OTB CookBook Documentation

Release 6.6.0

OTB Team

Jun 07, 2018

CONTENTS

CHAPTER

ONE

WELCOME TO ORFEO TOOLBOX!

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.

CHAPTER

TWO

INSTALLATION

We provide different standalone binary packages for OTB-Applications:

- for Windows platform (Seven or higher)
- for 64 bits 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.

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.
- `include`: A folder containing all the necessary headers to compile OTB based projects.
- `tool`: A folder containing useful scripts to test the installation or to uninstall OTB libraries and headers while keeping all the dependencies.

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`.

The package can be used to compile other projects using OTB (binaries, libraries and headers are included). If you want to build OTB from source using this package, you should first uninstall the specific OTB files from the package to leave only the dependencies (what we call an XDK). You can do it using the supplied script `tools/uninstall_otb.bat`.

In the package you also have a template project for Visual 2015 `OTB Project.zip`. This template can be placed in your user Visual 2015 template directory : `%USERPROFILE%\Documents\Visual Studio 2015\Templates\ProjectTemplates`. The script `start_devenv.bat` allows to copy the template in that folder and start Visual Studio.

Python bindings

Starting from OTB 5.8.0, OTB bindings for Python 2.7 are distributed with binary package. With OTB 6.4.0, additional bindings for Python 3.5 are also included. Please note that using a different Python version may not be compatible with OTB wrappings. If no compatible Python 2.x version is found a notification is generated during the installation process. If the installation completes without issue, information relating to your Python bindings will be provided.

You must have Python numpy bindings installed in your system. They can be installed locally without admin rights as follows: "pip install –user numpy". This is to give users the option to select their own existing Python installation rather than the one dibstributed by the OTB package.

By default, bindings for Python 2.7 will be enabled with the `otbenv` script. If you want to use bindings for Python 3.5, you can copy this script and modify:

- `lib/python` into `lib/python3`, for variable PYTHONPATH

Notes

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

Linux

We provide a binary package for GNU/Linux x86_64. This package includes all of the OTB applications along with command line and graphical launchers. It can be downloaded from OTB's download page.

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

```
chmod +x OTB-6.6.0-Linux64.run
./OTB-6.6.0-Linux64.run
```

The self-extractable archive only needs common tools found on most Linux distributions ("sed", "grep", "find", "cat", "printf", "ln", ...). However, be aware that it requires tools such as "which" and "file" (they are not always present, for instance when building a container).

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 consists 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.
- `include`: A folder containing all the necessary headers to compile OTB based projects.
- `share`: A folder containing common resources and copyright mentions.

- `tool`: A folder containing useful scripts to test the installation or to uninstall OTB libraries and headers while keeping all the dependencies.

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`.

The package can be used to compile other projects using OTB (binaries, libraries and headers are included). If you want to build OTB from source using this package, you should first uninstall the specific OTB files from the package to leave only the dependencies (what we call an XDK). You can do it using the supplied script `tools/uninstall_otb.sh`.

System dependencies

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.

Caveat on OTB 6.0

In OTB 6.0 binaries, there is a small caveat for "expat" as the supplied binaries depend on "libexpat.so", which is not contained in the package. It can be supplied by most package managers (apt, yum, ...). If not already present, it is necessary to install one of the following packages:

```
libexpat-dev   libexpat1-dev
```

Python bindings

Starting from OTB 5.8.0, OTB bindings for Python 2.7 are distributed with binary package. With OTB 6.4.0, additional bindings for Python 3.5 are also included. Please note that using a different Python version may not be compatible with OTB wrappings. If no compatible Python 2.x version is found a notification is generated during the installation process. If the installation completes without issue, information relating to your Python bindings will be provided.

You must have Python NumPy bindings installed in your system. They can be installed locally without admin rights as follows: "pip install –user numpy". This is to give users the option to select their own existing Python installation rather than the one distributed by the OTB package.

By default, bindings for Python 2.7 will be enabled with the `otbenv` script. If you want to use bindings for Python 3.5, you can copy this script and modify:

- `lib/python` into `lib/python3`, for variable PYTHONPATH

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-6.6.0-Linux64" into another location say /usr/local/ after extraction. But avoid this action!

- To have "OTB-6.6.0-Linux64" installed in /usr/local or /opt execute "OTB-6.6.0-Linux64.run" in that directory.
- Multiple installation of OTB can exists in same system without one conflicting the other!

FAQ

Q: Unable to import otbApplication library with Python3

```
ImportError: libpython3.5m.so.rh-python35-1.0: cannot open shared object file: No␣
↪such file or directory
```

A: You need to add a symlink to libpython3.5m.so.rh-python35-1.0 to make it works.

Here is the solution:

- Find the libpython3.5XX on your system : `find /usr/lib -iname *libpython3.5*` (on Ubuntu 14.04, it is `/usr/lib/x86_64-linux-gnu/libpython3.5m.so`)
- Create a symlink : `ln -s path/to/lib/python3.5XX path/to/lib/libpython3.5m.so.rh-python35-1.0`
- Try to import otbApplication again

See this discussion on OTB issue tracker

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-6.6.0-Darwin64.run
./OTB-6.6.0-Darwin64.run
```

Once the archive is extracted, you can see OTB-6.6.0-Darwin64 directory in the same direcrtory along with OTB-6.6.0-Darwin64.run

Contents of OTB-6.6.0-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.
- `include`: A folder containing all the necessary headers to compile OTB based projects.
- `share`: A folder containing common resources and copyright mentions.
- `tool`: A folder containing useful scripts to test the installation or to uninstall OTB libraries and headers while keeping all the dependencies.

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`.

The package can be used to compile other projects using OTB (binaries, libraries and headers are included). If you want to build OTB from source using this package, you should first uninstall the specific OTB files from the package to leave only the dependencies (what we call an XDK). You can do it using the supplied script `tools/uninstall_otb.sh`.

Python bindings

Starting from OTB 5.8.0, OTB bindings for Python 2.7 are distributed with binary package. With OTB 6.4.0, additional bindings for Python 3.5 are also included. Please note that using a different Python version may not be compatible with OTB wrappings. If no compatible Python 2.x version is found a notification is generated during the installation process. If the installation completes without issue, information relating to your Python bindings will be provided.

You must have Python numpy bindings installed in your system. They can be installed locally without admin rights as follows: "pip install –user numpy". This is to give users the option to select their own existing Python installation rather than the one dibstributed by the OTB package.

By default, bindings for Python 2.7 will be enabled with the `otbenv` script. If you want to use bindings for Python 3.5, you can copy this script and modify:

- `lib/python` into `lib/python3`, for variable PYTHONPATH

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-6.6.0-Darwin64 after extraction. In case you need to have OTB installed in some other directory. Extract the .run file there.

FAQ

Q: I am getting an error message...

```
xcrun: error: invalid active developer path
(/Library/Developer/CommandLineTools), missing xcrun at:
/Library/Developer/CommandLineTools/usr/bin/xcrun
```

A: You can get this error at startup running Monteverdi.app or Mapla.app. The solution is to run in a terminal the following command:

```
xcode-select --install
```

And then try to restart Monteverdi or Mapla.

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.

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.

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
```

You will then need to 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.

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 the following commands (please replace 11.4 with your version of OpenSuse):

```
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
```

You should then 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.

CHAPTER

THREE

A BRIEF TOUR OF OTB APPLICATIONS

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 integrate them into high level pipelines. OTB applications allow the user to:

- Combine two or more functions from the Orfeo ToolBox,
- Provide a high level interface to handle: input and output data, definition of parameters and communication with the user.

OTB applications can be launched in different ways, and accessed from different entry points. While the framework can be extended, the 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 complete list of applications is described in the *Applications Reference Documentation*.

All standard applications share 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.

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 the path to the shared library (or plugin) correpsonding to the `module_name` to be passed to the launcher.

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 without parameters, or with incomplete parameters, will cause the launcher to display a summary of the parameters. This summary will display the minimum set of parameters that are required to execute the application. 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/int16/uint32/int32/float/double]
        -map                            <string>         Output Map Projection [utm/
↪lambert2/lambert93/transmercator/wgs/epsg]
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/autospacing]
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
        -elev.geoid                     <string>         Geoid File
        -elev.default                   <float>          Average Elevation
        -interpolator                   <string>         Interpolation [nn/linear/
↪bco]
        -interpolator.bco.radius        <int32>          Radius for bicubic␣
↪interpolation
```

```
        -opt.rpc                        <int32>          RPC modeling (points per␣
↪axis)
        -opt.ram                        <int32>          Available memory for␣
↪processing (in MB)
        -opt.gridspacing                <float>          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.

Graphical 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 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.

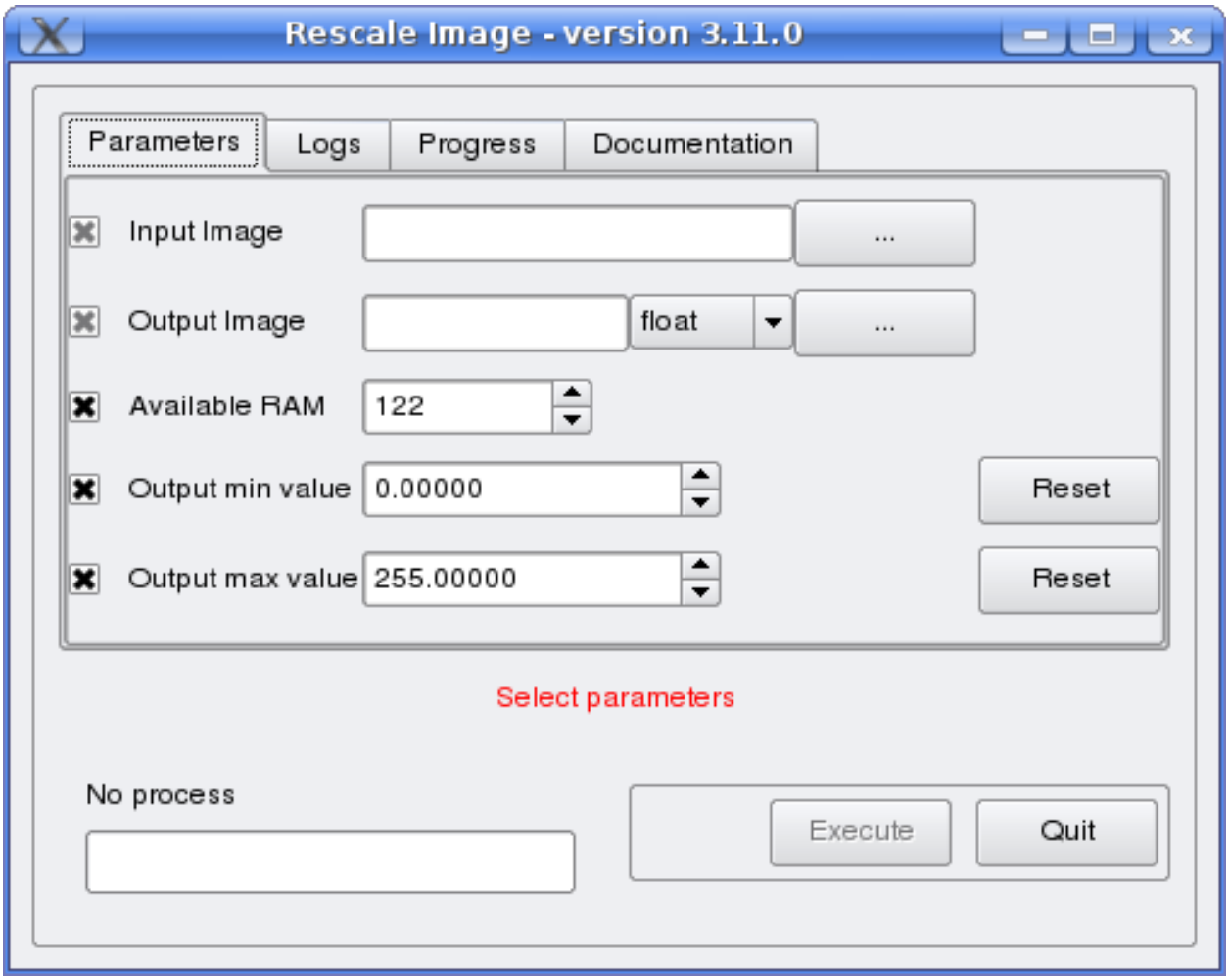
The interface of the application is shown here as an example.

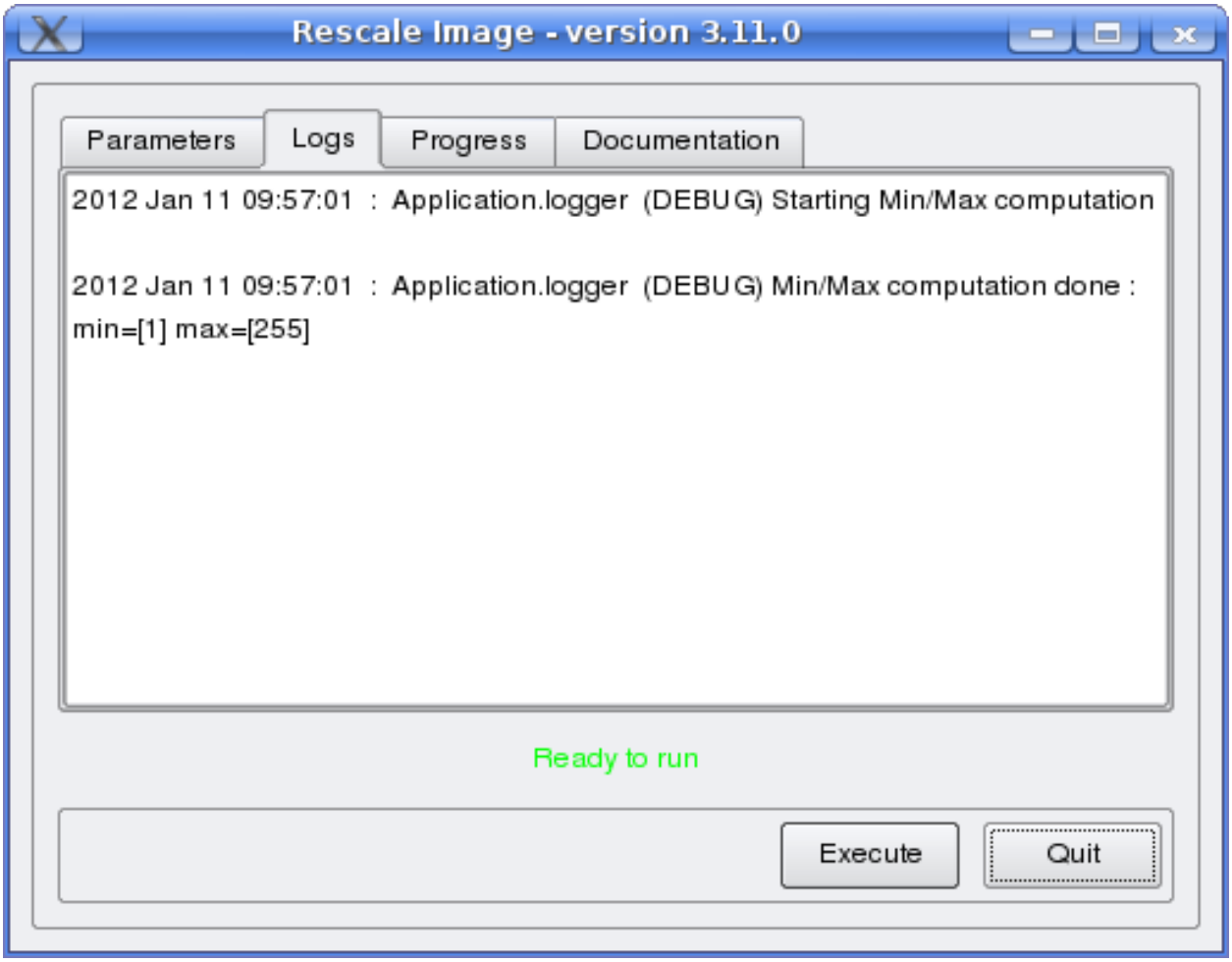
Python interface

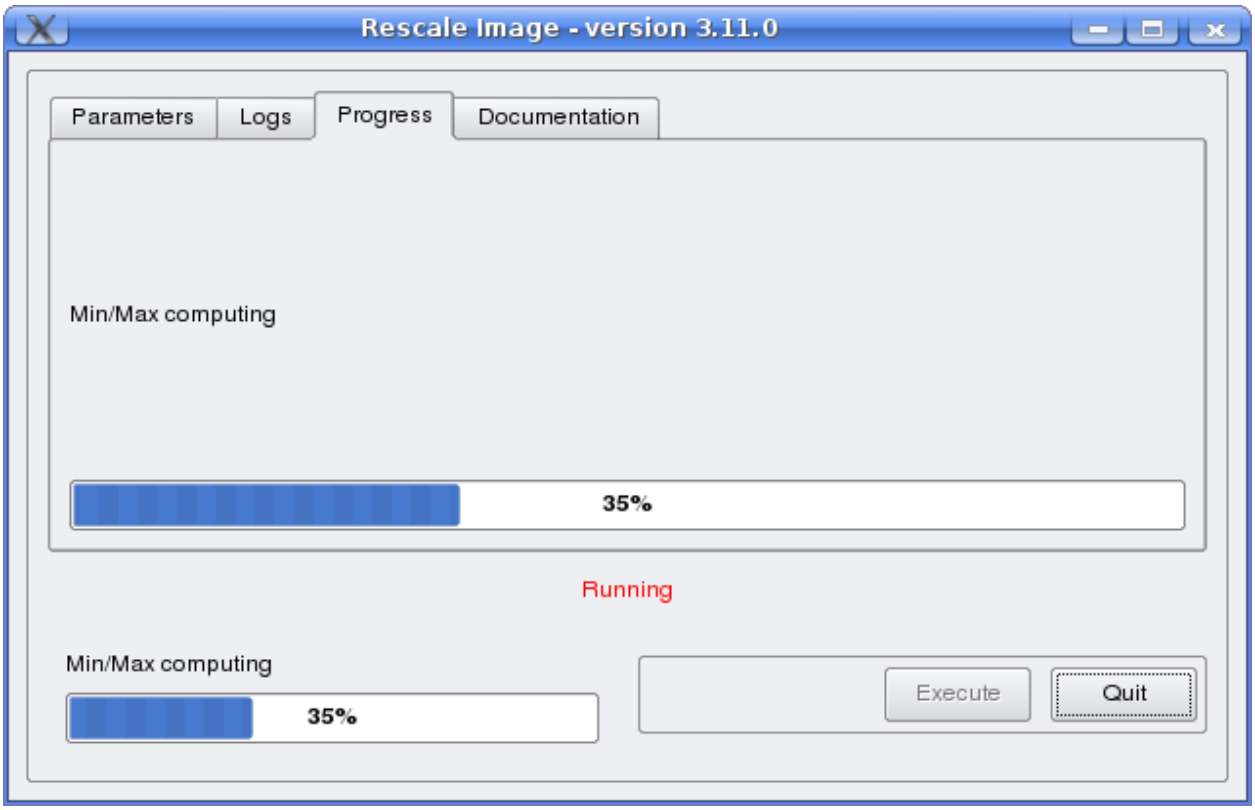
The applications can also be accessed from Python, through a module named otbApplication. However, there are technical requirements to use it. If you use OTB through standalone packages, you should use the supplied environment script otbenv to properly setup variables such as PYTHONPATH and OTB_APPLICATION_PATH (on Unix systems, don't forget to source the script). In other cases, you should set these variables depending on your configuration.

On Unix systems, it is typically available in the /usr/lib/otb/python directory. Depending on how you installed OTB, you may need to configure the environment variable PYTHONPATH to include this directory so that the module becomes available from Python.

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


Rescale Image - version 3.11.0
Parameters
Logs
Progress
Documentation
Input Image
...
Output Image
float
...
Available RAM
122
Output min value
0.00000
Reset
Output max value
255.00000
Reset
Select parameters
No process
Execute
Quit


Rescale Image - version 3.11.0
Parameters
Logs
Progress
Documentation
2012 Jan 11 09:57:01 : Application.logger (DEBUG) Starting Min/Max computation
2012 Jan 11 09:57:01 : Application.logger (DEBUG) Min/Max computation done : min=[1] max=[255]
Ready to run
Execute
Quit



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`.

Once your environment is set, you can use OTB applications from Python, just like this small example:

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

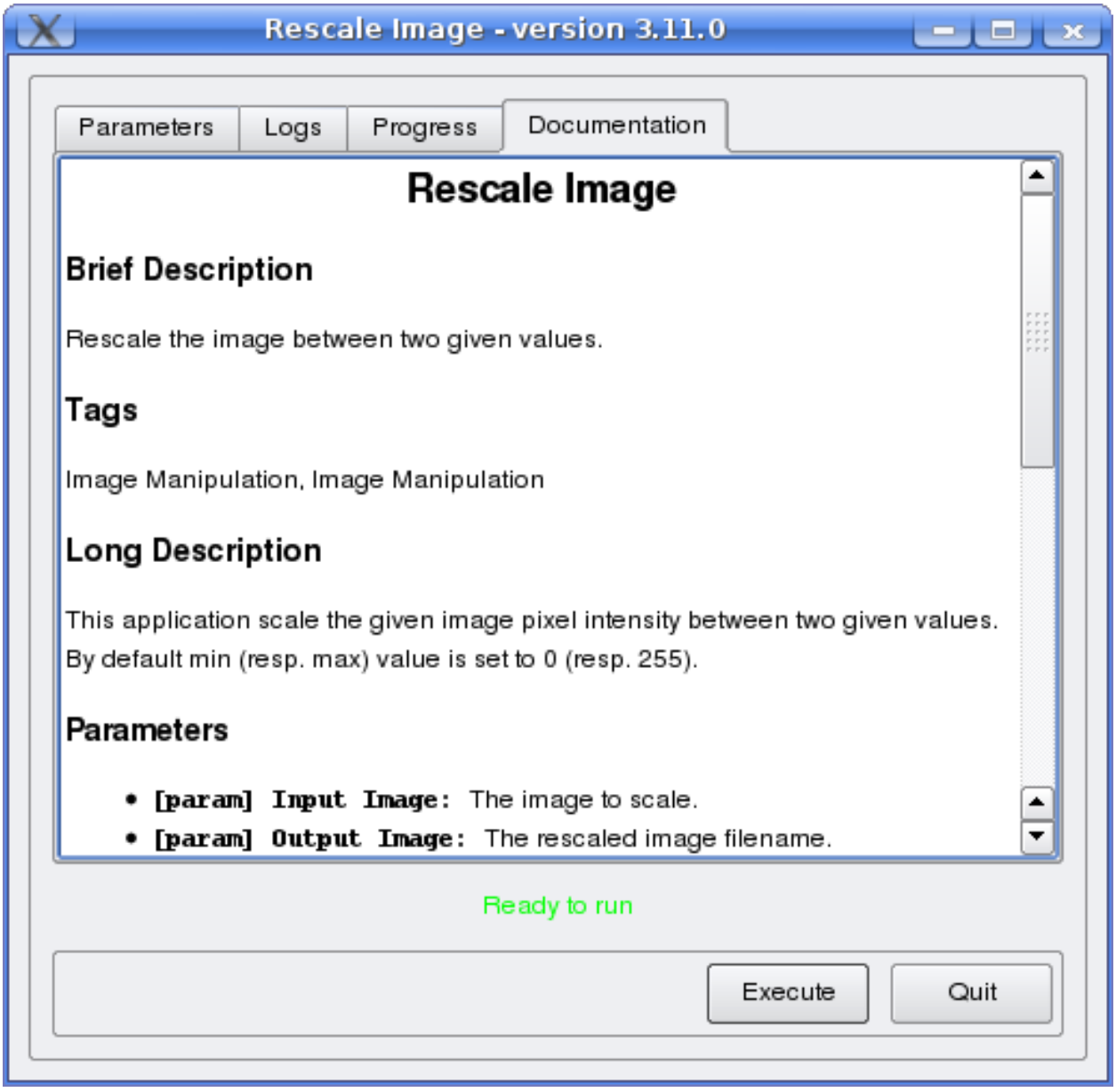
# The python module providing access to OTB applications is otbApplication
import otbApplication as otb

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

# We set its parameters
app.SetParameterString("in", "my_input_image.tif")
app.SetParameterString("type", "mean")
app.SetParameterString("out", "my_output_image.tif")

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

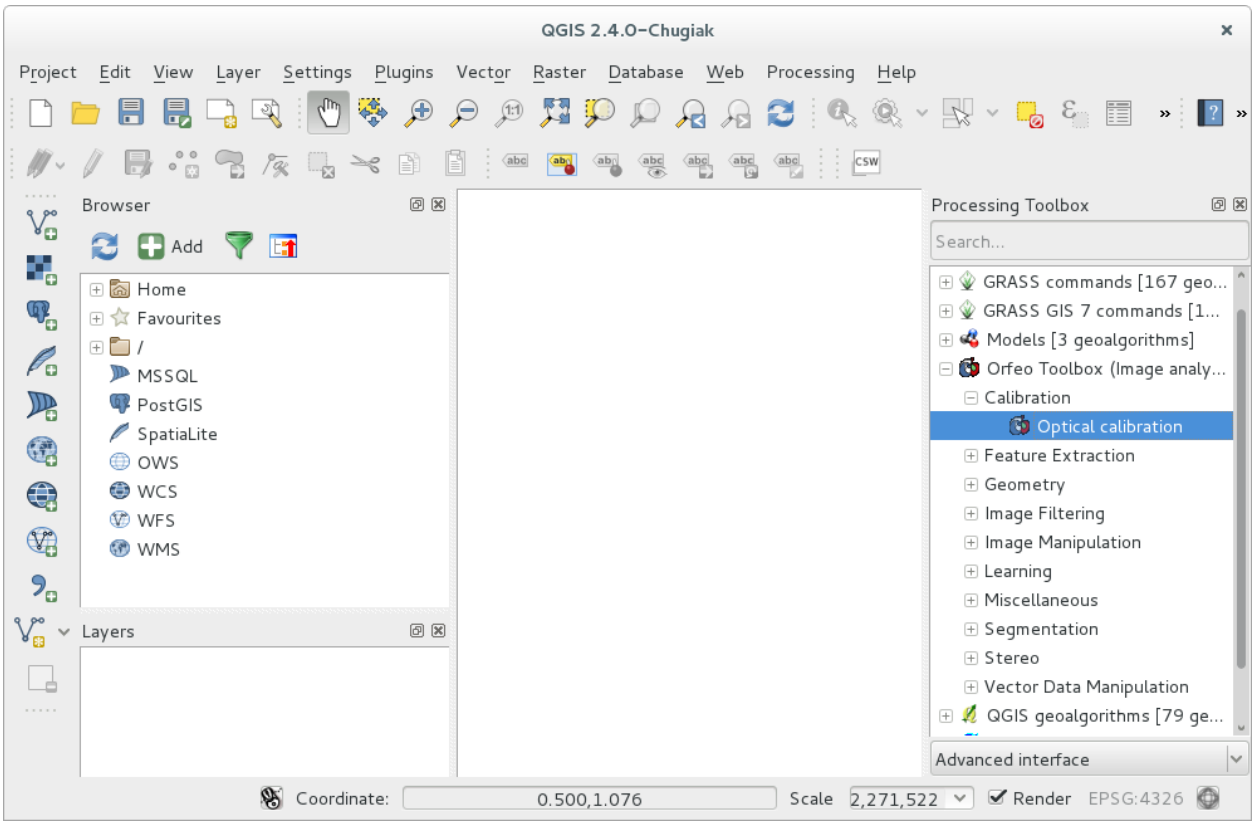
For more information about this Python interface, check the recipe section.


Rescale Image - version 3.11.0
Parameters
Logs
Progress
Documentation
Rescale Image
Brief Description
Rescale the image between two given values.
Tags
Image Manipulation, Image Manipulation
Long Description
This application scale the given image pixel intensity between two given values.
By default min (resp. max) value is set to 0 (resp. 255).
Parameters
[param] Input Image: The image to scale.
[param] Output Image: The rescaled image filename.
Ready to run
Execute
Quit

QGIS interface

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

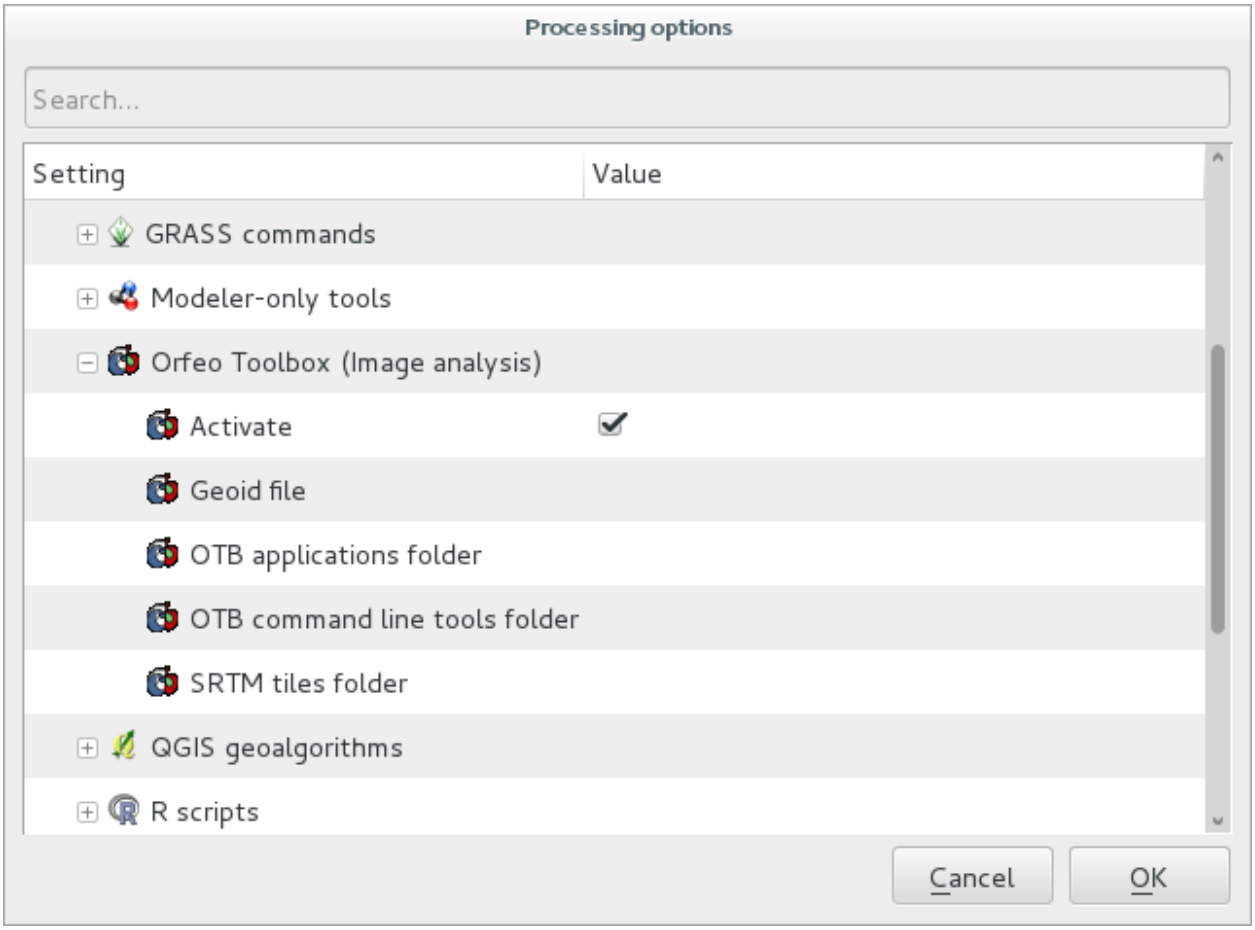
If QGIS cannot find OTB, the "applications folder" and "binaries folder" can be set from the settings in the Processing → Settings → "service provider".

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.

