Guide for OTB training courses

OTB team

2018

Contents

1 Foreword

1.1 Goals

The goal of this training course is to give an overview of the OTB remote sensing image processing library and to apply it to real case problems using OTB applications, which make the use of remote sensing data easy.

The course allows to acquire the ability to design and create remote sensing image processing chains using OTB applications, including:

- Feature extraction
- Calibration
- Classification
- Segmentation
- Synthetic Aperture Radar processing
- OTB Application Python API

1.2 Training kit

1.2.1 Data package

- Data used in all exercises (sub-folder for each exercise)
- Data folder is specified at the beginning of each exercise
- Download from: `www.orfeo-toolbox.org/packages/WorkshopData/WorkshopData.zip`

1.2.2 Guides

- Training guide
- Slides
- Installation guide
- Evaluation survey
- Solutions (at the end of the training session)

1.2.3 Software documentation

Software Guide C++ API (with algorithms definition)

`www.orfeo-toolbox.org/SoftwareGuide/index.html`

CookBook Guide for non developers (API of applications)

`www.orfeo-toolbox.org/CookBook/`

QGIS User Guide `docs.qgis.org/2.14/en/docs/user_manual`

1.3 Datasets

Sentinel-2 concatenation of several dates, cloud free (gap-filled) with ground truth (reference data) in ESRI Shapefile format (classification exercise)

Pléiades PHR Bundle PRIMARY Level 1A from the CNES Thematic Commissioning phase over OSR MiPy (Toulouse, France) acquired in November 2013 (©CNES (2013), distribution Airbus DS/ Spot Image),

Sentinel-1 SLC product (complex) SM (strip Map, 5m ground resolution), polarimetric (HH and HV) over the South of the Constance lake (Germany).

1.4 Software requirements

To complete exercises, you need to install the following tools:

- **Orfeo ToolBox** >= 6.2 with applications
- **Monteverdi** >= 3.2
- **QGIS** >= 2.8

To install the **Orfeo ToolBox** and **Monteverdi**, follow the instructions in the ORFEO ToolBox cookbook.

To install **QGIS** follow the instructions on the QGIS website.

1.5 Datasets

Datasets used during the training courses:

Sentinel-2 9 dates concatenated, cloud-free (gap-filled) with reference data in ESRI Shapefile provided by CESBIO (for the classification exercise),

Pléiades Bundle PRIMARY Level 1A from CNES Thematic Comissionning Phase over OSR MiPy (South West of Toulouse, France), 2013 (©CNES (2013), distribution Airbus DS/ Spot Image),

Sentinel-1 Single Look Complex (SLC) product - Strip Map (80 km swath, 5x5 m res), dual-polarization (HH and HV) near Constance lake (Germany).

For more information on these satellites:

Sentinel-2 level 2A available on THEIA website.

Pléiades access for public French organisms

Sentinel-1 Free access on ESA hub or on CNES PEPS

2 Basics

2.1 Use Monteverdi and QGIS

2.1.1 Description

Summary The goal is to discover software products and the data package that will be used during the course. You will be able to use both Monteverdi and QGIS to do all exercises of the course.

Prerequisites

- Install Monteverdi and QGIS
- Download data

Goals Goals of this exercise

- Know how to display an image and set up rendering in Monteverdi,
- Know how to display an image stack in Monteverdi,
- Know how to display an image and set up rendering in QGIS,
- Know how to display vector data in QGIS,
- Know how to display an image stack in QGIS,
- Tips and tricks.

2.1.2 Steps

For this exercise, we propose to work in pairs. At the end of the exercise every team will do a restitution of their experiments to show to other groups some tips they found and ask also some questions about functions available in both software products.

Display an image in Monteverdi Open the image `phr_xs_osr_mipy.tif` available in the `Data/preprocessing` directory with Monteverdi.

The list of all keyboard shortcuts is available in *Help* menu.

Change the displayed band and also change the rendering values.

Test effects like *Local contrast*, *Gradient* and *Spectral angle*. What is their purpose?

Change the zoom level by using the mouse wheel, you can edit also the image scale in the bottom in the status bar.

Close Monteverdi.

Display an image stack in Monteverdi Open the 5 Sentinel-2 images in the `Data/classification/images/` folder.

Change the bands to display natural colors (red: Band 3, green: band 2, blue: band 1). Use the *Apply all parameters to all layers* button to have the same rendering options on the whole stack.

Use *CTRL* + *mouse wheel* to cycle between dates.

Test effects *Local transparency*, *Chess board* and *Swipe*. What is their purpose?

Close Monteverdi.

Display an image in QGIS Open the first Sentinel-2 image in QGIS. Use the contextual menu of the layer to change color dynamics to have the same rendering as in Monteverdi. In the same contextual menu, add this style to be able to apply it to other images.

Move in resolution.

Open other Sentinel-2 images and apply the rendering style you've just created.

What are the differences between Monteverdi and QGIS in terms of image visualization and rendering functions ?

Display vector data in QGIS Open `training.shp` in the `Data/classification/references/training/` folder.

Use the contextual menu to open the *Properties* and *Style* tabs. Use *Categorised* using column *LC*. How many classes are there in the file? Change the color table to display different classes in different colors.

Open the attribute table. How many polygons are contained in the class *pelouse* (*lawn*)?

Try to find how to select and display only polygons from this class.

2.2 The Orfeo ToolBox Application mechanism

2.2.1 Description

Summary During this exercise, we will learn how to use the Orfeo Toolbox applications. Images including messages encoded by steganography are available. You must use the OTB applications to reveal this hidden message for each image.

All images used during this exercise are modified extracts from Pléiades images.

Prerequisites

- Software installed (Monteverdi and Orfeo ToolBox)
- Data downloaded

Objectives

- Know how to look for an application in the list of available applications
- Know how to set application parameters
- Know where the documentation of an application is
- Know how to use several classical applications.

2.2.2 Steps

Data are located in the `Data/stegano` folder.

For each message, one will first look at the image to try to detect the message, then try to use suggested applications to reveal the message.

Message 1 In `image1.tif` image, a sentence has been encoded in a high signal area, using a pixel value which is outside the possible range for Pléiades (reminder: Pléiades images are encoded with unsigned 12 bits integers).

Use the **BandMath** application to detect those out of range values and thus reveal the message.

Message 2 In `image2.tif` image, a sentence has been encoded within a homogeneous, low signal area. Modified pixels can not be seen with human eyes, but could be revealed by computing the image gradient, or some edge detection.

Use the **EdgeExtraction** application to reveal the message.

Message 3 In `image3.tif` image, a sentence has been encoded by slightly modifying the pixel values in the red and near infra-red bands. This modification can not be seen with human eyes, but could be revealed by computing an NDVI radiometric index.

Use the **RadiometricIndices** application to reveal the message.

You can also use the **BandMath** application to compute the NDVI index using the following formula:

$$NDVI = \frac{NIR-RED}{NIR+RED}$$

Reminder: For Pléiades images the red band is the first one, and the NIR band is the last one.

