters:
Quicklook.SetParameterString("in", "qb_RoadExtract.tif")

Quicklook.SetParameterString("out", "quicklookImage.tif")

The following line execute the application
Quicklook.ExecuteAndWriteOutput()
```

## Limitations

This application does not provide yet the optimal way to decode coarser level of resolution from JPEG2000 images (like in Monteverdi). Trying to subsampled huge JPEG200 image with the application will lead to poor performances for now.

## Authors

This application has been written by OTB-Team.

### 6.1.10 Read image information

Get information about the image

#### Detailed description

Display information about the input image like: image size, origin, spacing, metadata, projections...

## Parameters

This section describes in details the parameters available for this application. Table <sup>10</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ReadImageInfo* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
keywordlist	Boolean	Boolean
outkwl	Output File name	Output File name
indexx	Int	Int
indexy	Int	Int
sizeX	Int	Int
sizey	Int	Int
spacingx	Float	Float
spacingy	Float	Float
originx	Float	Float
originy	Float	Float
estimatedgroundspacingx	Float	Float
estimatedgroundspacingy	Float	Float
numberbands	Int	Int
sensor	String	String
id	String	String
time	String	String
ullat	Float	Float
ullon	Float	Float
urlat	Float	Float
urlon	Float	Float
lrlat	Float	Float
lrlon	Float	Float
lllat	Float	Float
lllon	Float	Float
town	String	String
country	String	String
rgb	Group	Group
rgb.r	Int	Int
rgb.g	Int	Int
rgb.b	Int	Int
projectionref	String	String
keyword	String	String
gcp	Group	Group
gcp.count	Int	Int
gcp.proj	String	String
gcp.ids	String list	String list
gcp.info	String list	String list
gcp.imcoord	String list	String list
gcp.geocoord	String list	String list
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

<sup>10</sup> Table: Parameters table for Read image information.

**Input Image:** Input image to analyse.

**Display the OSSIM keywordlist:** Output the OSSIM keyword list. It contains metadata information (sensor model, geometry ). Information is stored in keyword list (pairs of key/value).

**Write the OSSIM keywordlist to a geom file:** This option allows extracting the OSSIM keywordlist of the image into a geom file.

**Start index X:** X start index.

**Start index Y:** Y start index.

**Size X:** X size (in pixels).

**Size Y:** Y size (in pixels).

**Pixel Size X:** Pixel size along X (in physical units).

**Pixel Size Y:** Pixel size along Y (in physical units).

**Image Origin X:** Origin along X.

**Image Origin Y:** Origin along Y.

**Estimated ground spacing X:** Estimated ground spacing along X (in meters).

**Estimated ground spacing Y:** Estimated ground spacing along Y (in meters).

**Number Of Bands:** Number of bands.

**Sensor id:** Sensor identifier.

**Image id:** Image identifier.

**Acquisition time:** Acquisition time.

**Upper left latitude:** Latitude of the upper left corner.

**Upper left longitude:** Longitude of the upper left corner.

**Upper right latitude:** Latitude of the upper right corner.

**Upper right longitude:** Longitude of the upper right corner.

**Lower right latitude:** Latitude of the lower right corner.

**Lower right longitude:** Longitude of the lower right corner.

**Lower left latitude:** Latitude of the lower left corner.

**Lower left longitude:** Longitude of the lower left corner.

**Nearest town:** Main town near center of image.

**Country:** Country of the image.

**[Default RGB Display]:** This group of parameters provide information about the default rgb composition.

- **Red Band:** Red band Number.
- **Green Band:** Green band Number.
- **Blue Band:** Blue band Number.

**Projection:** Projection Coordinate System.

**Keywordlist:** Image keyword list.

**[Ground Control Points information]:** This group of parameters provide information about all GCPs.

- **GCPs Number:** Number of GCPs.

- **GCP Projection:** Projection Coordinate System for GCPs.
- **GCPs Id:** GCPs identifier.
- **GCPs Info:** GCPs Information.
- **GCPs Image Coordinates:** GCPs Image coordinates.
- **GCPs Geographic Coordinates:** GCPs Geographic Coordinates.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ReadImageInfo -in QB_Toulouse_Ortho_XS.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ReadImageInfo application
ReadImageInfo = otbApplication.Registry.CreateApplication("ReadImageInfo")

The following lines set all the application parameters:
ReadImageInfo.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

The following line execute the application
ReadImageInfo.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.1.11 Rescale Image

Rescale the image between two given values.

### Detailed description

This application scales the given image pixel intensity between two given values. By default min (resp. max) value is set to 0 (resp. 255).

## Parameters

This section describes in details the parameters available for this application. Table <sup>11</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Rescale*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
outmin	Float	Float
outmax	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The image to scale.
- **Output Image:** The rescaled image filename.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Output min value:** Minimum value of the output image.
- **Output max value:** Maximum value of the output image.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_Rescale -in QB_Toulouse_Ortho_PAN.tif -out rescaledImage.png uchar -outmin 0 -outmax 255
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Rescale application
Rescale = otbApplication.Registry.CreateApplication("Rescale")

The following lines set all the application parameters:
Rescale.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Rescale.SetParameterString("out", "rescaledImage.png")
Rescale.SetParameterOutputImagePixelFormat("out", 1)

Rescale.SetParameterFloat("outmin", 0)

Rescale.SetParameterFloat("outmax", 255)

The following line execute the application
Rescale.ExecuteAndWriteOutput()
```

<sup>11</sup> Table: Parameters table for Rescale Image.

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.1.12 Split Image

Split a N multiband image into N images

### Detailed description

This application splits a N-bands image into N mono-band images. The output images filename will be generated from the output parameter. Thus if the input image has 2 channels, and the user has set an output outimage.tif, the generated images will be outimage\_0.tif and outimage\_1.tif

### Parameters

This section describes in details the parameters available for this application. Table <sup>12</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SplitImage* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input multiband image filename.
- **Output Image:** Output filename that will be used to get the prefix and the extension of the output images to write.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_SplitImage -in VegetationIndex.hd -out splitImage.tif
```

To run this example from Python, use the following code snippet:

---

<sup>12</sup> Table: Parameters table for Split Image.

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SplitImage application
SplitImage = otbApplication.Registry.CreateApplication("SplitImage")

The following lines set all the application parameters:
SplitImage.SetParameterString("in", "VegetationIndex.hd")

SplitImage.SetParameterString("out", "splitImage.tif")

The following line execute the application
SplitImage.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.1.13 Image Tile Fusion

Fusion of an image made of several tile files.

### Detailed description

Concatenate several tile files into a single image file.

### Parameters

This section describes in details the parameters available for this application. Table <sup>13</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TileFusion*.

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
cols	Int	Int
rows	Int	Int
out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Tile Images:** Input tiles to concatenate (in lexicographic order : (0,0) (1,0) (0,1) (1,1)).
- **Number of tile columns:** Number of columns in the tile array.
- **Number of tile rows:** Number of rows in the tile array.

<sup>13</sup> Table: Parameters table for Image Tile Fusion.

- **Output Image:** Output entire image.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_TileFusion -il Scene_R1C1.tif Scene_R1C2.tif Scene_R2C1.tif Scene_R2C2.tif -cols 2 -rows 2 -o
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the TileFusion application
TileFusion = otbApplication.Registry.CreateApplication("TileFusion")

The following lines set all the application parameters:
TileFusion.SetParameterStringList("il", ['Scene_R1C1.tif', 'Scene_R1C2.tif', 'Scene_R2C1.tif', 'Scene_R2C2.tif'])

TileFusion.SetParameterInt("cols", 2)

TileFusion.SetParameterInt("rows", 2)

TileFusion.SetParameterString("out", "EntireImage.tif")

The following line execute the application
TileFusion.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.2 Vector Data Manipulation

### 6.2.1 Concatenate

Concatenate VectorDatas

#### Detailed description

This application concatenates a list of VectorData to produce a unique VectorData as output. Note that the VectorDatas must be of the same type (Storing polygons only, lines only, or points only)

## Parameters

This section describes in details the parameters available for this application. Table <sup>14</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConcatenateVectorData* .

Parameter Key	Parameter Type	Parameter Description
vd	Input vector data list	Input vector data list
out	Output vector data	Output vector data
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input VectorDatas to concatenate:** VectorData files to be concatenated in an unique VectorData.
- **Concatenated VectorData:** Output concatenated VectorData.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ConcatenateVectorData -vd ToulousePoints-examples.shp ToulouseRoad-examples.shp --out ConcatenatedVectorData.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ConcatenateVectorData application
ConcatenateVectorData = otbApplication.Registry.CreateApplication("ConcatenateVectorData")

The following lines set all the application parameters:
ConcatenateVectorData.SetParameterStringList("vd", ['ToulousePoints-examples.shp', 'ToulouseRoad-examples.shp'])

ConcatenateVectorData.SetParameterString("out", "ConcatenatedVectorData.shp")

The following line execute the application
ConcatenateVectorData.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

<sup>14</sup> Table: Parameters table for Concatenate.

## 6.2.2 Rasterization

Rasterize a vector dataset.

### Detailed description

**This application allows reprojecting and rasterize a vector dataset. The grid of the rasterized output can be set by using a reference image.**

There are two rasterize mode available in the application. The first is the binary mode: it allows rendering all pixels belonging to a geometry of the input dataset in the foreground color, while rendering the other in background color. The second one allows rendering pixels belonging to a geometry with respect to an attribute of this geometry. The field of the attribute to render can be set by the user. In the second mode, the background value is still used for unassociated pixels.

### Parameters

This section describes in details the parameters available for this application. Table <sup>15</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Rasterization*.

Parameter Key	Parameter Type	Parameter Description
in	Input vector data	Input vector data
out	Output image	Output image
im	Input image	Input image
szx	Int	Int
szy	Int	Int
epsg	Int	Int
orx	Float	Float
ory	Float	Float
spx	Float	Float
spy	Float	Float
background	Float	Float
mode	Choices	Choices
mode binary	<i>Choice</i>	Binary mode
mode attribute	<i>Choice</i>	Attribute burning mode
mode.binary.foreground	Float	Float
mode.attribute.field	String	String
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input vector dataset:** The input vector dataset to be rasterized.

**Output image:** An output image containing the rasterized vector dataset.

**Input reference image:** A reference image from which to import output grid and projection reference system information.

**Output size x:** Output size along x axis (useless if support image is given).

**Output size y:** Output size along y axis (useless if support image is given).

**Output EPSG code:** EPSG code for the output projection reference system (EPSG 4326 for WGS84, 32631 for UTM31N...,useless if support image is given).

<sup>15</sup> Table: Parameters table for Rasterization.

**Output Upper-left x:** Output upper-left corner x coordinate (useless if support image is given).

**Output Upper-left y:** Output upper-left corner y coordinate (useless if support image is given).

**Spacing (GSD) x:** Spacing (ground sampling distance) along x axis (useless if support image is given).

**Spacing (GSD) y:** Spacing (ground sampling distance) along y axis (useless if support image is given).

**Background value:** Default value for pixels not belonging to any geometry.

**Rasterization mode:** Choice of rasterization modes. Available choices are:

- **Binary mode:** In this mode, pixels within a geometry will hold the user-defined foreground value.
- **Foreground value:** Value for pixels inside a geometry.
- **Attribute burning mode:** In this mode, pixels within a geometry will hold the value of a user-defined field extracted from this geometry.
- **The attribute field to burn:** Name of the attribute field to burn.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_Rasterization -in qb_RoadExtract_classification.shp -out rasterImage.tif -spx 1. -spy 1.
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Rasterization application
Rasterization = otbApplication.Registry.CreateApplication("Rasterization")

The following lines set all the application parameters:
Rasterization.SetParameterString("in", "qb_RoadExtract_classification.shp")

Rasterization.SetParameterString("out", "rasterImage.tif")

Rasterization.SetParameterFloat("spx", 1.)

Rasterization.SetParameterFloat("spy", 1.)

The following line execute the application
Rasterization.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

For now, support of input dataset with multiple layers having different projection reference system is limited.

## 6.2.3 VectorData Extract ROI

Perform an extract ROI on the input vector data according to the input image extent

### Detailed description

This application extracts the vector data features belonging to a region specified by the support image envelope. Any features intersecting the support region is copied to output. The output geometries are NOT cropped.

### Parameters

This section describes in details the parameters available for this application. Table <sup>16</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataExtractROI*.

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.vd	Input vector data	Input vector data
io.in	Input image	Input image
io.out	Output vector data	Output vector data
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** Group containing input and output parameters.

- **Input Vector data:** Input vector data.
- **Support image:** Support image that specifies the extracted region.
- **Output Vector data:** Output extracted vector data.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.

---

<sup>16</sup> Table: Parameters table for VectorData Extract ROI.

- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with `no_data` in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with `no_data` in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_VectorDataExtractROI -io.in qb_RoadExtract.tif -io.vd qb_RoadExtract_classification.shp -io.out
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the VectorDataExtractROI application
VectorDataExtractROI = otbApplication.Registry.CreateApplication("VectorDataExtractROI")

The following lines set all the application parameters:
VectorDataExtractROI.SetParameterString("io.in", "qb_RoadExtract.tif")

VectorDataExtractROI.SetParameterString("io.vd", "qb_RoadExtract_classification.shp")

VectorDataExtractROI.SetParameterString("io.out", "apTvUtVectorDataExtractROIApplicationTest.shp")

The following line execute the application
VectorDataExtractROI.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.2.4 Vector Data reprojection

Reproject a vector data using support image projection reference, or a user specified map projection

### Detailed description

This application allows reprojecting a vector data using support image projection reference, or a user given map projection. If given, image keywordlist can be added to reprojected vectordata.

## Parameters

This section describes in details the parameters available for this application. Table <sup>17</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataReprojection* .

Parameter Key	Parameter Type	Parameter Description
in	Group	Group
in.vd	Input File name	Input File name
in.kwl	Input image	Input image
out	Group	Group
out.vd	Output File name	Output File name
out.proj	Choices	Choices
out.proj image	<i>Choice</i>	Use image projection ref
out.proj user	<i>Choice</i>	User defined projection
out.proj.image.in	Input image	Input image
out.proj.user.map	Choices	Choices
out.proj.user.map utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
out.proj.user.map lambert2	<i>Choice</i>	Lambert II Etendu
out.proj.user.map lambert93	<i>Choice</i>	Lambert93
out.proj.user.map wgs	<i>Choice</i>	WGS 84
out.proj.user.map epsg	<i>Choice</i>	EPSG Code
out.proj.user.map.utm.zone	Int	Int
out.proj.user.map.utm.northhem	Boolean	Boolean
out.proj.user.map.epsg.code	Int	Int
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

### [Input data]

- **Input vector data:** The input vector data to reproject.
- **Use image keywords list:** Optional input image to fill vector data with image kwl.

### [Output data]

- **Output vector data:** The reprojected vector data.
- **Output Projection choice** Available choices are:
  - **Use image projection ref:** Vector data will be reprojected in image projection ref.
  - **Image used to get projection map:** Projection map will be found using image metadata.
  - **User defined projection**
    - **Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:
      - **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
      - **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).

<sup>17</sup> Table: Parameters table for Vector Data reprojection.

- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.
- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occur if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_VectorDataReprojection -in.vd VectorData_QB1.shp -out.proj image -out.proj.image.in ROI_QB_MUL_1.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the VectorDataReprojection application
VectorDataReprojection = otbApplication.Registry.CreateApplication("VectorDataReprojection")

The following lines set all the application parameters:
VectorDataReprojection.SetParameterString("in.vd", "VectorData_QB1.shp")

VectorDataReprojection.SetParameterString("out.proj", "image")

VectorDataReprojection.SetParameterString("out.proj.image.in", "ROI_QB_MUL_1.tif")
```

```
VectorDataReprojection.SetParameterString("out.vd", "reprojected_vd.shp")

The following line execute the application
VectorDataReprojection.ExecuteAndWriteOutput()
```

## Authors

This application has been written by OTB-Team.

## 6.2.5 Vector data set field

Set a field in vector data.

### Detailed description

Set a specified field to a specified value on all features of a vector data.

### Parameters

This section describes in details the parameters available for this application. Table <sup>18</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataSetField*.

Parameter Key	Parameter Type	Parameter Description
in	Input vector data	Input vector data
out	Output vector data	Output vector data
fn	String	String
fv	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input:** Input Vector Data.
- **Output:** Output Vector Data.
- **Field:** Field name.
- **Value:** Field value.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_VectorDataSetField -in qb_RoadExtract_classification.shp -out VectorDataSetField.shp -fn Info
```

To run this example from Python, use the following code snippet:

---

<sup>18</sup> Table: Parameters table for Vector data set field.

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the VectorDataSetField application
VectorDataSetField = otbApplication.Registry.CreateApplication("VectorDataSetField")

The following lines set all the application parameters:
VectorDataSetField.SetParameterString("in", "qb_RoadExtract_classification.shp")

VectorDataSetField.SetParameterString("out", "VectorDataSetField.shp")

VectorDataSetField.SetParameterString("fn", "Info")

VectorDataSetField.SetParameterString("fv", "Sample polygon")

The following line execute the application
VectorDataSetField.ExecuteAndWriteOutput()
```

### Limitations

Doesn't work with KML files yet

### Authors

This application has been written by OTB-Team.

## 6.2.6 Vector Data Transformation

Apply a transform to each vertex of the input VectorData

### Detailed description

This application performs a transformation of an input vector data transforming each vertex in the vector data. The applied transformation manages translation, rotation and scale, and can be centered or not.

### Parameters

This section describes in details the parameters available for this application. Table <sup>19</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataTransform* .

<sup>19</sup> Table: Parameters table for Vector Data Transformation.

Parameter Key	Parameter Type	Parameter Description
vd	Input vector data	Input vector data
out	Output vector data	Output vector data
in	Input image	Input image
transform	Group	Group
transform.tx	Float	Float
transform.ty	Float	Float
transform.ro	Float	Float
transform.centerx	Float	Float
transform.centery	Float	Float
transform.scale	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Vector data:** Input vector data to transform.

**Output Vector data:** Output transformed vector data.

**Support image:** Image needed as a support to the vector data.

**[Transform parameters]:** Group of parameters to define the transform.

- **Translation X:** Translation in the X direction (in pixels).
- **Translation Y:** Translation in the Y direction (in pixels).
- **Rotation Angle:** Angle of the rotation to apply in degrees.
- **Center X:** X coordinate of the rotation center (in physical units).
- **Center Y:** Y coordinate of the rotation center (in physical units).
- **Scale:** The scale to apply.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_VectorDataTransform -vd qb_RoadExtract_easyClassification.shp -in qb_RoadExtract.tif -out Vect
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the VectorDataTransform application
VectorDataTransform = otbApplication.Registry.CreateApplication("VectorDataTransform")

The following lines set all the application parameters:
VectorDataTransform.SetParameterString("vd", "qb_RoadExtract_easyClassification.shp")

VectorDataTransform.SetParameterString("in", "qb_RoadExtract.tif")

VectorDataTransform.SetParameterString("out", "VectorDataTransform.shp")
```

```
VectorDataTransform.SetParameterFloat("transform.ro", 5)

The following line execute the application
VectorDataTransform.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.3 Calibration

### 6.3.1 Optical calibration

Perform optical calibration TOA/TOC (Top Of Atmosphere/Top Of Canopy). Supported sensors: QuickBird, Ikonos, WorldView2, Formosat, Spot5, Pleiades, Spot6, Spot7. For other sensors the application also allows providing calibration parameters manually.

#### Detailed description

The application allows converting pixel values from DN (for Digital Numbers) to reflectance. Calibrated values are called surface reflectivity and its values lie in the range [0, 1]. The first level is called Top Of Atmosphere (TOA) reflectivity. It takes into account the sensor gain, sensor spectral response and the solar illuminations. The second level is called Top Of Canopy (TOC) reflectivity. In addition to sensor gain and solar illuminations, it takes into account the optical thickness of the atmosphere, the atmospheric pressure, the water vapor amount, the ozone amount, as well as the composition and amount of aerosol gasses. It is also possible to indicate an AERONET file which contains atmospheric parameters (version 1 and version 2 of Aeronet file are supported. Note that computing TOC reflectivity will internally compute first TOA and then TOC reflectance.

If the sensor is not supported by the metadata interface factory of OTB, users still have the possibility to give the needed parameters to the application. For TOA conversion, these parameters are : - day and month of acquisition, or flux normalization coefficient; - sun elevation angle; - gains and biases, one pair of values for each band (passed by a file); - solar illuminations, one value for each band (passed by a file).

For the conversion from DN (for Digital Numbers) to spectral radiance (or 'TOA radiance') L, the following formula is used :

1.  $L(b) = DN(b)/gain(b)+bias(b)$  (in W/m2/steradians/micrometers) with b being a band ID.

These values are provided by the user thanks to a simple txt file with two lines, one for the gains and one for the biases. Each value must be separated with colons (:), with eventual spaces. Blank lines are not allowed. If a line begins with the '#' symbol, then it is considered as comments. Note that sometimes, the values provided by certain metadata files assume the formula  $L(b) = gain(b)*DC(b)+bias(b)$ . In this case, be sure to provide the inverse gain values so that the application can correctly interpret them.

In order to convert TOA radiance to TOA reflectance, the following formula is used :

2.  $R(b) = (\pi * L(b) * d * d) / (ESUN(b) * \cos(\theta))$  (no dimension) where :

- $L(b)$  is the spectral radiance for band  $b$
- $\pi$  is the famous mathematical constant (3.14159...)
- $d$  is the earth-sun distance (in astronomical units) and depends on the acquisition's day and month
- $ESUN(b)$  is the mean TOA solar irradiance (or solar illumination) in W/m2/micrometers
- $\theta$  is the solar zenith angle in degrees.

Note that the application asks for the solar elevation angle, and will perform the conversion to the zenith angle itself (zenith\_angle = 90 - elevation\_angle , units : degrees). Note also that  $ESUN(b)$  not only depends on the band  $b$ , but also on the spectral sensitivity of the sensor in this particular band. In other words, the influence of spectral sensitivities is included within the  $ESUN$  different values. These values are provided by the user thanks to a txt file following the same convention as before. Instead of providing the date of acquisition, the user can also provide a flux normalization coefficient 'fn'. The formula used instead will be the following :

3.  $R(b) = (\pi * L(b)) / (ESUN(b) * fn * fn * \cos(\theta))$

Whatever the formula used (2 or 3), the user should pay attention to the interpretation of the parameters he will provide to the application, by taking into account the original formula that the metadata files assumes.

Below, we give two examples of txt files containing information about gains/biases and solar illuminations :

- gainbias.txt :

```
Gain values for each band. Each value must be separated with colons (:), with eventual spaces. Blank lines not allowed. 10.4416 : 9.529 : 8.5175 : 14.0063 # Bias values for each band. 0.0 : 0.0 : 0.0 : 0.0
```

- solarillumination.txt :

```
Solar illumination values in watt/m2/micron ('micron' means actually 'for each band'). # Each value must be separated with colons (:), with eventual spaces. Blank lines not allowed. 1540.494123 : 1826.087443 : 1982.671954 : 1094.747446
```

Finally, the 'Logs' tab provides useful messages that can help the user in knowing the process different status.

## Parameters

This section describes in details the parameters available for this application. Table <sup>20</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *OpticalCalibration* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
level	Choices	Choices
level toa	Choice	Image to Top Of Atmosphere reflectance
level toatoim	Choice	TOA reflectance to Image
level toc	Choice	Image to Top Of Canopy reflectance (atmospheric corrections)
milli	Boolean	Boolean
clamp	Boolean	Boolean
acqui	Group	Group
acqui.minute	Int	Int

Continued on next page

<sup>20</sup> Table: Parameters table for Optical calibration.

Table 6.3 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
acqui.hour	Int	Int
acqui.day	Int	Int
acqui.month	Int	Int
acqui.year	Int	Int
acqui.fluxnormcoeff	Float	Float
acqui.sun	Group	Group
acqui.sun.elev	Float	Float
acqui.sun.azim	Float	Float
acqui.view	Group	Group
acqui.view.elev	Float	Float
acqui.view.azim	Float	Float
acqui.gainbias	Input File name	Input File name
acqui.solarilluminations	Input File name	Input File name
atmo	Group	Group
atmo.aerosol	Choices	Choices
atmo.aerosol noaerosol	<i>Choice</i>	No Aerosol Model
atmo.aerosol continental	<i>Choice</i>	Continental
atmo.aerosol maritime	<i>Choice</i>	Maritime
atmo.aerosol urban	<i>Choice</i>	Urban
atmo.aerosol desartic	<i>Choice</i>	Desartic
atmo.oz	Float	Float
atmo.wa	Float	Float
atmo.pressure	Float	Float
atmo.opt	Float	Float
atmo.aeronet	Input File name	Input File name
atmo.rsr	Input File name	Input File name
atmo.radius	Int	Int
atmo.pixsize	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input:** Input image filename (values in DN).

**Output:** Output calibrated image filename.

**Available RAM (Mb):** Available memory for processing (in MB).

**Calibration Level** Available choices are:

- **Image to Top Of Atmosphere reflectance**
- **TOA reflectance to Image**
- **Image to Top Of Canopy reflectance (atmospheric corrections)**

**Convert to milli reflectance:** Flag to use milli-reflectance instead of reflectance. This allows saving the image with integer pixel type (in the range [0, 1000] instead of floating point in the range [0, 1]. In order to do that, use this option and set the output pixel type (-out filename double for example).

**Clamp of reflectivity values between [0, 100]:** Clamping in the range [0, 100]. It can be useful to preserve area with specular reflectance.

**[Acquisition parameters]:** This group allows setting the parameters related to the acquisition conditions.

- **Minute:** Minute (0-59).

- **Hour:** Hour (0-23).
- **Day:** Day (1-31).
- **Month:** Month (1-12).
- **Year:** Year.
- **Flux Normalization:** Flux Normalization Coefficient.
- **Sun angles:** This group contains the sun angles.
- **Sun elevation angle (deg):** Sun elevation angle (in degrees).
- **Sun azimuth angle (deg):** Sun azimuth angle (in degrees).
- **Viewing angles:** This group contains the sensor viewing angles.
- **Viewing elevation angle (deg):** Viewing elevation angle (in degrees).
- **Viewing azimuth angle (deg):** Viewing azimuth angle (in degrees).
- **Gains | biases:** Gains | biases.
- **Solar illuminations:** Solar illuminations (one value per band).

[**Atmospheric parameters (for TOC)**]: This group allows setting the atmospheric parameters.

- **Aerosol Model** Available choices are:
  - **No Aerosol Model**
  - **Continental**
  - **Maritime**
  - **Urban**
  - **Desertic**
- **Ozone Amount:** Ozone Amount.
- **Water Vapor Amount:** Water Vapor Amount (in saturation fraction of water).
- **Atmospheric Pressure:** Atmospheric Pressure (in hPa).
- **Aerosol Optical Thickness:** Aerosol Optical Thickness.
- **Aeronet File:** Aeronet file containing atmospheric parameters.
- **Relative Spectral Response File:** Sensor relative spectral response file By default the application gets this information in the metadata.
- **Window radius (adjacency effects):** Window radius for adjacency effects corrections Setting this parameters will enable the correction of adjacency effects.
- **Pixel size (in km):** Pixel size (in km )used to compute adjacency effects, it doesn't have to match the image spacing.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_OpticalCalibration -in QB_1_ortho.tif -level toa -out OpticalCalibration.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the OpticalCalibration application
OpticalCalibration = otbApplication.Registry.CreateApplication("OpticalCalibration")

The following lines set all the application parameters:
OpticalCalibration.SetParameterString("in", "QB_1_ortho.tif")

OpticalCalibration.SetParameterString("level", "toa")

OpticalCalibration.SetParameterString("out", "OpticalCalibration.tif")

The following line execute the application
OpticalCalibration.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

The OTB CookBook

## 6.3.2 SAR Radiometric calibration

Perform radiometric calibration of SAR images. Following sensors are supported: TerraSAR-X, Sentinel1 and Radarsat-2. Both Single Look Complex (SLC) and detected products are supported as input.

### Detailed description

The objective of SAR calibration is to provide imagery in which the pixel values can be directly related to the radar backscatter of the scene. This application allows computing Sigma Naught (Radiometric Calibration) for TerraSAR-X, Sentinel1 L1 and Radarsat-2 sensors. Metadata are automatically retrieved from image products. The application supports complex and non-complex images (SLC or detected products).

## Parameters

This section describes in details the parameters available for this application. Table <sup>21</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARCalibration*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
noise	Boolean	Boolean
lut	Choices	Choices
lut sigma	<i>Choice</i>	Use sigma nought lookup
lut gamma	<i>Choice</i>	Use gamma nought lookup
lut beta	<i>Choice</i>	Use beta nought lookup
lut dn	<i>Choice</i>	Use DN value lookup
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input complex image.
- **Output Image:** Output calibrated image. This image contains the backscatter (sigmaNought) of the input image.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Disable Noise:** Flag to disable noise. For 5.2.0 release, the noise values are only read by TerraSARX product.
- **Lookup table sigma /gamma/ beta/ DN.:** Lookup table values are not available with all SAR products. Products that provide lookup table with metadata are: Sentinel1, Radarsat2. Available choices are:
- **Use sigma nought lookup:** Use Sigma nought lookup value from product metadata.
- **Use gamma nought lookup:** Use Gamma nought lookup value from product metadata.
- **Use beta nought lookup:** Use Beta nought lookup value from product metadata.
- **Use DN value lookup:** Use DN value lookup value from product metadata.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SARCalibration -in RSAT_imagery_HH.tif -out SarRadiometricCalibration.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SARCalibration application
SARCalibration = otbApplication.Registry.CreateApplication("SARCalibration")
```

---

<sup>21</sup> Table: Parameters table for SAR Radiometric calibration.

```
The following lines set all the application parameters:
SARCalibration.SetParameterString("in", "RSAT_imagery_HH.tif")

SARCalibration.SetParameterString("out", "SarRadiometricCalibration.tif")

The following line execute the application
SARCalibration.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.3.3 SARDecompositions

From one-band complex images (each one related to an element of the Sinclair matrix), returns the selected decomposition.

### Detailed description

From one-band complex images (HH, HV, VH, VV), returns the selected decomposition.

All the decompositions implemented are intended for the mono-static case (transmitter and receiver are co-located). There are two kinds of decomposition : coherent ones and incoherent ones. In the coherent case, only the Pauli decomposition is available. In the incoherent case, there the decompositions available : Huynen, Barnes, and H-alpha-A. User must provide three one-band complex images HH, HV or VH, and VV (mono-static case  $\Leftrightarrow$  HV = VH). Incoherent decompositions consist in averaging 3x3 complex coherency/covariance matrices; the user must provide the size of the averaging window, thanks to the parameter `inco.kernelsize`.

### Parameters

This section describes in details the parameters available for this application. Table <sup>22</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARDecompositions* .

---

<sup>22</sup> Table: Parameters table for SARDecompositions.

Parameter Key	Parameter Type	Parameter Description
inhh	Input image	Input image
inhv	Input image	Input image
invh	Input image	Input image
invv	Input image	Input image
out	Output image	Output image
decomp	Choices	Choices
decomp haa	<i>Choice</i>	H-alpha-A incoherent decomposition
decomp barnes	<i>Choice</i>	Barnes incoherent decomposition
decomp huynen	<i>Choice</i>	Huynen incoherent decomposition
decomp pauli	<i>Choice</i>	Pauli coherent decomposition
inco	Group	Group
inco.kernelsize	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image (HH).

**Input Image:** Input image (HV).

**Input Image:** Input image (VH).

**Input Image:** Input image (VV).

**Output Image:** Output image.

**Decompositions** Available choices are:

- **H-alpha-A incoherent decomposition:** H-alpha-A incoherent decomposition.
- **Barnes incoherent decomposition:** Barnes incoherent decomposition.
- **Huynen incoherent decomposition:** Huynen incoherent decomposition.
- **Pauli coherent decomposition:** Pauli coherent decomposition.

**[Incoherent decompositions]:** This group allows setting parameters related to the incoherent decompositions.

- **Kernel size for spatial incoherent averaging.:** Minute (0-59).