Load and save parameters to XML

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


Processing options
Search...
Setting
Value
GRASS commands
Modeler-only tools
Orfeo Toolbox (Image analysis)
Activate
Geoid file
OTB applications folder
OTB command line tools folder
SRTM tiles folder
QGIS geoalgorithms
R scripts
Cancel
OK

```
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.

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 REQSTED     : 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.

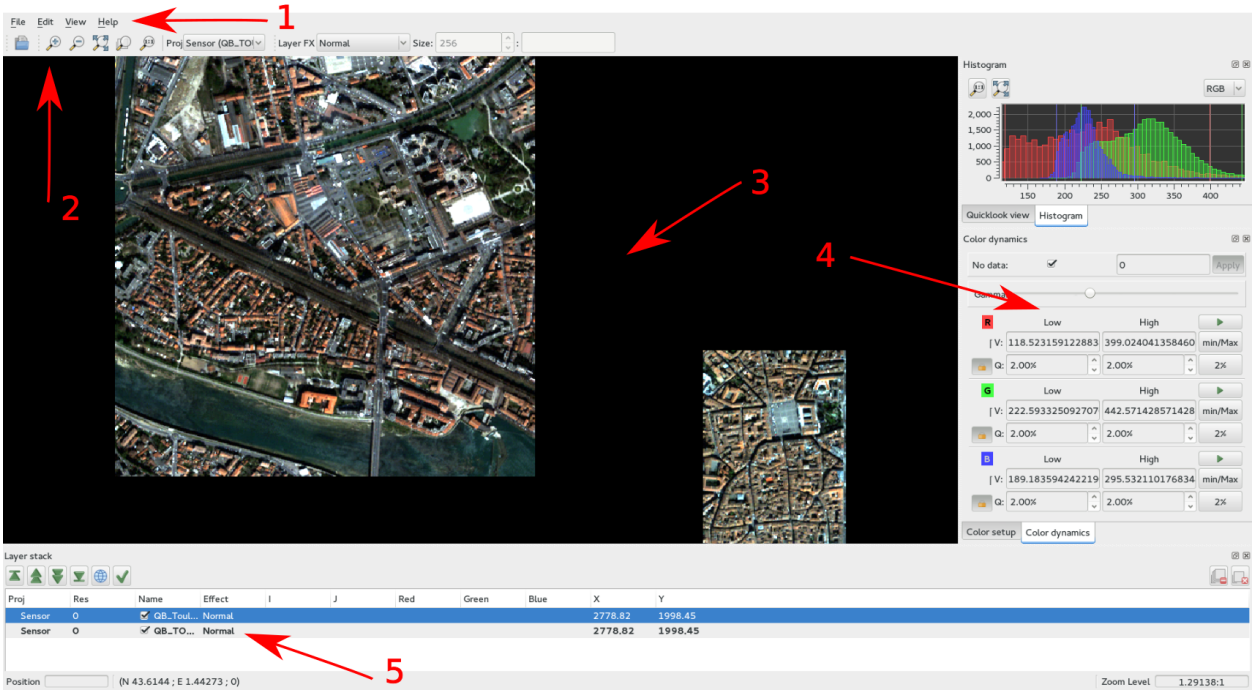
CHAPTER

FOUR

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!



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

1. Main menu
2. Top toolbar
3. Image View
4. Widgets
5. Layer stack

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.

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 constrast, 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 constrast, spectral angle

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 provided within the Help item of the main menu under Keymap. Here is a short list of the most commonly used ones:

The standard 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)
- Mouse wheel = Zoom
- – or - = Zoom

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

Right side dock

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

- Quicklook: provides an overview of the full extent of the image, and allows one to 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]).

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

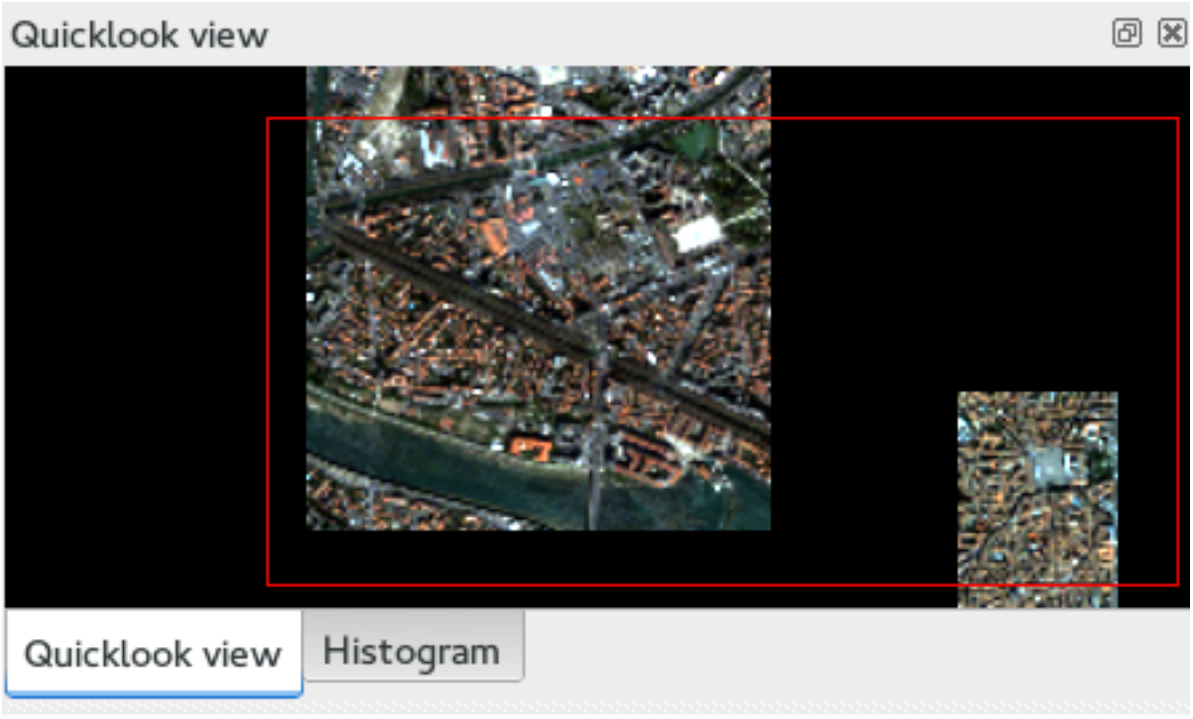
Concerning the six icons, from left to right:

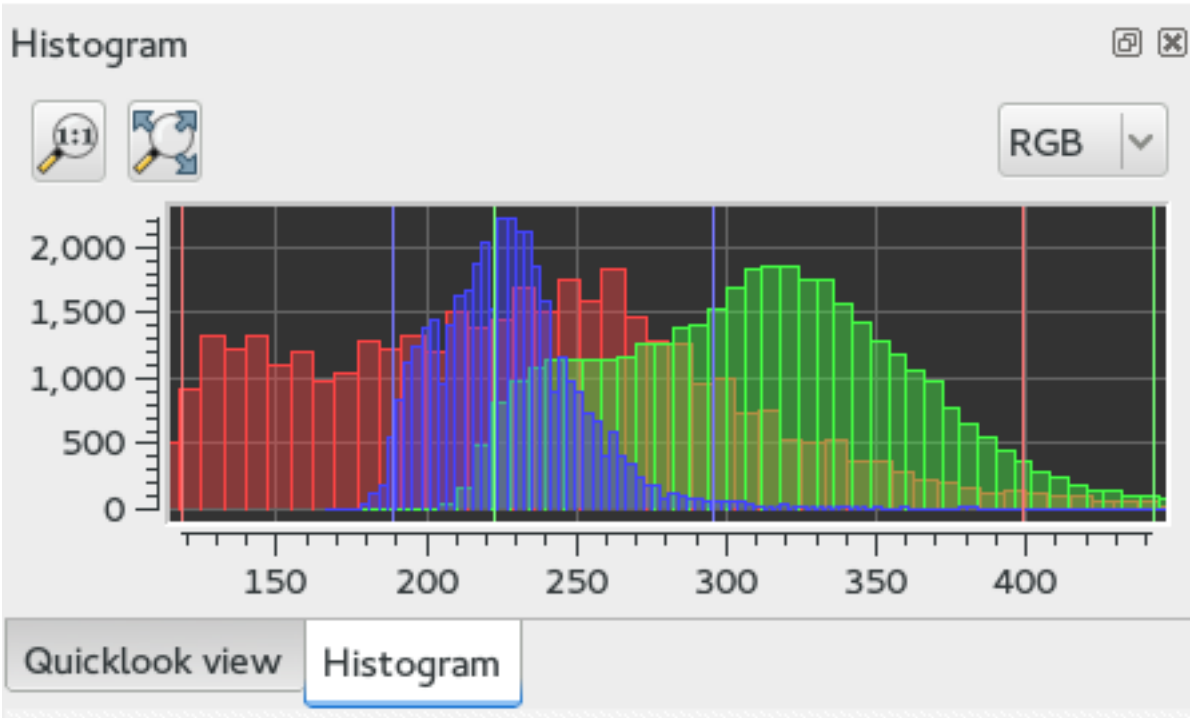
- 1st: moves the selected layer to the top of the stack
- 2nd: moves the selected layer up within the stack
- 3rd: moves the selected layer down within the stack
- 4th: moves the selected layer to the bottom of the stack
- 5th: use selected layer as projection reference
- 6th: applies all display settings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

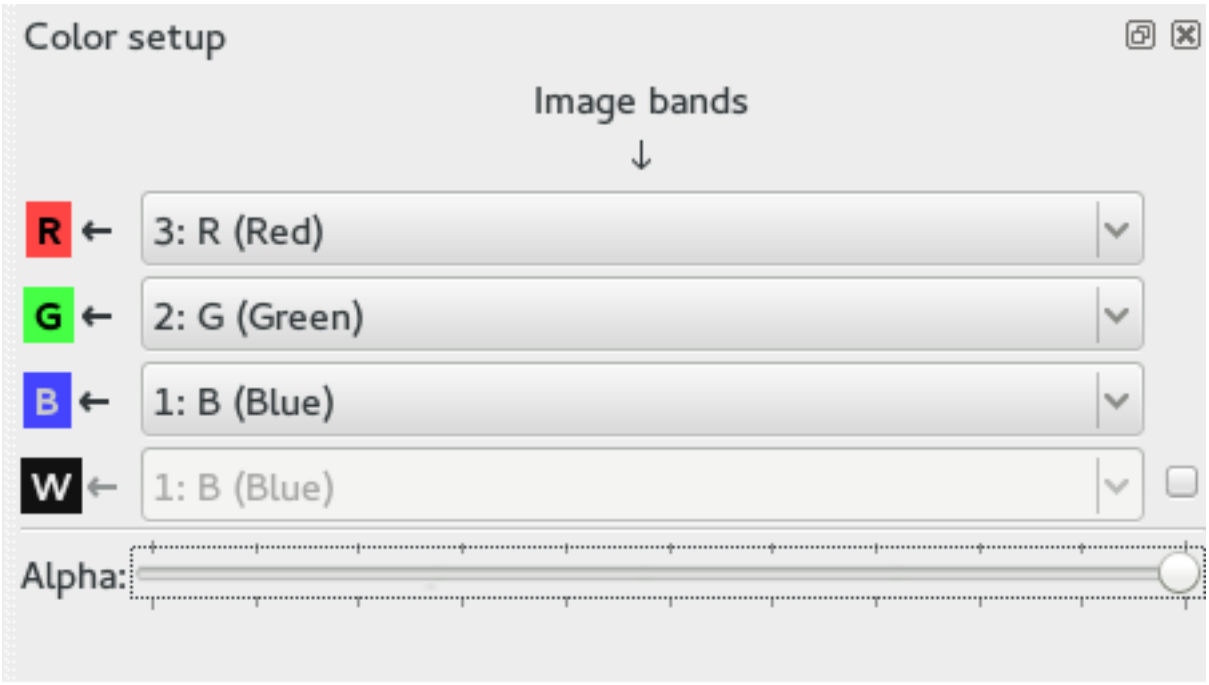
The layer stack is represented in the figure below ([fig:layerstack]):

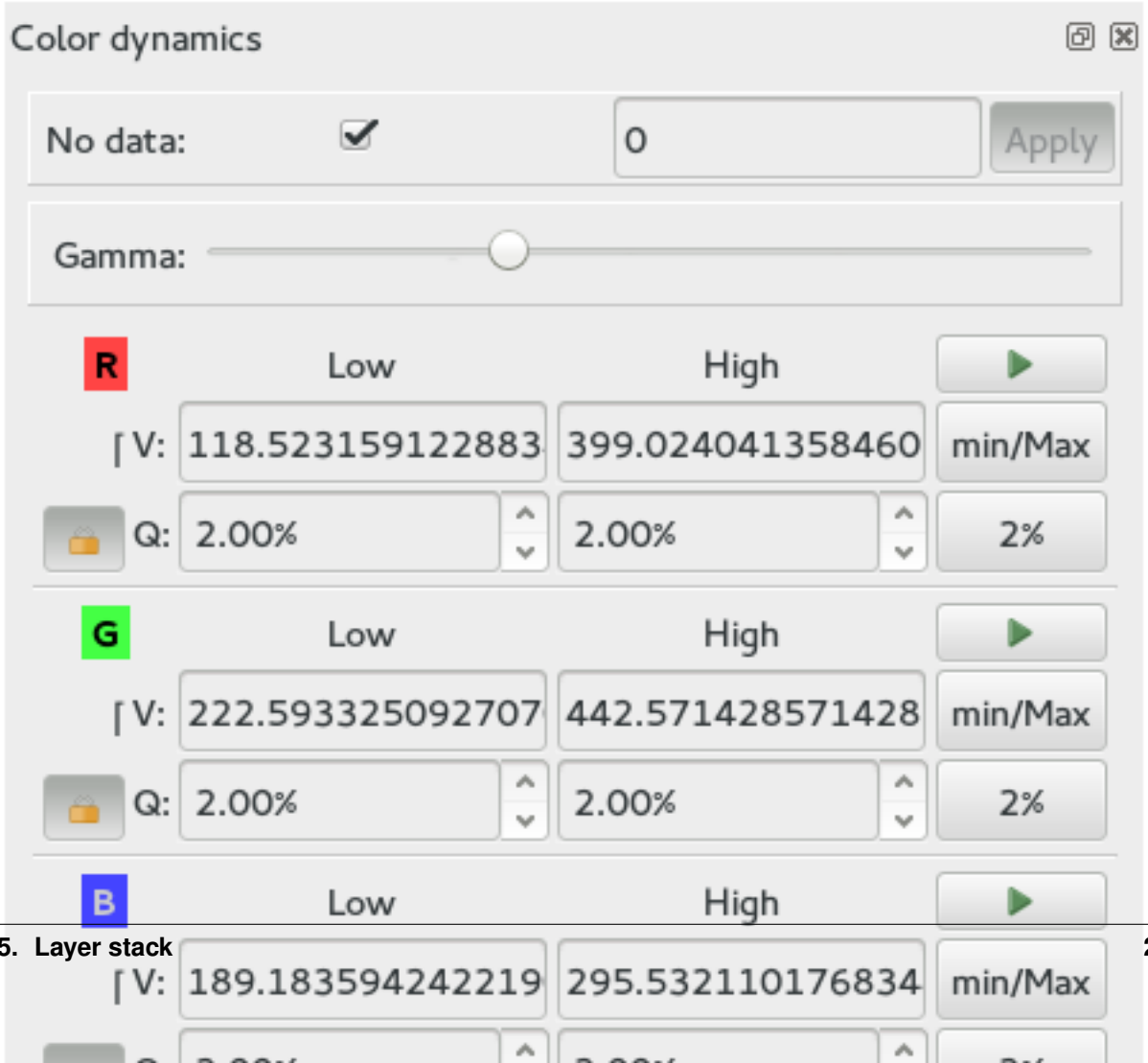
Examples

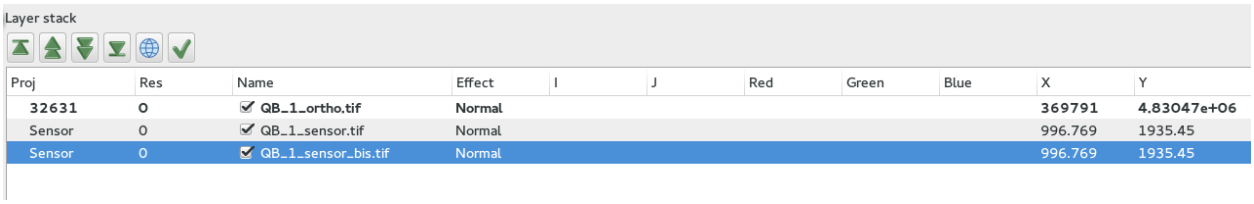
With , it is also possible to interactively load otb-applications and use them to process images. For that purpose, the user just has to load otb-applications by clicking on the Main menu, File/Load OTB-Applications (or by simply using the shortcut CTRL+A). The figure below ([fig:applications]) represents the otb-applications loading window. The


Quicklook view
Quicklook view
Histogram


Histogram
1:1
RGB
2,000
1,500
1,000
500
0
150
200
250
300
350
400
Quicklook view
Histogram


Color setup
Image bands
R
3: R (Red)
G
2: G (Green)
B
1: B (Blue)
W
1: B (Blue)
Alpha:


Color dynamics
No data:
0
Apply
Gamma:
R
Low
High
V:
118.523159122883
399.024041358460
min/Max
Q:
2.00%
2.00%
2%
G
Low
High
V:
222.593325092707
442.571428571428
min/Max
Q:
2.00%
2.00%
2%
B
Low
High
V:
189.183594242219
295.532110176834
min/Max



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.

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.

BandMath

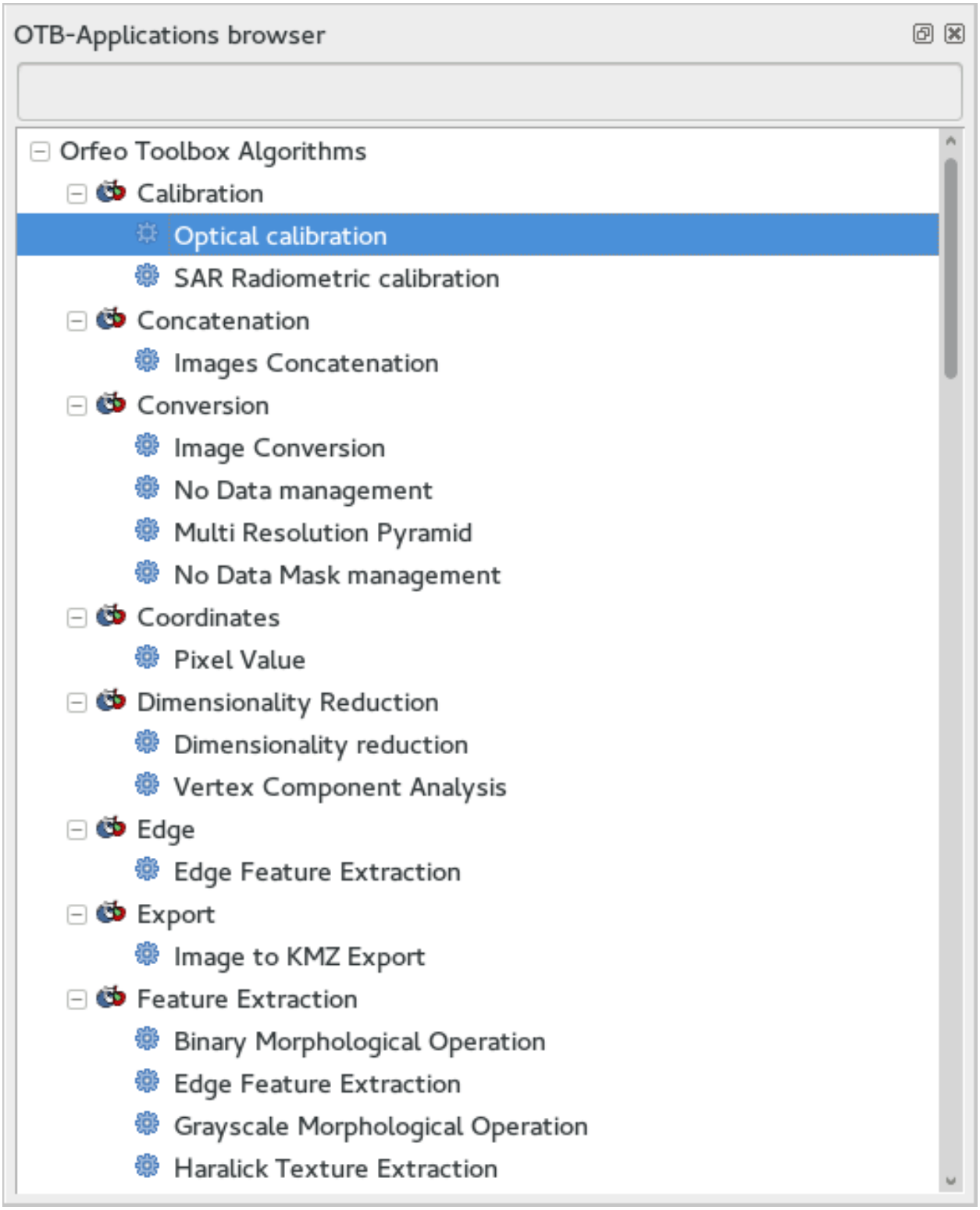
BandMath application is intended to apply mathematical operations on pixels (launch it with shortcut CTRL+A). In this example, we are going to use this application to change the dynamics of an image, and check the result by looking at the histogram tab on the right-hand side of the GUI. The formula used is the following: $\text{im1b1} \times 1000$. 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 approximately 1000 times further away (the cursors aren't placed exactly at the same position in the screenshots).

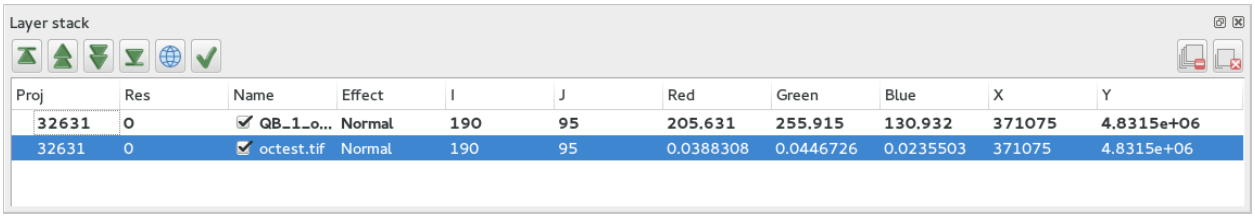
Segmentation

From within Monteverdi, the Segmentation application can be launched using the 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


OTB-Applications browser
Orfeo Toolbox Algorithms
Calibration
Optical calibration
SAR Radiometric calibration
Concatenation
Images Concatenation
Conversion
Image Conversion
No Data management
Multi Resolution Pyramid
No Data Mask management
Coordinates
Pixel Value
Dimensionality Reduction
Dimensionality reduction
Vertex Component Analysis
Edge
Edge Feature Extraction
Export
Image to KMZ Export
Feature Extraction
Binary Morphological Operation
Edge Feature Extraction
Grayscale Morphological Operation
Haralick Texture Extraction



to convenience. The figure below ([fig:seg4]) shows the result of such colorization. We encourage the reader to test the other layer effects.

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).

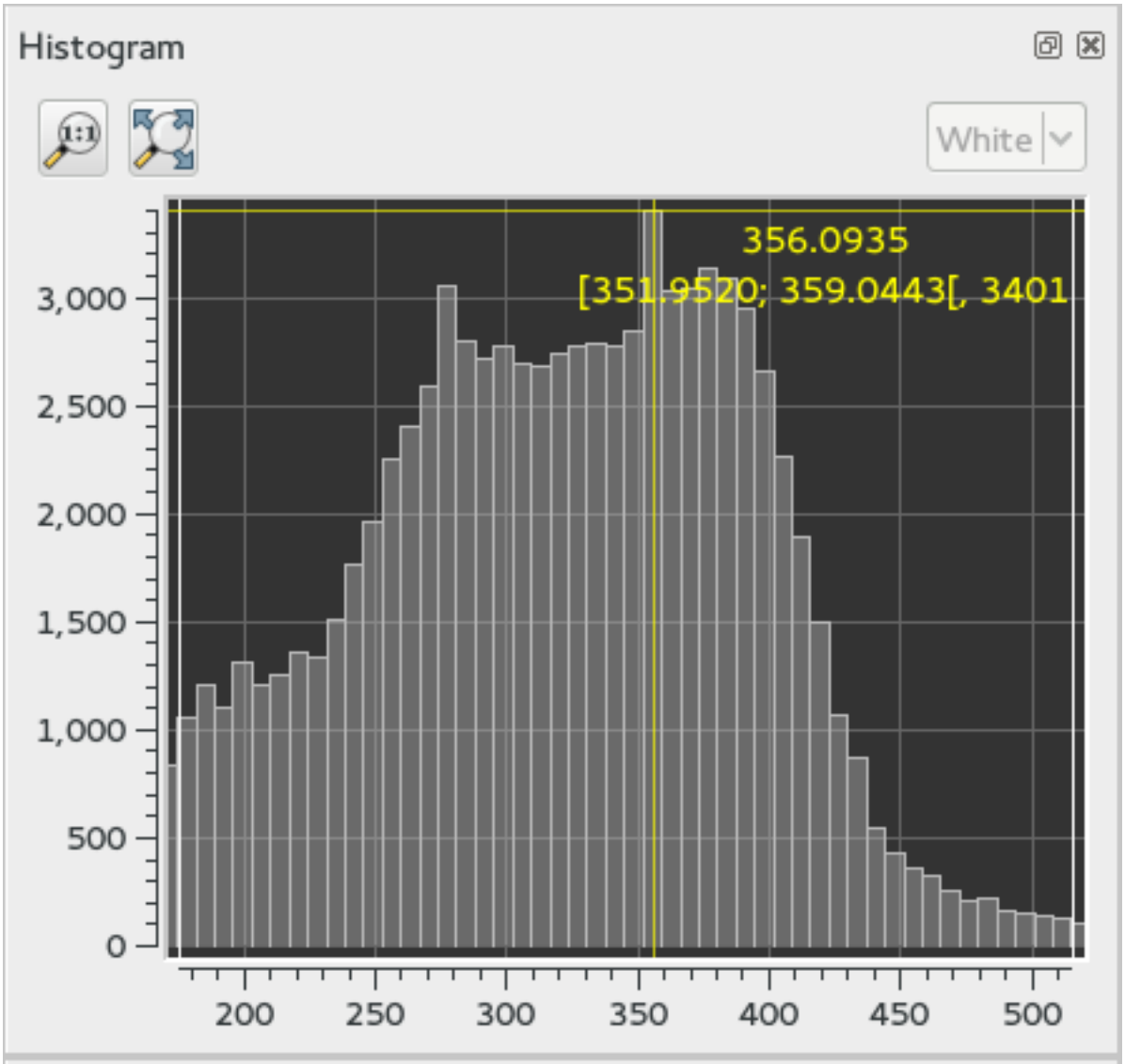
In the figure below ([fig:pol1]), we show the obtained result, with the local contrast layer effect.

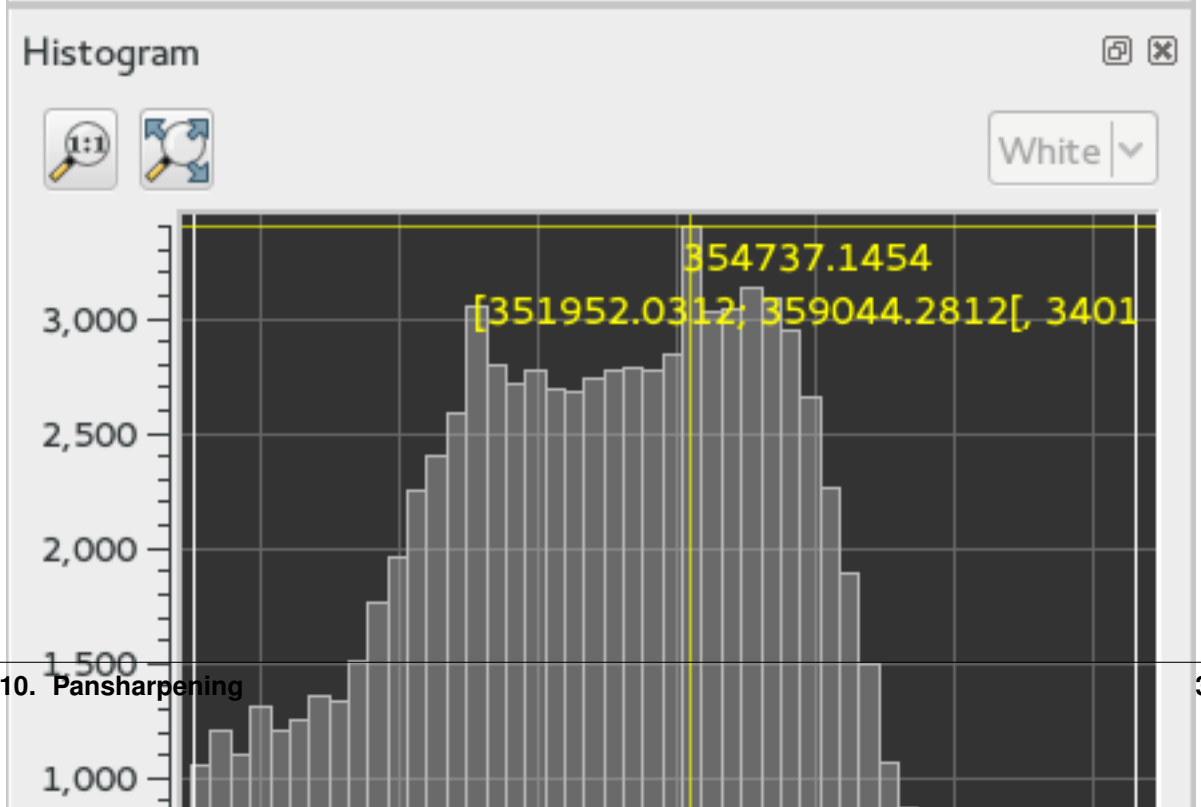
Pansharpening

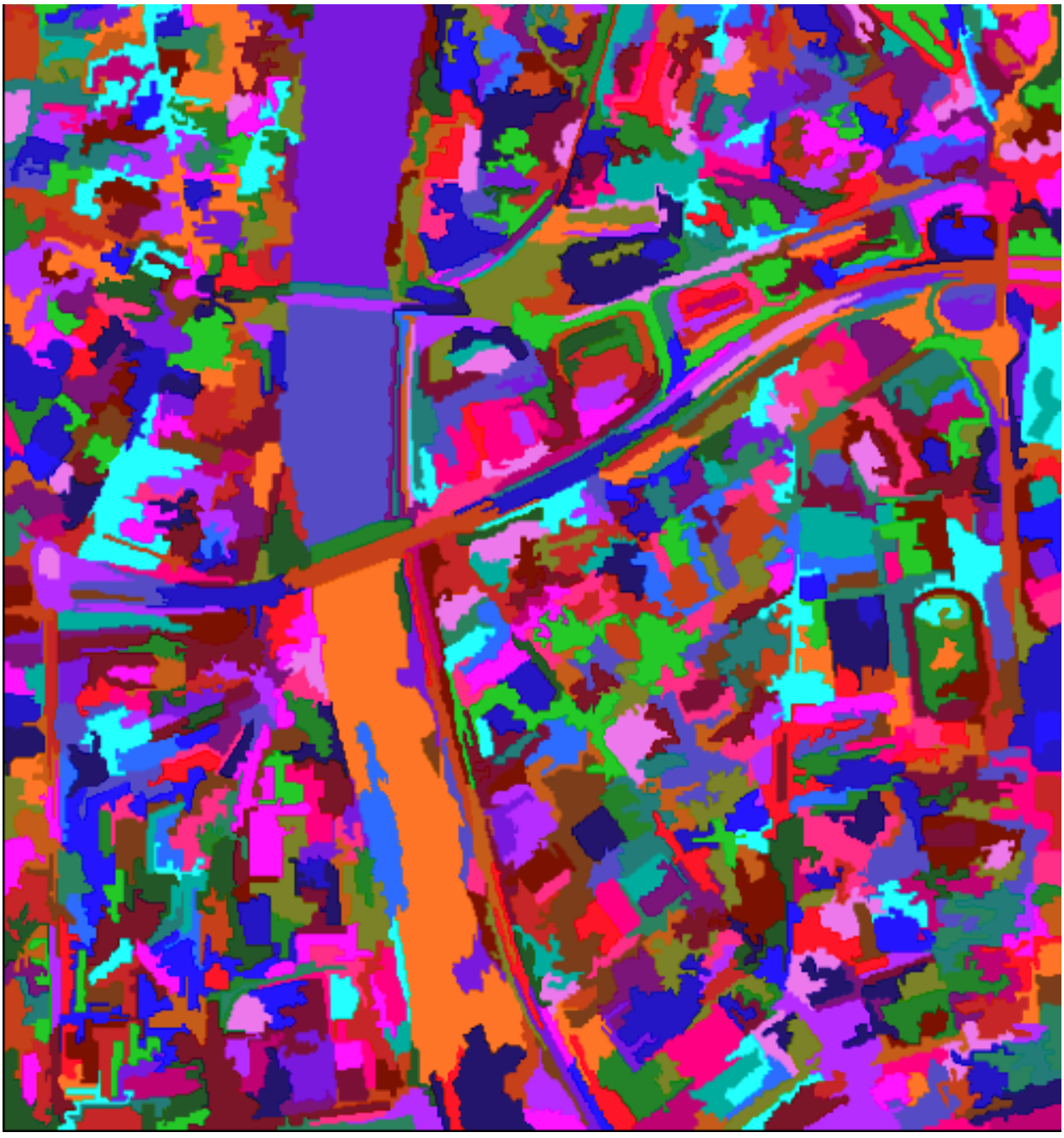
Finally, let's try a last example with the Pansharpening application (launch it with shortcut CTRL+A). The fields are quite easy to fill in: this application needs a panchromatic image, a XS image, and an output image. These images are represented in the figures below ([fig:ps12] and [fig:ps3]):

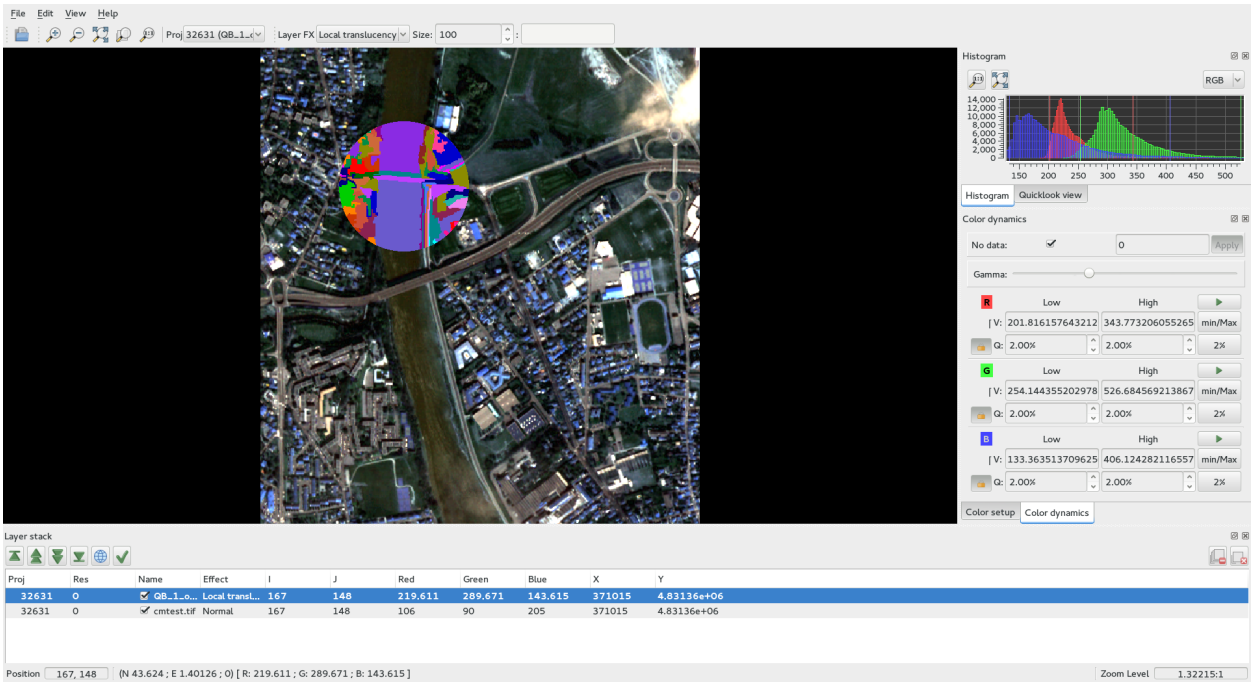
Now, in order to inspect the result properly, these three images are loaded in . The pansharpened image is placed to the top of the stack layer, and different layer effects are applied to it:

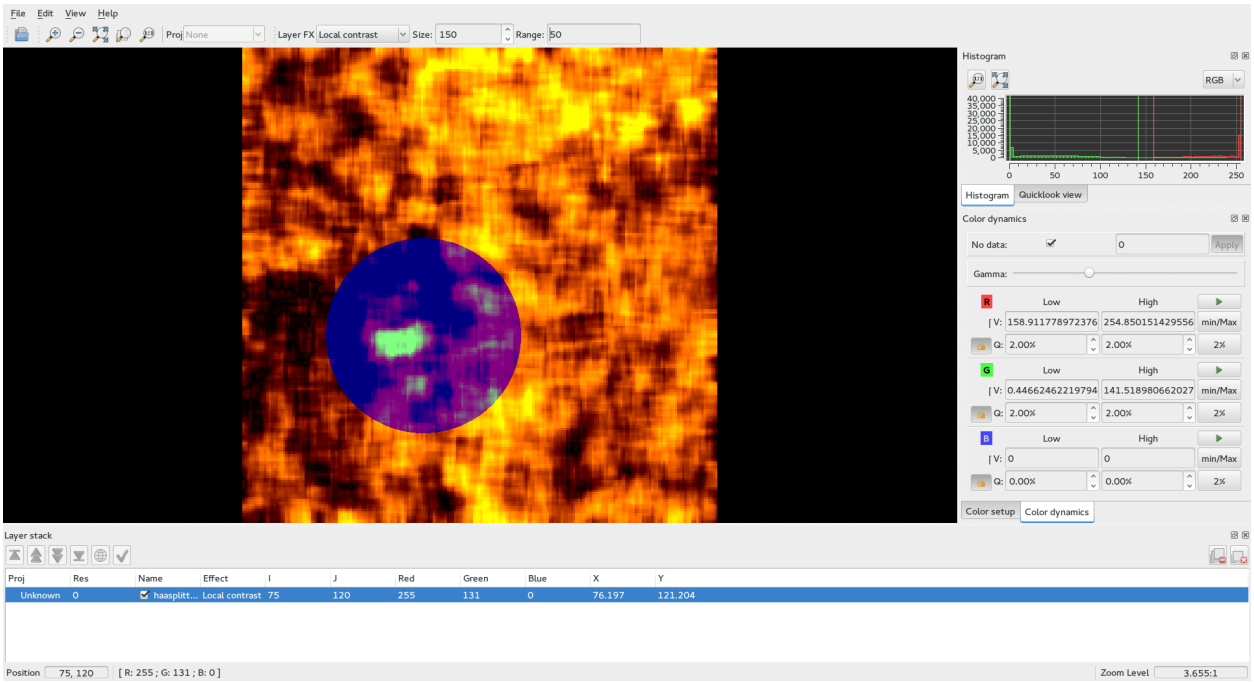
- in figure [fig:ps4]: chessboard effect, to compare the result with the XS image.
- in figure [fig:ps5]: translucency effect, to compare the result with the panchromatic image.

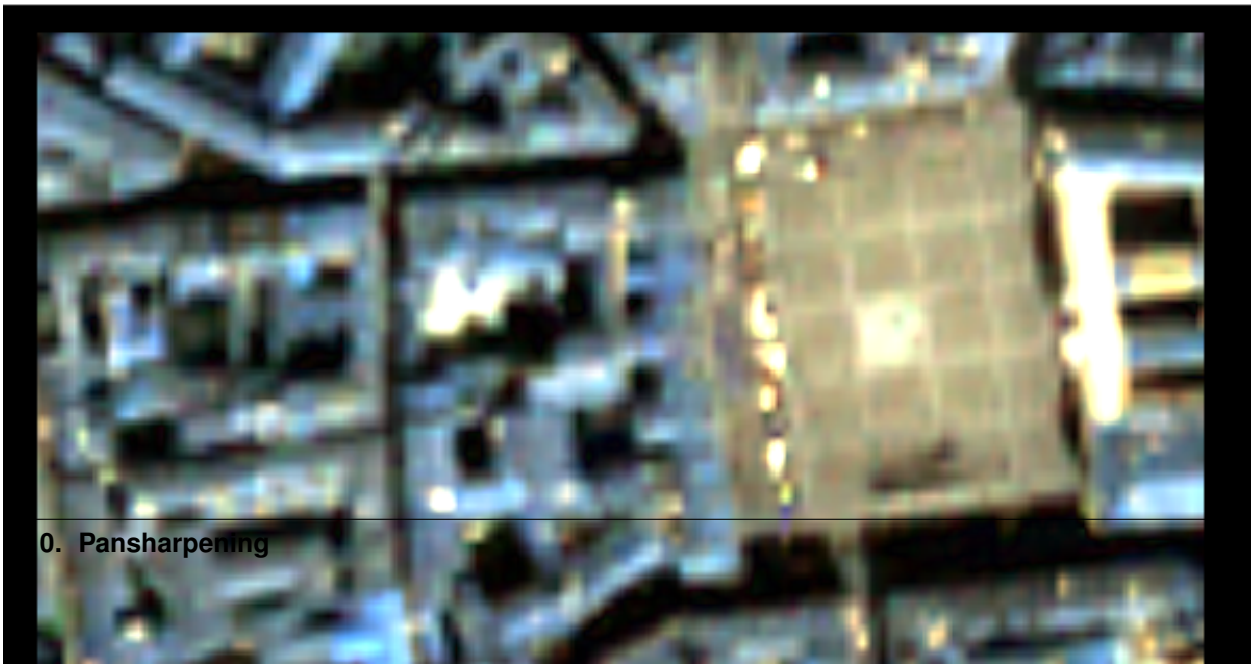

Histogram
1:1
White
356.0935
[351.9520; 359.0443[, 3401
3,000
2,500
2,000
1,500
1,000
500
0
200
250
300
350
400
450
500


Histogram
1:1
White
354737.1454
[351952.0312; 359044.2812[, 3401
3,000
2,500
2,000
1,500
1,000

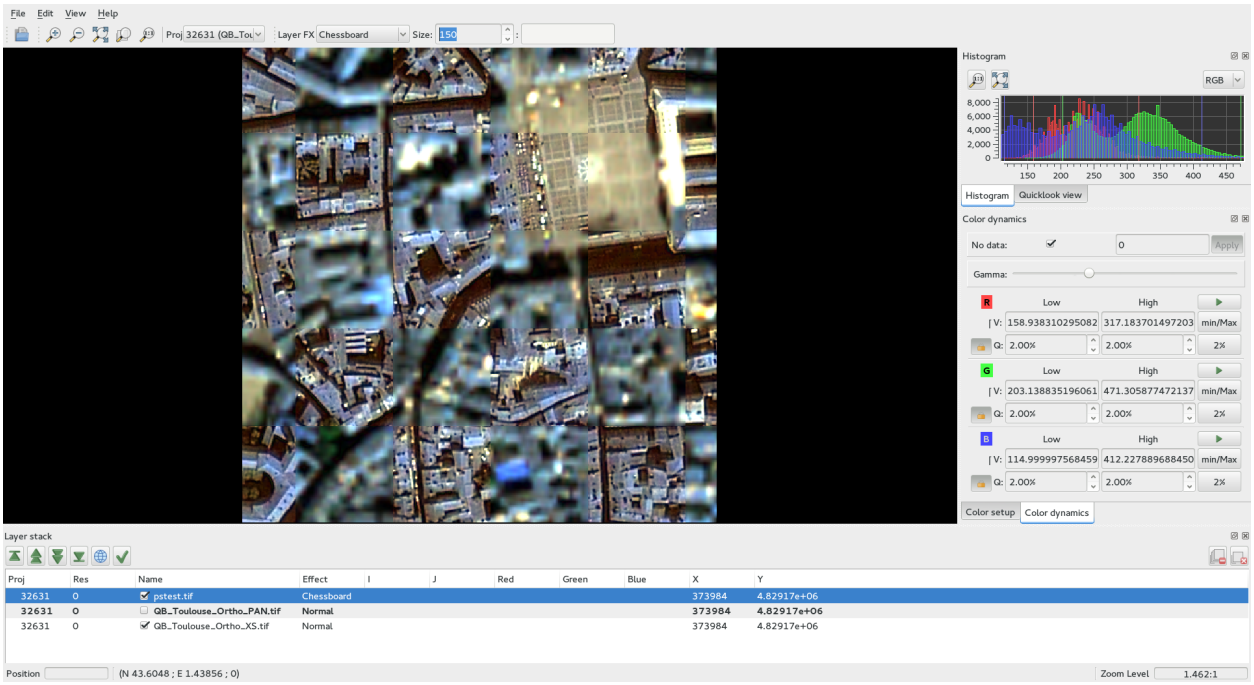



File Edit View Help
Proj 32631 (QB_1_c
Layer FX Local translucency
Size: 100
Histogram
RGB
Histogram Quicklook view
Color dynamics
No data:
Apply
Gamma:
Low High
[V: 201.816157643212 343.773206055265 min/Max
Q: 2.00% 2.00% 2%
[V: 254.144355202978 526.684569213867 min/Max
[V: 133.363513709625 406.124282116557 min/Max
Color setup Color dynamics
Layer stack
Proj Res Name Effect I J Red Green Blue X Y
32631 0 QB_1_o... 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


File Edit View Help
Proj None
Layer FX Local contrast
Size: 150
Range: 50
Histogram
RGB
Histogram Quicklook view
Color dynamics
No data:
Apply
Gamma:
Low High
[V: 158.911778972376 254.850151429556 min/Max
Q: 2.00% 2.00% 2%
[V: 0.44662462219794 141.518980662027 min/Max
[V: 0 0 min/Max
Q: 0.00% 0.00% 2%
Color setup Color dynamics
Layer stack
Proj Res Name Effect I J Red Green Blue X Y
Unknown 0 haasplitt... 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



0. Pansharpening


File Edit View Help
Proj 32631 (QB_Tou
Layer FX Chessboard
Size: 150
Histogram
RGB
Histogram
Quicklook view
Color dynamics
No data:
Apply
Gamma:
Low
High
min/Max
2x
Color setup
Color dynamics
Layer stack
Proj Res Name Effect I J Red Green Blue X Y
32631 0 pstest.tif Chessboard 373984 4.82917e+06
32631 0 QB_Toulouse_Ortho_PAN.tif Normal 373984 4.82917e+06
32631 0 QB_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 (QB_Tou
Layer FX Local translucency
Size: 80
Histogram
RGB
Histogram
Quicklook view
Color dynamics
No data:
Apply
Gamma:
Low
High
min/Max
2x
Color setup
Color dynamics
Layer stack
Proj Res Name Effect I J Red Green Blue X Y
32631 0 pstest.tif Local transl... 374 111 283 409 301 374375 4.82912e+06
32631 0 QB_Toulouse_Ortho_PAN.tif Normal 374 111 339 339 339 374375 4.82912e+06
32631 0 QB_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

Conclusion

The images used in this documentation can be found in the OTB-Data repository (https://gitlab.orfeo-toolbox.org/orfeotoolbox/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

CHAPTER

FIVE

ADVANCED USE

This section describes advanced configuration options and tricks.

Environment variables that affects Orfeo ToolBox

The following environment variables are parsed by Orfeo ToolBox. Note that they only affect default values, and that settings in extended filenames, applications, monteverdi or custom C++ code might override those values.

- `OTB_DEM_DIRECTORY`: Default directory were DEM tiles are stored. It should only contain `.hgt` or or georeferenced `.tif` files. Empty if not set (no directory set)
- `OTB_GEOID_FILE`: Default path to the geoid file that will be used to retrieve height of DEM above ellipsoid. Empty if not set (no geoid set)
- `OTB_MAX_RAM_HINT`: Default maximum memory that OTB should use for processing, in MB. If not set, default value is 128 MB.
- `OTB_LOGGER_LEVEL`: Default level of logging for OTB. Should be one of `DEBUG`, `INFO`, `WARNING`, `CRITICAL` or `FATAL`, by increasing order of priority. Only messages with a higher priority than the level of logging will be displayed. If not set, default level is `INFO`.

Extended filenames

Extended filenames is an interesting feature of OTB. With it, you can control several aspects of the beahvior of the OTB in the OTB-Applications or in our own C++ applications. Historically this feature has been desingn to solve an issue with how to handle geo-referencing information.

Indeed, there are multiple ways to define geo-referencing information. For instance, one can use a geographic transform, a cartographic projection, or a sensor model with RPC coefficients. A single image may contain several of these elements, such as in the "ortho-ready" products: this is a type of product still in sensor geometry (the sensor model is supplied with the image) but it also contains an approximative geographic transform that can be used to have a quick estimate of the image localisation. For instance, your product may contain a ".TIF" file for the image, along with a ".RPB" file that contains the sensor model coefficients and an ".IMD" file that contains a cartographic projection.

This case leads to the following question: which geo-referencing element should be used when opening this image in OTB. In fact, it depends on the users need. For an orthorectification application, the sensor model must be used. In order to specify which information should be skipped, a syntax of extended filenames has been developed for both reading and writing.

Since the development of this feature we have extend this mechanism for other aspaects: like band or overview selection in reader part or support create option of gdal in writer part.The reader and writer extended filename support

is based on the same syntax, only the options are different. To benefit from the extended file name mechanism, the following syntax is to be used:

```
Path/Image.ext?&key1=<value1>&key2=<value2>
```

Note that you'll probably need to "quote" the filename, especially if calling applications from the bash command line.

Reader options

```
&geom=<path/filename.geom>
```

- Contains the file name of a valid geom file
- Use the content of the specified geom file instead of image-embedded geometric information
- empty by default, use the image-embedded information if available

---

```
&sdataidx=<(int)idx>
```

- Select the sub-dataset to read
- 0 by default

```
&resol=<(int)resolution factor>
```

- Select the JPEG2000 sub-resolution image to read
- 0 by default

```
&bands=r1,r2,...,rn
```

- Select a subset of bands from the input image
- The syntax is inspired by Python indexing syntax with bands=r1,r2,r3,...,rn where each ri is a band range that can be :
  - a single index (1-based) :
    - `2` means 2nd band
    - `-1` means last band
  - or a range of bands :
    - `3:` means 3rd band until the last one
    - `:-2` means the first bands until the second to last
    - `2:4` means bands 2,3 and 4
- empty by default (all bands are read from the input image)

```
&skipcarto=<(bool)true>
```

- Skip the cartographic information
- Clears the projectionref, set the origin to $[0, 0]$ and the spacing to $[1/max(1, r), 1/max(1, r)]$ where r is the resolution factor.
- Keeps the keyword list
- false by default

```
&skipgeom=<(bool)true>
```

- Skip geometric information
- Clears the keyword list
- Keeps the projectionref and the origin/spacing information
- false by default.

```
&skiprpctag=<(bool)true>
```

- Skip the reading of internal RPC tags (see [sec:TypesofSensorModels] for details)
- false by default.

Writer options

```
&writegeom=<(bool)false>
```

- To activate writing of external geom file
- true by default

```
&writerpctags=<(bool)true>
```

- To activate writing of RPC tags in TIFF files
- false by default

```
&gdal:co:<GDALKEY>=<VALUE>
```

- To specify a gdal creation option
- For gdal creation option information, see dedicated gdal documentation for each driver. For example, you can find here the information about the GeoTiff create options
- None by default

```
&streaming:type=<VALUE>
```

- Activates configuration of streaming through extended filenames
- Override any previous configuration of streaming
- Allows to configure the kind of streaming to perform
- Available values are:
  - auto: tiled or stripped streaming mode chosen automatically depending on TileHint read from input files
  - tiled: tiled streaming mode
  - stripped: stripped streaming mode
  - none: explicitly deactivate streaming
- Not set by default

```
&streaming:sizemode=<VALUE>
```

- Allows to choose how the size of the streaming pieces is computed
- Available values are:
  - auto: size is estimated from the available memory setting by evaluating pipeline memory print
  - height: size is set by setting height of strips or tiles
  - nbsplits: size is computed from a given number of splits
- Default is auto

```
&streaming:sizevalue=<VALUE>
```

- Parameter for size of streaming pieces computation
- Value is :
  - if sizemode=auto: available memory in Mb
  - if sizemode=height: height of the strip or tile in pixels
  - if sizemode=nbsplits: number of requested splits for streaming
- If not provided, the default value is set to 0 and result in different behaviour depending on sizemode (if set to height or nbsplits, streaming is deactivated, if set to auto, value is fetched from configuration or cmake configuration file)

```
&box=<startx>:<starty>:<sizex>:<sizey>
```

- User defined parameters of output image region
- The region must be set with 4 unsigned integers (the separator used is the colon ':'). Values are:
  - startx: first index on X (starting with 0)
  - starty: first index on Y (starting with 0)
  - sizex: size along X

  - sizey: size along Y

- The definition of the region follows the same convention as itk::Region definition in C++. A region is defined by two classes: the itk::Index and itk::Size classes. The origin of the region within the image with which it is associated is defined by Index

---

```
&bands=r1,r2,...,rn
```

- Select a subset of bands from the output image
- The syntax is inspired by Python indexing syntax with bands=r1,r2,r3,...,rn where each ri is a band range that can be :
  - a single index (1-based) :
    - `2` means 2nd band
    - `-1` means last band
  - or a range of bands :
    - `3:` means 3rd band until the last one
    - `:-2` means the first bands until the second to last
    - `2:4` means bands 2,3 and 4
- Empty by default (all bands are write from the output image)

The available syntax for boolean options are:

- ON, On, on, true, True, 1 are available for setting a 'true' boolean value
- OFF, Off, off, false, False, 0 are available for setting a 'false' boolean value

OGR DataSource options

We extended this process to OGR DataSource. There are three different type of option : open, creation and layer creation. Those options come from the GDAL API. In order to use them one just need to specify to which of this family the option one want to use is from.

For open option :

```
&gdal:oo:<GDALKEY>=<VALUE>
```

For creation option :

```
&gdal:co:<GDALKEY>=<VALUE>
```

For layer creation option :

```
&gdal:lco:<GDALKEY>=<VALUE>
```

Examples

You can find below some examples:

- Write a file with blockSize equal to 256 and with DEFLATE compression

```
$ otbcli_Convert -in OTB-Data/Examples/QB_1_ortho.tif -out "/tmp/example1.tif?&
↪gdal:co:TILED=YES&gdal:co:COMPRESS=DEFLATE"
```

- Process only first band from a file

```
$ otbcli_Convert -in "OTB-Data/Examples/QB_1_ortho.tif?&bands=1" -out /tmp/example2.
↪tif
```

CHAPTER

SIX

RECIPES

This chapter presents guidelines to perform various remote sensing and image processing tasks with either , or both. Its goal is not to be exhaustive, but rather to familiarise users with the OTB package functionality and demonstrate how the can be applied.

From raw image to calibrated product

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

Optical radiometric calibration

In remote sensing imagery, pixel values are referred to as Digital Numbers (DN) and they cannot be physically interpreted or compared. They are influenced by various factors such as the amount of light flowing through 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
```

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 turn 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 turn 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.

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 transformations.

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 DEM with a resolution of 90 metres, 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 DEM with a resolution of 30 metres 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 contain the DEM tiles, either in DTED or SRTM format or as a GeoTIFF. Subdirectories are not supported.

Depending on the reference of the elevation, you also need to use a geoid to accurately manage the elevation. 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.

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 `gdalinfo` or OTB ReadImageInfo application. Check if your product verifies following two 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 *Extended filenames* section.

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.sizex x_size
                          -outputs.sizey 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.sizex` and `-outputs.sizey` 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.

SAR processing

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

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 is 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
```

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 ([ffig: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.gammamap.rad 5
                 -filter.gammamap.nblooks 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.

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 $2x2$ 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).

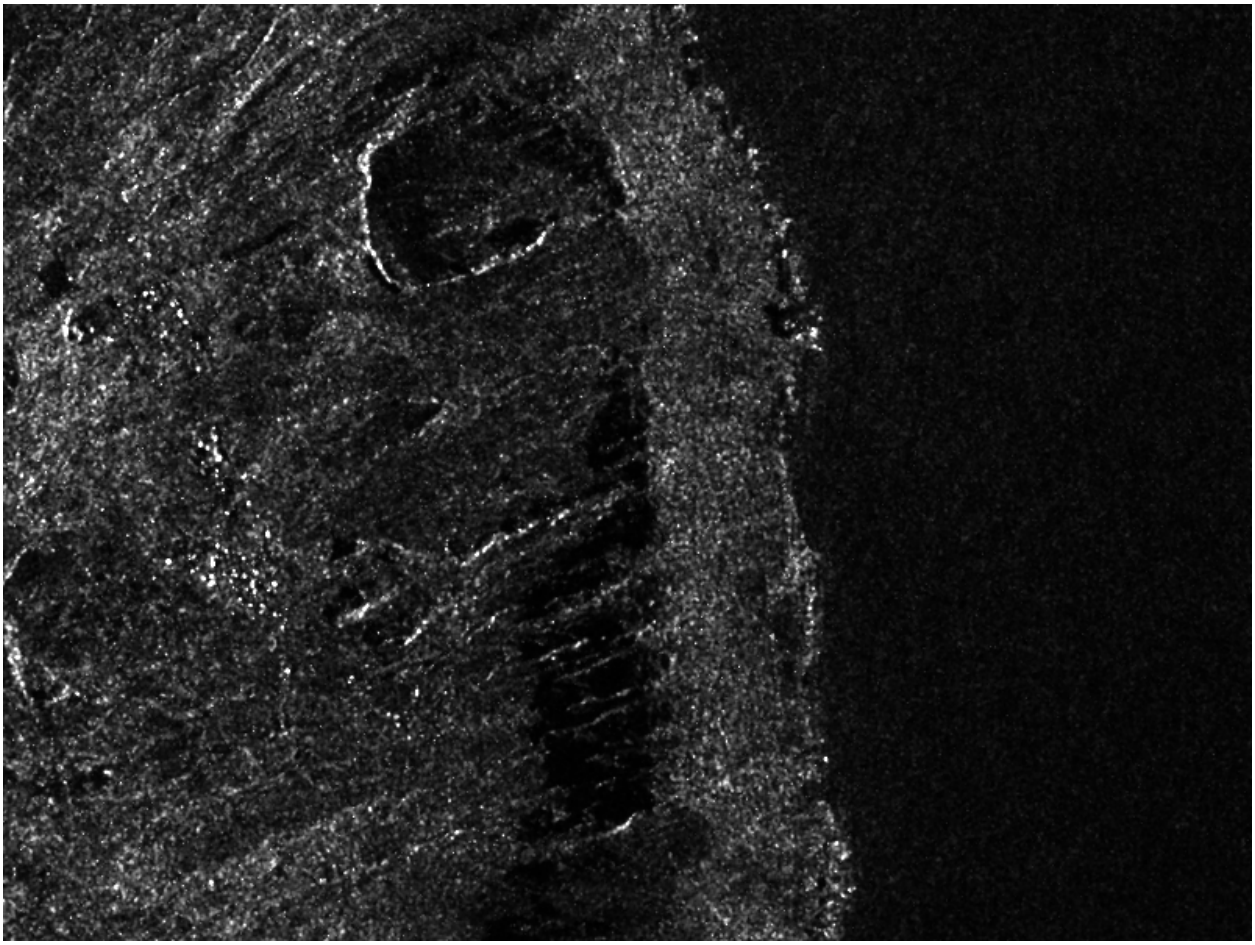
Matrix conversions

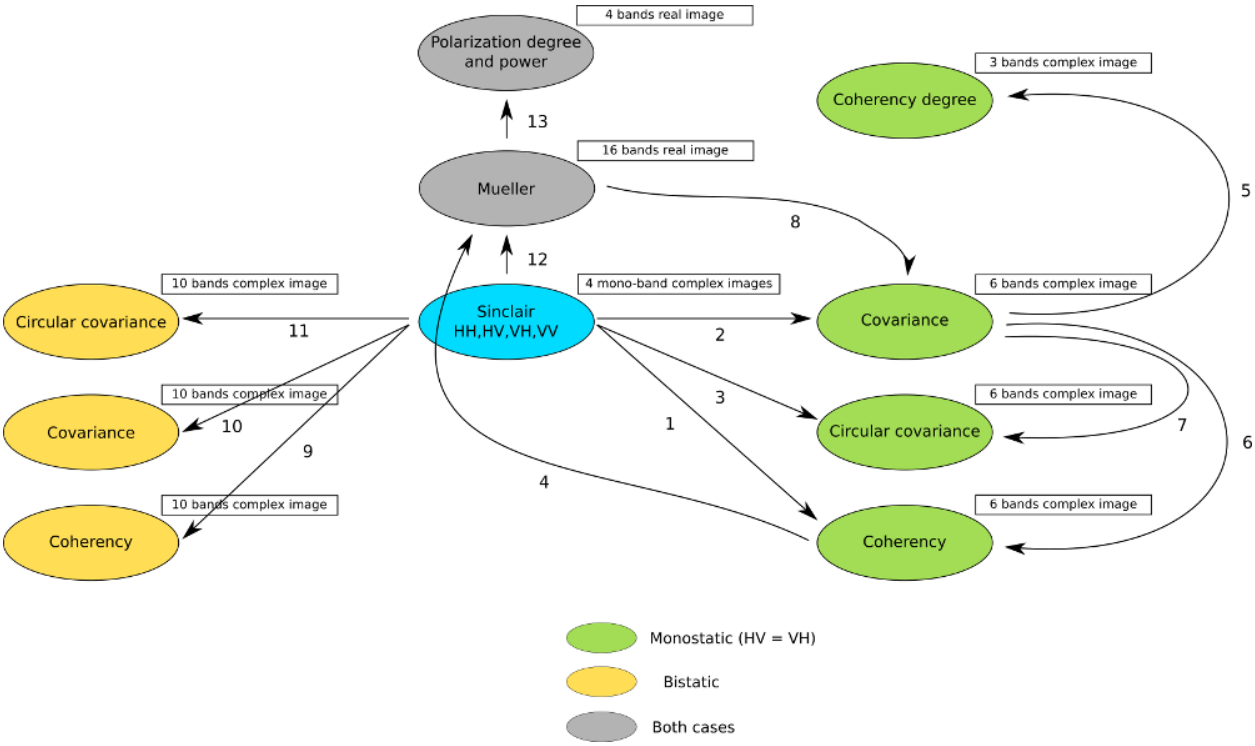
This applications allows converting classical polarimetric matrices to each other. For instance, it is possible to get the coherency matrix from the Sinclair one, or the Mueller matrix from the coherency one. The figure below ([fig:polconv]) shows the workflow used in this application.

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 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).

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




4 bands real image
Polarization degree and power
13
16 bands real image
Mueller
12
8
3 bands complex image
Coherency degree
5
10 bands complex image
Circular covariance
11
4 mono-band complex images
Sinclair HH,HV,VH,VV
2
6 bands complex image
Covariance
10 bands complex image
Covariance
10
9
3
1
6 bands complex image
Circular covariance
7
6
10 bands complex image
Coherency
4
6 bands complex image
Coherency
Monostatic (HV = VH)
Bistatic
Both cases

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 (SinclairToReciprocalCoherencyMatrixFunctor)
   (a) $0.5.(S_{hh} + S_{vv}).(S_{hh} + S_{vv})^*$
   (b) $0.5.(S_{hh} + S_{vv}).(S_{hh} - S_{vv})^*$
   (c) $0.5.(S_{hh} + S_{vv}).(2S_{hv})^*$
   (d) $0.5.(S_{hh} - S_{vv}).(S_{hh} - S_{vv})^*$
   (e) $0.5.(S_{hh} - S_{vv}).(2S_{hv})^*$
   (f) $0.5.(2S_{hv}).(2S_{hv})^*$
2. msinclairtocovariance (SinclairToReciprocalCovarianceMatrixFunctor)
   (a) $S_{hh}.S_{hh}^*$
   (b) $\sqrt{2}.S_{hh}.S_{hv}^*$
   (c) $S_{hh}.S_{vv}^*$
   (d) $2.S_{hv}.S_{hv}^*$
   (e) $\sqrt{2}.S_{hv}.S_{vv}^*$
   (f) $S_{vv}.S_{vv}^*$
3. msinclairtocircovariance (SinclairToReciprocalCircularCovarianceMatrixFunctor)
   (a) $S_{ll}.S_{ll}^*$
   (b) $S_{ll}.S_{lr}^*$
   (c) $S_{ll}.S_{rr}^*$
   (d) $S_{lr}.S_{lr}^*$
   (e) $S_{lr}.S_{rr}^*$
   (f) $S_{rr}.S_{rr}^*$

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) $abs(S_{hh}.S_{vv}^*)/sqrt(S_{hh}.S_{hh}^*)/sqrt(S_{vv}.S_{vv}^*)$
   (b) $abs(S_{hv}.S_{vv}^*)/sqrt(S_{hv}.S_{hv}^*)/sqrt(S_{vv}.S_{vv}^*)$
   (c) $abs(S_{hh}.S_{hv}^*)/sqrt(S_{hh}.S_{hh}^*)/sqrt(S_{hv}.S_{hv}^*)$

6. mcovariancetocoherency (ReciprocalCovarianceToReciprocalCoherencyFunctor)
   (a) $0.5.(C_{33} + C_{13} + C_{13}^* + C_{11})$
   (b) $0.5.(-C_{33} - C_{13} + C_{13}^* + C_{11})$
   (c) $0.5.(\sqrt{2}.C_{12} + \sqrt{2}.C_{23}^*)$
   (d) $0.5.(C_{33} - C_{13} - C_{13}^* + C_{11})$
   (e) $0.5.(\sqrt{2}.C_{12} - \sqrt{2}.C_{23}^*)$
   (f) $0.5.(2.C_{22})$

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

7. mlinearcovariancetocircularcovariance (ReciprocalLinearCovarianceToReciprocalCircularCovarianceFunctor)
   (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 (MuellerToReciprocalCovarianceFunctor)
   (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 (SinclairToCoherencyMatrixFunctor)
   (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. bsinclairtocovariance (SinclairToCovarianceMatrixFunctor)
   (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) $S_{hv}.S_{vv}^*$

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

(i) $S_{vh}.S_{vv}^*$

(j) $S_{vv}.S_{vv}^*$

3. bsinclairtocircovariance (SinclairToCircularCovarianceMatrixFunctor)

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

(b) $S_{ll}.S_{lr}^*$

(c) $S_{ll}.S_{rl}^*$

(d) $S_{ll}.S_{rr}^*$

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

(f) $S_{lr}.S_{rl}^*$

(g) $S_{lr}.S_{rr}^*$

(h) $S_{rl}.S_{rl}^*$

(i) $S_{rl}.S_{rr}^*$

(j) $S_{rr}.S_{rr}^*$

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) $0.5Re(T_{xx}.T_{xx}^* + T_{xy}.T_{xy}^* + T_{yx}.T_{yx}^* + T_{yy}.T_{yy}^*)$

(b) $0.5Re(T_{xx}.T_{xx}^* - T_{xy}.T_{xy}^* + T_{yx}.T_{yx}^* - T_{yy}.T_{yy}^*)$

(c) $Re(T_{xx}.T_{xy}^* + T_{yx}.T_{yy}^*)$

(d) $Im(T_{xx}.T_{xy}^* + T_{yx}.T_{yy}^*)$

(e) $0.5Re(T_{xx}.T_{xx}^* + T_{xy}.T_{xy}^* - T_{yx}.T_{yx}^* - T_{yy}.T_{yy}^*)$

(f) $0.5Re(T_{xx}.T_{xx}^* - T_{xy}.T_{xy}^* - T_{yx}.T_{yx}^* + T_{yy}.T_{yy}^*)$

(g) $Re(T_{xx}.T_{xy}^* - T_{yx}.T_{yy}^*)$

(h) $Im(T_{xx}.T_{xy}^* - T_{yx}.T_{yy}^*)$

(i) $Re(T_{xx}.T_{yx}^* + T_{xy}.T_{yy}^*)$

(j) $Im(T_{xx}.T_{yx}^* - T_{xy}.T_{yy}^*)$

(k) $Re(T_{xx}.T_{yy}^* + T_{xy}.T_{yx}^*)$

(l) $Im(T_{xx}.T_{yy}^* - T_{xy}.T_{yx}^*)$

(m) $Re(T_{xx}.T_{yx}^* + T_{xy}.T_{yy}^*)$

(n) $Im(T_{xx}.T_{yx}^* - T_{xy}.T_{yy}^*)$

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

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

With:

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

5. muellertopoldegandpower (MuellerToPolarisationDegreeAndPowerFunctor)
   (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. $entropy = -\sum_{i=0}^{2} \frac{p[i].\log p[i]}{\log 3}$
2. $\alpha = \sum_{i=0}^{2} p[i].\alpha_i$
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
```

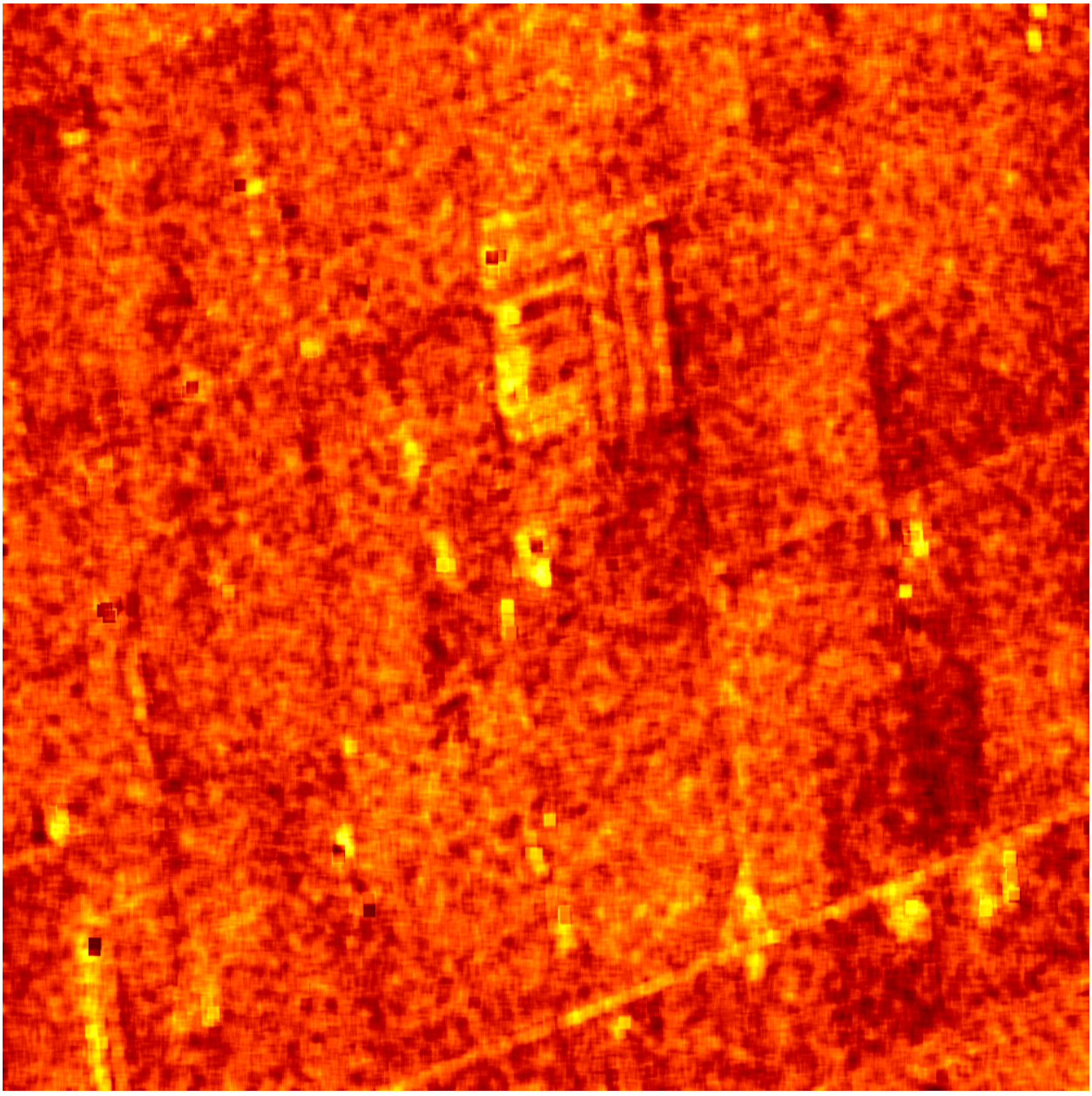
Each image is then colored thanks to a color look-up table 'hot'. Notice how minimum and maximum values are provided for each polarimetric variable.