Message 4 In `image4.tif`, a message has been hidden in the 2 least significant bits of the image. This modification can not be detected by human eyes, but could be revealed by isolating the values of those 2 bits.

Use the **BandMath** application to isolate the 2 least significant bits in the image (encoded on 12 bits), to reveal the message.

Note: The rint() function allows to round a floating point value to nearest integer in **BandMath** application.

Message 5 In image `image5.tif`, a message has been dissimulated by locally slightly modifying the image noise. It could be revealed by a transform that isolates noise.

Use the **DimensionalityReduction** application to isolate the image noise and reveal the message.

You can also try to figure out other techniques using the applications to highlight this local modification of the image noise.

Message 6 In `image6.tif` image, a message has been hidden by locally using a gray-level morphological operation (opening with radius=1). It could be revealed by using the idempotent property of this transform. A function f is said idempotent if:

$$f(f(x)) = f(x)$$

Use the **GrayscaleMorphologicalOperation** and **BandMath** applications to reveal the message by using this idempotent property.

2.2.3 Going further

What messages were you able to detect by analyzing the image with Monteverdi? What messages were impossible to detect?

Can you imagine another process to encode hidden messages in images? An image (`image.tif`) and a message (`message.tif`) are provided in the `Data/stegano/diy` folder for you to try.

2.3 Orfeo ToolBox internals

2.3.1 Description

Summary This exercise introduces Orfeo the ToolBox pipeline internals.

- Extended filenames,
- Streaming,
- Multi-threading,
- Environment variables,
- `geom` files.

Prerequisites

- Installed Monteverdi and Orfeo ToolBox software
- Downloaded data
- Understand Orfeo Toolbox applications (see relevant exercise)

Goal

- Understand OTB's transparent machinery
- Influence the data processing pipeline
- Know where to find important information

2.3.2 Steps

The data are in `Data/internals/` folder.

Encoding images With Orfeo ToolBox, the user chooses the image encoding type depending on the image. In general we'll use:

- 8 bits unsigned encoding (with domain [0,255]) for outputs used on screen or paper.
- 16 bits unsigned encoding to represent most satellite images
- 32 bits unsigned encoding to represent intermediate processing steps, computation results with real numbers (e.g. NDVI, radiometric calibration...)

In Orfeo ToolBox applications, the user can choose the image encoding (drop down menu in the GUI interface, or parameter in the command line).

type	domain	number of bits
uint8	[0,255]	8 bits
int16	[32 767, +32 767]	16 bits
uint16	[0, 65 535]	16 bits
int32	[2 147 483 647, +2 147 483 647]	32 bits
uint32	[0, 4 294 967 294]	32 bits
float	[-3.402823×10^{38},3.402823×10^{38}]	32 bits
double	[-10^{308}, 10^{308}]	64 bits

Use **gdalinfo** to know the pixel encoding of `image1.tif` (you can also access it from QGIS). Analyse pixel values of the image in Monteverdi. What can you conclude?

Use the **Convert** application to convert `image1.tif` to a 16 bits integer encoding. Compare both image file sizes. Use the **CompareImages** application to compare the content of both images. What can you conclude? Can you reduce the file size further? (See the extended filenames paragraph).

Use the **RadiometricIndices** application to compute an NDVI (Normalized Difference Vegetation Index) from the 16 bits image. Keep the output encoding to 16 bits. Visualize the result. What do you notice? Which encoding should be used to properly store this image?

.geom files Look into the `image1.geom` file with a text editor. What seems to be its purpose? Which processing operations require this type of information?

Extended filenames Extended filenames are used to influence the image reading and writing process in Orfeo ToolBox. They are not application specific, but can be used with any Orfeo ToolBox based tool.

The full list of options is listed in the section *Extended filename for reader and writer* Software Guide. We will illustrate here a few of them.


Read options Compare the following command outputs:

```
$ otbcli_ReadImageInfo -in "image1.tif"
$ otbcli_ReadImageInfo -in "image1.tif?&skipgeom=true"
```

What is the effect of the *skipgeom* parameter? Note the similar *skipcarto* which allows to ignore the coarse cartographic projection in the case of *Ortho Ready* type products (projected onto an average altitude by default).

Compare the following command outputs:

```
$ otbcli_ReadImageInfo -in "image2.tif"
$ otbcli_ReadImageInfo -in "image2.tif?&geom=image1.geom"
```

What is the effect of the *geom* parameter? What can be its purpose?

Write options Among available write options from extended filenames, the *gdal:co* option allows to pass file creation options directly to GDAL. With this option, and with the Tif file format options available in GDAL, re-encode `image1.tif` to signed 12 bits with LZW compression.

Compare file sizes and image contents with the **CompareImages** application.

Another useful extended filename option is *box*. Use the *box* option to write only to a 100x100 pixels square in the center of the image.

Streaming management By default, Orfeo ToolBox chooses the tiling method most suitable to the input image, and the block size which maximizes memory usage. Memory usage is specified to the application, or with the `OTB_MAX_RAM_HINT` environment variable. However, it is possible to modify this behavior with writer extended filenames.

1. Use the **LocalStatisticsExtraction** application to process the first band of `image1.tif` with a radius of 9.
2. Execute the above operation a second time, but this time completely disable streaming with the *streaming:type* extended filename option. What do you notice?
3. Execute the above operation a third time, but this time ask for a split of 1000 strips. Use the *streaming:type*, *streaming:sizemode* and *streaming:sizevalue*. What do you notice?

To observe these effects, it is recommended to open a system monitor tool and look at CPU load and I/O throughput.

Multi-threading By default, every filter in Orfeo ToolBox uses all available processing cores. This behavior can be altered with the `ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS` environment variable.

Follow the previous example with the **LocalStatisticsExtraction** application and disable streaming. Use the above environment variable to decrease, and then increase the number of threads. What do you notice?

You can use the *time* command to measure execution time.

3 Optical VHR image, from pre-processing to GIS

3.1 Preprocessing of Very High Resolution optical imagery

3.1.1 Description

Summary This exercise allows to get familiar with radiometric and geometric corrections using OTB applications.

Prerequisites

- Basic knowledge of remote sensing imagery
- Basic knowledge on how to use OTB applications

Goal

Know how to perform optical calibration

- Allow to compare and validate values between images and sensors (multi-temp) and multi-sensors
- Convert DN into TOA (Top of Atmosphere) reflectance and TOC (Top of Canopy)

Know how to perform image fusion (pan-sharpening)

- Pan (higher spatial resolution) and XS (higher spectral resolution)
- Test several fusion methods
- Special modes available for PHR Pleiades image fusion (Pléiades *mode*)
- See effects of image fusion (moving objects, radiometric calibration)

Know how to perform orthorectification

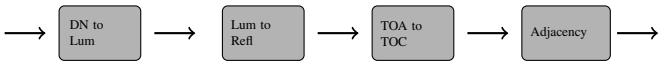
- Spatially locate satellite images
- Associate geographical coordinates to pixel positions
- Know prerequisites; bias, approximations

3.1.2 Steps

Data are located in the `Data/preprocessing/` folder. We will also use the `SRTM` and `Geoid` folders.