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SARDecompositions -inhh HH.tif -invh VH.tif -invv VV.tif -decomp haa -out HaA.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python
Import the otb applications package
import otbApplication
The following line creates an instance of the SARDecompositions application
SARDecompositions = otbApplication.Registry.CreateApplication("SARDecompositions")
```

```
The following lines set all the application parameters:
SARDecompositions.SetParameterString("inhh", "HH.tif")

SARDecompositions.SetParameterString("invh", "VH.tif")

SARDecompositions.SetParameterString("invv", "VV.tif")

SARDecompositions.SetParameterString("decomp", "haa")

SARDecompositions.SetParameterString("out", "HaA.tif")

The following line execute the application
SARDecompositions.ExecuteAndWriteOutput()
```

### Limitations

Some decompositions output real images, while this application outputs complex images for general purpose. Users should pay attention to extract the real part of the results provided by this application.

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

SARPolarMatrixConvert, SARPolarSynth

## 6.3.4 SARPolarMatrixConvert

This applications allows converting classical polarimetric matrices to each other.

### Detailed description

This application allows converting classical polarimetric matrices to each other. For instance, it is possible to get the coherency matrix from the Sinclair one, or the Mueller matrix from the coherency one. The filters used in this application never handle matrices, but images where each band is related to their elements. As most of the time SAR polarimetry handles symmetric/hermitian matrices, only the relevant elements are stored, so that the images representing them have a minimal number of bands. For instance, the coherency matrix size is 3x3 in the monostatic case, and 4x4 in the bistatic case : it will thus be stored in a 6-band or a 10-band complex image (the diagonal and the upper elements of the matrix).

The Sinclair matrix is a special case : it is always represented as 3 or 4 one-band complex images (for mono- or bistatic case). The available conversions are listed below:

- Monostatic case — 1 msinclairtocoherency → Sinclair matrix to coherency matrix (input : 3 x 1 complex channel (HH, HV or VH, VV) | output : 6 complex channels)
- 2 msinclairtocoherency → Sinclair matrix to covariance matrix (input : 3 x 1 complex channel (HH, HV or VH, VV) | output : 6 complex channels)
- 3 msinclairtocoherency → Sinclair matrix to circular covariance matrix (input : 3 x 1 complex channel (HH, HV or VH, VV) | output : 6 complex channels)
- 4 mcoherencytomueller → Coherency matrix to Mueller matrix (input : 6 complex channels | 16 real

channels) 5 mcovariancetocoherencydegree → Covariance matrix to coherency degree (input : 6 complex channels | 3 complex channels) 6 mcovariancetocoherency → Covariance matrix to coherency matrix (input : 6 complex channels | 6 complex channels) 7 mlinearcovariancetocircularcovariance → Covariance matrix to circular covariance matrix (input : 6 complex channels | output : 6 complex channels)

— Bistatic case — 8 bsinclairtocoherency → Sinclair matrix to coherency matrix (input : 4 x 1 complex channel (HH, HV, VH, VV) | 10 complex channels) 9 bsinclairtocovariance → Sinclair matrix to covariance matrix (input : 4 x 1 complex channel (HH, HV, VH, VV) | output : 10 complex channels) 10 bsinclairtocircovariance → Sinclair matrix to circular covariance matrix (input : 4 x 1 complex channel (HH, HV, VH, VV) | output : 10 complex channels)

— Both cases — 11 sinclairtomueller → Sinclair matrix to Mueller matrix (input : 4 x 1 complex channel (HH, HV, VH, VV) | output : 16 real channels) 12 muellertomcovariance → Mueller matrix to covariance matrix (input : 16 real channels | output : 6 complex channels) 13 muellertopoldegandpower → Mueller matrix to polarization degree and power (input : 16 real channels | output : 4 real channels)

## Parameters

This section describes in details the parameters available for this application. Table <sup>23</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARPolarMatrixConvert* .

---

<sup>23</sup> Table: Parameters table for SARPolarMatrixConvert.

Parameter Key	Parameter Type	Parameter Description
inc	Input image	Input image
inf	Input image	Input image
inhh	Input image	Input image
inhv	Input image	Input image
invh	Input image	Input image
invv	Input image	Input image
outc	Output image	Output image
outf	Output image	Output image
conv	Choices	Choices
conv msinclairtocoherency	<i>Choice</i>	1 Monostatic : Sinclair matrix to coherency matrix (complex output)
conv msinclairtocovariance	<i>Choice</i>	2 Monostatic : Sinclair matrix to covariance matrix (complex output)
conv msinclairtocircovariance	<i>Choice</i>	3 Monostatic : Sinclair matrix to circular covariance matrix (complex output)
conv mcoherencytomueller	<i>Choice</i>	4 Monostatic : Coherency matrix to Mueller matrix
conv mcovariancetocoherency-degree	<i>Choice</i>	5 Monostatic : Covariance matrix to coherency degree
conv mcovariancetocoherency	<i>Choice</i>	6 Monostatic : Covariance matrix to coherency matrix (complex output)
conv mlinearcovariancetocircularcovariance	<i>Choice</i>	7 Monostatic : Covariance matrix to circular covariance matrix (complex output)
conv muellertomcovariance	<i>Choice</i>	8 Bi/mono : Mueller matrix to monostatic covariance matrix
conv bsinclairtocoherency	<i>Choice</i>	9 Bistatic : Sinclair matrix to coherency matrix (complex output)
conv bsinclairtocovariance	<i>Choice</i>	10 Bistatic : Sinclair matrix to covariance matrix (complex output)
conv bsinclairtocircovariance	<i>Choice</i>	11 Bistatic : Sinclair matrix to circular covariance matrix (complex output)
conv sinclairtomueller	<i>Choice</i>	12 Bi/mono : Sinclair matrix to Mueller matrix
conv muellertopoldegandpower	<i>Choice</i>	13 Bi/mono : Mueller matrix to polarisation degree and power
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input : multi-band complex image:** Input : multi-band complex image.
- **Input : multi-band real image:** Input : multi-band real image.
- **Input : one-band complex image (HH):** Input : one-band complex image (HH).
- **Input : one-band complex image (HV):** Input : one-band complex image (HV).
- **Input : one-band complex image (VH):** Input : one-band complex image (VH).
- **Input : one-band complex image (VV):** Input : one-band complex image (VV).
- **Output Complex Image:** Output Complex image.
- **Output Real Image:** Output Real image.
- **Conversion** Available choices are:

- **1 Monostatic : Sinclair matrix to coherency matrix (complex output):** 1 Monostatic : Sinclair matrix to coherency matrix (complex output).
- **2 Monostatic : Sinclair matrix to covariance matrix (complex output):** 2 Monostatic : Sinclair matrix to covariance matrix (complex output).
- **3 Monostatic : Sinclair matrix to circular covariance matrix (complex output):** 3 Monostatic : Sinclair matrix to circular covariance matrix (complex output).
- **4 Monostatic : Coherency matrix to Mueller matrix:** 4 Monostatic : Coherency matrix to Mueller matrix.
- **5 Monostatic : Covariance matrix to coherency degree:** 5 Monostatic : Covariance matrix to coherency degree .
- **6 Monostatic : Covariance matrix to coherency matrix (complex output):** 6 Monostatic : Covariance matrix to coherency matrix (complex output).
- **7 Monostatic : Covariance matrix to circular covariance matrix (complex output):** 7 Monostatic : Covariance matrix to circular covariance matrix (complex output).
- **8 Bi/mono : Mueller matrix to monostatic covariance matrix:** 8 Bi/mono : Mueller matrix to monostatic covariance matrix.
- **9 Bistatic : Sinclair matrix to coherency matrix (complex output):** 9 Bistatic : Sinclair matrix to coherency matrix (complex output).
- **10 Bistatic : Sinclair matrix to covariance matrix (complex output):** 10 Bistatic : Sinclair matrix to covariance matrix (complex output).
- **11 Bistatic : Sinclair matrix to circular covariance matrix (complex output):** 11 Bistatic : Sinclair matrix to circular covariance matrix (complex output).
- **12 Bi/mono : Sinclair matrix to Mueller matrix:** 12 Bi/mono : Sinclair matrix to Mueller matrix.
- **13 Bi/mono : Mueller matrix to polarisation degree and power:** 13 Bi/mono : Mueller matrix to polarisation degree and power.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SARPolarMatrixConvert -inhh HH.tif -invh VH.tif -invv VV.tif -conv msinclairtocoherency -outc
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SARPolarMatrixConvert application
SARPolarMatrixConvert = otbApplication.Registry.CreateApplication("SARPolarMatrixConvert")

The following lines set all the application parameters:
SARPolarMatrixConvert.SetParameterString("inhh", "HH.tif")
```

```

SARPolarMatrixConvert.SetParameterString("invh", "VH.tif")
SARPolarMatrixConvert.SetParameterString("invv", "VV.tif")
SARPolarMatrixConvert.SetParameterString("conv", "msinclairtocoherency")
SARPolarMatrixConvert.SetParameterString("outc", "mcoherency.tif")

The following line execute the application
SARPolarMatrixConvert.ExecuteAndWriteOutput()

```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

SARPolarSynth, SARDecompositions

## 6.3.5 SARPolarSynth

Gives, for each pixel, the power that would have been received by a SAR system with a basis different from the classical (H,V) one (polarimetric synthetic).

### Detailed description

This application gives, for each pixel, the power that would have been received by a SAR system with a basis different from the classical (H,V) one (polarimetric synthetic). The new basis A and B are indicated through two Jones vectors, defined by the user thanks to orientation ( $\psi$ ) and ellipticity ( $\kappa$ ) parameters. These parameters are namely  $\psi_{ii}$ ,  $\kappa_{ii}$ ,  $\psi_{ir}$  and  $\kappa_{ir}$ . The suffixes (i) and (r) refer to the transmitting antenna and the receiving antenna respectively. Orientations and ellipticities are given in degrees, and are between -90/90 degrees and -45/45 degrees respectively.

Four polarization architectures can be processed :

1. HH\_HV\_VH\_VV : full polarization, general bistatic case.
2. HH\_HV\_VV or HH\_VH\_VV : full polarization, monostatic case (transmitter and receiver are co-located).
3. HH\_HV : dual polarization.
4. VH\_VV : dual polarization.

The application takes a complex vector image as input, where each band correspond to a particular emission/reception polarization scheme. User must comply with the band order given above, since the bands are used to build the Sinclair matrix.

In order to determine the architecture, the application first relies on the number of bands of the input image.

1. Architecture HH\_HV\_VH\_VV is the only one with four bands, there is no possible confusion.
2. Concerning HH\_HV\_VV and HH\_VH\_VV architectures, both correspond to a three channels image. But they are processed in the same way, as the Sinclair matrix is symmetric in the monostatic case.
3. Finally, the two last architectures (dual polarizations), can't be distinguished only by the number of bands of the input image. User must then use the parameters emissionh and emissionv to indicate the architecture of the system : emissionh=1 and emissionv=0 -> HH\_HV, emissionh=0 and emissionv=1 -> VH\_VV.

Note : if the architecture is HH\_HV, khii and psii are automatically both set to 0 degree; if the architecture is VH\_VV, khii and psii are automatically set to 0 degree and 90 degrees respectively.

It is also possible to force the calculation to co-polar or cross-polar modes. In the co-polar case, values for psir and khir will be ignored and forced to psii and khii; same as the cross-polar mode, where khir and psir will be forced to (psii + 90 degrees) and -khii.

Finally, the result of the polarimetric synthetis is expressed in the power domain, through a one-band scalar image. Note: this application doesn't take into account the terms which do not depend on the polarization of the antennas. The parameter gain can be used for this purpose.

More details can be found in the OTB CookBook (SAR processing chapter).

## Parameters

This section describes in details the parameters available for this application. Table <sup>24</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARPolarSynth* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
psii	Float	Float
khii	Float	Float
psir	Float	Float
khir	Float	Float
emissionh	Int	Int
emissionv	Int	Int
mode	Choices	Choices
mode none	<i>Choice</i>	None
mode co	<i>Choice</i>	Copolarization
mode cross	<i>Choice</i>	Crosspolarization
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input image.
- **Output Image:** Output image.
- **psii:** Orientation (transmitting antenna).
- **khii:** Ellipticity (transmitting antenna).
- **psir:** Orientation (receiving antenna).
- **khir:** Ellipticity (receiving antenna).
- **Emission H:** This parameter is useful in determining the polarization architecture (dual polarization case).

<sup>24</sup> Table: Parameters table for SARPolarSynth.

- **Emission V**: This parameter is useful in determining the polarization architecture (dual polarization case).
- **Forced mode** Available choices are:
  - **None**: Copolarization.
  - **Copolarization**
  - **Crosspolarization**: Crosspolarization.
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Load otb application from xml file**: Load otb application from xml file.
- **Save otb application to xml file**: Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SARPolarSynth -in sar.tif -psii 15. -khii 5. -psir -25. -khir 10. -out newbasis.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SARPolarSynth application
SARPolarSynth = otbApplication.Registry.CreateApplication("SARPolarSynth")

The following lines set all the application parameters:
SARPolarSynth.SetParameterString("in", "sar.tif")

SARPolarSynth.SetParameterFloat("psii", 15.)

SARPolarSynth.SetParameterFloat("khii", 5.)

SARPolarSynth.SetParameterFloat("psir", -25.)

SARPolarSynth.SetParameterFloat("khir", 10.)

SARPolarSynth.SetParameterString("out", "newbasis.tif")

The following line execute the application
SARPolarSynth.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

SARDecompositions, SARPolarMatrixConvert

### 6.3.6 SAR Radiometric calibration (DEPRECATED)

Perform radiometric calibration of SAR images. Following sensors are supported: TerraSAR-X, Sentinel1 and Radarsat-2. Both Single Look Complex (SLC) and detected products are supported as input.

#### Detailed description

The objective of SAR calibration is to provide imagery in which the pixel values can be directly related to the radar backscatter of the scene. This application allows computing Sigma Naught (Radiometric Calibration) for TerraSAR-X, Sentinel1 L1 and Radarsat-2 sensors. Metadata are automatically retrieved from image products. The application supports complex and non-complex images (SLC or detected products).

#### Parameters

This section describes in details the parameters available for this application. Table <sup>25</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SarRadiometricCalibration*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
noise	Boolean	Boolean
lut	Choices	Choices
lut sigma	Choice	Use sigma nought lookup
lut gamma	Choice	Use gamma nought lookup
lut beta	Choice	Use beta nought lookup
lut dn	Choice	Use DN value lookup
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input complex image.
- **Output Image:** Output calibrated image. This image contains the backscatter (sigmaNought) of the input image.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Disable Noise:** Flag to disable noise. For 5.2.0 release, the noise values are only read by TerraSARX product.
- **Lookup table sigma /gamma/ beta/ DN.:** Lookup table values are not available with all SAR products. Products that provide lookup table with metadata are: Sentinel1, Radarsat2. Available choices are:
- **Use sigma nought lookup:** Use Sigma nought lookup value from product metadata.
- **Use gamma nought lookup:** Use Gamma nought lookup value from product metadata.
- **Use beta nought lookup:** Use Beta nought lookup value from product metadata.

<sup>25</sup> Table: Parameters table for SAR Radiometric calibration (DEPRECATED).

- **Use DN value lookup:** Use DN value lookup value from product metadata.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SarRadiometricCalibration -in RSAT_imagery_HH.tif -out SarRadiometricCalibration.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SarRadiometricCalibration application
SarRadiometricCalibration = otbApplication.Registry.CreateApplication("SarRadiometricCalibration")

The following lines set all the application parameters:
SarRadiometricCalibration.SetParameterString("in", "RSAT_imagery_HH.tif")

SarRadiometricCalibration.SetParameterString("out", "SarRadiometricCalibration.tif")

The following line execute the application
SarRadiometricCalibration.ExecuteAndWriteOutput ()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.4 Geometry

### 6.4.1 Bundle to perfect sensor

Perform P+XS pansharpener

#### Detailed description

This application performs P+XS pansharpener. The default mode use Pan and XS sensor models to estimate the transformation to superimpose XS over Pan before the fusion (“default mode”). The application provides also a PHR mode for Pleiades images which does not use sensor models as Pan and XS products are already coregistered but only estimate an affine transformation to superimpose XS over the Pan. Note that this option is automatically activated in case Pleiades images are detected as input.

## Parameters

This section describes in details the parameters available for this application. Table <sup>26</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BundleToPerfectSensor* .

Parameter Key	Parameter Type	Parameter Description
inp	Input image	Input image
inxs	Input image	Input image
out	Output image	Output image
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
mode	Choices	Choices
mode default	<i>Choice</i>	Default mode
mode phr	<i>Choice</i>	Pleiades mode
lms	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input PAN Image:** Input panchromatic image.

**Input XS Image:** Input XS image.

**Output image:** Output image.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Mode:** Superimposition mode. Available choices are:

- **Default mode:** Default superimposition mode : uses any projection reference or sensor model found in the images.
- **Pleiades mode:** Pleiades superimposition mode, designed for the case of a P+XS bundle in SENSOR geometry. It uses a simple transform on the XS image : a scaling and a residual translation.

**Spacing of the deformation field:** Spacing of the deformation field. Default is 10 times the PAN image spacing.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

---

<sup>26</sup> Table: Parameters table for Bundle to perfect sensor.

## Example

To run this example in command-line, use the following:

```
otbcli_BundleToPerfectSensor -inp QB_Toulouse_Ortho_PAN.tif -inxs QB_Toulouse_Ortho_XS.tif -out Bund
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the BundleToPerfectSensor application
BundleToPerfectSensor = otbApplication.Registry.CreateApplication("BundleToPerfectSensor")

The following lines set all the application parameters:
BundleToPerfectSensor.SetParameterString("inp", "QB_Toulouse_Ortho_PAN.tif")

BundleToPerfectSensor.SetParameterString("inxs", "QB_Toulouse_Ortho_XS.tif")

BundleToPerfectSensor.SetParameterString("out", "BundleToPerfectSensor.png")
BundleToPerfectSensor.SetParameterOutputImagePixelFormat("out", 1)

The following line execute the application
BundleToPerfectSensor.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.4.2 Cartographic to geographic coordinates conversion

Convert cartographic coordinates to geographic one.

### Detailed description

This application computes the geographic coordinates from a cartographic one. User has to give the X and Y coordinate and the cartographic projection (UTM/LAMBERT/LAMBERT2/LAMBERT93/SINUS/ECKERT4/TRANSMERCATOR/MOLLWEID/SVY21).

### Parameters

This section describes in details the parameters available for this application. Table <sup>27</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConvertCartoToGeoPoint*.

<sup>27</sup> Table: Parameters table for Cartographic to geographic coordinates conversion.

Parameter Key	Parameter Type	Parameter Description
carto	Group	Group
carto.x	Float	Float
carto.y	Float	Float
mapproj	Choices	Choices
mapproj utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
mapproj lambert2	<i>Choice</i>	Lambert II Etendu
mapproj lambert93	<i>Choice</i>	Lambert93
mapproj wgs	<i>Choice</i>	WGS 84
mapproj epsg	<i>Choice</i>	EPSG Code
mapproj.utm.zone	Int	Int
mapproj.utm.northhem	Boolean	Boolean
mapproj.epsg.code	Int	Int
long	Float	Float
lat	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input cartographic coordinates]**

- **X cartographic coordinates:** X cartographic coordinates in the specified projection.
- **Y cartographic coordinates:** Y cartographic coordinates in the specified projection.

**Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
- **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).
- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.
- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**Output long:** Point longitude coordinates.

**Output lat:** Point latitude coordinates.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

**Example**

To run this example in command-line, use the following:

```
otbcli_ConvertCartoToGeoPoint -carto.x 367074.625 -carto.y 4835740 -mapproj utm -mapproj.utm.northhem
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ConvertCartoToGeoPoint application
ConvertCartoToGeoPoint = otbApplication.Registry.CreateApplication("ConvertCartoToGeoPoint")

The following lines set all the application parameters:
ConvertCartoToGeoPoint.SetParameterFloat("carto.x", 367074.625)

ConvertCartoToGeoPoint.SetParameterFloat("carto.y", 4835740)

ConvertCartoToGeoPoint.SetParameterString("mapproj", "utm")

ConvertCartoToGeoPoint.SetParameterString("mapproj.utm.northhem", "1")

ConvertCartoToGeoPoint.SetParameterInt("mapproj.utm.zone", 31)

The following line execute the application
ConvertCartoToGeoPoint.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.4.3 Convert Sensor Point To Geographic Point

Sensor to geographic coordinates conversion.

### Detailed description

This Application converts a sensor point of an input image to a geographic point using the Forward Sensor Model of the input image.

### Parameters

This section describes in details the parameters available for this application. Table <sup>28</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConvertSensorToGeoPoint*.

<sup>28</sup> Table: Parameters table for Convert Sensor Point To Geographic Point.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
input	Group	Group
input.idx	Float	Float
input.idy	Float	Float
output	Group	Group
output.idx	Float	Float
output.idy	Float	Float
output.town	String	String
output.country	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Sensor image:** Input sensor image.

- **X value of desired point:** X coordinate of the point to transform.
- **Y value of desired point:** Y coordinate of the point to transform.

**[Point Coordinates]**

- **X value of desired point:** X coordinate of the point to transform.
- **Y value of desired point:** Y coordinate of the point to transform.

**[Geographic Coordinates]**

- **Output Point Longitude:** Output point longitude coordinate.
- **Output Point Latitude:** Output point latitude coordinate.
- **Main town near the coordinates computed:** Nearest main town of the computed geographic point.
- **Country of the image:** Country of the input image.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ConvertSensorToGeoPoint -in QB_TOULOUSE_MUL_Extract_500_500.tif -input.idx 200 -input.idy 200
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ConvertSensorToGeoPoint application
ConvertSensorToGeoPoint = otbApplication.Registry.CreateApplication("ConvertSensorToGeoPoint")

The following lines set all the application parameters:
ConvertSensorToGeoPoint.SetParameterString("in", "QB_TOULOUSE_MUL_Extract_500_500.tif")

ConvertSensorToGeoPoint.SetParameterFloat("input.idx", 200)

ConvertSensorToGeoPoint.SetParameterFloat("input.idy", 200)
```

```
The following line execute the application
ConvertSensorToGeoPoint.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

ConvertCartoToGeoPoint application, otbObtainUTMZoneFromGeoPoint

## 6.4.4 Ply 3D files generation

Generate a 3D Ply file from a DEM and a color image.

### Detailed description

Generate a 3D Ply file from a DEM and a color image.

### Parameters

This section describes in details the parameters available for this application. Table <sup>29</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GeneratePlyFile*.

---

<sup>29</sup> Table: Parameters table for Ply 3D files generation.

Parameter Key	Parameter Type	Parameter Description
indem	Input image	Input image
mode	Choices	Choices
mode dem	<i>Choice</i>	DEM
mode 3dgrid	<i>Choice</i>	3D grid
map	Choices	Choices
map utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
map lambert2	<i>Choice</i>	Lambert II Etendu
map lambert93	<i>Choice</i>	Lambert93
map wgs	<i>Choice</i>	WGS 84
map epsg	<i>Choice</i>	EPSG Code
map.utm.zone	Int	Int
map.utm.northhem	Boolean	Boolean
map.epsg.code	Int	Int
incolor	Input image	Input image
out	Output File name	Output File name
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**The input DEM:** The input DEM.

**Conversion Mode** Available choices are:

- **DEM:** DEM conversion mode.
- **3D grid:** 3D grid conversion mode.

**Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
- **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).
- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.
- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;.
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**The input color image:** The input color image.

**The output Ply file:** The output Ply file.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_GeneratePlyFile -indem image_dem.tif -out out.ply -incolor image_color.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the GeneratePlyFile application
GeneratePlyFile = otbApplication.Registry.CreateApplication("GeneratePlyFile")

The following lines set all the application parameters:
GeneratePlyFile.SetParameterString("indem", "image_dem.tif")

GeneratePlyFile.SetParameterString("out", "out.ply")

GeneratePlyFile.SetParameterString("incolor", "image_color.tif")

The following line execute the application
GeneratePlyFile.ExecuteAndWriteOutput()
```

## Authors

This application has been written by OTB-Team.

## 6.4.5 Generate a RPC sensor model

Generate a RPC sensor model from a list of Ground Control Points.

### Detailed description

This application generates a RPC sensor model from a list of Ground Control Points. At least 20 points are required for estimation without elevation support, and 40 points for estimation with elevation support. Elevation support will be automatically deactivated if an insufficient amount of points is provided. The application can optionally output a file containing accuracy statistics for each point, and a vector file containing segments representing points residues. The map projection parameter allows defining a map projection in which the accuracy is evaluated.

### Parameters

This section describes in details the parameters available for this application. Table <sup>30</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GenerateRPCSensorModel*.

<sup>30</sup> Table: Parameters table for Generate a RPC sensor model.

Parameter Key	Parameter Type	Parameter Description
outgeom	Output File name	Output File name
inpoints	Input File name	Input File name
outstat	Output File name	Output File name
outvector	Output File name	Output File name
map	Choices	Choices
map utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
map lambert2	<i>Choice</i>	Lambert II Etendu
map lambert93	<i>Choice</i>	Lambert93
map wgs	<i>Choice</i>	WGS 84
map epsg	<i>Choice</i>	EPSG Code
map.utm.zone	Int	Int
map.utm.northhem	Boolean	Boolean
map.epsg.code	Int	Int
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Output geom file:** Geom file containing the generated RPC sensor model.

**Input file containing tie points:** Input file containing tie points. Points are stored in following format: col row lon lat. Line beginning with # are ignored.

**Output file containing output precision statistics:** Output file containing the following info: ref\_lon ref\_lat elevation predicted\_lon predicted\_lat x\_error\_ref(meters) y\_error\_ref(meters) global\_error\_ref(meters) x\_error(meters) y\_error(meters) overall\_error(meters).

**Output vector file with residues:** File containing segments representing residues.

**Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
- **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).
- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.
- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.

- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with `no_data` in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with `no_data` in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_GenerateRPCSensorModel -outgeom output.geom -inpoints points.txt -map epsg -map.epsg.code 32631
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the GenerateRPCSensorModel application
GenerateRPCSensorModel = otbApplication.Registry.CreateApplication("GenerateRPCSensorModel")

The following lines set all the application parameters:
GenerateRPCSensorModel.SetParameterString("outgeom", "output.geom")

GenerateRPCSensorModel.SetParameterString("inpoints", "points.txt")

GenerateRPCSensorModel.SetParameterString("map", "epsg")

GenerateRPCSensorModel.SetParameterInt("map.epsg.code", 32631)

The following line execute the application
GenerateRPCSensorModel.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

OrthoRectification, HomologousPointsExtraction, RefineSensorModel

## 6.4.6 Grid Based Image Resampling

Resamples an image according to a resampling grid

### Detailed description

This application allows performing image resampling from an input resampling grid.

### Parameters

This section describes in details the parameters available for this application. Table <sup>31</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GridBasedImageResampling* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.in	Input image	Input image
io.out	Output image	Output image
grid	Group	Group
grid.in	Input image	Input image
grid.type	Choices	Choices
grid.type def	<i>Choice</i>	Displacement grid: $G(x_{out}, y_{out}) = (x_{in} - x_{out}, y_{in} - y_{out})$
grid.type loc	<i>Choice</i>	Localisation grid: $G(x_{out}, y_{out}) = (x_{in}, y_{in})$
out	Group	Group
out.ulx	Float	Float
out.uly	Float	Float
out.sizeX	Int	Int
out.sizeY	Int	Int
out.spacingX	Float	Float
out.spacingY	Float	Float
out.default	Float	Float
interpolator	Choices	Choices
interpolator nn	<i>Choice</i>	Nearest Neighbor interpolation
interpolator linear	<i>Choice</i>	Linear interpolation
interpolator bco	<i>Choice</i>	Bicubic interpolation
interpolator.bco.radius	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting the input and output images.

- **Input image:** The input image to resample.
- **Output Image:** The resampled output image.

**[Resampling grid parameters]**

- **Input resampling grid:** The resampling grid.
- **Grid Type:** allows one to choose between two grid types. Available choices are:

<sup>31</sup> Table: Parameters table for Grid Based Image Resampling.

- **Displacement grid:**  $G(x_{out}, y_{out}) = (x_{in} - x_{out}, y_{in} - y_{out})$ : A deformation grid contains at each grid position the offset to apply to this position in order to get to the corresponding point in the input image to resample.
- **Localisation grid:**  $G(x_{out}, y_{out}) = (x_{in}, y_{in})$ : A localisation grid contains at each grid position the corresponding position in the input image to resample.

**[Output Image parameters]:** Parameters of the output image.

- **Upper Left X:** X Coordinate of the upper-left pixel of the output resampled image.
- **Upper Left Y:** Y Coordinate of the upper-left pixel of the output resampled image.
- **Size X:** Size of the output resampled image along X (in pixels).
- **Size Y:** Size of the output resampled image along Y (in pixels).
- **Pixel Size X:** Size of each pixel along X axis.
- **Pixel Size Y:** Size of each pixel along Y axis.
- **Default value:** The default value to give to pixel that falls outside of the input image.

**Interpolation:** This group of parameters allows one to define how the input image will be interpolated during resampling. Available choices are:

- **Nearest Neighbor interpolation:** Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation:** Linear interpolation leads to average image quality but is quite fast.
- **Bicubic interpolation**
  - **Radius for bicubic interpolation:** This parameter allows controlling the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artifacts.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_GridBasedImageResampling -io.in ROI_IKO_PAN_LesHalles_sub.tif -io.out ROI_IKO_PAN_LesHalles_sub.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the GridBasedImageResampling application
GridBasedImageResampling = otbApplication.Registry.CreateApplication("GridBasedImageResampling")

The following lines set all the application parameters:
GridBasedImageResampling.SetParameterString("io.in", "ROI_IKO_PAN_LesHalles_sub.tif")

GridBasedImageResampling.SetParameterString("io.out", "ROI_IKO_PAN_LesHalles_sub_resampled.tif")
GridBasedImageResampling.SetParameterOutputImagePixelType("io.out", 1)

GridBasedImageResampling.SetParameterString("grid.in", "ROI_IKO_PAN_LesHalles_sub_deformation_field.tif")
```

```

GridBasedImageResampling.SetParameterInt ("out.sizeX", 256)

GridBasedImageResampling.SetParameterInt ("out.sizeY", 256)

GridBasedImageResampling.SetParameterString ("grid.type", "def")

The following line execute the application
GridBasedImageResampling.ExecuteAndWriteOutput ()

```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

otbStereorecificationGridGeneration

## 6.4.7 Image Envelope

Extracts an image envelope.

### Detailed description

Build a vector data containing the polygon of the image envelope.

### Parameters

This section describes in details the parameters available for this application. Table <sup>32</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ImageEnvelope* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output vector data	Output vector data
sr	Int	Int
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
proj	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

<sup>32</sup> Table: Parameters table for Image Envelope.

**Input Image:** Input image.

**Output Vector Data:** Vector data file containing the envelope.

**Sampling Rate:** Sampling rate for image edges (in pixel).

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Projection:** Projection to be used to compute the envelope (default is WGS84).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ImageEnvelope -in QB_TOULOUSE_MUL_Extract_500_500.tif -out ImageEnvelope.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python
Import the otb applications package
import otbApplication

The following line creates an instance of the ImageEnvelope application
ImageEnvelope = otbApplication.Registry.CreateApplication("ImageEnvelope")

The following lines set all the application parameters:
ImageEnvelope.SetParameterString("in", "QB_TOULOUSE_MUL_Extract_500_500.tif")

ImageEnvelope.SetParameterString("out", "ImageEnvelope.shp")

The following line execute the application
ImageEnvelope.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.4.8 Ortho-rectification

This application allows ortho-rectification of optical images from supported sensors.

### Detailed description

An inverse sensor model is built from the input image metadata to convert geographical to raw geometry coordinates. This inverse sensor model is then combined with the chosen map projection to build a global coordinate mapping grid. Last, this grid is used to resample using the chosen interpolation algorithm. A Digital Elevation Model can be specified to account for terrain deformations. In case of SPOT5 images, the sensor model can be approximated by an RPC model in order to speed-up computation.

### Parameters

This section describes in details the parameters available for this application. Table <sup>33</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *OrthoRectification* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.in	Input image	Input image
io.out	Output image	Output image
map	Choices	Choices
map utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
map lambert2	<i>Choice</i>	Lambert II Etendu
map lambert93	<i>Choice</i>	Lambert93
map wgs	<i>Choice</i>	WGS 84
map epsg	<i>Choice</i>	EPSG Code
map.utm.zone	Int	Int
map.utm.northhem	Boolean	Boolean
map.epsg.code	Int	Int
outputs	Group	Group
outputs.mode	Choices	Choices
outputs.mode auto	<i>Choice</i>	User Defined
outputs.mode autosize	<i>Choice</i>	Automatic Size from Spacing
outputs.mode autospacing	<i>Choice</i>	Automatic Spacing from Size
outputs.mode outputroi	<i>Choice</i>	Automatic Size from Spacing and output corners
outputs.mode orthofit	<i>Choice</i>	Fit to ortho
outputs.ulx	Float	Float
outputs.uly	Float	Float
outputs.sizeX	Int	Int
outputs.sizeY	Int	Int
outputs.spacingX	Float	Float
outputs.spacingY	Float	Float
Continued on next page		

<sup>33</sup> Table: Parameters table for Ortho-rectification.

Table 6.4 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
outputs.lrx	Float	Float
outputs.lry	Float	Float
outputs.ortho	Input image	Input image
outputs.isotropic	Boolean	Boolean
outputs.default	Float	Float
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
interpolator	Choices	Choices
interpolator bco	<i>Choice</i>	Bicubic interpolation
interpolator nn	<i>Choice</i>	Nearest Neighbor interpolation
interpolator linear	<i>Choice</i>	Linear interpolation
interpolator.bco.radius	Int	Int
opt	Group	Group
opt.rpc	Int	Int
opt.ram	Int	Int
opt.gridspacing	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting the input and output images.

- **Input Image:** The input image to ortho-rectify.
- **Output Image:** The ortho-rectified output image.

**Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
- **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).
- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.
- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**[Output Image Grid]:** This group of parameters allows one to define the grid on which the input image will be resampled.

- **Parameters estimation modes** Available choices are:
- **User Defined:** This mode allows you to fully modify default values.
- **Automatic Size from Spacing:** This mode allows you to automatically compute the optimal image size from given spacing (pixel size) values.

- **Automatic Spacing from Size:** This mode allows you to automatically compute the optimal image spacing (pixel size) from the given size.
- **Automatic Size from Spacing and output corners:** This mode allows you to automatically compute the optimal image size from spacing (pixel size) and output corners.
- **Fit to ortho:** Fit the size, origin and spacing to an existing ortho image (uses the value of outputs.ortho).
- **Upper Left X:** Cartographic X coordinate of upper-left corner (meters for cartographic projections, degrees for geographic ones).
- **Upper Left Y:** Cartographic Y coordinate of the upper-left corner (meters for cartographic projections, degrees for geographic ones).
- **Size X:** Size of projected image along X (in pixels).
- **Size Y:** Size of projected image along Y (in pixels).
- **Pixel Size X:** Size of each pixel along X axis (meters for cartographic projections, degrees for geographic ones).
- **Pixel Size Y:** Size of each pixel along Y axis (meters for cartographic projections, degrees for geographic ones).
- **Lower right X:** Cartographic X coordinate of the lower-right corner (meters for cartographic projections, degrees for geographic ones).
- **Lower right Y:** Cartographic Y coordinate of the lower-right corner (meters for cartographic projections, degrees for geographic ones).
- **Model ortho-image:** A model ortho-image that can be used to compute size, origin and spacing of the output.
- **Force isotropic spacing by default:** Default spacing (pixel size) values are estimated from the sensor modeling of the image. It can therefore result in a non-isotropic spacing. This option allows you to force default values to be isotropic (in this case, the minimum of spacing in both direction is applied. Values overridden by user are not affected by this option.
- **Default pixel value:** Default value to write when outside of input image.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Interpolation:** This group of parameters allows one to define how the input image will be interpolated during resampling. Available choices are:

- **Bicubic interpolation**
- **Radius for bicubic interpolation:** This parameter allows one to control the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artifacts.
- **Nearest Neighbor interpolation:** Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation:** Linear interpolation leads to average image quality but is quite fast.