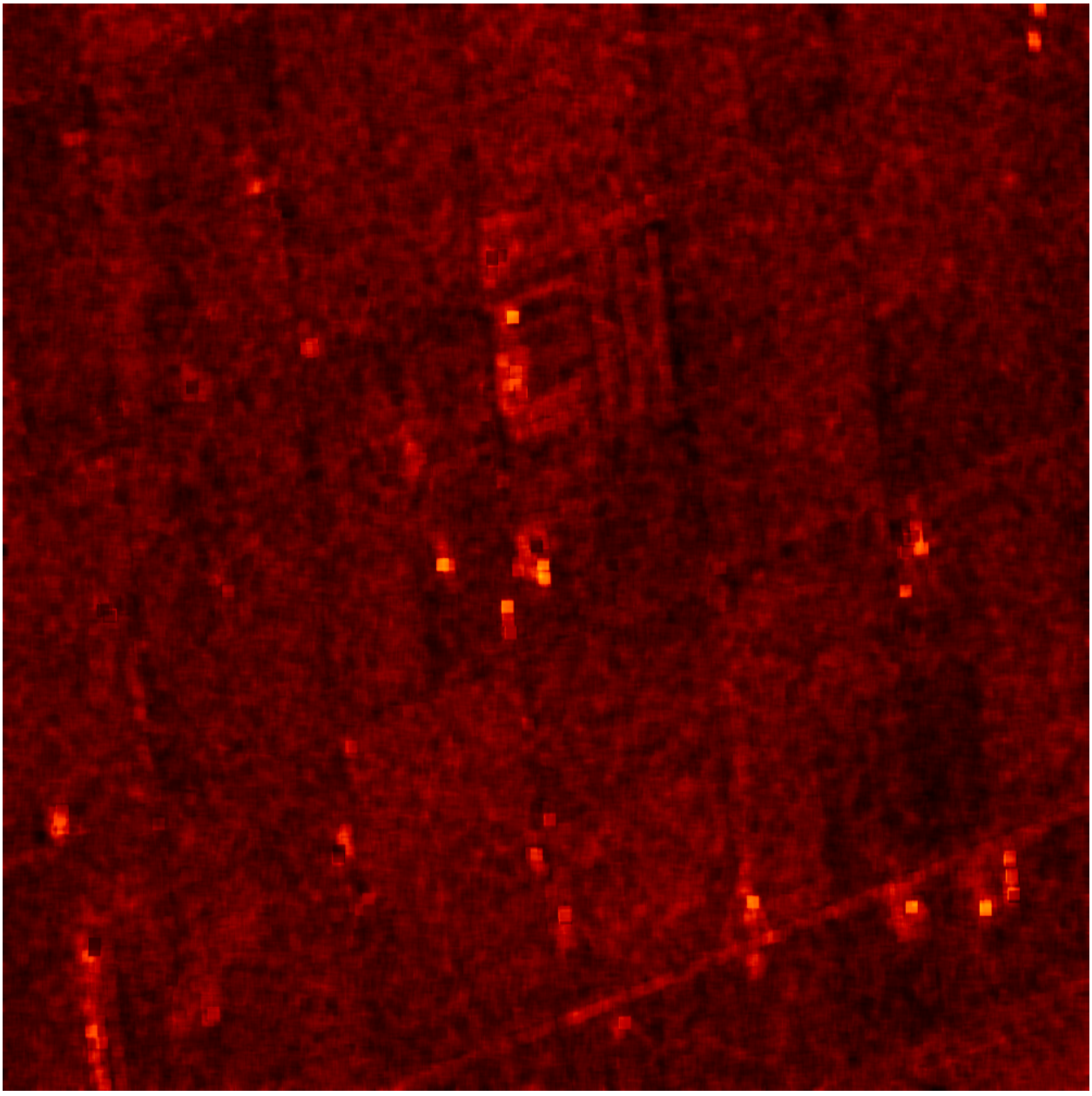
- ```
  otbcli_ColorMapping -in haa_extract_splitted_0.tif
                      -method continuous -method.continuous.lut hot
                      -method.continuous.min 0
                      -method.continuous.max 1
                      -out entropy_hot.tif uint8
  ```

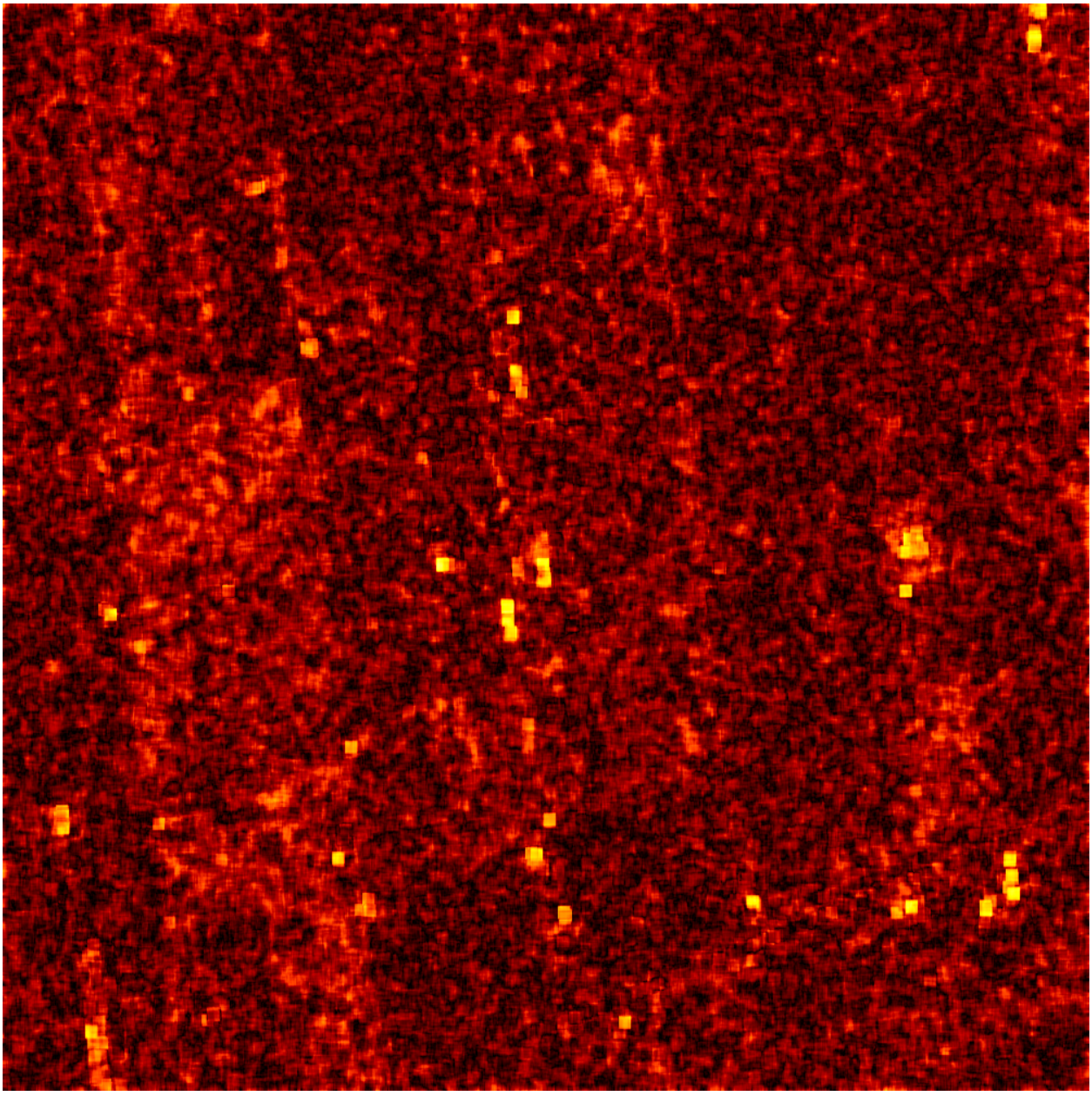
- ```
  otbcli_ColorMapping -in haa_extract_splitted_1.tif -method continuous
      -method.continuous.lut hot -method.continuous.min 0 -method.continuous.max
      90 -out alpha_hot.tif uint8
  ```

- ```
  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 (khi) parameters. These parameters are namely psii, khii, psir and khir. 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.

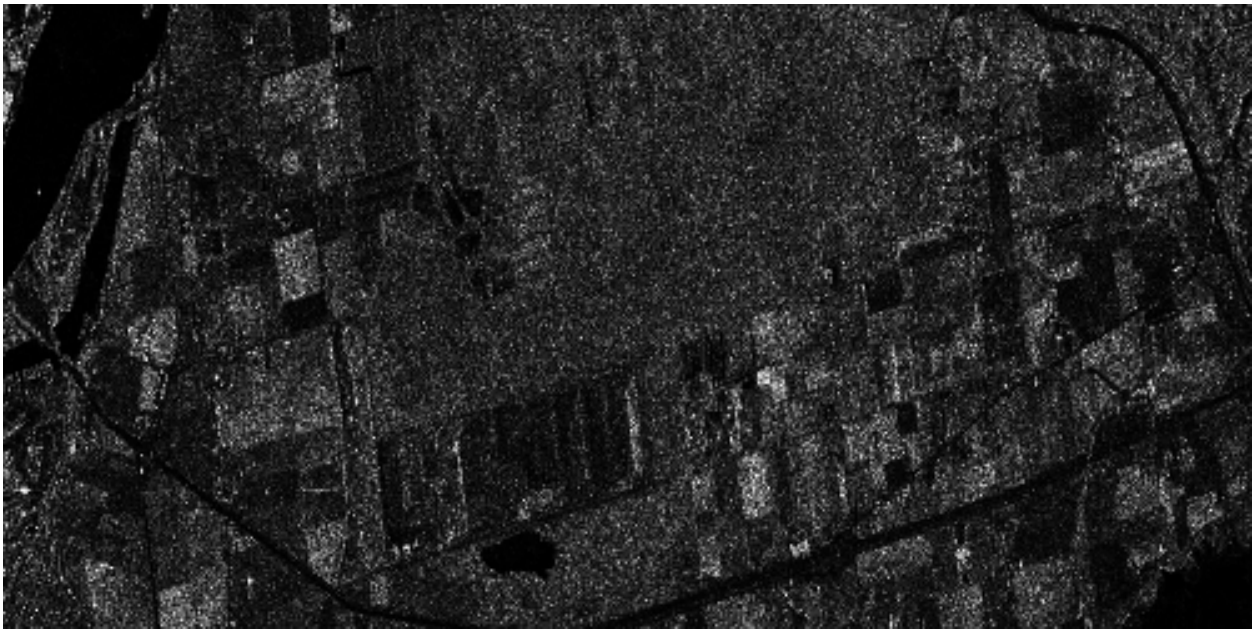
The final formula is thus: $P = \mid B^T.[S].A \mid^2$, where A ans B are two Jones vectors and S is a Sinclair matrix.

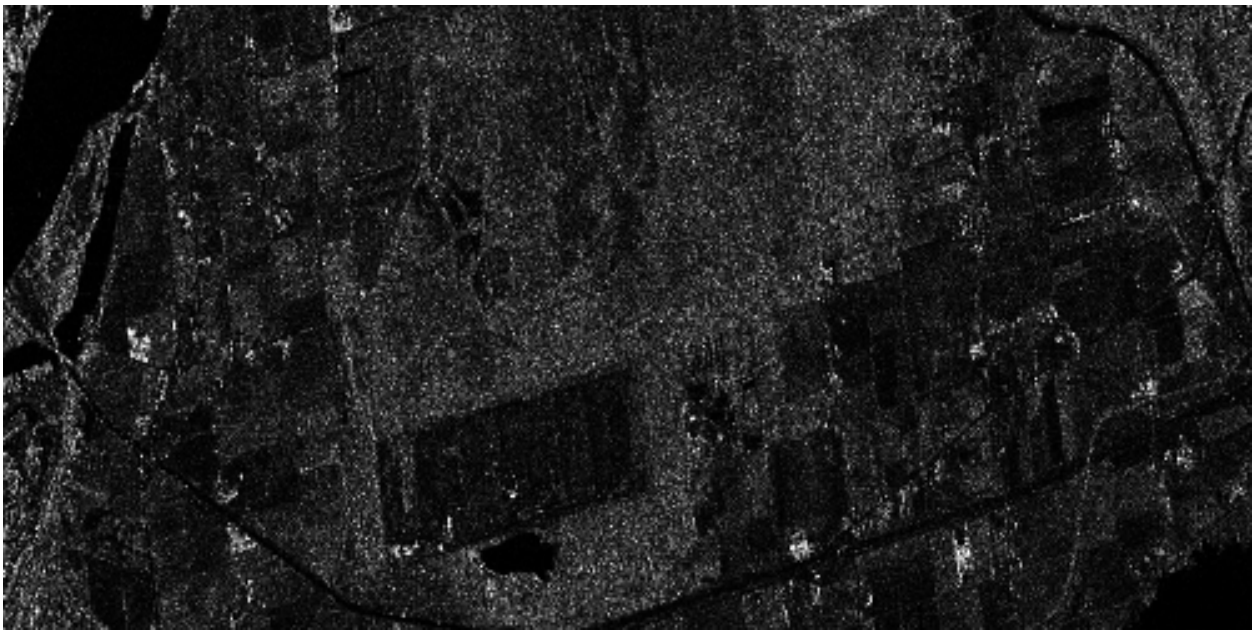
The two figures below ([fig:polsynthll] and [fig:polsynthlr]) show the two images obtained with the basis LL and LR (L for left circular polarization and R for right polarization), from a Radarsat-2 image taken over Vancouver, Canada. Once the four two-band images imagery_HH imagery_HV imagery_VH imagery_VV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

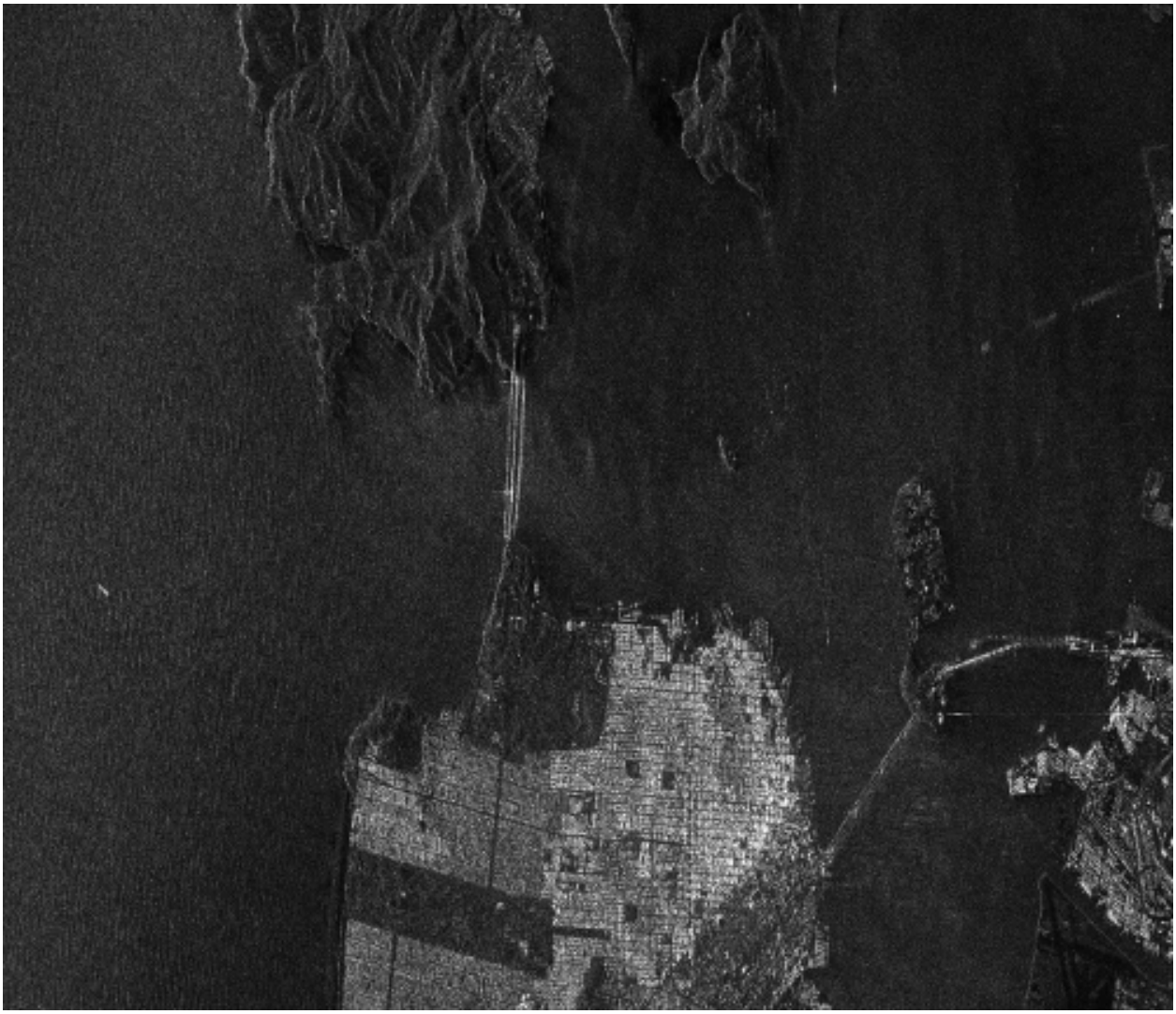
- ```
  otbcli_Rescale -in HV.tif -out HV_res.png uint8
  ```

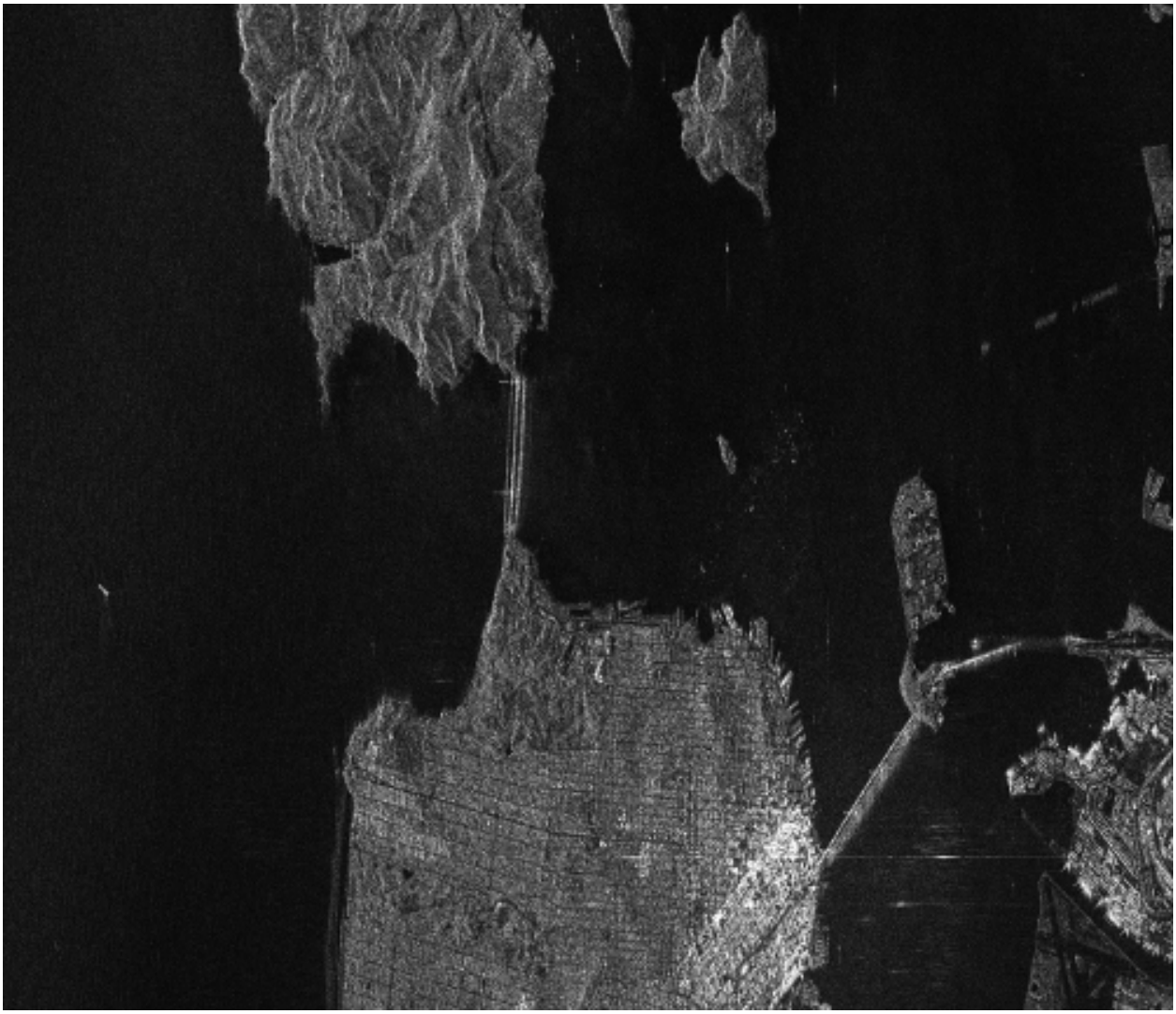
- ```
  otbcli_Rescale -in VV.tif -out VV_res.png uint8
  ```

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:

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





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]).

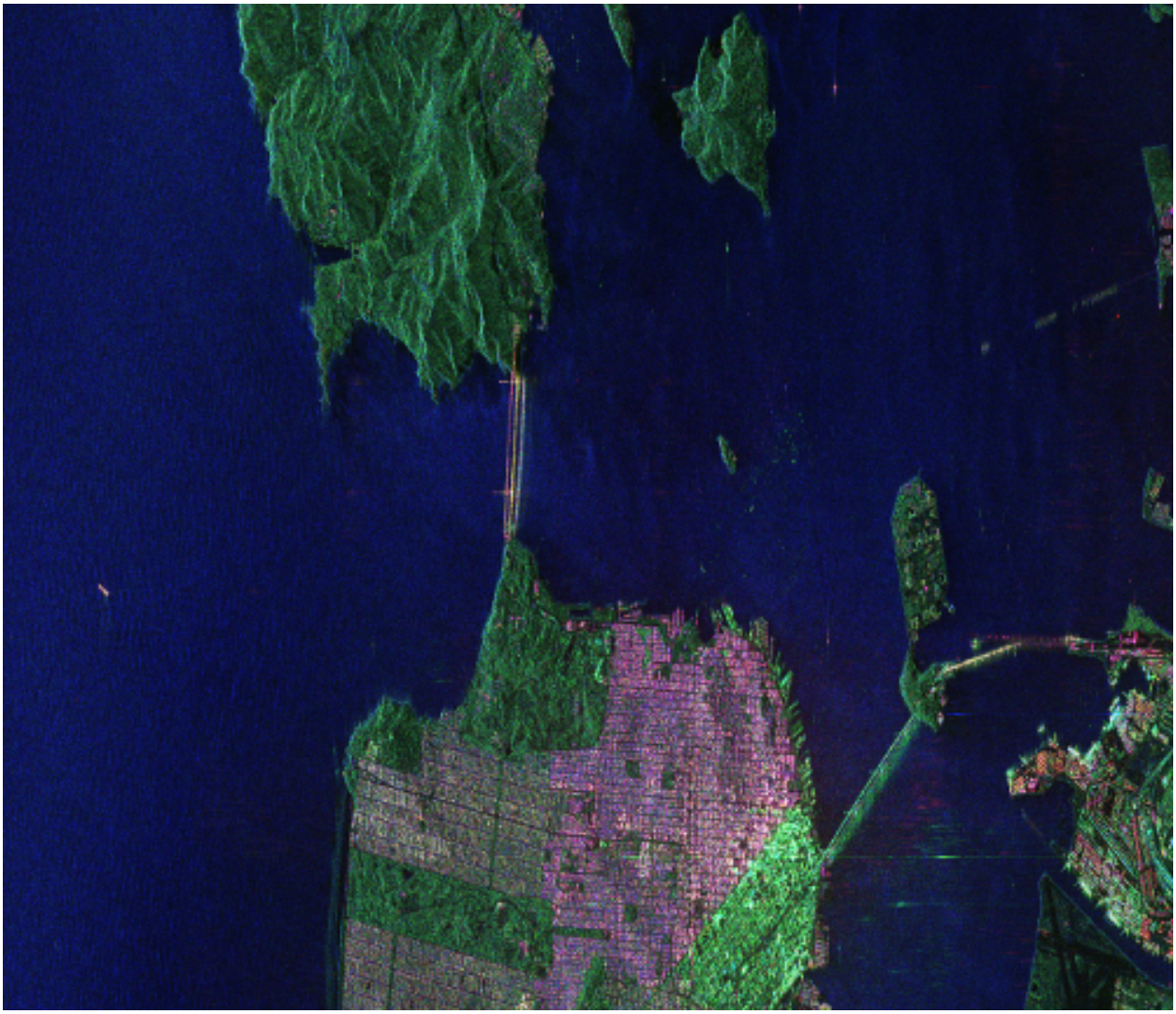
Residual registration

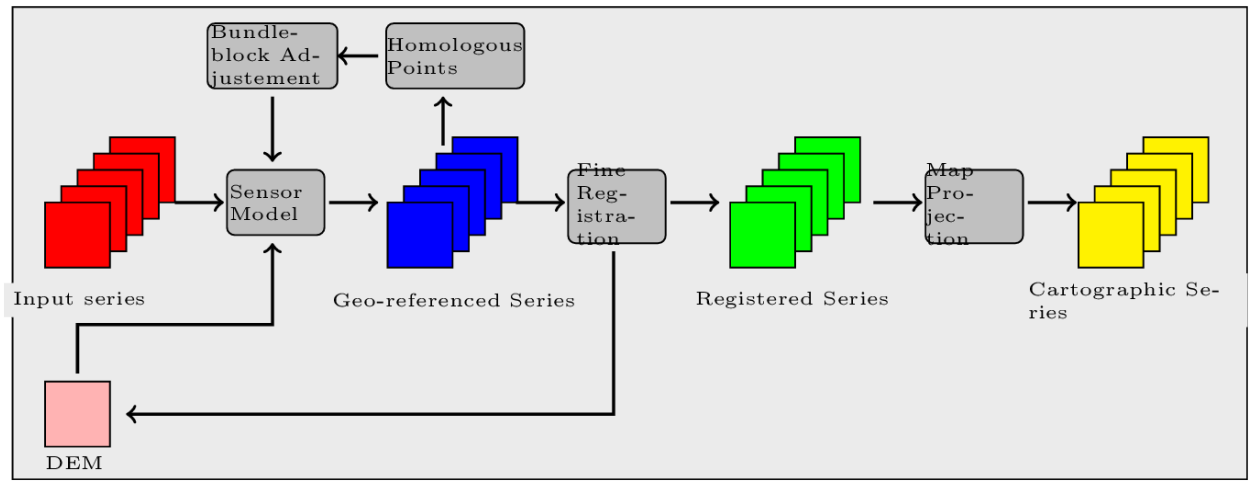
Image registration is a fundamental problem in image processing. The aim is to align two or more images of the same scene often taken at different times, from different viewpoints, or by different sensors. It is a basic step for orthorectification, image stitching, image fusion, change detection, and 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 orthorectify Pleiades image over the area of Toulouse, France.

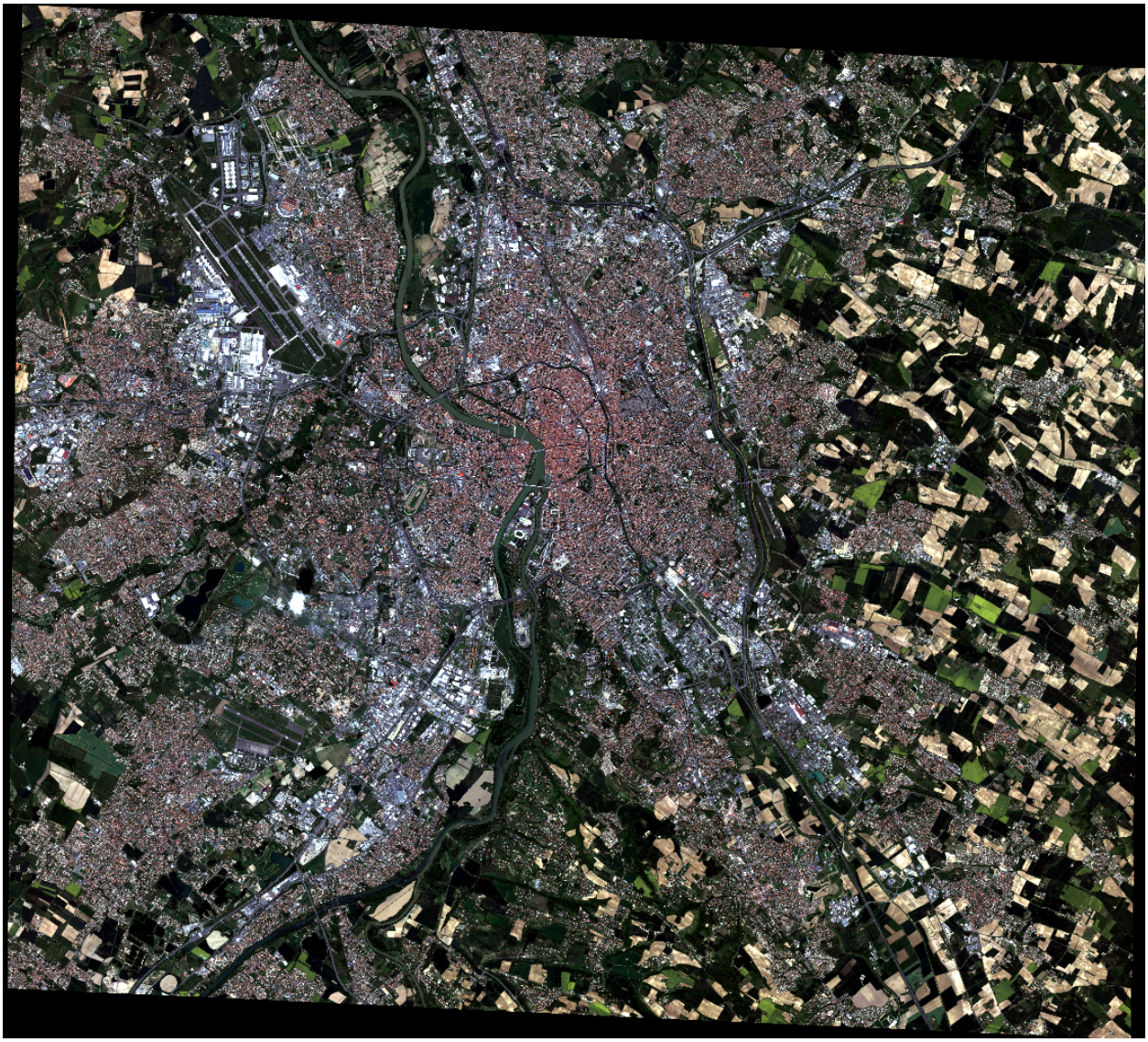


Figure 4.10: From left to right: Pleiades ortho-image, and original QuickBird image over Toulouse

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
```

Extract homologous points from images

The main idea of the residual registration is to estimate an 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).

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
                           -ingeom slave_image.geom
                           -outgeom refined_slave_image.geom
                           -inpoints homologous_points.txt
                           -outstat stats.txt
                           -outvector refined_slave_image.shp
```

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 hrefhttp://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

Image processing and information extraction

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:*m*ath 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.

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.

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.

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.tif
                          -fout filtered_range.tif
                          -foutpos filtered_spatial.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 *tilesizex* and *tilesizey* parameters, and by writing intermediate images to disk, thus keeping the memory consumption very low throughout the process. The segmentation will group together neighboring 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 contains 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.tif
                         -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.

All-in-one

The *LargeScaleMeanShift* application is a composite application that chains all the previous steps:

- Mean-Shift Smoothing
- Segmentation
- Small region merging
- Vectorization

Most of the settings from the previous applications are also exposed in this composite application. The range and spatial radius used for the segmentation step are half the values used for Mean-Shift smooting, which are obtained from LargeScaleMeanShift parameters. There are two output modes: vector (default) and raster. When the raster output is chosen, last step (vectorization) is skipped.

```
otbcli_LargeScaleMeanShift -in input_image.tif
                           -spatialr 5
                           -ranger 30
                           -minsize 10
                           -mode.vector.out segmentation_merged.shp
```

There is a cleanup option that can be disabled in order to check intermediate outputs of this composite application.

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 fiability 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 sudied 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
```

BandMathImageFilterX (based on muParserX)

This section describes how to use the BandMathImageFilterX.

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.

Syntax: first elements

The default prefix name for variables related to the ith 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
im1	a pixel from first input, made of n components/bands (first image is indexed by 1)	Vector
im1bj	jth component of a pixel from first input (first band is indexed by 1)	Scalar
im1bjNkxp	a neighbourhood (”N”) of pixels of the jth component from first input, of size kxp	Matrix
im1bjMini	global statistic: minimum of the jth band from first input	Scalar
im1bjMaxi	global statistic: maximum of the jth band from first input	Scalar
im1bjMean	global statistic: mean of the jth band from first input	Scalar
im1bjSum	global statistic: sum of the jth band from first input	Scalar
im1bjVar	global statistic: variance of the jth band from first input	Scalar
im1PhyX and im1PhyY	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

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

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
im1b1 + 1	correct
{im1b1} + {1}	correct
im1b1 + {1}	incorrect
{im1b1} + 1	incorrect
im1 + {im2b1,im2b2}	correct if im1 represents a pixel of two components (equivalent to im1 + im2)

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.

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 otbParserXPlugins.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:

$$im1\ div\ im2$$

$$im1\ dv\ 2.0$$

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

$$im1\ mult\ im2$$

$$im1\ mlt\ 2.0$$

Note that the operator '*' could have been used instead of 'pw' one. But 'pw' is a little bit more permisive, 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:

$$im1\ pow\ im2$$

$$im1\ pw\ 2.0$$

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:

$$(vmax(im3b1N3x5) + vmin(im3b1N3x5))\ div\ \{2.0\}$$

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]

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 puropose, 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:

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

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);
```

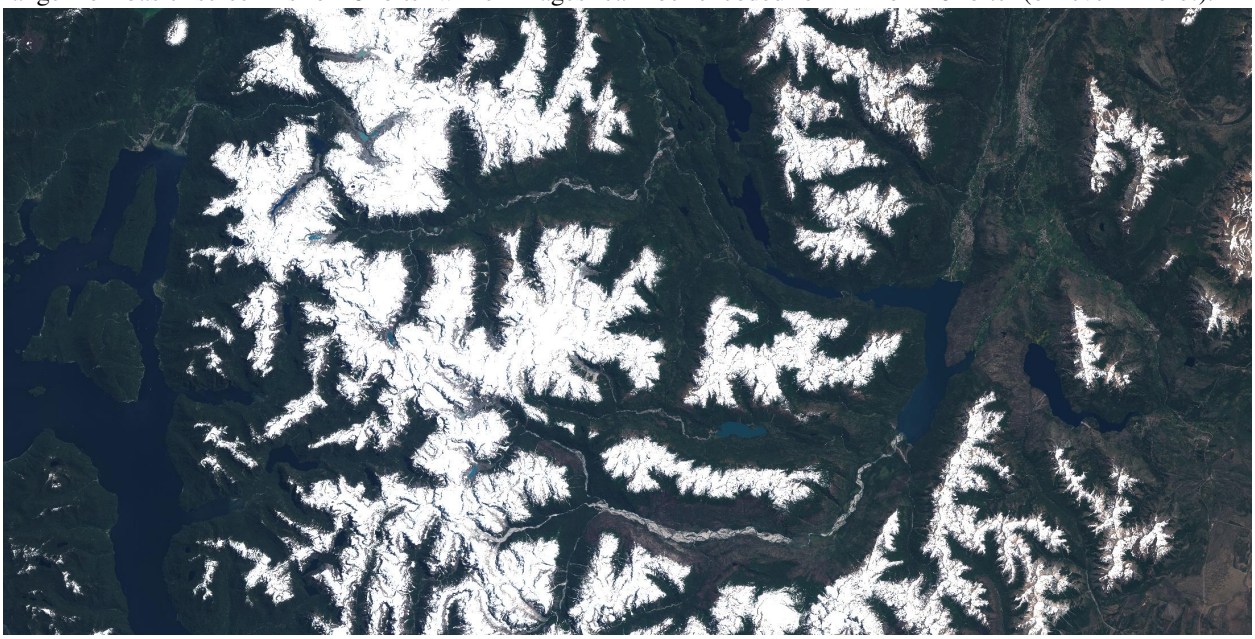
This function allows the user to export a txt file that saves its favorite constant or expression definitions. Such a file will be reusable by the ImportContext function (see above).

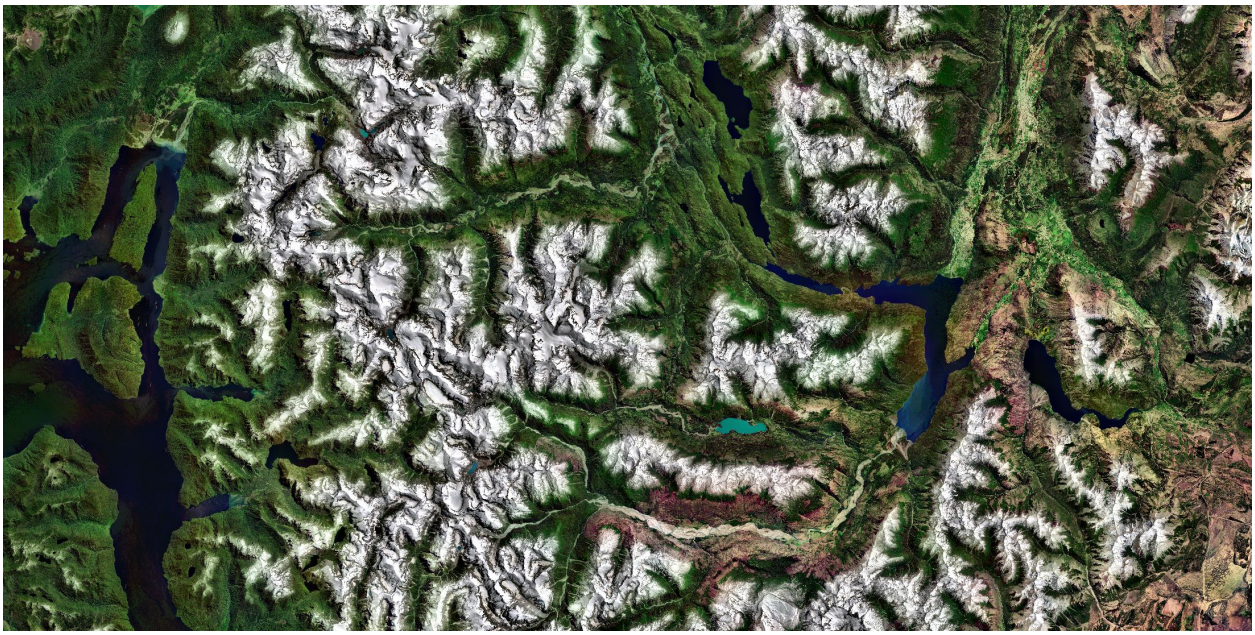
Please, also refer to the section dedicated to application.

Enhance local contrast

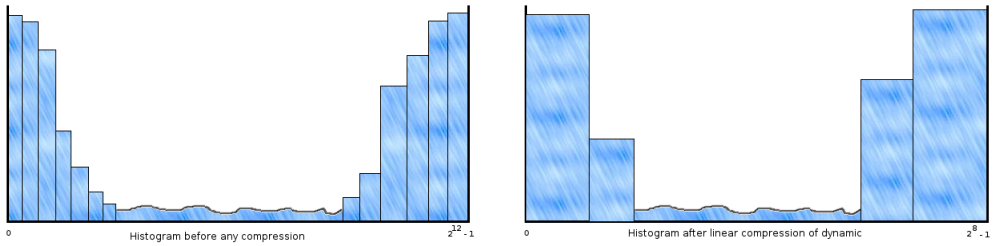
Principles

Sensor images have often a wide dynamic range. Whereas it is helpful to have high precision to do complex processing, it is pretty hard to display high dynamic images, even on modern screen as the dynamic range for basic screen is of 8 bits while images can be encoded on 12 or 16 bits (or even more!).





The *ContrastEnhancement* application aims to reduce the image dynamic by reorganizing it in a smarter way than just linear compression and improve the local contrast and enhance the definitions of edges.



The equalization of histogram creates a look up table in order to maximize the dynamic. The target histogram is perfectly flat. The gain applied on each pixel comes from the computation of the transfer function T such that :

$$\forall i \quad \int_{min}^{i*T(i)} h_{istogram}(j)dj = \int_{min}^{i} h_{target}(j)dj$$

where h_{target} is the corresponding flat histogram with the constraint that white and black are still white and black after equalization :

$$T(min) = T(max) = 1$$

You can apply this transformation with the *ContrastEnhancement* application:

```
otbcli_ContrastEnhancement -in input_image.tif
                           -out output_image.tif
                           -spatial global
```

It allows to compress the dynamic without losing details and contrast.

Advanced parameters

The *ContrastEnhancement* provides different options to configure the contrast enhancement method. Let us see what there are for.

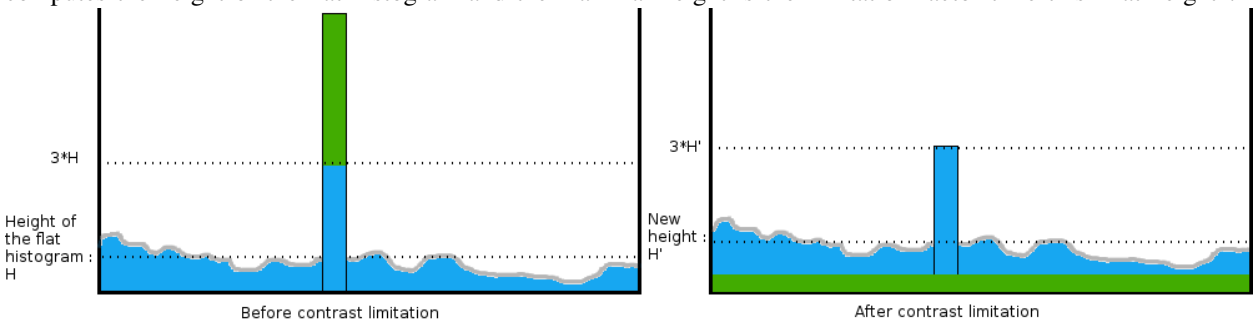
First what you want to equalize. Two modes are available:

- **luminance:** on 3 bands image, the equalization will be done on a single band which will be a composition of the original bands. The computed gain will then be applied on the different bands. The classical use of this method is to conserve ratio between the different color, conserve the hue.
- **channel:** each bands are equalized independently.

The other option is the local equalization. You can choose a window size that will be use to split the image in tiles and histograms will be computed over those tiles. Gain will be interpolated between the adjacent tiles in order to give a smooth result.

```
otbcli_ContrastEnhancement -in input_image.tif
                           -out output_image.tif spatial.local.h 500
                           -spatial.local.w 500
                           -mode lum
```

The *ContrastEnhancement* application also offers a way to limit contrast by adjusting original histogram with the **hfact** parameter. The limitation factor represents the limit height that can have any bucket of the histogram; the application computes the height of the flat histogram and the maximal height is the limitation factor time this "flat height".



Finally, you can ignore a particular value with the **nodata** parameter, and also set manually your minimum and maximum value. Any value out of bound will be ignored.

Classification

Feature classification and training

The Orfeo ToolBox provided applications to train a supervised or unsupervised classifier from different set of *features* and to use the generated classifier for vector data classification. Those *features* can be information extracted from images (see feature extraction section) or it can be different types of *features* such as the perimeter, width, or area of a surface present in a vector data file in an ogr compatible format.

Train a classifier with features

The *TrainVectorClassifier* application provide a way to train a classifier with an input set of labeled geometries and a list of *features* to consider for classification.

```
otbcli_TrainVectorClassifier -io.vd samples.sqlite
                             -cfield CODE
                             -io.out model.rf
                             -classifier rf
                             -feat perimeter area width
```

The `-classifier` parameter allows to choose which machine learning model algorithm to train. You have the possibility to do the unsupervised classification,for it, you must to choose the Shark kmeans classifier. Please refer to the `TrainVectorClassifier` application reference documentation.

In case of multiple sample files, you can add them to the `-io.vd` parameter.

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 for the specification of different sample files for this purpose, for a more fair estimation of the performances. Note that this scheme to estimate the performance can also be carried out afterwards (see *Validating the classification model* section).

Feature classification

Once the classifier has been trained, one can apply the model to classify a set of features on a new vector data file using the *VectorClassifier* application:

```
otbcli_VectorClassifier -in       vectorData.shp
                        -model    model.rf
                        -feat     perimeter area width
                        -cfield   predicted
                        -out      classifiedData.shp
```

This application outputs a vector data file storing sample values and classification labels. The output vector file is optional. If no output is given to the application, the input vector data classification label field is updated. If a statistics file was used to normalize the features during training, it shall also be used here, during classification.

Note that with this application, the machine learning model may come from a training on image or vector data, it doesn't matter. The only requirement is that the chosen features to use should be the same as the one used during training.

Validating classification

The performance of the model generated by the *TrainVectorClassifier* or *TrainImagesClassifier* applications 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 for supervised algorithms. For unsupervised algorithms a contingency table is generated. These results are output as an *.CSV file.

Pixel based classification

Orfeo ToolBox ships with a set of application to perform supervised or unsupervised 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" />
    ...
```

Sample selection

Now, we know exactly how many samples are available in the image for each class and each geometry in the training set. From these statistics, we can now compute the sampling rates to apply for each class, 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 select is divided proportionally among each class (class 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 sampling techniques:

- **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 may not be sampled.

