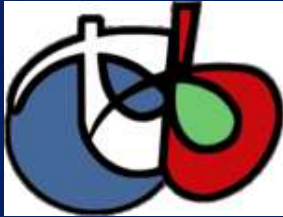


Your first steps with Orfeo Toolbox :



How to write an OTB processing chain within a Jupyter Notebook ?

David YOUSSEFI – david.youssefi@cnes.fr
Yannick TANGUY – yannick.tanguy@cnes.fr

CNES

OTB Guided Tour overview

Overview of Orfeo ToolBox

Get familiar with Monteverdi & OTB applications

Processing chain with OTB in Python, within a Jupyter Notebook



About the Virtual Machine

Linux Ubuntu « budgie » distribution

Orfeo ToolBox 6.6.1 (2018)

Python environment installed with Conda

→ [More information in README.md](#)

Login / Passwd : otb

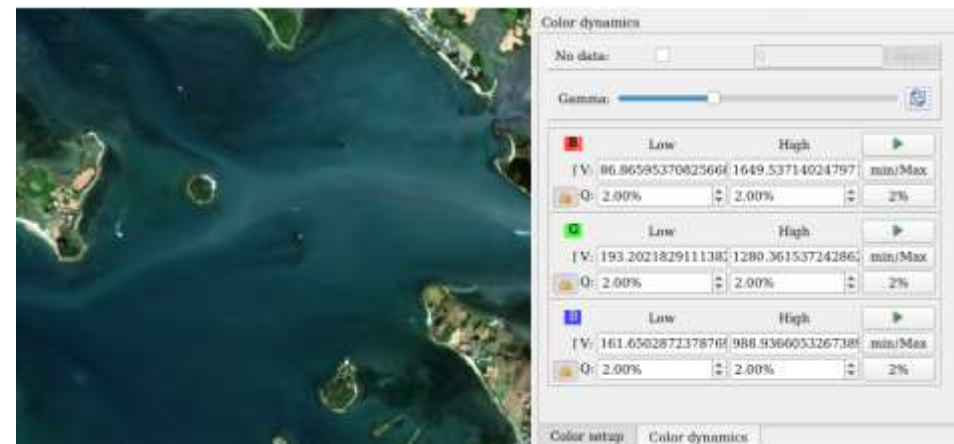
About the data

SENTINEL2 images :

- 4 bands have been concatenated : Red (B4), Green (B3), Blue (B2) and NIR (B8)
- 10 meter spatial resolution
- Level 2A products : calibrated, orthorectified
- ... but there are still clouds on several images
- 4 images, with various weather conditions, and also different tides levels !

Get familiar with Monteverdi and OTB applications

- Know how to display an image and set up rendering in Monteverdi,
 - Know how to display an image stack in Monteverdi,
 - Launch an OTB application
-
- Open a terminal and launch Monteverdi
 - `$> monteverdi &`
 - Sentinel2 extracts are in `~/otb-guided-tour/data`
 - Open the image of the 2018-06-21
 - Open the image of the 2018-07-06