**[Speed optimization parameters]:** This group of parameters allows optimization of processing time.

- **RPC modeling (points per axis):** Enabling RPC modeling allows one to speed-up SPOT5 ortho-rectification. Value is the number of control points per axis for RPC estimation.
- **Available RAM (Mb):** This allows setting the maximum amount of RAM available for processing. As the writing task is time consuming, it is better to write large pieces of data, which can be achieved by increasing this parameter (pay attention to your system capabilities).
- **Resampling grid spacing:** Resampling is done according to a coordinate mapping deformation grid, whose pixel size is set by this parameter, and expressed in the coordinate system of the output image. The closer to the output spacing this parameter is, the more precise will be the ortho-rectified image, but increasing this parameter will reduce processing time.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_OrthoRectification -io.in QB_TOULOUSE_MUL_Extract_500_500.tif -io.out QB_Toulouse_ortho.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

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

The following lines set all the application parameters:
OrthoRectification.SetParameterString("io.in", "QB_TOULOUSE_MUL_Extract_500_500.tif")

OrthoRectification.SetParameterString("io.out", "QB_Toulouse_ortho.tif")

The following line execute the application
OrthoRectification.ExecuteAndWriteOutput()
```

## Limitations

Supported sensors are Pleiades, SPOT5 (TIF format), Ikonos, Quickbird, Worldview2, GeoEye.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

Ortho-rectification chapter from the OTB Software Guide

## 6.4.9 Pansharpening

Perform P+XS pansharpening

### Detailed description

This application performs P+XS pansharpening. Pansharpening is a process of merging high-resolution panchromatic and lower resolution multispectral imagery to create a single high-resolution color image. Algorithms available in the applications are: RCS, bayesian fusion and Local Mean and Variance Matching(LMVM).

### Parameters

This section describes in details the parameters available for this application. Table <sup>34</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Pansharpening* .

Parameter Key	Parameter Type	Parameter Description
inp	Input image	Input image
inxs	Input image	Input image
out	Output image	Output image
method	Choices	Choices
method rcs	<i>Choice</i>	RCS
method lmvm	<i>Choice</i>	LMVM
method bayes	<i>Choice</i>	Bayesian
method.lmvm.radiusx	Int	Int
method.lmvm.radiusy	Int	Int
method.bayes.lambda	Float	Float
method.bayes.s	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input PAN Image:** Input panchromatic image.

**Input XS Image:** Input XS image.

**Output image:** Output image.

**Algorithm:** Selection of the pan-sharpening method. Available choices are:

- **RCS:** Simple RCS Pan sharpening operation.
- **LMVM:** Local Mean and Variance Matching (LMVM) Pan sharpening.
- **X radius:** Set the x radius of the sliding window.
- **Y radius:** Set the y radius of the sliding window.
- **Bayesian:** Bayesian fusion.
- **Weight:** Set the weighting value.
- **S coefficient:** Set the S coefficient.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

---

<sup>34</sup> Table: Parameters table for Pansharpening.

**Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_Pansharpening -inp QB_Toulouse_Ortho_PAN.tif -inxs QB_Toulouse_Ortho_XS.tif -out Pansharpening.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Pansharpening application
Pansharpening = otbApplication.Registry.CreateApplication("Pansharpening")

The following lines set all the application parameters:
Pansharpening.SetParameterString("inp", "QB_Toulouse_Ortho_PAN.tif")

Pansharpening.SetParameterString("inxs", "QB_Toulouse_Ortho_XS.tif")

Pansharpening.SetParameterString("out", "Pansharpening.tif")
Pansharpening.SetParameterOutputImagePixelType("out", 3)

The following line execute the application
Pansharpening.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.4.10 Refine Sensor Model

Perform least-square fit of a sensor model to a set of tie points

### Detailed description

This application reads a geom file containing a sensor model and a text file containing a list of ground control point, and performs a least-square fit of the sensor model adjustable parameters to these tie points. It produces an updated geom file as output, as well as an optional ground control points based statistics file and a vector file containing residues. The output geom file can then be used to ortho-rectify the data more accurately. Please note that for a proper use of the application, elevation must be correctly set (including DEM and geoid file). The map parameters allows one to choose a map projection in which the accuracy will be estimated in meters.

## Parameters

This section describes in details the parameters available for this application. Table <sup>35</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *RefineSensorModel*.

Parameter Key	Parameter Type	Parameter Description
inggeom	Input File name	Input File name
outgeom	Output File name	Output File name
inpoints	Input File name	Input File name
outstat	Output File name	Output File name
outvector	Output File name	Output File name
map	Choices	Choices
map utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
map lambert2	<i>Choice</i>	Lambert II Etendu
map lambert93	<i>Choice</i>	Lambert93
map wgs	<i>Choice</i>	WGS 84
map epsg	<i>Choice</i>	EPSG Code
map.utm.zone	Int	Int
map.utm.northhem	Boolean	Boolean
map.epsg.code	Int	Int
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input geom file:** Geom file containing the sensor model to refine.

**Output geom file:** Geom file containing the refined sensor model.

**Input file containing tie points:** Input file containing tie points. Points are stored in following format: row col lon lat. Line beginning with # are ignored.

**Output file containing output precision statistics:** Output file containing the following info: ref\_lon ref\_lat elevation predicted\_lon predicted\_lat x\_error\_ref(meters) y\_error\_ref(meters) global\_error\_ref(meters) x\_error(meters) y\_error(meters) overall\_error(meters).

**Output vector file with residues:** File containing segments representing residues.

**Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
- **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).
- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.

<sup>35</sup> Table: Parameters table for Refine Sensor Model.

- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_RefineSensorModel -inggeom input.geom -outgeom output.geom -inpoints points.txt -map epsg -map
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the RefineSensorModel application
RefineSensorModel = otbApplication.Registry.CreateApplication("RefineSensorModel")

The following lines set all the application parameters:
RefineSensorModel.SetParameterString("inggeom", "input.geom")

RefineSensorModel.SetParameterString("outgeom", "output.geom")

RefineSensorModel.SetParameterString("inpoints", "points.txt")

RefineSensorModel.SetParameterString("map", "epsg")

RefineSensorModel.SetParameterInt("map.epsg.code", 32631)

The following line execute the application
RefineSensorModel.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

OrthoRectification, HomologousPointsExtraction

## 6.4.11 Image resampling with a rigid transform

Resample an image with a rigid transform

### Detailed description

This application performs a parametric transform on the input image. Scaling, translation and rotation with scaling factor are handled. Parameters of the transform is expressed in physical units, thus particular attention must be paid on pixel size (value, and sign). Moreover transform is expressed from input space to output space (on the contrary ITK Transforms are expressed from output space to input space).

### Parameters

This section describes in details the parameters available for this application. Table <sup>36</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *RigidTransformResample* .

---

<sup>36</sup> Table: Parameters table for Image resampling with a rigid transform.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
transform	Group	Group
transform.type	Choices	Choices
transform.type id	<i>Choice</i>	id
transform.type translation	<i>Choice</i>	translation
transform.type rotation	<i>Choice</i>	rotation
transform.type.id.scalex	Float	Float
transform.type.id.scaley	Float	Float
transform.type.translation.tx	Float	Float
transform.type.translation.ty	Float	Float
transform.type.translation.scalex	Float	Float
transform.type.translation.scaley	Float	Float
transform.type.rotation.angle	Float	Float
transform.type.rotation.scalex	Float	Float
transform.type.rotation.scaley	Float	Float
interpolator	Choices	Choices
interpolator nn	<i>Choice</i>	Nearest Neighbor interpolation
interpolator linear	<i>Choice</i>	Linear interpolation
interpolator bco	<i>Choice</i>	Bicubic interpolation
interpolator.bco.radius	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input image:** The input image to translate.

**Output image:** The transformed output image.

**[Transform parameters]:** This group of parameters allows setting the transformation to apply.

- **Type of transformation:** Type of transformation. Available transformations are spatial scaling, translation and rotation with scaling factor. Available choices are:
  - **id:** Spatial scaling.
  - **X scaling:** Scaling factor between the output X spacing and the input X spacing.
  - **Y scaling:** Scaling factor between the output Y spacing and the input Y spacing.
  - **translation:** translation.
  - **The X translation (in physical units):** The translation value along X axis (in physical units).
  - **The Y translation (in physical units):** The translation value along Y axis (in physical units).
  - **X scaling:** Scaling factor between the output X spacing and the input X spacing.
  - **Y scaling:** Scaling factor between the output Y spacing and the input Y spacing.
  - **rotation:** rotation.
  - **Rotation angle:** The rotation angle in degree (values between -180 and 180).
  - **X scaling:** Scale factor between the X spacing of the rotated output image and the X spacing of the unrotated image.
  - **Y scaling:** Scale factor between the Y spacing of the rotated output image and the Y spacing of the unrotated image.

**Interpolation:** This group of parameters allows one to define how the input image will be interpolated during resampling. Available choices are:

- **Nearest Neighbor interpolation:** Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation:** Linear interpolation leads to average image quality but is quite fast.
- **Bicubic interpolation**
- **Radius for bicubic interpolation:** This parameter allows controlling the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artifacts.

**Available RAM (Mb):** This allows setting the maximum amount of RAM available for processing. As the writing task is time consuming, it is better to write large pieces of data, which can be achieved by increasing this parameter (pay attention to your system capabilities).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_RigidTransformResample -in qb_toulouse_sub.tif -out rigidTransformImage.tif -transform.type r
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the RigidTransformResample application
RigidTransformResample = otbApplication.Registry.CreateApplication("RigidTransformResample")

The following lines set all the application parameters:
RigidTransformResample.SetParameterString("in", "qb_toulouse_sub.tif")

RigidTransformResample.SetParameterString("out", "rigidTransformImage.tif")

RigidTransformResample.SetParameterString("transform.type", "rotation")

RigidTransformResample.SetParameterFloat("transform.type.rotation.angle", 20)

RigidTransformResample.SetParameterFloat("transform.type.rotation.scalex", 2.)

RigidTransformResample.SetParameterFloat("transform.type.rotation.scaley", 2.)

The following line execute the application
RigidTransformResample.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

Translation

## 6.4.12 Superimpose sensor

Using available image metadata, project one image onto another one

### Detailed description

This application performs the projection of an image into the geometry of another one.

### Parameters

This section describes in details the parameters available for this application. Table <sup>37</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Superimpose* .

Parameter Key	Parameter Type	Parameter Description
inr	Input image	Input image
inm	Input image	Input image
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
lms	Float	Float
out	Output image	Output image
mode	Choices	Choices
mode default	<i>Choice</i>	Default mode
mode phr	<i>Choice</i>	Pleiades mode
interpolator	Choices	Choices
interpolator bco	<i>Choice</i>	Bicubic interpolation
interpolator nn	<i>Choice</i>	Nearest Neighbor interpolation
interpolator linear	<i>Choice</i>	Linear interpolation
interpolator.bco.radius	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Reference input:** The input reference image.

**The image to reproject:** The image to reproject into the geometry of the reference input.

<sup>37</sup> Table: Parameters table for Superimpose sensor.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Spacing of the deformation field:** Generate a coarser deformation field with the given spacing.

**Output image:** Output reprojected image.

**Mode:** Superimposition mode. Available choices are:

- **Default mode:** Default superimposition mode : uses any projection reference or sensor model found in the images.
- **Pleiades mode:** Pleiades superimposition mode, designed for the case of a P+XS bundle in SENSOR geometry. It uses a simple transform on the XS image : a scaling and a residual translation.

**Interpolation:** This group of parameters allows defining how the input image will be interpolated during resampling. Available choices are:

- **Bicubic interpolation:** Bicubic interpolation leads to very good image quality but is slow.
- **Radius for bicubic interpolation:** This parameter allows controlling the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artifacts.
- **Nearest Neighbor interpolation:** Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation:** Linear interpolation leads to average image quality but is quite fast.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_Superimpose -inr QB_Toulouse_Ortho_PAN.tif -inm QB_Toulouse_Ortho_XS.tif -out SuperimposedXS_t
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Superimpose application
Superimpose = otbApplication.Registry.CreateApplication("Superimpose")

The following lines set all the application parameters:
```

```
Superimpose.SetParameterString("inr", "QB_Toulouse_Ortho_PAN.tif")
Superimpose.SetParameterString("inm", "QB_Toulouse_Ortho_XS.tif")
Superimpose.SetParameterString("out", "SuperimposedXS_to_PAN.tif")

The following line execute the application
Superimpose.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.5 Image Filtering

### 6.5.1 Despeckle

Perform speckle noise reduction on SAR image.

#### Detailed description

This application reduce speckle noise. Four methods are available: Lee, Frost, GammaMAP and Kuan.

#### Parameters

This section describes in details the parameters available for this application. Table <sup>38</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Despeckle* .

---

<sup>38</sup> Table: Parameters table for Despeckle.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
filter	Choices	Choices
filter lee	<i>Choice</i>	Lee
filter frost	<i>Choice</i>	Frost
filter gammamap	<i>Choice</i>	GammaMap
filter kuan	<i>Choice</i>	Kuan
filter.lee.rad	Int	Int
filter.lee.nblooks	Float	Float
filter.frost.rad	Int	Int
filter.frost.deramp	Float	Float
filter.gammamap.rad	Int	Int
filter.gammamap.nblooks	Float	Float
filter.kuan.rad	Int	Int
filter.kuan.nblooks	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image.

**Output Image:** Output image.

**Available RAM (Mb):** Available memory for processing (in MB).

**speckle filtering method** Available choices are:

- **Lee:** Lee filter.
- **Radius:** Radius for lee filter.
- **nb looks:** Nb looks for lee filter.
- **Frost:** Frost filter.
- **Radius:** Radius for frost filter.
- **deramp:** Decrease factor declaration.
- **GammaMap:** GammaMap filter.
- **Radius:** Radius for GammaMAP filter.
- **nb looks:** Nb looks for GammaMAP filter.
- **Kuan:** Kuan filter.
- **Radius:** Radius for Kuan filter.
- **nb looks:** Nb looks for Kuan filter.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_Despeckle -in sar.tif -filter lee -filter.lee.rad 5 -out despeckle.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Despeckle application
Despeckle = otbApplication.Registry.CreateApplication("Despeckle")

The following lines set all the application parameters:
Despeckle.SetParameterString("in", "sar.tif")

Despeckle.SetParameterString("filter", "lee")

Despeckle.SetParameterInt("filter.lee.rad", 5)

Despeckle.SetParameterString("out", "despeckle.tif")

The following line execute the application
Despeckle.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.5.2 Dimensionality reduction

Perform Dimension reduction of the input image.

### Detailed description

Performs dimensionality reduction on input image. PCA,NA-PCA,MAF,ICA methods are available. It is also possible to compute the inverse transform to reconstruct the image. It is also possible to optionnaly export the transformation matrix to a text file.

### Parameters

This section describes in details the parameters available for this application. Table <sup>39</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DimensionalityReduction* .

<sup>39</sup> Table: Parameters table for Dimensionality reduction.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
rescale	Group	Group
rescale.outmin	Float	Float
rescale.outmax	Float	Float
outinv	Output image	Output image
method	Choices	Choices
method pca	<i>Choice</i>	PCA
method napca	<i>Choice</i>	NA-PCA
method maf	<i>Choice</i>	MAF
method ica	<i>Choice</i>	ICA
method.napca.radiusx	Int	Int
method.napca.radiusy	Int	Int
method.ica.iter	Int	Int
method.ica.mu	Float	Float
nbcomp	Int	Int
normalize	Boolean	Boolean
outmatrix	Output File name	Output File name
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to apply dimensionality reduction.

**Output Image:** output image. Components are ordered by decreasing eigenvalues.

**[Rescale Output.]**

- **Output min value:** Minimum value of the output image.
- **Output max value:** Maximum value of the output image.

**Inverse Output Image:** reconstruct output image.

**Algorithm:** Selection of the reduction dimension method. Available choices are:

- **PCA:** Principal Component Analysis.
- **NA-PCA:** Noise Adjusted Principal Component Analysis.
- **Set the x radius of the sliding window.**
- **Set the y radius of the sliding window.**
- **MAF:** Maximum Autocorrelation Factor.
- **ICA:** Independent Component Analysis.
- **number of iterations**
- **Give the increment weight of W in [0, 1]**

**Number of Components.:** Number of relevant components kept. By default all components are kept.

**Normalize.:** center AND reduce data before Dimensionality reduction.

**Transformation matrix output (text format):** Filename to store the transformation matrix (csv format).

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_DimensionalityReduction -in cupriteSubHsi.tif -out FilterOutput.tif -method pca
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the DimensionalityReduction application
DimensionalityReduction = otbApplication.Registry.CreateApplication("DimensionalityReduction")

The following lines set all the application parameters:
DimensionalityReduction.SetParameterString("in", "cupriteSubHsi.tif")

DimensionalityReduction.SetParameterString("out", "FilterOutput.tif")

DimensionalityReduction.SetParameterString("method", "pca")

The following line execute the application
DimensionalityReduction.ExecuteAndWriteOutput()
```

## Limitations

This application does not provide the inverse transform and the transformation matrix export for the MAF.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

“Kernel maximum autocorrelation factor and minimum noise fraction transformations,” IEEE Transactions on Image Processing, vol. 20, no. 3, pp. 612-624, (2011)

## 6.5.3 Exact Large-Scale Mean-Shift segmentation, step 1 (smoothing)

Perform mean shift filtering

### Detailed description

This application performs mean shift filtering (multi-threaded).

## Parameters

This section describes in details the parameters available for this application. Table <sup>40</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MeanShiftSmoothing*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
fout	Output image	Output image
foutpos	Output image	Output image
ram	Int	Int
spatialr	Int	Int
ranger	Float	Float
thres	Float	Float
maxiter	Int	Int
rangeramp	Float	Float
modesearch	Boolean	Boolean
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The input image.
- **Filtered output:** The filtered output image.
- **Spatial image:** The spatial image output. Spatial image output is a displacement map (pixel position after convergence).
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Spatial radius:** Spatial radius of the neighborhood.
- **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multi-spectral space.
- **Mode convergence threshold:** Algorithm iterative scheme will stop if mean-shift vector is below this threshold or if iteration number reached maximum number of iterations.
- **Maximum number of iterations:** Algorithm iterative scheme will stop if convergence hasn't been reached after the maximum number of iterations.
- **Range radius coefficient:** This coefficient makes dependent the ranger of the colorimetry of the filtered pixel :  
 $y = \text{rangeramp} * x + \text{ranger}$ .
- **Mode search.:** If activated pixel iterative convergence is stopped if the path crosses an already converged pixel. Be careful, with this option, the result will slightly depend on thread number.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_MeanShiftSmoothing -in maur_rgb.png -fout MeanShift_FilterOutput.tif -foutpos MeanShift_Spatial
```

To run this example from Python, use the following code snippet:

<sup>40</sup> Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 1 (smoothing).

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the MeanShiftSmoothing application
MeanShiftSmoothing = otbApplication.Registry.CreateApplication("MeanShiftSmoothing")

The following lines set all the application parameters:
MeanShiftSmoothing.SetParameterString("in", "maur_rgb.png")

MeanShiftSmoothing.SetParameterString("fout", "MeanShift_FilterOutput.tif")

MeanShiftSmoothing.SetParameterString("foutpos", "MeanShift_SpatialOutput.tif")

MeanShiftSmoothing.SetParameterInt("spatialr", 16)

MeanShiftSmoothing.SetParameterFloat("ranger", 16)

MeanShiftSmoothing.SetParameterFloat("thres", 0.1)

MeanShiftSmoothing.SetParameterInt("maxiter", 100)

The following line execute the application
MeanShiftSmoothing.ExecuteAndWriteOutput()
```

### Limitations

With mode search option, the result will slightly depend on thread number.

### Authors

This application has been written by OTB-Team.

## 6.5.4 Smoothing

Apply a smoothing filter to an image

### Detailed description

This application applies smoothing filter to an image. Either gaussian, mean, or anisotropic diffusion are available.

### Parameters

This section describes in details the parameters available for this application. Table <sup>41</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Smoothing* .

---

<sup>41</sup> Table: Parameters table for Smoothing.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
type	Choices	Choices
type mean	<i>Choice</i>	Mean
type gaussian	<i>Choice</i>	Gaussian
type anidif	<i>Choice</i>	Anisotropic Diffusion
type.mean.radius	Int	Int
type.gaussian.radius	Float	Float
type.anidif.timestep	Float	Float
type.anidif.nbiter	Int	Int
type.anidif.conductance	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image to smooth.

**Output Image:** Output smoothed image.

**Available RAM (Mb):** Available memory for processing (in MB).

**Smoothing Type:** Smoothing kernel to apply. Available choices are:

- **Mean**
- **Radius:** Mean radius (in pixels).
- **Gaussian**
- **Radius:** Gaussian radius (in pixels).
- **Anisotropic Diffusion**
- **Time Step:** Diffusion equation time step.
- **Nb Iterations:** Controls the sensitivity of the conductance term.
- **Conductance**

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Examples

### Example 1

Image smoothing using a mean filter. To run this example in command-line, use the following:

```
otbcli_Smoothing -in Romania_Extract.tif -out smoothedImage_mean.png uchar -type mean
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Smoothing application
Smoothing = otbApplication.Registry.CreateApplication("Smoothing")
```

```
The following lines set all the application parameters:
Smoothing.SetParameterString("in", "Romania_Extract.tif")

Smoothing.SetParameterString("out", "smoothedImage_mean.png")
Smoothing.SetParameterOutputImagePixelFormat("out", 1)

Smoothing.SetParameterString("type", "mean")

The following line execute the application
Smoothing.ExecuteAndWriteOutput()
```

### Example 2

Image smoothing using an anisotropic diffusion filter. To run this example in command-line, use the following:

```
otbcli_Smoothing -in Romania_Extract.tif -out smoothedImage_ani.png float -type anidif -type.anidif.t
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Smoothing application
Smoothing = otbApplication.Registry.CreateApplication("Smoothing")

The following lines set all the application parameters:
Smoothing.SetParameterString("in", "Romania_Extract.tif")

Smoothing.SetParameterString("out", "smoothedImage_ani.png")
Smoothing.SetParameterOutputImagePixelFormat("out", 6)

Smoothing.SetParameterString("type", "anidif")

Smoothing.SetParameterFloat("type.anidif.timestep", 0.1)

Smoothing.SetParameterInt("type.anidif.nbiter", 5)

Smoothing.SetParameterFloat("type.anidif.conductance", 1.5)

The following line execute the application
Smoothing.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.6 Feature Extraction

### 6.6.1 Binary Morphological Operation

Performs morphological operations on an input image channel

#### Detailed description

This application performs binary morphological operations on a mono band image

#### Parameters

This section describes in details the parameters available for this application. Table <sup>42</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BinaryMorphologicalOperation*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
channel	Int	Int
ram	Int	Int
structype	Choices	Choices
structype ball	<i>Choice</i>	Ball
structype cross	<i>Choice</i>	Cross
structype.ball.xradius	Int	Int
structype.ball.yradius	Int	Int
filter	Choices	Choices
filter dilate	<i>Choice</i>	Dilate
filter erode	<i>Choice</i>	Erode
filter opening	<i>Choice</i>	Opening
filter closing	<i>Choice</i>	Closing
filter.dilate.foreval	Float	Float
filter.dilate.backval	Float	Float
filter.erode.foreval	Float	Float
filter.erode.backval	Float	Float
filter.opening.foreval	Float	Float
filter.opening.backval	Float	Float
filter.closing.foreval	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to be filtered.

**Feature Output Image:** Output image containing the filtered output image.

**Selected Channel:** The selected channel index.

**Available RAM (Mb):** Available memory for processing (in MB).

**Structuring Element Type:** Choice of the structuring element type. Available choices are:

- **Ball**

<sup>42</sup> Table: Parameters table for Binary Morphological Operation.

- **The Structuring Element X Radius:** The Structuring Element X Radius.
- **The Structuring Element Y Radius:** The Structuring Element Y Radius.
- **Cross**

**Morphological Operation:** Choice of the morphological operation. Available choices are:

- **Dilate**
- **Foreground Value:** The Foreground Value.
- **Background Value:** The Background Value.
- **Erode**
- **Foreground Value:** The Foreground Value.
- **Background Value:** The Background Value.
- **Opening**
- **Foreground Value:** The Foreground Value.
- **Background Value:** The Background Value.
- **Closing**
- **Foreground Value:** The Foreground Value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_BinaryMorphologicalOperation -in qb_RoadExtract.tif -out opened.tif -channel 1 -structype.ball
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the BinaryMorphologicalOperation application
BinaryMorphologicalOperation = otbApplication.Registry.CreateApplication("BinaryMorphologicalOperati

The following lines set all the application parameters:
BinaryMorphologicalOperation.SetParameterString("in", "qb_RoadExtract.tif")

BinaryMorphologicalOperation.SetParameterString("out", "opened.tif")

BinaryMorphologicalOperation.SetParameterInt("channel", 1)

BinaryMorphologicalOperation.SetParameterInt("structype.ball.xradius", 5)

BinaryMorphologicalOperation.SetParameterInt("structype.ball.yradius", 5)

BinaryMorphologicalOperation.SetParameterString("filter", "erode")
```

```
The following line execute the application
BinaryMorphologicalOperation.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

itkBinaryDilateImageFilter, itkBinaryErodeImageFilter, itkBinaryMorphologicalOpeningImageFilter and itkBinaryMorphologicalClosingImageFilter classes

## 6.6.2 Compute Polyline Feature From Image

This application compute for each studied polyline, contained in the input VectorData, the chosen descriptors.

### Detailed description

The first step in the classifier fusion based validation is to compute, for each studied polyline, the chosen descriptors.

### Parameters

This section describes in details the parameters available for this application. Table <sup>43</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ComputePolylineFeatureFromImage* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
vd	Input vector data	Input vector data
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
expr	String	String
field	String	String
out	Output vector data	Output vector data
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** An image to compute the descriptors on.

**Vector Data:** Vector data containing the polylines where the features will be computed.

<sup>43</sup> Table: Parameters table for Compute Polyline Feature From Image.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Feature expression:** The feature formula ( $b1 < 0.3$ ) where b1 is the standard name of input image first band.

**Feature name:** The field name corresponding to the feature codename (NONDVI, ROADS...).

**Output Vector Data:** The output vector data containing polylines with a new field.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ComputePolylineFeatureFromImage -in NDVI.TIF -vd roads_ground_truth.shp -expr "(b1 > 0.4)" -f
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ComputePolylineFeatureFromImage application
ComputePolylineFeatureFromImage = otbApplication.Registry.CreateApplication("ComputePolylineFeatureFromImage")

The following lines set all the application parameters:
ComputePolylineFeatureFromImage.SetParameterString("in", "NDVI.TIF")

ComputePolylineFeatureFromImage.SetParameterString("vd", "roads_ground_truth.shp")

ComputePolylineFeatureFromImage.SetParameterString("expr", "(b1 > 0.4)")

ComputePolylineFeatureFromImage.SetParameterString("field", "NONDVI")

ComputePolylineFeatureFromImage.SetParameterString("out", "PolylineFeatureFromImage_LI_NONDVI_gt.shp")

The following line execute the application
ComputePolylineFeatureFromImage.ExecuteAndWriteOutput()
```

## Limitations

Since it does not rely on streaming process, take care of the size of input image before launching application.

## Authors

This application has been written by OTB-Team.

## 6.6.3 Fuzzy Model estimation

Estimate feature fuzzy model parameters using 2 vector data (ground truth samples and wrong samples).

### Detailed description

Estimate feature fuzzy model parameters using 2 vector data (ground truth samples and wrong samples).

### Parameters

This section describes in details the parameters available for this application. Table <sup>44</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DSFuzzyModelEstimation* .

Parameter Key	Parameter Type	Parameter Description
psin	Input vector data	Input vector data
nsin	Input vector data	Input vector data
belsup	String list	String list
plasup	String list	String list
cri	String	String
wgt	Float	Float
initmod	Input File name	Input File name
desclist	String list	String list
maxnbit	Int	Int
optobs	Boolean	Boolean
out	Output File name	Output File name
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Positive Vector Data:** Ground truth vector data for positive samples.
- **Input Negative Vector Data:** Ground truth vector data for negative samples.
- **Belief Support:** Dempster Shafer study hypothesis to compute belief.
- **Plausibility Support:** Dempster Shafer study hypothesis to compute plausibility.
- **Criterion:** Dempster Shafer criterion (by default (belief+plausibility)/2).
- **Weighting:** Coefficient between 0 and 1 to promote undetection or false detections (default 0.5).
- **initialization model:** Initialization model (xml file) to be used. If the xml initialization model is set, the descriptor list is not used (specified using the option -desclist).
- **Descriptor list:** List of the descriptors to be used in the model (must be specified to perform an automatic initialization).
- **Maximum number of iterations:** Maximum number of optimizer iteration (default 200).
- **Optimizer Observer:** Activate the optimizer observer.

---

<sup>44</sup> Table: Parameters table for Fuzzy Model estimation.

- **Output filename:** Output model file name (xml file) contains the optimal model to perform information fusion.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_DSfuzzyModelEstimation -psin cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp -nsin cdbTv
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the DSfuzzyModelEstimation application
DSfuzzyModelEstimation = otbApplication.Registry.CreateApplication("DSfuzzyModelEstimation")

The following lines set all the application parameters:
DSfuzzyModelEstimation.SetParameterString("psin", "cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt")
DSfuzzyModelEstimation.SetParameterString("nsin", "cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_wr")
DSfuzzyModelEstimation.SetParameterStringList("belsup", ["ROADSA"])
DSfuzzyModelEstimation.SetParameterStringList("plasup", ["NONDVI", "ROADSA", "NOBUIL"])
DSfuzzyModelEstimation.SetParameterString("initmod", "Dempster-Shafer/DSfuzzyModel_Init.xml")
DSfuzzyModelEstimation.SetParameterInt("maxnbit", 4)
DSfuzzyModelEstimation.SetParameterString("optobs", "1")
DSfuzzyModelEstimation.SetParameterString("out", "DSfuzzyModelEstimation.xml")

The following line execute the application
DSfuzzyModelEstimation.ExecuteAndWriteOutput()
```

## Limitations

None.

## Authors

This application has been written by OTB-Team.

## 6.6.4 Edge Feature Extraction

Computes edge features on every pixel of the input image selected channel

## Detailed description

This application computes edge features on a mono band image

## Parameters

This section describes in details the parameters available for this application. Table <sup>45</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *EdgeExtraction* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
channel	Int	Int
ram	Int	Int
filter	Choices	Choices
filter gradient	<i>Choice</i>	Gradient
filter sobel	<i>Choice</i>	Sobel
filter touzi	<i>Choice</i>	Touzi
filter.touzi.xradius	Int	Int
filter.touzi.yradius	Int	Int
out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to compute the features on.

**Selected Channel:** The selected channel index.

**Available RAM (Mb):** Available memory for processing (in MB).

**Edge feature:** Choice of edge feature. Available choices are:

- **Gradient**
- **Sobel**
- **Touzi**
- **The X Radius**
- **The Y Radius**

**Feature Output Image:** Output image containing the edge features.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_EdgeExtraction -in qb_RoadExtract.tif -channel 1 -out Edges.tif
```

To run this example from Python, use the following code snippet:

---

<sup>45</sup> Table: Parameters table for Edge Feature Extraction.

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the EdgeExtraction application
EdgeExtraction = otbApplication.Registry.CreateApplication("EdgeExtraction")

The following lines set all the application parameters:
EdgeExtraction.SetParameterString("in", "qb_RoadExtract.tif")

EdgeExtraction.SetParameterInt("channel", 1)

EdgeExtraction.SetParameterString("out", "Edges.tif")

The following line execute the application
EdgeExtraction.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

otb class

## 6.6.5 Grayscale Morphological Operation

Performs morphological operations on a grayscale input image

### Detailed description

This application performs grayscale morphological operations on a mono band image

### Parameters

This section describes in details the parameters available for this application. Table <sup>46</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GrayScaleMorphologicalOperation* .

---

<sup>46</sup> Table: Parameters table for Grayscale Morphological Operation.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
channel	Int	Int
ram	Int	Int
structype	Choices	Choices
structype ball	<i>Choice</i>	Ball
structype cross	<i>Choice</i>	Cross
structype.ball.xradius	Int	Int
structype.ball.yradius	Int	Int
filter	Choices	Choices
filter dilate	<i>Choice</i>	Dilate
filter erode	<i>Choice</i>	Erode
filter opening	<i>Choice</i>	Opening
filter closing	<i>Choice</i>	Closing
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to be filtered.