In this part of the exercise, you will use the `phr_xs_osr_mipy.tif` and `phr_pan_osr_mipy.tif` images. Those are extracts of PAN and XS from a Pleiades product (Bundle Primary Level 1A) acquired over the south west of Toulouse.

Atmospheric corrections The goal is to convert Digital Numbers into reflectances (physical values). The process is composed of 2 steps: computation of Top Of Atmosphere reflectance and then surface reflectance which consists in taking into account atmospheric effects.



With `phr_xs_osr_mipy.tif`:

1. Use the **OpticalCalibration** application to compute TOA reflectance.
2. Use the **OpticalCalibration** application to compute surface reflectance (TOC, top of canopy).
3. Compare both images using Monteverdi or another OTB application (TOA-TOC). Compare the red, green and blue (B0,B1,B2) bands of the TOA and TOC images. Which band is more impacted by atmospheric effects ?

   Note: The application BandMathX can apply one operation to all the bands simultaneously.
4. Apply operations 1, 2 and optionally 3 to the panchromatic image `phr_pan_osr_mipy.tif`.

Tips :

- Use '-milli' option which allows to save the output image in integer (16 bits). By default reflectance images are saved as float values (between 0 and 1).

P+XS Fusion The goal of the exercise is to create a pan-sharpened image from the PAN and XS bundle. Physical constraints on sensor and telescope design do not allow to have at the same time high spatial and spectral resolutions. Indeed, the reduction of the sampling is accompanied by a decrease of the received signal, so the SNR decreases. This is compensated by increasing the diameter of the entrance pupil or by using specific detectors to charge accumulation (TDI) and also varying the width of the spectral band. There is also a constraint on the amount of data that can be archived in the satellite memory. Finally, the bandwidth of the downlink to send the image to the ground is also limited.

As a consequence most VHR sensors deliver 2 types of images:

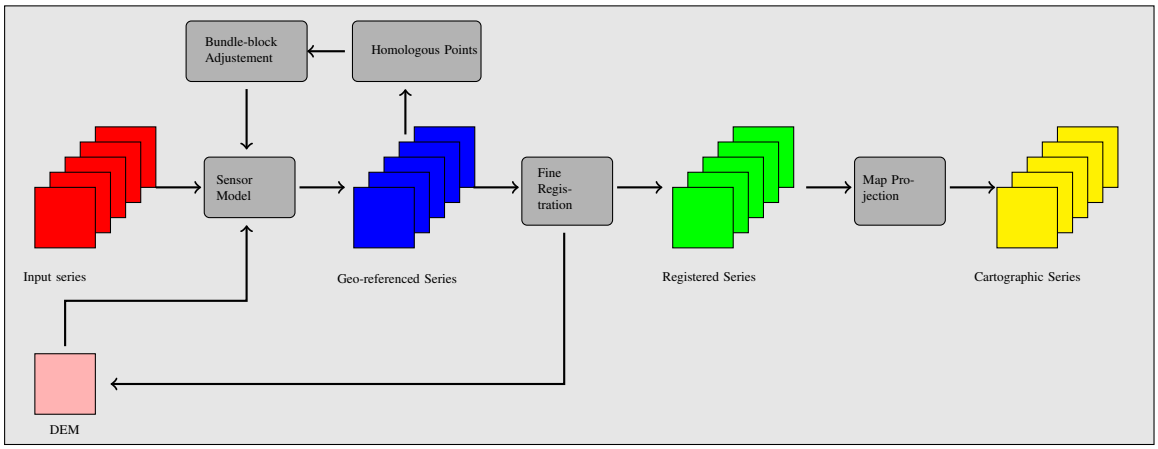
- Multi-spectral (XS): separate spectral bands each on a spectral range (can overlap). For Pléiades, 4 bands (B,G,R,NIR) with a spatial resolution of 2.8m (resampled to 2m).
- Panchromatic (PAN): grey level image with a detector which covers a larger spectral range (for an improved SNR) which allows to acquire images at 0.7m in the case of Pléiades (resampled to 0.5m).

We will perform pan-sharpening using TOA reflectance PAN and XS images (`phr_pan_osr_mipy_toa.tif` and `phr_xs_osr_mipy_toa.tif`)

1. Use the **BundleToPerfectSensor** application to superimpose and fuse the PAN and XS images. Note that the application has a *phr* mode (Pléiades) which allows to perform image registration without the need of sensor model parameters (default mode). Indeed, the PHR mode takes advantage of the fact that Pléiades bundle are colocalised on the same grid.
2. Which fusion algorithm is used in the **BundleToPerfectSensor** application?
3. (optional) Use applications **Superimpose** and **Pansharpening** to perform pan-sharpening with other fusion methods available in OTB.

Orthorectification This operation allows to associate a ground coordinate (geographical position) to every pixel in the image.

The schema below describes all steps that can be required to go from a set of Level 1 products to a coregistered and colocalized image stack.



These steps can vary depending on the sensor and the product levels.

In the exercise we will only perform image orthorectification of the pan-sharpened product to discover the modes available in OTB.

Use the **Orthorectification** application to perform the orthorectification:

1. Without Digital Elevation Model (DEM)
2. With a DEM (use the `SRTM` directory) and a geoid file (`Geoid/egm96.grd`)

3. Compare the 2 images in Monteverdi. What do you notice?
4. What is the projection system by default in the **Orthorectification** application?
5. In which UTM zone is located the Pleiades image?
6. Test different projection systems (WGS84, Lambert 93) and open the ortho image in QGIS.

3.2 Image segmentation and export to GIS

3.2.1 Description

Summary This exercise allows to get familiar with large scale segmentation using the MeanShift algorithm. It goes through each step of the procedure and shows how to export and use output segments in a GIS.

Pre-requisites

- Installed software (Monteverdi and QGIS)
- Output data from the VHR image pre-processing exercise
- Know how OTB applications work (see corresponding exercise)

Goals

- Know the steps required to perform a segmentation
- Know how to optimize segmentation parameters
- Know how to export results to a GIS

3.2.2 Steps

Data needed for this exercise can be found in the `Data/segmentation` folder.

Image smoothing with the MeanShift algorithm The MeanShift algorithm performs a smoothing of the image by preserving edges. It produces more homogeneous images and also keeps sharp boundaries between regions. It aims at facilitating the segmentation step done by simple methods like connected component computation. The algorithm performs iterative mean over pixels using a given neighborhood (spatial and spectral).

Compute the smoothing of the image `phr_orthopxs_osr_mipy_xt.tif` using the application **MeanShiftSmoothing**. Change the values of the *ranger* and *spatialr* parameters to see their influence.