The application will accept as input the input image and training geometries, as well class statistics XML file computed during the 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 sample 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
```

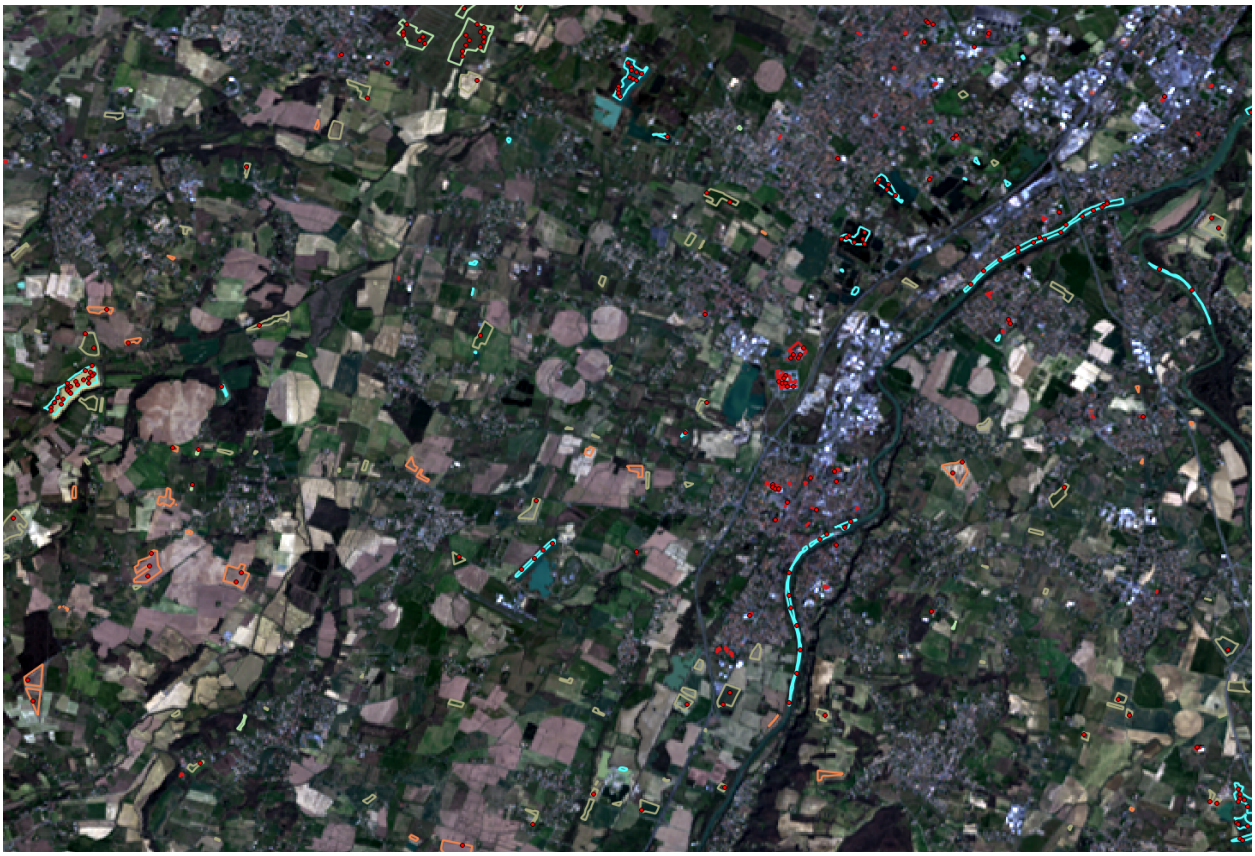


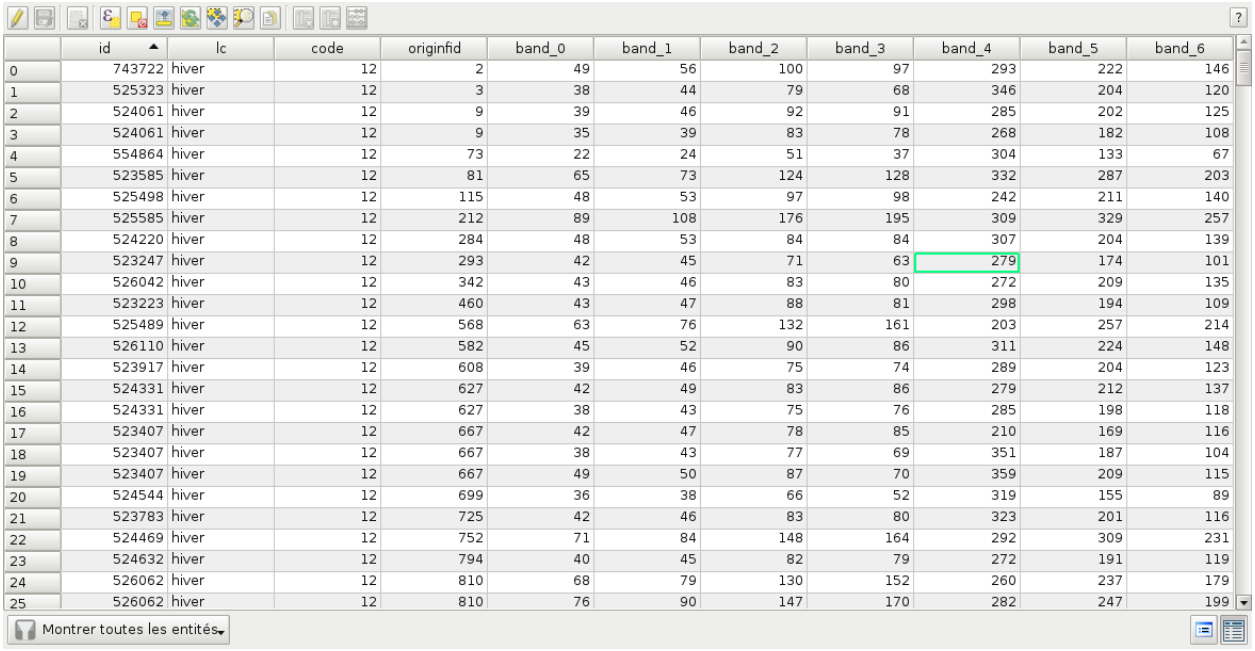
Fig. 6.1: This image shows the polygons of the training with a color corresponding to their class. The red dot shows the samples that have been selected.

Samples extraction

Now that the locations of the samples are selected, we will attach measurements 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

Fig. 6.2: 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 span the entire extents of the images.

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, $Ni(c) = M_i(c)$ 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, $N_i(c) = p * \frac{sum_k(Tk(c))}{L}$ where p is the user-defined percentage
  - *Mode = custom:* For each image i and each class c, $Ni(c) = p(i) * T_i(c)$ where $p(i)$ is the user-defined percentage for image i
- **Strategy = total**
  - *Mode = proportional:* For each image i and each class c, $N_i(c) = total * (\frac{sum_k(Ti(k))}{sum_k l(Tl(k))}) * (\frac{Ti(c)}{sum_k(Ti(k))})$ where $total$ is the total number of samples specified
  - *Mode = equal:* For each image i and each class c, $N_i(c) = (total/L) * (\frac{Ti(c)}{sum_k(Ti(k))})$ where $total$ is the total number of samples specified
  - *Mode = custom:* For each image i and each class c, $Ni(c) = total(i) * (\frac{Ti(c)}{sum_k(Ti(k))})$ 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 b y 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_ComputeImagesStatistics -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
```

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

For more information about the training process for features please refer to the *Train a classifier with features* 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 Orfeo ToolBox training applications provides information about the performance of the generated model (see *Validating classification*).

With the *ConputeConfusionMatrix* 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.

If you have made an unsupervised classification, it must be specified to the `ConputeConfusionMatrix` application. In this case, a contingency table have to be create rather than a confusion matrix. For further details, see `format` parameter in the application help of *ConputeConfusionMatrix*.

```
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 .

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.

Unsupervised learning

Using the same machine learning framework, it is also possible to perform unsupervised classification. In this case, the main difference is that the training samples don't need a real class label. However, in order to use the same *TrainImagesClassifier* application, you still need to provide a vector data file with a label field. This vector file will be used to extract samples for the training. Each label value is can be considered as a source area for samples, the same logic as in supervised learning is applied for the computation of extracted samples per area. Hence, for unsupervised classification, the samples are selected based on classes that are not actually used during the training. For the moment, only the KMeans algorithm is proposed in this framework.

```
otbcli_TrainImageClassifier
  -io.il                image.tif
  -io.vd                training_areas.shp
  -io.out               model.txt
  -sample.vfn           Class
  -classifier           sharkkm
  -classifier.sharkkm.k 4
```

If your training samples are in a vector data file, you can use the application *TrainVectorClassifier*. In this case, you don't need a fake label field. You just need to specify which fields shall be used to do the training.

```
otbcli_TrainVectorClassifier
  -io.vd                training_samples.shp
  -io.out               model.txt
  -feat                 perimeter area width red nir
  -classifier           sharkkm
  -classifier.sharkkm.k 4
```

Once you have the model file, the actual classification step is the same as the supervised case. The model will predict labels on your input data.

```
otbcli_ImageClassifier
  -in input_image.tif
  -model model.txt
  -out kmeans_labels.tif
```

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 *Figure2*) generated from 6 different SVM models. The *Figure3* 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.

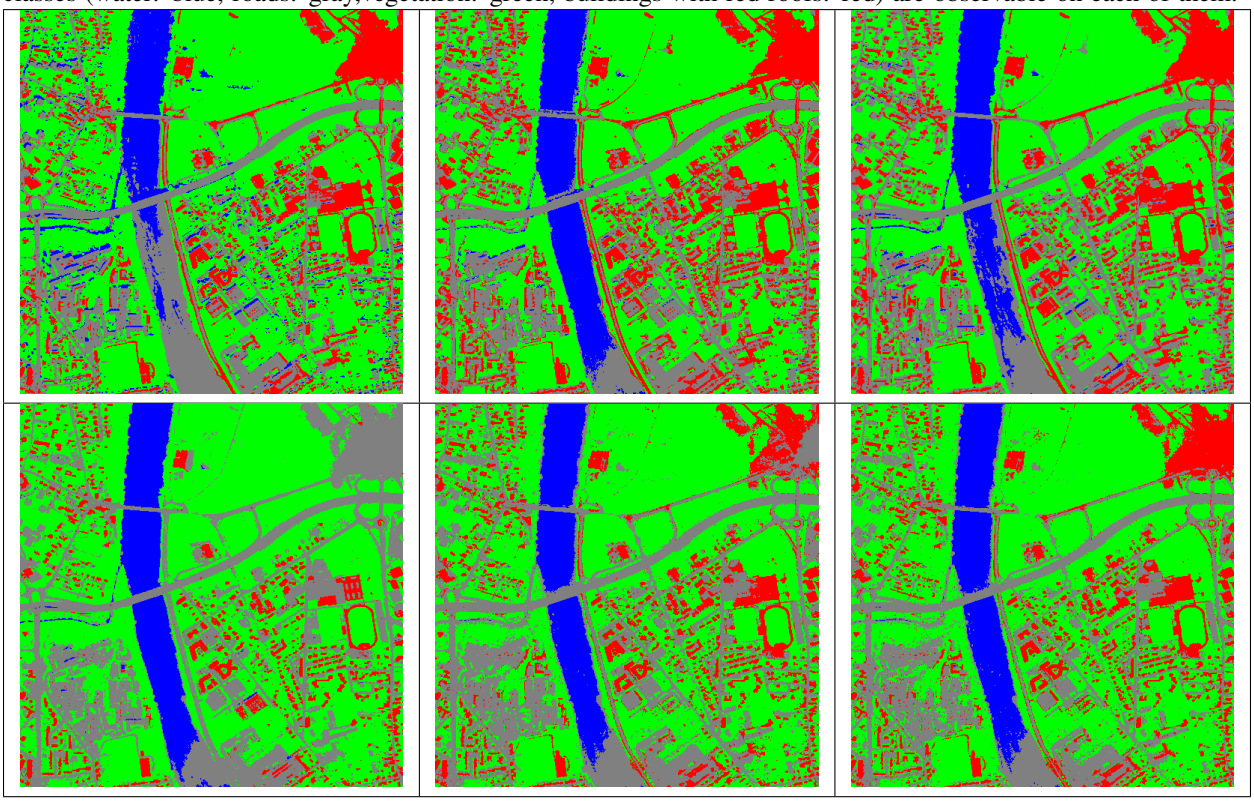


Figure 3: Six fancy colored classified images to be fused, generated from 6 different SVM models.

As an example of the *FusionOfClassifications* application by *majority voting*, the fusion of the six input classification maps represented in *Figure3* leads to the classification map illustrated on the right in *Figure4*. 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.

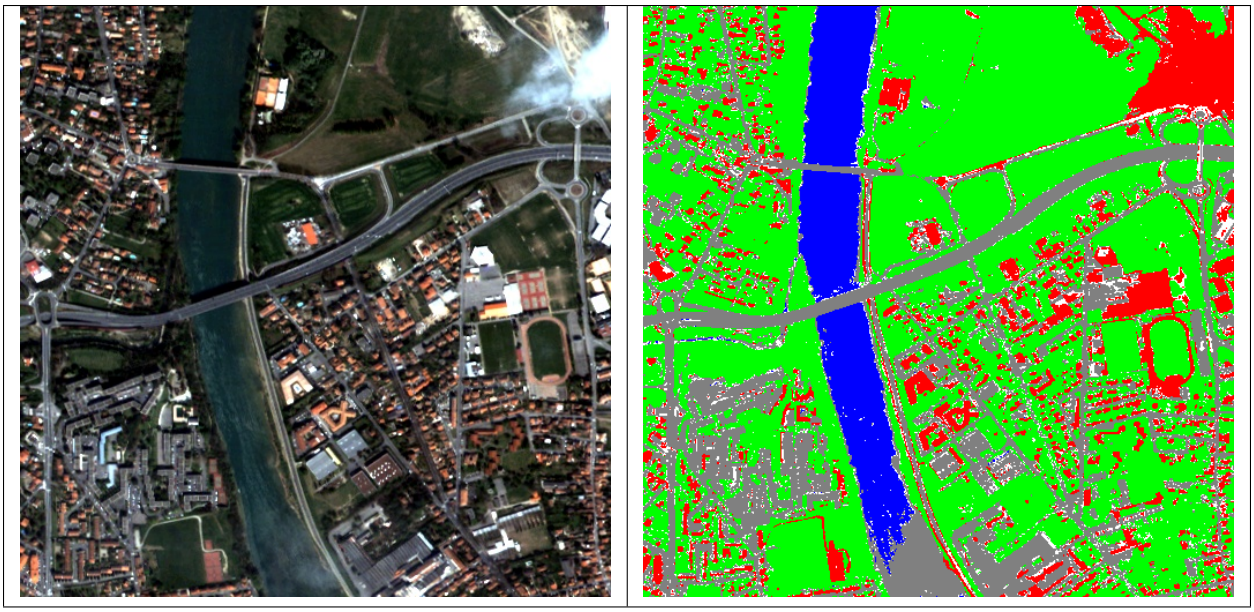


Figure 4: From left to right: Original image, and fancy colored classified image obtained by a majority voting fusion of the 6 classification maps represented in Fig. 4.13 (water: blue, roads: gray, vegetation: green, buildings with red roofs: red, undecided: white)

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 *Figure3* leads to the classification map illustrated on the right in *Figure5*. Thus, it appears that

this fusion gives access to a more precise and robust classification map based on the confidence level in each classifier.

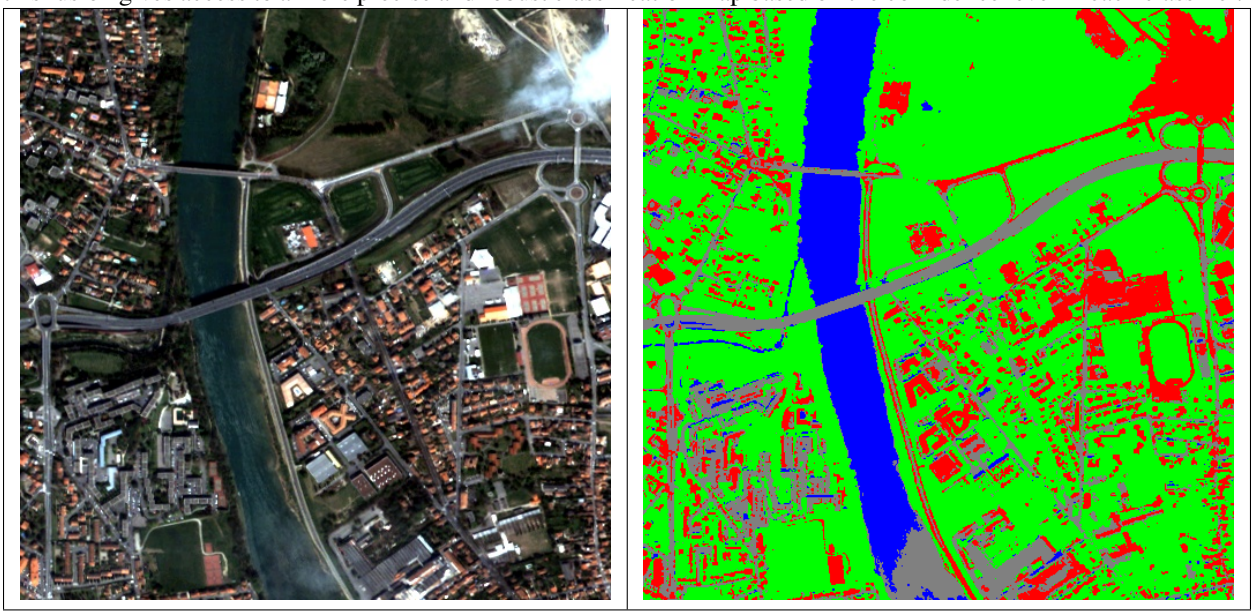


Figure 5: From left to right: Original image, and fancy colored classified image obtained by a Dempster-Shafer fusion of the 6 classification maps represented in *Figure3* (water: blue, roads: gray, vegetation: green, buildings with red roofs: red, undecided: white).

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`.

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 *Figure2*, the *Figure6* 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.

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 .

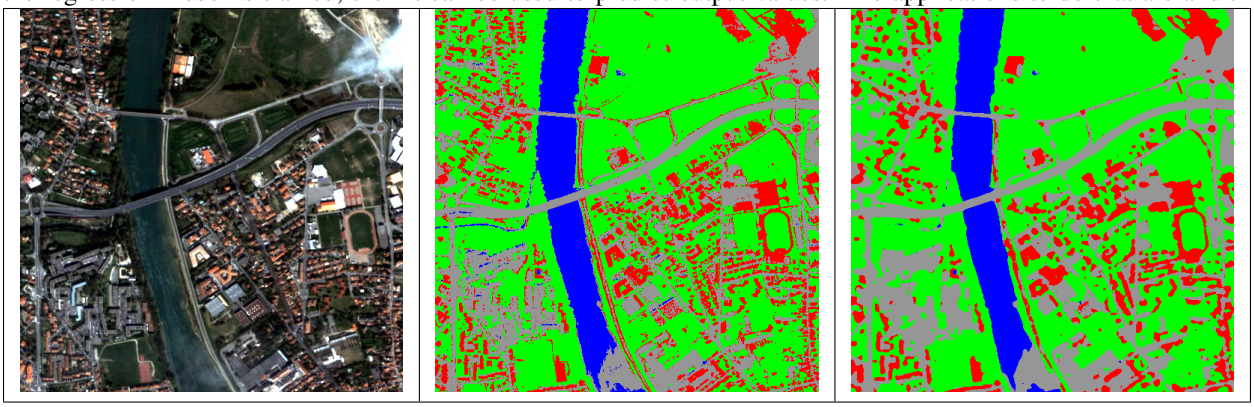


Figure 6: From left to right: Original image, fancy colored classified image and regularized classification map with radius equal to 3 pixels.

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 . 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 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
```

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 .

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
```

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
```

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 as follows, which would produce an output image containing 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 as follows, which would produce a single band output image with the Water:NDWI2 radiometric index:

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

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
```

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. :(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).

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

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

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

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

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

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

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

"Haralick's Correlation" $= f_8 = \frac{\sum_{i,j}(i,j)g(i,j)-\mu_t^2}{\sigma_t^2}$ where μ_t and σ_t are the mean and standard deviation of the row (or column, due to symmetry) sums. Above, μ = (weighted pixel average) $= \sum_{i,j} i \cdot g(i, j) = \sum_{i,j} j \cdot g(i, j)$ (due to matrix symmetry), and σ = (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).

"Mean" $= \sum_{i,j} ig(i,j)$

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

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

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

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

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

"Difference variance" $= f_10 = variance of g_{x-y}(i)$

"Difference entropy" $= f_11 = -\sum_i g_{x-y}(i) log(g_{x-y}(i))$

"Information Measures of Correlation IC1" $= f_12 = \frac{f_9 - HXY1}{H}$

"Information Measures of Correlation IC2" $= f_13 = \sqrt{1 - \exp -2|HXY2 - f_9|}$

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

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

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

`--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):

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

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

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

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

"Run Percentage" $= RP = \frac{n_r}{n_p}$

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

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

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

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

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

"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              InputImage
                                  -channel         1
                                  -texture         simple
                                  -parameters.min  0
```

```
                                -parameters.max 255
                                -out            OutputImage
```

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 adjustement 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.spathre` 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
```

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!

Estimate epipolar geometry transformation

The aim of this step is to generate resampled grids to transform images into 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 row in the right image.

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

The forward function allows to go from the image referential to the geographic referential. For the first image, this function will be noted:

$$(long, lat) = f_1(i, j, h)$$

where h is the elevation hypothesis, (i, j) are the pixel coordinates in image 1 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:

$$(i, j) = f_2^{Inv}(long, lat, h)$$

Using jointly the forward and inverse functions from the image pair, we can construct a co-localization function $g_{1\to2}$ between the position of a pixel in the first and its position in the second one:

$$(i_2, j_2) = g_{1\to2}(i_1, j_1, h)$$

The expression of this function is:

$$g_{1\to2}(i_1, j_1, h) = f_2^{Inv}[f_1(i_1, j_1, h)]$$

The expression is not really important, what you need to understand is that if we are able to determine for a given pixel in image 1 the corresponding pixel in image 2, 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.default 50
                                        -epi.step 10
                                        -io.outleft grid_image1.tif
                                        -io.outright grid_image2.tif
```

The application estimates the displacement to apply to each pixel in both input images to obtain epipolar geometry. The application accepts 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 per meters) of the baseline to sensor altitude ratio (also called B/H in the litterature). It can be used to do an approximate conversion of disparities to physical elevation :

$$h = h_{REF} + \frac{d}{B/H}$$

where h_{REF} is the reference altitude used to generate the epipolar grids (here: 50m), and d is a disparity value (in pixels) for a given object between images 1 and 2.

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

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 to resample both images. This application can obviously be used in a lot of other contexts.

The two commands to generate epipolar images are:

```
otbcli_GridBasedImageResampling -io.in image1.tif
                                -io.out epi_image1.tif
                                -grid.in grid_image1.tif
                                -out.sizex 4462
                                -out.sizey 2951
```

```
otbcli_GridBasedImageResampling -io.in image2.tif
                                -io.out epi_image2.tif
                                -grid.in grid_image2.tif
                                -out.sizex 4462
                                -out.sizey 2951
```

As you can see, we set *sizex* and *sizey* parameters using output values given by the *StereoRectificationGridGenerator* application to set the size of the output epipolar images. The two epipolar images should have the same size.

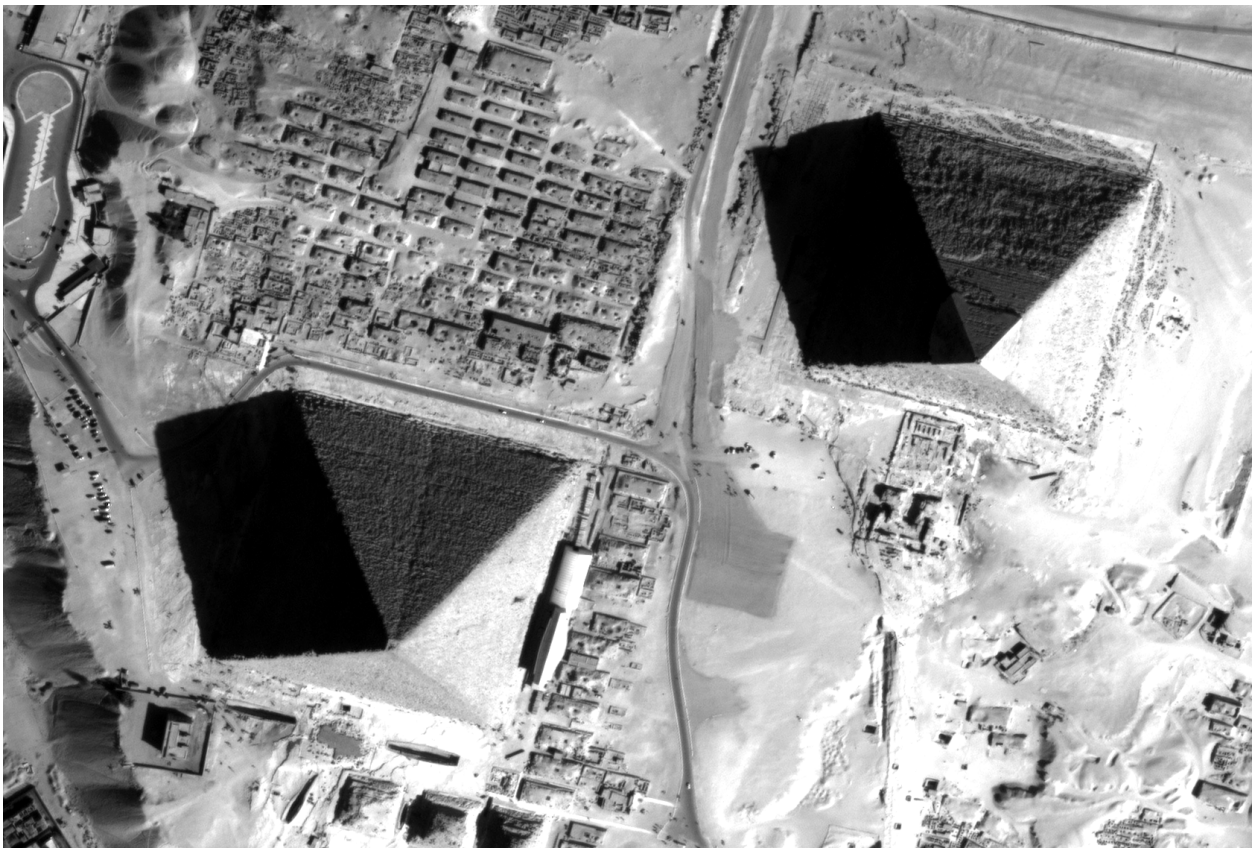
Figure 1: Extract of resample image1 and image2 in epipolar geometry over Pyramids of Cheops. ©CNES 2012

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.

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! The **Orfeo ToolBox** implements different local strategies for block matching:

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

An other important parameter (mandatory in the application!) is the range of disparities. In theory, the block matching can perform a blind exploration and search for a 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 choose one point on a *flat* area. Its coordinates are $[1525, 1970]$ in epipolar image 1 and $[1526, 1970]$ in epipolar image 2. 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 coordinates of this pixel are $[1661, 1299]$ in image 1 and $[1633, 1300]$ in image 2. We check the difference between column coordinates in images 1 and 2 in order to derive the useful disparity interval for horizontal exploration. In our case, this interval is at least $[-28, 1]$ (the convention for the sign of the disparity range is from image 1 to image 2).

Note that this exploration interval can be reduced using an external DEM in the *StereoRectificationGridGenerator* application. Indeed, the disparities measured between the epipolar images are relative to the reference altitude used when computing epipolar grids (hence, defining the epipolar geometry). Using an external DEM should produce epipolar images where altitude deviations from the reference are smaller, and as a consequence, disparities closer to 0.

Regarding the vertical disparity, in the first step we said that we reduced the problem of 2D exploration to a 1D 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 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 for minimum and maximum disparities (always from image1 to image2).

The command line for the *BlockMatching* application is:

```
otbcli_BlockMatching -io.inleft epi_image1.tif
                     -io.inright epi_image2.tif
                     -io.out disparity_map_ncc.tif
                     -bm.minhd -45
                     -bm.maxhd 5
                     -bm.minvd -1
                     -bm.maxvd 1
                     -mask.inleft epi_mask_image1.tif
                     -mask.inright epi_mask_image2.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 epi_image1.tif
                -out epi_mask_image1.tif
                -exp "im1b1<=0 ? 0 : 255"
```

```
otbcli_BandMath -il epi_image2.tif
                -out epi_mask_image2.tif
                -exp "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** and **bm.medianfilter.incoherence**: 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 (cases where the absolute difference is greater than the 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

From disparity to Digital Surface Model

Using the previous application, we evaluated disparities between epipolar 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 are computed from the triangulation of the "left-right" lines of sight for each matched pixels. When several elevations are available on a DSM cell, the highest one is kept.

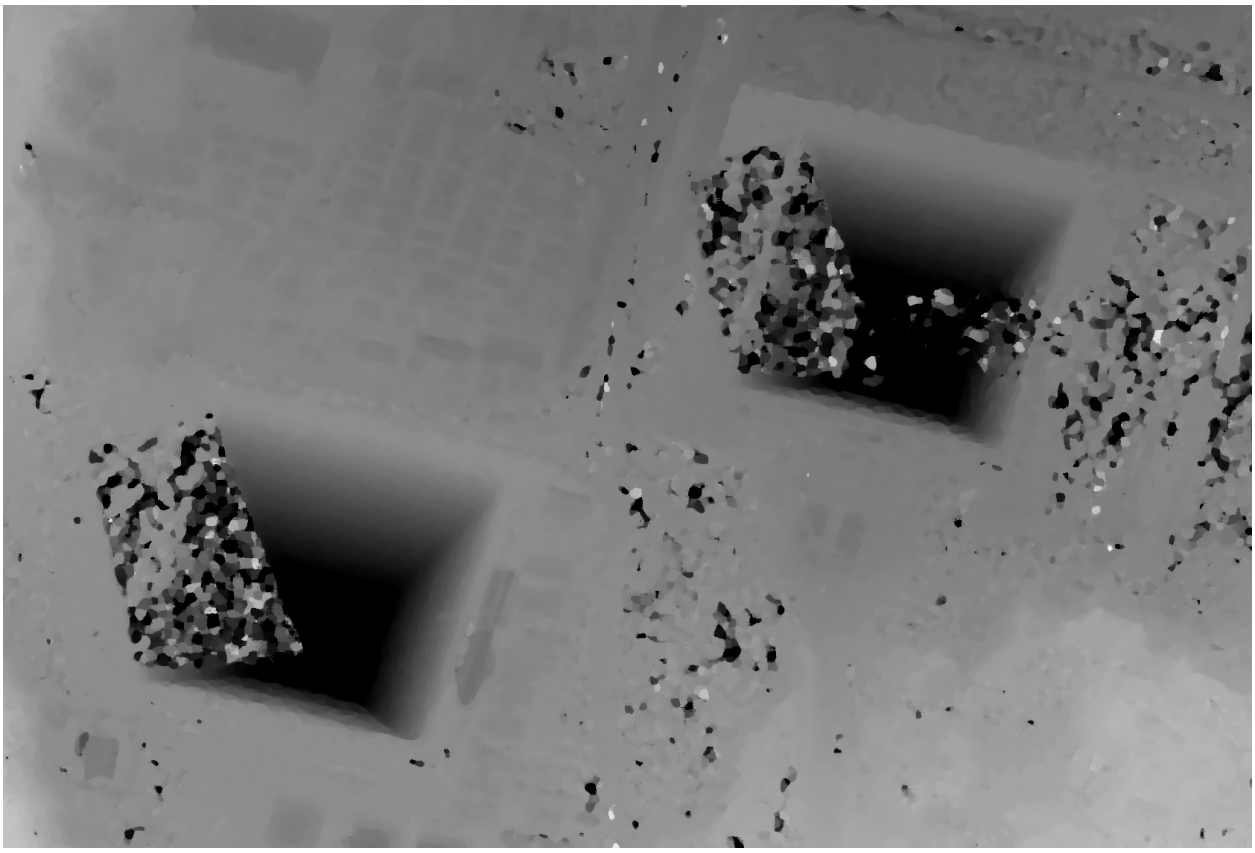
First, an important point is that it is often a good idea to rework the disparity map given by the *BlockMatching* application to only keep relevant disparities. For this purpose, we can use the output optimal metric image and filter disparities with respect to this value.

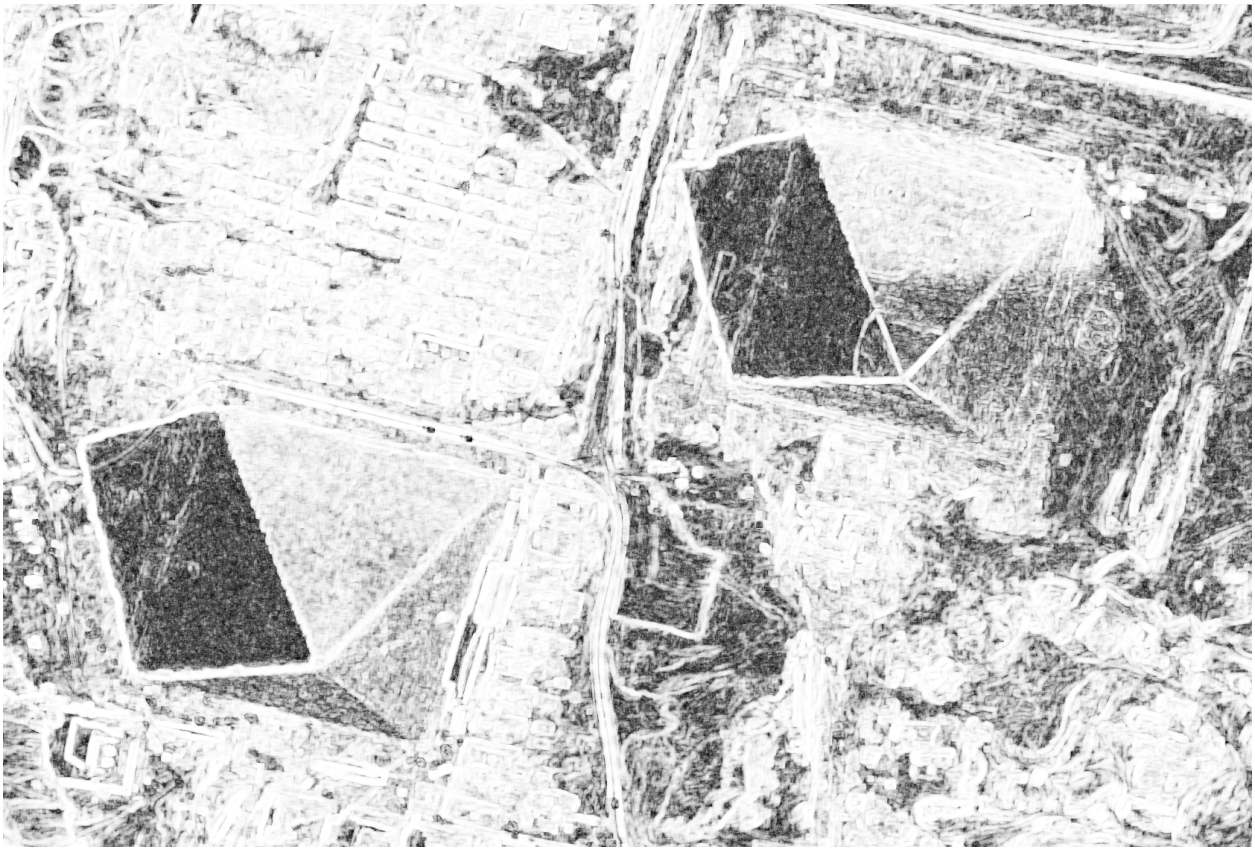
For example, if we used Normalized Cross-Correlation (NCC), we can keep only disparities where optimal metric value is superior to 0.9. Disparities below this value can be considered as inaccurate and will not be used to compute elevation information (the *-io.mask* parameter can be used for this purpose).

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_disparity.tif uint8
                -exp "im1b3>0.9 ? 255 : 0"
```





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 grid_image1.tif
                                  -io.rgrid grid_image2.tif
                                  -io.mask thres_disparity.tif
                                  -io.out elevation_map.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.

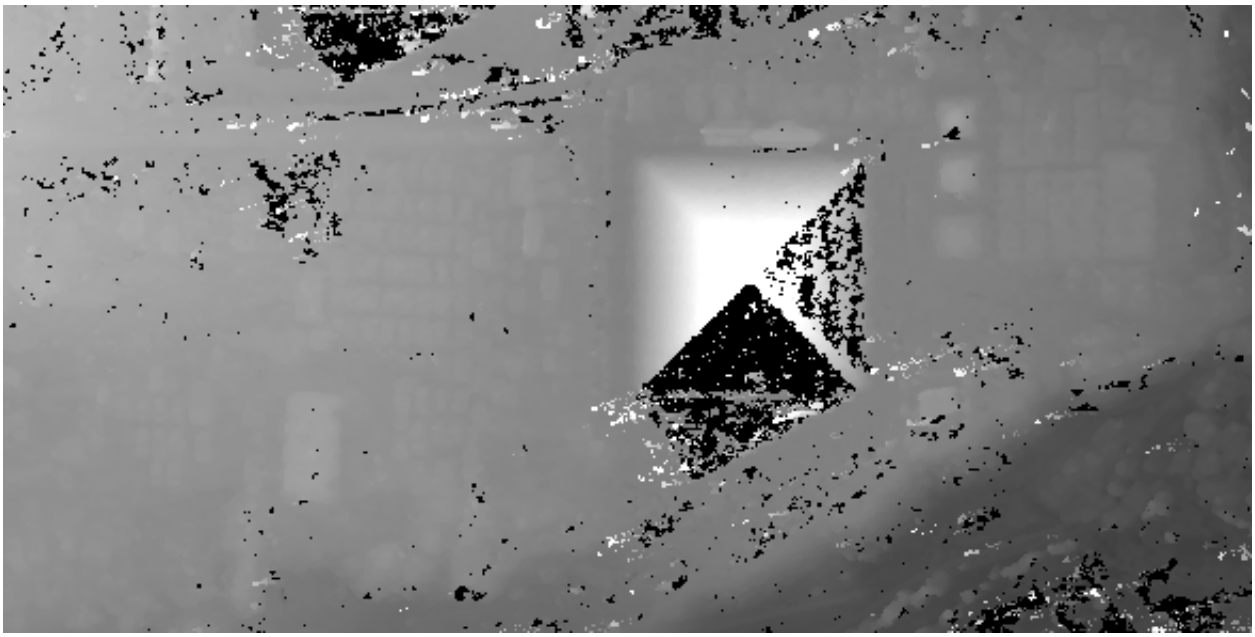


Figure 3: Extract of the elevation map over Pyramids of Cheops.

The *Figure 3* shows the output DEM from the Cheops pair.

One application to rule them all in multi stereo framework scheme

An application has been created to fuse one or multiple stereo reconstruction(s) using all-in-one approach: *StereoFramework* . It computes the DSM from one or several stereo pairs. First of all the user has to choose his input data and defines stereo couples using *-input.co* string parameter. Each couple is defined by 2 image indexes "a b" (starting at 0) separated by a space character. The different pairs are concatenated with coma. For instance "0 1,0 2" will define the image pairs "first with second", and "first with third". 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 a float value (-32768 by default)
  - Choice of fusion strategy in each DSM cell (max, min, mean, acc)
  - Output DSM
  - Extent of output DSM
- **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)

Stereo reconstruction good practices

The parameters for altitude offset 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.

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.

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 all 3D maps are fused to produce a DSM with desired geographic or cartographic projection and parametrizable extent.

OTB processing in Python

Basics

In the `otbApplication` 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`.

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  "Smoothing"
app = otbApplication.Registry.CreateApplication("Smoothing")

# We print the keys of all its parameters
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')

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

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

# This will execute the application and save the output to argv[2]
app.ExecuteAndWriteOutput()
```

If you want to handle the parameters from a Python dictionary, you can use the functions *SetParameters()* and *GetParameters()*.

```
params = {"in":"myInput.tif", "type.mean.radius":4}
app.SetParameters(params)
params2 = app.GetParameters()
```

Numpy array processing

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 Application engine module (called *ApplicationEngine*) which allows 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 it to NumPy array or just provide a file as explained in the previous section via Application.SetParameterString(...).

The bridge between 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 library (fork of PIL) and convert it to NumPy array. The NumPy array is used as an input to the application via *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 thus avoid writing output to a temporary file.

```
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)
inshow(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)
```

In-memory connection

Applications are often use as parts of larger processing workflow. 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.

Here is a Python code sample which connects several applications together:

```
import otbApplication 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.

Interactions with OTB pipeline

[Since OTB 6.6]

The application framework has been extended in order to provide ways to interact with the pipelines inside each application. It applies only to applications that use input or output images. Let's check what are the functions added to the `Application` class. There are a lot of getter functions:

Function name	return value
`GetImageOrigin(...)`	origin of the image (physical position of the first pixel center)
`GetImageSpacing(...)`	signed spacing of the image
`GetImageSize(...)`	size of the LargestPossibleRegion
`GetImageNbBands(...)`	number of components per pixel
`GetImageProjection(...)`	Projection WKT string
`GetImageKeywordlist(...)`	Ossim keywordlist (sensor model)
`GetImageMetaData(...)`	the entire MetaDataDictionary
`GetImageRequestedRegion(...)`	requested region
`GetImageBasePixelType(...)`	pixel type of the underlying Image/VectorImage.

All these getters functions use the following arguments:

- `key`: a string containing the key of the image parameter
- `idx`: an optional index (default is 0) that can be used to access ImageList parameters transparently

There is also a function to send orders to the pipeline:

> `PropagateRequestedRegion(key, region, idx=0)`: sets a given RequestedRegion on the image and propagate it, returns the memory print estimation. This function can be used to measure the requested portion of input images necessary to produce an extract of the full output.

Note: a requested region (like other regions in the C++ API of otb::Image) is just a pair of an image index and a size, that define a rectangular extract of the full image.

This set of function has been used to enhance the bridge between OTB images and Numpy arrays. There are now import and export functions available in Python that preserve the metadata of the image during conversions to Numpy arrays:

- `ExportImage(self, key)`: exports an output image parameter into a Python dictionary.
- `ImportImage(self, key, dict, index=0)`: imports the image from a Python dictionary into an image parameter (as a monoband image).
- `ImportVectorImage(self, key, dict, index=0)`: imports the image from a Python dictionary into an image parameter (as a multiband image).

The Python dictionary used has the following entries:

- `'array'`: the Numpy array containing the pixel buffer
- `'origin'`: origin of the image
- `'spacing'`: signed spacing of the image
- `'size'`: full size of the image
- `'region'`: region of the image present in the buffer
- `'metadata'`: metadata dictionary (contains projection, sensor model,...)

Now some basic Q&A about this interface:

Q: What portion of the image is exported to Numpy array? A: By default, the whole image is exported. If you had a non-empty requested region (the result of calling PropagateRequestedRegion()), then this region is exported.

Q: What is the difference between ImportImage and ImportVectorImage? A: The first one is here for Applications that expect a monoband otb::Image. In most cases, you will use the second one: ImportVectorImage.

Q: What kind of object are there in this dictionary export? A: The array is a numpy.ndarray. The other fields are wrapped objects from the OTB library but you can interact with them in a Python way: they support `len()` and `str()` operator, as well as bracket operator `[]`. Some of them also have a `keys()` function just like dictionaries.

This interface allows you to export OTB images (or extracts) to Numpy array, process them by other means, and re-import them with preserved metadatas. Please note that this is different from an in-memory connection.

Here is a small example of what can be done:

```
import otbApplication as otb

# Create a smoothing application
app = otb.Registry.CreateApplication("Smoothing")
app.SetParameterString("in",argv[1])

# only call Execute() to setup the pipeline, not ExecuteAndWriteOutput() which would
# run it and write the output image
app.Execute()

# Setup a special requested region
myRegion = otb.itkRegion()
myRegion['size'][0] = 20
myRegion['size'][1] = 25
myRegion['index'].Fill(10)
ram = app.PropagateRequestedRegion("out",myRegion)

# Check the requested region on the input image
print(app.GetImageRequestedRegion("in"))

# Create a ReadImageInfo application
app2 = otb.Registry.CreateApplication("ReadImageInfo")

# export "out" from Smoothing and import it as "in" in ReadImageInfo
ex = app.ExportImage("out")
app2.ImportVectorImage("in", ex)
app2.Execute()

# Check the result of ReadImageInfo
someKeys = ['sizex', 'sizey', 'spacingx', 'spacingy', 'sensor', 'projectionref']
for key in someKeys:
  print(key + ' : ' + str(app2.GetParameterValue(key)))

# Only a portion of "out" was exported but ReadImageInfo is still able to detect the
# correct full size of the image
```

Corner cases

There are a few corner cases to be aware of when using Python wrappers. They are often limitations, that one day may be solved in future versions. If it happens, this documentation will report the OTB version that fixes the issue.

Calling UpdateParameters()

These wrappers are made as a mirror of the C++ API, so there is a function `UpdateParameters()`. Its role is to update parameters that depend on others. It is called at least once at the beginning of `Execute()`.

In command line and GUI launchers, this functions gets called each time a parameter of the application is modified. In Python, this mechanism is not automated: there are cases where you may have to call it yourself.

Let's take an example with the application `PolygonClassStatictics`. In this application, the choices available in the parameter `field` depend on the list of fields actually present in the vector file `vec`. If you try to set the parameters `vec` and `field`, you will get an error:

```
import otbApplication as otb
app = otb.Registry.CreateApplication("PolygonClassStatistics")
app.SetParameterString("vec","../../src/OTB-Data/Input/Classification/variousVectors.
↪sqlite")
app.SetParameterString("field", "label")
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/gpasero/Projet_OTB/build/OTB/lib/otb/python/otbApplication.py", line␣
↪897, in SetParameterString
    def SetParameterString(self, *args): return _otbApplication.Application_
↪SetParameterString(self, *args)
RuntimeError: Exception thrown in otbApplication Application_SetParameterString: /
↪home/gpasero/Projet_OTB/src/OTB/Modules/Wrappers/ApplicationEngine/src/
↪otbWrapperListViewParameter.cxx:141:
itk::ERROR: ListViewParameter(0x149da10): Cannot find label
```

The error says that the choice `label` is not recognized, because `UpdateParameters()` was not called after setting the vector file. The solution is to call it before setting the `field` parameter:

```
app.UpdateParameters()
app.SetParameterString("field", "label")
```

No metadata in NumPy arrays

With the NumPy module, it is possible to convert images between OTB and NumPy arrays. For instance, when converting from OTB to NumPy array:

- An `Update()` of the underlying `otb::VectorImage` is requested. Be aware that the full image is generated.
- The pixel buffer is copied into a `numpy.array`

As you can see, there is no export of the metadata, such as origin, spacing, geographic projection. It means that if you want to import back a NumPy array into OTB, the image won't have any of these metadata. It can be a problem for applications doing geometry, projections, and also calibration.

Future developments will probably offer a more adapted structure to import and export images between OTB and the Python world.

Setting of EmptyParameter

Most of the parameters are set using functions `SetParameterXXX()`, except for one type of parameter: the `EmptyParameter`. This class was the first implementation of a boolean. It is now **deprecated**, you should use `BoolParameter` instead.

Let's take an example with the application `ReadImageInfo` when it was still using an `EmptyParameter` for parameter `keywordlist`:

```
import otbApplication as otb
app = otb.Registry.CreateApplication("ReadImageInfo")
```

If you want the get the state of parameter `keywordlist`, a boolean, use:

```
app.IsParameterEnabled("keywordlist")
```

To set this parameter ON/OFF, use the functions:

```
app.EnableParameter("keywordlist")
app.DisableParameter("keywordlist")
```

Don't try to use other functions to set the state of a boolean. For instance, try the following commands:

```
app.SetParameterInt("keywordlist", 0)
app.IsParameterEnabled("keywordlist")
```

You will get a state `True` even if you asked the opposite.

CHAPTER

SEVEN

APPLICATIONS REFERENCE DOCUMENTATION

Miscellaneous

BandMath - Band Math

Outputs a monoband image which is the result of a mathematical operation on several multi-band images.

Detailed description

This application performs a mathematical operation on several multi-band images and outputs the result into a monoband image. The given expression is computed at each pixel position. Evaluation of the mathematical formula is done by the muParser libraries.

The formula can be written using:

- numerical values (2.3, -5, 3.1e4, ...)
- variables containing pixel values (e.g. : 'im2b3' is the pixel value in 2nd image, 3rd band)
- binary operators:
  - '+' addition, '-' subtraction, '*' multiplication, '/' division
  - '^' raise x to the power of y
  - '<' less than, '>' greater than, '<=' less or equal, '>=' greater or equal
  - '==' equal, '!=' not equal
  - 'ǁ' logical or, '&&' logical and
- if-then-else operator: '(condition ? value_true : value_false)'
- functions : exp(), log(), sin(), cos(), min(), max(), ...

The full list of features and operators is available on the muParser website [1].

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BandMath* .

[1] Table: Parameters table for Band Math.

Parameter Key	Parameter Name	Parameter Type
il	Input image-list	Input image list
out	Output Image	Output image
ram	Available RAM (Mb)	Int
exp	Expression	String
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input image-list**: Image-list of operands to the mathematical expression.
- **Output Image**: Output image which is the result of the mathematical expressions on input image-list operands.
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Expression**: The muParser mathematical expression to apply on input 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_BandMath -il verySmallFSATSW_r.tif verySmallFSATSW_nir.tif verySmallFSATSW.tif␣
↪-out apTvUtBandMathOutput.tif -exp 'cos( im1b1 ) > cos( im2b1 ) ? im3b1 : im3b2'
```

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()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] http://beltoforion.de/article.php?a=muparser

BandMathX - Band Math X

This application performs mathematical operations on several multiband images.

Detailed description

This application performs a mathematical operation on several multi-band images and outputs the result into an image (multi- or mono-band, as opposed to the BandMath OTB-application). The mathematical formula is done by the muParserX libraries.

The list of features and the syntax of muParserX is available at [1].

As opposed to muParser (and thus the BandMath OTB-application [2]), muParserX supports vector expressions which allows outputting multi-band images.

Hereafter is a brief reference of the muParserX syntax

Fundamentals

The formula can be written using:

- numerical values (2.3, -5, 3.1e4, ...)
- variables containing pixel values (please, note the indexing of inputs from 1 to N). Examples for the first input image:
  - 'im1' a pixel from 1st input, made of n components (n bands)
  - 'im1b2' the 2nd component of a pixel from 1st input (band index is 1-based)
  - 'im1b2N3x4' a 3x4 pixels 'N'eighbourhood of a pixel the 2nd component of a pixel from the 1st input
  - 'im1PhyX' horizontal (X-axis) spacing of the 1st input.
  - 'im1PhyY' vertical spacing of the 1st input input.
  - 'im1b2Mean' mean of the 2nd component of the 1st input (global statistics)
  - 'im1b2Mini' minimum of the 2nd component of the 1st input (global statistics)
  - 'im1b2Maxi' maximum of the 2nd component of the 1st input (global statistics)
  - 'im1b2Sum' sum of the 2nd component of the 1st input (global statistics)
  - 'im1b2Var' variance of the 2nd component of the 1st input (global statistics)
  - 'idxX' and 'idxY' are the indices of the current pixel (generic variables)
- binary operators:
  - '+' addition, '-' subtraction, '*' multiplication, '/' division
  - '^' raise x to the power of y
  - '<' less than, '>' greater than, '<=' less or equal, '>=' greater or equal

  - '==' equal, '!=' not equal
  - logical operators: 'or', 'and', 'xor'
- if-then-else operator: '(condition ? value_true : value_false)'
- functions : abs(), exp(), log(), sin(), cos(), min(), max(), ...

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 (e.g. addition of a row vector with a column vector), or a scalar to a vector or matrix, or divide two vectors, etc.

Thus, it is important to remember that a pixel of n components is always represented as a row vector.

Example: im1 + im2 represents the addition of pixels from the 1st and 2nd inputs. This expression is consistent only if both inputs have the same number of bands.

Please, note that it is also possible to use the following expressions to obtain the same result:

- im1b1 + im2b1
- im1b2 + im2b2
- ...

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 feature is the possibility to perform operations that involve neighborhoods of pixels. Variables related to such neighborhoods are always defined following the imIbJNKxP pattern, 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 by1)
- KxP are two numbers that represent the size of the neighborhood (first one is related to the horizontal direction)

NB: 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
- etc.

For instance, if im1 represents the pixel of 3 bands image:

```
im1 - mean( im1b1N5x5, im1b2N5x5, im1b3N5x5 )
```

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).

In addition to the previous operators, other operators are available:

- existing operators/functions from muParserX, that were not originally defined for vectors and matrices (e.g. cos, sin). These new operators/functions keep the original names to which we added the prefix 'v' for vector (vcos, vsin, etc.)
- mult, div and pow operators, that perform element-wise multiplication, division or exponentiation of vector/matrices (e.g. im1 div im2).
- mlt, dv and pw operators, that perform multiplication, division or exponentiation of vector/matrices by a scalar (e.g. 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 (e.g.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 bandindices.

The application itself

The application can use an expression supplied with the 'exp' parameter. It can also use an input context file, that defines variables and expressions. An example of context 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 }
#E $dotpr( kernel1, im1b1N3x5 ); im2b1^expo$
```

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 #E <expr> (see limitations, below). Finally, we strongly recommend to read the OTB Cookbook which can be found at: http://www.orfeo-toolbox.org/packages/OTBCookBook.pdf

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
il	Input image-list	Input image list
out	Output Image	Output image
ram	Available RAM (Mb)	Int
exp	Expressions	String
incontext	Import context	Input File name
outcontext	Export context	Output File name
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).
- **Expressions**: Mathematical expression to apply.