**Feature Output Image:** Output image containing the filtered output image.

**Selected Channel:** The selected channel index.

**Available RAM (Mb):** Available memory for processing (in MB).

**Structuring Element Type:** Choice of the structuring element type. Available choices are:

- **Ball**
- **The Structuring Element X Radius:** The Structuring Element X Radius.
- **The Structuring Element Y Radius:** The Structuring Element Y Radius.
- **Cross**

**Morphological Operation:** Choice of the morphological operation. Available choices are:

- **Dilate**
- **Erode**
- **Opening**
- **Closing**

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_GrayScaleMorphologicalOperation -in qb_RoadExtract.tif -out opened.tif -channel 1 -structype.Ball
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication
```

```

The following line creates an instance of the GrayScaleMorphologicalOperation application
GrayScaleMorphologicalOperation = otbApplication.Registry.CreateApplication("GrayScaleMorphologicalO

The following lines set all the application parameters:
GrayScaleMorphologicalOperation.SetParameterString("in", "qb_RoadExtract.tif")

GrayScaleMorphologicalOperation.SetParameterString("out", "opened.tif")

GrayScaleMorphologicalOperation.SetParameterInt("channel", 1)

GrayScaleMorphologicalOperation.SetParameterInt("structype.ball.xradius", 5)

GrayScaleMorphologicalOperation.SetParameterInt("structype.ball.yradius", 5)

GrayScaleMorphologicalOperation.SetParameterString("filter", "erode")

The following line execute the application
GrayScaleMorphologicalOperation.ExecuteAndWriteOutput()

```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

`itkGrayscaleDilateImageFilter`, `itkGrayscaleErodeImageFilter`, `itkGrayscaleMorphologicalOpeningImageFilter` and `itkGrayscaleMorphologicalClosingImageFilter` classes

## 6.6.6 Haralick Texture Extraction

Computes textures on every pixel of the input image selected channel

### Detailed description

This application computes Haralick, advanced and higher order textures on a mono band image

### Parameters

This section describes in details the parameters available for this application. Table <sup>47</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *HaralickTextureExtraction*.

<sup>47</sup> Table: Parameters table for Haralick Texture Extraction.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
channel	Int	Int
ram	Int	Int
parameters	Group	Group
parameters.xrad	Int	Int
parameters.yrad	Int	Int
parameters.xoff	Int	Int
parameters.yoff	Int	Int
parameters.min	Float	Float
parameters.max	Float	Float
parameters.nbbin	Int	Int
texture	Choices	Choices
texture simple	<i>Choice</i>	Simple Haralick Texture Features
texture advanced	<i>Choice</i>	Advanced Texture Features
texture higher	<i>Choice</i>	Higher Order Texture Features
out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to compute the features on.

**Selected Channel:** The selected channel index.

**Available RAM (Mb):** Available memory for processing (in MB).

**[Texture feature parameters]:** This group of parameters allows one to define texture parameters.

- **X Radius:** X Radius.
- **Y Radius:** Y Radius.
- **X Offset:** X Offset.
- **Y Offset:** Y Offset.
- **Image Minimum:** Image Minimum.
- **Image Maximum:** Image Maximum.
- **Histogram number of bin:** Histogram number of bin.

**Texture Set Selection:** Choice of The Texture Set. Available choices are:

- **Simple Haralick Texture Features:** This group of parameters defines the 8 local Haralick texture feature output image. The image channels are: Energy, Entropy, Correlation, Inverse Difference Moment, Inertia, Cluster Shade, Cluster Prominence and Haralick Correlation.
- **Advanced Texture Features:** This group of parameters defines the 9 advanced texture feature output image. The image channels are: Mean, Variance, Sum Average, Sum Variance, Sum Entropy, Difference of Entropies, Difference of Variances, IC1 and IC2.
- **Higher Order Texture Features:** This group of parameters defines the 11 higher order texture feature output image. The image channels are: Short Run Emphasis, Long Run Emphasis, Grey-Level Nonuniformity, Run Length Nonuniformity, Run Percentage, Low Grey-Level Run Emphasis, High Grey-Level Run Emphasis, Short Run Low Grey-Level Emphasis, Short Run High Grey-Level Emphasis, Long Run Low Grey-Level Emphasis and Long Run High Grey-Level Emphasis.

**Output Image:** Output image containing the selected texture features.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_HaralickTextureExtraction -in qb_RoadExtract.tif -channel 2 -parameters.xrad 3 -parameters.yrad 3
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the HaralickTextureExtraction application
HaralickTextureExtraction = otbApplication.Registry.CreateApplication("HaralickTextureExtraction")

The following lines set all the application parameters:
HaralickTextureExtraction.SetParameterString("in", "qb_RoadExtract.tif")

HaralickTextureExtraction.SetParameterInt("channel", 2)

HaralickTextureExtraction.SetParameterInt("parameters.xrad", 3)

HaralickTextureExtraction.SetParameterInt("parameters.yrad", 3)

HaralickTextureExtraction.SetParameterString("texture", "simple")

HaralickTextureExtraction.SetParameterString("out", "HaralickTextures.tif")

The following line execute the application
HaralickTextureExtraction.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

`otbScalarImageToTexturesFilter`, `otbScalarImageToAdvancedTexturesFilter` and `otbScalarImageToHigherOrderTexturesFilter` classes

## 6.6.7 Homologous Points Extraction

Compute homologous points between images using keypoints

## Detailed description

This application allows computing homologous points between images using keypoints. SIFT or SURF keypoints can be used and the band on which keypoints are computed can be set independently for both images. The application offers two modes : the first is the full mode where keypoints are extracted from the full extent of both images (please note that in this mode large image file are not supported). The second mode, called geobins, allows one to set-up spatial binning to get fewer points spread across the entire image. In this mode, the corresponding spatial bin in the second image is estimated using geographical transform or sensor modelling, and is padded according to the user defined precision. Last, in both modes the application can filter matches whose colocalisation in first image exceed this precision. The elevation parameters are to deal more precisely with sensor modelling in case of sensor geometry data. The outvector option allows creating a vector file with segments corresponding to the localisation error between the matches. It can be useful to assess the precision of a registration for instance. The vector file is always reprojected to EPSG:4326 to allow display in a GIS. This is done via reprojection or by applying the image sensor models.

## Parameters

This section describes in details the parameters available for this application. Table <sup>48</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *HomologousPointsExtraction* .

---

<sup>48</sup> Table: Parameters table for Homologous Points Extraction.

Parameter Key	Parameter Type	Parameter Description
in1	Input image	Input image
band1	Int	Int
in2	Input image	Input image
band2	Int	Int
algorithm	Choices	Choices
algorithm surf	<i>Choice</i>	SURF algorithm
algorithm sift	<i>Choice</i>	SIFT algorithm
threshold	Float	Float
backmatching	Boolean	Boolean
mode	Choices	Choices
mode full	<i>Choice</i>	Extract and match all keypoints (no streaming)
mode geobins	<i>Choice</i>	Search keypoints in small spatial bins regularly spread across first image
mode.geobins.binsize	Int	Int
mode.geobins.binsizey	Int	Int
mode.geobins.binstep	Int	Int
mode.geobins.binstepy	Int	Int
mode.geobins.margin	Int	Int
precision	Float	Float
mfilter	Boolean	Boolean
2wgs84	Boolean	Boolean
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
out	Output File name	Output File name
outvector	Output File name	Output File name
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image 1:** First input image.

**Input band 1:** Index of the band from input image 1 to use for keypoints extraction.

**Input Image 2:** Second input image.

**Input band 2:** Index of the band from input image 1 to use for keypoints extraction.

**Keypoints detection algorithm:** Choice of the detection algorithm to use. Available choices are:

- **SURF algorithm**
- **SIFT algorithm**

**Distance threshold for matching:** The distance threshold for matching.

**Use back-matching to filter matches.:** If set to true, matches should be consistent in both ways.

**Keypoints search mode** Available choices are:

- **Extract and match all keypoints (no streaming):** Extract and match all keypoints, loading both images entirely into memory.
- **Search keypoints in small spatial bins regularly spread across first image:** This method allows retrieving a set of tie points regularly spread across image 1. Corresponding bins in image 2 are retrieved using sensor and

geographical information if available. The first bin position takes into account the margin parameter. Bins are cropped to the largest image region shrunk by the margin parameter for both in1 and in2 images.

- **Size of bin:** Radius of the spatial bin in pixels.
- **Size of bin (y direction):** Radius of the spatial bin in pixels (y direction). If not set, the mode.geobins.binsize value is used.
- **Steps between bins:** Steps between bins in pixels.
- **Steps between bins (y direction):** Steps between bins in pixels (y direction). If not set, the mode.geobins.binstep value is used.
- **Margin from image border to start/end bins (in pixels):** Margin from image border to start/end bins (in pixels).

**Estimated precision of the colocalisation function (in pixels):** Estimated precision of the colocalisation function in pixels.

**Filter points according to geographical or sensor based colocalisation:** If enabled, this option allows one to filter matches according to colocalisation from sensor or geographical information, using the given tolerancy expressed in pixels.

**If enabled, points from second image will be exported in WGS84**

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Output file with tie points:** File containing the list of tie points.

**Output vector file with tie points:** File containing segments representing matches .

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_HomologousPointsExtraction -in1 sensor_stereo_left.tif -in2 sensor_stereo_right.tif -mode full
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python
Import the otb applications package
import otbApplication
```

```

The following line creates an instance of the HomologousPointsExtraction application
HomologousPointsExtraction = otbApplication.Registry.CreateApplication("HomologousPointsExtraction")

The following lines set all the application parameters:
HomologousPointsExtraction.SetParameterString("in1", "sensor_stereo_left.tif")

HomologousPointsExtraction.SetParameterString("in2", "sensor_stereo_right.tif")

HomologousPointsExtraction.SetParameterString("mode", "full")

HomologousPointsExtraction.SetParameterString("out", "homologous.txt")

The following line execute the application
HomologousPointsExtraction.ExecuteAndWriteOutput()

```

### Limitations

Full mode does not handle large images.

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

RefineSensorModel

## 6.6.8 Line segment detection

Detect line segments in raster

### Detailed description

**This application detects locally straight contours in a image. It is based on Burns, Hanson, and Riseman method and use an a c**

The given approach computes gradient and level lines of the image and detects aligned points in line support region. The application allows exporting the detected lines in a vector data.

### Parameters

This section describes in details the parameters available for this application. Table <sup>49</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LineSegmentDetection* .

<sup>49</sup> Table: Parameters table for Line segment detection.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output vector data	Output vector data
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
norescale	Boolean	Boolean
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image on which lines will be detected.

**Output Detected lines:** Output detected line segments (vector data).

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occur if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**No rescaling in [0, 255]:** By default, the input image amplitude is rescaled between [0,255]. Turn on this parameter to skip rescaling.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_LineSegmentDetection -in QB_Suburb.png -out LineSegmentDetection.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the LineSegmentDetection application
LineSegmentDetection = otbApplication.Registry.CreateApplication("LineSegmentDetection")

The following lines set all the application parameters:
LineSegmentDetection.SetParameterString("in", "QB_Suburb.png")
```

```

LineSegmentDetection.SetParameterString("out", "LineSegmentDetection.shp")

The following line execute the application
LineSegmentDetection.ExecuteAndWriteOutput()

```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

On Line demonstration of the LSD algorithm is available here: [http://www.ipol.im/pub/algo/gjmr\\_line\\_segment\\_detector/](http://www.ipol.im/pub/algo/gjmr_line_segment_detector/)

## 6.6.9 Local Statistic Extraction

Computes local statistical moments on every pixel in the selected channel of the input image

### Detailed description

This application computes the 4 local statistical moments on every pixel in the selected channel of the input image, over a specified neighborhood. The output image is multi band with one statistical moment (feature) per band. Thus, the 4 output features are the Mean, the Variance, the Skewness and the Kurtosis. They are provided in this exact order in the output image.

### Parameters

This section describes in details the parameters available for this application. Table <sup>50</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LocalStatisticExtraction*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
channel	Int	Int
ram	Int	Int
radius	Int	Int
out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The input image to compute the features on.
- **Selected Channel:** The selected channel index.

<sup>50</sup> Table: Parameters table for Local Statistic Extraction.

- **Available RAM (Mb):** Available memory for processing (in MB).
- **Neighborhood radius:** The computational window radius.
- **Feature Output Image:** Output image containing the local statistical moments.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_LocalStatisticExtraction -in qb_RoadExtract.tif -channel 1 -radius 3 -out Statistics.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the LocalStatisticExtraction application
LocalStatisticExtraction = otbApplication.Registry.CreateApplication("LocalStatisticExtraction")

The following lines set all the application parameters:
LocalStatisticExtraction.SetParameterString("in", "qb_RoadExtract.tif")

LocalStatisticExtraction.SetParameterInt("channel", 1)

LocalStatisticExtraction.SetParameterInt("radius", 3)

LocalStatisticExtraction.SetParameterString("out", "Statistics.tif")

The following line execute the application
LocalStatisticExtraction.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

`otbRadiometricMomentsImageFunction` class

## 6.6.10 Multivariate alteration detector

Multivariate Alteration Detector

## Detailed description

This application detects change between two given images.

## Parameters

This section describes in details the parameters available for this application. Table <sup>51</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MultivariateAlterationDetector* .

Parameter Key	Parameter Type	Parameter Description
in1	Input image	Input image
in2	Input image	Input image
out	Output image	Output image
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image 1:** Image which describe initial state of the scene.
- **Input Image 2:** Image which describe scene after perturbations.
- **Change Map:** Image of detected changes.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_MultivariateAlterationDetector -in1 Spot5-Gloucester-before.tif -in2 Spot5-Gloucester-after.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the MultivariateAlterationDetector application
MultivariateAlterationDetector = otbApplication.Registry.CreateApplication("MultivariateAlterationDetector")

The following lines set all the application parameters:
MultivariateAlterationDetector.SetParameterString("in1", "Spot5-Gloucester-before.tif")

MultivariateAlterationDetector.SetParameterString("in2", "Spot5-Gloucester-after.tif")

MultivariateAlterationDetector.SetParameterString("out", "detectedChangeImage.tif")

The following line execute the application
MultivariateAlterationDetector.ExecuteAndWriteOutput()
```

<sup>51</sup> Table: Parameters table for Multivariate alteration detector.

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

### **This filter implements the Multivariate Alteration Detector, based on the following work:**

1. (a) Nielsen and K. Conradsen, Multivariate alteration detection (mad) in multispectral, bi-temporal image data: a new approach to change detection studies, *Remote Sens. Environ.*, vol. 64, pp. 1-19, (1998)

Multivariate Alteration Detector takes two images as inputs and produce a set of N change maps as a VectorImage (where N is the maximum of number of bands in first and second image) with the following properties: - Change maps are differences of a pair of linear combinations of bands from image 1 and bands from image 2 chosen to maximize the correlation. - Each change map is orthogonal to the others.

This is a statistical method which can handle different modalities and even different bands and number of bands between images.

If numbers of bands in image 1 and 2 are equal, then change maps are sorted by increasing correlation. If number of bands is different, the change maps are sorted by decreasing correlation.

The GetV1() and GetV2() methods allow retrieving the linear combinations used to generate the Mad change maps as a vnl\_matrix of double, and the GetRho() method allows retrieving the correlation associated to each Mad change maps as a vnl\_vector.

This filter has been implemented from the Matlab code kindly made available by the authors here: <http://www2.imm.dtu.dk/~aa/software.html>

Both cases (same and different number of bands) have been validated by comparing the output image to the output produced by the Matlab code, and the reference images for testing have been generated from the Matlab code using Octave.

## 6.6.11 Radiometric Indices

Compute radiometric indices.

### Detailed description

This application computes radiometric indices using the relevant channels of the input image. The output is a multi band image into which each channel is one of the selected indices.

### Parameters

This section describes in details the parameters available for this application. Table <sup>52</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is

---

<sup>52</sup> Table: Parameters table for Radiometric Indices.

*RadiometricIndices* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
channels	Group	Group
channels.blue	Int	Int
channels.green	Int	Int
channels.red	Int	Int
channels.nir	Int	Int
channels.mir	Int	Int
list	List	List
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** Input image.

**Output Image:** Radiometric indices output image.

**Available RAM (Mb):** Available memory for processing (in MB).

**[Channels selection]:** Channels selection.

- **Blue Channel:** Blue channel index.
- **Green Channel:** Green channel index.
- **Red Channel:** Red channel index.
- **NIR Channel:** NIR channel index.
- **Mir Channel:** Mir channel index.

**Available Radiometric Indices:** List of available radiometric indices with their relevant channels in brackets: Vegetation:NDVI - Normalized difference vegetation index (Red, NIR) Vegetation:TNDVI - Transformed normalized difference vegetation index (Red, NIR) Vegetation:RVI - Ratio vegetation index (Red, NIR) Vegetation:SAVI - Soil adjusted vegetation index (Red, NIR) Vegetation:TSAVI - Transformed soil adjusted vegetation index (Red, NIR) Vegetation:MSAVI - Modified soil adjusted vegetation index (Red, NIR) Vegetation:MSAVI2 - Modified soil adjusted vegetation index 2 (Red, NIR) Vegetation:GEMI - Global environment monitoring index (Red, NIR) Vegetation:IPVI - Infrared percentage vegetation index (Red, NIR) Water:NDWI - Normalized difference water index (Gao 1996) (NIR, MIR) Water:NDWI2 - Normalized difference water index (Mc Feeters 1996) (Green, NIR) Water:MNDWI - Modified normalized difference water index (Xu 2006) (Green, MIR) Water:NDPI - Normalized difference pond index (Lacaux et al.) (MIR, Green) Water:NDTI - Normalized difference turbidity index (Lacaux et al.) (Red, Green) Soil:RI - Redness index (Red, Green) Soil:CI - Color index (Red, Green) Soil:BI - Brightness index (Red, Green) Soil:BI2 - Brightness index 2 (NIR, Red, Green).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_RadiometricIndices -in qb_RoadExtract.tif -list Vegetation:NDVI Vegetation:RVI Vegetation:IPVI
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the RadiometricIndices application
RadiometricIndices = otbApplication.Registry.CreateApplication("RadiometricIndices")

The following lines set all the application parameters:
RadiometricIndices.SetParameterString("in", "qb_RoadExtract.tif")

The following line execute the application
RadiometricIndices.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

otbVegetationIndicesFunctor, otbWaterIndicesFunctor and otbSoilIndicesFunctor classes

## 6.6.12 SFS Texture Extraction

Computes Structural Feature Set textures on every pixel of the input image selected channel

### Detailed description

This application computes SFS textures on a mono band image

### Parameters

This section describes in details the parameters available for this application. Table <sup>53</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SFSTextureExtraction* .

---

<sup>53</sup> Table: Parameters table for SFS Texture Extraction.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
channel	Int	Int
ram	Int	Int
parameters	Group	Group
parameters.spethre	Float	Float
parameters.spathre	Int	Int
parameters.nmdir	Int	Int
parameters.alpha	Float	Float
parameters.maxcons	Int	Int
out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to compute the features on.

**Selected Channel:** The selected channel index.

**Available RAM (Mb):** Available memory for processing (in MB).

**[Texture feature parameters]:** This group of parameters allows one to define SFS texture parameters. The available texture features are SFS'Length, SFS'Width, SFS'PSI, SFS'W-Mean, SFS'Ratio and SFS'SD. They are provided in this exact order in the output image.

- **Spectral Threshold:** Spectral Threshold.
- **Spatial Threshold:** Spatial Threshold.
- **Number of Direction:** Number of Direction.
- **Alpha:** Alpha.
- **Ratio Maximum Consideration Number:** Ratio Maximum Consideration Number.

**Feature Output Image:** Output image containing the SFS texture features.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SFSTextureExtraction -in qb_RoadExtract.tif -channel 1 -parameters.spethre 50.0 -parameters.sp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SFSTextureExtraction application
SFSTextureExtraction = otbApplication.Registry.CreateApplication("SFSTextureExtraction")

The following lines set all the application parameters:
SFSTextureExtraction.SetParameterString("in", "qb_RoadExtract.tif")

SFSTextureExtraction.SetParameterInt("channel", 1)
```

```
SFSTextureExtraction.SetParameterFloat("parameters.spethre", 50.0)
SFSTextureExtraction.SetParameterInt("parameters.spathre", 100)
SFSTextureExtraction.SetParameterString("out", "SFSTextures.tif")

The following line execute the application
SFSTextureExtraction.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

otbSFSTexturesImageFilter class

## 6.6.13 Vector Data validation

Vector data validation based on the fusion of features using Dempster-Shafer evidence theory framework.

### Detailed description

This application validates or unvalidate the studied samples using the Dempster-Shafer theory.

### Parameters

This section describes in details the parameters available for this application. Table <sup>54</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataDSValidation* .

Parameter Key	Parameter Type	Parameter Description
in	Input vector data	Input vector data
descmod	Input File name	Input File name
belsup	String list	String list
plasup	String list	String list
cri	String	String
thd	Float	Float
out	Output vector data	Output vector data
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Vector Data:** Input vector data to validate.

---

<sup>54</sup> Table: Parameters table for Vector Data validation.

- **Descriptors model filename:** Fuzzy descriptors model (xml file).
- **Belief Support:** Dempster Shafer study hypothesis to compute belief.
- **Plausibility Support:** Dempster Shafer study hypothesis to compute plausibility.
- **Criterion:** Dempster Shafer criterion (by default (belief+plausibility)/2).
- **Criterion threshold:** Criterion threshold (default 0.5).
- **Output Vector Data:** Output VectorData containing only the validated samples.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_VectorDataDSValidation -in cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp -belsup cdbT
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the VectorDataDSValidation application
VectorDataDSValidation = otbApplication.Registry.CreateApplication("VectorDataDSValidation")

The following lines set all the application parameters:
VectorDataDSValidation.SetParameterString("in", "cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp")
VectorDataDSValidation.SetParameterStringList("belsup", ['cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp'])
VectorDataDSValidation.SetParameterString("descmod", "DSFuzzyModel.xml")
VectorDataDSValidation.SetParameterString("out", "VectorDataDSValidation.shp")

The following line execute the application
VectorDataDSValidation.ExecuteAndWriteOutput()
```

## Limitations

None.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

[http://en.wikipedia.org/wiki/Dempster-Shafer\\_theory](http://en.wikipedia.org/wiki/Dempster-Shafer_theory)

## 6.7 Stereo

### 6.7.1 Pixel-wise Block-Matching

Performs block-matching to estimate pixel-wise disparities between two images

#### Detailed description

This application allows one to performs block-matching to estimate pixel-wise disparities between two images. One must chose block-matching method and input masks (related to the left and right input image) of pixels for which the disparity should be investigated. Additionally, two criteria can be optionally used to disable disparity investigation for some pixel: a no-data value, and a threshold on the local variance. This allows one to speed-up computation by avoiding to investigate disparities that will not be reliable anyway. For efficiency reasons, if the optimal metric values image is desired, it will be concatenated to the output image (which will then have three bands : horizontal disparity, vertical disparity and metric value). One can split these images afterward.

#### Parameters

This section describes in details the parameters available for this application. Table <sup>55</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BlockMatching*.

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.inleft	Input image	Input image
io.inright	Input image	Input image
io.out	Output image	Output image
io.outmask	Output image	Output image
io.outmetric	Boolean	Boolean
mask	Group	Group
mask.inleft	Input image	Input image
mask.inright	Input image	Input image
mask.nodata	Float	Float
mask.variancet	Float	Float
bm	Group	Group
bm.metric	Choices	Choices
bm.metric ssd	<i>Choice</i>	Sum of Squared Distances
bm.metric ncc	<i>Choice</i>	Normalized Cross-Correlation
bm.metric lp	<i>Choice</i>	Lp pseudo-norm
bm.metric.lp.p	Float	Float
bm.radius	Int	Int
bm.minhd	Int	Int
bm.maxhd	Int	Int
bm.minvd	Int	Int
bm.maxvd	Int	Int
bm.subpixel	Choices	Choices
bm.subpixel none	<i>Choice</i>	None
bm.subpixel parabolic	<i>Choice</i>	Parabolic

Continued on next page

<sup>55</sup> Table: Parameters table for Pixel-wise Block-Matching.

Table 6.5 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
bm.subpixel triangular	<i>Choice</i>	Triangular
bm.subpixel dichotomy	<i>Choice</i>	Dichotomy
bm.step	Int	Int
bm.startx	Int	Int
bm.starty	Int	Int
bm.medianfilter	Group	Group
bm.medianfilter.radius	Int	Int
bm.medianfilter.incoherence	Float	Float
bm.initdisp	Choices	Choices
bm.initdisp none	<i>Choice</i>	None
bm.initdisp uniform	<i>Choice</i>	Uniform initial disparity
bm.initdisp maps	<i>Choice</i>	Initial disparity maps
bm.initdisp.uniform.hdisp	Int	Int
bm.initdisp.uniform.vdisp	Int	Int
bm.initdisp.uniform.hrad	Int	Int
bm.initdisp.uniform.vrad	Int	Int
bm.initdisp.maps.hmap	Input image	Input image
bm.initdisp.maps.vmap	Input image	Input image
bm.initdisp.maps.hrad	Int	Int
bm.initdisp.maps.vrad	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting the input and output images.

- **Left input image:** The left input image (reference).
- **Right input image:** The right input (secondary).
- **The output disparity map:** An image containing the estimated disparities as well as the metric values if the option is used.
- **The output mask corresponding to all criterions:** A mask image corresponding to all criterions (see masking parameters). Only required if variance threshold or nodata criterions are set.
- **Output optimal metric values as well:** If used, the output image will have a second component with metric optimal values.

**[Image masking parameters]:** This group of parameters allows determining the masking parameters to prevent disparities estimation for some pixels of the left image.

- **Discard left pixels from mask image:** This parameter allows providing a custom mask for the left image. Block matching will be only perform on pixels inside the mask.
- **Discard right pixels from mask image:** This parameter allows providing a custom mask for the right image. Block matching will be perform only on pixels inside the mask.
- **Discard pixels with no-data value:** This parameter allows discarding pixels whose value is equal to the user-defined no-data value.
- **Discard pixels with low local variance:** This parameter allows discarding pixels whose local variance is too small (the size of the neighborhood is given by the radius parameter).

**[Block matching parameters]:** This group of parameters allow tuning the block-matching behaviour.

- **Block-matching metric** Available choices are:

- **Sum of Squared Distances:** Sum of squared distances between pixels value in the metric window.
- **Normalized Cross-Correlation:** Normalized Cross-Correlation between the left and right windows.
- **Lp pseudo-norm:** Lp pseudo-norm between the left and right windows.
- **p value:** Value of the p parameter in Lp pseudo-norm (must be positive).
- **Radius of blocks:** The radius (in pixels) of blocks in Block-Matching.
- **Minimum horizontal disparity:** Minimum horizontal disparity to explore (can be negative).
- **Maximum horizontal disparity:** Maximum horizontal disparity to explore (can be negative).
- **Minimum vertical disparity:** Minimum vertical disparity to explore (can be negative).
- **Maximum vertical disparity:** Maximum vertical disparity to explore (can be negative).
- **Sub-pixel interpolation:** Estimate disparities with sub-pixel precision. Available choices are:
  - **None:** No sub-pixel .
  - **Parabolic:** Parabolic fit.
  - **Triangular:** Triangular fit.
  - **Dichotomy:** Dichotomic search.
- **Computation step:** Location step between computed disparities.
- **X start index:** X start index of the subsampled grid (wrt the input image grid).
- **Y start index:** Y start index of the subsampled grid (wrt the input image grid).
- **Median filtering:** Use a median filter to get a smooth disparity map.
- **Radius:** Radius for median filter.
- **Incoherence threshold:** Incoherence threshold between original and filtered disparity.
- **Initial disparities** Available choices are:
  - **None:** No initial disparity used.
  - **Uniform initial disparity:** Use an uniform initial disparity estimate.
  - **Horizontal initial disparity:** Value of the uniform horizontal disparity initial estimate (in pixels).
  - **Vertical initial disparity:** Value of the uniform vertical disparity initial estimate (in pixels).
  - **Horizontal exploration radius:** Horizontal exploration radius around the initial disparity estimate (in pixels).
  - **Vertical exploration radius:** Vertical exploration radius around the initial disparity estimate (in pixels).
  - **Initial disparity maps:** Use initial disparity maps.
  - **Horizontal initial disparity map:** Map of the initial horizontal disparities.
  - **Vertical initial disparity map:** Map of the initial vertical disparities.
  - **Horizontal exploration radius:** Horizontal exploration radius around the initial disparity estimate (in pixels).
  - **Vertical exploration radius:** Vertical exploration radius around the initial disparity estimate (in pixels).

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_BlockMatching -io.inleft StereoFixed.png -io.inright StereoMoving.png -bm.minhd -10 -bm.maxhd
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the BlockMatching application
BlockMatching = otbApplication.Registry.CreateApplication("BlockMatching")

The following lines set all the application parameters:
BlockMatching.SetParameterString("io.inleft", "StereoFixed.png")

BlockMatching.SetParameterString("io.inright", "StereoMoving.png")

BlockMatching.SetParameterInt("bm.minhd", -10)

BlockMatching.SetParameterInt("bm.maxhd", 10)

BlockMatching.SetParameterFloat("mask.variancet", 10)

BlockMatching.SetParameterString("io.out", "MyDisparity.tif")

The following line execute the application
BlockMatching.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

otbStereoRectificationGridGenerator

## 6.7.2 Disparity map to elevation map

Projects a disparity map into a regular elevation map

### Detailed description

This application uses a disparity map computed from a stereo image pair to produce an elevation map on the ground area covered by the stereo pair. The needed inputs are : the disparity map, the stereo pair (in original geometry) and the epipolar deformation grids. These grids have to link the original geometry (stereo pair) and the epipolar geometry (disparity map).

### Parameters

This section describes in details the parameters available for this application. Table <sup>56</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DisparityMapToElevationMap* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.in	Input image	Input image
io.left	Input image	Input image
io.right	Input image	Input image
io.lgrid	Input image	Input image
io.rgrid	Input image	Input image
io.out	Output image	Output image
io.mask	Input image	Input image
step	Float	Float
hmin	Float	Float
hmax	Float	Float
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows one to set input images, output images and grids.

- **Input disparity map:** The input disparity map (horizontal disparity in first band, vertical in second).
- **Left sensor image:** Left image in original (sensor) geometry.
- **Right sensor image:** Right image in original (sensor) geometry.
- **Left Grid:** Left epipolar grid (deformation grid between sensor et disparity spaces).
- **Right Grid:** Right epipolar grid (deformation grid between sensor et disparity spaces).
- **Output elevation map:** Output elevation map in ground projection.
- **Disparity mask:** Masked disparity cells won't be projected.

**DEM step:** Spacing of the output elevation map (in meters).

**Minimum elevation expected:** Minimum elevation expected (in meters).

<sup>56</sup> Table: Parameters table for Disparity map to elevation map.

**Maximum elevation expected:** Maximum elevation expected (in meters).

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_DisparityMapToElevationMap -io.in disparity.tif -io.left sensor_left.tif -io.right sensor_right.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the DisparityMapToElevationMap application
DisparityMapToElevationMap = otbApplication.Registry.CreateApplication("DisparityMapToElevationMap")

The following lines set all the application parameters:
DisparityMapToElevationMap.SetParameterString("io.in", "disparity.tif")

DisparityMapToElevationMap.SetParameterString("io.left", "sensor_left.tif")

DisparityMapToElevationMap.SetParameterString("io.right", "sensor_right.tif")

DisparityMapToElevationMap.SetParameterString("io.lgrid", "grid_epi_left.tif")

DisparityMapToElevationMap.SetParameterString("io.rgrid", "grid_epi_right.tif")

DisparityMapToElevationMap.SetParameterString("io.out", "dem.tif")

The following line execute the application
DisparityMapToElevationMap.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

`otbStereoRectificationGridGenerator` `otbBlockMatching`

## 6.7.3 Fine Registration

Estimate disparity map between two images.

### Detailed description

Estimate disparity map between two images. Output image contain x offset, y offset and metric value.

### Parameters

This section describes in details the parameters available for this application. Table <sup>57</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *FineRegistration* .

---

<sup>57</sup> Table: Parameters table for Fine Registration.

Parameter Key	Parameter Type	Parameter Description
ref	Input image	Input image
sec	Input image	Input image
out	Output image	Output image
erx	Int	Int
ery	Int	Int
mrx	Int	Int
mry	Int	Int
w	Input image	Input image
wo	Output image	Output image
cox	Float	Float
coy	Float	Float
ssrx	Float	Float
ssry	Float	Float
rgsx	Float	Float
rgsy	Float	Float
sgsx	Float	Float
sgsy	Float	Float
m	String	String
spa	Float	Float
cva	Float	Float
vmlt	Float	Float
vmut	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Reference Image:** The reference image.
- **Secondary Image:** The secondary image.
- **Output Image:** The output image.
- **Exploration Radius X:** The exploration radius along x (in pixels).
- **Exploration Radius Y:** The exploration radius along y (in pixels).
- **Metric Radius X:** Radius along x (in pixels) of the metric computation window.
- **Metric Radius Y:** Radius along y (in pixels) of the metric computation window.
- **Image To Warp:** The image to warp after disparity estimation is completed.
- **Output Warped Image:** The output warped image.
- **Coarse Offset X:** Coarse offset along x (in physical space) between the two images.
- **Coarse Offset Y:** Coarse offset along y (in physical space) between the two images.
- **Sub-Sampling Rate X:** Generates a result at a coarser resolution with a given sub-sampling rate along X.
- **Sub-Sampling Rate Y:** Generates a result at a coarser resolution with a given sub-sampling rate along Y.
- **Reference Gaussian Smoothing X:** Performs a gaussian smoothing of the reference image. Parameter is gaussian sigma (in pixels) in X direction.
- **Reference Gaussian Smoothing Y:** Performs a gaussian smoothing of the reference image. Parameter is gaussian sigma (in pixels) in Y direction.
- **Secondary Gaussian Smoothing X:** Performs a gaussian smoothing of the secondary image. Parameter is gaussian sigma (in pixels) in X direction.

- **Secondary Gaussian Smoothing Y:** Performs a gaussian smoothing of the secondary image. Parameter is gaussian sigma (in pixels) in Y direction.
- **Metric:** Choose the metric used for block matching. Available metrics are cross-correlation (CC), cross-correlation with subtracted mean (CCSM), mean-square difference (MSD), mean reciprocal square difference (MRSD) and mutual information (MI). Default is cross-correlation.
- **SubPixelAccuracy:** Metric extrema location will be refined up to the given accuracy. Default is 0.01.
- **ConvergenceAccuracy:** Metric extrema will be refined up to the given accuracy. Default is 0.01.
- **Validity Mask Lower Threshold:** Lower threshold to obtain a validity mask.
- **Validity Mask Upper Threshold:** Upper threshold to obtain a validity mask.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_FineRegistration -ref StereoFixed.png -sec StereoMoving.png -out FineRegistration.tif -erx 2 -
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the FineRegistration application
FineRegistration = otbApplication.Registry.CreateApplication("FineRegistration")

The following lines set all the application parameters:
FineRegistration.SetParameterString("ref", "StereoFixed.png")

FineRegistration.SetParameterString("sec", "StereoMoving.png")

FineRegistration.SetParameterString("out", "FineRegistration.tif")

FineRegistration.SetParameterInt("erx", 2)

FineRegistration.SetParameterInt("ery", 2)

FineRegistration.SetParameterInt("mrx", 3)

FineRegistration.SetParameterInt("mry", 3)

The following line execute the application
FineRegistration.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.7.4 Stereo Framework

Compute the ground elevation based on one or multiple stereo pair(s)

### Detailed description

Compute the ground elevation with a stereo block matching algorithm between one or multiple stereo pair in sensor geometry. The output is projected in desired geographic or cartographic map projection (UTM by default). The pipeline is made of the following steps: for each sensor pair :

- compute the epipolar displacement grids from the stereo pair (direct and inverse)
- resample the stereo pair into epipolar geometry using BCO interpolation
- create masks for each epipolar image : remove black borders and resample input masks
- compute horizontal disparities with a block matching algorithm
- refine disparities to sub-pixel precision with a dichotomy algorithm
- apply an optional median filter
- filter disparities based on the correlation score and exploration bounds
- translate disparities in sensor geometry convert disparity to 3D Map.

Then fuse all 3D maps to produce DSM.

### Parameters

This section describes in details the parameters available for this application. Table <sup>58</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *StereoFramework* .

Parameter Key	Parameter Type	Parameter Description
input	Group	Group
input.il	Input image list	Input image list
input.co	String	String
input.channel	Int	Int
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
output	Group	Group
output.res	Float	Float
output.nodata	Float	Float
output.fusionmethod	Choices	Choices
output.fusionmethod max	<i>Choice</i>	The cell is filled with the maximum measured elevation values
output.fusionmethod min	<i>Choice</i>	The cell is filled with the minimum measured elevation values

<sup>58</sup> Table: Parameters table for Stereo Framework.

Table 6.6 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
output.fusionmethod mean	<i>Choice</i>	The cell is filled with the mean of measured elevation values
output.fusionmethod acc	<i>Choice</i>	accumulator mode. The cell is filled with the the number of values (for d
output.out	Output image	Output image
output.mode	Choices	Choices
output.mode fit	<i>Choice</i>	Fit to sensor image
output.mode user	<i>Choice</i>	User Defined
output.mode.user.ulx	Float	Float
output.mode.user.uly	Float	Float
output.mode.user.sizeX	Int	Int
output.mode.user.sizeY	Int	Int
output.mode.user.spacingX	Float	Float
output.mode.user.spacingY	Float	Float
map	Choices	Choices
map utm	<i>Choice</i>	Universal Trans-Mercator (UTM)
map lambert2	<i>Choice</i>	Lambert II Etendu
map lambert93	<i>Choice</i>	Lambert93
map wgs	<i>Choice</i>	WGS 84
map epsg	<i>Choice</i>	EPSG Code
map.utm.zone	Int	Int
map.utm.northhem	Boolean	Boolean
map.epsg.code	Int	Int
stereorect	Group	Group
stereorect.fwdgridstep	Int	Int
stereorect.invgridssrate	Int	Int
bm	Group	Group
bm.metric	Choices	Choices
bm.metric ssdmean	<i>Choice</i>	Sum of Squared Distances divided by mean of block
bm.metric ssd	<i>Choice</i>	Sum of Squared Distances
bm.metric ncc	<i>Choice</i>	Normalized Cross-Correlation
bm.metric lp	<i>Choice</i>	Lp pseudo-norm
bm.metric.lp.p	Float	Float
bm.radius	Int	Int
bm.minhoffset	Float	Float
bm.maxhoffset	Float	Float
postproc	Group	Group
postproc.bij	Boolean	Boolean
postproc.med	Boolean	Boolean
postproc.metrict	Float	Float
mask	Group	Group
mask.left	Input image	Input image
mask.right	Input image	Input image
mask.variancet	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input parameters]:** This group of parameters allows one to parametrize input data.

- **Input images list:** The list of images.
- **Couples list:** List of index of couples in image list. Couples must be separated by a comma. (index start at 0).

for example : 0 1,1 2 will process a first couple composed of the first and the second image in image list, then the first and the third image . note that images are handled by pairs. if left empty couples are created from input index i.e. a first couple will be composed of the first and second image, a second couple with third and fourth image etc. (in this case image list must be even).

- **Image channel used for the block matching:** Used channel for block matching (used for all images).

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**[Output parameters]:** This group of parameters allows one to choose the DSM resolution, nodata value, and projection parameters.

- **Output resolution:** Spatial sampling distance of the output elevation : the cell size (in m).
- **NoData value:** DSM empty cells are filled with this value (optional -32768 by default).
- **Method to fuse measures in each DSM cell:** This parameter allows one to choose the method used to fuse elevation measurements in each output DSM cell. Available choices are:
  - **The cell is filled with the maximum measured elevation values**
  - **The cell is filled with the minimum measured elevation values**
  - **The cell is filled with the mean of measured elevation values**
- **accumulator mode. The cell is filled with the the number of values (for debugging purposes).**
- **Output DSM:** Output elevation image.
- **Parameters estimation modes** Available choices are:
  - **Fit to sensor image:** Fit the size, origin and spacing to an existing ortho image (uses the value of outputs.ortho).
  - **User Defined:** This mode allows you to fully modify default values.
  - **Upper Left X:** Cartographic X coordinate of upper-left corner (meters for cartographic projections, degrees for geographic ones).
  - **Upper Left Y:** Cartographic Y coordinate of the upper-left corner (meters for cartographic projections, degrees for geographic ones).
  - **Size X:** Size of projected image along X (in pixels).
  - **Size Y:** Size of projected image along Y (in pixels).
  - **Pixel Size X:** Size of each pixel along X axis (meters for cartographic projections, degrees for geographic ones).
  - **Pixel Size Y:** Size of each pixel along Y axis (meters for cartographic projections, degrees for geographic ones).

**Output Cartographic Map Projection:** Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM):** A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
- **Zone number:** The zone number ranges from 1 to 60 and allows defining the transverse mercator projection (along with the hemisphere).
- **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu:** This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93:** This is a Lambert 93 projection mainly used in France.
- **WGS 84:** This is a Geographical projection.
- **EPSG Code:** This code is a generic way of identifying map projections, and allows specifying a large amount of them. See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection;.
- **EPSG Code:** See [www.spatialreference.org](http://www.spatialreference.org) to find which EPSG code is associated to your projection.

**[Stereorectification Grid parameters]:** This group of parameters allows one to choose direct and inverse grid sub-sampling. These parameters are very useful to tune time and memory consumption.

- **Step of the displacement grid (in pixels):** Stereo-rectification displacement grid only varies slowly. Therefore, it is recommended to use a coarser grid (higher step value) in case of large images.
- **Sub-sampling rate for epipolar grid inversion:** Grid inversion is an heavy process that implies spline regression on control points. To avoid eating too much memory, this parameter allows one to first sub-sample the field to invert.

**[Block matching parameters]:** This group of parameters allow tuning the block-matching behavior.

- **Block-matching metric** Available choices are:
  - **Sum of Squared Distances divided by mean of block:** derived version of Sum of Squared Distances between pixels value in the metric window (SSD divided by mean over window).
  - **Sum of Squared Distances:** Sum of squared distances between pixels value in the metric window.
  - **Normalized Cross-Correlation:** Normalized Cross-Correlation between the left and right windows.
  - **Lp pseudo-norm:** Lp pseudo-norm between the left and right windows.
  - **p value:** Value of the p parameter in Lp pseudo-norm (must be positive).
- **Radius of blocks for matching filter (in pixels):** The radius of blocks in Block-Matching (in pixels).
- **Minimum altitude offset (in meters):** Minimum altitude below the selected elevation source (in meters).
- **Maximum altitude offset (in meters):** Maximum altitude above the selected elevation source (in meters).

**[Postprocessing parameters]:** This group of parameters allow use optional filters.

- **Use bijection consistency in block matching strategy:** use bijection consistency. Right to Left correlation is computed to validate Left to Right disparities. If bijection is not found pixel is rejected.
- **Use median disparities filtering:** disparities output can be filtered using median post filtering (disabled by default).
- **Correlation metric threshold:** Use block matching metric output to discard pixels with low correlation value (disabled by default, float value).

**[Masks]**

- **Input left mask:** Mask for left input image.

- **Input right mask:** Mask for right input image.
- **Discard pixels with low local variance:** This parameter allows one to discard pixels whose local variance is too small (the size of the neighborhood is given by the radius parameter).

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_StereoFramework -input.il sensor_stereo_left.tif sensor_stereo_right.tif -elev.default 200 -st
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the StereoFramework application
StereoFramework = otbApplication.Registry.CreateApplication("StereoFramework")

The following lines set all the application parameters:
StereoFramework.SetParameterStringList("input.il", ['sensor_stereo_left.tif', 'sensor_stereo_right.tif'])

StereoFramework.SetParameterFloat("elev.default", 200)

StereoFramework.SetParameterInt("stereorect.fwdgridstep", 8)

StereoFramework.SetParameterInt("stereorect.invgridssrate", 4)

StereoFramework.SetParameterString("postproc.med", "1")

StereoFramework.SetParameterFloat("output.res", 2.5)

StereoFramework.SetParameterString("output.out", "dem.tif")

The following line execute the application
StereoFramework.ExecuteAndWriteOutput()
```