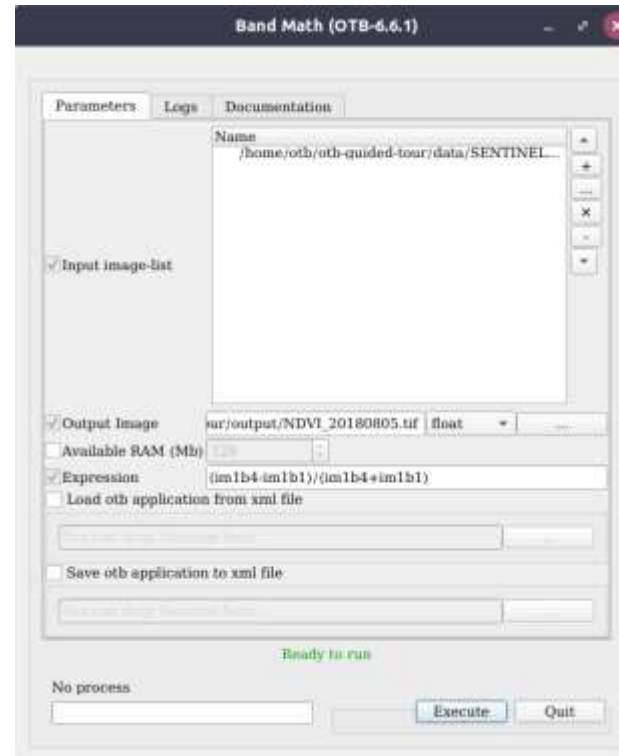
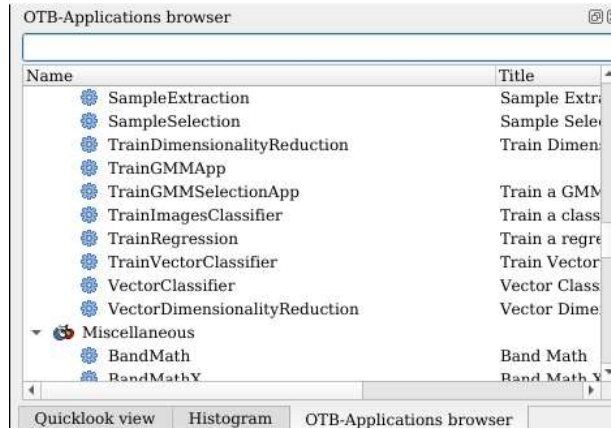
Get familiar with Monteverdi and OTB applications

- Try the different “Layer effects” on the bottom layer
- Scroll between layers (Ctrl + mouse-wheel)
- Switch between two layers (right click)
- Apply the display settings
- Change band selection to make a false-color composite
 - Red : band 4 (Near InfraRed)
 - Green : band 3 (Red)
 - Blue : band 2 (Green)