[1] Table: Parameters table for Band Math X.

- **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 -exp 'cos( im1b1 ) + im2b1 * im3b1 - im3b2 +␣
↪ndvi( im3b3, im3b4 )'
```

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.

See Also

These additional resources can be useful for further information:

[1] http://articles.beltoforion.de/article.php?a=muparserx
[2] BandMath

CompareImages - 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[1] 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 Name	Parameter Type
ref	Reference image properties	Group
ref.in	Reference image	Input image
ref.channel	Reference image channel	Int
meas	Measured image properties	Group
meas.in	Measured image	Input image
meas.channel	Measured image channel	Int
roi	Region Of Interest (relative to reference image)	Group
roi.startx	Start X	Int
roi.starty	Start Y	Int
roi.sizex	Size X	Int
roi.sizey	Size Y	Int
mse	MSE	Float
mae	MAE	Float
psnr	PSNR	Float
count	count	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] Table: Parameters table for Images comparison.

- **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.

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 -roi.startx 20 -roi.starty 30 -roi.sizex 150 -roi.sizey 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 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

HyperspectralUnmixing - Hyperspectral data unmixing

Estimate abundance maps from an hyperspectral image and a set of endmembers.

Detailed description

The application applies a linear unmixing algorithmto an hyperspectral data cube. This method supposes that the mixture betweenaterials in the scene is macroscopic and simulates a linear mixing model ofspectra.

The Linear Mixing Model (LMM) acknowledges that reflectancespectrum associated with each pixel is a linear combination of purematerials in the recovery area, commonly known as endmembers. Endmembers canbe estimated using the VertexComponentAnalysis application.

The application allows estimating the abundance maps with several algorithms :

- Unconstrained Least Square (ucls)
- Image Space Reconstruction Algorithm (isra)
- Non-negative constrained
- Least Square (ncls)
- Minimum Dispersion Constrained Non Negative Matrix Factorization (MDMDNMF).

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input Image Filename	Input image
out	Output Image	Output image
ie	Input endmembers	Input image
ua	Unmixing algorithm	Choices
ua ucls	UCLS	*Choice*
ua ncls	NCLS	*Choice*
ua isra	ISRA	*Choice*
ua mdmdnmf	MDMDNMF	*Choice*
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image Filename**: The hyperspectral data cube input.

[1] Table: Parameters table for Hyperspectral data unmixing.

- **Output Image**: The output abundance map. The abundance fraction are stored in a multispectral image where band N corresponds to the fraction of endmembers N in each pixel.
- **Input endmembers**: The endmembers (estimated pure pixels) to use for unmixing. Must be stored as a multispectral 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␣
↪HyperspectralUnmixing.tif double -ua ucls
```

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.SetParameterOutputImagePixelType("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

KmzExport - 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. Furthemore, 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[1] 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 Name	Parameter Type
in	Input image	Input image
out	Output .kmz product	Output File name
tilesize	Tile Size	Int
logo	Image logo	Input image
legend	Image legend	Input image
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] Table: Parameters table for Image to KMZ Export.

- **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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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:

```
#!/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

OSMDownloader - Open Street Map layers import

Download vector data from OSM and store it to file

Detailed description

The application connects to Open Street Map server, downloads the data corresponding to the spatial extent of the support image, and filters the geometries based on OSM tags to produce a vector data file.

This application can be used to download reference data to perform the training of a machine learning model (see for instance [1]).

By default, the entire layer is downloaded. The application has a special mode to provide the list of available classes in the layers. The downloaded features are filtered by giving an OSM tag 'key'. In addition, the user can also choose what 'value' this key should have. More information about the OSM project at [2].

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *OSMDownloader* .

Parameter Key	Parameter Name	Parameter Type
out	Output vector data	Output vector data
support	Support image	Input image
key	OSM tag key	String
value	OSM tag value	String
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
printclasses	Displays available key/value classes	Boolean
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Output vector data: Vector data file to store downloaded features.

Support image: Image used to derive the spatial extent to be requested from OSM server (the bounding box of the extent is used). Be aware that a request with a large extent may be rejected by the server.

OSM tag key: OSM tag key to extract (highway, building...). It defines a category to select features.

OSM tag value: OSM tag value to extract (motorway, footway...). It defines the type of feature to select inside a category.

[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.

[1] Table: Parameters table for Open Street Map layers import.

- **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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

Displays available key/value classes: Print the key/value classes available for the selected support image. If enabled, the OSM tag Key (-key) and the output (-out) become optional.

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

This application requires an Internet access.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] TrainImagesClassifier
[2] http://www.openstreetmap.fr/

ObtainUTMZoneFromGeoPoint - 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[1] 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 Name	Parameter Type
lat	Latitude	Float
lon	Longitude	Float
utm	UTMZone	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Latitude**: Latitude value of desired point.
- **Longitude**: Longitude value of desired point.
- **UTMZone**: UTM Zone.
- **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)
```

[1] Table: Parameters table for Obtain UTM Zone From Geo Point.

```
ObtainUTMZoneFromGeoPoint.SetParameterFloat("lon", 124.0)

# The following line execute the application
ObtainUTMZoneFromGeoPoint.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

PixelValue - Pixel Value

Get the value of a pixel.

Detailed description

This application gives the value of a selected pixel. There are three ways to designate a pixel, with its index, its physical coordinate (in the physical space attached to the image), and with geographical coordinate system. Coordinates will be interpreted differently depending on which mode is chosen.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *PixelValue* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
coordx	X coordinate	Float
coordy	Y coordinate	Float
mode	Coordinate system used to designate the pixel	Choices
mode index	Index	*Choice*
mode physical	Image physical space	*Choice*
mode epsg	EPSG coordinates	*Choice*
mode.epsg.code	EPSG code	Int
cl	Channels	List
value	Pixel Value	String
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: Input image.

X coordinate: This will be the X coordinate interpreted depending on the chosen mode.

Y coordinate: This will be the Y coordinate interpreted depending on the chosen mode.

[1] Table: Parameters table for Pixel Value.

Coordinate system used to designate the pixel: Different modes can be selected, default mode is Index. Available choices are:

- **Index**: This mode uses the given coordinates as index to locate the pixel.
- **Image physical space**: This mode interprets the given coordinates in the image physical space.
- **EPSG coordinates**: This mode interprets the given coordinates in the specified geographical coordinate system by the EPSG code.
- **EPSG code**: This code is used to define a geographical coordinate system. If no system is specified, WGS84 (EPSG : 4326) is used by default.

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.SetParameterFloat("coordx", 50)

PixelValue.SetParameterFloat("coordy", 100)

# The following line execute the application
PixelValue.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

VertexComponentAnalysis - Vertex Component Analysis

Given a set of mixed spectral vectors, estimatereference substances also known as endmembers using the VertexComponent Analysis algorithm.

Detailed description

Apply the Vertex Component Analysis [1] toan hyperspectral image to extract endmembers. Given a set of mixed-spectral vectors (multispectral or hyperspectral), the applicationestimates the spectral signature of reference substances also knownas endmembers.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input Image	Input image
ne	Number of endmembers	Int
outendm	Output Endmembers	Output image
rand	set user defined seed	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image**: Input hyperspectral data cube.
- **Number of endmembers**: The number of endmembers to extract from the hyperspectral image.
- **Output Endmembers**: Endmembers, stored in aone-line multi-spectral image.Each pixel corresponds to oneendmembers and each band values corresponds to the spectralsignature of the corresponding 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")
```

[1] Table: Parameters table for Vertex Component Analysis.

```
# The following lines set all the application parameters:
VertexComponentAnalysis.SetParameterString("in", "cupriteSubHsi.tif")

VertexComponentAnalysis.SetParameterInt("ne", 5)

VertexComponentAnalysis.SetParameterString("outendm", "VertexComponentAnalysis.tif")
VertexComponentAnalysis.SetParameterOutputImagePixelType("outendm", 7)

# The following line execute the application
VertexComponentAnalysis.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] J. M. P. Nascimento and J. M. B. Dias, Vertexcomponent analysis: a fast algorithm to unmix hyperspectral data,in IEEE Transactions on Geoscience and Remote Sensing, vol. 43,no. 4, pp. 898-910, April 2005.J. M. P. Nascimento andJ. M. B. Dias, Vertex component analysis: a fast algorithm tounmix hyperspectral data, in IEEE Transactions on Geoscience andRemote Sensing, vol. 43, no. 4, pp. 898-910, April 2005.

Feature Extraction

BinaryMorphologicalOperation - Binary Morphological Operation

Performs morphological operations on an input image channel

Detailed description

This application performs binary morphological operations on a mono band image or a channel of the input.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *BinaryMorphologicalOperation* .

[1] Table: Parameters table for Binary Morphological Operation.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
structype	Type of structuring element	Choices
structype ball	Ball	*Choice*
structype cross	Cross	*Choice*
structype.ball.xradius	Structuring element X radius	Int
structype.ball.yradius	Structuring element Y radiuss	Int
filter	Morphological Operation	Choices
filter dilate	Dilate	*Choice*
filter erode	Erode	*Choice*
filter opening	Opening	*Choice*
filter closing	Closing	*Choice*
filter.dilate.foreval	Foreground value	Float
filter.dilate.backval	Background value	Float
filter.erode.foreval	Foreground value	Float
filter.erode.backval	Background value	Float
filter.opening.foreval	Foreground value	Float
filter.opening.backval	Background value	Float
filter.closing.foreval	Foreground value	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: The input image to be filtered.

Output Image: Output image.

Selected Channel: The selected channel index.

Available RAM (Mb): Available memory for processing (in MB).

Type of structuring element: Choice of the structuring element type. Available choices are:

- **Ball**
- **Structuring element X radius**: The structuring element radius along the X axis.
- **Structuring element Y radiuss**: The structuring element radius along the y axis.
- **Cross**

Morphological Operation: Choice of the morphological operation. Available choices are:

- **Dilate**
- **Foreground value**: Set the foreground value, default is 1.0.
- **Background value**: Set the background value, default is 0.0.
- **Erode**
- **Foreground value**: Set the foreground value, default is 1.0.
- **Background value**: Set the background value, default is 0.0.
- **Opening**
- **Foreground value**: Set the foreground value, default is 1.0.
- **Background value**: Set the background value, default is 0.0.

- **Closing**
- **Foreground value**: Set the foreground value, default is 1.0.

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.xradius 5 -structype.ball.yradius 5 -filter erode
```

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(
↪"BinaryMorphologicalOperation")

# 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.

ComputePolylineFeatureFromImage - 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[1] 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 Name	Parameter Type
in	Input Image	Input image
vd	Vector Data	Input vector data
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
expr	Feature expression	String
field	Feature name	String
out	Output Vector Data	Output vector data
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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, ROADSA...).

Output Vector Data: The output vector data containing polylines with a new field.

[1] Table: Parameters table for Compute Polyline Feature From 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_ComputePolylineFeatureFromImage -in NDVI.TIF -vd roads_ground_truth.shp -expr
↪"(b1 > 0.4)" -field NONDVI -out PolylineFeatureFromImage_LI_NONDVI_gt.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 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.

DSFuzzyModelEstimation - 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[1] 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 Name	Parameter Type
psin	Input Positive Vector Data	Input vector data
nsin	Input Negative Vector Data	Input vector data
belsup	Belief Support	String list
plasup	Plausibility Support	String list
cri	Criterion	String
wgt	Weighting	Float
initmod	initialization model	Input File name
desclist	Descriptor list	String list
maxnbit	Maximum number of iterations	Int
optobs	Optimizer Observer	Boolean
out	Output filename	Output File name
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.
- **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 cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_wr.shp -belsup "ROADSA" -
↪plasup "NONDVI" "ROADSA" "NOBUIL" -initmod Dempster-Shafer/DSFuzzyModel_Init.xml -
↪maxnbit 4 -optobs true -out DSFuzzyModelEstimation.xml
```

[1] Table: Parameters table for Fuzzy Model estimation.

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.shp")

DSFuzzyModelEstimation.SetParameterString("nsin",
↪"cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_wr.shp")

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","true")

DSFuzzyModelEstimation.SetParameterString("out", "DSFuzzyModelEstimation.xml")

# The following line execute the application
DSFuzzyModelEstimation.ExecuteAndWriteOutput()
```

Limitations

None.

Authors

This application has been written by OTB-Team.

EdgeExtraction - Edge Feature Extraction

This application computes edge features on every pixel of the input image selected channel

Detailed description

This application computes edge features on a selected channel of the input.It uses different filter such as gradient, Sobel and Touzi

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input Image	Input image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
filter	Edge feature	Choices
filter gradient	Gradient	*Choice*
filter sobel	Sobel	*Choice*
filter touzi	Touzi	*Choice*
filter.touzi.xradius	The X radius of the neighborhood.	Int
filter.touzi.yradius	The Y radius of the neighborhood.	Int
out	Feature Output Image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: The input image on which the features are computed.

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**: This filter computes the gradient magnitude of the image at each pixel.
- **Sobel**: This filter uses the Sobel operator to calculate the image gradient and then finds the magnitude of this gradient vector.
- **Touzi**: This filter is more suited for radar images. It has a spatial parameter to avoid speckle noise perturbations. The larger the radius is, less sensible to the speckle noise the filter is, but micro edge will be missed.
- **The X radius of the neighborhood.**
- **The Y radius of the neighborhood.**

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:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication
```

[1] Table: Parameters table for Edge Feature Extraction.

```
# 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

GrayScaleMorphologicalOperation - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GrayScaleMorphologicalOperation* .

[1] Table: Parameters table for Grayscale Morphological Operation.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Feature Output Image	Output image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
structype	Structuring Element Type	Choices
structype ball	Ball	*Choice*
structype cross	Cross	*Choice*
structype.ball.xradius	The Structuring Element X Radius	Int
structype.ball.yradius	The Structuring Element Y Radius	Int
filter	Morphological Operation	Choices
filter dilate	Dilate	*Choice*
filter erode	Erode	*Choice*
filter opening	Opening	*Choice*
filter closing	Closing	*Choice*
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.xradius 5 -structype.ball.yradius 5 -filter erode
```

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(
↪"GrayScaleMorphologicalOperation")

# 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

HaralickTextureExtraction - Haralick Texture Extraction

Computes Haralick textural features on the selected channel of the input image

Detailed description

This application computes three sets of Haralick features [1][2].

- simple: a set of 8 local Haralick features: Energy (texture uniformity) , Entropy (measure of randomness of intensity image), Correlation (how correlated a pixel is to its neighborhood), Inverse Difference Moment

(measures the texture homogeneity), Inertia (intensity contrast between a pixel and its neighborhood), Cluster Shade, Cluster Prominence, Haralick Correlation;

- advanced: a set of 10 advanced Haralick features : Mean, Variance (measures the texture heterogeneity), Dissimilarity, Sum Average, Sum Variance, Sum Entropy, Difference of Entropies, Difference of Variances, IC1, IC2;
- higher: a set of 11 higher Haralick features : Short Run Emphasis (measures the texture sharpness), Long Run Emphasis (measures the texture roughness), Grey-Level Nonuniformity, Run Length Nonuniformity, Run Percentage (measures the texture sharpness homogeneity), 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.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *HaralickTextureExtraction* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
channel	Selected Channel	Int
step	Computation step	Int
ram	Available RAM (Mb)	Int
parameters	Texture feature parameters	Group
parameters.xrad	X Radius	Int
parameters.yrad	Y Radius	Int
parameters.xoff	X Offset	Int
parameters.yoff	Y Offset	Int
parameters.min	Image Minimum	Float
parameters.max	Image Maximum	Float
parameters.nbbin	Histogram number of bin	Int
texture	Texture Set Selection	Choices
texture simple	Simple Haralick Texture Features	*Choice*
texture advanced	Advanced Texture Features	*Choice*
texture higher	Higher Order Texture Features	*Choice*
out	Output Image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: The input image to compute the features on.

Selected Channel: The selected channel index.

Computation step: Step (in pixels) to compute output texture values. The first computed pixel position is shifted by (step-1)/2 in both directions.

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.

[1] Table: Parameters table for Haralick Texture Extraction.

- **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 10 advanced texture feature output image. The image channels are: Mean, Variance, Dissimilarity, 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 -texture simple -out HaralickTextures.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 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

The computation of the features is based on a Gray Level Co-occurrence matrix (GLCM) from the quantized input image. Consequently the quantization parameters (min, max, nbbin) must be appropriate to the range of the pixel values.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] HARALICK, Robert M., SHANMUGAM, Karthikeyan, et al. Textural features for image classification. IEEE Transactions on systems, man, and cybernetics, 1973, no 6, p. 610-621.

[2] otbScalarImageToTexturesFilter, otbScalarImageToAdvancedTexturesFilter and otbScalarImageToHigherOrderTexturesFilter classes

HomologousPointsExtraction - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *HomologousPointsExtraction* .

[1] Table: Parameters table for Homologous Points Extraction.

Parameter Key	Parameter Name	Parameter Type
in1	Input Image 1	Input image
band1	Input band 1	Int
in2	Input Image 2	Input image
band2	Input band 2	Int
algorithm	Keypoints detection algorithm	Choices
algorithm surf	SURF algorithm	*Choice*
algorithm sift	SIFT algorithm	*Choice*
threshold	Distance threshold for matching	Float
backmatching	Use back-matching to filter matches.	Boolean
mode	Keypoints search mode	Choices
mode full	Extract and match all keypoints (no streaming)	*Choice*
mode geobins	Search keypoints in small spatial bins regularly spread across first image	*Choice*
mode.geobins.binsize	Size of bin	Int
mode.geobins.binsizey	Size of bin (y direction)	Int
mode.geobins.binstep	Steps between bins	Int
mode.geobins.binstepy	Steps between bins (y direction)	Int
mode.geobins.margin	Margin from image border to start/end bins (in pixels)	Int
precision	Estimated precision of the colocalisation function (in pixels).	Float
mfilter	Filter points according to geographical or sensor based colocalisation	Boolean
2wgs84	If enabled, points from second image will be exported in WGS84	Boolean
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
out	Output file with tie points	Output File name
outvector	Output vector file with tie points	Output File name
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 regulary 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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 -out homologous.txt
```

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

LineSegmentDetection - 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 contrario validation approach (Desolneux, Moisan, and Morel). The algorithm was published by Rafael Gromponevon Gioi, Jérémie Jakubowicz, Jean-Michel Morel and Gregory Randall. 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LineSegmentDetection* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Detected lines	Output vector data
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
norescale	No rescaling in [0, 255]	Boolean
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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:

[1] Table: Parameters table for Line segment detection.

```
#!/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/

LocalStatisticExtraction - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LocalStatisticExtraction* .

[1] Table: Parameters table for Local Statistic Extraction.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
radius	Neighborhood radius	Int
out	Feature Output Image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).
- **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

MorphologicalClassification - Morphological Classification

Performs morphological convex, concave and flat classification on an input image channel

Detailed description

This algorithm is based on the following publication: 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.

This application perform the following decision rule to classify a pixel between the three classes Convex, Concave and Flat. Let f denote the input image and $\psi_N(f)$ the geodesic leveling of f with a structuring element of size N. One can derive the following decision rule to classify f into Convex (label $\breve{k}$), Concave (label $\widehat{k}$) and Flat (label $\bar{k}$):

$$f(n) = \begin{cases} \breve{k} & : f - \psi_N(f) > \sigma \\ \widehat{k} & : \psi_N(f) - f > \sigma \\ \bar{k} & :\mid f - \psi_N(f) \mid \leq \sigma \end{cases}$$

The output is a labeled image (0 : Flat, 1 : Convex, 2 : Concave)

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MorphologicalClassification* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
structype	Structuring Element Type	Choices
structype ball	Ball	*Choice*
structype cross	Cross	*Choice*
radius	Radius	Int
sigma	Sigma value for leveling tolerance	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image**: The input image to be classified.
- **Output Image**: The output classified image with 3 different values (0 : Flat, 1 : Convex, 2 : Concave).

[1] Table: Parameters table for Morphological Classification.

- **Selected Channel**: The selected channel index for input image.
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Structuring Element Type**: Choice of the structuring element type. Available choices are:
- **Ball**
- **Cross**
- **Radius**: Radius of the structuring element (in pixels), default value is 5.
- **Sigma value for leveling tolerance**: Sigma value for leveling tolerance, default value is 0.5.
- **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_MorphologicalClassification -in ROI_IKO_PAN_LesHalles.tif -channel 1 -
↪structype ball -radius 5 -sigma 0.5 -out output.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 MorphologicalClassification␣
↪application
MorphologicalClassification = otbApplication.Registry.CreateApplication(
↪"MorphologicalClassification")

# The following lines set all the application parameters:
MorphologicalClassification.SetParameterString("in", "ROI_IKO_PAN_LesHalles.tif")

MorphologicalClassification.SetParameterInt("channel", 1)

MorphologicalClassification.SetParameterString("structype","ball")

MorphologicalClassification.SetParameterInt("radius", 5)

MorphologicalClassification.SetParameterFloat("sigma", 0.5)

MorphologicalClassification.SetParameterString("out", "output.tif")

# The following line execute the application
MorphologicalClassification.ExecuteAndWriteOutput()
```

Limitations

Generation of the morphological classification is not streamable, pay attention to this fact when setting the radius size of the structuring element.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

otbConvexOrConcaveClassificationFilter class

MorphologicalMultiScaleDecomposition - Morphological Multi Scale Decomposition

Perform a geodesic morphology based image analysis on an input image channel

Detailed description

This application recursively apply geodesic decomposition.

This algorithm is derived from the following publication:

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.

It provides a geodesic decomposition of the input image, with the following scheme. Let f_0 denote the input image, $\breve{\mu}_N\ (f)$ denote the convex membership function, $\widehat{\mu}_N\ (f)$ denote the concave membership function and $\psi_N(f)$ denote the leveling function, for a given radius N as defined in the documentation of the GeodesicMorphologyDecompositionImageFilter. Let $[N_1, \ldots, N_n]$ denote a range of increasing radius (or scales). The iterative decomposition is defined as follows:

$f_i = \psi_{N_i}(f_{i-1})$

$\widehat{f}_i = \widehat{\mu}_{N_i}\ (f_i)$

$\breve{f}_i = \breve{\mu}_{N_i}\ (f_i)$

The $\breve{f}_i$ and $\widehat{f}_i$ are membership function for the convex (resp. concave) objects whose size is comprised between N_{i-1} and N_i

Output convex, concave and leveling images with B bands, where n is the number of levels.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MorphologicalMultiScaleDecomposition* .

[1] Table: Parameters table for Morphological Multi Scale Decomposition.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
outconvex	Output Convex Image	Output image
outconcave	Output Concave Image	Output image
outleveling	Output Image	Output image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
structype	Structuring Element Type	Choices
structype ball	Ball	*Choice*
structype cross	Cross	*Choice*
radius	Initial radius	Int
step	Radius step.	Int
levels	Number of levels use for multi scale	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image**: The input image to be classified.
- **Output Convex Image**: The output convex image with N bands.
- **Output Concave Image**: The output concave concave with N bands.
- **Output Image**: The output leveling image with N bands.
- **Selected Channel**: The selected channel index for input image.
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Structuring Element Type**: Choice of the structuring element type. Available choices are:
- **Ball**
- **Cross**
- **Initial radius**: Initial radius of the structuring element (in pixels).
- **Radius step.**: Radius step along the profile (in pixels).
- **Number of levels use for multi scale**: Number of levels use for multi scale.
- **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_MorphologicalMultiScaleDecomposition -in ROI_IKO_PAN_LesHalles.tif -structype␣
↪ball -channel 1 -radius 2 -levels 2 -step 3 -outconvex convex.tif -outconcave␣
↪concave.tif -outleveling leveling.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 MorphologicalMultiScaleDecomposition␣
↪application
```

```
MorphologicalMultiScaleDecomposition = otbApplication.Registry.CreateApplication(
↪"MorphologicalMultiScaleDecomposition")

# The following lines set all the application parameters:
MorphologicalMultiScaleDecomposition.SetParameterString("in", "ROI_IKO_PAN_LesHalles.
↪tif")

MorphologicalMultiScaleDecomposition.SetParameterString("structype","ball")

MorphologicalMultiScaleDecomposition.SetParameterInt("channel", 1)

MorphologicalMultiScaleDecomposition.SetParameterInt("radius", 2)

MorphologicalMultiScaleDecomposition.SetParameterInt("levels", 2)

MorphologicalMultiScaleDecomposition.SetParameterInt("step", 3)

MorphologicalMultiScaleDecomposition.SetParameterString("outconvex", "convex.tif")

MorphologicalMultiScaleDecomposition.SetParameterString("outconcave", "concave.tif")

MorphologicalMultiScaleDecomposition.SetParameterString("outleveling", "leveling.tif")

# The following line execute the application
MorphologicalMultiScaleDecomposition.ExecuteAndWriteOutput()
```

Limitations

Generation of the multi scale decomposition is not streamable, pay attention to this fact when setting the number of iterating levels.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

otbGeodesicMorphologyDecompositionImageFilter class

MorphologicalProfilesAnalysis - Morphological Profiles Analysis

Performs morphological profiles analysis on an input image channel.

Detailed description

This algorithm is derived from the following publication:

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.

Depending of the profile selection, the application provides:

```
- The multi scale geodesic morphological opening or closing profile of the input␣
↪image.
- The multi scale derivative of the opening or closing profile.
- The parameter (called characteristic) of the maximum derivative value of the multi␣
↪scale closing or opening profile for which this maxima occurs.
- The labeled classification of the input image.
```

The behavior of the classification is :

Given x_1 and x_2 two membership values, L_1, L_2 two labels associated, and σ a tolerance value, the following decision rule is applied:

$$L = \begin{cases} L_1 & : x_1 > x_2 \quad and \quad x_1 > \sigma \\ L_2 & : x_2 > x_1 \quad and \quad x_2 > \sigma \\ 0 & : otherwise. \end{cases}$$

The output image can be :- A N multi band image for the opening/closing normal or derivative profiles. - A mono band image for the opening/closing characteristics. - A labeled image for the classification.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MorphologicalProfilesAnalysis* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
structype	Structuring Element Type	Choices
structype ball	Ball	*Choice*
structype cross	Cross	*Choice*
size	Profile Size	Int
radius	Initial radius	Int
step	Radius step.	Int
profile	Profile	Choices
profile opening	opening	*Choice*
profile closing	closing	*Choice*
profile derivativeopening	derivativeopening	*Choice*
profile derivativeclosing	derivativeclosing	*Choice*
profile openingcharacteristics	openingcharacteristics	*Choice*
profile closingcharacteristics	closingcharacteristics	*Choice*
profile classification	classification	*Choice*
profile.classification.sigma	Sigma value for leveling tolerance	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: The input image.

Output Image: The output image.

Selected Channel: The selected channel index for input image.

[1] Table: Parameters table for Morphological Profiles Analysis.

Available RAM (Mb): Available memory for processing (in MB).

Structuring Element Type: Choice of the structuring element type. Available choices are:

- **Ball**
- **Cross**

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).

Profile Available choices are:

- **opening**
- **closing**
- **derivativeopening**
- **derivativeclosing**
- **openingcharacteristics**
- **closingcharacteristics**
- **classification**
- **Sigma value for leveling tolerance**: Sigma value for leveling tolerance.

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_MorphologicalProfilesAnalysis -in ROI_IKO_PAN_LesHalles.tif -channel 1 -
↪structype ball -profile classification -size 5 -radius 1 -step 1 -profile.
↪classification.sigma 1 -out output.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 MorphologicalProfilesAnalysis␣
↪application
MorphologicalProfilesAnalysis = otbApplication.Registry.CreateApplication(
↪"MorphologicalProfilesAnalysis")

# The following lines set all the application parameters:
MorphologicalProfilesAnalysis.SetParameterString("in", "ROI_IKO_PAN_LesHalles.tif")

MorphologicalProfilesAnalysis.SetParameterInt("channel", 1)

MorphologicalProfilesAnalysis.SetParameterString("structype","ball")

MorphologicalProfilesAnalysis.SetParameterString("profile","classification")
```

```
MorphologicalProfilesAnalysis.SetParameterInt("size", 5)

MorphologicalProfilesAnalysis.SetParameterInt("radius", 1)

MorphologicalProfilesAnalysis.SetParameterInt("step", 1)

MorphologicalProfilesAnalysis.SetParameterFloat("profile.classification.sigma", 1)

MorphologicalProfilesAnalysis.SetParameterString("out", "output.tif")

# The following line execute the application
MorphologicalProfilesAnalysis.ExecuteAndWriteOutput()
```

Limitations

Generation of the morphological profile is not streamable, pay attention to this fact when setting the radius initial size and step of the structuring element.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

otbMorphologicalOpeningProfileFilter, otbMorphologicalClosingProfileFilter, otbProfileToProfileDerivativeFilter, otbProfileDerivativeToMultiScaleCharacteristicsFilter, otbMultiScaleConvexOrConcaveClassificationFilter, classes

RadiometricIndices - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *RadiometricIndices* .

[1] Table: Parameters table for Radiometric Indices.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
channels	Channels selection	Group
channels.blue	Blue Channel	Int
channels.green	Green Channel	Int
channels.red	Red Channel	Int
channels.nir	NIR Channel	Int
channels.mir	Mir Channel	Int
list	Available Radiometric Indices	List
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 -out RadiometricIndicesImage.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 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

SFSTextureExtraction - SFS Texture Extraction

Computes Structural Feature Set textures on every pixel of the input image selected channel

Detailed description

Structural Feature Set [1] are based on the histograms of the pixels in multiple directions of the image. The SFS-TextureExtraction application computes the 6 following features: SFS'Length, SFS'Width, SFS'PSI, SFS'W-Mean, SFS'Ratio and SFS'SD (Standard Deviation). The texture indices are computed from the neighborhood of each pixel. It is possible to change the length of the calculation line (spatial threshold), as well as the maximum difference between a pixel of the line and the pixel at the center of the neighborhood (spectral threshold) [2].

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SFSTextureExtraction* .

[1] Table: Parameters table for SFS Texture Extraction.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
channel	Selected Channel	Int
ram	Available RAM (Mb)	Int
parameters	Texture feature parameters	Group
parameters.spethre	Spectral Threshold	Float
parameters.spathre	Spatial Threshold	Int
parameters.nbdir	Number of Direction	Int
parameters.alpha	Alpha	Float
parameters.maxcons	Ratio Maximum Consideration Number	Int
out	Feature Output Image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.spathre 100 -out SFSTextures.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 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:

[1] HUANG, Xin, ZHANG, Liangpei, et LI, Pingxiang. Classification and extraction of spatial features in urban areas using high-resolution multispectral imagery. IEEE Geoscience and Remote Sensing Letters, 2007, vol. 4, no 2, p. 260-264.

[2] otbSFSTexturesImageFilter class

VectorDataDSValidation - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataDSValidation* .

[1] Table: Parameters table for Vector Data validation.

Parameter Key	Parameter Name	Parameter Type
in	Input Vector Data	Input vector data
descmod	Descriptors model filename	Input File name
belsup	Belief Support	String list
plasup	Plausibility Support	String list
cri	Criterion	String
thd	Criterion threshold	Float
out	Output Vector Data	Output vector data
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Vector Data**: Input vector data to validate.
- **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 cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp -descmod␣
↪DSFuzzyModel.xml -out VectorDataDSValidation.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 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

Stereo

BlockMatching - 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 for a pair of images in epipolar geometry.

This application is part of the stereovision pipeline. It can be used after having computed epipolar grids (with StereoRectificationGridGenerator) and resampled each input image into epipolar geometry (with GridBasedImageResampling).

The application searches locally for the displacement between a reference image and a secondary image. The correspondence is

- SSD : Sum of Squared Distances
- NCC : Normalized Cross-Correlation
- Lp : Lp pseudo norm

Once the best integer disparity is found, an optional step of sub-pixel disparity estimation can be performed, with various algorithms (triangular interpolation, parabollic interpolation, dichotimic search). As post-processing, there is an optional step of median filtering on the disparities. One can chose 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 image of optimal metric values 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[1] 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 Name	Parameter Type
io	Input and output data	Group
io.inleft	Left input image	Input image
io.inright	Right input image	Input image
io.out	The output disparity map	Output image
io.outmask	The output mask corresponding to all criterions	Output image
io.outmetric	Flag to output optimal metric values as well	Boolean
mask	Image masking parameters	Group
mask.inleft	Mask to discard left pixels	Input image
mask.inright	Mask to discard right pixels	Input image
mask.nodata	Discard pixels with no-data value	Float
mask.variancet	Discard pixels with low local variance	Float
bm	Block matching parameters	Group
bm.metric	Block-matching metric	Choices
bm.metric ssd	Sum of Squared Distances	*Choice*
bm.metric ncc	Normalized Cross-Correlation	*Choice*
bm.metric lp	Lp pseudo-norm	*Choice*
bm.metric.lp.p	p value	Float
bm.radius	Radius of blocks	Int
bm.minhd	Minimum horizontal disparity	Int
bm.maxhd	Maximum horizontal disparity	Int
bm.minvd	Minimum vertical disparity	Int
bm.maxvd	Maximum vertical disparity	Int
bm.subpixel	Sub-pixel interpolation	Choices
bm.subpixel none	None	*Choice*
bm.subpixel parabolic	Parabolic fit	*Choice*
bm.subpixel triangular	Triangular fit	*Choice*
bm.subpixel dichotomy	Dichotomy search	*Choice*
bm.step	Computation step	Int
bm.startx	X start index	Int
bm.starty	Y start index	Int
bm.medianfilter	Median filtering of disparity map	Group
bm.medianfilter.radius	Radius	Int
bm.medianfilter.incoherence	Incoherence threshold	Float
bm.initdisp	Initial disparities	Choices
bm.initdisp none	None	*Choice*
bm.initdisp uniform	Uniform initial disparity	*Choice*
bm.initdisp maps	Initial disparity maps	*Choice*
bm.initdisp.uniform.hdisp	Horizontal initial disparity	Int
bm.initdisp.uniform.vdisp	Vertical initial disparity	Int
bm.initdisp.uniform.hrad	Horizontal exploration radius	Int
bm.initdisp.uniform.vrad	Vertical exploration radius	Int
bm.initdisp.maps.hmap	Horizontal initial disparity map	Input image
bm.initdisp.maps.vmap	Vertical initial disparity map	Input image

Continued on next page

[1] Table: Parameters table for Pixel-wise Block-Matching.

Table 7.1 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
bm.initdisp.maps.hrad	Horizontal exploration radius	Int
bm.initdisp.maps.vrad	Vertical exploration radius	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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). It should have the same size and same physical space as the right input. This image can be generated by GridBasedImageResampling.
- **Right input image**: The right input (secondary). It should have the same size and same physical space as the left input. This image can be generated by GridBasedImageResampling.
- **The output disparity map**: An image containing the estimated disparities as well as the metric values if the option is used. If no metric is output and no sub-pixel interpolation is done, pixel type canbe a signed integer. In the other cases, floating point pixel is advised.
- **The output mask corresponding to all criterions**: An output mask image corresponding to all citerions (see masking parameters). Only required if variance threshold or nodata criterions are set. Output pixel type is unsigned 8bit by default.
- **Flag to output optimal metric values as well**: If enabled, the output image will have a third 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.

- **Mask to discard left pixels**: This parameter allows providing a custom mask for the left image. Block matching will be only perform on pixels inside the mask (non-zero values).
- **Mask to discard right pixels**: This parameter allows providing a custom mask for the right image. Block matching will be perform only on pixels inside the mask (non-zero values).
- **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**: Metric to evaluate matching between two local windows. 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 search.
- **Parabolic fit**: The metric values closest to the best match are used in order to fit a parabola to the local extremum of the metric surface. The peak position of this parabola is output.
- **Triangular fit**: The metric values closest to the best match are used in order to fit a triangular peak to the local extremum of the metric surface. The peak position of this triangle is output.
- **Dichotomy search**: An iterative dichotomic search is performed to find the best sub-pixel position. The window in the right image is resampled at sub-pixel positions to estimate the match.
- **Computation step**: Location step between computed disparities. Disparities will be computed every 'step' pixels in the left image (step for both rows and columns). For instance, a value of 1 corresponds to the classic dense disparity map.
- **X start index**: X start index of the subsampled grid (wrt the input image grid). See parameter bm.step.
- **Y start index**: Y start index of the subsampled grid (wrt the input image grid). See parameter bm.step.
- **Median filtering of disparity map**: Use a median filter to get a smooth disparity map.
- **Radius**: Radius (in pixels) 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 to define the exploration area. This area in the right image is centered on the current position shifted by the initial disparity estimate, and has a given exploration radius in horizontal and vertical directions.
  - **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 10 -mask.variancet 10 -io.out MyDisparity.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 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:

[1] StereoRectificationGridGenerator
[2] GridBasedImageResampling

DisparityMapToElevationMap - 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.

This application is part of the stereo reconstruction pipeline. It can be used after having computed the disparity map with BlockMatching.

The needed inputs are [the disparity map, the stereo pair (in original geometry) and the epipolar deformation grids. These grids (computed by StereoRectificationGridGenerator) have to contain the transform between the original geometry (stereo pair) and the epipolar geometry (disparity map). The algorithm for each disparity is the following :]

- skip if position is discarded by the disparity mask
- compute left ray : transform the current position from epipolar geometry to left sensor geometry (left rectification grid)
- compute right ray : shift the current position with current disparity and transform from epipolar geometry to right sensor (right rectification grid)
- estimate best 3D intersection between left and right rays
- for the ground cell of the obtained 3D point, keep its elevation if greater than current elevation (keeps the maximum of elevations of all 3D points in each cell)

Minimum and maximum elevations settings are here to bound the reconstructed DEM.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
io	Input and output data	Group
io.in	Input disparity map	Input image
io.left	Left sensor image	Input image
io.right	Right sensor image	Input image
io.lgrid	Left Grid	Input image
io.rgrid	Right Grid	Input image
io.out	Output elevation map	Output image
io.mask	Disparity mask	Input image
step	DEM step	Float
hmin	Minimum elevation expected	Float
hmax	Maximum elevation expected	Float
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

[Input and output data]: This group of parameters allows one to set input images, output images and grids.

[1] Table: Parameters table for Disparity map to elevation map.

- **Input disparity map**: The input disparity map (horizontal disparity in first band, vertical in second). This map can be computed by BlockMatching application.
- **Left sensor image**: Left image in original (sensor) geometry. Only the geometric model of this image will be used, not the pixel values.
- **Right sensor image**: Right image in original (sensor) geometry. Only the geometric model of this image will be used, not the pixel values.
- **Left Grid**: Left epipolar grid (deformation grid between left sensor et disparity spaces).
- **Right Grid**: Right epipolar grid (deformation grid between right sensor et disparity spaces).
- **Output elevation map**: Output elevation map in ground projection. Elevation values are in meters. Floating point pixel type are expected.
- **Disparity mask**: Masked disparity pixels won't be projected (mask values equal to zero).

DEM step: Spacing of the output elevation map (in meters).

Minimum elevation expected: Minimum elevation expected (in meters).

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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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 -io.lgrid grid_epi_left.tif -io.rgrid grid_epi_right.tif -io.
↪out dem.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

The epipolar deformation grid should be able to entirely fit in memory.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] StereoRectificationGridGenerator
[2] BlockMatching

FineRegistration - Fine Registration

Estimate disparity map between two images.

Detailed description

This application computes a disparity map between two images that correspond to the same scene. It is intended for case where small misregistration between images should be estimated and fixed. The search is performed in 2D.

The algorithm uses an iterative approach to estimate a best match between local patches. The typical use case is registration betwween similar bands, or between two acquisitions. The output image contains X and Y offsets, as well as the metric value. A sub-pixel accuracy can be expected. The input images should have the same size and same physical space.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *FineRegistration* .

Parameter Key	Parameter Name	Parameter Type
ref	Reference Image	Input image
sec	Secondary Image	Input image
out	Output Image	Output image
erx	Exploration Radius X	Int
ery	Exploration Radius Y	Int
mrx	Metric Radius X	Int
mry	Metric Radius Y	Int
w	Image To Warp	Input image
wo	Output Warped Image	Output image
cox	Coarse Offset X	Float
coy	Coarse Offset Y	Float
ssrx	Sub-Sampling Rate X	Float
ssry	Sub-Sampling Rate Y	Float
rgsx	Reference Gaussian Smoothing X	Float
rgsy	Reference Gaussian Smoothing Y	Float
sgsx	Secondary Gaussian Smoothing X	Float
sgsy	Secondary Gaussian Smoothing Y	Float
m	Metric	String
spa	SubPixelAccuracy	Float
cva	ConvergenceAccuracy	Float
vmlt	Validity Mask Lower Threshold	Float
vmut	Validity Mask Upper Threshold	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Reference Image**: The reference image.
- **Secondary Image**: The secondary image.
- **Output Image**: The output image contains 3 bands, for X offset, Y offset and the metric value. It may contain a 4th one with the validity mask (if used).
- **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, used as an initial offset for all pixels.
- **Coarse Offset Y**: Coarse offset along y (in physical space) between the two images, used as an initial offset for all pixels.

[1] Table: Parameters table for Fine Registration.

- **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 compute the validity mask. This mask will be the 4th output band.
- **Validity Mask Upper Threshold**: Upper threshold to obtain a validity mask. This mask will be the 4th output band.
- **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 -ery 2 -mrx 3 -mry 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 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.

StereoFramework - 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 (WGS84 by default).

This application is chaining different processing steps. Some of them are also performed by other applications in the stereo-reco

- StereoRectificationGridGenerator [1] : for the generation of deformation grids
- GridBasedImageResampling [2] : resampling into epipolar geometry
- BlockMatching [3] : estimation of dense disparity maps

The pipeline executes the following steps on each stereo 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 all 3D maps are fused to produce DSM. The fusion method in each DEM cell can be chosen between maximum, minimum and average.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
input	Input parameters	Group
input.il	Input images list	Input image list
input.co	Couples list	String
input.channel	Input Image channel	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
output	Output parameters	Group
output.res	Output resolution	Float
output.nodata	NoData value	Float
output.fusionmethod	Method to fuse measures in each DSM cell	Choices
output.fusionmethod max	Maximum	*Choice*
output.fusionmethod min	Minimum	*Choice*
output.fusionmethod mean	Mean	*Choice*
output.fusionmethod acc	Accumulator	*Choice*
output.out	Output DSM	Output image
output.mode	Parameters estimation modes	Choices
output.mode fit	Fit to sensor image	*Choice*
output.mode user	User Defined	*Choice*
output.mode.user.ulx	Upper Left X	Float
output.mode.user.uly	Upper Left Y	Float
output.mode.user.sizex	Size X	Int
output.mode.user.sizey	Size Y	Int
output.mode.user.spacingx	Pixel Size X	Float
output.mode.user.spacingy	Pixel Size Y	Float
map	Map Projection	Choices
map utm	Universal Trans-Mercator (UTM)	*Choice*
map lambert2	Lambert II Etendu	*Choice*
map lambert93	Lambert93	*Choice*
map wgs	WGS 84	*Choice*
map epsg	EPSG Code	*Choice*
map.utm.zone	Zone number	Int
map.utm.northhem	Northern Hemisphere	Boolean
map.epsg.code	EPSG Code	Int
stereorect	Stereorectification Grid parameters	Group
stereorect.fwdgridstep	Step of the displacement grid (in pixels)	Int
stereorect.invgridssrate	Sub-sampling rate for epipolar grid inversion	Int
bm	Block matching parameters	Group
bm.metric	Block-matching metric	Choices
bm.metric ssdmean	Sum of Squared Distances divided by mean of block	*Choice*
bm.metric ssd	Sum of Squared Distances	*Choice*
bm.metric ncc	Normalized Cross-Correlation	*Choice*

Continued on next page

[1] Table: Parameters table for Stereo Framework.

Table 7.2 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
bm.metric lp	Lp pseudo-norm	*Choice*
bm.metric.lp.p	p value	Float
bm.radius	Correlation window radius (in pixels)	Int
bm.minhoffset	Minimum altitude offset (in meters)	Float
bm.maxhoffset	Maximum altitude offset (in meters)	Float
postproc	Postprocessing parameters	Group
postproc.bij	Use bijection consistency in block matching strategy	Boolean
postproc.med	Use median disparities filtering	Boolean
postproc.metrict	Correlation metric threshold	Float
mask	Masks	Group
mask.left	Input left mask	Input image
mask.right	Input right mask	Input image
mask.variancet	Discard pixels with low local variance	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

[Input parameters]: This group of parameters allows one to set input data.

- **Input images list**: List of images corresponding to multiple views on a single scene, in sensor geometry.
- **Couples list**: List of index of couples im 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 second 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).
- **Input Image channel**: Channel used for block matching (the same 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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:
- **Maximum**: The cell is filled with the maximum measured elevation values.

- **Minimum**: The cell is filled with the minimum measured elevation values.
- **Mean**: The cell is filled with the mean of measured elevation values.
- **Accumulator**: 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).

Map Projection: Defines the 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See 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 subsampling. 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 to 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**: Metric used to compute matching score. 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).
- **Correlation window radius (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, the disparity is rejected.
- **Use median disparities filtering**: Disparity map 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. Pixel with a null mask value are discarded.
- **Input right mask**: Mask for right input image. Pixel with a null mask value are discarded.
- **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 correlation window radius).

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 -stereorect.fwdgridstep 8 -stereorect.invgridssrate 4 -postproc.med 1 -
↪output.res 2.5 -output.out dem.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 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.

See Also

These additional resources can be useful for further information:

[1] StereoRectificationGridGenerator
[2] GridBasedImageResampling
[3] BlockMatching

StereoRectificationGridGenerator - Stereo-rectification deformation grid generator

Generates two deformation fields to 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.

This application is the first part of the stereo reconstruction framework. The output deformation grids can be passed to the GridBasedImageResampling application for actual resampling into epipolar geometry.

There are several ways to set the elevation source:

- An arbitrary constant elevation
- A DEM directory
- Compute an average elevation from a DEM

If needed, the application can compute inverse resampling grids (from epipolar to original sensor geometry). Don't forget to check the other outputs from the application. For instance, the application gives the X and Y size of the rectified images, along with an estimated baseline ratio.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
io	Input and output data	Group
io.inleft	Left input image	Input image
io.inright	Right input image	Input image
io.outleft	Left output deformation grid	Output image
io.outright	Right output deformation grid	Output image
epi	Epipolar geometry and grid parameters	Group
epi.elevation	Elevation management	Group
epi.elevation.dem	DEM directory	Directory
epi.elevation.geoid	Geoid File	Input File name
epi.elevation.default	Default elevation	Float
epi.elevation.avgdem	Average elevation computed from DEM	Group
epi.elevation.avgdem.step	Sub-sampling step	Int
epi.elevation.avgdem.value	Average elevation value	Float
epi.elevation.avgdem.mindisp	Minimum disparity from DEM	Float
epi.elevation.avgdem.maxdisp	Maximum disparity from DEM	Float
epi.scale	Scale of epipolar images	Float
epi.step	Step of the deformation grid (in nb. of pixels)	Int
epi.rectsizex	Rectified image size X	Int
epi.rectsizey	Rectified image size Y	Int
epi.baseline	Mean baseline ratio	Float
inverse	Write inverse fields	Group
inverse.outleft	Left inverse deformation grid	Output image
inverse.outright	Right inverse deformation grid	Output image
inverse.ssrate	Sub-sampling rate for inversion	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 image from the stereo pair, in sensor geometry.
- **Right input image**: The right image from the stereo pair, in sensor geometry.
- **Left output deformation grid**: The deformation grid to resample the left image from sensor geometry to epipolar geometry.
- **Right output deformation grid**: The deformation grid to resample the right image from sensor geometry to epipolar geometry.

[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.
    - **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.

[1] Table: Parameters table for Stereo-rectification deformation grid generator.

    - **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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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^-1, 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 deformation grid to resample the left image from the epipolar geometry back into its original sensor geometry.
- **Right inverse deformation grid**: The output deformation grid to resample the right image from the epipolar geometry back into its original sensor geometry.
- **Sub-sampling rate for inversion**: Grid inversion is an heavy process that implies spline regression on control points. To avoid eating to 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_
↪left.tif -io.outleft wv2_xs_left_epi_field.tif -io.outright wv2_xs_right_epi_field.
↪tif -epi.elevation.default 400
```

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

Geometry

BundleToPerfectSensor - Bundle to perfect sensor

Perform P+XS pansharpening

Detailed description

This application performs P+XS pansharpening. 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[1] 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 Name	Parameter Type
inp	Input PAN Image	Input image
inxs	Input XS Image	Input image
out	Output image	Output image
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
mode	Mode	Choices
mode default	Default mode	*Choice*
mode phr	Pleiades mode	*Choice*
method	Algorithm	Choices
method rcs	RCS	*Choice*
method lmvm	LMVM	*Choice*
method bayes	Bayesian	*Choice*
method.lmvm.radiusx	X radius	Int
method.lmvm.radiusy	Y radius	Int
method.bayes.lambda	Weight	Float
method.bayes.s	S coefficient	Float
lms	Spacing of the deformation field	Float
interpolator	Interpolation	Choices
interpolator bco	Bicubic interpolation	*Choice*
interpolator nn	Nearest Neighbor interpolation	*Choice*
interpolator linear	Linear interpolation	*Choice*
interpolator.bco.radius	Radius for bicubic interpolation	Int
fv	Fill Value	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] Table: Parameters table for Bundle to perfect sensor.

- **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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

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.

Spacing of the deformation field: Spacing of the deformation field. Default is 10 times the PAN image spacing.

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.

Fill Value: Fill value for area outside the reprojected 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_BundleToPerfectSensor -inp QB_Toulouse_Ortho_PAN.tif -inxs QB_Toulouse_Ortho_
↪XS.tif -out BundleToPerfectSensor.png uchar
```

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.SetParameterOutputImagePixelType("out", 1)

# The following line execute the application
BundleToPerfectSensor.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

ConvertCartoToGeoPoint - Cartographic to geographic coordinates conversion

Convert cartographic coordinates to geographic ones.