## Authors

This application has been written by OTB-Team.

## 6.7.5 Stereo-rectification deformation grid generator

Generates two deformation fields to stereo-rectify (i.e. resample in epipolar geometry) a pair of stereo images up to the sensor model precision

## Detailed description

This application generates a pair of deformation grid to stereo-rectify a pair of stereo images according to sensor modelling and a mean elevation hypothesis. The deformation grids can be passed to the StereoRectificationGridGenerator application for actual resampling in epipolar geometry.

## Parameters

This section describes in details the parameters available for this application. Table <sup>59</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *StereoRectificationGridGenerator* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.inleft	Input image	Input image
io.inright	Input image	Input image
io.outleft	Output image	Output image
io.outright	Output image	Output image
epi	Group	Group
epi.elevation	Group	Group
epi.elevation.dem	Directory	Directory
epi.elevation.geoid	Input File name	Input File name
epi.elevation.default	Float	Float
epi.elevation.avgdem	Group	Group
epi.elevation.avgdem.step	Int	Int
epi.elevation.avgdem.value	Float	Float
epi.elevation.avgdem.mindisp	Float	Float
epi.elevation.avgdem.maxdisp	Float	Float
epi.scale	Float	Float
epi.step	Int	Int
epi.rectsizeX	Int	Int
epi.rectsizeY	Int	Int
epi.baseline	Float	Float
inverse	Group	Group
inverse.outleft	Output image	Output image
inverse.outright	Output image	Output image
inverse.ssrates	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting the input and output images.

- **Left input image:** The left input image to resample.
- **Right input image:** The right input image to resample.
- **Left output deformation grid:** The output deformation grid to be used to resample the left input image.
- **Right output deformation grid:** The output deformation grid to be used to resample the right input image.

**[Epipolar geometry and grid parameters]:** Parameters of the epipolar geometry and output grids.

- **Elevation management:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

<sup>59</sup> Table: Parameters table for Stereo-rectification deformation grid generator.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occur if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.
- **Average elevation computed from DEM:** Average elevation computed from the provided DEM.
- **Sub-sampling step:** Step of sub-sampling for average elevation estimation.
- **Average elevation value:** Average elevation value estimated from DEM.
- **Minimum disparity from DEM:** Disparity corresponding to estimated minimum elevation over the left image.
- **Maximum disparity from DEM:** Disparity corresponding to estimated maximum elevation over the left image.
- **Scale of epipolar images:** The scale parameter allows generating zoomed-in (scale < 1) or zoomed-out (scale > 1) epipolar images.
- **Step of the deformation grid (in nb. of pixels):** Stereo-rectification deformation grid only varies slowly. Therefore, it is recommended to use a coarser grid (higher step value) in case of large images.
- **Rectified image size X:** The application computes the optimal rectified image size so that the whole left input image fits into the rectified area. However, due to the scale and step parameter, this size may not match the size of the deformation field output. In this case, one can use these output values.
- **Rectified image size Y:** The application computes the optimal rectified image size so that the whole left input image fits into the rectified area. However, due to the scale and step parameter, this size may not match the size of the deformation field output. In this case, one can use these output values.
- **Mean baseline ratio:** This parameter is the mean value, in pixels.meters<sup>-1</sup>, of the baseline to sensor altitude ratio. It can be used to convert disparities to physical elevation, since a disparity of one pixel will correspond to an elevation offset of the invert of this value with respect to the mean elevation.

[Write inverse fields]: This group of parameter allows generating the inverse fields as well.

- **Left inverse deformation grid:** The output deformation grid to be used to resample the epipolar left image.
- **Right inverse deformation grid:** The output deformation grid to be used to resample the epipolar right image.
- **Sub-sampling rate for inversion:** Grid inversion is an heavy process that implies spline regression on control points. To avoid eating too much memory, this parameter allows one to first sub-sample the field to invert.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_StereoRectificationGridGenerator -io.inleft wv2_xs_left.tif -io.inright wv2_xs_right.tif -io.out
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the StereoRectificationGridGenerator application
StereoRectificationGridGenerator = otbApplication.Registry.CreateApplication("StereoRectificationGridGenerator")

The following lines set all the application parameters:
StereoRectificationGridGenerator.SetParameterString("io.inleft", "wv2_xs_left.tif")

StereoRectificationGridGenerator.SetParameterString("io.inright", "wv2_xs_left.tif")

StereoRectificationGridGenerator.SetParameterString("io.outleft", "wv2_xs_left_epi_field.tif")

StereoRectificationGridGenerator.SetParameterString("io.outright", "wv2_xs_right_epi_field.tif")

StereoRectificationGridGenerator.SetParameterFloat("epi.elevation.default", 400)

The following line execute the application
StereoRectificationGridGenerator.ExecuteAndWriteOutput()
```

### Limitations

Generation of the deformation grid is not streamable, pay attention to this fact when setting the grid step.

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

[otbGridBasedImageResampling](#)

## 6.8 Learning

### 6.8.1 Classification Map Regularization

Filters the input labeled image using Majority Voting in a ball shaped neighborhood.

#### Detailed description

**This application filters the input labeled image (with a maximal class label = 65535) using Majority Voting in a ball shaped neighborhood.**

-NoData is the label of the NOT classified pixels in the input image. These input pixels keep their NoData label in the output image. -Pixels with more than 1 majority class are marked as Undecided if the parameter 'ip.suvbool == true', or keep their Original labels otherwise.

## Parameters

This section describes in details the parameters available for this application. Table <sup>60</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ClassificationMapRegularization*.

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.in	Input image	Input image
io.out	Output image	Output image
ip	Group	Group
ip.radius	Int	Int
ip.suvbool	Boolean	Boolean
ip.nodatalabel	Int	Int
ip.undecidedlabel	Int	Int
ip.onlyisolatedpixels	Boolean	Boolean
ip.isolatedthreshold	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output images]:** This group of parameters allows setting input and output images for classification map regularization by Majority Voting.

- **Input classification image:** The input labeled image to regularize.
- **Output regularized image:** The output regularized labeled image.

**[Regularization parameters]:** This group allows setting parameters for classification map regularization by Majority Voting.

- **Structuring element radius (in pixels):** The radius of the ball shaped structuring element (expressed in pixels). By default, 'ip.radius = 1 pixel'.
- **Multiple majority: Undecided(X)/Original:** Pixels with more than 1 majority class are marked as Undecided if this parameter is checked (true), or keep their Original labels otherwise (false). Please note that the Undecided value must be different from existing labels in the input labeled image. By default, 'ip.suvbool = false'.
- **Label for the NoData class:** Label for the NoData class. Such input pixels keep their NoData label in the output image. By default, 'ip.nodatalabel = 0'.
- **Label for the Undecided class:** Label for the Undecided class. By default, 'ip.undecidedlabel = 0'.
- **Process isolated pixels only:** Only pixels whose label is unique in the neighborhood will be processed. By default, 'ip.onlyisolatedpixels = false'.
- **Threshold for isolated pixels:** Maximum number of neighbours with the same label as the center pixel to consider that it is an isolated pixel. By default, 'ip.isolatedthreshold = 1'.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

<sup>60</sup> Table: Parameters table for Classification Map Regularization.

```
otbcli_ClassificationMapRegularization -io.in clLabeledImageQB123_1.tif -io.out clLabeledImageQB123_1.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ClassificationMapRegularization application
ClassificationMapRegularization = otbApplication.Registry.CreateApplication("ClassificationMapRegularization")

The following lines set all the application parameters:
ClassificationMapRegularization.SetParameterString("io.in", "clLabeledImageQB123_1.tif")

ClassificationMapRegularization.SetParameterString("io.out", "clLabeledImageQB123_1_CMR_r2_nod1_10.tif")

ClassificationMapRegularization.SetParameterInt("ip.radius", 2)

ClassificationMapRegularization.SetParameterString("ip.suvbool", "1")

ClassificationMapRegularization.SetParameterString("ip.onlyisolatedpixels", "1")

ClassificationMapRegularization.SetParameterInt("ip.nodatalabel", 10)

ClassificationMapRegularization.SetParameterInt("ip.undecidedlabel", 7)

The following line execute the application
ClassificationMapRegularization.ExecuteAndWriteOutput()
```

## Limitations

The input image must be a single band labeled image (with a maximal class label = 65535). The structuring element radius must have a minimum value equal to 1 pixel. Please note that the Undecided value must be different from existing labels in the input labeled image.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:  
Documentation of the ClassificationMapRegularization application.

## 6.8.2 Confusion matrix Computation

Computes the confusion matrix of a classification

## Detailed description

This application computes the confusion matrix of a classification map relatively to a ground truth. This ground truth can be given as a raster or a vector data. Only reference and produced pixels with values different from NoData are handled in the calculation of the confusion matrix. The confusion matrix is organized the following way: rows = reference labels, columns = produced labels. In the header of the output file, the reference and produced class labels are ordered according to the rows/columns of the confusion matrix.

## Parameters

This section describes in details the parameters available for this application. Table <sup>61</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ComputeConfusionMatrix*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output File name	Output File name
ref	Choices	Choices
ref raster	<i>Choice</i>	Ground truth as a raster image
ref vector	<i>Choice</i>	Ground truth as a vector data file
ref.raster.in	Input image	Input image
ref.vector.in	Input File name	Input File name
ref.vector.field	String	String
nodatalabel	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input classification image.

**Matrix output:** Filename to store the output matrix (csv format).

**Ground truth:** Choice of ground truth format. Available choices are:

- **Ground truth as a raster image**
- **Input reference image:** Input image containing the ground truth labels.
- **Ground truth as a vector data file**
- **Input reference vector data:** Input vector data of the ground truth.
- **Field name:** Field name containing the label values.

**Value for nodata pixels:** Label for the NoData class. Such input pixels will be discarded from the ground truth and from the input classification map. By default, 'nodatalabel = 0'.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

<sup>61</sup> Table: Parameters table for Confusion matrix Computation.

```
otbcli_ComputeConfusionMatrix -in clLabeledImageQB1.tif -out ConfusionMatrix.csv -ref vector -ref.ve
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ComputeConfusionMatrix application
ComputeConfusionMatrix = otbApplication.Registry.CreateApplication("ComputeConfusionMatrix")

The following lines set all the application parameters:
ComputeConfusionMatrix.SetParameterString("in", "clLabeledImageQB1.tif")

ComputeConfusionMatrix.SetParameterString("out", "ConfusionMatrix.csv")

ComputeConfusionMatrix.SetParameterString("ref", "vector")

ComputeConfusionMatrix.SetParameterString("ref.vector.in", "VectorData_QB1_bis.shp")

ComputeConfusionMatrix.SetParameterString("ref.vector.field", "Class")

ComputeConfusionMatrix.SetParameterInt("nodatalabel", 255)

The following line execute the application
ComputeConfusionMatrix.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.8.3 Compute Images second order statistics

Computes global mean and standard deviation for each band from a set of images and optionally saves the results in an XML file.

### Detailed description

This application computes a global mean and standard deviation for each band of a set of images and optionally saves the results in an XML file. The output XML is intended to be used as an input for the TrainImagesClassifier application to normalize samples before learning.

### Parameters

This section describes in details the parameters available for this application. Table <sup>62</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is

---

<sup>62</sup> Table: Parameters table for Compute Images second order statistics.

*ComputeImagesStatistics* .

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
bv	Float	Float
out	Output File name	Output File name
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input images:** List of input images filenames.
- **Background Value:** Background value to ignore in statistics computation.
- **Output XML file:** XML filename where the statistics are saved for future reuse.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ComputeImagesStatistics -il QB_1_ortho.tif -out EstimateImageStatisticsQB1.xml
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ComputeImagesStatistics application
ComputeImagesStatistics = otbApplication.Registry.CreateApplication("ComputeImagesStatistics")

The following lines set all the application parameters:
ComputeImagesStatistics.SetParameterStringList("il", ['QB_1_ortho.tif'])

ComputeImagesStatistics.SetParameterString("out", "EstimateImageStatisticsQB1.xml")

The following line execute the application
ComputeImagesStatistics.ExecuteAndWriteOutput()
```

## Limitations

Each image of the set must contain the same bands as the others (i.e. same types, in the same order).

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

Documentation of the TrainImagesClassifier application.

## 6.8.4 Fusion of Classifications

Fuses several classifications maps of the same image on the basis of class labels.

### Detailed description

This application allows you to fuse several classification maps and produces a single more robust classification map. Fusion is done either by mean of Majority Voting, or with the Dempster Shafer combination method on class labels.

- **MAJORITY VOTING:** for each pixel, the class with the highest number of votes is selected.
- **DEMPSTER SHAFER:** for each pixel, the class label for which the Belief Function is maximal is selected. This Belief Function is calculated by mean of the Dempster Shafer combination of Masses of Belief, and indicates the belief that each input classification map presents for each label value. Moreover, the Masses of Belief are based on the input fusion matrices of each classification map, either by using the PRECISION or RECALL rates, or the OVERALL ACCURACY, or the KAPPA coefficient. Thus, each input classification map needs to be associated with its corresponding input confusion matrix file for the Dempster Shafer fusion.
- Input pixels with the NODATA label are not handled in the fusion of classification maps. Moreover, pixels for which all the input classifiers are set to NODATA keep this value in the output fused image.
- In case of number of votes equality, the UNDECIDED label is attributed to the pixel.

### Parameters

This section describes in details the parameters available for this application. Table <sup>63</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *FusionOfClassifications* .

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
method	Choices	Choices
method majorityvoting	Choice	Majority Voting
method dempstershafer	Choice	Dempster Shafer combination
method.dempstershafer.cmfl	Input File name list	Input File name list
method.dempstershafer.mob	Choices	Choices
method.dempstershafer.mob precision	Choice	Precision
method.dempstershafer.mob recall	Choice	Recall
method.dempstershafer.mob accuracy	Choice	Overall Accuracy
method.dempstershafer.mob kappa	Choice	Kappa
nodatalabel	Int	Int
undecidedlabel	Int	Int
out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

<sup>63</sup> Table: Parameters table for Fusion of Classifications.

**Input classifications:** List of input classification maps to fuse. Labels in each classification image must represent the same class.

**Fusion method:** Selection of the fusion method and its parameters. Available choices are:

- **Majority Voting:** Fusion of classification maps by majority voting for each output pixel.
- **Dempster Shafer combination:** Fusion of classification maps by the Dempster Shafer combination method for each output pixel.
  - **Confusion Matrices:** A list of confusion matrix files (\*.CSV format) to define the masses of belief and the class labels. Each file should be formatted the following way: the first line, beginning with a '#' symbol, should be a list of the class labels present in the corresponding input classification image, organized in the same order as the confusion matrix rows/columns.
  - **Mass of belief measurement:** Type of confusion matrix measurement used to compute the masses of belief of each classifier. Available choices are:
  - **Precision:** Masses of belief = Precision rates of each classifier (one rate per class label).
  - **Recall:** Masses of belief = Recall rates of each classifier (one rate per class label).
  - **Overall Accuracy:** Mass of belief = Overall Accuracy of each classifier (one unique value for all the class labels).
  - **Kappa:** Mass of belief = Kappa coefficient of each classifier (one unique value for all the class labels).

**Label for the NoData class:** Label for the NoData class. Such input pixels keep their NoData label in the output image and are not handled in the fusion process. By default, 'nodatalabel = 0'.

**Label for the Undecided class:** Label for the Undecided class. Pixels with more than 1 fused class are marked as Undecided. Please note that the Undecided value must be different from existing labels in the input classifications. By default, 'undecidedlabel = 0'.

**The output classification image:** The output classification image resulting from the fusion of the input classification images.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_FusionOfClassifications -il classification1.tif classification2.tif classification3.tif -meth
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the FusionOfClassifications application
FusionOfClassifications = otbApplication.Registry.CreateApplication("FusionOfClassifications")

The following lines set all the application parameters:
FusionOfClassifications.SetParameterStringList("il", ['classification1.tif', 'classification2.tif', '
FusionOfClassifications.SetParameterString("method", "dempstershafer")
```

```
FusionOfClassifications.SetParameterString("method.dempstershafer.mob","precision")
FusionOfClassifications.SetParameterInt("nodatalabel", 0)
FusionOfClassifications.SetParameterInt("undecidedlabel", 10)
FusionOfClassifications.SetParameterString("out", "classification_fused.tif")

The following line execute the application
FusionOfClassifications.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

ImageClassifier application

## 6.8.5 Image Classification

Performs a classification of the input image according to a model file.

### Detailed description

This application performs an image classification based on a model file produced by the TrainImagesClassifier application. Pixels of the output image will contain the class labels decided by the classifier (maximal class label = 65535). The input pixels can be optionally centered and reduced according to the statistics file produced by the ComputeImagesStatistics application. An optional input mask can be provided, in which case only input image pixels whose corresponding mask value is greater than 0 will be classified. The remaining of pixels will be given the label 0 in the output image.

### Parameters

This section describes in details the parameters available for this application. Table <sup>64</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ImageClassifier* .

---

<sup>64</sup> Table: Parameters table for Image Classification.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
mask	Input image	Input image
model	Input File name	Input File name
imstat	Input File name	Input File name
out	Output image	Output image
confmap	Output image	Output image
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The input image to classify.
- **Input Mask:** The mask allows restricting classification of the input image to the area where mask pixel values are greater than 0.
- **Model file:** A model file (produced by TrainImagesClassifier application, maximal class label = 65535).
- **Statistics file:** A XML file containing mean and standard deviation to center and reduce samples before classification (produced by ComputeImagesStatistics application).
- **Output Image:** Output image containing class labels.
- **Confidence map:** Confidence map of the produced classification. The confidence index depends on the model : - LibSVM : difference between the two highest probabilities (needs a model with probability estimates, so that classes probabilities can be computed for each sample) - OpenCV \* Boost : sum of votes \* DecisionTree : (not supported) \* GradientBoostedTree : (not supported) \* KNearestNeighbors : number of neighbors with the same label \* NeuralNetwork : difference between the two highest responses \* NormalBayes : (not supported) \* RandomForest : Confidence (proportion of votes for the majority class). Margin (normalized difference of the votes of the 2 majority classes) is not available for now. \* SVM : distance to margin (only works for 2-class models) .
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ImageClassifier -in QB_1_ortho.tif -imstat EstimateImageStatisticsQB1.xml -model clsvmModelQB1
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ImageClassifier application
ImageClassifier = otbApplication.Registry.CreateApplication("ImageClassifier")

The following lines set all the application parameters:
ImageClassifier.SetParameterString("in", "QB_1_ortho.tif")

ImageClassifier.SetParameterString("imstat", "EstimateImageStatisticsQB1.xml")
```

```
ImageClassifier.SetParameterString("model", "clsvmModelQB1.svm")
ImageClassifier.SetParameterString("out", "clLabeledImageQB1.tif")
The following line execute the application
ImageClassifier.ExecuteAndWriteOutput ()
```

### Limitations

The input image must have the same type, order and number of bands than the images used to produce the statistics file and the SVM model file. If a statistics file was used during training by the TrainImagesClassifier, it is mandatory to use the same statistics file for classification. If an input mask is used, its size must match the input image size.

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

TrainImagesClassifier, ValidateImagesClassifier, ComputeImagesStatistics

## 6.8.6 Unsupervised KMeans image classification

Unsupervised KMeans image classification

### Detailed description

Performs unsupervised KMeans image classification.

### Parameters

This section describes in details the parameters available for this application. Table <sup>65</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *KMeansClassification* .

---

<sup>65</sup> Table: Parameters table for Unsupervised KMeans image classification.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ram	Int	Int
vm	Input image	Input image
ts	Int	Int
nc	Int	Int
maxit	Int	Int
ct	Float	Float
outmeans	Output File name	Output File name
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input image to classify.
- **Output Image:** Output image containing the class indexes.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Validity Mask:** Validity mask. Only non-zero pixels will be used to estimate KMeans modes.
- **Training set size:** Size of the training set (in pixels).
- **Number of classes:** Number of modes, which will be used to generate class membership.
- **Maximum number of iterations:** Maximum number of iterations for the learning step.
- **Convergence threshold:** Convergence threshold for class centroid (L2 distance, by default 0.0001).
- **Centroid filename:** Output text file containing centroid positions.
- **set user defined seed:** Set specific seed. with integer value.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_KMeansClassification -in QB_1_ortho.tif -ts 1000 -nc 5 -maxit 1000 -ct 0.0001 -out Classification
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the KMeansClassification application
KMeansClassification = otbApplication.Registry.CreateApplication("KMeansClassification")

The following lines set all the application parameters:
KMeansClassification.SetParameterString("in", "QB_1_ortho.tif")

KMeansClassification.SetParameterInt("ts", 1000)

KMeansClassification.SetParameterInt("nc", 5)
```

```
KMeansClassification.SetParameterInt("maxit", 1000)
KMeansClassification.SetParameterFloat("ct", 0.0001)
KMeansClassification.SetParameterString("out", "ClassificationFilterOutput.tif")

The following line execute the application
KMeansClassification.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.8.7 Multi-image sampling rate estimation

Compute sampling rate for an input set of images.

### Detailed description

The application computes sampling rates for a set of input images. Before calling this application, each pair of image and training vectors has to be analysed with the application PolygonClassStatistics. The statistics file is then used to compute the sampling rates for each class in each image. Several types of sampling are implemented. Each one is a combination of a mono-image strategy and a multi-image mode. The mono-image strategies are :

- smallest (default) : select the same number of sample in each class so that the smallest one is fully sampled.
- constant : select the same number of samples N in each class (with N below or equal to the size of the smallest class).
- byclass : set the required number for each class manually, with an input CSV file (first column is class name, second one is the required samples number).

**The multi-image modes (mim) are proportional, equal and custom. The custom mode lets the users choose the distribution of sa**

- strategy = all
  - Same behaviour for all modes : take all samples
- strategy = constant : let's call M the global number of samples required per class. For each image i and each class c:
  - if mim = proportional, then  $N_i(c) = M * T_i(c) / \sum_k(T_k(c))$
  - if mim = equal , then  $N_i(c) = M / L$
  - if mim = custom , then  $N_i(c) = M_i$  where  $M_i$  is the custom requested number of samples for image i
- strategy = byClass : let's call M(c) the global number of samples for class c). For each image i and each class c:
  - if mim = proportional, then  $N_i(c) = M(c) * T_i(c) / \sum_k(T_k(c))$

- if `mim = equal` , then  $Ni(c) = M(c) / L$
- if `mim = custom` , then  $Ni(c) = Mi(c)$  where  $Mi(c)$  is the custom requested number of samples for image  $i$  and class  $c$
- `strategy = percent` : For each image  $i$  and each class  $c$ :
  - if `mim = proportional`, then  $Ni(c) = p * Ti(c)$  where  $p$  is the global percentage of samples
  - if `mim = equal` , then  $Ni(c) = p * \sum_k(Tk(c))/L$  where  $p$  is the global percentage of samples
  - if `mim = custom` , then  $Ni(c) = p(i) * Ti(c)$  where  $p(i)$  is the percentage of samples for image  $i$ .  $c$
- `strategy = total` : For each image  $i$  and each class  $c$ :
  - if `mim = proportional`, then  $Ni(c) = total * (\sum_k(Ti(k))/\sum_k(Tl(k))) * (Ti(c)/\sum_k(Ti(k)))$  where  $total$  is the total number of samples specified.
  - if `mim = equal` , then  $Ni(c) = (total / L) * (Ti(c)/\sum_k(Ti(k)))$  where  $total$  is the total number of samples specified.
  - if `mim = custom` , then  $Ni(c) = total(i) * (Ti(c)/\sum_k(Ti(k)))$  where  $total(i)$  is the total number of samples specified for image  $i$ .
- `strategy = smallest class`
  - if `mim = proportional`, then the smallest class size (computed globally) is used for the strategy constant+proportional.
  - if `mim = equal` , then the smallest class size (computed globally) is used for the strategy constant+equal.
  - if `mim = custom` , then the smallest class is computed and used for each image separately.

## Parameters

This section describes in details the parameters available for this application. Table <sup>66</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MultiImageSamplingRate* .

<sup>66</sup> Table: Parameters table for Multi-image sampling rate estimation.

Parameter Key	Parameter Type	Parameter Description
il	Input File name list	Input File name list
out	Output File name	Output File name
strategy	Choices	Choices
strategy byclass	<i>Choice</i>	Set samples count for each class
strategy constant	<i>Choice</i>	Set the same samples counts for all classes
strategy smallest	<i>Choice</i>	Set same number of samples for all classes, with the smallest class fully sampled
strategy percent	<i>Choice</i>	Use a percentage of the samples available for each class
strategy total	<i>Choice</i>	Set the total number of samples to generate, and use class proportions.
strategy all	<i>Choice</i>	Take all samples
strategy.byclass.in	Input File name list	Input File name list
strategy.constant.nb	String	String
strategy.percent.p	String	String
strategy.total.v	String	String
mim	Choices	Choices
mim proportional	<i>Choice</i>	Proportional
mim equal	<i>Choice</i>	equal
mim custom	<i>Choice</i>	Custom
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input statistics:** List of statistics files for each input image.

**Output sampling rates:** Output filename storing sampling rates (CSV format with class name, required samples, total samples, and rate). The given filename will be used with a suffix to indicate the corresponding input index (for instance: rates.csv will give rates\_1.csv, rates\_2.csv, ...).

**Sampling strategy** Available choices are:

- **Set samples count for each class:** Set samples count for each class.
- **Number of samples by class:** Number of samples by class (CSV format with class name in 1st column and required samples in the 2nd). In the case of the custom multi-image mode, several inputs may be given for each image.
- **Set the same samples counts for all classes:** Set the same samples counts for all classes.
- **Number of samples for all classes:** Number of samples for all classes. In the case of the custom multi-image mode, several values can be given for each image.
- **Set same number of samples for all classes, with the smallest class fully sampled:** Set same number of samples for all classes, with the smallest class fully sampled.
- **Use a percentage of the samples available for each class:** Use a percentage of the samples available for each class.
- **The percentage(s) to use:** The percentage(s) to use. In the case of the custom multi-image mode, several values can be given for each image.

- **Set the total number of samples to generate, and use class proportions.:** Set the total number of samples to generate, and use class proportions.
- **The number of samples to generate:** The number of samples to generate. In the case of the custom multi-image mode, several values can be given for each image.
- **Take all samples:** Take all samples.

**Multi-Image Mode** Available choices are:

- **Proportional:** Split proportionally the required number of samples.
- **equal:** Split equally the required number of samples.
- **Custom:** Split the required number of samples following user choice.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_MultiImageSamplingRate -il stats_1.xml stats_2.xml -out rates.csv -strategy smallest -mim prop
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the MultiImageSamplingRate application
MultiImageSamplingRate = otbApplication.Registry.CreateApplication("MultiImageSamplingRate")

The following lines set all the application parameters:

MultiImageSamplingRate.SetParameterString("out", "rates.csv")

MultiImageSamplingRate.SetParameterString("strategy", "smallest")

MultiImageSamplingRate.SetParameterString("mim", "proportional")

The following line execute the application
MultiImageSamplingRate.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.8.8 Polygon Class Statistics

Computes statistics on a training polygon set.