Notes:

- the *modesearch* option allows to speedup computation but it is less stable. We will disable it in this exercise.
- Spatial (*spatialr*) and spectral (*ranger*) radiuses are the most important parameters of the algorithm. The first one corresponds to a number of pixels and the other one corresponds to a distance between pixels in the spectral domain.
- *foutpos* output the spectral mean result for each pixel and will be used in the next questions.
- You can use the *extended filename* parameter *box* to find the optimal parameters *ranger* and *spatialr* without processing all the input image.

  In the next step of the exercise we will use *ranger* around 25 and *spatialr* around 3 (to limit computation time).

Segmentation This step allows to produce the initial segmentation using the output images from the smoothing step (*fout* and *foutpos*). Adjacent pixels in the *fout* image for which the spectral distance is lower than *ranger* and for which the estimated position in *foutpos* is lower than *spatialr* will be grouped in the same connected component (segment).

This step is performed on a tile basis. The application allows to set the tile size with parameters *tilesizex* and *tilesizey*.

Finally, *minsize* allows to remove small regions (the threshold corresponds to a number of pixels).

Use the **LSMSSegmentation** application to perform this step. Some tips regarding the application configuration:

- Use *minsize* equal to 0 in this first step
- The output image is a labelled image in which each pixel corresponds to an integer value (unique for each segment). It is advisable to encode the output image in *uint32* to have enough unique labels. Indeed the segmentation can generate a large number of segments.
- Parameters *ranger* and *spatialr* should have smaller values than in the smoothing step. You can start by dividing the value by two.

Use the *optimal* mode from the **ColorMapping** application to generate a nice output from the labelled image.

Handle small regions After the initial segmentation, there is generally lots of small regions which do not correspond to any object or region of interest. The **LSMSSegmentation** application allows to filter those regions (*minsize* parameter), but it is also possible to add a post-processing to fuse iterative adjacent regions (using a spectral criteria). This post-processing can be done with **LSMSSmallRegionsMerging**.

Use this application to re-process the initial segmentation by fusing all regions smaller than 50 pixels.

Use the *optimal* mode from the **ColorMapping** application to generate a nice output from the labelled image.

Vectorization The last step allows to export polygons to a vector file. The **LSMSVectorization** application performs the vectorization and computes the mean and the variance on each segment using a support image.

Start by computing the NDVI from input image.

Then, use the **ConcatenateImages** application to stack the input radiometry with the NDVI (be careful with the encoding of the output image).

Use the **LSMSVectorization** using this image and the segmentation results. You can choose for instance *ESRI Shapefile* (.shp extension) or sqlite (.sqlite extension) as the vector data format (all formats supported by GDAL/OGR can be used here).

Open the input image and the vector layer in QGIS. Adapt the vector rendering to display polygon edges in red and the interior ring with 100% transparency. Analyze the segmentation and comment the result.

Open and comment the attribute table.

Filter segments in QGIS Use the selection tool in QGIS using SQL expressions and the layer attributes table to select polygons which do not correspond to shadow areas and copy them in another layer.

Use the calculator on attributes to create a new column which will store polygon's compacity:

$$compactness = \frac{\sqrt{area}}{perimeter}$$

Use the attribute selector interface to select compact polygons with a high NDVI value (trees?). Copy the result to a new layer in QGIS.

4 Supervised classification of a satellite image time series

4.1 Description

4.1.1 Summary

This exercise allows to get familiar with supervised, pixel-wise classification applications in Orfeo ToolBox, using a Sentinel-2 time series and a reference dataset for supervision.

4.1.2 Pre-requisites

- Software installed (Monteverdi and Orfeo ToolBox)
- Data downloaded
- Basic knowledge on using applications in OTB
- Basic knowledge on supervised classification

4.1.3 Objectives

Objectives are the following:

- Knowing the different applications of the supervised classification process
- Using different learning algorithms
- Measuring classification performances
- Post-processing of classification

4.2 Steps

Data are available in the `Data/classification` folder, with following sub-folders:

- `images` contains the Sentinel-2 time series,
- `references/training` contains training data in *shp* format,
- `references/testing` contains testing data in *shp* format

4.2.1 Introduction to Sentinel-2 data

In the data package, folder `Data/classification/images` contains 5 Sentinel-2 images, extracted on tile T31TCJ, at the following dates:

2016-06-07
2016-07-07
2016-08-06
2016-09-05
2016-10-05

Those images are multispectral, with 10 bands resampled at 20 m:

#	Band name	S2 band id	Wavelength	Initial resolution
0	Blue	B2	490 nm	10 m
1	Green	B3	560 nm	10 m
2	Red	B4	665 nm	10 m
3	NIR - Narrow 1	B5	705 nm	20 m
4	NIR - Narrow 2	B6	740 nm	20 m
5	NIR - Narrow 3	B7	783 nm	20 m
6	NIR - Wide	B8	842 nm	10 m
7	NIR - Narrow 4	B8A	865 nm	20 m
8	SWIR 1	B11	1610 nm	20 m
9	SWIR 2	B12	2190 nm	20 m

We therefore have 50 bands for each pixels. Images are encoded over 16 bits.

Open one image in monteverdi and set the bands for a true color display (red, green, blue)

Open the remaining four images and look for changes.

Note: The QGIS style file `support/images.qml` can be loaded into QGIS to set the rendering and color channels for each image.

Files `references/training/training.shp` and `references/testing/testing.shp` contain polygons defining 13 classes over the scene:

Code	Name	#polygons training	#polygons testing
10	Annual crops	3129	3078
31	Deciduous Forests	176	292
32	Evergreen Forests	23	29
34	Grass	2	2
36	Woody Moorlands	63	38
41	Dense Urban Area	30	33
42	Light Urban Area	326	239
43	Industrial Area	154	212
44	Roads	162	114
51	Water	243	332
211	Meadow	320	311
221	Orchards	227	254
222	Vineyards	129	97

Open one of the files in QGIS. The attribute table can be accessed by right-clicking on the layer -> *Open attributes table*. Each label is visible, and the list can be filtered with SQL expressions.

Note : There is a style file `support/polygons.qml` that can be loaded into QGIS to colorize polygons according to their classes.

Polygons are split into two sets: training and validation.

4.2.2 Single date training

We are going to use the **TrainImagesClassifier** application in order to do supervised learning from the training date in `references/training/training.shp`. First, we are going to work with the image from the 07.06.2016.

The **TrainImagesClassifier** application will sample some image pixels within the training polygons, in order to build a well-balanced training set. This set is then passed to the learning algorithm.

This application has some mandatory parameters:

- The input images, which bands will be used as features by the learning algorithm,
- The vector layer containing references polygons,
- The name of the field in that layer that contains the class identifier,

- The output file containing the learning model (call it `model.rf`).

Some optional parameters should also be set as follows:

- Random forest classifier for the learning algorithm,
- The maximal tree depth to 20,
- The minimum number of samples in each node to 70,
- The number of clusters to 13 (equal to the number of classes),
- The maximum number of tree set to 50

Look at the application logs, in particular the confusion matrix, the Kappa coefficients, and scores per class. What do you think of those results ? Without using polygons dedicated to validation, the application will use a part of the generated samples for validation. What can be deduced regarding the displayed performances ?