Detailed description

This application computes the geographic coordinates from cartographic ones. User has to give the X and Y coordinate and the cartographic projection (see mapproj parameter for details).

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConvertCartoToGeoPoint* .

[1] Table: Parameters table for Cartographic to geographic coordinates conversion.

Parameter Key	Parameter Name	Parameter Type
carto	Input cartographic coordinates	Group
carto.x	X cartographic coordinates	Float
carto.y	Y cartographic coordinates	Float
mapproj	Map Projection	Choices
mapproj utm	Universal Trans-Mercator (UTM)	*Choice*
mapproj lambert2	Lambert II Etendu	*Choice*
mapproj lambert93	Lambert93	*Choice*
mapproj wgs	WGS 84	*Choice*
mapproj epsg	EPSG Code	*Choice*
mapproj.utm.zone	Zone number	Int
mapproj.utm.northhem	Northern Hemisphere	Boolean
mapproj.epsg.code	EPSG Code	Int
long	Output long	Float
lat	Output lat	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

[Input cartographic coordinates]

- **X cartographic coordinates**: X cartographic coordinates in the projection defined by mapproj parameter.
- **Y cartographic coordinates**: Y cartographic coordinates in the projection defined by mapproj parameter.

Map Projection: Defines the 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See www.spatialreference.org to find which EPSG code is associated to your projection.

Output long: Point longitude coordinates in decimal degrees.

Output lat: Point latitude coordinates in decimal degrees.

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 true -mapproj.utm.zone 31
```

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","true")

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.

ConvertSensorToGeoPoint - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConvertSensorToGeoPoint* .

[1] Table: Parameters table for Convert Sensor Point To Geographic Point.

Parameter Key	Parameter Name	Parameter Type
in	Sensor image	Input image
input	Point Coordinates	Group
input.idx	X value of desired point	Float
input.idy	Y value of desired point	Float
output	Geographic Coordinates	Group
output.idx	Output Point Longitude	Float
output.idy	Output Point Latitude	Float
output.town	Main town near the coordinates computed	String
output.country	Country of the image	String
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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

GeneratePlyFile - Ply 3D files generation

Generate a 3D Ply file from a DEM and a color image.

Detailed description

The application converts an image containing elevations into a PLY file, which is a file format to store 3D models. This format is adpated for visualization on software such as MeshLab [2] or CloudCompare [3]

This application is part of the stereo reconstruction framework. The input data can be produced by the application DisparityMapToElevationMap.

There are two types of supported input images:

- A DEM image, with a ground projection, containing elevation values. Each elevation value can be considered as a 3D point.
- A 3D grid image, containing 5 bands (the first 3 are the 3D coordinates of each point, the 5th is a validity mask where valid values are larger or equal to 1)

The user shall also give a support image that contains color values for each 3D point. The color values will be embedded in the PLY file.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GeneratePlyFile* .

[1] Table: Parameters table for Ply 3D files generation.

Parameter Key	Parameter Name	Parameter Type
indem	The input DEM image	Input image
mode	Conversion Mode	Choices
mode dem	DEM	*Choice*
mode 3dgrid	3D grid	*Choice*
map	Map Projection	Choices
map utm	Universal Trans-Mercator (UTM)	*Choice*
map lambert2	Lambert II Etendu	*Choice*
map lambert93	Lambert93	*Choice*
map wgs	WGS 84	*Choice*
map epsg	EPSG Code	*Choice*
map.utm.zone	Zone number	Int
map.utm.northhem	Northern Hemisphere	Boolean
map.epsg.code	EPSG Code	Int
incolor	The input color image	Input image
out	The output Ply file	Output File name
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

The input DEM image: The image should be either a projected DEM or a 3D grid containing 3D point coordinates and a validity mask.

Conversion Mode Available choices are:

- **DEM**: DEM conversion mode (the projection information of the DEM is used to derive the X and Y coordinates of each point).
- **3D grid**: 3D grid conversion mode.

Map Projection: Defines the 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See www.spatialreference.org to find which EPSG code is associated to your projection.

The input color image: If the color image has 4 bands it will be interpreted as Red, Green, Blue, NIR. In other cases, only the first one is used (gray scale colors). The color values are expected in the range 0 - 255, and will be embedded with each 3D point of the PLY file.

The output Ply file: The output Ply file will contain as many 3D points as pixels in the input DEM.

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()
```

Limitations

The input DEM image has to entirely fit into memory.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

- [1] DisparityMapToElevationMap
- [2] http://www.meshlab.net/
- [3] http://www.cloudcompare.org/

GenerateRPCSensorModel - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GenerateRPCSensorModel* .

Parameter Key	Parameter Name	Parameter Type
outgeom	Output geom file	Output File name
inpoints	Input file containing tie points	Input File name
outstat	Output file containing output precision statistics	Output File name
outvector	Output vector file with residues	Output File name
map	Map Projection	Choices
map utm	Universal Trans-Mercator (UTM)	*Choice*
map lambert2	Lambert II Etendu	*Choice*
map lambert93	Lambert93	*Choice*
map wgs	WGS 84	*Choice*
map epsg	EPSG Code	*Choice*
map.utm.zone	Zone number	Int
map.utm.northhem	Northern Hemisphere	Boolean
map.epsg.code	EPSG Code	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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. Spaced by a space or tab character. 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.

Map Projection: Defines the 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.

[1] Table: Parameters table for Generate a RPC 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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:

OrthoRectication,HomologousPointsExtraction,RefineSensorModel

GridBasedImageResampling - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *GridBasedImageResampling* .

[1] Table: Parameters table for Grid Based Image Resampling.

Parameter Key	Parameter Name	Parameter Type
io	Input and output data	Group
io.in	Input image	Input image
io.out	Output Image	Output image
grid	Resampling grid parameters	Group
grid.in	Input resampling grid	Input image
grid.type	Grid Type	Choices
grid.type def	Displacement grid: $G(x_out,y_out) = (x_in-x_out, y_in-y_out)$	*Choice*
grid.type loc	Localisation grid: $G(x_out,y_out) = (x_in, y_in)$	*Choice*
out	Output Image parameters	Group
out.ulx	Upper Left X	Float
out.uly	Upper Left Y	Float
out.sizex	Size X	Int
out.sizey	Size Y	Int
out.spacingx	Pixel Size X	Float
out.spacingy	Pixel Size Y	Float
out.default	Default value	Float
interpolator	Interpolation	Choices
interpolator nn	Nearest Neighbor interpolation	*Choice*
interpolator linear	Linear interpolation	*Choice*
interpolator bco	Bicubic interpolation	*Choice*
interpolator.bco.radius	Radius for bicubic interpolation	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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:
- **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_resampled.tif uint8 -grid.in ROI_IKO_PAN_LesHalles_sub_
↪deformation_field.tif -out.sizex 256 -out.sizey 256 -grid.type def
```

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

ImageEnvelope - Image Envelope

Extracts an image envelope.

Detailed description

Build a vector data containing the image envelope polygon. Useful for some projection, you can set the polygon with more points with the sr parameter. This filter supports user-specified output projection. If no projection is defined, the standard WGS84 projection will be used.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Vector Data	Output vector data
sr	Sampling Rate	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
proj	Projection	String
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: Input image filename.

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.

[1] Table: Parameters table for Image Envelope.

- **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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

OrthoRectification - Ortho-rectification

This application allows ortho-rectifying optical and radar images from supported sensors.

Detailed description

This application uses inverse sensor modelling combined with a choice of interpolation functions to resample a sensor geometry image into a ground geometry regular grid. The ground geometry regular grid is defined with respect to a map projection (see map parameter). The application offers several modes to estimate the output grid parameters (origin and ground sampling distance), including automatic estimation of image size, ground sampling distance, or both, from image metadata, user-defined ROI corners, or another ortho-image.A digital Elevation Model along with a geoid file 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[1] 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 Name	Parameter Type
io	Input and output data	Group
io.in	Input Image	Input image
io.out	Output Image	Output image
map	Map Projection	Choices
map utm	Universal Trans-Mercator (UTM)	*Choice*
map lambert2	Lambert II Etendu	*Choice*
map lambert93	Lambert93	*Choice*
map wgs	WGS 84	*Choice*
map epsg	EPSG Code	*Choice*
map.utm.zone	Zone number	Int
map.utm.northhem	Northern Hemisphere	Boolean
map.epsg.code	EPSG Code	Int
outputs	Output Image Grid	Group
outputs.mode	Parameters estimation modes	Choices
outputs.mode auto	User Defined	*Choice*
outputs.mode autosize	Automatic Size from Spacing	*Choice*
outputs.mode autospacing	Automatic Spacing from Size	*Choice*
outputs.mode outputroi	Automatic Size from Spacing and output corners	*Choice*
outputs.mode orthofit	Fit to ortho	*Choice*
outputs.ulx	Upper Left X	Float
outputs.uly	Upper Left Y	Float
outputs.sizex	Size X	Int
outputs.sizey	Size Y	Int
outputs.spacingx	Pixel Size X	Float
outputs.spacingy	Pixel Size Y	Float
outputs.lrx	Lower right X	Float
outputs.lry	Lower right Y	Float
outputs.ortho	Model ortho-image	Input image
outputs.isotropic	Force isotropic spacing by default	Boolean
outputs.default	Default pixel value	Float
elev	Elevation management	Group
elev.dem	DEM directory	Directory

Continued on next page

[1] Table: Parameters table for Ortho-rectification.

Table 7.3 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
interpolator	Interpolation	Choices
interpolator bco	Bicubic interpolation	*Choice*
interpolator nn	Nearest Neighbor interpolation	*Choice*
interpolator linear	Linear interpolation	*Choice*
interpolator.bco.radius	Radius for bicubic interpolation	Int
opt	Speed optimization parameters	Group
opt.rpc	RPC modeling (points per axis)	Int
opt.ram	Available RAM (Mb)	Int
opt.gridspacing	Resampling grid spacing	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

Map Projection: Defines the 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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 (both optical and radar) are: GeoEye, Ikonos, Pleiades, Quickbird, RadarSat, Sentinel-1, SPOT5 (TIF forma
Also note that the opt.gridspacing default value may not be suitable for all sensors. In particular, if this value is lower than the target ground sampling distance, the processing time may increase a lot. A warning is issued in this case. Typical values should be half the DEM ground sampling distance.

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

Pansharpening - 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[1] 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 Name	Parameter Type
inp	Input PAN Image	Input image
inxs	Input XS Image	Input image
out	Output image	Output image
method	Algorithm	Choices
method rcs	RCS	*Choice*
method lmvm	LMVM	*Choice*
method bayes	Bayesian	*Choice*
method.lmvm.radiusx	X radius	Int
method.lmvm.radiusy	Y radius	Int
method.bayes.lambda	Weight	Float
method.bayes.s	S coefficient	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] 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 uint16
```

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.

RefineSensorModel - 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. Plaease 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[1] 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 Name	Parameter Type
ingeom	Input geom file	Input File name
outgeom	Output geom file	Output File name
inpoints	Input file containing tie points	Input File name
outstat	Output file containing output precision statistics	Output File name
outvector	Output vector file with residues	Output File name
map	Map Projection	Choices
map utm	Universal Trans-Mercator (UTM)	*Choice*
map lambert2	Lambert II Etendu	*Choice*
map lambert93	Lambert93	*Choice*
map wgs	WGS 84	*Choice*
map epsg	EPSG Code	*Choice*
map.utm.zone	Zone number	Int
map.utm.northhem	Northern Hemisphere	Boolean
map.epsg.code	EPSG Code	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

Map Projection: Defines the 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.

[1] 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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 -ingeom input.geom -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 RefineSensorModel application
RefineSensorModel = otbApplication.Registry.CreateApplication("RefineSensorModel")

# The following lines set all the application parameters:
RefineSensorModel.SetParameterString("ingeom", "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

RigidTransformResample - 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 form output space to input space).

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *RigidTransformResample* .

[1] Table: Parameters table for Image resampling with a rigid transform.

Parameter Key	Parameter Name	Parameter Type
in	Input image	Input image
out	Output image	Output image
transform	Transform parameters	Group
transform.type	Type of transformation	Choices
transform.type id	id	*Choice*
transform.type translation	translation	*Choice*
transform.type rotation	rotation	*Choice*
transform.type.id.scalex	X scaling	Float
transform.type.id.scaley	Y scaling	Float
transform.type.translation.tx	The X translation (in physical units)	Float
transform.type.translation.ty	The Y translation (in physical units)	Float
transform.type.translation.scalex	X scaling	Float
transform.type.translation.scaley	Y scaling	Float
transform.type.rotation.angle	Rotation angle	Float
transform.type.rotation.scalex	X scaling	Float
transform.type.rotation.scaley	Y scaling	Float
interpolator	Interpolation	Choices
interpolator nn	Nearest Neighbor interpolation	*Choice*
interpolator linear	Linear interpolation	*Choice*
interpolator bco	Bicubic interpolation	*Choice*
interpolator.bco.radius	Radius for bicubic interpolation	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 rigitTransformImage.tif -
↪transform.type rotation -transform.type.rotation.angle 20 -transform.type.rotation.
↪scalex 2. -transform.type.rotation.scaley 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 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", "rigitTransformImage.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

Superimpose - 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[1] 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 Name	Parameter Type
inr	Reference input	Input image
inm	The image to reproject	Input image
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
lms	Spacing of the deformation field	Float
fv	Fill Value	Float
out	Output image	Output image
mode	Mode	Choices
mode default	Default mode	*Choice*
mode phr	Pleiades mode	*Choice*
interpolator	Interpolation	Choices
interpolator bco	Bicubic interpolation	*Choice*
interpolator nn	Nearest Neighbor interpolation	*Choice*
interpolator linear	Linear interpolation	*Choice*
interpolator.bco.radius	Radius for bicubic interpolation	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

Fill Value: Fill value for area outside the reprojected image.

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_to_PAN.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 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.

Learning

ClassificationMapRegularization - Classification Map Regularization

Filters the input labeled image using Majority Voting in a ball shaped neighbordhood.

Detailed description

This application filters the input labeled image (with a maximal class label = 65535) using Majority Voting in a ball shaped neigh
-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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ClassificationMapRegularization* .

[1] Table: Parameters table for Classification Map Regularization.

Parameter Key	Parameter Name	Parameter Type
io	Input and output images	Group
io.in	Input classification image	Input image
io.out	Output regularized image	Output image
ip	Regularization parameters	Group
ip.radius	Structuring element radius (in pixels)	Int
ip.suvbool	Multiple majority: Undecided(X)/Original	Boolean
ip.nodatalabel	Label for the NoData class	Int
ip.undecidedlabel	Label for the Undecided class	Int
ip.onlyisolatedpixels	Process isolated pixels only	Boolean
ip.isolatedthreshold	Threshold for isolated pixels	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 neighbordhood 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:

```
otbcli_ClassificationMapRegularization -io.in clLabeledImageQB123_1.tif -io.out␣
↪clLabeledImageQB123_1_CMR_r2_nodl_10_undl_7.tif -ip.radius 2 -ip.suvbool true -ip.
↪onlyisolatedpixels true -ip.nodatalabel 10 -ip.undecidedlabel 7
```

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_nodl_10_undl_7.tif")

ClassificationMapRegularization.SetParameterInt("ip.radius", 2)

ClassificationMapRegularization.SetParameterString("ip.suvbool","true")

ClassificationMapRegularization.SetParameterString("ip.onlyisolatedpixels","true")

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.

ComputeConfusionMatrix - 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Matrix output	Output File name
format	set the output format to contingency table or confusion matrix	Choices
format confusionmatrix	Choice of a confusion matrix as output.	*Choice*
format contingencytable	Choice of a contingency table as output.	*Choice*
ref	Ground truth	Choices
ref raster	Ground truth as a raster image	*Choice*
ref vector	Ground truth as a vector data file	*Choice*
ref.raster.in	Input reference image	Input image
ref.raster.nodata	Value for nodata pixels in ref raster	Int
ref.vector.in	Input reference vector data	Input File name
ref.vector.field	Field name	List
ref.vector.nodata	Value for nodata pixels in ref vector	Int
nodatalabel	Value for nodata pixels in input image	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: The input classification image.

Matrix output: Filename to store the output matrix (csv format).

set the output format to contingency table or confusion matrix: Choice of the output format as a contingency table for unsupervised algorithmsor confusion matrix for supervised ones. Available choices are:

- **Choice of a confusion matrix as output.**
- **Choice of a contingency table as output.**

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.
- **Value for nodata pixels in ref raster**: Label to be treated as no data in ref raster.

[1] Table: Parameters table for Confusion matrix Computation.

- **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 in ref vector**: Label to be treated as no data in ref vector. Please note that this value is always used in vector mode, to generate default values. Please set it to a value that does not correspond to a class label.

Value for nodata pixels in input image: Label to be treated as no data in input 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_ComputeConfusionMatrix -in clLabeledImageQB1.tif -out ConfusionMatrix.csv -ref␣
↪vector -ref.vector.in VectorData_QB1_bis.shp -ref.vector.field Class -ref.vector.
↪nodata 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 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")

# The following line execute the application
ComputeConfusionMatrix.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

ComputeImagesStatistics - 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 an input for the TrainImagesClassifier application to normalize samples before learning. You can also normalize the image with the XML file in the ImageClassifier application.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ComputeImagesStatistics* .

Parameter Key	Parameter Name	Parameter Type
il	Input images	Input image list
bv	Background Value	Float
out	Output XML file	Output File name
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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")
```

[1] Table: Parameters table for Compute Images second order statistics.

```
# 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 and ImageClassifier application.

FusionOfClassifications - 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 confusion 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *FusionOfClassifications* .

[1] Table: Parameters table for Fusion of Classifications.

Parameter Key	Parameter Name	Parameter Type
il	Input classifications	Input image list
method	Fusion method	Choices
method majorityvoting	Majority Voting	*Choice*
method dempstershafer	Dempster Shafer combination	*Choice*
method.dempstershafer.cmfl	Confusion Matrices	Input File name list
method.dempstershafer.mob	Mass of belief measurement	Choices
method.dempstershafer.mob precision	Precision	*Choice*
method.dempstershafer.mob recall	Recall	*Choice*
method.dempstershafer.mob accuracy	Overall Accuracy	*Choice*
method.dempstershafer.mob kappa	Kappa	*Choice*
nodatalabel	Label for the NoData class	Int
undecidedlabel	Label for the Undecided class	Int
out	The output classification image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

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 -method dempstershafer -method.dempstershafer.cmfl␣
↪classification1.csv classification2.csv classification3.csv -method.dempstershafer.
↪mob precision -nodatalabel 0 -undecidedlabel 10 -out classification_fused.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 FusionOfClassifications application
FusionOfClassifications = otbApplication.Registry.CreateApplication(
↪"FusionOfClassifications")

# The following lines set all the application parameters:
FusionOfClassifications.SetParameterStringList("il", ['classification1.tif',
↪'classification2.tif', 'classification3.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

ImageClassifier - 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. By default, 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ImageClassifier* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
mask	Input Mask	Input image
model	Model file	Input File name
imstat	Statistics file	Input File name
nodatalabel	Label mask value	Int
out	Output Image	Output image
confmap	Confidence map	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).
- **Label mask value**: By default, hidden pixels will have the assigned label 0 in the output image. It's possible to define the label mask by another value, but be careful to not take a label from another class (max. 65535).
- **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) .

[1] Table: Parameters table for Image Classification.

- **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.svm -out clLabeledImageQB1.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 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

ImageDimensionalityReduction - Image Dimensionality Reduction

Performs dimensionality reduction of the input image according to a dimensionality reduction model file.

Detailed description

This application reduces the dimension of an input image, based on a machine learning model file produced by the TrainDimensionalityReduction application. Pixels of the output image will contain the reduced values fromthe model. The input pixels can be optionally centered and reduced according to the statistics file produced by the ComputeImagesStatistics application.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ImageDimensionalityReduction* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
mask	Input Mask	Input image
model	Model file	Input File name
imstat	Statistics file	Input File name
out	Output Image	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 dimensionality reduction 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 containsone more bands than the sample size, the last stat of last band will beapplied to expand the output predicted value.
- **Output Image**: Output image containing reduced 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_ImageDimensionalityReduction -in QB_1_ortho.tif -imstat␣
↪EstimateImageStatisticsQB1.xml -model clsvmModelQB1.model -out ReducedImageQB1.tif
```

To run this example from Python, use the following code snippet:

[1] Table: Parameters table for Image Dimensionality Reduction.

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ImageDimensionalityReduction␣
↪application
ImageDimensionalityReduction = otbApplication.Registry.CreateApplication(
↪"ImageDimensionalityReduction")

# The following lines set all the application parameters:
ImageDimensionalityReduction.SetParameterString("in", "QB_1_ortho.tif")

ImageDimensionalityReduction.SetParameterString("imstat", "EstimateImageStatisticsQB1.
↪xml")

ImageDimensionalityReduction.SetParameterString("model", "clsvmModelQB1.model")

ImageDimensionalityReduction.SetParameterString("out", "ReducedImageQB1.tif")

# The following line execute the application
ImageDimensionalityReduction.ExecuteAndWriteOutput()
```

Limitations

The input image must contain the feature bands used for the model training. If a statistics file was used during training by the Training application, it is mandatory to use the same statistics file for reduction.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

TrainDimensionalityReduction, ComputeImagesStatistics

KMeansClassification - Unsupervised KMeans image classification

Unsupervised KMeans image classification

Detailed description

Performs unsupervised KMeans image classification.KMeansClassification is a composite application, using an existing training and classification application.The SharkKMeans model is used. KMeansClassification application is only available if OTB is compiled with Shark support(CMake option OTB_USE_SHARK=ON) The steps of this composite application : 1) ImageEnveloppe : create a shapefile (1 polygon), 2) PolygonClassStatistics : compute the statistics, 3) SampleSelection : select the samples by constant strategy in the shapefile (1000000 samples max), 4) SamplesExtraction : extract the samples descriptors (update of SampleSelection output file), 5) ComputeImagesStatistics : compute

images second order statistics, 6) TrainVectorClassifier : train the SharkKMeans model, 7) ImageClassifier : performs the classification of the input image according to a model file.

It's possible to choice random/periodic modes of the SampleSelection application. If you want keep the temporary files (sample selected, model file, ...), initialize cleanup parameter. For more information on shark KMeans algorithm [1].

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *KMeansClassification* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
nc	Number of classes	Int
ts	Training set size	Int
maxit	Maximum number of iterations	Int
outmeans	Centroid filename	Output File name
ram	Available RAM (Mb)	Int
sampler	Sampler type	Choices
sampler periodic	Periodic sampler	*Choice*
sampler random	Random sampler	*Choice*
sampler.periodic.jitter	Jitter amplitude	Int
vm	Validity Mask	Input image
nodatalabel	Label mask value	Int
cleanup	Temporary files cleaning	Boolean
rand	set user defined seed	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: Input image filename.

Output Image: Output image containing class labels.

Number of classes: Number of modes, which will be used to generate class membership.

Training set size: Size of the training set (in pixels).

Maximum number of iterations: Maximum number of iterations for the learning step.

Centroid filename: Output text file containing centroid positions.

Available RAM (Mb): Available memory for processing (in MB).

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.

Validity Mask: Validity mask, only non-zero pixels will be used to estimate KMeans modes.

Label mask value: By default, hidden pixels will have the assigned label 0 in the output image. It's possible to define the label mask by another value, but be careful to not take a label from another class. This application initialize the labels from 0 to N-1, N is the number of class (defined by 'nc' parameter).

[1] Table: Parameters table for Unsupervised KMeans image classification.

Temporary files cleaning: If activated, the application will try to clean all temporary files it created.

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 -out␣
↪ClassificationFilterOutput.tif uint8
```

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.SetParameterString("out", "ClassificationFilterOutput.tif")
KMeansClassification.SetParameterOutputImagePixelType("out", 1)

# The following line execute the application
KMeansClassification.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

ImageEnveloppe PolygonClassStatistics SampleSelection SamplesExtraction PolygonClassStatistics TrainVectorClassifier ImageClassifier

[1] http://image.diku.dk/shark/sphinx_pages/build/html/rest_sources/tutorials/algorithms/kmeans.html

MultiImageSamplingRate - 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 Ni(c) = M * Ti(c) / sum_k(Tk(c))
  - if mim = equal , then Ni(c) = M / L
  - if mim = custom , then Ni(c) = Mi where Mi 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 Ni(c) = M(c) * Ti(c) / sum_k(Tk(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_kl(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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MultiImageSamplingRate* .

Parameter Key	Parameter Name	Parameter Type
il	Input statistics	Input File name list
out	Output sampling rates	Output File name
strategy	Sampling strategy	Choices
strategy byclass	Set samples count for each class	*Choice*
strategy constant	Set the same samples counts for all classes	*Choice*
strategy smallest	Set same number of samples for all classes, with the smallest class fully sampled	*Choice*
strategy percent	Use a percentage of the samples available for each class	*Choice*
strategy total	Set the total number of samples to generate, and use class proportions.	*Choice*
strategy all	Take all samples	*Choice*
strategy.byclass.in	Number of samples by class	Input File name list
strategy.constant.nb	Number of samples for all classes	String
strategy.percent.p	The percentage(s) to use	String
strategy.total.v	The number of samples to generate	String
mim	Multi-Image Mode	Choices
mim proportional	Proportional	*Choice*
mim equal	equal	*Choice*
mim custom	Custom	*Choice*
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input statistics: List of statistics files for each input image.

[1] Table: Parameters table for Multi-image sampling rate estimation.

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 generateIn 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 proportional
```

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.

PolygonClassStatistics - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *PolygonClassStatistics* .

[1] Table: Parameters table for Polygon Class Statistics.

Parameter Key	Parameter Name	Parameter Type
in	Input image	Input image
mask	Input validity mask	Input image
vec	Input vectors	Input File name
out	Output XML statistics file	Output File name
field	Field Name	List
layer	Layer Index	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input image: Support image that will be classified.

Input validity mask: Validity mask (only pixels corresponding to a mask value greater than 0 will be used for statistics).

Input vectors: Input geometries to analyze.

Output XML statistics file: 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.

[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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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_PolygonClassStatistics -in support_image.tif -vec variousVectors.sqlite -field␣
↪label -out polygonStat.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 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")

# The following line execute the application
PolygonClassStatistics.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

PredictRegression - 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 TrainRegression application. Pixels of the output image will contain the predicted values fromthe 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *PredictRegression* .

[1] Table: Parameters table for Predict Regression.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
mask	Input Mask	Input image
model	Model file	Input File name
imstat	Statistics file	Input File name
out	Output Image	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 containsone more band than the sample size, the last stat of last band will beapplied 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 -out clLabeledImageQB1.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 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")

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

SOMClassification - 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[1] 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 Name	Parameter Type
in	InputImage	Input image
out	OutputImage	Output image
vm	ValidityMask	Input image
tp	TrainingProbability	Float
ts	TrainingSetSize	Int
som	SOM Map	Output image
sx	SizeX	Int
sy	SizeY	Int
nx	NeighborhoodX	Int
ny	NeighborhoodY	Int
ni	NumberIteration	Int
bi	BetaInit	Float
bf	BetaFinal	Float
iv	InitialValue	Float
ram	Available RAM (Mb)	Int
rand	set user defined seed	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

[1] Table: Parameters table for SOM Classification.

- **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.
- **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 -nx 10 -ny 10 -ni 5 -bi 1.0 -bf 0.1 -iv 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 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.

SampleAugmentation - Sample Augmentation

Generates synthetic samples from a sample data file.

Detailed description

The application takes a sample data file as generated by the SampleExtraction application and generates synthetic samples to increase the number of available samples.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SampleAugmentation* .

[1] Table: Parameters table for Sample Augmentation.

Parameter Key	Parameter Name	Parameter Type
in	Input samples	Input File name
out	Output samples	Output File name
field	Field Name	List
layer	Layer Index	Int
label	Label of the class to be augmented	Int
samples	Number of generated samples	Int
exclude	Field names for excluded features.	List
strategy	Augmentation strategy	Choices
strategy replicate	Replicate input samples	*Choice*
strategy jitter	Jitter input samples	*Choice*
strategy smote	Smote input samples	*Choice*
strategy.jitter.stdfactor	Factor for dividing the standard deviation of each feature	Float
strategy.smote.neighbors	Number of nearest neighbors.	Int
seed	set user defined seed	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input samples: Vector data file containing samples (OGR format).

Output samples: Output vector data file storing new samples(OGR 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.

Label of the class to be augmented: Label of the class of the input file for which new samples will be generated.

Number of generated samples: Number of synthetic samples that will be generated.

Field names for excluded features.: List of field names in the input vector data that will not be generated in the output file.

Augmentation strategy Available choices are:

- **Replicate input samples**: The new samples are generated by replicating input samples which are randomly selected with replacement.
- **Jitter input samples**: The new samples are generated by adding gaussian noise to input samples which are randomly selected with replacement.
- **Factor for dividing the standard deviation of each feature**: The noise added to the input samples will have the standard deviation of the input features divided by the value of this parameter. .
- **Smote input samples**: The new samples are generated by using the SMOTE algorithm (http://dx.doi.org/10.1613/jair.953) on input samples which are randomly selected with replacement.
- **Number of nearest neighbors.**: Number of nearest neighbors to be used in the SMOTE algorithm.

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_SampleAugmentation -in samples.sqlite -field class -label 3 -samples 100 -out␣
↪augmented_samples.sqlite -exclude OGC_FID name class originfid -strategy smote -
↪strategy.smote.neighbors 5
```

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 SampleAugmentation application
SampleAugmentation = otbApplication.Registry.CreateApplication("SampleAugmentation")

# The following lines set all the application parameters:
SampleAugmentation.SetParameterString("in", "samples.sqlite")

# The following line execute the application
SampleAugmentation.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

SampleExtraction - Sample Extraction

Extracts samples values from an image.

Detailed description

The application extracts samples values from animage using positions contained in a vector data file.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SampleExtraction* .

[1] Table: Parameters table for Sample Extraction.

Parameter Key	Parameter Name	Parameter Type
in	InputImage	Input image
vec	Input sampling positions	Input File name
out	Output samples	Output File name
outfield	Output field names	Choices
outfield prefix	Use a prefix and an incremental counter	*Choice*
outfield list	Use the given name list	*Choice*
outfield.prefix.name	Output field prefix	String
outfield.list.names	Output field names	String list
field	Field Name	List
layer	Layer Index	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 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.

Example

To run this example in command-line, use the following:

```
otbcli_SampleExtraction -in support_image.tif -vec sample_positions.sqlite -outfield␣
↪prefix -outfield.prefix.name band_ -field label -out sample_values.sqlite
```

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_")

# The following line execute the application
SampleExtraction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

SampleSelection - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SampleSelection* .

Parameter Key	Parameter Name	Parameter Type
in	InputImage	Input image
mask	InputMask	Input image
vec	Input vectors	Input File name
out	Output vectors	Output File name
instats	Input Statistics	Input File name
outrates	Output rates	Output File name
sampler	Sampler type	Choices
sampler periodic	Periodic sampler	*Choice*
sampler random	Random sampler	*Choice*
sampler.periodic.jitter	Jitter amplitude	Int
strategy	Sampling strategy	Choices
strategy byclass	Set samples count for each class	*Choice*
strategy constant	Set the same samples counts for all classes	*Choice*
strategy percent	Use a percentage of the samples available for each class	*Choice*
strategy total	Set the total number of samples to generate, and use class proportions.	*Choice*
strategy smallest	Set same number of samples for all classes, with the smallest class fully sampled	*Choice*
strategy all	Take all samples	*Choice*
strategy.byclass.in	Number of samples by class	Input File name
strategy.constant.nb	Number of samples for all classes	Int
strategy.percent.p	The percentage to use	Float
strategy.total.v	The number of samples to generate	Int
field	Field Name	List
layer	Layer Index	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
ram	Available RAM (Mb)	Int
rand	set user defined seed	Int

Continued on next pa

[1] Table: Parameters table for Sample Selection.

Table 7.4 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
inxml	Load otb application from xml file	XML input parameters fi
outxml	Save otb application to xml file	XML output parameters

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.

[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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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).

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 apTvClPolygonClassStatisticsOut.xml -out resampledVectors.sqlite
```

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")

# The following line execute the application
SampleSelection.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

TrainDimensionalityReduction - Train Dimensionality Reduction

Train a dimensionality reduction model

Detailed description

Trainer for dimensionality reduction algorithms (autoencoders, PCA, SOM). All input samples are used to compute the model, like other machine learning models. The model can be used in the ImageDimensionalityReduction and VectorDimensionalityReduction applications.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TrainDimensionalityReduction* .

Parameter Key	Parameter Name	Parameter Type
io	Input and output data	Group
io.vd	Input Vector Data	Input vector data
io.out	Output model	Output File name
io.stats	Input XML image statistics file	Input File name
feat	Field names to be used for training.	String list
algorithm	algorithm to use for the training	Choices
algorithm som	OTB SOM	*Choice*
algorithm autoencoder	Shark Autoencoder	*Choice*
algorithm pca	Shark PCA	*Choice*
algorithm.som.s	Map size	String list
algorithm.som.n	Neighborhood sizes	String list
algorithm.som.ni	NumberIteration	Int
algorithm.som.bi	BetaInit	Float
algorithm.som.bf	BetaFinal	Float
algorithm.som.iv	InitialValue	Float
algorithm.autoencoder.nbiter	Maximum number of iterations during training	Int
algorithm.autoencoder.nbiterfinetuning	Maximum number of iterations during training	Int
algorithm.autoencoder.epsilon	Epsilon	Float
algorithm.autoencoder.initfactor	Weight initialization factor	Float
algorithm.autoencoder.nbneuron	Size	String list
algorithm.autoencoder.regularization	Strength of the regularization	String list
algorithm.autoencoder.noise	Strength of the noise	String list
algorithm.autoencoder.rho	Sparsity parameter	String list
algorithm.autoencoder.beta	Sparsity regularization strength	String list
algorithm.autoencoder.learningcurve	Learning curve	Output File name
algorithm.pca.dim	Dimension of the output of the pca transformation	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

[Input and output data]: This group of parameters allows setting input and output data.

[1] Table: Parameters table for Train Dimensionality Reduction.

- **Input Vector Data**: Input geometries used for training (note : all geometries from the layer will be used).
- **Output model**: Output file containing the estimated model (.txt format).
- **Input XML image statistics file**: XML file containing mean and variance of each feature.

Field names to be used for training.: List of field names in the input vector data used as features for training.

algorithm to use for the training: Choice of the dimensionality reduction algorithm to use for the training. Available choices are:

- **OTB SOM**: This group of parameters allows setting SOM parameters. .
- **Map size**: Sizes of the SOM map (one per dimension). For instance, [12;15] means a 2D map of size 12x15. Support2D to 5D maps.
- **Neighborhood sizes**: Sizes of the initial neighborhood in the SOM map (one per dimension). The number of sizes should be the same as the map sizes.
- **NumberIteration**: Number of iterations for SOM learning.
- **BetaInit**: Initial learning coefficient.
- **BetaFinal**: Final learning coefficient.
- **InitialValue**: Maximum initial neuron weight.
- **Shark Autoencoder**: This group of parameters allows setting Shark autoencoder parameters. .
- **Maximum number of iterations during training**: The maximum number of iterations used during training.
- **Maximum number of iterations during training**: The maximum number of iterations used during fine tuning of the whole network.
- **Epsilon**: Epsilon.
- **Weight initialization factor**: Parameter that control the weight initialization of the autoencoder.
- **Size**: The number of neurons in each hidden layer.
- **Strength of the regularization**: Strength of the L2 regularization used during training.
- **Strength of the noise**: Strength of the noise.
- **Sparsity parameter**: Sparsity parameter.
- **Sparsity regularization strength**: Sparsity regularization strength.
- **Learning curve**: Learning error values.
- **Shark PCA**: This group of parameters allows setting Shark PCA parameters. .
- **Dimension of the output of the pca transformation**: Dimension of the output of the pca transformation.

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_TrainDimensionalityReduction -io.vd cuprite_samples.sqlite -io.out mode.ae -
↪algorithm pca -algorithm.pca.dim 8 -feat value_0 value_1 value_2 value_3 value_4␣
↪value_5 value_6 value_7 value_8 value_9
```

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 TrainDimensionalityReduction␣
↪application
TrainDimensionalityReduction = otbApplication.Registry.CreateApplication(
↪"TrainDimensionalityReduction")

# The following lines set all the application parameters:
TrainDimensionalityReduction.SetParameterString("io.vd", "cuprite_samples.sqlite")

TrainDimensionalityReduction.SetParameterString("io.out", "mode.ae")

TrainDimensionalityReduction.SetParameterString("algorithm","pca")

TrainDimensionalityReduction.SetParameterInt("algorithm.pca.dim", 8)

TrainDimensionalityReduction.SetParameterStringList("feat", ['value_0', 'value_1',
↪'value_2', 'value_3', 'value_4', 'value_5', 'value_6', 'value_7', 'value_8', 'value_
↪9'])

# The following line execute the application
TrainDimensionalityReduction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

ImageDimensionalityReduction, VectorDimensionalityReduction

TrainImagesClassifier - 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[1] 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 Name
io	Input and output data
io.il	Input Image List
io.vd	Input Vector Data List
io.valid	Validation Vector Data List
io.imstat	Input XML image statistics file
io.out	Output model
io.confmatout	Output confusion matrix or contingency table
cleanup	Temporary files cleaning
sample	Training and validation samples parameters
sample.mt	Maximum training sample size per class
sample.mv	Maximum validation sample size per class
sample.bm	Bound sample number by minimum
sample.vtr	Training and validation sample ratio
sample.vfn	Field containing the class integer label for supervision
ram	Available RAM (Mb)
elev	Elevation management
elev.dem	DEM directory
elev.geoid	Geoid File
elev.default	Default elevation
classifier	Classifier to use for the training
classifier libsvm	LibSVM classifier
classifier boost	Boost classifier
classifier dt	Decision Tree classifier
classifier gbt	Gradient Boosted Tree classifier
classifier ann	Artificial Neural Network classifier
classifier bayes	Normal Bayes classifier
classifier rf	Random forests classifier
classifier knn	KNN classifier
classifier sharkrf	Shark Random forests classifier
classifier sharkkm	Shark kmeans classifier
classifier.libsvm.k	SVM Kernel Type
classifier.libsvm.k linear	Linear
classifier.libsvm.k rbf	Gaussian radial basis function
classifier.libsvm.k poly	Polynomial
classifier.libsvm.k sigmoid	Sigmoid

[1] Table: Parameters table for Train a classifier from multiple images.

Table 7.5 – continued from previous page

Parameter Key	Parameter Name
classifier.libsvm.m	SVM Model Type
classifier.libsvm.m csvc	C support vector classification
classifier.libsvm.m nusvc	Nu support vector classification
classifier.libsvm.m oneclass	Distribution estimation (One Class SVM)
classifier.libsvm.c	Cost parameter C
classifier.libsvm.nu	Cost parameter Nu
classifier.libsvm.opt	Parameters optimization
classifier.libsvm.prob	Probability estimation
classifier.boost.t	Boost Type
classifier.boost.t discrete	Discrete AdaBoost
classifier.boost.t real	Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)
classifier.boost.t logit	LogitBoost (technique producing good regression fits)
classifier.boost.t gentle	Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often g
classifier.boost.w	Weak count
classifier.boost.r	Weight Trim Rate
classifier.boost.m	Maximum depth of the tree
classifier.dt.max	Maximum depth of the tree
classifier.dt.min	Minimum number of samples in each node
classifier.dt.ra	Termination criteria for regression tree
classifier.dt.cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split
classifier.dt.f	K-fold cross-validations
classifier.dt.r	Set Use1seRule flag to false
classifier.dt.t	Set TruncatePrunedTree flag to false
classifier.gbt.w	Number of boosting algorithm iterations
classifier.gbt.s	Regularization parameter
classifier.gbt.p	Portion of the whole training set used for each algorithm iteration
classifier.gbt.max	Maximum depth of the tree
classifier.ann.t	Train Method Type
classifier.ann.t back	Back-propagation algorithm
classifier.ann.t reg	Resilient Back-propagation algorithm
classifier.ann.sizes	Number of neurons in each intermediate layer
classifier.ann.f	Neuron activation function type
classifier.ann.f ident	Identity function
classifier.ann.f sig	Symmetrical Sigmoid function
classifier.ann.f gau	Gaussian function (Not completely supported)
classifier.ann.a	Alpha parameter of the activation function
classifier.ann.b	Beta parameter of the activation function
classifier.ann.bpdw	Strength of the weight gradient term in the BACKPROP method
classifier.ann.bpms	Strength of the momentum term (the difference between weights on the 2 previous iterations)
classifier.ann.rdw	Initial value Delta_0 of update-values Delta_{ij} in RPROP method
classifier.ann.rdwm	Update-values lower limit Delta_{min} in RPROP method
classifier.ann.term	Termination criteria
classifier.ann.term iter	Maximum number of iterations
classifier.ann.term eps	Epsilon
classifier.ann.term all	Max. iterations + Epsilon
classifier.ann.eps	Epsilon value used in the Termination criteria
classifier.ann.iter	Maximum number of iterations used in the Termination criteria
classifier.rf.max	Maximum depth of the tree
classifier.rf.min	Minimum number of samples in each node

Table 7.5 – continued from previous page

Parameter Key	Parameter Name
classifier.rf.ra	Termination Criteria for regression tree
classifier.rf.cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split
classifier.rf.var	Size of the randomly selected subset of features at each tree node
classifier.rf.nbtrees	Maximum number of trees in the forest
classifier.rf.acc	Sufficient accuracy (OOB error)
classifier.knn.k	Number of Neighbors
classifier.sharkrf.nbtrees	Maximum number of trees in the forest
classifier.sharkrf.nodesize	Min size of the node for a split
classifier.sharkrf.mtry	Number of features tested at each node
classifier.sharkrf.oobr	Out of bound ratio
classifier.sharkkm.maxiter	Maximum number of iteration for the kmeans algorithm.
classifier.sharkkm.k	The number of class used for the kmeans algorithm.
rand	set user defined seed
inxml	Load otb application from xml file
outxml	Save otb application to xml 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.
- **Validation Vector Data List**: A list of vector data to select the validation samples.
- **Input XML image statistics file**: XML file containing mean and variance of each feature.
- **Output model**: Output file containing the model estimated (.txt format).
- **Output confusion matrix or contingency table**: Output file containing the confusion matrix or contingency table (.csv format).The contingency table is output when we unsupervised algorithms is used otherwise the confusion matrix is output.

Temporary files cleaning: If activated, the application will try to clean all temporary files it created.

[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).
- **Training and validation sample ratio**: Ratio between training and validation samples (0.0 = all training, 1.0 = all validation) (default = 0.5).
- **Field containing the class integer label for supervision**: Field containing the class id for supervision. The values in this field shall be cast into integers.

Available RAM (Mb): Available memory for processing (in MB).

[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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

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**: Linear Kernel, no mapping is done, this is the fastest option.
  - **Gaussian radial basis function**: This kernel is a good choice in most of the case. It is an exponential function of the euclidian distance between the vectors.
  - **Polynomial**: Polynomial Kernel, the mapping is a polynomial function.
  - **Sigmoid**: The kernel is a hyperbolic tangente function of the vectors.
  - **SVM Model Type**: Type of SVM formulation. Available choices are:
  - **C support vector classification**: This formulation allows imperfect separation of classes. The penalty is set through the cost parameter C.
  - **Nu support vector classification**: This formulation allows imperfect separation of classes. The penalty is set through the cost parameter Nu. As compared to C, Nu is harder to optimize, and may not be as fast.
  - **Distribution estimation (One Class SVM)**: All the training data are from the same class, SVM builds a boundary that separates the class from the rest of the feature space.
  - **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.
  - **Cost parameter Nu**: Cost parameter Nu, in the range 0..1, the larger the value, the smoother the decision.
  - **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**: This procedure trains the classifiers on weighted versions of the training sample, giving higher weight to cases that are currently misclassified. This is done for a sequence of weighter samples, and then the final classifier is defined as a linear combination of the classifier from each stage.
  - **Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)**: Adaptation of the Discrete Adaboost algorithm with Real value.

    - **LogitBoost (technique producing good regression fits)**: This procedure is an adaptive Newton algorithm for fitting an additive logistic regression model. Beware it can produce numeric instability.
    - **Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often good with regression data)**: A modified version of the Real Adaboost algorithm, using Newton stepping rather than exact optimization at each step.
    - **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 <= (1 - 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}.
- **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 the number of samples in a node is smaller than this parameter, then this node will not be split.
- **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 further.
- **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.
- **K-fold cross-validations**: If cv_folds > 1, then it prunes a tree with K-fold cross-validation where K is equal to cv_folds.
- **Set Use1seRule 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}.
- **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}.
    - **Train Method Type**: Type of training method for the multilayer perceptron (MLP) neural network. Available choices are:

- **Back-propagation algorithm**: Method to compute the gradient of the loss function and adjust weights in the network to optimize the result.
- **Resilient Back-propagation algorithm**: Almost the same as the Back-prop algorithm except that it does not take into account the magnitude of the partial derivative (coordinate of the gradient) but only its sign.
- **Number of neurons in each intermediate layer**: The number of neurons in each intermediate layer (excluding input and output layers).
- **Neuron activation function type**: This function determine whether the output of the node is positive or not depending on the output of the transfert 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**: Set the number of iterations allowed to the network for its training. Training will stop regardless of the result when this number is reached.
- **Epsilon**: Training will focus on result and will stop once the precision isat most epsilon.
- **Max. iterations + Epsilon**: Both termination criteria are used. Training stop at the first reached.
- **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}.
- **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}.
- **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}.
- **Number of Neighbors**: The number of neighbors to use.
- **Shark Random forests classifier**: This group of parameters allows setting Shark Random Forests classifier parameters. See complete documentation here url{http://image.diku.dk/shark/doxygen_pages/html/classshark_1_1_r_f_trainer.html}. It is noteworthy that training is parallel.
- **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.
- **Min size of the node for a split**: 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.
- **Number of features tested at each node**: The number of features (variables) which will be tested at each node in order to compute the split. If set to zero, the square root of the number of features is used.
- **Out of bound ratio**: Set the fraction of the original training dataset to use as the out of bag sample.A good default value is 0.66. .
- **Shark kmeans classifier**: This group of parameters allows setting Shark kMeans classifier parameters. See complete documentation here url{http://image.diku.dk/shark/sphinx_pages/build/html/rest_sources/tutorials/algorithms/kmeans.html}. .
- **Maximum number of iteration for the kmeans algorithm.**: The maximum number of iteration for the kmeans algorithm. 0=unlimited.
- **The number of class used for the kmeans algorithm.**: The number of class used for the kmeans algorithm. Default set to 2 class.

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 EstimateImageStatisticsQB1.xml -sample.mv 100 -sample.mt 100 -sample.vtr 0.5␣
↪-sample.vfn Class -classifier libsvm -classifier.libsvm.k linear -classifier.libsvm.
↪c 1 -classifier.libsvm.opt false -io.out svmModelQB1.txt -io.confmatout␣
↪svmConfusionMatrixQB1.csv
```

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", "EstimateImageStatisticsQB1.xml
↪")

TrainImagesClassifier.SetParameterInt("sample.mv", 100)

TrainImagesClassifier.SetParameterInt("sample.mt", 100)

TrainImagesClassifier.SetParameterFloat("sample.vtr", 0.5)

# 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

TrainRegression - 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 compo

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 between the ground truth and the estimated model. 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[1] 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 Name	Parar
io	Input and output data	Group
io.il	Input Image List	Input
io.csv	Input CSV file	Input
io.imstat	Input XML image statistics file	Input
io.out	Output regression model	Outpu
io.mse	Mean Square Error	Float
sample	Training and validation samples parameters	Group
sample.mt	Maximum training predictors	Int
sample.mv	Maximum validation predictors	Int
sample.vtr	Training and validation sample ratio	Float
classifier	Classifier to use for the training	Choic
classifier libsvm	LibSVM classifier	*Choic*
classifier dt	Decision Tree classifier	*Choic*
classifier gbt	Gradient Boosted Tree classifier	*Choic*
classifier ann	Artificial Neural Network classifier	*Choic*
classifier rf	Random forests classifier	*Choic*
classifier knn	KNN classifier	*Choic*
classifier sharkrf	Shark Random forests classifier	*Choic*
classifier sharkkm	Shark kmeans classifier	*Choic*
classifier.libsvm.k	SVM Kernel Type	Choic
classifier.libsvm.k linear	Linear	*Choic*
classifier.libsvm.k rbf	Gaussian radial basis function	*Choic*
classifier.libsvm.k poly	Polynomial	*Choic*
classifier.libsvm.k sigmoid	Sigmoid	*Choic*
classifier.libsvm.m	SVM Model Type	Choic
classifier.libsvm.m epssvr	Epsilon Support Vector Regression	*Choic*
classifier.libsvm.m nusvr	Nu Support Vector Regression	*Choic*
classifier.libsvm.c	Cost parameter C	Float
classifier.libsvm.nu	Cost parameter Nu	Float
classifier.libsvm.opt	Parameters optimization	Boole
classifier.libsvm.prob	Probability estimation	Boole
classifier.libsvm.eps	Epsilon	Float
		Cor

[1] Table: Parameters table for Train a regression model.