### Detailed description

The application processes a set of geometries intended for training (they should have a field giving the associated class). The geo

- number of samples per class
- number of samples per geometry

An optional raster mask can be used to discard samples. Different types of geometry are supported [polygons, lines, points. The behaviour is different for each type of geometry :]

- polygon: select pixels whose center is inside the polygon
- lines : select pixels intersecting the line
- points : select closest pixel to the point

### Parameters

This section describes in details the parameters available for this application. Table <sup>67</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *PolygonClassStatistics* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
mask	Input image	Input image
vec	Input File name	Input File name
out	Output File name	Output File name
field	String	String
layer	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **InputImage**: Support image that will be classified.
- **InputMask**: Validity mask (only pixels corresponding to a mask value greater than 0 will be used for statistics).
- **Input vectors**: Input geometries to analyse.
- **Output Statistics**: Output file to store statistics (XML format).
- **Field Name**: Name of the field carrying the class name in the input vectors.
- **Layer Index**: Layer index to read in the input vector file.
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Load otb application from xml file**: Load otb application from xml file.
- **Save otb application to xml file**: Save otb application to xml file.

<sup>67</sup> Table: Parameters table for Polygon Class Statistics.

## Example

To run this example in command-line, use the following:

```
otbcli_PolygonClassStatistics -in support_image.tif -vec variousVectors.sqlite -field label -out poly
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the PolygonClassStatistics application
PolygonClassStatistics = otbApplication.Registry.CreateApplication("PolygonClassStatistics")

The following lines set all the application parameters:
PolygonClassStatistics.SetParameterString("in", "support_image.tif")

PolygonClassStatistics.SetParameterString("vec", "variousVectors.sqlite")

PolygonClassStatistics.SetParameterString("field", "label")

PolygonClassStatistics.SetParameterString("out", "polygonStat.xml")

The following line execute the application
PolygonClassStatistics.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.8.9 Predict Regression

Performs a prediction of the input image according to a regression model file.

### Detailed description

This application predict output values from an input image, based on a regression model file produced by the Train-Regression application. Pixels of the output image will contain the predicted values from the regression model (single band). The input pixels can be optionally centered and reduced according to the statistics file produced by the ComputeImagesStatistics application. An optional input mask can be provided, in which case only input image pixels whose corresponding mask value is greater than 0 will be processed. The remaining of pixels will be given the value 0 in the output image.

## Parameters

This section describes in details the parameters available for this application. Table <sup>68</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *PredictRegression*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
mask	Input image	Input image
model	Input File name	Input File name
imstat	Input File name	Input File name
out	Output image	Output image
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The input image to predict.
- **Input Mask:** The mask allow restricting classification of the input image to the area where mask pixel values are greater than 0.
- **Model file:** A regression model file (produced by TrainRegression application).
- **Statistics file:** A XML file containing mean and standard deviation to center and reduce samples before prediction (produced by ComputeImagesStatistics application). If this file contains one more band than the sample size, the last stat of last band will be applied to expand the output predicted value.
- **Output Image:** Output image containing predicted values.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_PredictRegression -in QB_1_ortho.tif -imstat EstimateImageStatisticsQB1.xml -model clsvmModelQB1.svm
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the PredictRegression application
PredictRegression = otbApplication.Registry.CreateApplication("PredictRegression")

The following lines set all the application parameters:
PredictRegression.SetParameterString("in", "QB_1_ortho.tif")

PredictRegression.SetParameterString("imstat", "EstimateImageStatisticsQB1.xml")

PredictRegression.SetParameterString("model", "clsvmModelQB1.svm")
```

<sup>68</sup> Table: Parameters table for Predict Regression.

```
PredictRegression.SetParameterString("out", "clLabeledImageQB1.tif")

The following line execute the application
PredictRegression.ExecuteAndWriteOutput()
```

### Limitations

The input image must contain the feature bands used for the model training (without the predicted value). If a statistics file was used during training by the TrainRegression, it is mandatory to use the same statistics file for prediction. If an input mask is used, its size must match the input image size.

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

TrainRegression, ComputeImagesStatistics

## 6.8.10 Sample Extraction

Extracts samples values from an image.

### Detailed description

The application extracts samples values from an image using positions contained in a vector data file.

### Parameters

This section describes in details the parameters available for this application. Table <sup>69</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SampleExtraction*.

---

<sup>69</sup> Table: Parameters table for Sample Extraction.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
vec	Input File name	Input File name
out	Output File name	Output File name
outfield	Choices	Choices
outfield prefix	<i>Choice</i>	Use a prefix and an incremental counter
outfield list	<i>Choice</i>	Use the given name list
outfield.prefix.name	String	String
outfield.list.names	String list	String list
field	String	String
layer	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**InputImage:** Support image.

**Input sampling positions:** Vector data file containing samplingpositions. (OGR format).

**Output samples:** Output vector data file storing samplevalues (OGR format). If not given, the input vector data file is updated.

**Output field names:** Choice between naming method for output fields. Available choices are:

- **Use a prefix and an incremental counter:** Use a prefix and an incremental counter.
- **Output field prefix:** Prefix used to form the field names thatwill contain the extracted values.
- **Use the given name list:** Use the given name list.
- **Output field names:** Full list of output field names.

**Field Name:** Name of the field carrying the classname in the input vectors. This field is copied to output.

**Layer Index:** Layer index to read in the input vector file.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SampleExtraction -in support_image.tif -vec sample_positions.sqlite -outfield prefix -outfield
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SampleExtraction application
SampleExtraction = otbApplication.Registry.CreateApplication("SampleExtraction")

The following lines set all the application parameters:
SampleExtraction.SetParameterString("in", "support_image.tif")
```

```

SampleExtraction.SetParameterString("vec", "sample_positions.sqlite")

SampleExtraction.SetParameterString("outfield","prefix")

SampleExtraction.SetParameterString("outfield.prefix.name", "band_")

SampleExtraction.SetParameterString("field", "label")

SampleExtraction.SetParameterString("out", "sample_values.sqlite")

The following line execute the application
SampleExtraction.ExecuteAndWriteOutput ()

```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.8.11 Sample Selection

Selects samples from a training vector data set.

### Detailed description

The application selects a set of samples from geometries intended for training (they should have a field giving the associated class).

First of all, the geometries must be analyzed by the PolygonClassStatistics application to compute statistics about the geometries, which are summarized in an xml file. Then, this xml file must be given as input to this application (parameter instats).

The input support image and the input training vectors shall be given in parameters 'in' and 'vec' respectively. Only the sampling grid (origin, size, spacing) will be read in the input image. There are several strategies to select samples (parameter strategy) :

- smallest (default) : select the same number of sample in each class so that the smallest one is fully sampled.
- constant : select the same number of samples N in each class (with N below or equal to the size of the smallest class).
- byclass : set the required number for each class manually, with an input CSV file (first column is class name, second one is the required samples number).
- percent: set a target global percentage of samples to use. Class proportions will be respected.
- total: set a target total number of samples to use. Class proportions will be respected.

There is also a choice on the sampling type to performs :

- periodic : select samples uniformly distributed
- random : select samples randomly distributed

Once the strategy and type are selected, the application outputs samples positions(parameter out).

The other parameters to look at are :

- layer : index specifying from which layer to pick geometries.
- field : set the field name containing the class.
- mask : an optional raster mask can be used to discard samples.
- outrates : allows outputting a CSV file that summarizes the sampling rates for each class.

As with the PolygonClassStatistics application, different types of geometry are supported : polygons, lines, points. The behavior of this application is different for each type of geometry :

- polygon: select points whose center is inside the polygon
- lines : select points intersecting the line
- points : select closest point to the provided point

### Parameters

This section describes in details the parameters available for this application. Table <sup>70</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SampleSelection* .

---

<sup>70</sup> Table: Parameters table for Sample Selection.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
mask	Input image	Input image
vec	Input File name	Input File name
out	Output File name	Output File name
instats	Input File name	Input File name
outrates	Output File name	Output File name
sampler	Choices	Choices
sampler periodic	<i>Choice</i>	Periodic sampler
sampler random	<i>Choice</i>	Random sampler
sampler.periodic.jitter	Int	Int
strategy	Choices	Choices
strategy byclass	<i>Choice</i>	Set samples count for each class
strategy constant	<i>Choice</i>	Set the same samples counts for all classes
strategy percent	<i>Choice</i>	Use a percentage of the samples available for each class
strategy total	<i>Choice</i>	Set the total number of samples to generate, and use class proportions.
strategy smallest	<i>Choice</i>	Set same number of samples for all classes, with the smallest class fully sampled
strategy all	<i>Choice</i>	Take all samples
strategy.byclass.in	Input File name	Input File name
strategy.constant.nb	Int	Int
strategy.percent.p	Float	Float
strategy.total.v	Int	Int
field	String	String
layer	Int	Int
ram	Int	Int
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**InputImage:** Support image that will be classified.

**InputMask:** Validity mask (only pixels corresponding to a mask value greater than 0 will be used for statistics).

**Input vectors:** Input geometries to analyse.

**Output vectors:** Output resampled geometries.

**Input Statistics:** Input file storing statistics (XML format).

**Output rates:** Output rates (CSV formatted).

**Sampler type:** Type of sampling (periodic, pattern based, random). Available choices are:

- **Periodic sampler:** Takes samples regularly spaced.
- **Jitter amplitude:** Jitter amplitude added during sample selection (0 = no jitter).
- **Random sampler:** The positions to select are randomly shuffled.

**Sampling strategy** Available choices are:

- **Set samples count for each class:** Set samples count for each class.

- **Number of samples by class:** Number of samples by class (CSV format with class name in 1st column and required samples in the 2nd).
- **Set the same samples counts for all classes:** Set the same samples counts for all classes.
- **Number of samples for all classes:** Number of samples for all classes.
- **Use a percentage of the samples available for each class:** Use a percentage of the samples available for each class.
- **The percentage to use:** The percentage to use.
- **Set the total number of samples to generate, and use class proportions.:** Set the total number of samples to generate, and use class proportions.
- **The number of samples to generate:** The number of samples to generate.
- **Set same number of samples for all classes, with the smallest class fully sampled:** Set same number of samples for all classes, with the smallest class fully sampled.
- **Take all samples:** Take all samples.

**Field Name:** Name of the field carrying the class name in the input vectors.

**Layer Index:** Layer index to read in the input vector file.

**Available RAM (Mb):** Available memory for processing (in MB).

**set user defined seed:** Set specific seed. with integer value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_SampleSelection -in support_image.tif -vec variousVectors.sqlite -field label -instats apTvCl
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SampleSelection application
SampleSelection = otbApplication.Registry.CreateApplication("SampleSelection")

The following lines set all the application parameters:
SampleSelection.SetParameterString("in", "support_image.tif")

SampleSelection.SetParameterString("vec", "variousVectors.sqlite")

SampleSelection.SetParameterString("field", "label")

SampleSelection.SetParameterString("instats", "apTvClPolygonClassStatisticsOut.xml")

SampleSelection.SetParameterString("out", "resampledVectors.sqlite")

The following line execute the application
SampleSelection.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.8.12 SOM Classification

SOM image classification.

### Detailed description

Unsupervised Self Organizing Map image classification.

### Parameters

This section describes in details the parameters available for this application. Table <sup>71</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SOMClassification* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
vm	Input image	Input image
tp	Float	Float
ts	Int	Int
som	Output image	Output image
sx	Int	Int
sy	Int	Int
nx	Int	Int
ny	Int	Int
ni	Int	Int
bi	Float	Float
bf	Float	Float
iv	Float	Float
ram	Int	Int
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **InputImage:** Input image to classify.
- **OutputImage:** Output classified image (each pixel contains the index of its corresponding vector in the SOM).
- **ValidityMask:** Validity mask (only pixels corresponding to a mask value greater than 0 will be used for learning).
- **TrainingProbability:** Probability for a sample to be selected in the training set.

<sup>71</sup> Table: Parameters table for SOM Classification.

- **TrainingSetSize:** Maximum training set size (in pixels).
- **SOM Map:** Output image containing the Self-Organizing Map.
- **SizeX:** X size of the SOM map.
- **SizeY:** Y size of the SOM map.
- **NeighborhoodX:** X size of the initial neighborhood in the SOM map.
- **NeighborhoodY:** Y size of the initial neighborhood in the SOM map.
- **NumberIteration:** Number of iterations for SOM learning.
- **BetaInit:** Initial learning coefficient.
- **BetaFinal:** Final learning coefficient.
- **InitialValue:** Maximum initial neuron weight.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **set user defined seed:** Set specific seed. with integer value.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_SOMClassification -in QB_1_ortho.tif -out SOMClassification.tif -tp 1.0 -ts 16384 -sx 32 -sy 32
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the SOMClassification application
SOMClassification = otbApplication.Registry.CreateApplication("SOMClassification")

The following lines set all the application parameters:
SOMClassification.SetParameterString("in", "QB_1_ortho.tif")

SOMClassification.SetParameterString("out", "SOMClassification.tif")

SOMClassification.SetParameterFloat("tp", 1.0)

SOMClassification.SetParameterInt("ts", 16384)

SOMClassification.SetParameterInt("sx", 32)

SOMClassification.SetParameterInt("sy", 32)

SOMClassification.SetParameterInt("nx", 10)

SOMClassification.SetParameterInt("ny", 10)

SOMClassification.SetParameterInt("ni", 5)
```

```
SOMClassification.SetParameterFloat("bi", 1.0)
SOMClassification.SetParameterFloat("bf", 0.1)
SOMClassification.SetParameterFloat("iv", 0)

The following line execute the application
SOMClassification.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.8.13 Train a classifier from multiple images

Train a classifier from multiple pairs of images and training vector data.

### Detailed description

**This application performs a classifier training from multiple pairs of input images and training vector data. Samples are compo**

The training vector data must contain polygons with a positive integer field representing the class label. The name of this field can be set using the “Class label field” parameter. Training and validation sample lists are built such that each class is equally represented in both lists. One parameter allows controlling the ratio between the number of samples in training and validation sets. Two parameters allow managing the size of the training and validation sets per class and per image. Several classifier parameters can be set depending on the chosen classifier. In the validation process, the confusion matrix is organized the following way: rows = reference labels, columns = produced labels. In the header of the optional confusion matrix output file, the validation (reference) and predicted (produced) class labels are ordered according to the rows/columns of the confusion matrix. This application is based on LibSVM and OpenCV Machine Learning (2.3.1 and later).

### Parameters

This section describes in details the parameters available for this application. Table <sup>72</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TrainImagesClassifier* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.il	Input image list	Input image list
io.vd	Input vector data list	Input vector data list
io.imstat	Input File name	Input File name
io.confmatout	Output File name	Output File name
io.out	Output File name	Output File name

<sup>72</sup> Table: Parameters table for Train a classifier from multiple images.

Table 6.7 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
sample	Group	Group
sample.mt	Int	Int
sample.mv	Int	Int
sample.bm	Int	Int
sample.edg	Boolean	Boolean
sample.vtr	Float	Float
sample.vfn	String	String
classifier	Choices	Choices
classifier libsvm	<i>Choice</i>	LibSVM classifier
classifier boost	<i>Choice</i>	Boost classifier
classifier dt	<i>Choice</i>	Decision Tree classifier
classifier gbt	<i>Choice</i>	Gradient Boosted Tree classifier
classifier ann	<i>Choice</i>	Artificial Neural Network classifier
classifier bayes	<i>Choice</i>	Normal Bayes classifier
classifier rf	<i>Choice</i>	Random forests classifier
classifier knn	<i>Choice</i>	KNN classifier
classifier.libsvm.k	Choices	Choices
classifier.libsvm.k linear	<i>Choice</i>	Linear
classifier.libsvm.k rbf	<i>Choice</i>	Gaussian radial basis function
classifier.libsvm.k poly	<i>Choice</i>	Polynomial
classifier.libsvm.k sigmoid	<i>Choice</i>	Sigmoid
classifier.libsvm.m	Choices	Choices
classifier.libsvm.m csvc	<i>Choice</i>	C support vector classification
classifier.libsvm.m nusvc	<i>Choice</i>	Nu support vector classification
classifier.libsvm.m oneclass	<i>Choice</i>	Distribution estimation (One Class SVM)
classifier.libsvm.c	Float	Float
classifier.libsvm.opt	Boolean	Boolean
classifier.libsvm.prob	Boolean	Boolean
classifier.boost.t	Choices	Choices
classifier.boost.t discrete	<i>Choice</i>	Discrete AdaBoost
classifier.boost.t real	<i>Choice</i>	Real AdaBoost (technique using confidence-rated predictions and working on real-valued targets)
classifier.boost.t logit	<i>Choice</i>	LogitBoost (technique producing good regression fits)
classifier.boost.t gentle	<i>Choice</i>	Gentle AdaBoost (technique setting less weight on outlier data points and using a different loss function)
classifier.boost.w	Int	Int
classifier.boost.r	Float	Float
classifier.boost.m	Int	Int
classifier.dt.max	Int	Int
classifier.dt.min	Int	Int
classifier.dt.ra	Float	Float
classifier.dt.cat	Int	Int
classifier.dt.f	Int	Int
classifier.dt.r	Boolean	Boolean
classifier.dt.t	Boolean	Boolean
classifier.gbt.w	Int	Int
classifier.gbt.s	Float	Float

Table 6.7 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
classifier.gbt.p	Float	Float
classifier.gbt.max	Int	Int
classifier.ann.t	Choices	Choices
classifier.ann.t reg	<i>Choice</i>	RPROP algorithm
classifier.ann.t back	<i>Choice</i>	Back-propagation algorithm
classifier.ann.sizes	String list	String list
classifier.ann.f	Choices	Choices
classifier.ann.f ident	<i>Choice</i>	Identity function
classifier.ann.f sig	<i>Choice</i>	Symmetrical Sigmoid function
classifier.ann.f gau	<i>Choice</i>	Gaussian function (Not completely supported)
classifier.ann.a	Float	Float
classifier.ann.b	Float	Float
classifier.ann.bpdw	Float	Float
classifier.ann.bpms	Float	Float
classifier.ann.rdw	Float	Float
classifier.ann.rdwm	Float	Float
classifier.ann.term	Choices	Choices
classifier.ann.term iter	<i>Choice</i>	Maximum number of iterations
classifier.ann.term eps	<i>Choice</i>	Epsilon
classifier.ann.term all	<i>Choice</i>	Max. iterations + Epsilon
classifier.ann.eps	Float	Float
classifier.ann.iter	Int	Int
classifier.rf.max	Int	Int
classifier.rf.min	Int	Int
classifier.rf.ra	Float	Float
classifier.rf.cat	Int	Int
classifier.rf.var	Int	Int
classifier.rf.nbtrees	Int	Int
classifier.rf.acc	Float	Float
classifier.knn.k	Int	Int
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting input and output data.

- **Input Image List:** A list of input images.
- **Input Vector Data List:** A list of vector data to select the training samples.
- **Input XML image statistics file:** Input XML file containing the mean and the standard deviation of the input images.
- **Output confusion matrix:** Output file containing the confusion matrix (.csv format).
- **Output model:** Output file containing the model estimated (.txt format).

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occur if other images are found in this directory.

- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**[Training and validation samples parameters]:** This group of parameters allows you to set training and validation sample lists parameters.

- **Maximum training sample size per class:** Maximum size per class (in pixels) of the training sample list (default = 1000) (no limit = -1). If equal to -1, then the maximal size of the available training sample list per class will be equal to the surface area of the smallest class multiplied by the training sample ratio.
- **Maximum validation sample size per class:** Maximum size per class (in pixels) of the validation sample list (default = 1000) (no limit = -1). If equal to -1, then the maximal size of the available validation sample list per class will be equal to the surface area of the smallest class multiplied by the validation sample ratio.
- **Bound sample number by minimum:** Bound the number of samples for each class by the number of available samples by the smaller class. Proportions between training and validation are respected. Default is true (=1).
- **On edge pixel inclusion:** Takes pixels on polygon edge into consideration when building training and validation samples.
- **Training and validation sample ratio:** Ratio between training and validation samples (0.0 = all training, 1.0 = all validation) (default = 0.5).
- **Name of the discrimination field:** Name of the field used to discriminate class labels in the input vector data files.

**Classifier to use for the training:** Choice of the classifier to use for the training. Available choices are:

- **LibSVM classifier:** This group of parameters allows setting SVM classifier parameters.
  - **SVM Kernel Type:** SVM Kernel Type. Available choices are:
    - **Linear**
    - **Gaussian radial basis function**
    - **Polynomial**
    - **Sigmoid**
  - **SVM Model Type:** Type of SVM formulation. Available choices are:
    - **C support vector classification**
    - **Nu support vector classification**
    - **Distribution estimation (One Class SVM)**
  - **Cost parameter C:** SVM models have a cost parameter C (1 by default) to control the trade-off between training errors and forcing rigid margins.
  - **Parameters optimization:** SVM parameters optimization flag.
  - **Probability estimation:** Probability estimation flag.
- **Boost classifier:** This group of parameters allows setting Boost classifier parameters. See complete documentation here url{<http://docs.opencv.org/modules/ml/doc/boosting.html>}.
  - **Boost Type:** Type of Boosting algorithm. Available choices are:
    - **Discrete AdaBoost**

- **Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)**
- **LogitBoost (technique producing good regression fits)**
- **Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often good with regression data)**
- **Weak count:** The number of weak classifiers.
- **Weight Trim Rate:** A threshold between 0 and 1 used to save computational time. Samples with summary weight  $\leq (1 - \text{weight\_trim\_rate})$  do not participate in the next iteration of training. Set this parameter to 0 to turn off this functionality.
- **Maximum depth of the tree:** Maximum depth of the tree.
- **Decision Tree classifier:** This group of parameters allows setting Decision Tree classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/decision\\_trees.html}](http://docs.opencv.org/modules/ml/doc/decision_trees.html).
- **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Minimum number of samples in each node:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
- **Termination criteria for regression tree**
- **Cluster possible values of a categorical variable into  $K \leq \text{cat}$  clusters to find a suboptimal split:** Cluster possible values of a categorical variable into  $K \leq \text{cat}$  clusters to find a suboptimal split.
- **K-fold cross-validations:** If  $\text{cv\_folds} > 1$ , then it prunes a tree with K-fold cross-validation where K is equal to  $\text{cv\_folds}$ .
- **Set UseIsRule flag to false:** If true, then a pruning will be harsher. This will make a tree more compact and more resistant to the training data noise but a bit less accurate.
- **Set TruncatePrunedTree flag to false:** If true, then pruned branches are physically removed from the tree.
- **Gradient Boosted Tree classifier:** This group of parameters allows setting Gradient Boosted Tree classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/gradient\\_boosted\\_trees.html}](http://docs.opencv.org/modules/ml/doc/gradient_boosted_trees.html).
- **Number of boosting algorithm iterations:** Number “w” of boosting algorithm iterations, with  $w * K$  being the total number of trees in the GBT model, where K is the output number of classes.
- **Regularization parameter:** Regularization parameter.
- **Portion of the whole training set used for each algorithm iteration:** Portion of the whole training set used for each algorithm iteration. The subset is generated randomly.
- **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Artificial Neural Network classifier:** This group of parameters allows setting Artificial Neural Network classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/neural\\_networks.html}](http://docs.opencv.org/modules/ml/doc/neural_networks.html).
  - **Train Method Type:** Type of training method for the multilayer perceptron (MLP) neural network. Available choices are:
    - **RPROP algorithm**

- **Back-propagation algorithm**
- **Number of neurons in each intermediate layer:** The number of neurons in each intermediate layer (excluding input and output layers).
- **Neuron activation function type:** Neuron activation function. Available choices are:
  - **Identity function**
  - **Symmetrical Sigmoid function**
  - **Gaussian function (Not completely supported)**
- **Alpha parameter of the activation function:** Alpha parameter of the activation function (used only with sigmoid and gaussian functions).
- **Beta parameter of the activation function:** Beta parameter of the activation function (used only with sigmoid and gaussian functions).
- **Strength of the weight gradient term in the BACKPROP method:** Strength of the weight gradient term in the BACKPROP method. The recommended value is about 0.1.
- **Strength of the momentum term (the difference between weights on the 2 previous iterations):** Strength of the momentum term (the difference between weights on the 2 previous iterations). This parameter provides some inertia to smooth the random fluctuations of the weights. It can vary from 0 (the feature is disabled) to 1 and beyond. The value 0.1 or so is good enough.
- **Initial value Delta\_0 of update-values Delta\_{ij} in RPROP method:** Initial value Delta\_0 of update-values Delta\_{ij} in RPROP method (default = 0.1).
- **Update-values lower limit Delta\_{min} in RPROP method:** Update-values lower limit Delta\_{min} in RPROP method. It must be positive (default = 1e-7).
- **Termination criteria:** Termination criteria. Available choices are:
  - **Maximum number of iterations**
  - **Epsilon**
  - **Max. iterations + Epsilon**
  - **Epsilon value used in the Termination criteria:** Epsilon value used in the Termination criteria.
  - **Maximum number of iterations used in the Termination criteria:** Maximum number of iterations used in the Termination criteria.
- **Normal Bayes classifier:** Use a Normal Bayes Classifier. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/normal\\_bayes\\_classifier.html}](http://docs.opencv.org/modules/ml/doc/normal_bayes_classifier.html).
- **Random forests classifier:** This group of parameters allows setting Random Forests classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/random\\_trees.html}](http://docs.opencv.org/modules/ml/doc/random_trees.html).
- **Maximum depth of the tree:** The depth of the tree. A low value will likely underfit and conversely a high value will likely overfit. The optimal value can be obtained using cross validation or other suitable methods.
- **Minimum number of samples in each node:** If the number of samples in a node is smaller than this parameter, then the node will not be split. A reasonable value is a small percentage of the total data e.g. 1 percent.
- **Termination Criteria for regression tree:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
- **Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split:** Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split.

- **Size of the randomly selected subset of features at each tree node:** The size of the subset of features, randomly selected at each tree node, that are used to find the best split(s). If you set it to 0, then the size will be set to the square root of the total number of features.
- **Maximum number of trees in the forest:** The maximum number of trees in the forest. Typically, the more trees you have, the better the accuracy. However, the improvement in accuracy generally diminishes and reaches an asymptote for a certain number of trees. Also to keep in mind, increasing the number of trees increases the prediction time linearly.
- **Sufficient accuracy (OOB error):** Sufficient accuracy (OOB error).
- **KNN classifier:** This group of parameters allows setting KNN classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/k\\_nearest\\_neighbors.html}](http://docs.opencv.org/modules/ml/doc/k_nearest_neighbors.html).
- **Number of Neighbors:** The number of neighbors to use.

**set user defined seed:** Set specific seed. with integer value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_TrainImagesClassifier -io.il QB_1_ortho.tif -io.vd VectorData_QB1.shp -io.imstat EstimateImageStatistics_QB1.xml
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the TrainImagesClassifier application
TrainImagesClassifier = otbApplication.Registry.CreateApplication("TrainImagesClassifier")

The following lines set all the application parameters:
TrainImagesClassifier.SetParameterStringList("io.il", ['QB_1_ortho.tif'])

TrainImagesClassifier.SetParameterStringList("io.vd", ['VectorData_QB1.shp'])

TrainImagesClassifier.SetParameterString("io.imstat", "EstimateImageStatistics_QB1.xml")

TrainImagesClassifier.SetParameterInt("sample.mv", 100)

TrainImagesClassifier.SetParameterInt("sample.mt", 100)

TrainImagesClassifier.SetParameterFloat("sample.vtr", 0.5)

TrainImagesClassifier.SetParameterString("sample.edg", "1")

TrainImagesClassifier.SetParameterString("sample.vfn", "Class")

TrainImagesClassifier.SetParameterString("classifier", "libsvm")

TrainImagesClassifier.SetParameterString("classifier.libsvm.k", "linear")

TrainImagesClassifier.SetParameterFloat("classifier.libsvm.c", 1)
```

```

TrainImagesClassifier.SetParameterString("classifier.libsvm.opt", "1")

TrainImagesClassifier.SetParameterString("io.out", "svmModelQB1.txt")

TrainImagesClassifier.SetParameterString("io.confmatout", "svmConfusionMatrixQB1.csv")

The following line execute the application
TrainImagesClassifier.ExecuteAndWriteOutput ()

```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

OpenCV documentation for machine learning <http://docs.opencv.org/modules/ml/doc/ml.html>

## 6.8.14 Train a regression model

Train a classifier from multiple images to perform regression.

### Detailed description

**This application trains a classifier from multiple input images or a csv file, in order to perform regression. Predictors are computed**

The output value for each predictor is assumed to be the last band (or the last column for CSV files). Training and validation predictor lists are built such that their size is inferior to maximum bounds given by the user, and the proportion corresponds to the balance parameter. Several classifier parameters can be set depending on the chosen classifier. In the validation process, the mean square error is computed This application is based on LibSVM and on OpenCV Machine Learning classifiers, and is compatible with OpenCV 2.3.1 and later.

### Parameters

This section describes in details the parameters available for this application. Table <sup>73</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TrainRegression* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.il	Input image list	Input image list
io.csv	Input File name	Input File name
Continued on next page		

<sup>73</sup> Table: Parameters table for Train a regression model.

Table 6.8 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
io.imstat	Input File name	Input File name
io.out	Output File name	Output File name
io.mse	Float	Float
sample	Group	Group
sample.mt	Int	Int
sample.mv	Int	Int
sample.vtr	Float	Float
classifier	Choices	Choices
classifier libsvm	<i>Choice</i>	LibSVM classifier
classifier dt	<i>Choice</i>	Decision Tree classifier
classifier gbt	<i>Choice</i>	Gradient Boosted Tree classifier
classifier ann	<i>Choice</i>	Artificial Neural Network classifier
classifier rf	<i>Choice</i>	Random forests classifier
classifier knn	<i>Choice</i>	KNN classifier
classifier.libsvm.k	Choices	Choices
classifier.libsvm.k linear	<i>Choice</i>	Linear
classifier.libsvm.k rbf	<i>Choice</i>	Gaussian radial basis function
classifier.libsvm.k poly	<i>Choice</i>	Polynomial
classifier.libsvm.k sigmoid	<i>Choice</i>	Sigmoid
classifier.libsvm.m	Choices	Choices
classifier.libsvm.m epsvr	<i>Choice</i>	Epsilon Support Vector Regression
classifier.libsvm.m nusvr	<i>Choice</i>	Nu Support Vector Regression
classifier.libsvm.c	Float	Float
classifier.libsvm.opt	Boolean	Boolean
classifier.libsvm.prob	Boolean	Boolean
classifier.libsvm.eps	Float	Float
classifier.libsvm.nu	Float	Float
classifier.dt.max	Int	Int
classifier.dt.min	Int	Int
classifier.dt.ra	Float	Float
classifier.dt.cat	Int	Int
classifier.dt.f	Int	Int
classifier.dt.r	Boolean	Boolean
classifier.dt.t	Boolean	Boolean
classifier.gbt.t	Choices	Choices
classifier.gbt.t sqr	<i>Choice</i>	Squared Loss
classifier.gbt.t abs	<i>Choice</i>	Absolute Loss
classifier.gbt.t hub	<i>Choice</i>	Huber Loss
classifier.gbt.w	Int	Int
classifier.gbt.s	Float	Float
classifier.gbt.p	Float	Float
classifier.gbt.max	Int	Int
classifier.ann.t	Choices	Choices
classifier.ann.t reg	<i>Choice</i>	RPROP algorithm
classifier.ann.t back	<i>Choice</i>	Back-propagation algorithm
classifier.ann.sizes	String list	String list
classifier.ann.f	Choices	Choices
classifier.ann.f ident	<i>Choice</i>	Identity function
classifier.ann.f sig	<i>Choice</i>	Symmetrical Sigmoid function

Continued on next page

Table 6.8 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
classifier.ann.f gau	<i>Choice</i>	Gaussian function (Not completely supported)
classifier.ann.a	Float	Float
classifier.ann.b	Float	Float
classifier.ann.bpdw	Float	Float
classifier.ann.bpms	Float	Float
classifier.ann.rdw	Float	Float
classifier.ann.rdwm	Float	Float
classifier.ann.term	Choices	Choices
classifier.ann.term iter	<i>Choice</i>	Maximum number of iterations
classifier.ann.term eps	<i>Choice</i>	Epsilon
classifier.ann.term all	<i>Choice</i>	Max. iterations + Epsilon
classifier.ann.eps	Float	Float
classifier.ann.iter	Int	Int
classifier.rf.max	Int	Int
classifier.rf.min	Int	Int
classifier.rf.ra	Float	Float
classifier.rf.cat	Int	Int
classifier.rf.var	Int	Int
classifier.rf.nbtrees	Int	Int
classifier.rf.acc	Float	Float
classifier.knn.k	Int	Int
classifier.knn.rule	Choices	Choices
classifier.knn.rule mean	<i>Choice</i>	Mean of neighbors values
classifier.knn.rule median	<i>Choice</i>	Median of neighbors values
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting input and output data.

- **Input Image List:** A list of input images. First (n-1) bands should contain the predictor. The last band should contain the output value to predict.
- **Input CSV file:** Input CSV file containing the predictors, and the output values in last column. Only used when no input image is given.
- **Input XML image statistics file:** Input XML file containing the mean and the standard deviation of the input images.
- **Output regression model:** Output file containing the model estimated (.txt format).
- **Mean Square Error:** Mean square error computed with the validation predictors.

**[Training and validation samples parameters]:** This group of parameters allows you to set training and validation sample lists parameters.

- **Maximum training predictors:** Maximum number of training predictors (default = 1000) (no limit = -1).
- **Maximum validation predictors:** Maximum number of validation predictors (default = 1000) (no limit = -1).
- **Training and validation sample ratio:** Ratio between training and validation samples (0.0 = all training, 1.0 = all validation) (default = 0.5).

**Classifier to use for the training:** Choice of the classifier to use for the training. Available choices are:

- **LibSVM classifier:** This group of parameters allows setting SVM classifier parameters.

- **SVM Kernel Type:** SVM Kernel Type. Available choices are:
  - **Linear**
  - **Gaussian radial basis function**
  - **Polynomial**
  - **Sigmoid**
- **SVM Model Type:** Type of SVM formulation. Available choices are:
  - **Epsilon Support Vector Regression**
  - **Nu Support Vector Regression**
- **Cost parameter C:** SVM models have a cost parameter C (1 by default) to control the trade-off between training errors and forcing rigid margins.
- **Parameters optimization:** SVM parameters optimization flag.
- **Probability estimation:** Probability estimation flag.
- **Epsilon**
- **Nu**
- **Decision Tree classifier:** This group of parameters allows setting Decision Tree classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/decision\\_trees.html}](http://docs.opencv.org/modules/ml/doc/decision_trees.html).
- **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Minimum number of samples in each node:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
- **Termination criteria for regression tree**
- **Cluster possible values of a categorical variable into  $K \leq \text{cat clusters}$  to find a suboptimal split:** Cluster possible values of a categorical variable into  $K \leq \text{cat clusters}$  to find a suboptimal split.
- **K-fold cross-validations:** If  $\text{cv\_folds} > 1$ , then it prunes a tree with K-fold cross-validation where K is equal to  $\text{cv\_folds}$ .
- **Set UseIsRule flag to false:** If true, then a pruning will be harsher. This will make a tree more compact and more resistant to the training data noise but a bit less accurate.
- **Set TruncatePrunedTree flag to false:** If true, then pruned branches are physically removed from the tree.
- **Gradient Boosted Tree classifier:** This group of parameters allows setting Gradient Boosted Tree classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/gradient\\_boosted\\_trees.html}](http://docs.opencv.org/modules/ml/doc/gradient_boosted_trees.html).
  - **Loss Function Type:** Type of loss function used for training. Available choices are:
    - **Squared Loss**
    - **Absolute Loss**
    - **Huber Loss**
  - **Number of boosting algorithm iterations:** Number “w” of boosting algorithm iterations, with  $w \cdot K$  being the total number of trees in the GBT model, where K is the output number of classes.
  - **Regularization parameter:** Regularization parameter.