Get familiar with Monteverdi and OTB applications

- Display list of applications
- Launch an application
- Compute NDVI



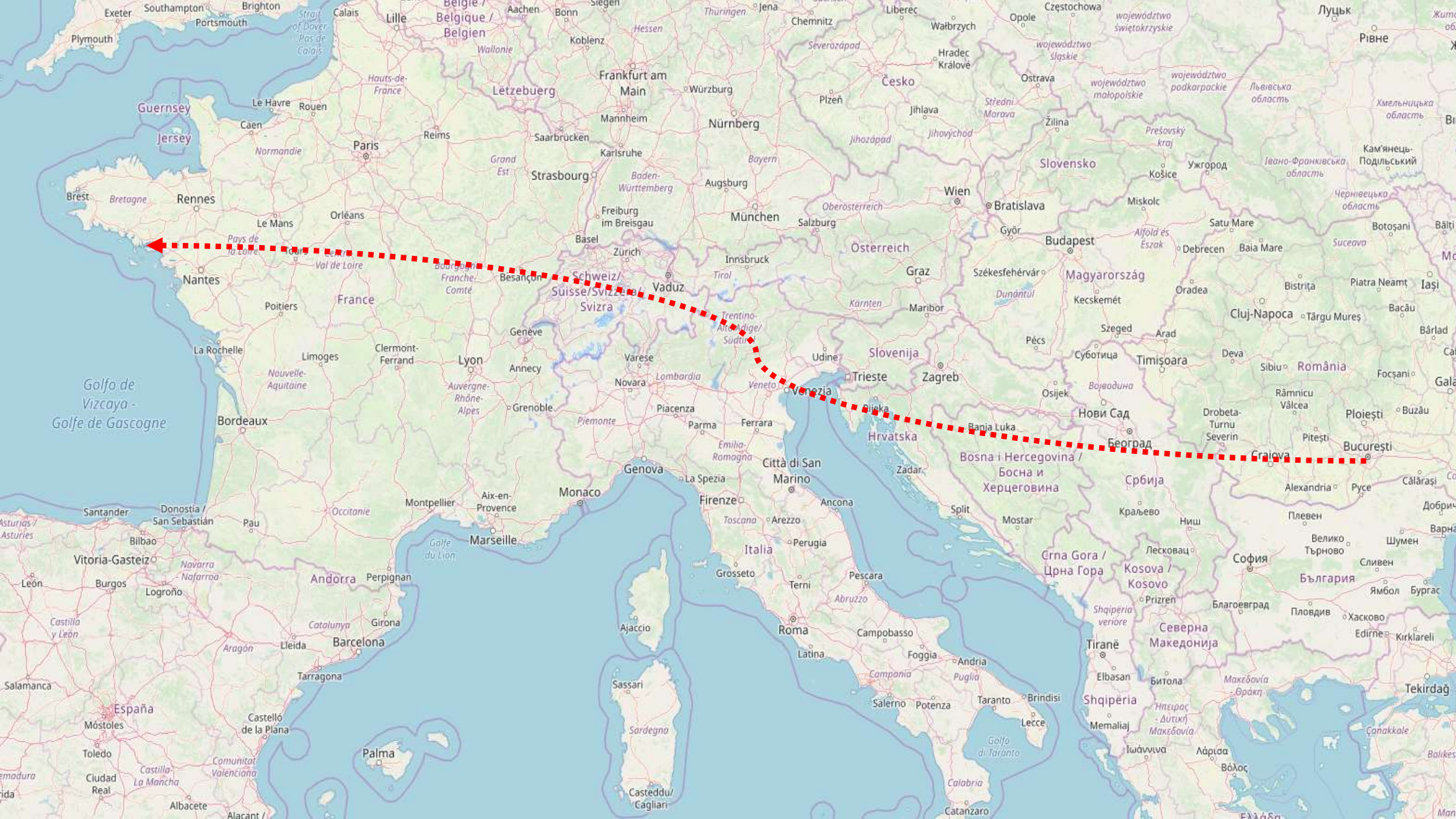
OTB Guided Tour overview

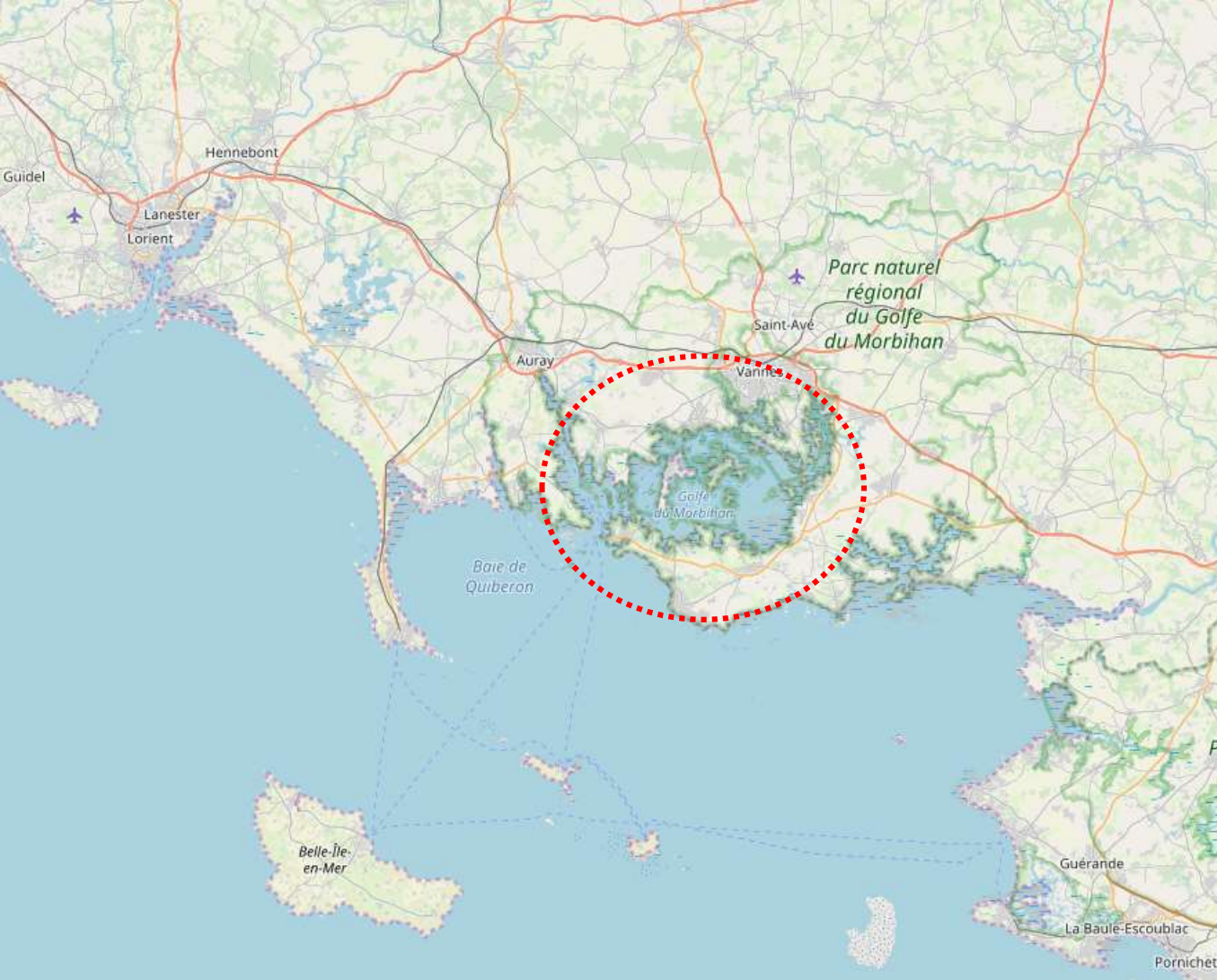
Overview of Orfeo ToolBox

Get familiar with Monteverdi & OTB applications

Processing chain with OTB in Python, within a Jupyter Notebook