Table 7.6 – continued from previous page

Parameter Key	Parameter Name	Param
classifier.dt.max	Maximum depth of the tree	Int
classifier.dt.min	Minimum number of samples in each node	Int
classifier.dt.ra	Termination criteria for regression tree	Float
classifier.dt.cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split	Int
classifier.dt.f	K-fold cross-validations	Int
classifier.dt.r	Set Use1seRule flag to false	Boole
classifier.dt.t	Set TruncatePrunedTree flag to false	Boole
classifier.gbt.t	Loss Function Type	Choic
classifier.gbt.t sqr	Squared Loss	*Choic*
classifier.gbt.t abs	Absolute Loss	*Choic*
classifier.gbt.t hub	Huber Loss	*Choic*
classifier.gbt.w	Number of boosting algorithm iterations	Int
classifier.gbt.s	Regularization parameter	Float
classifier.gbt.p	Portion of the whole training set used for each algorithm iteration	Float
classifier.gbt.max	Maximum depth of the tree	Int
classifier.ann.t	Train Method Type	Choic
classifier.ann.t back	Back-propagation algorithm	*Choic*
classifier.ann.t reg	Resilient Back-propagation algorithm	*Choic*
classifier.ann.sizes	Number of neurons in each intermediate layer	String
classifier.ann.f	Neuron activation function type	Choic
classifier.ann.f ident	Identity function	*Choic*
classifier.ann.f sig	Symmetrical Sigmoid function	*Choic*
classifier.ann.f gau	Gaussian function (Not completely supported)	*Choic*
classifier.ann.a	Alpha parameter of the activation function	Float
classifier.ann.b	Beta parameter of the activation function	Float
classifier.ann.bpdw	Strength of the weight gradient term in the BACKPROP method	Float
classifier.ann.bpms	Strength of the momentum term (the difference between weights on the 2 previous iterations)	Float
classifier.ann.rdw	Initial value Delta_0 of update-values Delta_{ij} in RPROP method	Float
classifier.ann.rdwm	Update-values lower limit Delta_{min} in RPROP method	Float
classifier.ann.term	Termination criteria	Choic
classifier.ann.term iter	Maximum number of iterations	*Choic*
classifier.ann.term eps	Epsilon	*Choic*
classifier.ann.term all	Max. iterations + Epsilon	*Choic*
classifier.ann.eps	Epsilon value used in the Termination criteria	Float
classifier.ann.iter	Maximum number of iterations used in the Termination criteria	Int
classifier.rf.max	Maximum depth of the tree	Int
classifier.rf.min	Minimum number of samples in each node	Int
classifier.rf.ra	Termination Criteria for regression tree	Float
classifier.rf.cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split	Int
classifier.rf.var	Size of the randomly selected subset of features at each tree node	Int
classifier.rf.nbtrees	Maximum number of trees in the forest	Int
classifier.rf.acc	Sufficient accuracy (OOB error)	Float
classifier.knn.k	Number of Neighbors	Int
classifier.knn.rule	Decision rule	Choic
classifier.knn.rule mean	Mean of neighbors values	*Choic*
classifier.knn.rule median	Median of neighbors values	*Choic*
classifier.sharkrf.nbtrees	Maximum number of trees in the forest	Int
classifier.sharkrf.nodesize	Min size of the node for a split	Int
classifier.sharkrf.mtry	Number of features tested at each node	Int

Cor

Table 7.6 – continued from previous page

Parameter Key	Parameter Name	Parar
classifier.sharkrf.oobr	Out of bound ratio	Float
classifier.sharkkm.maxiter	Maximum number of iteration for the kmeans algorithm.	Int
classifier.sharkkm.k	The number of class used for the kmeans algorithm.	Int
rand	set user defined seed	Int
inxml	Load otb application from xml file	XML
outxml	Save otb application to xml file	XML

[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**: Linear Kernel, no mapping is done, this is the fastest option.
  - **Gaussian radial basis function**: This kernel is a good choice in most of the case. It is an exponential function of the euclidian distance between the vectors.
  - **Polynomial**: Polynomial Kernel, the mapping is a polynomial function.
  - **Sigmoid**: The kernel is a hyperbolic tangente function of the vectors.
  - **SVM Model Type**: Type of SVM formulation. Available choices are:
  - **Epsilon Support Vector Regression**: The distance between feature vectors from the training set and the fitting hyper-plane must be less than Epsilon. For outliers the penalty multiplier C is used .
  - **Nu Support Vector Regression**: Same as the epsilon regression except that this time the bounded parameter nu is used instead of epsilon.
  - **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.
  - **Cost parameter Nu**: Cost parameter Nu, in the range 0..1, the larger the value, the smoother the decision.
  - **Parameters optimization**: SVM parameters optimization flag.

    - **Probability estimation**: Probability estimation flag.
    - **Epsilon**: The distance between feature vectors from the training set and the fitting hyper-plane must be less than Epsilon. For outliersthe penalty mutliplier is set by C.
- **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}.
- **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 the number of samples in a node is smaller than this parameter, then this node will not be split.
- **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 further.
- **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.
- **K-fold cross-validations**: If cv_folds > 1, then it prunes a tree with K-fold cross-validation where K is equal to cv_folds.
- **Set Use1seRule 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}.
    - **Loss Function Type**: Type of loss functionused for training. Available choices are:
    - **Squared Loss**
    - **Absolute Loss**
    - **Huber Loss**
    - **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}.
    - **Train Method Type**: Type of training method for the multilayer perceptron (MLP) neural network. Available choices are:
    - **Back-propagation algorithm**: Method to compute the gradient of the loss function and adjust weights in the network to optimize the result.

- **Resilient Back-propagation algorithm**: Almost the same as the Back-prop algorithm except that it does not take into account the magnitude of the partial derivative (coordinate of the gradient) but only its sign.
- **Number of neurons in each intermediate layer**: The number of neurons in each intermediate layer (excluding input and output layers).
- **Neuron activation function type**: This function determine whether the output of the node is positive or not depending on the output of the transfert 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**: Set the number of iterations allowed to the network for its training. Training will stop regardless of the result when this number is reached.
- **Epsilon**: Training will focus on result and will stop once the precision isat most epsilon.
- **Max. iterations + Epsilon**: Both termination criteria are used. Training stop at the first reached.
- **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}.
- **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}.
  - **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.
- **Shark Random forests classifier**: This group of parameters allows setting Shark Random Forests classifier parameters. See complete documentation here url{http://image.diku.dk/shark/doxygen_pages/html/classshark_1_1_r_f_trainer.html}. It is noteworthy that training is parallel.
- **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.
- **Min size of the node for a split**: 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.
- **Number of features tested at each node**: The number of features (variables) which will be tested at each node in order to compute the split. If set to zero, the square root of the number of features is used.
- **Out of bound ratio**: Set the fraction of the original training dataset to use as the out of bag sample.A good default value is 0.66. .
- **Shark kmeans classifier**: This group of parameters allows setting Shark kMeans classifier parameters. See complete documentation here url{http://image.diku.dk/shark/sphinx_pages/build/html/rest_sources/tutorials/algorithms/kmeans.html}. .
- **Maximum number of iteration for the kmeans algorithm.**: The maximum number of iteration for the kmeans algorithm. 0=unlimited.
- **The number of class used for the kmeans algorithm.**: The number of class used for the kmeans algorithm. Default set to 2 class.

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_statistics.xml -classifier libsvm
```

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

TrainVectorClassifier - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *TrainVectorClassifier* .

[1] Table: Parameters table for Train Vector Classifier.

Parameter Key	Parameter Name
io	Input and output data
io.vd	Input Vector Data
io.stats	Input XML image statistics file
io.out	Output model
io.confmatout	Output confusion matrix or contingency table
layer	Layer Index
feat	Field names for training features.
valid	Validation data
valid.vd	Validation Vector Data
valid.layer	Layer Index
cfield	Field containing the class integer label for supervision
v	Verbose mode
classifier	Classifier to use for the training
classifier libsvm	LibSVM classifier
classifier boost	Boost classifier
classifier dt	Decision Tree classifier
classifier gbt	Gradient Boosted Tree classifier
classifier ann	Artificial Neural Network classifier
classifier bayes	Normal Bayes classifier
classifier rf	Random forests classifier
classifier knn	KNN classifier
classifier sharkrf	Shark Random forests classifier
classifier sharkkm	Shark kmeans classifier
classifier.libsvm.k	SVM Kernel Type
classifier.libsvm.k linear	Linear
classifier.libsvm.k rbf	Gaussian radial basis function
classifier.libsvm.k poly	Polynomial
classifier.libsvm.k sigmoid	Sigmoid
classifier.libsvm.m	SVM Model Type
classifier.libsvm.m csvc	C support vector classification
classifier.libsvm.m nusvc	Nu support vector classification
classifier.libsvm.m oneclass	Distribution estimation (One Class SVM)
classifier.libsvm.c	Cost parameter C
classifier.libsvm.nu	Cost parameter Nu
classifier.libsvm.opt	Parameters optimization
classifier.libsvm.prob	Probability estimation
classifier.boost.t	Boost Type
classifier.boost.t discrete	Discrete AdaBoost
classifier.boost.t real	Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)
classifier.boost.t logit	LogitBoost (technique producing good regression fits)
classifier.boost.t gentle	Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often g
classifier.boost.w	Weak count
classifier.boost.r	Weight Trim Rate
classifier.boost.m	Maximum depth of the tree
classifier.dt.max	Maximum depth of the tree
classifier.dt.min	Minimum number of samples in each node
classifier.dt.ra	Termination criteria for regression tree
classifier.dt.cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split
classifier.dt.f	K-fold cross-validations
classifier.dt.r	Set Use1seRule flag to false

Table 7.7 – continued from previous page

Parameter Key	Parameter Name
classifier.dt.t	Set TruncatePrunedTree flag to false
classifier.gbt.w	Number of boosting algorithm iterations
classifier.gbt.s	Regularization parameter
classifier.gbt.p	Portion of the whole training set used for each algorithm iteration
classifier.gbt.max	Maximum depth of the tree
classifier.ann.t	Train Method Type
classifier.ann.t back	Back-propagation algorithm
classifier.ann.t reg	Resilient Back-propagation algorithm
classifier.ann.sizes	Number of neurons in each intermediate layer
classifier.ann.f	Neuron activation function type
classifier.ann.f ident	Identity function
classifier.ann.f sig	Symmetrical Sigmoid function
classifier.ann.f gau	Gaussian function (Not completely supported)
classifier.ann.a	Alpha parameter of the activation function
classifier.ann.b	Beta parameter of the activation function
classifier.ann.bpdw	Strength of the weight gradient term in the BACKPROP method
classifier.ann.bpms	Strength of the momentum term (the difference between weights on the 2 previous iterations)
classifier.ann.rdw	Initial value Delta_0 of update-values Delta_{ij} in RPROP method
classifier.ann.rdwm	Update-values lower limit Delta_{min} in RPROP method
classifier.ann.term	Termination criteria
classifier.ann.term iter	Maximum number of iterations
classifier.ann.term eps	Epsilon
classifier.ann.term all	Max. iterations + Epsilon
classifier.ann.eps	Epsilon value used in the Termination criteria
classifier.ann.iter	Maximum number of iterations used in the Termination criteria
classifier.rf.max	Maximum depth of the tree
classifier.rf.min	Minimum number of samples in each node
classifier.rf.ra	Termination Criteria for regression tree
classifier.rf.cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split
classifier.rf.var	Size of the randomly selected subset of features at each tree node
classifier.rf.nbtrees	Maximum number of trees in the forest
classifier.rf.acc	Sufficient accuracy (OOB error)
classifier.knn.k	Number of Neighbors
classifier.sharkrf.nbtrees	Maximum number of trees in the forest
classifier.sharkrf.nodesize	Min size of the node for a split
classifier.sharkrf.mtry	Number of features tested at each node
classifier.sharkrf.oobr	Out of bound ratio
classifier.sharkkm.maxiter	Maximum number of iteration for the kmeans algorithm.
classifier.sharkkm.k	The number of class used for the kmeans algorithm.
rand	set user defined seed
inxml	Load otb application from xml file
outxml	Save otb application to xml 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 model**: Output file containing the model estimated (.txt format).
- **Output confusion matrix or contingency table**: Output file containing the confusion matrix or contingency

table (.csv format).The contingency table is output when we unsupervised algorithms is used otherwise the confusion matrix is output.

Layer Index: Index of the layer to use in the input vector file.

Field names for training features.: List of field names in the input vector data to be used as features for training.

[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.

Field containing the class integer label for supervision: Field containing the class id for supervision. The values in this field shall be cast into integers. Only geometries with this field available will be taken into account.

Verbose mode: Verbose mode, display the contingency table result.

- **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**: Linear Kernel, no mapping is done, this is the fastest option.
  - **Gaussian radial basis function**: This kernel is a good choice in most of the case. It is an exponential function of the euclidian distance between the vectors.
  - **Polynomial**: Polynomial Kernel, the mapping is a polynomial function.
  - **Sigmoid**: The kernel is a hyperbolic tangente function of the vectors.
  - **SVM Model Type**: Type of SVM formulation. Available choices are:
  - **C support vector classification**: This formulation allows imperfect separation of classes. The penalty is set through the cost parameter C.
  - **Nu support vector classification**: This formulation allows imperfect separation of classes. The penalty is set through the cost parameter Nu. As compared to C, Nu is harder to optimize, and may not be as fast.
  - **Distribution estimation (One Class SVM)**: All the training data are from the same class, SVM builds a boundary that separates the class from the rest of the feature space.
  - **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.
  - **Cost parameter Nu**: Cost parameter Nu, in the range 0..1, the larger the value, the smoother the decision.
  - **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**: This procedure trains the classifiers on weighted versions of the training sample, giving higher weight to cases that are currently misclassified. This is done for a sequence of weighter samples, and then the final classifier is defined as a linear combination of the classifier from each stage.
  - **Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)**: Adaptation of the Discrete Adaboost algorithm with Real value.
  - **LogitBoost (technique producing good regression fits)**: This procedure is an adaptive Newton algorithm for fitting an additive logistic regression model. Beware it can produce numeric instability.
  - **Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often good with regression data)**: A modified version of the Real Adaboost algorithm, using Newton stepping rather than exact optimization at each step.
  - **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 <= (1 - 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}.
- **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 the number of samples in a node is smaller than this parameter, then this node will not be split.
- **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 further.
- **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.
- **K-fold cross-validations**: If cv_folds > 1, then it prunes a tree with K-fold cross-validation where K is equal to cv_folds.
- **Set Use1seRule 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}.
- **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}.
  - **Train Method Type**: Type of training method for the multilayer perceptron (MLP) neural network. Available choices are:
  - **Back-propagation algorithm**: Method to compute the gradient of the loss function and adjust weights in the network to optimize the result.
  - **Resilient Back-propagation algorithm**: Almost the same as the Back-prop algorithm except that it does not take into account the magnitude of the partial derivative (coordinate of the gradient) but only its sign.
  - **Number of neurons in each intermediate layer**: The number of neurons in each intermediate layer (excluding input and output layers).
  - **Neuron activation function type**: This function determine whether the output of the node is positive or not depending on the output of the transfert 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**: Set the number of iterations allowed to the network for its training. Training will stop regardless of the result when this number is reached.
  - **Epsilon**: Training will focus on result and will stop once the precision isat most epsilon.
  - **Max. iterations + Epsilon**: Both termination criteria are used. Training stop at the first reached.
  - **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}.
- **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}.

- **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}.
- **Number of Neighbors**: The number of neighbors to use.
- **Shark Random forests classifier**: This group of parameters allows setting Shark Random Forests classifier parameters. See complete documentation here url{http://image.diku.dk/shark/doxygen_pages/html/classshark_1_1_r_f_trainer.html}. It is noteworthy that training is parallel.
- **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.
- **Min size of the node for a split**: 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.
- **Number of features tested at each node**: The number of features (variables) which will be tested at each node in order to compute the split. If set to zero, the square root of the number of features is used.
- **Out of bound ratio**: Set the fraction of the original training dataset to use as the out of bag sample.A good default value is 0.66. .
- **Shark kmeans classifier**: This group of parameters allows setting Shark kMeans classifier parameters. See complete documentation here url{http://image.diku.dk/shark/sphinx_pages/build/html/rest_sources/tutorials/algorithms/kmeans.html}. .
- **Maximum number of iteration for the kmeans algorithm.**: The maximum number of iteration for the kmeans algorithm. 0=unlimited.
- **The number of class used for the kmeans algorithm.**: The number of class used for the kmeans algorithm. Default set to 2 class.

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 perimeter  area  width -cfield predicted
```

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.

VectorClassifier - Vector Classification

Performs a classification of the input vector data according to a model file.

Detailed description

This application performs a vector data classification based on a model file produced by the TrainVectorClassifier application.Features of the vector data output will contain the class labels decided by the classifier (maximal class label = 65535). There are two modes: 1) Update mode: add of the 'cfield' field containing the predicted class in the input file. 2) Write mode: copies the existing fields of the input file in the output file and add the 'cfield' field containing the predicted class. If you have declared the output file, the write mode applies. Otherwise, the input file update mode will be applied.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorClassifier* .

[1] Table: Parameters table for Vector Classification.

Parameter Key	Parameter Name	Parameter Type
in	Name of the input vector data	Input vector data
instat	Statistics file	Input File name
model	Model file	Input File name
cfield	Field class	String
feat	Field names to be calculated.	List
confmap	Confidence map	Boolean
out	Output vector data file containing class labels	Output File name
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Name of the input vector data**: The input vector data file to classify.
- **Statistics file**: A XML file containing mean and standard deviation to centerand reduce samples before classification, produced by ComputeImagesStatistics application.
- **Model file**: Model file produced by TrainVectorClassifier application.
- **Field class**: Field containing the predicted class.Only geometries with this field available will be taken into account. The field is added either in the input file (if 'out' off) or in the output file. Caution, the 'cfield' must not exist in the input file if you are updating the file.
- **Field names to be calculated.**: List of field names in the input vector data used as features for training. Put the same field names as the TrainVectorClassifier application.
- **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). .
- **Output vector data file containing class labels**: Output vector data file storing sample values (OGR format).If not given, the input vector data file is updated.
- **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_VectorClassifier -in vectorData.shp -instat meanVar.xml -model svmModel.svm -
↪out vectorDataLabeledVector.shp -feat perimeter  area  width -cfield predicted
```

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 VectorClassifier application
VectorClassifier = otbApplication.Registry.CreateApplication("VectorClassifier")

# The following lines set all the application parameters:
```

```
VectorClassifier.SetParameterString("in", "vectorData.shp")

VectorClassifier.SetParameterString("instat", "meanVar.xml")

VectorClassifier.SetParameterString("model", "svmModel.svm")

VectorClassifier.SetParameterString("out", "vectorDataLabeledVector.shp")

# The following line execute the application
VectorClassifier.ExecuteAndWriteOutput()
```

Limitations

Shapefiles are supported. But the SQLite format is only supported in update mode.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

TrainVectorClassifier

VectorDimensionalityReduction - Vector Dimensionality Reduction

Performs dimensionality reduction of the input vector data according to a model file.

Detailed description

This application performs a vector data dimensionality reduction based on a model file produced by the TrainDimensionalityReduction application.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDimensionalityReduction* .

[1] Table: Parameters table for Vector Dimensionality Reduction.

Parameter Key	Parameter Name	Parameter Type
in	Name of the input vector data	Input vector data
instat	Statistics file	Input File name
model	Model file	Input File name
out	Output vector data file containing the reduced vector	Output File name
feat	Input features to use for reduction.	List
featout	Output feature	Choices
featout prefix	Prefix	*Choice*
featout list	List	*Choice*
featout.prefix.name	Feature name prefix	String
featout.list.names	Feature name list	String list
pcadim	Principal component dimension	Int
mode	Writting mode	Choices
mode overwrite	Overwrite	*Choice*
mode update	Update	*Choice*
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Name of the input vector data: The input vector data to reduce.

Statistics file: A XML file containing mean and standard deviation to center and reduce samples before dimensionality reduction (produced by ComputeImagesStatistics application).

Model file: A model file (produced by the TrainDimensionalityReduction application,.

Output vector data file containing the reduced vector: Output vector data file storing sample values (OGR format). If not given, the input vector data file is used. In overwrite mode, the original features will be lost.

Input features to use for reduction.: List of field names in the input vector data used as features for reduction.

Output feature: Naming of output features. Available choices are:

- **Prefix**: Use a name prefix.
- **Feature name prefix**: Name prefix for output features. This prefix is followed by the numeric index of each output feature.
- **List**: Use a list with all names.
- **Feature name list**: List of field names for the output features which result from the reduction.

Principal component dimension: This optional parameter can be set to reduce the number of eignevectors used in the PCA model file. This parameter can't be used for other models.

Writting mode: This parameter determines if the output file is overwritten or updated [overwrite/update]. If an output file name is given, the original file is copied before creating the new features. Available choices are:

- **Overwrite**: Overwrite mode.
- **Update**: Update mode.

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_VectorDimensionalityReduction -in vectorData.shp -instat meanVar.xml -model␣
↪model.txt -out vectorDataOut.shp -feat perimeter area width
```

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 VectorDimensionalityReduction␣
↪application
VectorDimensionalityReduction = otbApplication.Registry.CreateApplication(
↪"VectorDimensionalityReduction")

# The following lines set all the application parameters:
VectorDimensionalityReduction.SetParameterString("in", "vectorData.shp")

VectorDimensionalityReduction.SetParameterString("instat", "meanVar.xml")

VectorDimensionalityReduction.SetParameterString("model", "model.txt")

VectorDimensionalityReduction.SetParameterString("out", "vectorDataOut.shp")

# The following line execute the application
VectorDimensionalityReduction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

TrainDimensionalityReduction

Image Manipulation

ColorMapping - 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 been 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
op	Operation	Choices
op labeltocolor	Label to color	*Choice*
op colortolabel	Color to label	*Choice*
op.colortolabel.notfound	Not Found Label	Int
method	Color mapping method	Choices
method custom	Color mapping with custom labeled look-up table	*Choice*
method continuous	Color mapping with continuous look-up table	*Choice*
method optimal	Compute an optimized look-up table	*Choice*
method image	Color mapping with look-up table calculated on support image	*Choice*
method.custom.lut	Look-up table file	Input File name
method.continuous.lut	Look-up tables	Choices
method.continuous.lut red	Red	*Choice*
method.continuous.lut green	Green	*Choice*
method.continuous.lut blue	Blue	*Choice*
method.continuous.lut grey	Grey	*Choice*
method.continuous.lut hot	Hot	*Choice*
method.continuous.lut cool	Cool	*Choice*
method.continuous.lut spring	Spring	*Choice*
method.continuous.lut summer	Summer	*Choice*
method.continuous.lut autumn	Autumn	*Choice*
method.continuous.lut winter	Winter	*Choice*
method.continuous.lut copper	Copper	*Choice*
method.continuous.lut jet	Jet	*Choice*
method.continuous.lut hsv	HSV	*Choice*
method.continuous.lut overunder	OverUnder	*Choice*
method.continuous.lut relief	Relief	*Choice*
method.continuous.min	Mapping range lower value	Float
method.continuous.max	Mapping range higher value	Float
method.optimal.background	Background label	Int
method.image.in	Support Image	Input image
method.image.nodatavalue	NoData value	Float
method.image.low	lower quantile	Int
method.image.up	upper quantile	Int

Continued on next page

[1] Table: Parameters table for Color Mapping.

Table 7.8 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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_MUL_1_SVN_CLASS_MULTI_PNG_ColorTable.txt -out Colorized_ROI_QB_
↪MUL_1_SVN_CLASS_MULTI.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 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 implen
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

ConcatenateImages - 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 reads the input image list (single or multi-channel) and generates a single multi-channel image. The channel order is the same as the list.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ConcatenateImages* .

Parameter Key	Parameter Name	Parameter Type
il	Input images list	Input image list
out	Output Image	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input images list**: The list of images to concatenate, must have the same size.
- **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.

[1] Table: Parameters table for Images Concatenation.

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, SplitImage

DEMConvert - 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[1] 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 Name	Parameter Type
in	Input geo-referenced DEM	Input image
out	Prefix of the output files	Output File name
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 withtout 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

Authors

This application has been written by OTB-Team.

[1] Table: Parameters table for DEM Conversion.

DownloadSRTMTiles - Download or list SRTM tiles related to a set of images

Download or list SRTM tiles

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/), 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[1] 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 Name	Parameter Type
il	Input images list	Input image list
vl	Input vector data list	Input vector data list
names	Input tile names	String list
tiledir	Tiles directory	Directory
mode	Download/List corresponding SRTM tiles.	Choices
mode download	Download	*Choice*
mode list	List tiles	*Choice*
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input images list**: List of images on which you want to determine corresponding SRTM tiles.
- **Input vector data list**: List of vector data files on which you want to determine corresponding SRTM tiles.
- **Input tile names**: List of SRTM tile names to download. This list is added to the tiles derived from input images or vectors.The names should follow the SRTM tile naming convention, for instance N43E001.
- **Tiles directory**: Directory where SRTM tiles are stored. In download mode, the zipped archives will be downloaded to this directory. You'll need to unzip all tile files before using them in your application. In any case, this directory will be inspected to check which tiles are already downloaded.
- **Download/List corresponding SRTM tiles.** Available choices are:
- **Download**: Download corresponding tiles on USGE server.
- **List tiles**: List tiles in an existing local directory.
- **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 -tiledir /home/user/
↪srtm_dir/
```

[1] 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("tiledir", "/home/user/srtm_dir/")

# The following line execute the application
DownloadSRTMTiles.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

DynamicConvert - Dynamic Conversion

Change the pixel type and rescale the image's dynamic

Detailed description

This application performs an image pixel type conversion (short, ushort, uchar, int, uint, float and double types are handled). Th

The conversion can include a rescale of the data range, by default it's set between the 2nd to the 98th percentile. The rescale can be linear or log2. The choice of the output channels can be done with the extended filename, but less easy to handle. To do this, a 'channels' parameter allows you to select the desired bands at the output. There are 3 modes, the available choices are: * grayscale : to display mono image as standard color image * rgb : select 3 bands in the input image (multi-bands) * all : keep all bands.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DynamicConvert* .

[1] Table: Parameters table for Dynamic Conversion.

Parameter Key	Parameter Name	Parameter Type
in	Input image	Input image
out	Output Image	Output image
type	Rescale type	Choices
type linear	Linear	*Choice*
type log2	Log2	*Choice*
type.linear.gamma	Gamma correction factor	Float
mask	Input mask	Input image
quantile	Histogram quantile cutting	Group
quantile.high	High cut quantile	Float
quantile.low	Low cut quantile	Float
channels	Channels selection	Choices
channels all	Default mode	*Choice*
channels grayscale	Grayscale mode	*Choice*
channels rgb	RGB composition	*Choice*
channels.grayscale.channel	Grayscale channel	Int
channels.rgb.red	Red Channel	Int
channels.rgb.green	Green Channel	Int
channels.rgb.blue	Blue Channel	Int
outmin	Output min value	Float
outmax	Output max value	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input image: Input image.

Output Image: Output image.

Rescale type: Transfer function for the rescaling. Available choices are:

- **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 quantile cutting]: 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).

Channels selection: It's possible to select the channels of the output image. There are 3 modes, the available choices are:. Available choices are:

- **Default mode**: Select all bands in the input image, (1,...,n).
- **Grayscale mode**: Display single channel as standard color image.
- **Grayscale channel**
- **RGB composition**: Select 3 bands in the input image (multi-bands), by default (1,2,3).
- **Red Channel**: Red channel index.
- **Green Channel**: Green channel index.

- **Blue Channel**: Blue channel index.

Output min value: Minimum value of the output image.

Output max value: Maximum value of the 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_DynamicConvert -in QB_Toulouse_Ortho_XS.tif -out otbConvertWithScalingOutput.
↪png -type linear -channels rgb -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 DynamicConvert application
DynamicConvert = otbApplication.Registry.CreateApplication("DynamicConvert")

# The following lines set all the application parameters:
DynamicConvert.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

DynamicConvert.SetParameterString("out", "otbConvertWithScalingOutput.png")

DynamicConvert.SetParameterString("type","linear")

DynamicConvert.SetParameterString("channels","rgb")

DynamicConvert.SetParameterFloat("outmin", 0)

DynamicConvert.SetParameterFloat("outmax", 255)

# The following line execute the application
DynamicConvert.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

Convert, Rescale

ExtractROI - Extract ROI

Extract a ROI defined by the user.

Detailed description

This application extracts a Region Of Interest with user parameters. There are four mode of extraction. The standard mode allows the user to enter one point (upper left corner of the region to extract) and a size. The extent mode needs two points (upper left corner and lower right) and the radius mode need the center of the region and the radius : it will extract the rectangle containing the circle defined and limited by the image dimension. The fit mode needs a reference image or vector and the dimension of the extracted region will be the same as the extent of the reference. Different units are available such as pixel, image physical space or longitude and latitude.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ExtractROI* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
mode	Extraction mode	Choices
mode standard	Standard	*Choice*
mode fit	Fit	*Choice*
mode extent	Extent	*Choice*
mode radius	Radius	*Choice*
mode.fit.im	Reference image	Input image
mode.fit.vect	Reference vector	Input vector data
mode.extent.ulx	X coordinate of the Upper left corner	Float
mode.extent.uly	Y coordinate of Upper Left corner point.	Float
mode.extent.lrx	X coordinate of Lower Right corner point.	Float
mode.extent.lry	Y coordinate of Lower Right corner point.	Float
mode.extent.unit	Unit	Choices
mode.extent.unit pxl	Pixel	*Choice*
mode.extent.unit phy	Image physical space	*Choice*
mode.extent.unit lonlat	Longitude and latitude	*Choice*
mode.radius.r	Radius	Float
mode.radius.unitr	Radius unit	Choices
mode.radius.unitr pxl	Pixel	*Choice*
mode.radius.unitr phy	Image physical space	*Choice*
mode.radius.cx	X coordinate of the center	Float

Continued on next page

[1] Table: Parameters table for Extract ROI.

Table 7.9 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
mode.radius.cy	Y coordinate of the center	Float
mode.radius.unitc	Center unit	Choices
mode.radius.unitc pxl	Pixel	*Choice*
mode.radius.unitc phy	Image physical space	*Choice*
mode.radius.unitc lonlat	Lon/Lat	*Choice*
startx	Start X	Int
starty	Start Y	Int
sizex	Size X	Int
sizey	Size Y	Int
cl	Output Image channels	List
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: Image to be processed.

Output Image: Region of interest from the input image.

Extraction mode Available choices are:

- **Standard**: In standard mode extraction is done with 2 parameters : the upper left corner and the size of the region, decomposed in X and Y coordinates.
- **Fit**: In fit mode, extract is made from a reference : image or vector dataset.
- **Reference image**: Reference image to define the ROI.
- **Reference vector**: The extent of the input vector file is computed and then gives a region of interest that will be extracted.
- **Extent**: In extent mode, the ROI is defined by two points, the upper left corner and the lower right corner, decomposed in 2 coordinates : X and Y. The unit for those coordinates can be set.
  - **X coordinate of the Upper left corner**: X coordinate of upper left corner point.
  - **Y coordinate of Upper Left corner point.**: Y coordinate of upper left corner point.
  - **X coordinate of Lower Right corner point.**: X coordinate of lower right corner point.
  - **Y coordinate of Lower Right corner point.**: Y coordinate of lower right corner point.
  - **Unit** Available choices are:
  - **Pixel**: The unit for the parameters coordinates will be the pixel, meaning the index of the two points.
  - **Image physical space**: The unit for the parameters coordinates will be the physical measure of the image.
  - **Longitude and latitude**: The unit for the parameters coordinates will be the longitude and the latitude.
- **Radius**: In radius mode, the ROI is defined by a point and a radius. The unit of those two parameters can be chosen independently.
  - **Radius**: This is the radius parameter of the radius mode.

- **Radius unit** Available choices are:
- **Pixel**: The unit of the radius will be the pixel.
- **Image physical space**: The unit of the radius will be the physical measure of the image.
- **X coordinate of the center**: This is the center coordinate of the radius mode, it will be either an ordinate or a latitude.
- **Y coordinate of the center**
- **Center unit** Available choices are:
- **Pixel**: The unit for the center coordinates will be the pixel.
- **Image physical space**: The unit for the center coordinates will be the physical measure of the image.
- **Lon/Lat**: The unit for the center coordinates will be the longitude and the latitude.

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.

[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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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_ExtractROI -in VegetationIndex.hd -mode extent -mode.extent.ulx 40 -mode.
↪extent.uly 40 -mode.extent.lrx 150 -mode.extent.lry 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.SetParameterString("mode","extent")

ExtractROI.SetParameterFloat("mode.extent.ulx", 40)

ExtractROI.SetParameterFloat("mode.extent.uly", 40)

ExtractROI.SetParameterFloat("mode.extent.lrx", 150)

ExtractROI.SetParameterFloat("mode.extent.lry", 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.

ManageNoData - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ManageNoData* .

[1] Table: Parameters table for No Data management.

Parameter Key	Parameter Name	Parameter Type
in	Input image	Input image
out	Output Image	Output image
usenan	Consider NaN as no-data	Boolean
mode	No-data handling mode	Choices
mode buildmask	Build a no-data Mask	*Choice*
mode changevalue	Change the no-data value	*Choice*
mode apply	Apply a mask as no-data	*Choice*
mode.buildmask.inv	Inside Value	Float
mode.buildmask.outv	Outside Value	Float
mode.changevalue.newv	The new no-data value	Float
mode.apply.mask	Mask image	Input image
mode.apply.ndval	Nodata value used	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 -mode.buildmask.inv 255 -mode.buildmask.outv 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 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.SetParameterOutputImagePixelType("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:

BandMath

MultiResolutionPyramid - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *MultiResolutionPyramid* .

[1] Table: Parameters table for Multi Resolution Pyramid.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
level	Number Of Levels	Int
sfactor	Subsampling factor	Int
vfactor	Variance factor	Float
fast	Use Fast Scheme	Boolean
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 -sfactor 2 -vfactor 0.6 -fast false
```

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","false")

# The following line execute the application
MultiResolutionPyramid.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

Quicklook - 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
cl	Channel List	List
rox	ROI Origin X	Int
roy	ROI Origin Y	Int
rsx	ROI Size X	Int
rsy	ROI Size Y	Int
sr	Sampling ratio	Int
sx	Size X	Int
sy	Size Y	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] Table: Parameters table for Quick Look.

- **ROI Origin Y**: first point of ROI in y-direction.
- **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.

ReadImageInfo - 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[1] 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 Name	Parameter Type
in	Input Image	Input image
keywordlist	Display the OSSIM keywordlist	Boolean
outkwl	Write the OSSIM keywordlist to a geom file	Output File name
indexx	Start index X	Int
indexy	Start index Y	Int
sizex	Size X	Int
sizey	Size Y	Int
spacingx	Pixel Size X	Float
spacingy	Pixel Size Y	Float
originx	Image Origin X	Float
originy	Image Origin Y	Float
estimatedgroundspacingx	Estimated ground spacing X	Float
estimatedgroundspacingy	Estimated ground spacing Y	Float
numberbands	Number Of Bands	Int
sensor	Sensor id	String
id	Image id	String
time	Acquisition time	String
ullat	Upper left latitude	Float
ullon	Upper left longitude	Float
urlat	Upper right latitude	Float
urlon	Upper right longitude	Float
lrlat	Lower right latitude	Float
lrlon	Lower right longitude	Float
lllat	Lower left latitude	Float
lllon	Lower left longitude	Float
town	Nearest town	String
country	Country	String
rgb	Default RGB Display	Group
rgb.r	Red Band	Int
rgb.g	Green Band	Int
rgb.b	Blue Band	Int
projectionref	Projection	String
keyword	Keywordlist	String
gcp	Ground Control Points information	Group
gcp.count	GCPs Number	Int
gcp.proj	GCP Projection	String
gcp.ids	GCPs Id	String list
gcp.info	GCPs Info	String list

Continued on next page

[1] Table: Parameters table for Read image information.

Table 7.10 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
gcp.imcoord	GCPs Image Coordinates	String list
gcp.geocoord	GCPs Geographic Coordinates	String list
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

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.

SplitImage - 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 as output parameter, 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image**: Input multiband image filename.
- **Output Image**: The output filename will be used to get the prefix an the extension of the output written's image. For example with outimage.tif as output filename, the generated images will had an indice (corresponding at each bands) between the prefix and the extension, such as: outimage_0.tif and outimage_1.tif (if 2 bands).
- **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:

```
#!/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()
```

[1] Table: Parameters table for Split Image.

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

ConcatenateImages

TileFusion - Image Tile Fusion

Fusion of an image made of several tile files.

Detailed description

Automatically mosaic a set of non overlapping tile files into a single image. Images must have a matching number of bands and they must be listed in lexicographic order.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
il	Input Tile Images	Input image list
cols	Number of tile columns	Int
rows	Number of tile rows	Int
out	Output Image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Tile Images**: Input images to concatenate (in lexicographic order, for instance : (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.
- **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.

[1] Table: Parameters table for Image Tile Fusion.

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 -out EntireImage.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 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.

SAR

ComputeModulusAndPhase - Compute Modulus And Phase

This application computes the modulus and the phase of a complex SAR image.

Detailed description

This application computes the modulus and the phase of a complex SAR image. The input should be a single band image with complex pixels.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ComputeModulusAndPhase* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
modulus	Modulus	Output image
phase	Phase	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image**: Input image (complex single band).
- **Modulus**: Modulus of the input image computes with the following formula: $\sqrt{realreal + imagimag}$ where real and imag are respectively the real and the imaginary part of the input complex image. .
- **Phase**: Phase of the input image computes with the following formula: $\tan^{-1}(\frac{imag}{real})$ where real and imag are respectively the real and the imaginary part of the input complex 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_ComputeModulusAndPhase -in monobandComplexFloat.tif -modulus modulus.tif -
↪phase phase.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 ComputeModulusAndPhase application
ComputeModulusAndPhase = otbApplication.Registry.CreateApplication(
↪"ComputeModulusAndPhase")

# The following lines set all the application parameters:
ComputeModulusAndPhase.SetParameterString("in", "monobandComplexFloat.tif")

ComputeModulusAndPhase.SetParameterString("modulus", "modulus.tif")

ComputeModulusAndPhase.SetParameterString("phase", "phase.tif")

# The following line execute the application
ComputeModulusAndPhase.ExecuteAndWriteOutput()
```

[1] Table: Parameters table for Compute Modulus And Phase.

Limitations

The application takes as input single band image with complex pixels.

Authors

This application has been written by Alexia Mondot (alexia.mondot@c-s.fr) and Mickael Savinaud (mickael.savinaud@c-s.fr).

See Also

These additional resources can be useful for further information:

Despeckle, SARPolarMatrixConvert, SARPolarSynth

SARDeburst - SAR Deburst

This application performs deburst of Sentinel1 IW SLC images by removing redundant lines.

Detailed description

Sentinel1 IW SLC products are composed of several burst overlapping in azimuth time for each subswath, separated by black lines [1]. The deburst operation consist in generating a continuous image in terms of azimuth time, by removing black separation lines as well as redundant lines between bursts.

Note that the output sensor model is updated accordingly. This deburst operation is the perfect preprocessing step to orthorectify S1 IW SLC product with OTB [2] without suffering from artifacts caused by bursts separation.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARDeburst* .

Parameter Key	Parameter Name	Parameter Type
in	Input Sentinel1 IW SLC Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Sentinel1 IW SLC Image**: Raw Sentinel1 IW SLC image, or any extract of such made by OTB (geom file needed).
- **Output Image**: Deburst image, with updated geom file that can be further used by Orthorectification application. If the input image is a raw Sentinel1 product, uint16 output type should be used (encoding of S1 product). Otherwise, output type should match type of input image.
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Load otb application from xml file**: Load otb application from xml file.

[1] Table: Parameters table for SAR Deburst.

- **Save otb application to xml file**: Save otb application to xml file.

Example

To run this example in command-line, use the following:

```
otbcli_SARDeburst -in s1_iw_slc.tif -out s1_iw_slc_deburst.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 SARDeburst application
SARDeburst = otbApplication.Registry.CreateApplication("SARDeburst")

# The following lines set all the application parameters:
SARDeburst.SetParameterString("in", "s1_iw_slc.tif")

SARDeburst.SetParameterString("out", "s1_iw_slc_deburst.tif")

# The following line execute the application
SARDeburst.ExecuteAndWriteOutput()
```

Limitations

Only Sentinel1 IW SLC products are supported for now. Processing of other Sentinel1 modes or TerrasarX images will result in no changes in the image and metadata. Images from other sensors will lead to an error.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] Sentinel1 User Handbook, p. 52:
https://sentinel.esa.int/documents/247904/685163/Sentinel-1_User_Handbook
[2] OrthoRectification application

SARDecompositions - 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 <=> 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARDecompositions* .

Parameter Key	Parameter Name	Parameter Type
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	Decompositions	Choices
decomp haa	H-alpha-A incoherent decomposition	*Choice*
decomp barnes	Barnes incoherent decomposition	*Choice*
decomp huynen	Huynen incoherent decomposition	*Choice*
decomp pauli	Pauli coherent decomposition	*Choice*
inco	Incoherent decompositions	Group
inco.kernelsize	Kernel size for spatial incoherent averaging.	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).

[1] Table: Parameters table for SARDecompositions.

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

SARPolarMatrixConvert - 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 Sinclar 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 msinclairtocovariance –> Sinclair matrix to covariance matrix (input : 3 x 1 complex channel (HH, HV or VH, VV) | output : 6 complex channels) 3 msinclairtocircovariance –> 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *SARPolarMatrixConvert* .

[1] Table: Parameters table for SARPolarMatrixConvert.

Parameter Key	Parameter Name	Parameter Type
inc	Input : multi-band complex image	Input image
inf	Input : multi-band real image	Input image
inhh	Input : one-band complex image (HH)	Input image
inhv	Input : one-band complex image (HV)	Input image
invh	Input : one-band complex image (VH)	Input image
invv	Input : one-band complex image (VV)	Input image
outc	Output Complex Image	Output image
outf	Output Real Image	Output image
conv	Conversion	Choices
conv msinclairtocoherency	1 Monostatic : Sinclair matrix to coherency matrix (complex output)	*Choice*
conv msinclairtocovariance	2 Monostatic : Sinclair matrix to covariance matrix (complex output)	*Choice*
conv msinclairtocircovariance	3 Monostatic : Sinclair matrix to circular covariance matrix (complex output)	*Choice*
conv mcoherencytomueller	4 Monostatic : Coherency matrix to Mueller matrix	*Choice*
conv mcovariancetocoherency-degree	5 Monostatic : Covariance matrix to coherency degree	*Choice*
conv mcovariancetocoherency	6 Monostatic : Covariance matrix to coherency matrix (complex output)	*Choice*
conv mlinearcovariancetocir-cularcovariance	7 Monostatic : Covariance matrix to circular covariance matrix (complex output)	*Choice*
conv muellertomcovariance	8 Bi/mono : Mueller matrix to monostatic covariance matrix	*Choice*
conv bsinclairtocoherency	9 Bistatic : Sinclair matrix to coherency matrix (complex output)	*Choice*
conv bsinclairtocovariance	10 Bistatic : Sinclair matrix to covariance matrix (complex output)	*Choice*
conv bsinclairtocircovariance	11 Bistatic : Sinclair matrix to circular covariance matrix (complex output)	*Choice*
conv sinclairtomueller	12 Bi/mono : Sinclair matrix to Mueller matrix	*Choice*
conv muellertopoldegandpower	13 Bi/mono : Mueller matrix to polarisation degree and power	*Choice*
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 mcoherency.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 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

SARPolarSynth - 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 synthetis).

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 synthetis). The new basis A and B are indicated through two Jones vectors, defined by the user thanks to orientation (psi) and ellipticity (khi) parameters. These parameters are namely psii, khii, psir and khir. 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
psii	psii	Float
khii	khii	Float
psir	psir	Float
khir	khir	Float
emissionh	Emission H	Int
emissionv	Emission V	Int
mode	Forced mode	Choices
mode none	None	*Choice*
mode co	Copolarization	*Choice*
mode cross	Crosspolarization	*Choice*
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).

[1] Table: Parameters table for SARPolarSynth.

- **Emission H**: This parameter is useful in determining the polarization architecture (dual polarization case).
- **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

Segmentation

ComputeOGRLayersFeaturesStatistics - 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[1] 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 Name	Parameter Type
inshp	Vector Data	Input vector data
outstats	Output XML file	Output File name
feat	Feature	List
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Vector Data**: Name of the input shapefile.
- **Output XML file**: XML file containing mean and variance of each feature.
- **Feature**: 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.

Example

To run this example in command-line, use the following:

```
otbcli_ComputeOGRLayersFeaturesStatistics -inshp vectorData.shp -outstats results.xml␣
↪-feat perimeter
```

[1] Table: Parameters table for ComputeOGRLayersFeaturesStatistics.

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(
↪"ComputeOGRLayersFeaturesStatistics")

# 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,TrainVectorClassifier

ConnectedComponentSegmentation - 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Shape	Output vector data
mask	Mask expression	String
expr	Connected Component Expression	String
minsize	Minimum Object Size	Int
obia	OBIA Expression	String
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

[1] Table: Parameters table for Connected Component Segmentation.

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" -minsize 15 -obia "SHAPE_Elongation>8" -out␣
↪ConnectedComponentSegmentation.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 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.

HooverCompareSegmentation - 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 ar
The application can output the overall Hoover scores along with coloredimages 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[1] 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 Name	Parameter Type
ingt	Input ground truth	Input image
inms	Input machine segmentation	Input image
bg	Background label	Int
th	Overlapping threshold	Float
outgt	Colored ground truth output	Output image
outms	Colored machine segmentation output	Output image
rc	Correct detection score	Float
rf	Over-segmentation score	Float
ra	Under-segmentation score	Float
rm	Missed detection score	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 uint8
```

To run this example from Python, use the following code snippet:

[1] 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

LSMSSegmentation - Exact Large-Scale Mean-Shift segmentation, step 2

This application performs the second step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS) [1].

Detailed description

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 tilesizex and tilesizey parameters for tile-wise processing, with the guarantees of identical results.

Filtered range image and spatial image should be created with the MeanShiftSmoothing application outputs (fout and foutpos) [2], 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.

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.

The output of this application can be passed to the LSMSSmallRegionMerging [3] or LSMSVectorization [4] applications to complete the LSMS workflow.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Filtered image	Input image
inpos	Filtered position image	Input image
out	Output labeled Image	Output image
spatialr	Spatial radius	Float
ranger	Range radius	Float
minsize	Minimum Segment Size	Int
tilesizex	Size of tiles in pixel (X-axis)	Int
tilesizey	Size of tiles in pixel (Y-axis)	Int
tmpdir	Directory where to write temporary files	Directory
cleanup	Temporary files cleaning	Boolean
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Filtered image**: The filtered image, corresponding to the fout output parameter of the MeanShiftSmoothing application.
- **Filtered position image**: The filtered position image, corresponding to the foutpos output parameter of the MeanShiftSmoothing application.
- **Output labeled Image**: This output contains the segmented image, where each pixel value is the unique integer label of the segment it belongs to. It is recommended to set the pixel type to uint32.
- **Spatial radius**: Threshold on Spatial distance to consider pixels in the same segment. A good value is half the spatial radius used in the MeanShiftSmoothing application (spatialr parameter).
- **Range radius**: Threshold on spectral signature euclidean distance (expressed in radiometry unit) to consider pixels in the same segment. A good value is half the range radius used in the MeanShiftSmoothing application (ranger parameter).
- **Minimum Segment Size**: Minimum Segment Size. If, after the segmentation, a segment is of size lower than this criterion, the segment is discarded.
- **Size of tiles in pixel (X-axis)**: Size of tiles along the X-axis for tile-wise processing.
- **Size of tiles in pixel (Y-axis)**: Size of tiles along the Y-axis for tile-wise processing.
- **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 remove 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.

[1] Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 2.

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 15 -minsize 0 -tilesizex 256 -tilesizey 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 LSMSSegmentation application
LSMSSegmentation = otbApplication.Registry.CreateApplication("LSMSSegmentation")

# The following lines set all the application parameters:
LSMSSegmentation.SetParameterString("in", "smooth.tif")

LSMSSegmentation.SetParameterString("inpos", "position.tif")

LSMSSegmentation.SetParameterString("out", "segmentation.tif")

LSMSSegmentation.SetParameterFloat("spatialr", 5)

LSMSSegmentation.SetParameterFloat("ranger", 15)

LSMSSegmentation.SetParameterInt("minsize", 0)

LSMSSegmentation.SetParameterInt("tilesizex", 256)

LSMSSegmentation.SetParameterInt("tilesizey", 256)

# The following line execute the application
LSMSSegmentation.ExecuteAndWriteOutput()
```

Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) [1] and may not be suited for any other purpose. This application is not compatible with in-memory connection since it does its own internal streaming.

Authors

This application has been written by David Youssefi.

See Also

These additional resources can be useful for further information:

[1] Michel, J., Youssefi, D., & Grizonnet, M. (2015). Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images. IEEE Transactions on Geoscience and Remote Sensing, 53(2), 952-964.

[2] MeanShiftSmoothing
[3] LSMSSmallRegionsMerging
[4] LSMSVectorization

LSMSSmallRegionsMerging - Exact Large-Scale Mean-Shift segmentation, step 3 (optional)

This application performs the third (optional) step of the exact Large-Scale Mean-Shift segmentation workflow [1].

Detailed description

Given a segmentation result (can be the out output parameter of the LSMSSegmentation application [2]) and the original image, it will merge segments whose size in pixels is lower than minsize parameter with the adjacent segments with the adjacent segment with closest radiometry and acceptable size.

Small segments will be processed by increasing size: first all segments for which area is equal to 1 pixel will be merged with adjacent segments, then all segments of area equal to 2 pixels will be processed, until segments of area minsize. For large images one can use the tilesizex and tilesizey parameters for tile-wise processing, with the guarantees of identical results.

The output of this application can be passed to the LSMSVectorization application [3] to complete the LSMS workflow.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input image	Input image
inseg	Segmented image	Input image
out	Output Image	Output image
minsize	Minimum Segment Size	Int
tilesizex	Size of tiles in pixel (X-axis)	Int
tilesizey	Size of tiles in pixel (Y-axis)	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input image**: The input image, containing initial spectral signatures corresponding to the segmented image (inseg).
- **Segmented image**: Segmented image where each pixel value is the unique integer label of the segment it belongs to.
- **Output Image**: The output image. The output image is the segmented image where the minimal segments have been merged. An ecoding of uint32 is advised.
- **Minimum Segment Size**: Minimum Segment Size. If, after the segmentation, a segment is of size lower than this criterion, the segment is merged with the segment that has the closest sepctral signature.
- **Size of tiles in pixel (X-axis)**: Size of tiles along the X-axis for tile-wise processing.
- **Size of tiles in pixel (Y-axis)**: Size of tiles along the Y-axis for tile-wise processing.

[1] Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 3 (optional).

- **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 -tilesizex 256 -tilesizey 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 LSMSSmallRegionsMerging application
LSMSSmallRegionsMerging = otbApplication.Registry.CreateApplication(
↪"LSMSSmallRegionsMerging")

# 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("tilesizex", 256)

LSMSSmallRegionsMerging.SetParameterInt("tilesizey", 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. This application is not compatible with in-memory connection since it does its own internal streaming.

Authors

This application has been written by David Youssefi.

See Also

These additional resources can be useful for further information:

[1] Michel, J., Youssefi, D., & Grizonnet, M. (2015). Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images. IEEE Transactions on Geoscience and Remote Sensing, 53(2), 952-964.

[2] LSMSegmentation

[3] LSMSVectorization

LSMSVectorization - Exact Large-Scale Mean-Shift segmentation, step 4

This application performs the fourth step of the exact Large-Scale Mean-Shift segmentation workflow [1].

Detailed description

Given a segmentation result (label image), that may come from the LSMSSegmentation [2] application (out parameter) or have been processed for small regions merging [3] (out parameter), 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 tilesizex and tilesizey parameters for tile-wise processing, with the guarantees of identical results.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LSMSVectorization* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
inseg	Segmented image	Input image
out	Output GIS vector file	Output File name
tilesizex	Size of tiles in pixel (X-axis)	Int
tilesizey	Size of tiles in pixel (Y-axis)	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image**: The input image, containing initial spectral signatures corresponding to the segmented image (inseg).
- **Segmented image**: Segmented image where each pixel value is the unique integer label of the segment it belongs to.
- **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 for tile-wise processing.
- **Size of tiles in pixel (Y-axis)**: Size of tiles along the Y-axis for tile-wise processing.
- **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.

[1] Table: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 4.