Do the training again, but this time also use the validation dataset in `reference/testing/testing.shp` as validation layer (you therefore set two different shp files in the application). What can be observed ?

Do the training again, and deactivate the `temporary files cleaning` option. Look at the intermediate data that have been generated. What are they used for ?

4.2.3 Spot the date with the best performance

Do the training again, this time for each image date. What is the date with the best performances ? Does the Kappa coefficient change a lot ?

Keep the `model.rf` file corresponding to the best date.

4.2.4 Classifying and producing a colored classification map

Use the **ImageClassifier** application to produce the classification map corresponding to the best date (the one from 05.09.2016). Be careful to use the model file training with this date.

The output map is a TIFF image where each pixel value corresponds to the class. To visualize such images, the **ColorMapping** application allows to set a given color for each class.

Use the **custom** mode from the **ColorMapping** application with the look-up table in `support/color_map.txt` to produced the colored map.

Note : The image may not display correctly in QGIS, because of default no data value set in the file. No data can be deactivated in QGIS layer properties dialog.

4.2.5 Evaluate global performance

We are now going to use the **ComputeConfusionMatrix** application in order to compute global classification performances. With respect to the performance evaluation during the training step, this application allows to:

- Take into account all pixels in the reference date,
- Evaluate performances of a post-processed classification map (for instance with regularization).

The `ref.vector.field CODE` parameter is mandatory to indicate the field corresponding to class ids.

Compute the global performance of the classification. What do you observe with respect to the performance computed during training ? How to explain this phenomena ?

4.2.6 Classification regularization

We are now going to use the **ClassificationMapRegularization** application. It filters the classification map using a local majority voting scheme.

Parameters are:$

ip.radius 1 Radius of the voting zone

ip.suvbool 0 What to do in case of tie vote. 0 is to keep the initial value.

Filter the result of previous classification. Evaluate the performances of the filtered map. What do you observe ?

4.2.7 Classification with multiple dates

We are now going to use all dates for classifications. For this, you can use the `images/all.vrt` which contains all bands from each date concatenated (the image has therefore 50 bands).

Do the whole workshop, but this time with this 50 bands image. What do temporal series bring to the performance of classification ?

Compare both results in QGIS.

4.2.8 Going further

1. Can we obtain better performances with other classification algorithms ?
2. In QGIS, merge in reference data the grass and Woody Moorlands. Are the performances better ?
3. The `TrainImagesClassifier` also has an unsupervised algorithm (Shark KMeans). Compare results with supervised and unsupervised classification.

5 SAR processing on Sentinel-1 images

5.1 Introduction to SAR image processing

5.1.1 Description

Summary The following exercise is an introduction to processing and analysis of SAR images using the ORFEO ToolBox.

There are also a lot of other open source tools to process SAR images that can be used in combination with OTB (see PolSARPro, Sentinel-1 toolbox...)

Prerequisites

- Installed Monteverdi and Orfeo ToolBox software
- Downloaded data
- Understandig of Orfeo Toolbox applications (see relevant exercise)

Goals

- SAR image manipulation
- Radiometric calibration
- Geometric corrections
- Speckle filtering
- Basic SAR polarimetry

5.1.2 Steps

Data located in `Data/sar` folder.

Introduction to SAR imagery In this exercise we will use an extract from a Sentinel-1 SLC Strip Map (80 km Swath, 5 x 5 m spatial resolution): `s1_hh.tif` et `s1_hv.tif`. The image is located in Germany near the Constance lake (47.456276, 9.638616).

1. Open the image in Monteverdi. How many bands are present in each image?
2. What do these bands correspond to in the complex product?
3. Compute the image intensity of the complex products `s1_hh.tif` and `s1_hv.tif`. Compute also the image intensity in decibels (dB).

Radiometric calibration SAR images, like this Sentinel-1 product, provide pixel values without units (radar reflectivity or radar brightness). These are called digital numbers (DN).

Image metadata allows to transform DN into physical values (in case of SAR backscattering coefficient) which allows to compare images from different sensors or acquired at different dates in order to perform analysis.

SAR calibration consists in computing one of the following physical magnitudes:

- β_0: radar brightness coefficient, the reflectivity per unit area in slant range which is dimensionless.
- σ_0: radar backscatter (ratio of the outgoing radar signal that the target redirects back towards the antenna).
- γ_0: backscatter normalized by the incidence angle.

For Sentinel-1, Look Up Tables available in image metadata allow to convert DN into those values. ORFEO ToolBox will retrieve these values automatically and compute the backscattering coeficient.

1. Find the application which allows to perform this operation with OTB
2. Compute γ_0 for HH and HV images.
3. Convert in decibels (dB).

Geometry corrections We will use here the output of the radiometric calibration exercise ($gamma_0$) as input.

Use the **Orthorectification** application to perform geometric corrections using S1 metadata from the SLC product:

1. Without Digital Elevation Model (DEM)
2. With a DEM and a geoid (use the `SRTM` folder)
3. Compare the 2 images in Monteverdi.

Speckle filtering 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 backscattered waves from multiple distributed targets. It is locally strong and it increases the mean grey level of a local area.

Several different methods are used to eliminate speckle noise, based upon different mathematical models of the phenomenon. In the following part we will use the *Frost* filter. This algorithm has 2 parameters:

- *Radius*: window radius
- *deramp*: controls the exponential function used to weight effect of the distance between the central pixel and its neighborhood.

You can find more details about the Frost filter here.

1. What despeckle methods are available in OTB?
2. Use the *Frost* filter with different radiuses (3, 5 and 10) and comment about the effects on the output image.
3. Compare the histogram of the filtered image with the one of the intensity?
4. Use the *Frost* filter with `radius=5` and different *deramp* values (0.05, 0.2 and 0.8). Comment about the effects of the deramp parameter.

Polarimetry We will introduce basic notions of SAR polarimetry to show that HH and HV polarizations allows to access different types of information.

1. Compute the difference between HH and HV (use the intensity images as input).
2. Create a color composite of bands HH, HV and HH-HV. We can take here 2*HV as HV backscattered values are generally lower.
3. Convert the color composition into decibels (dB).
4. Display the image in Monteverdi and comment about the differences between HH and HV (layover areas, vegetation, soil, water...)
5. Using the image which combines HH, HV and HH-HV in decibels try to find in the extract:
   - the coordinates of 2 parallel power lines;
   - harder to find, locate in the image an area which looks like a corner reflector (reflects waves back directly towards the source);
   - what can you see in the lake at pixel coordinates (930,1170)?

Feature extraction To go further, we can explore available feature extractions methods for SAR available in OTB.

See for instance Touzi filter in **EdgeExtraction** application.

6 Using OTB Application in Python

6.1 OTB Applications on Python API

6.1.1 Description

Summary The following exercise is an introduction to the Python API for ORFEO ToolBox applications.