“Legend says that there are as many islands in the Gulf as there are days of the year.

However, this is untrue and the gulf has about 40, depending on the tide.”
(Wikipedia)

Let's count the Islands with Orfeo
ToolBox and a few Sentinel 2 images !

Write a processing chain with OTB in Python

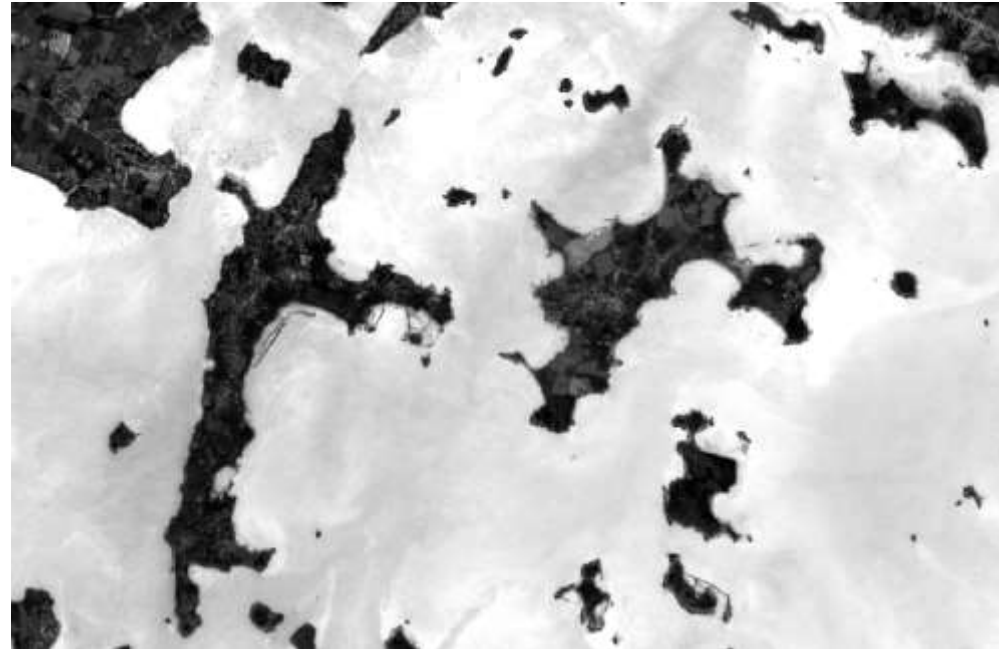
Goals of this session :