Example

To run this example in command-line, use the following:

```
otbcli_LSMSVectorization -in maur_rgb.png -inseg merged.tif -out vector.shp -
↪tilesizex 256 -tilesizey 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", "maur_rgb.png")

LSMSVectorization.SetParameterString("inseg", "merged.tif")

LSMSVectorization.SetParameterString("out", "vector.shp")

LSMSVectorization.SetParameterInt("tilesizex", 256)

LSMSVectorization.SetParameterInt("tilesizey", 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:

[1] Michel, J., Youssefi, D., & Grizonnet, M. (2015). Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images. IEEE Transactions on Geoscience and Remote Sensing, 53(2), 952-964.

[2] LSMSegmentation

[3] LSMSmallRegionMerging

LargeScaleMeanShift - Large-Scale MeanShift

Large-scale segmentation using MeanShift

Detailed description

This application chains together the 4 steps of the MeanShit framework, that is the MeanShiftSmoothing [1], the LSMSSegmentation [2], the LSMSSmallRegionsMerging [3] and the LSMSVectorization [4].

This application can be a preliminary step for an object-based analysis.

It generates a vector data file containing the regions extracted with the MeanShift algorithm. The spatial and range radius parameters allow adapting the sensitivity of the algorithm depending on the image dynamic and resolution. There is a step to remove small regions whose size (in pixels) is less than the given 'minsize' parameter. These regions are merged to a similar neighbor region. In the output vectors, there are additional fields to describe each region. In particular the mean and standard deviation (for each band) is computed for each region using the input image as support. If an optional 'imfield' image is given, it will be used as support image instead.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *LargeScaleMeanShift* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
spatialr	Spatial radius	Int
ranger	Range radius	Float
minsize	Minimum Segment Size	Int
tilesizex	Size of tiles in pixel (X-axis)	Int
tilesizey	Size of tiles in pixel (Y-axis)	Int
mode	Output mode	Choices
mode vector	Segmentation as vector output	*Choice*
mode raster	Standard segmentation with labeled raster output	*Choice*
mode.vector.imfield	Support image for field computation	Input image
mode.vector.out	Output GIS vector file	Output File name
mode.raster.out	The output raster image	Output image
cleanup	Temporary files cleaning	Boolean
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: The input image can be any single or multiband image. Beware of pontential imbalance between bands ranges as it may alter euclidean distance.

Spatial radius: Radius of the spatial neighborhood for averaging. Higher values will result in more smoothing and higher processing time.

Range radius: Threshold on spectral signature euclidean distance (expressed in radiometry unit) to consider neighborhood pixel for averaging. Higher values will be less edge-preserving (more similar to simple average in neighborhood), whereas lower values will result in less noise smoothing. Note that this parameter has no effect on processing time.

Minimum Segment Size: Minimum Segment Size. If, after the segmentation, a segment is of size lower than this criterion, the segment is merged with the segment that has the closest sepctral signature.

[1] Table: Parameters table for Large-Scale MeanShift.

Size of tiles in pixel (X-axis): Size of tiles along the X-axis for tile-wise processing.

Size of tiles in pixel (Y-axis): Size of tiles along the Y-axis for tile-wise processing.

Output mode: Type of segmented output. Available choices are:

- **Segmentation as vector output**: In this mode, the application will produce a vector file or database and compute field values for each region.
- **Support image for field computation**: This is an optional support image that can be used to compute field values in each region. Otherwise, the input image is used as support.
- **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.
- **Standard segmentation with labeled raster output**: In this mode, the application will produce a standard labeled raster.
- **The output raster image**: It corresponds to the output of the small region merging step.

Temporary files cleaning: If activated, the application will try to clean all temporary files it created.

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_LargeScaleMeanShift -in QB_1_ortho.tif -spatialr 4 -ranger 80 -minsize 16 -
↪mode.vector.out regions.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 LargeScaleMeanShift application
LargeScaleMeanShift = otbApplication.Registry.CreateApplication("LargeScaleMeanShift")

# The following lines set all the application parameters:
LargeScaleMeanShift.SetParameterString("in", "QB_1_ortho.tif")

LargeScaleMeanShift.SetParameterInt("spatialr", 4)

LargeScaleMeanShift.SetParameterFloat("ranger", 80)

LargeScaleMeanShift.SetParameterInt("minsize", 16)

LargeScaleMeanShift.SetParameterString("mode.vector.out", "regions.shp")

# The following line execute the application
LargeScaleMeanShift.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] MeanShiftSmoothing
[2] LSMSSegmentation
[3] LSMSSmallRegionsMerging
[4] LSMSVectorization

OGRLayerClassifier - 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[1] 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 Name	Parameter Type
inshp	Name of the input shapefile	Input vector data
instats	XML file containing mean and variance of each feature.	Input File name
insvm	Input model filename.	Output File name
feat	Features	List
cfield	Field containing the predicted class.	String
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] Table: Parameters table for OGRLayerClassifier.

- **Field containing the predicted class.**: Field containing the predicted class.
- **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_OGRLayerClassifier -inshp vectorData.shp -instats meanVar.xml -insvm svmModel.
↪svm -feat perimeter -cfield predicted
```

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

Segmentation - 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 tilesize 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[1] 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 Name	Parameter Type
in	Input Image	Input image
filter	Segmentation algorithm	Choices
filter meanshift	Mean-Shift	*Choice*
filter cc	Connected components	*Choice*
filter watershed	Watershed	*Choice*
filter mprofiles	Morphological profiles based segmentation	*Choice*
filter.meanshift.spatialr	Spatial radius	Int
filter.meanshift.ranger	Range radius	Float
filter.meanshift.thres	Mode convergence threshold	Float
filter.meanshift.maxiter	Maximum number of iterations	Int
filter.meanshift.minsize	Minimum region size	Int
filter.cc.expr	Condition	String
filter.watershed.threshold	Depth Threshold	Float
filter.watershed.level	Flood Level	Float
filter.mprofiles.size	Profile Size	Int
filter.mprofiles.start	Initial radius	Int
filter.mprofiles.step	Radius step.	Int
filter.mprofiles.sigma	Threshold of the final decision rule	Float
mode	Processing mode	Choices
mode vector	Tile-based large-scale segmentation with vector output	*Choice*
mode raster	Standard segmentation with labeled raster output	*Choice*
mode.vector.out	Output vector file	Output File name
mode.vector.outmode	Writing mode for the output vector file	Choices
mode.vector.outmode ulco	Update output vector file, only allow creating new layers	*Choice*

Continued on next page

[1] Table: Parameters table for Segmentation.

Table 7.11 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
mode.vector.outmode ovw	Overwrite output vector file if existing.	*Choice*
mode.vector.outmode ulovw	Update output vector file, overwrite existing layer	*Choice*
mode.vector.outmode ulu	Update output vector file, update existing layer	*Choice*
mode.vector.inmask	Mask Image	Input image
mode.vector.neighbor	8-neighbor connectivity	Boolean
mode.vector.stitch	Stitch polygons	Boolean
mode.vector.minsize	Minimum object size	Int
mode.vector.simplify	Simplify polygons	Float
mode.vector.layername	Layer name	String
mode.vector.fieldname	Geometry index field name	String
mode.vector.tilesize	Tiles size	Int
mode.vector.startlabel	Starting geometry index	Int
mode.vector.ogroptions	OGR options for layer creation	String list
mode.raster.out	Output labeled image	Output image
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 si 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 segmentationTo run this example in command-line, use the following:

```
otbcli_Segmentation -in QB_Toulouse_Ortho_PAN.tif -mode vector -mode.vector.out␣
↪SegmentationVector.sqlite -filter watershed
```

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 segmentationTo run this example in command-line, use the following:

```
otbcli_Segmentation -in QB_Toulouse_Ortho_PAN.tif -mode raster -mode.raster.out␣
↪SegmentationRaster.tif uint16 -filter meanshift
```

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.SetParameterOutputImagePixelType("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

Vector Data Manipulation

ConcatenateVectorData - Concatenate Vector Data

Concatenate vector data files

Detailed description

This application concatenates a list of vector data files to produce a unique vector data output file.

This application will gather all the geometries from the input files and write them into an output vector data file. Any format supported by OGR can be used. Ideally, all inputs should have the same set of fields and the same spatial reference system.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
vd	Input vector files	Input vector data list
out	Concatenated output	Output vector data
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input vector files**: Vector data files to be concatenated.
- **Concatenated output**: Output conctenated vector data 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.

[1] Table: Parameters table for Concatenate Vector Data.

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

The vector data must be contain the same type of geometries (point / lines / polygons). The fields present in the output file are the ones from the first input.

Authors

This application has been written by OTB-Team.

Rasterization - 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 refere

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 woth 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[1] 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 Name	Parameter Type
in	Input vector dataset	Input vector data
out	Output image	Output image
im	Input reference image	Input image
szx	Output size x	Int
szy	Output size y	Int
epsg	Output EPSG code	Int
orx	Output Upper-left x	Float
ory	Output Upper-left y	Float
spx	Spacing (GSD) x	Float
spy	Spacing (GSD) y	Float
background	Background value	Float
mode	Rasterization mode	Choices
mode binary	Binary mode	*Choice*
mode attribute	Attribute burning mode	*Choice*
mode.binary.foreground	Foreground value	Float
mode.attribute.field	The attribute field to burn	String
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).

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.

[1] Table: Parameters table for Rasterization.

- **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.

VectorDataExtractROI - 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[1] 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 Name	Parameter Type
io	Input and output data	Group
io.vd	Input Vector data	Input vector data
io.in	Support image	Input image
io.out	Output Vector data	Output vector data
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.
- **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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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.

[1] Table: Parameters table for VectorData Extract ROI.

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 apTvUtVectorDataExtractROIApplicationTest.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 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.

VectorDataReprojection - 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[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is

[1] Table: Parameters table for Vector Data reprojection.

VectorDataReprojection .

Parameter Key	Parameter Name	Parameter Type
in	Input data	Group
in.vd	Input vector data	Input File name
in.kwl	Use image keywords list	Input image
out	Output data	Group
out.vd	Output vector data	Output File name
out.proj	Output Projection choice	Choices
out.proj image	Use image projection ref	*Choice*
out.proj user	User defined projection	*Choice*
out.proj.image.in	Image used to get projection map	Input image
out.proj.user.map	Map Projection	Choices
out.proj.user.map utm	Universal Trans-Mercator (UTM)	*Choice*
out.proj.user.map lambert2	Lambert II Etendu	*Choice*
out.proj.user.map lambert93	Lambert93	*Choice*
out.proj.user.map wgs	WGS 84	*Choice*
out.proj.user.map epsg	EPSG Code	*Choice*
out.proj.user.map.utm.zone	Zone number	Int
out.proj.user.map.utm.northhem	Northern Hemisphere	Boolean
out.proj.user.map.epsg.code	EPSG Code	Int
elev	Elevation management	Group
elev.dem	DEM directory	Directory
elev.geoid	Geoid File	Input File name
elev.default	Default elevation	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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**
    - **Map Projection**: Defines the 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 to find which EPSG code is associated to your projection;.
- **EPSG Code**: See 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(https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-data/blob/master/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 -out.vd reprojected_vd.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 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.

VectorDataSetField - 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[1] 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 Name	Parameter Type
in	Input	Input vector data
out	Output	Output vector data
fn	Field	String
fv	Value	String
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 -fv Sample polygon
```

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 VectorDataSetField application
```

[1] Table: Parameters table for Vector data set field.

```
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.

VectorDataTransform - Vector Data Transformation

Apply a transform to each vertex of the input VectorData

Detailed description

This application iterates over each vertex in the input vector data file and performs a transformation on this vertex.

It is the equivalent of [1] that transforms images. For instance, if you extract the envelope of an image with [2], and you transform this image with [1], you may want to use this application to operate the same transform on the envelope.

The applied transformation is a 2D similarity. It manages translation, rotation, scaling, and can be centered or not. Note that the support image is used to define the reference coordinate system in which the transform is applied. For instance the input vector data can have WGS84 coordinates, the support image is in UTM, so a translation of 1 pixel along X corresponds to the X pixel size of the input image along the X axis of the UTM coordinates frame. This image can also be in sensor geometry.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *VectorDataTransform* .

[1] Table: Parameters table for Vector Data Transformation.

Parameter Key	Parameter Name	Parameter Type
vd	Input Vector data	Input vector data
out	Output Vector data	Output vector data
in	Support image	Input image
transform	Transform parameters	Group
transform.tx	X Translation	Float
transform.ty	Y Translation	Float
transform.ro	Rotation Angle	Float
transform.centerx	Center X	Float
transform.centery	Center Y	Float
transform.scale	Scale	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Vector data: Input vector data file to transform.

Output Vector data: Output vector data with .

Support image: Image defining the reference coordinate system in which the transform is applied. Both projected and sensor images are supported.

[Transform parameters]: Group of parameters to define the transform.

- **X Translation**: Translation in the X direction (in pixels).
- **Y Translation**: Translation in the Y direction (in pixels).
- **Rotation Angle**: Angle of the rotation (in degrees).
- **Center X**: X coordinate of the rotation and scaling center (in physical units).
- **Center Y**: Y coordinate of the rotation and scaling center (in physical units).
- **Scale**: The scale coefficient 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 VectorDataTransform.shp -transform.ro 5
```

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.

See Also

These additional resources can be useful for further information:

[1] RigidTransformResample
[2] ImageEnvelope

Image Filtering

ContrastEnhancement - Contrast Enhancement

This application is the implementation of the histogram equalization algorithm. It can be used to enhance contrast in an image or to reduce the dynamic of the image without losing too much contrast. It offers several options as a no data value, a contrast limitation factor, a local version of the algorithm and also a mode to equalize the luminance of the image.

Detailed description

This application is the implementation of the histogram equalization algorithm. The idea of the algorithm is to use the whole available dynamic. In order to do so it computes a histogram over the image and then use the whole dynamic: meaning flattening the histogram. That gives us gain for each bin that transform the original histogram into the flat one. This gain is then apply on the original image. The application proposes several options to allow a finer result: - There is an option to limit contrast. We choose to limit the contrast by modifying the original histogram. To do so we clip the histogram at a given height and redistribute equally among the bins the clipped population. Then we add a local version of the algorithm. - It is possible to apply the algorithm on tiles of the image, instead of on the whole image. That gives us gain depending on the value of the pixel and its position in the image. In order to smoothen the result we interpolate the gain between tiles.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *ContrastEnhancement* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
bins	Number of bins	Int
hfact	Contrast Limitation	Float
nodata	Nodata Value	Float
spatial	Spatial parameters for the histogram computation	Choices
spatial local	Local	*Choice*
spatial global	Global	*Choice*
spatial.local.h	Thumbnail height	Int
spatial.local.w	Thumbnail width	Int
minmax	Minimum and maximum settings	Choices
minmax auto	Automatic	*Choice*
minmax manual	Manual settings for min/max values	*Choice*
minmax.auto.global	Global	Boolean
minmax.manual.min	Minimum value	Float
minmax.manual.max	Maximum value	Float
mode	What to equalized	Choices
mode each	Channels	*Choice*
mode lum	Luminance	*Choice*
mode.lum.red	Red channel	Group
mode.lum.red.ch	Red channel	Int
mode.lum.red.coef	Value for luminance computation for the red channel	Float
mode.lum.green	Green channel	Group
mode.lum.green.ch	Green channel	Int
mode.lum.green.coef	Value for luminance computation of the green channel	Float
mode.lum.blue	Blue channel	Group
mode.lum.blue.ch	Blue channel	Int
mode.lum.blue.coef	Value for luminance computation of the blue channel	Float
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: Input image.

Output Image: Output image.

Number of bins: Number of bins in the histogram.

Contrast Limitation: This parameter will set the maximum height accepted in a bin on the input image histogram. The maximum height will be computed as hfact*eqHeight where eqHeight is the height of the theoretical flat histogram. The higher hfact, the higher the contrast. When using 'luminance mode', it is recommended to limit this factor to a small value (ex: 4).

Nodata Value: If there is a value in the image that has no visualization meaning, it can be ignored by the algorithm.

Spatial parameters for the histogram computation Available choices are:

[1] Table: Parameters table for Contrast Enhancement.

- **Local**: The histograms will be computed on each thumbnail. Each of the histogram will be equalized and the corresponding gain will be interpolated.
- **Thumbnail height**: Height of the thumbnail over which the histogram will be computed. The value is in pixels.
- **Thumbnail width**: Width of the thumbnail over which the histogram will be computed. The value is in pixels.
- **Global**: The histogram will be computed on the whole image. The equalization will be computed on this histogram.

Minimum and maximum settings: Minimum and maximum value that will bound the histogram and thus the dynamic of the resulting image. Values over those boundaries will be clipped. Available choices are:

- **Automatic**: Minimum and maximum value will be computed on the image (nodata value won't be taken into account) . Each band will have a minimum and a maximum.
- **Global**: Min/max computation will result in the same minimum and maximum for all the bands.
- **Manual settings for min/max values**: Minimum and maximum value will be set by the user.
- **Minimum value**
- **Maximum value**

What to equalized Available choices are:

- **Channels**: Each channel is equalized independently.
- **Luminance**: The relative luminance is computed according to the coefficients.Then the histogram is equalized and the gain is applied to each of the channels. The channel gain will depend on the weight (coef) of the channel in the luminance. Note that default values come from color space theories on how human eyes perceive colors).
  - **Red channel**
  - **Red channel**
  - **Value for luminance computation for the red channel**
  - **Green channel**
  - **Green channel**
  - **Value for luminance computation of the green channel**
  - **Blue channel**
  - **Blue channel**
  - **Value for luminance computation of the blue channel**

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

Local contrast enhancement by luminanceTo run this example in command-line, use the following:

```
otbcli_ContrastEnhancement -in colours.tif -out equalizedcolors.tif float -bins 256 -
↪spatial.local.w 500 -spatial.local.h 500 -mode lum
```

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 ContrastEnhancement application
ContrastEnhancement = otbApplication.Registry.CreateApplication("ContrastEnhancement")

# The following lines set all the application parameters:
ContrastEnhancement.SetParameterString("in", "colours.tif")

ContrastEnhancement.SetParameterString("out", "equalizedcolors.tif")
ContrastEnhancement.SetParameterOutputImagePixelType("out", 6)

ContrastEnhancement.SetParameterInt("bins", 256)

ContrastEnhancement.SetParameterInt("spatial.local.w", 500)

ContrastEnhancement.SetParameterInt("spatial.local.h", 500)

ContrastEnhancement.SetParameterString("mode","lum")

# The following line execute the application
ContrastEnhancement.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

Despeckle - Despeckle

Perform speckle noise reduction on SAR image.

Detailed description

SAR images are affected by speckle noise that inherently exists in and which degrades the image quality. It is caused by the coherent nature of back-scattered waves from multiple distributed targets. It is locally strong and it increases the mean Grey level of a local area.

Reducing the speckle noise enhances radiometric resolution but tend to decrease the spatial resolution.Several different methods are used to eliminate speckle noise, based upon different mathematical models of the phenomenon. The application includes four methods: Lee [1], Frost [2], GammaMAP [3] and Kuan [4].

We sum up below the basic principle of this four methods:

- Lee : Estimate the signal by mean square error minimization (MMSE) on a sliding window.
- Frost : Also derived from the MMSE criteria with a weighted sum of the values within the window. The weighting factors decrease with distance from the pixel of interest.

- GammaMAP : Derived under the assumption of the image follows a Gamma distribution.
- Kuan : Also derived from the MMSE criteria under the assumption of non stationary mean and variance. It is quite similar to Lee filter in form.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Despeckle* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
filter	Speckle filtering method	Choices
filter lee	Lee	*Choice*
filter frost	Frost	*Choice*
filter gammamap	GammaMap	*Choice*
filter kuan	Kuan	*Choice*
filter.lee.rad	Radius	Int
filter.lee.nblooks	Number of looks	Float
filter.frost.rad	Radius	Int
filter.frost.deramp	Deramp factor	Float
filter.gammamap.rad	Radius	Int
filter.gammamap.nblooks	Number of looks	Float
filter.kuan.rad	Radius	Int
filter.kuan.nblooks	Number of looks	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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 in pixel.
- **Number of looks**: Number of looks in the input image.
- **Frost**: Frost filter.
- **Radius**: Radius in pixel.
- **Deramp factor**: factor use to control the exponential function used to weight effect of the distance between the central pixel and its neighborhood. Increasing the deramp parameter will lead to take more into account pixels farther from the center and therefore increase the smoothing effects.
- **GammaMap**: GammaMap filter.
- **Radius**: Radius in pixel.
- **Number of looks**: Number of looks in the input image.

[1] Table: Parameters table for Despeckle.

- **Kuan**: Kuan filter.
- **Radius**: Radius in pixel.
- **Number of looks**: Number of looks in 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_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

The application does not handle complex image as input.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] J. Lee. Digital image enhancement and noise filtering byuse of local statistics. IEEE Transactions on Pattern Analysis and MachineIntelligence, 2:165–168, 1980.

[2] V. S. Frost, et al., A Model for Radar Images and ItsApplication to Adaptive Digital Filtering of MultiplicativeNoise, IEEE Trans. Pattern Anal., Machine Intell., vol. 4,no. 2, pp. 157-166, Mar. 1982.

[3] A. Lopes, E. Nezry, R. Touzi and H. Laur, Maximum APosteriori Speckle Filtering And First Order Texture ModelsIn Sar Images, 10thAnnual International Symposium onGeoscience and Remote Sensing, 1990,pp. 2409-2412. doi:10.1109/IGARSS.1990.689026

[4] Kuan, D. T., Sawchuk, A. A., Strand, T. C, and Chavel,P., 1987. Adaptive restoration of image with speckle. IEEETrans on Acoustic Speech and Signal Processing, 35,pp. 373-383.

DimensionalityReduction - 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 optionally export the transformation matrix to a text file.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DimensionalityReduction* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
rescale	Rescale Output.	Group
rescale.outmin	Output min value	Float
rescale.outmax	Output max value	Float
outinv	Inverse Output Image	Output image
method	Algorithm	Choices
method pca	PCA	*Choice*
method napca	NA-PCA	*Choice*
method maf	MAF	*Choice*
method ica	ICA	*Choice*
method.napca.radiusx	Set the x radius of the sliding window.	Int
method.napca.radiusy	Set the y radius of the sliding window.	Int
method.ica.iter	number of iterations	Int
method.ica.mu	Give the increment weight of W in [0, 1]	Float
nbcomp	Number of Components.	Int
normalize	Normalize.	Boolean
outmatrix	Transformation matrix output (text format)	Output File name
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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.

[1] Table: Parameters table for Dimensionality reduction.

- **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)

DomainTransform - DomainTransform

Domain Transform application for wavelet and fourier

Detailed description

Domain Transform application for wavelet and fourier

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *DomainTransform* .

[1] Table: Parameters table for DomainTransform.

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
mode	Mode	Choices
mode fft	FFT transform	*Choice*
mode wavelet	Wavelet	*Choice*
mode.fft.shift	Shift fft transform	Boolean
mode.wavelet.form	Select wavelet form	Choices
mode.wavelet.form haar	HAAR	*Choice*
mode.wavelet.form db4	DAUBECHIES4	*Choice*
mode.wavelet.form db6	DAUBECHIES6	*Choice*
mode.wavelet.form db8	DAUBECHIES8	*Choice*
mode.wavelet.form db12	DAUBECHIES12	*Choice*
mode.wavelet.form db20	DAUBECHIES20	*Choice*
mode.wavelet.form sb24	SPLINE_BIORTHOGONAL_2_4	*Choice*
mode.wavelet.form sb44	SPLINE_BIORTHOGONAL_4_4	*Choice*
mode.wavelet.form sym8	SYMLET8	*Choice*
mode.wavelet.nlevels	Number of decomposition levels	Int
direction	Direction	Choices
direction forward	Forward	*Choice*
direction inverse	Inverse	*Choice*
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input Image: This will take an input image to be transformed image. For FFT inverse transform, it expects a complex image as two-band image in which first band represent real part and second band represent imaginary part.

Output Image: This parameter holds the output file name to which transformed image will be written. This has a slightly different behaviour depending on transform type. For Wavelet, output is a single band image for both forward and inverse transform. For FFT forward transform, output is two band image where first band represents real part and second band represents imaginary part of a complex image.

Mode: This parameter allows one to select between fft(fourier) and wavelet. Available choices are:

- **FFT transform**: FFT transform.
- **Shift fft transform**: Shift transform of fft filter.
- **Wavelet**: Wavelet transform.
  - **Select wavelet form** Available choices are:
  - **HAAR**
  - **DAUBECHIES4**
  - **DAUBECHIES6**
  - **DAUBECHIES8**
  - **DAUBECHIES12**
  - **DAUBECHIES20**
  - **SPLINE_BIORTHOGONAL_2_4**
  - **SPLINE_BIORTHOGONAL_4_4**
  - **SYMLET8**
  - **Number of decomposition levels**: Number of decomposition levels.

Direction Available choices are:

- **Forward**
- **Inverse**

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_DomainTransform -in input.tif -mode.wavelet.form haar -out output_wavelet_haar.
↪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 DomainTransform application
DomainTransform = otbApplication.Registry.CreateApplication("DomainTransform")

# The following lines set all the application parameters:
DomainTransform.SetParameterString("in", "input.tif")

DomainTransform.SetParameterString("mode.wavelet.form","haar")

DomainTransform.SetParameterString("out", "output_wavelet_haar.tif")

# The following line execute the application
DomainTransform.ExecuteAndWriteOutput()
```

Limitations

This application is not streamed, check your system resources when processing large images

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

otbWaveletImageFilter, otbWaveletInverseImageFilter, otbWaveletTransform

MeanShiftSmoothing - MeanShift Smoothing

This application smooths an image using the MeanShift algorithm.

Detailed description

MeanShift [1,2,3] is an iterative edge-preserving image smoothing algorithm often used in image processing and as a first step for image segmentation. The MeanShift algorithm can be applied to multispectral images.

At first iteration, for any given pixel of the input image, the filtered value correspond to the average spectral signature of neighborhood pixels that are both spatially closer than the spatial radius parameter (spatialr) and with spectral signature that have an euclidean distance to the input pixel lower than the range radius (ranger), that is, pixels that are both close in space and in spectral signatures. Subsequent iterations will repeat this process by considering that the pixel signature corresponds to the average spectral signature computed during previous iteration, and that the pixel position corresponds to the average position of pixels used to compute the average signature.The algorithm stops when the maximum number of iterations (maxiter) is reached, or when the position and spectral signature does not change much between iterations, according to the convergence threshold (thres). If the modesearch option is used then convergence will also stops if the spatial position reaches a pixel that has already converged. This will speed-up convergence, at the expense of stability of the result.

The application outputs the image of the final averaged spectral signatures (fout), and can also optionally output the 2D displacement field between input pixel position and final pixel position after convergence (foutpos).

Note that computing an euclidean distance between spectral signatures may be inaccurate and that techniques such as color space transform or image normalisation could be applied before using this application. Also note that most satellite images noise model is not gaussian, since noise variance linearly depends on radiance (the higher the radiance, the higher the noise variance). To account for such noise model, the application provides the range radius ramp option (rangeramp), which will vary the range radius linearly with the central pixel intensity. Default value is 1. (no ramp).

This application is the first step of the large scale MeanShift method depicted in [4]. Both outputs (fout and foutpos) can be passed to the large scale MeanShift segmentation application [5]. If the application is used for large scale MeanShift, modesearch option should be off.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input Image	Input image
fout	Spectral filtered output	Output image
foutpos	Spatial filtered displacement output	Output image
ram	Available RAM (Mb)	Int
spatialr	Spatial radius	Int
ranger	Range radius	Float
thres	Mode convergence threshold	Float
maxiter	Maximum number of iterations	Int
rangeramp	Range radius ramp coefficient	Float
modesearch	Mode search.	Boolean
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

[1] Table: Parameters table for MeanShift Smoothing.

- **Input Image**: The input image can be any single or multiband image. Beware of pontential imbalance between bands ranges as it may alter euclidean distance.
- **Spectral filtered output**: This output image contains the final average spectral signatures of each pixel. The output type should be at least as wide as the input image type. Floating point encoding is advised. This output can be used as input image (in) of the LSMSSegmentation application [4,5].
- **Spatial filtered displacement output**: This output image contains the 2D displacement between the input pixel spatial position and the final position after convergence. Floating point encoding is mandatory. This output can be used as input image (in) of the LSMSSegmentation application [4,5].
- **Available RAM (Mb)**: Available memory for processing (in MB).
- **Spatial radius**: Radius of the spatial neighborhood for averaging. Higher values will result in more smoothing and higher processing time.
- **Range radius**: Threshold on spectral signature euclidean distance (expressed in radiometry unit) to consider neighborhood pixel for averaging. Higher values will be less edge-preserving (more similar to simple average in neighborhood), whereas lower values will result in less noise smoothing. Note that this parameter has no effect on processing time.
- **Mode convergence threshold**: Algorithm will stop if update of average spectral signature and spatial position is below this threshold.
- **Maximum number of iterations**: Algorithm will stop if convergence threshold is not met after the maximum number of iterations.
- **Range radius ramp coefficient**: Vary the range radius linearly with the central pixel intensity (experimental).
- **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 and the results will not be stable (see [4] for more details).
- **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 smooth.tif -foutpos position.tif -
↪spatialr 16 -ranger 16 -thres 0.1 -maxiter 100
```

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 MeanShiftSmoothing application
MeanShiftSmoothing = otbApplication.Registry.CreateApplication("MeanShiftSmoothing")

# The following lines set all the application parameters:
MeanShiftSmoothing.SetParameterString("in", "maur_rgb.png")

MeanShiftSmoothing.SetParameterString("fout", "smooth.tif")

MeanShiftSmoothing.SetParameterString("foutpos", "position.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

When modesearch is on, the application will yield slightly different results between executions, due to multi-threading. Results will also not be stable [4].

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. IEEE Transactions on pattern analysis and machine intelligence, 24(5), 603-619.

[2] Comaniciu, D., & Meer, P. (1997, June). Robust analysis of feature spaces: color image segmentation. In Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on (pp. 750-755). IEEE.

[3] Comaniciu, D., & Meer, P. (1999). Mean shift analysis and applications. In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on (Vol. 2, pp. 1197-1203). IEEE.

[4] Michel, J., Youssefi, D., & Grizonnet, M. (2015). Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images. IEEE Transactions on Geoscience and Remote Sensing, 53(2), 952-964.

[5] LSMSSegmentation application

Smoothing - Smoothing

Apply a smoothing filter to an image

Detailed description

This application applies a smoothing filter to an image. Three methodes can be used : a gaussian filter , a mean filter , or an anisotropic diffusion using the Perona-Malik algorithm.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Smoothing* .

Parameter Key	Parameter Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
type	Smoothing Type	Choices
type mean	Mean	*Choice*
type gaussian	Gaussian	*Choice*
type anidif	Anisotropic Diffusion	*Choice*
type.mean.radius	Radius	Int
type.gaussian.radius	Radius	Float
type.anidif.timestep	Time Step	Float
type.anidif.nbiter	Nb Iterations	Int
type.anidif.conductance	Conductance	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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**: Kernel's radius (in pixels).
- **Gaussian**
- **Radius**: Standard deviation of the gaussian kernel used to filter the image.
- **Anisotropic Diffusion**
- **Time Step**: Time step that will be used to discretize the diffusion equation.
- **Nb Iterations**: Number of iterations needed to get the result.
- **Conductance**: Controls the sensitivity of the conductance term in the diffusion equation. The lower it is the stronger the features will be preserved.

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
```

[1] Table: Parameters table for Smoothing.

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.SetParameterOutputImagePixelType("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.timestep 0.1 -type.anidif.nbiter 5 -type.anidif.conductance 1.5
```

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.SetParameterOutputImagePixelType("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.

Deprecated

Convert - Image Conversion

Convert an image to a different format, optionally 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). Th

The conversion can include a rescale of the data range, by default it's set from 2% to 98% of the data values. The rescale can be linear or log2. The choice of the output channels can be done with the extended filename, but less easy to handle. To do this, a 'channels' parameter allows you to select the desired bands at the output. There are 3 modes, the available choices are: * grayscale : to display mono image as standard color image * rgb : select 3 bands in the input image (multi-bands) * all : keep all bands.

Parameters

This section describes in details the parameters available for this application. Table[1] presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is *Convert* .

[1] Table: Parameters table for Image Conversion.

Parameter Key	Parameter Name	Parameter Type
in	Input image	Input image
type	Rescale type	Choices
type none	None	*Choice*
type linear	Linear	*Choice*
type log2	Log2	*Choice*
type.linear.gamma	Gamma correction factor	Float
mask	Input mask	Input image
hcp	Histogram Cutting Parameters	Group
hcp.high	High Cut Quantile	Float
hcp.low	Low Cut Quantile	Float
out	Output Image	Output image
channels	Channels selection	Choices
channels all	Default mode	*Choice*
channels grayscale	Grayscale mode	*Choice*
channels rgb	RGB composition	*Choice*
channels.grayscale.channel	Grayscale channel	Int
channels.rgb.red	Red Channel	Int
channels.rgb.green	Green Channel	Int
channels.rgb.blue	Blue Channel	Int
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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).

Output Image: Output image.

Channels selection: It's possible to select the channels of the output image. There are 3 modes, the available choices are:. Available choices are:

- **Default mode**: Select all bands in the input image, (1,...,n).
- **Grayscale mode**: Display single channel as standard color image.
- **Grayscale channel**
- **RGB composition**: Select 3 bands in the input image (multi-bands), by default (1,2,3).
- **Red Channel**: Red channel index.

- **Green Channel**: Green channel index.
- **Blue Channel**: Blue channel index.

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 -
↪type linear -channels rgb
```

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.SetParameterString("type","linear")

Convert.SetParameterString("channels","rgb")

# 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

Rescale - 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). Input minimum and maximum values is automatically computed for all image bands.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
outmin	Output min value	Float
outmax	Output max value	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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")
```

[1] Table: Parameters table for Rescale Image.

```
# The following lines set all the application parameters:
Rescale.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Rescale.SetParameterString("out", "rescaledImage.png")
Rescale.SetParameterOutputImagePixelType("out", 1)

Rescale.SetParameterFloat("outmin", 0)

Rescale.SetParameterFloat("outmax", 255)

# The following line execute the application
Rescale.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

Change Detection

MultivariateAlterationDetector - Multivariate Alteration Detector

Change detection by Multivariate Alteration Detector (MAD) algorithm

Detailed description

This application performs change detection between two multispectral images using the Multivariate Alteration Detector (MAD) [1] algorithm.

The MAD algorithm produces a set of N change maps (where N is the maximum number of bands in first and second input ima

- 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.

The application will output all change maps into a single multiband image. 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 application will also print the following information: - Mean1 and Mean2 which are the mean values of bands for both input images, - V1 and V2 which are the two linear transform that are applied to input image 1 and input image 2 to build the change map, - Rho, the vector of correlation associated to each change map.

The OTB filter used in this application has been implemented from the Matlab code kindly made available by the authors here [2]. 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.

Parameters

This section describes in details the parameters available for this application. Table[1] 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 Name	Parameter Type
in1	Input Image 1	Input image
in2	Input Image 2	Input image
out	Change Map	Output image
ram	Available RAM (Mb)	Int
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

- **Input Image 1**: Multiband image of the scene before perturbations.
- **Input Image 2**: Mutliband image of the scene after perturbations.
- **Change Map**: Multiband image containing change maps. Each map will be in range [-1,1], so a floating point output type is advised.
- **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 -out detectedChangeImage.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")
```

[1] Table: Parameters table for Multivariate Alteration Detector.

```
# The following line execute the application
MultivariateAlterationDetector.ExecuteAndWriteOutput()
```

Limitations

Input images 1 and 2 should share exactly the same origin, spacing, size, and projection if any.

Authors

This application has been written by OTB-Team.

See Also

These additional resources can be useful for further information:

[1] Nielsen, A. A., & Conradsen, K. (1997). Multivariate alterationdetection (MAD) in multispectral, bi-temporal image data: A newapproach to change detection studies.
[2] http://www2.imm.dtu.dk/~aa/software.html

Calibration

OpticalCalibration - 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(θ)) (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
- θ 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(θ))

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[1] 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 Name	Parameter Type
in	Input	Input image
out	Output	Output image
ram	Available RAM (Mb)	Int
level	Calibration Level	Choices

Continued on next page

[1] Table: Parameters table for Optical calibration.

Table 7.13 – continued from previous page

Parameter Key	Parameter Name	Parameter Type
level toa	Image to Top Of Atmosphere reflectance	*Choice*
level toatoim	TOA reflectance to Image	*Choice*
level toc	Image to Top Of Canopy reflectance (atmospheric corrections)	*Choice*
milli	Convert to milli reflectance	Boolean
clamp	Clamp of reflectivity values between [0, 1]	Boolean
acqui	Acquisition parameters	Group
acqui.minute	Minute	Int
acqui.hour	Hour	Int
acqui.day	Day	Int
acqui.month	Month	Int
acqui.year	Year	Int
acqui.fluxnormcoeff	Flux Normalization	Float
acqui.sun	Sun angles	Group
acqui.sun.elev	Sun elevation angle (deg)	Float
acqui.sun.azim	Sun azimuth angle (deg)	Float
acqui.view	Viewing angles	Group
acqui.view.elev	Viewing elevation angle (deg)	Float
acqui.view.azim	Viewing azimuth angle (deg)	Float
acqui.gainbias	Gains or biases	Input File name
acqui.solarilluminations	Solar illuminations	Input File name
atmo	Atmospheric parameters (for TOC)	Group
atmo.aerosol	Aerosol Model	Choices
atmo.aerosol noaersol	No Aerosol Model	*Choice*
atmo.aerosol continental	Continental	*Choice*
atmo.aerosol maritime	Maritime	*Choice*
atmo.aerosol urban	Urban	*Choice*
atmo.aerosol desertic	Desertic	*Choice*
atmo.oz	Ozone Amount	Float
atmo.wa	Water Vapor Amount	Float
atmo.pressure	Atmospheric Pressure	Float
atmo.opt	Aerosol Optical Thickness	Float
atmo.aeronet	Aeronet File	Input File name
atmo.rsr	Relative Spectral Response File	Input File name
atmo.radius	Window radius (adjacency effects)	Int
atmo.pixsize	Pixel size (in km)	Float
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml file	XML output parameters file

Input: Input image filename.

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, 1]: Clamping in the range [0, 1]. 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 or biases**: Gains or 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 correctionsSetting this parameters will enable the correction ofadjacency effects.
- **Pixel size (in km)**: Pixel size (in km)used tocompute adjacency effects, it doesn't have tomatch 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

SARCalibration - 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[1] 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 Name	Parameter Type
in	Input Image	Input image
out	Output Image	Output image
ram	Available RAM (Mb)	Int
noise	Disable Noise	Boolean
lut	Lookup table sigma /gamma/ beta/ DN.	Choices
lut sigma	Use sigma nought lookup	*Choice*
lut gamma	Use gamma nought lookup	*Choice*
lut beta	Use beta nought lookup	*Choice*
lut dn	Use DN value lookup	*Choice*
inxml	Load otb application from xml file	XML input parameters file
outxml	Save otb application to xml 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:

[1] Table: Parameters table for SAR Radiometric calibration.

```
#!/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")

# 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.

CHAPTER

EIGHT

FREQUENTLY ASKED QUESTIONS

Introduction

What's in OTB?

- Image access: optimized read/write access for most of remote sensing image formats, meta-data access, simple visualization;
- Sensor geometry: sensor models, cartographic projections;
- Radiometry: atmospheric corrections, vegetation indices;
- Filtering: blurring, denoising, enhancement;
- Fusion: image pansharpening;
- Feature extraction: interest points, alignments, lines;
- Image segmentation: region growing, watershed, level sets;
- Classification: K-means, SVM, Markov random fields;
- Change detection.
- Object based image analysis.
- Geospatial analysis.

For a full list of applications see the *Applications Reference Documentation*. For an introduction to the C++ API see the Software Guide. And for exhaustive descrpition of the C++ API see the Doxygen.

What is ORFEO?

ORFEO stands for Optical and Radar Federated Earth Observation. In 2001 a cooperation program was set between France and Italy to develop ORFEO, an Earth observation dual system with metric resolution: Italy is in charge of COSMO-Skymed the radar component development, and France of PLEIADES the optic component.

The PLEIADES optic component is composed of two "small satellites" (mass of one ton) offering a spatial resolution at nadir of 0.7 m and a field of view of 20 km. Their great agility enables a daily access all over the world, essentially for defense and civil security applications, and a coverage capacity necessary for the cartography kind of applications at scales better than those accessible to SPOT family satellites. Moreover, PLEIADES have stereoscopic acquisition capacity to meet the fine cartography needs, notably in urban regions, and to bring more information when used with aerial photography.

The ORFEO "targeted" acquisition capacities made it a system particularly adapted to defense or civil security missions, as well as critical geophysical phenomena survey such as volcanic eruptions, which require a priority use of the system resources.

With respect to the constraints of the Franco-Italian agreement, cooperation have been set up for the PLEIADES optical component with Sweden, Belgium, Spain and Austria.

Where can I get more information about ORFEO?

At the PLEIADES HR web site: http://smsc.cnes.fr/PLEIADES/.

What is the ORFEO Accompaniment Program?

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

The creation of a preparatory program is needed because of:

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

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

- A Methodological part,
- A Thematic part.

This Accompaniment Program uses simulated data (acquired during airborne campaigns) and satellite images quite similar to Pleiades (as QuickBird and Ikonos), used in a communal way on a set of special sites. The validation of specified products and services will be realized with Pleiades data

Apart from the initial cooperation with Italy, the ORFEO Accompaniment Program enlarged to Belgium, with integration of Belgian experts in the different WG as well as a participation to the methodological part.

Where can I get more information about the ORFEO Accompaniment Program?

Go to the following web site: http://smsc.cnes.fr/PLEIADES/A_prog_accomp.htm.

Who is responsible for OTB's development?

The French Centre National d'Études Spatiales, CNES, initiated the ORFEO Toolbox and is responsible for the specification of the library. CNES funds the industrial development contracts and research contracts needed for the evolution of OTB.

License

What is OTB's license?

OTB is distributed under the permissive open source license Apache v2.0 - aka Apache Software License (ASL) v2.0: http://www.apache.org/licenses/LICENSE-2.0

Am I forced to distribute my code based on OTB?

No. The license gives you the option to distribute your application if you want to. You do not have to exercise this option in the license.

Am I forced to contribute my code based on OTB into the official repo?

No.

If I wanted to distribute an application using OTB what license would I need to use?

The license of your choice. The OTB license only requires you to include a copy of the Apache license and to provide a clear attribution to the OTB project in any distribution including a piece of OTB software.

I am a commercial user. Is there any restriction on the use of OTB?

No. The OTB license only requires you to include a copy of the Apache license and to provide a clear attribution to the OTB project in any distribution including a piece of OTB software.

Getting OTB

Who can download OTB?

Anybody can download OTB at no cost.

Where can I download OTB?

Go to http://www.orfeo-toolbox.org and follow the "download OTB" link. You will have access to the OTB source code, to the Software User's Guide and to the Cookbook of the last release. Binary packages are also provided for the current version. OTB and Monteverdi are also integrated in OSGeo-Live since version 4.5. You can find more information about the project at http://live.osgeo.org/. Moreover you can found the last release of Monteverdi and OTB applications through the OSGeo4W installer.

How to get the latest bleeding-edge version?

You can get the current development version, as our repository is public, using Git (available at http://git-scm.com). Be aware that, even if the golden rule is *what is committed will compile*, this is not always the case. Changes are usually more than ten per day.

The first time, you can get the source code using:

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

Then you can build OTB as usual using this directory as the source (refer to build instructions). Later if you want to update your source, from OTB's source directory, just do:

```
git pull
```

A simple `make` in your OTB binary directory will be enough to update the library (recompiling only the necessary files).

Special issues about compiling OTB from source

All information about OTB compilation can be found in the Software Guide. We present here only the special issues which can be encountered.

Debian Linux / Ubuntu

On some Debian and Ubuntu versions, the system GDAL library and its tiff internal symbol might conflict with the system Tiff library (bugs.debian.org/558733). This is most likely the case if you get odd segmentation fault whenever trying to open a tiff image. This symbol clash happens when using OTB. A workaround to the issue has been provided in GDAL sources, but is available in the 1.9.0 release.

The recommended procedure is to get this recent source and build GDAL from sources, with the following configure command:

```
./configure --prefix=INSTALL_DIR --with-libtiff=internal
            --with-geotiff=internal
            --with-rename-internal-libtiff-symbols=yes
            --with-rename-internal-libgeotiff-symbol=yes
```

Errors when compiling internal libkml

The internal version of libkml cannot be compiled when using an external build of ITK. See http://bugs.orfeo-toolbox.org/view.php?id=879 for more details.

To workaround the problem, either use an external build of libkml (it is provided on most systems), or use an internal build of ITK by setting to OFF the CMake variable OTB_USE_EXTERNAL_ITK.

OTB compilation and Windows platform

To build OTB on Windows, you should prepare an environment with the following tools:

- Visual Studio 2015 or later
- CMake 3.1 or later
- OTB XDK : download a Windows binary package of OTB and use the supplied uninstall script to remove OTB binaries and headers. Now, this package only contains the dependencies needed to build OTB.

Then, you can download OTB sources (preferably, a version compatible with your XDK), and compile them as a standard CMake project. More details are available in the SoftwareGuide.

There is an other solution, using OSGeo4W distribution. However, the dependencies may be outdated.

Using OTB

What is the image size limitation of OTB ?

The maximum physical space a user can allocate depends on her platform. Therefore, image allocation in OTB is restricted by image dimension, size, pixel type and number of bands.

Fortunately, thanks to the streaming mechanism implemented within OTB's pipeline (actually ITK's), this limitation can be bypassed. The use of the at the end of the pipeline, will seamlessly break the large, problematic data into small pieces whose allocation is possible. These pieces are processed one after the other, so that there is not allocation problem anymore. We are often working with images of 25000×25000 pixels.

For the streaming to work, all the filters in the pipeline must be streaming capable (this is the case for most of the filters in OTB). The output image format also need to be streamable (not PNG or JPEG, but TIFF or ENVI formats, for instance).

The class manage the steaming process following two strategies: by tile or by strip. Different size configuration for these two strategies are available into the interface. The default mode use the information about how the file is streamed on the disk and will try to minimize the memory consumption along the pipeline. More information can be found into the documentation of the class.

Problems using OTB python wrapping along with other software

If you use OTB standalone binaries, there should not be any dependency conflict with other libraries installed on your system. OTB will always try to grab supplied libraries in the standalone package.

However, when using Python wrappings, there can be conflicts if you import *otbApplications* along with other software that share common dependencies with OTB. For instance, if you want to use OTB Applications and Fiona in a Python script, they both rely on GDAL library. As the libraries loaded by Python must be unique, the first library *SomeLib* loaded will be used by any other binary depending on it. Thus, the order of the imports has an effect. In some cases, symbol problems have been observed in libcrypto, and the solution was to import OTB Applications before importing Fiona.

Getting help

Is there any mailing list?

Yes. There is a discussion group at http://groups.google.com/group/otb-users/ where you can get help on the set up and the use of OTB.

Which is the main source of documentation?

The main source of documentation is the CookBook located at https://www.orfeo-toolbox.org/CookBook/.

Secondly there is the OTB Software Guide which can be found at https://www.orfeo-toolbox.org/SoftwareGuide/ It contains many examples and a tutorial which should be a good starting point for any new OTB user. The code source

for these examples is distributed with the toolbox. Another information source is the on-line API documentation which is available at http://www.orfeo-toolbox.org/doxygen.

You can also find some information about how to use Monteverdi and the OTB-Applications into the Cookbook at http://www.orfeo-toolbox.org/CookBook/.

Contributing to OTB

I want to contribute to OTB, where to begin?

There are many ways to join us in the OTB adventure. The more people contribute, the better the library is for everybody!

First, you can send an email to the user mailing list (otb-users@googlegroups.com) to let us know what functionality you would like to introduce in OTB. If the functionality seems important for OTB users, we will then discuss on how to retrieve your code, make the necessary adaptions, check with you that the results are correct and finally include it in the next release.

You can also run the nightly tests so we have a wider range of platforms to detect bugs earlier.

You can also find more information about how to contribute at https://www.orfeo-toolbox.org/community

What are the benefits of contributing to OTB?

Besides the satisfaction of contributing to an open source project, we will include the references to relevant papers in the software guide. Having algorithms published in the form of reproducible research helps science move faster and encourages people who needs your algorithms to use them.

You will also benefit from the strengths of OTB: multi-platform, streaming and threading, etc.

What functionality can I contribute?

All functionalities which are useful for remote sensing data are of interest. As OTB is a library, it should be generic algorithms: change, detection, fusion, object detection, segmentation, interpolation, etc.

More specific applications can be contributed using the framework directly in the Applications directory of OTB.

Running the tests

What are the tests?

OTB is an ever changing library, it is quite active and have scores of changes per day from different people. It would be a headache to make sure that the brand new improvement that you introduced didn't break anything, if we didn't have automated tests. You also have to take into account differences in OS, compilers, options, versions of external libraries, etc. By running the tests and submitting it to the dashboard, you will help us detect problems and fix them early.

For each class, at minimum there is a test which tries to instantiate it and another one which uses the class. The output of each test (image, text file, binary file) is controlled against a baseline to make sure that the result hasn't changed.

All OTB tests source code are available in the directory `Testing` and are also good examples on how to use the different classes.

How to run the tests?

There is more than 2500 tests for OTB and it takes from 20 minutes to 3 hours to run all the test, mainly depending on your compilation options (Release mode does make a difference) and of course your hardware.

To run the tests, you first have to make sure that you set the option `BUILD_TESTING` to `ON` before building the library. If you want to modify it, just rerun ccmake, change the option, then make.

For some of the tests, you also need the test data and the baselines (see [sec:FAQTestData]).

Once OTB is built with the tests, you just have to go to the binary directory where you built OTB and run `ctest -N` to have a list of all the tests. Just using `ctest` will run all the tests. To select a subset, you can do `ctest -R Kml` to run all tests related to kml files or `ctest -I 1,10` to run tests from 1 to 10.

How to get the test data?

Data used for the tests are also versioned using Git (see [sec:FAQGit]).

You can get the base doing:

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

This is about 1 GB of data, so it will take a while, but you have to do it only once, as after, a simple

```
git pull
```

will update you to the latest version of the repository.

You can also easily synchronize the directory you retrieve between different computers on your network, so you don't have to get it several times from the main server. Check out Git capabilities.

How to submit the results?

Once you know how to run the tests, you can also help us to detect the bugs or configuration problems specific to your configuration. As mentioned before, the possible combinations between OS, compiler, options, external libraries version is too big to be tested completely, but the more the better.

You just have to launch ctest with the `-D Experimental` switch. Hence:

```
ctest -D Experimental -A CMakeCache.txt
```

And you will be able to see the result at

http://dash.orfeo-toolbox.org/Dashboard/index.php?project=OTB.

If you are interested in setting up a nightly test (automatically launched every night), please contact us and we will give you the details.

What features will the OTB include and when?

There is no detailed plan about the availability of OTB new features, since OTB's content depends on ongoing research work and on feedback from thematic users of the ORFEO Accompaniment Program. You can find ideas and plans for the future on the Wishlist at https://wiki.orfeo-toolbox.org/index.php/Wishlist.