This exercise will show to chain OTB applications using an optical dataset for an hydrological goal : water surfaces analysis

Prerequisites

- Installed Monteverdi and Orfeo ToolBox software
- Installed Python (2.7.X or 3.5.X), with NumPy dependencies and the right environmental variables setup (help: source the otbenv.profile in Linux or launch otbenv.bat in Windows). **Test**: launch "import otbApplication" on the Python command line to check this point
- Downloaded dataset (`Data/app-python`)
- Understanding of Orfeo Toolbox applications (see relevant exercises)

Goals

- Create, configure and launch OTB Applications from Python scripts
- Use of in-memory connection between OTB Applications
- Process Sentinel2-Level 2A as input dataset (optical satellite images)
- Show a simple method for water detection based on NDVI
- Evaluate the resulting water map with a reference layer

6.1.2 Steps

Data are located in `Data/app-python` folder, with the following sub-folders:

- `images` contains a set of Sentinel 2 images (Level 2A) in Laguna de la Nina, Peru
- `ref` contains ancillary testing data (occurrence water masks) in raster format

This folder also contains the following Python scripts: `exercise1.py` ... `exercise5.py`

Introduction : Water monitoring in the Laguna de la Nina(Peru) event The region of interest for this exercise is Laguna de la Nina, Peru (-5.8101 lat, -80.7155 lon). In 2017 water surface extents have drastically changed due to heavy rains during "El nino" periods. The final objective is to analyze this change by means of satellite image processing.

In this exercise we will use three Sentinel-2 Level2A images (folder `app-python/images`) at the following dates:

2016-12-18
2017-04-07
2017-12-03

1. Open in Monteverdi the composite$*$.vrt file (RGB composition) of each of the dates. What do you observe in these images? How does the water extent change?

Note: The VRT compositions have been created with the tool `gdalbuildvrt` for this exercise. They are not included by default in Sentinel-2 products.

Sentinel 2 - Level 2A Format One of the goals of this exercise is to process this product as downloaded from the product provider (Theia server). Level 2A is an orthorectified product in ground reflectance, constructed from L1C products (orthorectified product in Top of Atmosphere).

Each Sentinel2-Level 2A product contains several **files**, which are classified as:

- SRE: (SRE for Surface REflectance) corrected for atmospheric effects, including adjacency effects
- FRE: (FRE for Flat REflectance) are also corrected for slope effect, which consists in suppressing the apparent reflectance variations. The corrected images look like if the land was flat.
- MTD: Metadata
- QKL: quicklook file (low resolution image to show an RGB overview)
- ATB: atmospheric and biophysical parameters with 2 bands :
  - 1st band: water vapor content (WVC) coded over 8 bits
  - 2st band: aerosol optical thickness (AOT) coded over 8 bits
- CLM: cloud mask computed by MAJA software, made of 1 band coded over 8 useful bits.


- SAT: saturation mask coded over 8 bits

In this exercise, water maps will be calculated from ground reflectance files SRE or FRE.
The SRE and FRE files consist of 13 files, one per spectral channel in GeoTiff format (.tif). Each band image may have a different resolution (10m or 20m).

Band name	S2 band id	Wavelength	Resolution	Used in this exercise
Blue	B2	490 nm	10 m	-
Green	B3	560 nm	10 m	-
Red	B4	665 nm	10 m	Yes
NIR - Narrow 1	B5	705 nm	20 m	-
NIR - Narrow 2	B6	740 nm	20 m	-
NIR - Narrow 3	B7	783 nm	20 m	-
NIR - Wide	B8	842 nm	10 m	-
NIR - Narrow 4	B8A	865 nm	20 m	Yes
SWIR 1	B11	1610 nm	20 m	-
SWIR 2	B12	2190 nm	20 m	-

For this exercise, only some bands will be used to obtain water extents maps: Red (B4) and NIR - Narrow4 (B8A). Also, the Cloud Mask will be used.
Note: To reduce the dataset size, we have deleted all the bands not used and replaced them with an empty file with the same name.
Let's play:

1. Since we are interested in ground reflectance images to calculate water surfaces, what band kind of file would you use : SRE or FRE?

2. Look at the disk size of B3 and B11 files of one the datasets in `app-python/images/SENTINEL2A_*/` Do all files have the same disk size? Why?

3. On the command line, launch the `gdalinfo` command on different band files to check the pixel size, the number of pixels and the minimum and maximum values. Do we have common minimum values between different bands? Why?

   Note: Make sure that OTB binary files ($otb-path/bin) is included in your PATH environment variable.

4. Look at /MASKS sub-folder : there is a CLM file that contains a cloud mask. Do you think that this information might be interesting to make better water detections? How?

5. Open in Monteverdi the B8A and B4 and check the values in a water surface. Which band has higher reflectance values on water surfaces? (Use the 20170407 date to have wider water surfaces)

Simple OTB application in Python : exercise1.py Take a look to the script: `app-python/exercise1.py`. The aim of this script is to launch the Superimpose application from OTB to resample the B8A band (20m pixel size) to a new resolution.
At the beginning, there is an otbApplication import. In the otbApplication module, two main classes can be manipulated:

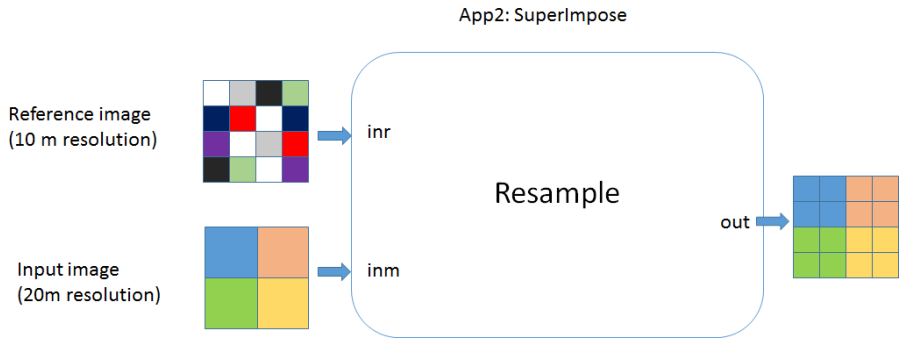
- *Registry*, which provides access to the list of available applications, and can create applications.
- *Application*, the base class for all applications. This allows to interact with an application instance created by the Registry
- In order to show the available applications, launch `exercise1.py` with the command :

```
$ python exercise1.py
```


At the output you will see the list of available applications. Which line in the script allowed to show the list of applications? This method is present in the Registry or in an Application module?

On the second part of the script, we want to launch the Superimpose application to do the resampling of the B8A image (20m pixel size) using the image B4 (10m pixel size) as a reference.

The script `exercise1.py` launches the Superimpose OTB application as presented in the following scheme:



Note: Superimpose may be configured to used different interpolations (linear, bi-cubic or nearest neighbor)

The necessary inputs and outputs of the Superimpose application are described in the following table:

Parameter Key	Parameter Name	Parameter Type
inr	Reference Input	input image
inm	Image to re-project	input image
out	Output image	output image

1. Open `exercise1.py` and complete the "FILL THE GAP 1". You need to complete the path of `app-python/images` of your system.