- **Portion of the whole training set used for each algorithm iteration:** Portion of the whole training set used for each algorithm iteration. The subset is generated randomly.
- **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Artificial Neural Network classifier:** This group of parameters allows setting Artificial Neural Network classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/neural\\_networks.html}](http://docs.opencv.org/modules/ml/doc/neural_networks.html).
  - **Train Method Type:** Type of training method for the multilayer perceptron (MLP) neural network. Available choices are:
    - **RPROP algorithm**
    - **Back-propagation algorithm**
  - **Number of neurons in each intermediate layer:** The number of neurons in each intermediate layer (excluding input and output layers).
  - **Neuron activation function type:** Neuron activation function. Available choices are:
    - **Identity function**
    - **Symmetrical Sigmoid function**
    - **Gaussian function (Not completely supported)**
  - **Alpha parameter of the activation function:** Alpha parameter of the activation function (used only with sigmoid and gaussian functions).
  - **Beta parameter of the activation function:** Beta parameter of the activation function (used only with sigmoid and gaussian functions).
  - **Strength of the weight gradient term in the BACKPROP method:** Strength of the weight gradient term in the BACKPROP method. The recommended value is about 0.1.
  - **Strength of the momentum term (the difference between weights on the 2 previous iterations):** Strength of the momentum term (the difference between weights on the 2 previous iterations). This parameter provides some inertia to smooth the random fluctuations of the weights. It can vary from 0 (the feature is disabled) to 1 and beyond. The value 0.1 or so is good enough.
  - **Initial value Delta\_0 of update-values Delta\_{ij} in RPROP method:** Initial value Delta\_0 of update-values Delta\_{ij} in RPROP method (default = 0.1).
  - **Update-values lower limit Delta\_{min} in RPROP method:** Update-values lower limit Delta\_{min} in RPROP method. It must be positive (default = 1e-7).
  - **Termination criteria:** Termination criteria. Available choices are:
    - **Maximum number of iterations**
    - **Epsilon**
    - **Max. iterations + Epsilon**
  - **Epsilon value used in the Termination criteria:** Epsilon value used in the Termination criteria.
  - **Maximum number of iterations used in the Termination criteria:** Maximum number of iterations used in the Termination criteria.
- **Random forests classifier:** This group of parameters allows setting Random Forests classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/random\\_trees.html}](http://docs.opencv.org/modules/ml/doc/random_trees.html).

- **Maximum depth of the tree:** The depth of the tree. A low value will likely underfit and conversely a high value will likely overfit. The optimal value can be obtained using cross validation or other suitable methods.
- **Minimum number of samples in each node:** If the number of samples in a node is smaller than this parameter, then the node will not be split. A reasonable value is a small percentage of the total data e.g. 1 percent.
- **Termination Criteria for regression tree:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
- **Cluster possible values of a categorical variable into  $K \leq$  cat clusters to find a suboptimal split:** Cluster possible values of a categorical variable into  $K \leq$  cat clusters to find a suboptimal split.
- **Size of the randomly selected subset of features at each tree node:** The size of the subset of features, randomly selected at each tree node, that are used to find the best split(s). If you set it to 0, then the size will be set to the square root of the total number of features.
- **Maximum number of trees in the forest:** The maximum number of trees in the forest. Typically, the more trees you have, the better the accuracy. However, the improvement in accuracy generally diminishes and reaches an asymptote for a certain number of trees. Also to keep in mind, increasing the number of trees increases the prediction time linearly.
- **Sufficient accuracy (OOB error):** Sufficient accuracy (OOB error).
- **KNN classifier:** This group of parameters allows setting KNN classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/k\\_nearest\\_neighbors.html}](http://docs.opencv.org/modules/ml/doc/k_nearest_neighbors.html).
  - **Number of Neighbors:** The number of neighbors to use.
  - **Decision rule:** Decision rule for regression output. Available choices are:
  - **Mean of neighbors values:** Returns the mean of neighbors values.
  - **Median of neighbors values:** Returns the median of neighbors values.

**set user defined seed:** Set specific seed. with integer value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_TrainRegression -io.il training_dataset.tif -io.out regression_model.txt -io.imstat training_s
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the TrainRegression application
TrainRegression = otbApplication.Registry.CreateApplication("TrainRegression")

The following lines set all the application parameters:
TrainRegression.SetParameterStringList("io.il", ['training_dataset.tif'])

TrainRegression.SetParameterString("io.out", "regression_model.txt")
```

```
TrainRegression.SetParameterString("io.imstat", "training_statistics.xml")

TrainRegression.SetParameterString("classifier", "libsvm")

The following line execute the application
TrainRegression.ExecuteAndWriteOutput ()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

OpenCV documentation for machine learning <http://docs.opencv.org/modules/ml/doc/ml.html>

## 6.8.15 Train Vector Classifier

Train a classifier based on labeled geometries and a list of features to consider.

### Detailed description

This application trains a classifier based on labeled geometries and a list of features to consider for classification.

### Parameters

This section describes in details the parameters available for this application. Table <sup>74</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TrainVectorClassifier* .

Parameter Key	Parameter Type	Parameter Description
io	Group	Group
io.vd	Input vector data list	Input vector data list
io.stats	Input File name	Input File name
io.confmatout	Output File name	Output File name
io.out	Output File name	Output File name
feat	List	List
cfield	String	String
layer	Int	Int
valid	Group	Group
valid.vd	Input vector data list	Input vector data list

---

<sup>74</sup> Table: Parameters table for Train Vector Classifier.

Table 6.9 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
valid.layer	Int	Int
classifier	Choices	Choices
classifier libsvm	<i>Choice</i>	LibSVM classifier
classifier boost	<i>Choice</i>	Boost classifier
classifier dt	<i>Choice</i>	Decision Tree classifier
classifier gbt	<i>Choice</i>	Gradient Boosted Tree classifier
classifier ann	<i>Choice</i>	Artificial Neural Network classifier
classifier bayes	<i>Choice</i>	Normal Bayes classifier
classifier rf	<i>Choice</i>	Random forests classifier
classifier knn	<i>Choice</i>	KNN classifier
classifier.libsvm.k	Choices	Choices
classifier.libsvm.k linear	<i>Choice</i>	Linear
classifier.libsvm.k rbf	<i>Choice</i>	Gaussian radial basis function
classifier.libsvm.k poly	<i>Choice</i>	Polynomial
classifier.libsvm.k sigmoid	<i>Choice</i>	Sigmoid
classifier.libsvm.m	Choices	Choices
classifier.libsvm.m csvc	<i>Choice</i>	C support vector classification
classifier.libsvm.m nusvc	<i>Choice</i>	Nu support vector classification
classifier.libsvm.m oneclass	<i>Choice</i>	Distribution estimation (One Class SVM)
classifier.libsvm.c	Float	Float
classifier.libsvm.opt	Boolean	Boolean
classifier.libsvm.prob	Boolean	Boolean
classifier.boost.t	Choices	Choices
classifier.boost.t discrete	<i>Choice</i>	Discrete AdaBoost
classifier.boost.t real	<i>Choice</i>	Real AdaBoost (technique using confidence-rated predictions and worki
classifier.boost.t logit	<i>Choice</i>	LogitBoost (technique producing good regression fits)
classifier.boost.t gentle	<i>Choice</i>	Gentle AdaBoost (technique setting less weight on outlier data points ar
classifier.boost.w	Int	Int
classifier.boost.r	Float	Float
classifier.boost.m	Int	Int
classifier.dt.max	Int	Int
classifier.dt.min	Int	Int
classifier.dt.ra	Float	Float
classifier.dt.cat	Int	Int
classifier.dt.f	Int	Int
classifier.dt.r	Boolean	Boolean
classifier.dt.t	Boolean	Boolean
classifier.gbt.w	Int	Int
classifier.gbt.s	Float	Float
classifier.gbt.p	Float	Float
classifier.gbt.max	Int	Int
classifier.ann.t	Choices	Choices
classifier.ann.t reg	<i>Choice</i>	RPROP algorithm
classifier.ann.t back	<i>Choice</i>	Back-propagation algorithm
classifier.ann.sizes	String list	String list
classifier.ann.f	Choices	Choices
classifier.ann.f ident	<i>Choice</i>	Identity function
classifier.ann.f sig	<i>Choice</i>	Symmetrical Sigmoid function
classifier.ann.f gau	<i>Choice</i>	Gaussian function (Not completely supported)

Table 6.9 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
classifier.ann.a	Float	Float
classifier.ann.b	Float	Float
classifier.ann.bpdw	Float	Float
classifier.ann.bpms	Float	Float
classifier.ann.rdw	Float	Float
classifier.ann.rdwm	Float	Float
classifier.ann.term	Choices	Choices
classifier.ann.term iter	<i>Choice</i>	Maximum number of iterations
classifier.ann.term eps	<i>Choice</i>	Epsilon
classifier.ann.term all	<i>Choice</i>	Max. iterations + Epsilon
classifier.ann.eps	Float	Float
classifier.ann.iter	Int	Int
classifier.rf.max	Int	Int
classifier.rf.min	Int	Int
classifier.rf.ra	Float	Float
classifier.rf.cat	Int	Int
classifier.rf.var	Int	Int
classifier.rf.nbtrees	Int	Int
classifier.rf.acc	Float	Float
classifier.knn.k	Int	Int
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**[Input and output data]:** This group of parameters allows setting input and output data.

- **Input Vector Data:** Input geometries used for training (note : all geometries from the layer will be used).
- **Input XML image statistics file:** XML file containing mean and variance of each feature.
- **Output confusion matrix:** Output file containing the confusion matrix (.csv format).
- **Output model:** Output file containing the model estimated (.txt format).

**Field names for training features.:** List of field names in the input vector data to be used as features for training.

**Field containing the class id for supervision:** Field containing the class id for supervision. Only geometries with this field available will be taken into account.

**Layer Index:** Index of the layer to use in the input vector file.

**[Validation data]:** This group of parameters defines validation data.

- **Validation Vector Data:** Geometries used for validation (must contain the same fields used for training, all geometries from the layer will be used).
- **Layer Index:** Index of the layer to use in the validation vector file.

**Classifier to use for the training:** Choice of the classifier to use for the training. Available choices are:

- **LibSVM classifier:** This group of parameters allows setting SVM classifier parameters.
  - **SVM Kernel Type:** SVM Kernel Type. Available choices are:
    - **Linear**
    - **Gaussian radial basis function**

- **Polynomial**
- **Sigmoid**
- **SVM Model Type:** Type of SVM formulation. Available choices are:
  - **C support vector classification**
  - **Nu support vector classification**
  - **Distribution estimation (One Class SVM)**
  - **Cost parameter C:** SVM models have a cost parameter C (1 by default) to control the trade-off between training errors and forcing rigid margins.
  - **Parameters optimization:** SVM parameters optimization flag.
  - **Probability estimation:** Probability estimation flag.
- **Boost classifier:** This group of parameters allows setting Boost classifier parameters. See complete documentation here url{<http://docs.opencv.org/modules/ml/doc/boosting.html>}.
  - **Boost Type:** Type of Boosting algorithm. Available choices are:
    - **Discrete AdaBoost**
    - **Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)**
    - **LogitBoost (technique producing good regression fits)**
    - **Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often good with regression data)**
  - **Weak count:** The number of weak classifiers.
  - **Weight Trim Rate:** A threshold between 0 and 1 used to save computational time. Samples with summary weight  $\leq (1 - \text{weight\_trim\_rate})$  do not participate in the next iteration of training. Set this parameter to 0 to turn off this functionality.
  - **Maximum depth of the tree:** Maximum depth of the tree.
- **Decision Tree classifier:** This group of parameters allows setting Decision Tree classifier parameters. See complete documentation here url{[http://docs.opencv.org/modules/ml/doc/decision\\_trees.html](http://docs.opencv.org/modules/ml/doc/decision_trees.html)}.
- **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Minimum number of samples in each node:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
- **Termination criteria for regression tree**
- **Cluster possible values of a categorical variable into  $K \leq \text{cat}$  clusters to find a suboptimal split:** Cluster possible values of a categorical variable into  $K \leq \text{cat}$  clusters to find a suboptimal split.
- **K-fold cross-validations:** If  $\text{cv\_folds} > 1$ , then it prunes a tree with K-fold cross-validation where K is equal to  $\text{cv\_folds}$ .
- **Set UseIsRule flag to false:** If true, then a pruning will be harsher. This will make a tree more compact and more resistant to the training data noise but a bit less accurate.
- **Set TruncatePrunedTree flag to false:** If true, then pruned branches are physically removed from the tree.

- **Gradient Boosted Tree classifier:** This group of parameters allows setting Gradient Boosted Tree classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/gradient\\_boosted\\_trees.html}](http://docs.opencv.org/modules/ml/doc/gradient_boosted_trees.html).
- **Number of boosting algorithm iterations:** Number “w” of boosting algorithm iterations, with  $w \cdot K$  being the total number of trees in the GBT model, where K is the output number of classes.
- **Regularization parameter:** Regularization parameter.
- **Portion of the whole training set used for each algorithm iteration:** Portion of the whole training set used for each algorithm iteration. The subset is generated randomly.
- **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Artificial Neural Network classifier:** This group of parameters allows setting Artificial Neural Network classifier parameters. See complete documentation here [url{http://docs.opencv.org/modules/ml/doc/neural\\_networks.html}](http://docs.opencv.org/modules/ml/doc/neural_networks.html).
  - **Train Method Type:** Type of training method for the multilayer perceptron (MLP) neural network. Available choices are:
    - **RPROP algorithm**
    - **Back-propagation algorithm**
  - **Number of neurons in each intermediate layer:** The number of neurons in each intermediate layer (excluding input and output layers).
  - **Neuron activation function type:** Neuron activation function. Available choices are:
    - **Identity function**
    - **Symmetrical Sigmoid function**
    - **Gaussian function (Not completely supported)**
  - **Alpha parameter of the activation function:** Alpha parameter of the activation function (used only with sigmoid and gaussian functions).
  - **Beta parameter of the activation function:** Beta parameter of the activation function (used only with sigmoid and gaussian functions).
  - **Strength of the weight gradient term in the BACKPROP method:** Strength of the weight gradient term in the BACKPROP method. The recommended value is about 0.1.
  - **Strength of the momentum term (the difference between weights on the 2 previous iterations):** Strength of the momentum term (the difference between weights on the 2 previous iterations). This parameter provides some inertia to smooth the random fluctuations of the weights. It can vary from 0 (the feature is disabled) to 1 and beyond. The value 0.1 or so is good enough.
  - **Initial value Delta\_0 of update-values Delta\_{ij} in RPROP method:** Initial value Delta\_0 of update-values Delta\_{ij} in RPROP method (default = 0.1).
  - **Update-values lower limit Delta\_{min} in RPROP method:** Update-values lower limit Delta\_{min} in RPROP method. It must be positive (default = 1e-7).
  - **Termination criteria:** Termination criteria. Available choices are:
    - **Maximum number of iterations**
    - **Epsilon**
    - **Max. iterations + Epsilon**

- **Epsilon value used in the Termination criteria:** Epsilon value used in the Termination criteria.
- **Maximum number of iterations used in the Termination criteria:** Maximum number of iterations used in the Termination criteria.
- **Normal Bayes classifier:** Use a Normal Bayes Classifier. See complete documentation here url{[http://docs.opencv.org/modules/ml/doc/normal\\_bayes\\_classifier.html](http://docs.opencv.org/modules/ml/doc/normal_bayes_classifier.html)}.
- **Random forests classifier:** This group of parameters allows setting Random Forests classifier parameters. See complete documentation here url{[http://docs.opencv.org/modules/ml/doc/random\\_trees.html](http://docs.opencv.org/modules/ml/doc/random_trees.html)}.
- **Maximum depth of the tree:** The depth of the tree. A low value will likely underfit and conversely a high value will likely overfit. The optimal value can be obtained using cross validation or other suitable methods.
- **Minimum number of samples in each node:** If the number of samples in a node is smaller than this parameter, then the node will not be split. A reasonable value is a small percentage of the total data e.g. 1 percent.
- **Termination Criteria for regression tree:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
- **Cluster possible values of a categorical variable into  $K \leq$  cat clusters to find a suboptimal split:** Cluster possible values of a categorical variable into  $K \leq$  cat clusters to find a suboptimal split.
- **Size of the randomly selected subset of features at each tree node:** The size of the subset of features, randomly selected at each tree node, that are used to find the best split(s). If you set it to 0, then the size will be set to the square root of the total number of features.
- **Maximum number of trees in the forest:** The maximum number of trees in the forest. Typically, the more trees you have, the better the accuracy. However, the improvement in accuracy generally diminishes and reaches an asymptote for a certain number of trees. Also to keep in mind, increasing the number of trees increases the prediction time linearly.
- **Sufficient accuracy (OOB error):** Sufficient accuracy (OOB error).
- **KNN classifier:** This group of parameters allows setting KNN classifier parameters. See complete documentation here url{[http://docs.opencv.org/modules/ml/doc/k\\_nearest\\_neighbors.html](http://docs.opencv.org/modules/ml/doc/k_nearest_neighbors.html)}.
- **Number of Neighbors:** The number of neighbors to use.

**set user defined seed:** Set specific seed. with integer value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_TrainVectorClassifier -io.vd vectorData.shp -io.stats meanVar.xml -io.out svmModel.svm -feat p
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the TrainVectorClassifier application
TrainVectorClassifier = otbApplication.Registry.CreateApplication("TrainVectorClassifier")
```

```
The following lines set all the application parameters:
TrainVectorClassifier.SetParameterStringList("io.vd", ['vectorData.shp'])

TrainVectorClassifier.SetParameterString("io.stats", "meanVar.xml")

TrainVectorClassifier.SetParameterString("io.out", "svmModel.svm")

The following line execute the application
TrainVectorClassifier.ExecuteAndWriteOutput()
```

## Authors

This application has been written by OTB Team.

## 6.9 Segmentation

### 6.9.1 ComputeOGRLayersFeaturesStatistics

Compute statistics of the features in a set of OGR Layers

#### Detailed description

Compute statistics (mean and standard deviation) of the features in a set of OGR Layers, and write them in an XML file. This XML file can then be used by the training application.

#### Parameters

This section describes in details the parameters available for this application. Table <sup>75</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ComputeOGRLayersFeaturesStatistics* .

Parameter Key	Parameter Type	Parameter Description
inshp	Input vector data	Input vector data
outstats	Output File name	Output File name
feat	List	List
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Name of the input shapefile:** Name of the input shapefile.
- **XML file containing mean and variance of each feature.:** XML file containing mean and variance of each feature.
- **List of features to consider for statistics.:** List of features to consider for statistics.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

<sup>75</sup> Table: Parameters table for ComputeOGRLayersFeaturesStatistics.

## Example

To run this example in command-line, use the following:

```
otbcli_ComputeOGRLayersFeaturesStatistics -inshp vectorData.shp -outstats results.xml -feat perimeter
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ComputeOGRLayersFeaturesStatistics application
ComputeOGRLayersFeaturesStatistics = otbApplication.Registry.CreateApplication("ComputeOGRLayersFeatu

The following lines set all the application parameters:
ComputeOGRLayersFeaturesStatistics.SetParameterString("inshp", "vectorData.shp")

ComputeOGRLayersFeaturesStatistics.SetParameterString("outstats", "results.xml")

The following line execute the application
ComputeOGRLayersFeaturesStatistics.ExecuteAndWriteOutput()
```

## Limitations

Experimental. For now only shapefiles are supported.

## Authors

This application has been written by David Youssefi during internship at CNES.

## See Also

These additional resources can be useful for further information:

OGRLayerClassifier, TrainOGRLayersClassifier

## 6.9.2 Connected Component Segmentation

Connected component segmentation and object based image filtering of the input image according to user-defined criterions.

### Detailed description

This application allows one to perform a masking, connected components segmentation and object based image filtering. First and optionally, a mask can be built based on user-defined criterions to select pixels of the image which will be segmented. Then a connected component segmentation is performed with a user defined criterion to decide whether two neighbouring pixels belong to the same segment or not. After this segmentation step, an object based image filtering is applied using another user-defined criterion reasoning on segment properties, like shape or radiometric attributes. Criterions are mathematical expressions analysed by the MuParser library (<http://muparser.sourceforge.net/>). For instance, expression “((b1>80) and intensity>95)” will merge two neighbouring pixel in a single segment if their

intensity is more than 95 and their value in the first image band is more than 80. See parameters documentation for a list of available attributes. The output of the object based image filtering is vectorized and can be written in shapefile or KML format. If the input image is in raw geometry, resulting polygons will be transformed to WGS84 using sensor modelling before writing, to ensure consistency with GIS software. For this purpose, a Digital Elevation Model can be provided to the application. The whole processing is done on a per-tile basis for large images, so this application can handle images of arbitrary size.

## Parameters

This section describes in details the parameters available for this application. Table <sup>76</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConnectedComponentSegmentation* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output vector data	Output vector data
mask	String	String
expr	String	String
minsize	Int	Int
obia	String	String
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The image to segment.

**Output Shape:** The segmentation shape.

**Mask expression:** Mask mathematical expression (only if support image is given).

**Connected Component Expression:** Formula used for connected component segmentation.

**Minimum Object Size:** Min object size (area in pixel).

**OBIA Expression:** OBIA mathematical expression.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

<sup>76</sup> Table: Parameters table for Connected Component Segmentation.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_ConnectedComponentSegmentation -in ROI_QB_MUL_4.tif -mask "(b1>80)*intensity>95" -expr "distance<10"
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ConnectedComponentSegmentation application
ConnectedComponentSegmentation = otbApplication.Registry.CreateApplication("ConnectedComponentSegmentation")

The following lines set all the application parameters:
ConnectedComponentSegmentation.SetParameterString("in", "ROI_QB_MUL_4.tif")

ConnectedComponentSegmentation.SetParameterString("mask", "(b1>80)*intensity>95")

ConnectedComponentSegmentation.SetParameterString("expr", "distance<10")

ConnectedComponentSegmentation.SetParameterInt("minsize", 15)

ConnectedComponentSegmentation.SetParameterString("obia", "SHAPE_Elongation>8")

ConnectedComponentSegmentation.SetParameterString("out", "ConnectedComponentSegmentation.shp")

The following line execute the application
ConnectedComponentSegmentation.ExecuteAndWriteOutput()
```

## Limitations

Due to the tiling scheme in case of large images, some segments can be arbitrarily split across multiple tiles.

## Authors

This application has been written by OTB-Team.

## 6.9.3 Hoover compare segmentation

Compare two segmentations with Hoover metrics

### Detailed description

**This application compares a machine segmentation (MS) with a partial ground truth segmentation (GT). The Hoover metrics are calculated for each segment.**  
The application can output the overall Hoover scores along with colored images of the MS and GT segmentation

showing the state of each region (correct detection, over-segmentation, under-segmentation, missed) The Hoover metrics are described in : Hoover et al., “An experimental comparison of range image segmentation algorithms”, IEEE PAMI vol. 18, no. 7, July 1996.

## Parameters

This section describes in details the parameters available for this application. Table <sup>77</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *HooverCompareSegmentation* .

Parameter Key	Parameter Type	Parameter Description
ingt	Input image	Input image
inms	Input image	Input image
bg	Int	Int
th	Float	Float
outgt	Output image	Output image
outms	Output image	Output image
rc	Float	Float
rf	Float	Float
ra	Float	Float
rm	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input ground truth:** A partial ground truth segmentation image.
- **Input machine segmentation:** A machine segmentation image.
- **Background label:** Label value of the background in the input segmentations.
- **Overlapping threshold:** Overlapping threshold used to find Hoover instances.
- **Colored ground truth output:** The colored ground truth output image.
- **Colored machine segmentation output:** The colored machine segmentation output image.
- **Correct detection score:** Overall score for correct detection (RC).
- **Over-segmentation score:** Overall score for over segmentation (RF).
- **Under-segmentation score:** Overall score for under segmentation (RA).
- **Missed detection score:** Overall score for missed detection (RM).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_HooverCompareSegmentation -ingt maur_GT.tif -inms maur_labelled.tif -outgt maur_colored_GT.tif
```

To run this example from Python, use the following code snippet:

<sup>77</sup> Table: Parameters table for Hoover compare segmentation.

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the HooverCompareSegmentation application
HooverCompareSegmentation = otbApplication.Registry.CreateApplication("HooverCompareSegmentation")

The following lines set all the application parameters:
HooverCompareSegmentation.SetParameterString("ingt", "maur_GT.tif")

HooverCompareSegmentation.SetParameterString("inms", "maur_labelled.tif")

HooverCompareSegmentation.SetParameterString("outgt", "maur_colored_GT.tif")
HooverCompareSegmentation.SetParameterOutputImagePixelType("outgt", 1)

The following line execute the application
HooverCompareSegmentation.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

[otbHooverMatrixFilter](#), [otbHooverInstanceFilter](#), [otbLabelMapToAttributeImageFilter](#)

## 6.9.4 Exact Large-Scale Mean-Shift segmentation, step 2

Second step of the exact Large-Scale Mean-Shift segmentation workflow.

### Detailed description

This application performs the second step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS). Filtered range image and spatial image should be created with the `MeanShiftSmoothing` application, with `modesearch` parameter disabled. If spatial image is not set, the application will only process the range image and spatial radius parameter will not be taken into account. This application will produce a labeled image where neighbor pixels whose range distance is below range radius (and optionally spatial distance below spatial radius) will be grouped together into the same cluster. For large images one can use the `nbtilsx` and `nbtilesy` parameters for tile-wise processing, with the guarantees of identical results. Please note that this application will generate a lot of temporary files (as many as the number of tiles), and will therefore require twice the size of the final result in term of disk space. The `cleanup` option (activated by default) allows removing all temporary file as soon as they are not needed anymore (if `cleanup` is activated, `tmpdir` set and `tmpdir` does not exists before running the application, it will be removed as well during cleanup). The `tmpdir` option allows defining a directory where to write the temporary files. Please also note that the output image type should be set to `uint32` to ensure that there are enough labels available.

## Parameters

This section describes in details the parameters available for this application. Table <sup>78</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LSMSSegmentation*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
inpos	Input image	Input image
out	Output image	Output image
spatialr	Float	Float
ranger	Float	Float
minsize	Int	Int
tilesizeX	Int	Int
tilesizeY	Int	Int
tmpdir	Directory	Directory
cleanup	Boolean	Boolean
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Filtered image:** The filtered image (cf. Adaptive MeanShift Smoothing application).
- **Spatial image:** The spatial image. Spatial input is the displacement map (output of the Adaptive MeanShift Smoothing application).
- **Output Image:** The output image. The output image is the segmentation of the filtered image. It is recommended to set the pixel type to uint32.
- **Spatial radius:** Spatial radius of the neighborhood.
- **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multi-spectral space.
- **Minimum Region Size:** Minimum Region Size. If, after the segmentation, a region is of size lower than this criterion, the region is deleted.
- **Size of tiles in pixel (X-axis):** Size of tiles along the X-axis.
- **Size of tiles in pixel (Y-axis):** Size of tiles along the Y-axis.
- **Directory where to write temporary files:** This applications need to write temporary files for each tile. This parameter allows choosing the path where to write those files. If disabled, the current path will be used.
- **Temporary files cleaning:** If activated, the application will try to clean all temporary files it created.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_LSMSSegmentation -in smooth.tif -inpos position.tif -out segmentation.tif -spatialr 5 -ranger
```

To run this example from Python, use the following code snippet:

<sup>78</sup> Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 2.

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the LSMS Segmentation application
LSMS Segmentation = otbApplication.Registry.CreateApplication("LSMS Segmentation")

The following lines set all the application parameters:
LSMS Segmentation.SetParameterString("in", "smooth.tif")

LSMS Segmentation.SetParameterString("inpos", "position.tif")

LSMS Segmentation.SetParameterString("out", "segmentation.tif")

LSMS Segmentation.SetParameterFloat("spatialr", 5)

LSMS Segmentation.SetParameterFloat("ranger", 15)

LSMS Segmentation.SetParameterInt("minsize", 0)

LSMS Segmentation.SetParameterInt("tilesizeX", 256)

LSMS Segmentation.SetParameterInt("tilesizeY", 256)

The following line execute the application
LSMS Segmentation.ExecuteAndWriteOutput()
```

## Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) and may not be suited for any other purpose.

## Authors

This application has been written by David Youssefi.

## See Also

These additional resources can be useful for further information:

MeanShiftSmoothing, LSMS SmallRegionsMerging, LSMS Vectorization

## 6.9.5 Exact Large-Scale Mean-Shift segmentation, step 3 (optional)

Third (optional) step of the exact Large-Scale Mean-Shift segmentation workflow.

### Detailed description

This application performs the third step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS). Given a segmentation result (label image) and the original image, it will merge regions whose size in pixels is lower than minsize parameter with the adjacent regions with the adjacent region with closest radiometry and acceptable size.

Small regions will be processed by size: first all regions of area, which is equal to 1 pixel will be merged with adjacent region, then all regions of area equal to 2 pixels, until regions of area minsize. For large images one can use the `nbtilex` and `nbtiley` parameters for tile-wise processing, with the guarantees of identical results.

## Parameters

This section describes in details the parameters available for this application. Table <sup>79</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `LSMSSmallRegionsMerging`.

Parameter Key	Parameter Type	Parameter Description
<code>in</code>	Input image	Input image
<code>inseg</code>	Input image	Input image
<code>out</code>	Output image	Output image
<code>minsize</code>	Int	Int
<code>tilesizeX</code>	Int	Int
<code>tilesizeY</code>	Int	Int
<code>ram</code>	Int	Int
<code>inxml</code>	XML input parameters file	XML input parameters file
<code>outxml</code>	XML output parameters file	XML output parameters file

- **Input image:** The input image.
- **Segmented image:** The segmented image input. Segmented image input is the segmentation of the input image.
- **Output Image:** The output image. The output image is the input image where the minimal regions have been merged.
- **Minimum Region Size:** Minimum Region Size. If, after the segmentation, a region is of size lower than this criterion, the region is merged with the “nearest” region (radiometrically).
- **Size of tiles in pixel (X-axis):** Size of tiles along the X-axis.
- **Size of tiles in pixel (Y-axis):** Size of tiles along the Y-axis.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_LSMSSmallRegionsMerging -in smooth.tif -inseg segmentation.tif -out merged.tif -minsize 20 -t
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the LSMSSmallRegionsMerging application
LSMSSmallRegionsMerging = otbApplication.Registry.CreateApplication("LSMSSmallRegionsMerging")
```

<sup>79</sup> Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 3 (optional).

```
The following lines set all the application parameters:
LSMSSmallRegionsMerging.SetParameterString("in", "smooth.tif")

LSMSSmallRegionsMerging.SetParameterString("inseg", "segmentation.tif")

LSMSSmallRegionsMerging.SetParameterString("out", "merged.tif")

LSMSSmallRegionsMerging.SetParameterInt("minsize", 20)

LSMSSmallRegionsMerging.SetParameterInt("tilesex", 256)

LSMSSmallRegionsMerging.SetParameterInt("tilesey", 256)

The following line execute the application
LSMSSmallRegionsMerging.ExecuteAndWriteOutput()
```

### Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) and may not be suited for any other purpose.

### Authors

This application has been written by David Youssefi.

### See Also

These additional resources can be useful for further information:

LSMSSegmentation, LSMSVectorization, MeanShiftSmoothing

## 6.9.6 Exact Large-Scale Mean-Shift segmentation, step 4

Fourth step of the exact Large-Scale Mean-Shift segmentation workflow.

### Detailed description

This application performs the fourth step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS). Given a segmentation result (label image), that may have been processed for small regions merging or not, it will convert it to a GIS vector file containing one polygon per segment. Each polygon contains additional fields: mean and variance of each channels from input image (in parameter), segmentation image label, number of pixels in the polygon. For large images one can use the `nbtilsx` and `nbtilexy` parameters for tile-wise processing, with the guarantees of identical results.

### Parameters

This section describes in details the parameters available for this application. Table <sup>80</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LSMSVectorization*.

<sup>80</sup> Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 4.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
inseg	Input image	Input image
out	Output File name	Output File name
tilesex	Int	Int
tilesey	Int	Int
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** The input image.
- **Segmented image:** The segmented image input. Segmented image input is the segmentation of the input image.
- **Output GIS vector file:** The output GIS vector file, representing the vectorized version of the segmented image where the features of the polygons are the radiometric means and variances.
- **Size of tiles in pixel (X-axis):** Size of tiles along the X-axis.
- **Size of tiles in pixel (Y-axis):** Size of tiles along the Y-axis.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_LSMSVectorization -in avions.tif -inseg merged.tif -out vector.shp -tilesex 256 -tilesey 256
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the LSMSVectorization application
LSMSVectorization = otbApplication.Registry.CreateApplication("LSMSVectorization")

The following lines set all the application parameters:
LSMSVectorization.SetParameterString("in", "avions.tif")

LSMSVectorization.SetParameterString("inseg", "merged.tif")

LSMSVectorization.SetParameterString("out", "vector.shp")

LSMSVectorization.SetParameterInt("tilesex", 256)

LSMSVectorization.SetParameterInt("tilesey", 256)

The following line execute the application
LSMSVectorization.ExecuteAndWriteOutput()
```

## Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) and may not be suited for any other purpose.

## Authors

This application has been written by David Youssefi.

## See Also

These additional resources can be useful for further information:

MeanShiftSmoothing, LSMSSegmentation, LSMSSmallRegionsMerging

## 6.9.7 OGRLayerClassifier

Classify an OGR layer based on a machine learning model and a list of features to consider.

### Detailed description

This application will apply a trained machine learning model on the selected feature to get a classification of each geometry contained in an OGR layer. The list of feature must match the list used for training. The predicted label is written in the user defined field for each geometry.

### Parameters

This section describes in details the parameters available for this application. Table <sup>81</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *OGRLayerClassifier*.

Parameter Key	Parameter Type	Parameter Description
inshp	Input vector data	Input vector data
instats	Input File name	Input File name
insvm	Output File name	Output File name
feat	List	List
cfield	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Name of the input shapefile:** Name of the input shapefile.
- **XML file containing mean and variance of each feature.:** XML file containing mean and variance of each feature.
- **Input model filename.:** Input model filename.
- **Features:** Features to be calculated.
- **Field containing the predicted class.:** Field containing the predicted class.
- **Load otb application from xml file:** Load otb application from xml file.

<sup>81</sup> Table: Parameters table for OGRLayerClassifier.

- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_OGRLayerClassifier -inshp vectorData.shp -instats meanVar.xml -insvm svmModel.svm -feat perim
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the OGRLayerClassifier application
OGRLayerClassifier = otbApplication.Registry.CreateApplication("OGRLayerClassifier")

The following lines set all the application parameters:
OGRLayerClassifier.SetParameterString("inshp", "vectorData.shp")

OGRLayerClassifier.SetParameterString("instats", "meanVar.xml")

OGRLayerClassifier.SetParameterString("insvm", "svmModel.svm")

The following line execute the application
OGRLayerClassifier.ExecuteAndWriteOutput()
```

## Limitations

Experimental. Only shapefiles are supported for now.