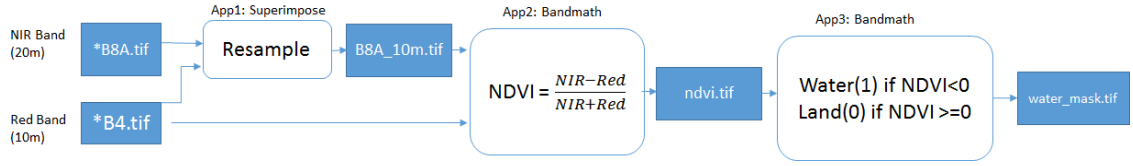
2. Open `exercise1.py` and complete the "FILL THE GAP 2". You need to initialize the Superimpose OTB application.

3. Open `exercise1.py` and complete the "FILL THE GAP 3" to set `inr` ,~inm~ and `out` parameters of the Superimpose application.

4. Launch `exercise1.py` the script with the command python exercise1.py. How does the output file `B8A_10m.tif` look like?

Chain OTB applications : exercise2.py In this part, the aim is to calculate an NDVI image and obtain a water mask by means of thresholding the NDVI value. We will launch different OTB applications in the same script to obtain the desired result.

The script `exercise2.py` chains OTB applications as presented in the following scheme:



Use the Superimpose and Bandmath applications to calculate the NDVI and the water map image using Red band (B4) and NIR band (B8A) from the S2 product:

1. Open `exercise2.py` and complete the "FILL THE GAP 1". You need to complete the path of `app-python/images` of your system.
2. Open `exercise2.py` and complete the "FILL THE GAP" 2 and 3. You need to :
   - configure the application2 "BandMath" parameters: `il,out,exp`
   - configure the application3 "BandMath" parameters: `il,out,exp`

   Note: Take a look to lines 23-30 to understand the filepath of each band image. Check also the online help of the applications if necessary.
3. Launch `exercise2.py` script with the command:

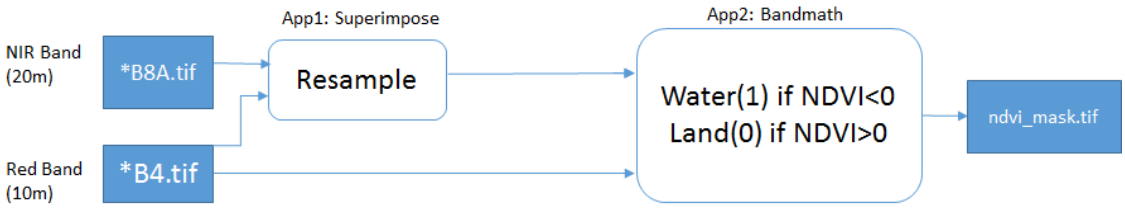
   ```
   $ python exercise2.py
   ```

   What are the resulting files? Check the results in Monteverdi.

Chain OTB applications in-memory: exercise3.py This exercise is equivalent to exercise2.py, but avoiding to write on intermediate files. The goal is to process the intermediary results using only RAM memory.

As a second improvement, the NDVI calculation is left to the last step: NDVI and water mask are calculated at the same time. In OTB terms, we perform just one BandMath calculation (instead of two).

The script `exercise3.py` chains OTB applications as presented in the following scheme:



In-memory connection: the output of app1 might be declared as input of app2 using an expression as:

- app2.SetParameterInputImage("in",app1.GetParameterOutputImage("out")) if the input of application2 is an Image(like in the Superimpose application)
- app2.AddImageToParameterInputImageList("il",app1.GetParameterOutputImage("out")) if the input of application2 is an ImageList(like the BandMath application)

Let's optimize our water mask calculator:

1. Open `exercise3.py` and complete the "FILL THE GAP 1". You need to complete the path of `app-python/data` of your system.
2. Open `exercise3.py` and complete the "FILL THE GAP 2" to declare the output of application1 as input of application2.
3. Open `exercise3.py` and complete the "FILL THE GAP 3" to set the BandMath expression that sets value 1 if ndvi value<0 and 0 if ndvi value>1
4. Launch `exercise3.py` with the command:

   ```
   python exercise3.py
   ```

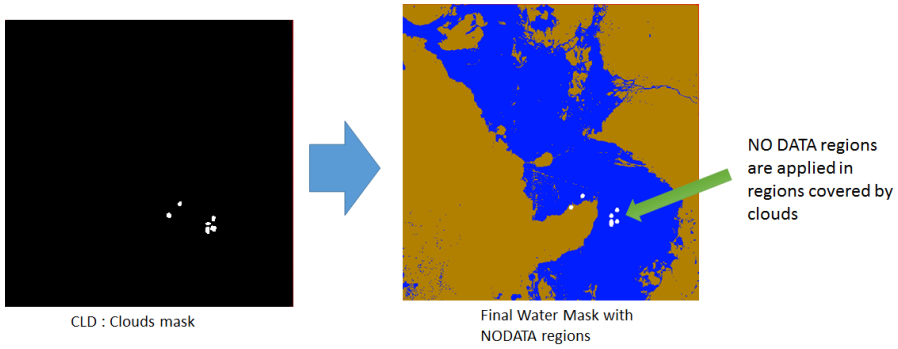
   As you see in the code, the ApplicationX.ExecuteAndWriteOutput() has been changed to ApplicationX.Execute() in `exercise3.py`. How does it affect the execution sequence?

5. In Application1, the output parameter has been declared with a filename. Has it been written as a file after the execution? Why?

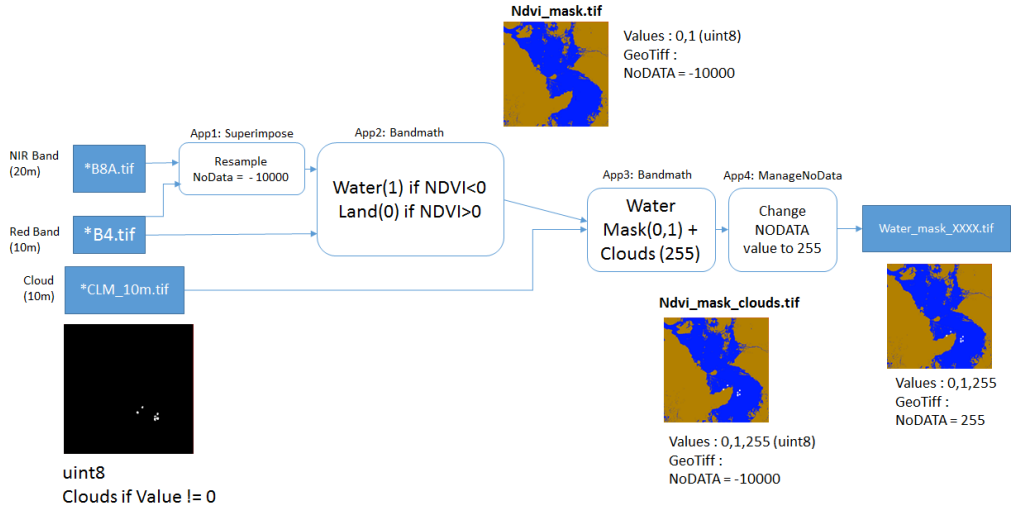
6. At the generation of the NDVI mask(with two possible values: water(1) and land(0)), there is a line like : appX.SetParameterOutputImagePixelType("out", otbApplication.ImagePixelType_ uint8) What is the purpose of this line? What would have happened without it?

Water detection chain with NoData management: exercise4.py There are some parts of the images that are covered by clouds. In this exercise, we will use the CLD band in the S2 product to set NODATA regions.

If a CLD pixel value is different of zero, that means that a cloud has been detected in the pixel. The aim of this exercise is to use an special value (255) in the final mask when clouds are present.



The script `exercise4.py` chains OTB applications as presented in the following scheme:



At the end of the chain, an OTB application "ManageNoData" is used to set the NODATA value as 255 in the GeoTiff metadata.

Let's do it:

1. Open `exercise4.py` and complete the "FILL THE GAP 1". You need to complete the path of `app-python/data` of your system.

2. Open `exercise4.py` and complete the "FILL THE GAP 2" to set the BandMath expression to set the 255 value where the clouds image is different to zero, and otherwise keep the NDVI mask image.

3. Launch `exercise4.py` with the different dates as arguments:

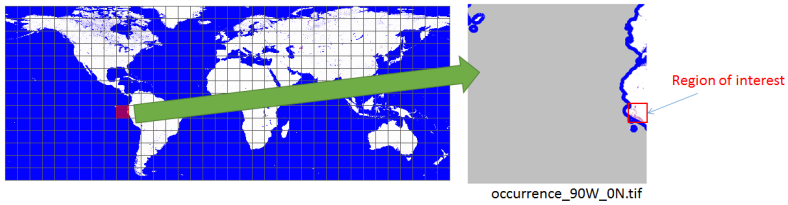
   ```
   python exercise4.py SENTINEL2A_20161218-153729-222_L2A_T17MNP_D_V1-4
   python exercise4.py SENTINEL2A_20170407-154054-255_L2A_T17MNP_D_V1-4
   python exercise4.py SENTINEL2A_20171203-154308-349_L2A_T17MNP_D_V1-4
   ```

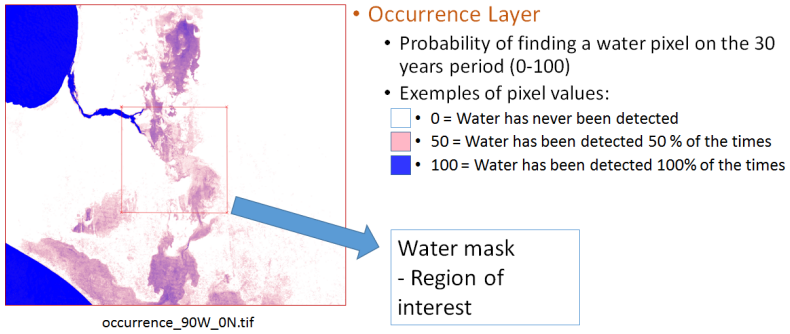
   and you will obtain three different masks. Open them with monteverdi to check the water extent variations.

4. Look at the 20161218 final water mask. What are the lines detected as water?

Comparison with a reference : exercise5.py The water masks obtained after February 2017 correspond to an special flood event in the Laguna de la Nina. How often do we observe floods in this region? Let's try to answer it.

The Global Surface Water(GSW) map, a water extent map based on optical images (Landsat satellite) over the last 30 years, can be helpful to understand how frequently water is detected in a given area. This product contains an occurrence layer that provides the ratio between "water found" occurrences and the number of valid observations in the last 30 years for each pixel. Hence, a pixel with occurrence value equal to 10 means that water has been detected in 10% of valid observations.





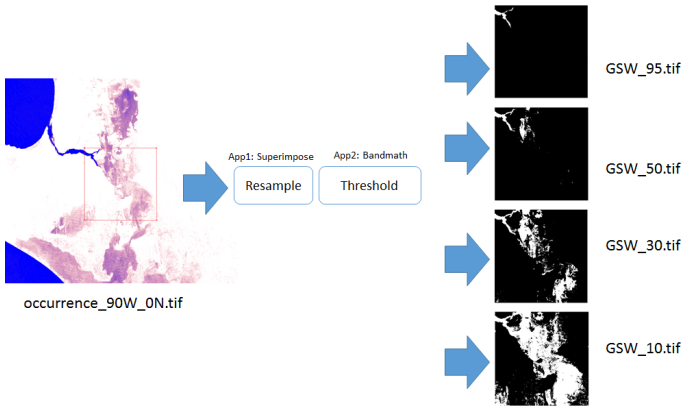
The Global Surface Water data are available for download in tiles 10°x10° on Global Surface Water AppSpot website.

You will find the required dataset already downloaded in the folder: `app-python/ref`

For this exercise, we are going to compare the 3 water masks (20161218, 20170407 and 20171203) produced in exercise 4 with the GSW product.

The method to follow for this exercise is:

- Crop and resample the GSW occurrence layer(30m resolution) to match with the water mask grid (10m resolution).
- Apply a threshold on the GSW resampled product with different probabilities: 10%, 20%, 30%, 50%, 75%, 95% to obtain different reference images
- Compare the water extent masks of exercise 4 with each of the reference images issued from GSW. This comparison will help us to understand how often do we observe a water extent map along time.



Steps:

1. Open the image `ref/occurrence_90W_0N.tif` in QGIS or Monteverdi.

   WARNING: Select "ignore" in the pop-up message for the overviews generation.

   What values do you observe around the coordinates lon: -80.6767, lat: -5.91. Do you observe any zone with 90-100 occurrence? What does it means: permanent or rare waters?

2. Our zone of study is just a portion of the `ref/occurrence_90W_0N.tif`. The `exercise5.py` script will launch Superimpose and Bandmath OTB applications to obtain the GSW layer cropped, resampled and thresholded at given value. To obtain the reference mask launch:

   ```
   python exercise5.py 10
   python exercise5.py 20
   python exercise5.py 30
   python exercise5.py 50
   python exercise5.py 75
   python exercise5.py 90
   ```

3. Now it's time to compare each of the water masks of exercise 4 (on 3 dates: 20161218, 20170407 and 20171203) with each of the reference images. Check the following list of applications in the OTB Applications reference documentation, and look the Learning section. Which application do you think that might be helpful to compare two raster maps? Use this application to calculate the reference mask that yields the best likelihood (Kappa index) for each of the 3 water masks.You may launch the otb application in the commandline: otbcli-XXXXX -in XXXX -out XXXX -param1 XXXX

4. What would you conclude about the results?