## Authors

This application has been written by David Youssefi during internship at CNES.

## See Also

These additional resources can be useful for further information:

[ComputeOGRLayersFeaturesStatistics](#), [TrainOGRLayersClassifier](#)

## 6.9.8 Segmentation

Performs segmentation of an image, and output either a raster or a vector file. In vector mode, large input datasets are supported.

### Detailed description

This application allows one to perform various segmentation algorithms on a multispectral image. Available segmentation algorithms are two different versions of Mean-Shift segmentation algorithm (one being multi-threaded), simple

pixel based connected components according to a user-defined criterion, and watershed from the gradient of the intensity (norm of spectral bands vector). The application has two different modes that affects the nature of its output.

In raster mode, the output of the application is a classical image of unique labels identifying the segmented regions. The labeled output can be passed to the ColorMapping application to render regions with contrasted colours. Please note that this mode loads the whole input image into memory, and as such can not handle large images.

To segment large data, one can use the vector mode. In this case, the output of the application is a vector file or database. The input image is split into tiles (whose size can be set using the `tilsize` parameter), and each tile is loaded, segmented with the chosen algorithm, vectorized, and written into the output file or database. This piece-wise behavior ensure that memory will never get overloaded, and that images of any size can be processed. There are few more options in the vector mode. The `simplify` option allows simplifying the geometry (i.e. remove nodes in polygons) according to a user-defined tolerance. The `stitch` option tries to stitch together the polygons corresponding to segmented region that may have been split by the tiling scheme.

## Parameters

This section describes in details the parameters available for this application. Table <sup>82</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Segmentation* .

Parameter Key	Parameter Type	Parameter Description
<code>in</code>	Input image	Input image
<code>filter</code>	Choices	Choices
<code>filter meanshift</code>	<i>Choice</i>	Mean-Shift
<code>filter cc</code>	<i>Choice</i>	Connected components
<code>filter watershed</code>	<i>Choice</i>	Watershed
<code>filter mprofiles</code>	<i>Choice</i>	Morphological profiles based segmentation
<code>filter.meanshift.spatialr</code>	Int	Int
<code>filter.meanshift.ranger</code>	Float	Float
<code>filter.meanshift.thres</code>	Float	Float
<code>filter.meanshift.maxiter</code>	Int	Int
<code>filter.meanshift.minsize</code>	Int	Int
<code>filter.cc.expr</code>	String	String
<code>filter.watershed.threshold</code>	Float	Float
<code>filter.watershed.level</code>	Float	Float
<code>filter.mprofiles.size</code>	Int	Int
<code>filter.mprofiles.start</code>	Int	Int
<code>filter.mprofiles.step</code>	Int	Int
<code>filter.mprofiles.sigma</code>	Float	Float
<code>mode</code>	Choices	Choices
<code>mode vector</code>	<i>Choice</i>	Tile-based large-scale segmentation with vector output
<code>mode raster</code>	<i>Choice</i>	Standard segmentation with labeled raster output
<code>mode.vector.out</code>	Output File name	Output File name
<code>mode.vector.outmode</code>	Choices	Choices
<code>mode.vector.outmode ulco</code>	<i>Choice</i>	Update output vector file, only allow creating new layers
<code>mode.vector.outmode ovw</code>	<i>Choice</i>	Overwrite output vector file if existing.
<code>mode.vector.outmode ulovw</code>	<i>Choice</i>	Update output vector file, overwrite existing layer
<code>mode.vector.outmode ulu</code>	<i>Choice</i>	Update output vector file, update existing layer
<code>mode.vector.inmask</code>	Input image	Input image

Continued on next page

<sup>82</sup> Table: Parameters table for Segmentation.

Table 6.10 – continued from previous page

Parameter Key	Parameter Type	Parameter Description
mode.vector.neighbor	Boolean	Boolean
mode.vector.stitch	Boolean	Boolean
mode.vector.minsize	Int	Int
mode.vector.simplify	Float	Float
mode.vector.layername	String	String
mode.vector.fieldname	String	String
mode.vector.tilesize	Int	Int
mode.vector.startlabel	Int	Int
mode.vector.ogroptions	String list	String list
mode.raster.out	Output image	Output image
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input Image:** The input image to segment.

**Segmentation algorithm:** Choice of segmentation algorithm (mean-shift by default). Available choices are:

- **Mean-Shift:** OTB implementation of the Mean-Shift algorithm (multi-threaded).
- **Spatial radius:** Spatial radius of the neighborhood.
- **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multispectral space.
- **Mode convergence threshold:** Algorithm iterative scheme will stop if mean-shift vector is below this threshold or if iteration number reached maximum number of iterations.
- **Maximum number of iterations:** Algorithm iterative scheme will stop if convergence hasn't been reached after the maximum number of iterations.
- **Minimum region size:** Minimum size of a region (in pixel unit) in segmentation. Smaller clusters will be merged to the neighboring cluster with the closest radiometry. If set to 0 no pruning is done.
- **Connected components:** Simple pixel-based connected-components algorithm with a user-defined connection condition.
- **Condition:** User defined connection condition, written as a mathematical expression. Available variables are  $p(i)b(i)$ ,  $intensity\_p(i)$  and  $distance$  (example of expression :  $distance < 10$  ).
- **Watershed:** The traditional watershed algorithm. The height function is the gradient magnitude of the amplitude (square root of the sum of squared bands).
- **Depth Threshold:** Depth threshold Units in percentage of the maximum depth in the image.
- **Flood Level:** flood level for generating the merge tree from the initial segmentation (between 0 and 1).
- **Morphological profiles based segmentation:** Segmentation based on morphological profiles, as described in Martino Pesaresi and Jon Alti Benediktsson, Member, IEEE: A new approach for the morphological segmentation of high resolution satellite imagery. IEEE Transactions on geoscience and remote sensing, vol. 39, NO. 2, February 2001, p. 309-320.
- **Profile Size:** Size of the profiles.
- **Initial radius:** Initial radius of the structuring element (in pixels).
- **Radius step.:** Radius step along the profile (in pixels).
- **Threshold of the final decision rule:** Profiles values under the threshold will be ignored.

**Processing mode:** Choice of processing mode, either raster or large-scale. Available choices are:

- **Tile-based large-scale segmentation with vector output:** In this mode, the application will output a vector file or database, and process the input image piecewise. This allows performing segmentation of very large images.
  - **Output vector file:** The output vector file or database (name can be anything understood by OGR).
  - **Writing mode for the output vector file:** This allows one to set the writing behaviour for the output vector file. Please note that the actual behaviour depends on the file format. Available choices are:
    - **Update output vector file, only allow creating new layers:** The output vector file is opened in update mode if existing. If the output layer already exists, the application stops, leaving it untouched.
    - **Overwrite output vector file if existing.:** If the output vector file already exists, it is completely destroyed (including all its layers) and recreated from scratch.
    - **Update output vector file, overwrite existing layer:** The output vector file is opened in update mode if existing. If the output layer already exists, it is completely destroyed and recreated from scratch.
    - **Update output vector file, update existing layer:** The output vector file is opened in update mode if existing. If the output layer already exists, the new geometries are appended to the layer.
  - **Mask Image:** Only pixels whose mask value is strictly positive will be segmented.
  - **8-neighbor connectivity:** Activate 8-Neighborhood connectivity (default is 4).
  - **Stitch polygons:** Scan polygons on each side of tiles and stitch polygons which connect by more than one pixel.
  - **Minimum object size:** Objects whose size is below the minimum object size (area in pixels) will be ignored during vectorization.
  - **Simplify polygons:** Simplify polygons according to a given tolerance (in pixel). This option allows reducing the size of the output file or database.
  - **Layer name:** Name of the layer in the vector file or database (default is Layer).
  - **Geometry index field name:** Name of the field holding the geometry index in the output vector file or database.
  - **Tiles size:** User defined tiles size for tile-based segmentation. Optimal tile size is selected according to available RAM if null.
  - **Starting geometry index:** Starting value of the geometry index field.
  - **OGR options for layer creation:** A list of layer creation options in the form KEY=VALUE that will be passed directly to OGR without any validity checking. Options may depend on the file format, and can be found in OGR documentation.
- **Standard segmentation with labeled raster output:** In this mode, the application will output a standard labeled raster. This mode can not handle large data.
  - **Output labeled image:** The output labeled image.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Examples

### Example 1

Example of use with vector mode and watershed segmentation To run this example in command-line, use the following:

```
otbcli_Segmentation -in QB_Toulouse_Ortho_PAN.tif -mode vector -mode.vector.out SegmentationVector.s
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Segmentation application
Segmentation = otbApplication.Registry.CreateApplication("Segmentation")

The following lines set all the application parameters:
Segmentation.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Segmentation.SetParameterString("mode", "vector")

Segmentation.SetParameterString("mode.vector.out", "SegmentationVector.sqlite")

Segmentation.SetParameterString("filter", "watershed")

The following line execute the application
Segmentation.ExecuteAndWriteOutput()
```

## Example 2

Example of use with raster mode and mean-shift segmentation To run this example in command-line, use the following:

```
otbcli_Segmentation -in QB_Toulouse_Ortho_PAN.tif -mode raster -mode.raster.out SegmentationRaster.t
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the Segmentation application
Segmentation = otbApplication.Registry.CreateApplication("Segmentation")

The following lines set all the application parameters:
Segmentation.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Segmentation.SetParameterString("mode", "raster")

Segmentation.SetParameterString("mode.raster.out", "SegmentationRaster.tif")
Segmentation.SetParameterOutputImagePixelFormat("mode.raster.out", 3)

Segmentation.SetParameterString("filter", "meanshift")

The following line execute the application
Segmentation.ExecuteAndWriteOutput()
```

## Limitations

In raster mode, the application can not handle large input images. Stitching step of vector mode might become slow with very large input images. MeanShift filter results depends on the number of threads used. Watershed and multiscale geodesic morphology segmentation will be performed on the amplitude of the input image.

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

MeanShiftSegmentation

## 6.9.9 TrainOGRLayersClassifier (DEPRECATED)

Train a SVM classifier based on labeled geometries and a list of features to consider.

### Detailed description

This application trains a SVM classifier based on labeled geometries and a list of features to consider for classification. This application is deprecated, prefer using TrainVectorClassifier which offers access to all the classifiers.

### Parameters

This section describes in details the parameters available for this application. Table <sup>83</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TrainOGRLayersClassifier*.

Parameter Key	Parameter Type	Parameter Description
inshp	Input vector data	Input vector data
instats	Input File name	Input File name
outsvm	Output File name	Output File name
feat	List	List
cfield	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Name of the input shapefile:** Name of the input shapefile.
- **XML file containing mean and variance of each feature.:** XML file containing mean and variance of each feature.
- **Output model filename.:** Output model filename.
- **List of features to consider for classification.:** List of features to consider for classification.
- **Field containing the class id for supervision:** Field containing the class id for supervision. Only geometries with this field available will be taken into account.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

<sup>83</sup> Table: Parameters table for TrainOGRLayersClassifier (DEPRECATED).

## Example

To run this example in command-line, use the following:

```
otbcli_TrainOGRLayersClassifier -inshp vectorData.shp -instats meanVar.xml -outsvm svmModel.svm -feat
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the TrainOGRLayersClassifier application
TrainOGRLayersClassifier = otbApplication.Registry.CreateApplication("TrainOGRLayersClassifier")

The following lines set all the application parameters:
TrainOGRLayersClassifier.SetParameterString("inshp", "vectorData.shp")

TrainOGRLayersClassifier.SetParameterString("instats", "meanVar.xml")

TrainOGRLayersClassifier.SetParameterString("outsvm", "svmModel.svm")

The following line execute the application
TrainOGRLayersClassifier.ExecuteAndWriteOutput()
```

## Limitations

Experimental. For now only shapefiles are supported. Tuning of SVM classifier is not available.

## Authors

This application has been written by David Youssefi during internship at CNES.

## See Also

These additional resources can be useful for further information:

[OGRLayerClassifier](#), [ComputeOGRLayersFeaturesStatistics](#)

## 6.10 Miscellaneous

### 6.10.1 Band Math

Perform a mathematical operation on monoband images

#### Detailed description

This application performs a mathematical operation on monoband images. Mathematical formula interpretation is done via MuParser libraries. For MuParser version superior to 2.0 uses '&&' and '||' logical operators, and ternary operator 'boolean\_expression ? if\_true : if\_false' For older version of MuParser (prior to v2) use 'and' and 'or' log-

ical operators, and ternary operator 'if( ; )'. The list of features and operators is available on MuParser website: <http://muparser.sourceforge.net/>

## Parameters

This section describes in details the parameters available for this application. Table <sup>84</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BandMath* .

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
out	Output image	Output image
ram	Int	Int
exp	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input image list:** Image list to perform computation on.
- **Output Image:** Output image.
- **Available RAM (Mb):** Available memory for processing (in MB).
- **Expression:** The mathematical expression to apply. Use im1b1 for the first band, im1b2 for the second one...
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_BandMath -il verySmallFSATSW_r.tif verySmallFSATSW_nir.tif verySmallFSATSW.tif -out apTvUtBand
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the BandMath application
BandMath = otbApplication.Registry.CreateApplication("BandMath")

The following lines set all the application parameters:
BandMath.SetParameterStringList("il", ['verySmallFSATSW_r.tif', 'verySmallFSATSW_nir.tif', 'verySmallFSATSW.tif'])

BandMath.SetParameterString("out", "apTvUtBandMathOutput.tif")

BandMath.SetParameterString("exp", "cos(im1b1) > cos(im2b1) ? im3b1 : im3b2 ")

The following line execute the application
BandMath.ExecuteAndWriteOutput()
```

<sup>84</sup> Table: Parameters table for Band Math.

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.10.2 Band Math X

This application performs mathematical operations on multiband images. Mathematical formula interpretation is done via muParserX library : <http://articles.beltoforion.de/article.php?a=muparserx>

### Detailed description

The goal of this documentation is to give the user some hints about the syntax used in this application. The syntax is mainly constrained by the muparserx library, which can be considered as the core of the application.

- Fundamentals:

The default prefix name for variables related to the  $i$ th input is 'im(i+1)' (note the indexing from 1 to N, for N inputs). The following list summarizes the available variables for input #0 (and so on for every input):

im1 → a pixel from first input, made of n components (n bands) im1bj → jth component of a pixel from first input (first band is indexed by 1) im1bjNkxp → a neighbourhood ('N') of pixels of the jth component from first input, of size kxp  
 im1PhyX and im1PhyY → spacing of first input in X and Y directions (horizontal and vertical) im1bjMean im1bjMin  
 im1bjMax im1bjSum im1bjVar → mean, min, max, sum, variance of jth band from first input (global statistics)

Moreover, we also have the following generic variables: idxX and idxY → indices of the current pixel

Always keep in mind that this application only addresses mathematically well-defined formulas. For instance, it is not possible to add vectors of different dimensions (this implies the addition of a row vector with a column vector), or add a scalar to a vector or a matrix, or divide two vectors, and so on... Thus, it is important to remember that a pixel of n components is always represented as a row vector.

Example :

im1 + im2 (1)

represents the addition of pixels from first and second inputs. This expression is consistent only if both inputs have the same number of bands. Note that it is also possible to use the following expressions to obtain the same result:

im1b1 + im2b1 im1b2 + im2b2 (2) ...

Nevertheless, the first expression is by far much pleaseant. We call this new functionality the 'batch mode' (performing the same operation in a band-to-band fashion).

- Operations involving neighborhoods of pixels:

Another new fonctionnality is the possibility to perform operations that involve neighborhoods of pixels. Variable related to such neighborhoods are always defined following the pattern imIbJNKxP, where: - I is an number identifying the image input (remember, input #0 = im1, and so on) - J is an number identifying the band (remember, first band is indexed by 1) - KxP are two numbers that represent the size of the neighborhood (first one is related to the horizontal direction) All neighborhood are centered, thus K and P must be odd numbers. Many operators come with this new functionality: dotpr, mean var median min max... For instance, if im1 represents the pixel of 3 bands image:

im1 - mean(im1b1N5x5,im1b2N5x5,im1b3N5x5) (3)

could represent a high pass filter (Note that by implying three neighborhoods, the operator mean returns a row vector of three components. It is a typical behaviour for many operators of this application).

- Operators:

In addition to the previous operators, other operators are available: - existing operators/functions from muParserX, that were not originally defined for vectors and matrices (for instance cos, sin, ...). These new operators/ functions keep the original names to which we added the prefix 'v' for vector (vcos, vsin, ...). - mult, div and pow operators, that perform element-wise multiplication, division or exponentiation of vector/matrices (for instance im1 div im2) - mlt, dv and pw operators, that perform multiplication, division or exponentiation of vector/matrices by a scalar (for instance im1 dv 2.0) - bands, which is a very useful operator. It allows selecting specific bands from an image, and/or to rearrange them in a new vector; for instance bands(im1,{1,2,1,1}) produces a vector of 4 components made of band 1, band 2, band 1 and band 1 values from the first input. Note that curly brackets must be used in order to select the desired band indices. ... and so on.

- Application itself:

The application takes the following parameters : - Setting the list of inputs can be done with the 'il' parameter. - Setting expressions can be done with the 'exp' parameter (see also limitations section below). - Setting constants can be done with the 'incontext' parameter. User must provide a txt file with a specific syntax: #type name value An example of such a file is given below:

```
#F expo 1.1 #M kernel1 { 0.1 , 0.2 , 0.3; 0.4 , 0.5 , 0.6; 0.7 , 0.8 , 0.9; 1 , 1.1 , 1.2; 1.3 , 1.4 , 1.5 }
```

As we can see, #I/#F allows the definition of an integer/float constant, whereas #M allows the definition of a vector/matrix. In the latter case, elements of a row must be separated by commas, and rows must be separated by semicolons. It is also possible to define expressions within the same txt file, with the pattern #E expr. For instance (two expressions; see also limitations section below):

```
#E $dotpr(kernel1,im1b1N3x5); im2b1^expo$
```

- The 'outcontext' parameter allows saving user's constants and expressions (context).
- Setting the output image can be done with the 'out' parameter (multi-outputs is not implemented yet).

Finally, we strongly recommend that the reader takes a look at the cookbook, where additional information can be found (<http://www.orfeo-toolbox.org/packages/OTBCookBook.pdf>).

## Parameters

This section describes in details the parameters available for this application. Table <sup>85</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BandMathX*.

Parameter Key	Parameter Type	Parameter Description
il	Input image list	Input image list
out	Output image	Output image
ram	Int	Int
exp	String	String
incontext	Input File name	Input File name
outcontext	Output File name	Output File name
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input image list:** Image list to perform computation on.
- **Output Image:** Output image.
- **Available RAM (Mb):** Available memory for processing (in MB).

<sup>85</sup> Table: Parameters table for Band Math X.

- **Expressions:** Mathematical expression to apply.
- **Import context:** A txt file containing user's constants and expressions.
- **Export context:** A txt file where to save user's constants and expressions.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_BandMathX -il verySmallFSATSW_r.tif verySmallFSATSW_nir.tif verySmallFSATSW.tif --out apTvUtBandMathOutput.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the BandMathX application
BandMathX = otbApplication.Registry.CreateApplication("BandMathX")

The following lines set all the application parameters:
BandMathX.SetParameterStringList("il", ['verySmallFSATSW_r.tif', 'verySmallFSATSW_nir.tif', 'verySmallFSATSW.tif'])
BandMathX.SetParameterString("out", "apTvUtBandMathOutput.tif")
BandMathX.SetParameterString("exp", "cos(im1b1)+im2b1*im3b1-im3b2+ndvi(im3b3, im3b4)")

The following line execute the application
BandMathX.ExecuteAndWriteOutput()
```

## Limitations

The application is currently unable to produce one output image per expression, contrary to `otbBandMathXImageFilter`. Separating expressions by semi-colons (;) will concatenate their results into a unique multiband output image.

## Authors

This application has been written by OTB-Team.

## 6.10.3 Images comparison

Estimator between 2 images.

### Detailed description

This application computes MSE (Mean Squared Error), MAE (Mean Absolute Error) and PSNR (Peak Signal to Noise Ratio) between the channel of two images (reference and measurement). The user has to set the used channel and can specify a ROI.

## Parameters

This section describes in details the parameters available for this application. Table <sup>86</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *CompareImages*.

Parameter Key	Parameter Type	Parameter Description
ref	Group	Group
ref.in	Input image	Input image
ref.channel	Int	Int
meas	Group	Group
meas.in	Input image	Input image
meas.channel	Int	Int
roi	Group	Group
roi.startx	Int	Int
roi.starty	Int	Int
roi.sizeX	Int	Int
roi.sizeY	Int	Int
mse	Float	Float
mae	Float	Float
psnr	Float	Float
count	Float	Float
ram	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

### [Reference image properties]

- **Reference image:** Image used as reference in the comparison.
- **Reference image channel:** Used channel for the reference image.

### [Measured image properties]

- **Measured image:** Image used as measured in the comparison.
- **Measured image channel:** Used channel for the measured image.

### [Region Of Interest (relative to reference image)]

- **Start X:** ROI start x position.
- **Start Y:** ROI start y position.
- **Size X:** Size along x in pixels.
- **Size Y:** Size along y in pixels.

**MSE:** Mean Squared Error value.

**MAE:** Mean Absolute Error value.

**PSNR:** Peak Signal to Noise Ratio value.

**count:** Nb of pixels which are different.

**Available RAM (Mb):** Available memory for processing (in MB).

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

<sup>86</sup> Table: Parameters table for Images comparison.

## Example

To run this example in command-line, use the following:

```
otbcli_CompareImages -ref.in GomaApres.png -ref.channel 1 -meas.in GomaAvant.png -meas.channel 2 -ro
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the CompareImages application
CompareImages = otbApplication.Registry.CreateApplication("CompareImages")

The following lines set all the application parameters:
CompareImages.SetParameterString("ref.in", "GomaApres.png")

CompareImages.SetParameterInt("ref.channel", 1)

CompareImages.SetParameterString("meas.in", "GomaAvant.png")

CompareImages.SetParameterInt("meas.channel", 2)

CompareImages.SetParameterInt("roi.startx", 20)

CompareImages.SetParameterInt("roi.starty", 30)

CompareImages.SetParameterInt("roi.sizeX", 150)

CompareImages.SetParameterInt("roi.sizeY", 200)

The following line execute the application
CompareImages.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

BandMath application, ImageStatistics

## 6.10.4 Hyperspectral data unmixing

Estimate abundance maps from an hyperspectral image and a set of endmembers.

## Detailed description

The application applies a linear unmixing algorithm to an hyperspectral data cube. This method supposes that the mixture between materials in the scene is macroscopic and simulates a linear mixing model of spectra. The Linear Mixing Model (LMM) acknowledges that reflectance spectrum associated with each pixel is a linear combination of pure materials in the recovery area, commonly known as endmembers. Endmembers can be estimated using the Vertex-ComponentAnalysis application. The application allows one to estimate the abundance maps with several algorithms : Unconstrained Least Square (ucls), Fully Constrained Least Square (fcls), Image Space Reconstruction Algorithm (isra) and Non-negative constrained Least Square (ncls) and Minimum Dispersion Constrained Non Negative Matrix Factorization (MDMDNMF).

## Parameters

This section describes in details the parameters available for this application. Table <sup>87</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *HyperspectralUnmixing* .

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output image	Output image
ie	Input image	Input image
ua	Choices	Choices
ua ucls	<i>Choice</i>	UCLS
ua ncls	<i>Choice</i>	NCLS
ua isra	<i>Choice</i>	ISRA
ua mdmdnmf	<i>Choice</i>	MDMDNMF
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image Filename:** The hyperspectral data cube to unmix.
- **Output Image:** The output abundance map.
- **Input endmembers:** The endmembers (estimated pure pixels) to use for unmixing. Must be stored as a multi-spectral image, where each pixel is interpreted as an endmember.
- **Unmixing algorithm:** The algorithm to use for unmixing. Available choices are:
  - **UCLS:** Unconstrained Least Square.
  - **NCLS:** Non-negative constrained Least Square.
  - **ISRA:** Image Space Reconstruction Algorithm.
  - **MDMDNMF:** Minimum Dispersion Constrained Non Negative Matrix Factorization.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_HyperspectralUnmixing -in cupriteSubHsi.tif -ie cupriteEndmembers.tif -out HyperspectralUnmix
```

<sup>87</sup> Table: Parameters table for Hyperspectral data unmixing.

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the HyperspectralUnmixing application
HyperspectralUnmixing = otbApplication.Registry.CreateApplication("HyperspectralUnmixing")

The following lines set all the application parameters:
HyperspectralUnmixing.SetParameterString("in", "cupriteSubHsi.tif")

HyperspectralUnmixing.SetParameterString("ie", "cupriteEndmembers.tif")

HyperspectralUnmixing.SetParameterString("out", "HyperspectralUnmixing.tif")
HyperspectralUnmixing.SetParameterOutputImagePixelFormat("out", 7)

HyperspectralUnmixing.SetParameterString("ua", "ucls")

The following line execute the application
HyperspectralUnmixing.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

[VertexComponentAnalysis](#)

## 6.10.5 Image to KMZ Export

Export the input image in a KMZ product.

### Detailed description

This application exports the input image in a kmz product that can be display in the Google Earth software. The user can set the size of the product size, a logo and a legend to the product. Furthermore, to obtain a product that fits the relief, a DEM can be used.

## Parameters

This section describes in details the parameters available for this application. Table <sup>88</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *KmzExport*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
out	Output File name	Output File name
tilesize	Int	Int
logo	Input image	Input image
legend	Input image	Input image
elev	Group	Group
elev.dem	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Input image:** Input image.

**Output .kmz product:** Output Kmz product directory (with .kmz extension).

**Tile Size:** Size of the tiles in the kmz product, in number of pixels (default = 512).

**Image logo:** Path to the image logo to add to the KMZ product.

**Image legend:** Path to the image legend to add to the KMZ product.

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occurs if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_KmzExport -in qb_RoadExtract2.tif -out otbKmzExport.kmz -logo otb_big.png
```

To run this example from Python, use the following code snippet:

<sup>88</sup> Table: Parameters table for Image to KMZ Export.

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the KmzExport application
KmzExport = otbApplication.Registry.CreateApplication("KmzExport")

The following lines set all the application parameters:
KmzExport.SetParameterString("in", "qb_RoadExtract2.tif")

KmzExport.SetParameterString("out", "otbKmzExport.kmz")

KmzExport.SetParameterString("logo", "otb_big.png")

The following line execute the application
KmzExport.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

### See Also

These additional resources can be useful for further information:

Conversion

## 6.10.6 Open Street Map layers importations applications

Generate a vector data from OSM on the input image extend

### Detailed description

Generate a vector data from Open Street Map data. A DEM could be use. By default, the entire layer is downloaded, an image can be use as support for the OSM data. The application can provide also available classes in layers . This application required an Internet access. Information about the OSM project : <http://www.openstreetmap.fr/>

### Parameters

This section describes in details the parameters available for this application. Table <sup>89</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *OSMDownloader* .

---

<sup>89</sup> Table: Parameters table for Open Street Map layers importations applications.

Parameter Key	Parameter Type	Parameter Description
out	Output vector data	Output vector data
support	Input image	Input image
key	String	String
value	String	String
elev	Group	Group
elev.dir	Directory	Directory
elev.geoid	Input File name	Input File name
elev.default	Float	Float
printclasses	Boolean	Boolean
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

**Output vector data:** Generated output vector data path.

**Support image:** Image used as support to estimate the models.

**OSM tag key:** OSM tag key to extract (highway, building...).

**OSM tag value:** OSM tag value to extract (motorway, footway...).

**[Elevation management]:** This group of parameters allows managing elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows selecting a directory containing Digital Elevation Model files. Note that this directory should contain only DEM files. Unexpected behaviour might occur if other images are found in this directory.
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles. A version of the geoid can be found on the OTB website (<http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd>).
- **Default elevation:** This parameter allows setting the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no\_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

**option to display available key/value classes:** Print the key/value classes available for the bounding box of the input image \*\* If not used : Note that the options OSMKey (-key) and Output (-out) become mandatory.

**Load otb application from xml file:** Load otb application from xml file.

**Save otb application to xml file:** Save otb application to xml file.

## Example

To run this example in command-line, use the following:

```
otbcli_OSMDownloader -support qb_RoadExtract.tif -key highway -out apTvUtOSMDownloader.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python
Import the otb applications package
import otbApplication

The following line creates an instance of the OSMDownloader application
OSMDownloader = otbApplication.Registry.CreateApplication("OSMDownloader")
```

```
The following lines set all the application parameters:
OSMDownloader.SetParameterString("support", "qb_RoadExtract.tif")

OSMDownloader.SetParameterString("key", "highway")

OSMDownloader.SetParameterString("out", "apTvUtOSMDownloader.shp")

The following line execute the application
OSMDownloader.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## See Also

These additional resources can be useful for further information:

Conversion

## 6.10.7 Obtain UTM Zone From Geo Point

UTM zone determination from a geographic point.

### Detailed description

This application returns the UTM zone of an input geographic point.

### Parameters

This section describes in details the parameters available for this application. Table <sup>90</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ObtainUTMZoneFromGeoPoint*.

Parameter Key	Parameter Type	Parameter Description
lat	Float	Float
lon	Float	Float
utm	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Latitude:** Latitude value of desired point.
- **Longitude:** Longitude value of desired point.
- **UTMZone:** UTM Zone.

---

<sup>90</sup> Table: Parameters table for Obtain UTM Zone From Geo Point.

- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

## Example

Obtain a UTM ZoneTo run this example in command-line, use the following:

```
otbcli_ObtainUTMZoneFromGeoPoint -lat 10.0 -lon 124.0
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the ObtainUTMZoneFromGeoPoint application
ObtainUTMZoneFromGeoPoint = otbApplication.Registry.CreateApplication("ObtainUTMZoneFromGeoPoint")

The following lines set all the application parameters:
ObtainUTMZoneFromGeoPoint.SetParameterFloat("lat", 10.0)

ObtainUTMZoneFromGeoPoint.SetParameterFloat("lon", 124.0)

The following line execute the application
ObtainUTMZoneFromGeoPoint.ExecuteAndWriteOutput()
```

## Limitations

None

## Authors

This application has been written by OTB-Team.

## 6.10.8 Pixel Value

Get the value of a pixel.

### Detailed description

Get the value of a pixel. Pay attention, index starts at 0.

### Parameters

This section describes in details the parameters available for this application. Table <sup>91</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *PixelValue* .

---

<sup>91</sup> Table: Parameters table for Pixel Value.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
coordx	Int	Int
coordy	Int	Int
cl	List	List
value	String	String
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input image.
- **Col index:** Column index of the wanted pixel (starts at 0).
- **Line index:** Line index of the wanted pixel (starts at 0).
- **Channels:** Displayed channels.
- **Pixel Value:** Pixel radiometric value.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_PixelValue -in QB_Toulouse_Ortho_XS.tif -coordx 50 -coordy 100 -cl Channel1
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the PixelValue application
PixelValue = otbApplication.Registry.CreateApplication("PixelValue")

The following lines set all the application parameters:
PixelValue.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

PixelValue.SetParameterInt("coordx", 50)

PixelValue.SetParameterInt("coordy", 100)

The following line execute the application
PixelValue.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.

## 6.10.9 Vertex Component Analysis

Find endmembers in hyperspectral images with Vertex Component Analysis

### Detailed description

Applies the Vertex Component Analysis to an hyperspectral image to extract endmembers

### Parameters

This section describes in details the parameters available for this application. Table <sup>92</sup> presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VertexComponentAnalysis*.

Parameter Key	Parameter Type	Parameter Description
in	Input image	Input image
ne	Int	Int
outendm	Output image	Output image
rand	Int	Int
inxml	XML input parameters file	XML input parameters file
outxml	XML output parameters file	XML output parameters file

- **Input Image:** Input hyperspectral data cube.
- **Number of endmembers:** The number of endmembers to extract from the data cube.
- **Output Endmembers:** The endmembers, stored in a one-line multi-spectral image, each pixel representing an endmember.
- **set user defined seed:** Set specific seed. with integer value.
- **Load otb application from xml file:** Load otb application from xml file.
- **Save otb application to xml file:** Save otb application to xml file.

### Example

To run this example in command-line, use the following:

```
otbcli_VertexComponentAnalysis -in cupriteSubHsi.tif -ne 5 -outendm VertexComponentAnalysis.tif double
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

Import the otb applications package
import otbApplication

The following line creates an instance of the VertexComponentAnalysis application
VertexComponentAnalysis = otbApplication.Registry.CreateApplication("VertexComponentAnalysis")

The following lines set all the application parameters:
VertexComponentAnalysis.SetParameterString("in", "cupriteSubHsi.tif")

VertexComponentAnalysis.SetParameterInt("ne", 5)
```

<sup>92</sup> Table: Parameters table for Vertex Component Analysis.

```
VertexComponentAnalysis.SetParameterString("outendm", "VertexComponentAnalysis.tif")
VertexComponentAnalysis.SetParameterOutputImagePixelFormat("outendm", 7)
```

```
The following line execute the application
VertexComponentAnalysis.ExecuteAndWriteOutput()
```

### Limitations

None

### Authors

This application has been written by OTB-Team.