- Focus on OTB applications mechanisms
- Image analysis
- From remote-sensing to GIS tools with OTB : GDAL, Rasterio, shapely, etc.
- Python scripts

Write a processing chain with OTB in Python

- **Open a terminal :**
 - `$> cd otb-guided-tour`
 - `$> jupyter notebook`
- **Several Jupyter Notebook have been prepared for this training session :**
 - `Otb_workshop` : overview of the dataset / display on a map
 - Step 1 : How to launch an OTB application and compute radiometric indices ?
 - Step 2 : How to launch the same processing on several images ? How to make a synthesis of NDWI2 ?
 - Step 3 : Finalize our watermask
 - Step 4 : Polygonize the watermask and filter the islands !

1st step : compute radiometric indice



1st step : compute radiometric indice

Normalized Difference Water Index « 2 »

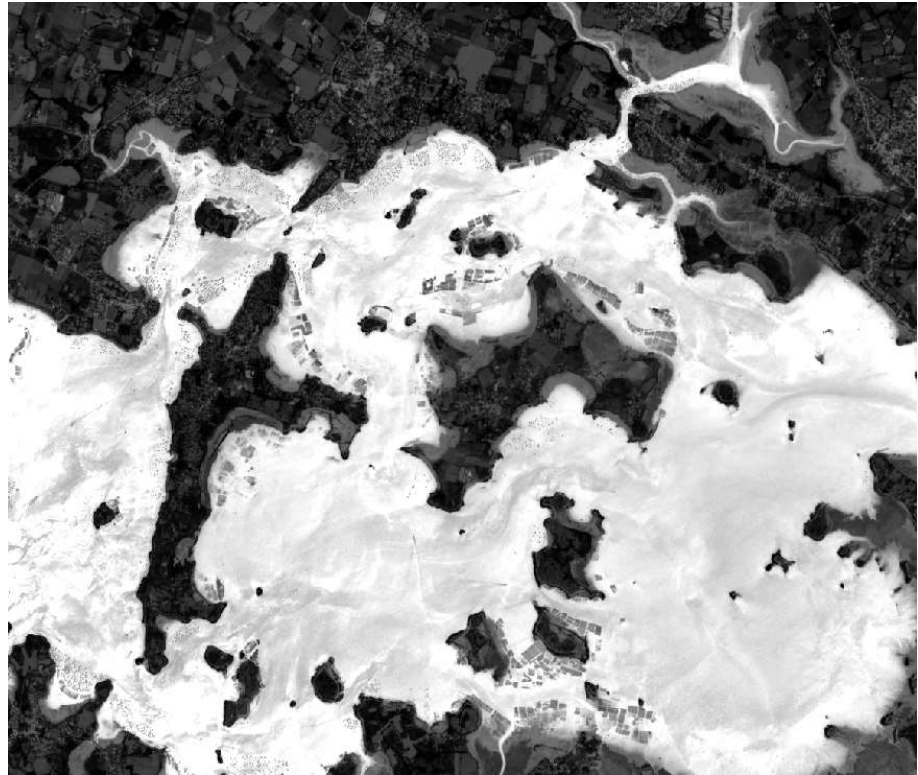
$$(\text{green} - \text{nir}) / (\text{green} + \text{nir})$$

Very convenient to identify water bodies

Lands < 0

Lakes, sea > 0,3

Note : NDWI (based on NIR and MIR) is more adapted to monitor water in vegetation



OTB Python API in a nutshell 😊 (1/2)

Use OTB within Python :

```
import otbApplication as otb  
  
app = otb.Registry.CreateApplication (« BandMath»)
```

Set parameters :

```
app.SetParameterString("in", argv[1])  
  
app.SetParameterValue(<key>, <value>)  
  
app.SetParameterStringList(<key>, [<value1>, <value2>, ..])
```

Launch the application

```
App.ExecuteAndWriteOutput() -> wire the pipeline and write output
```



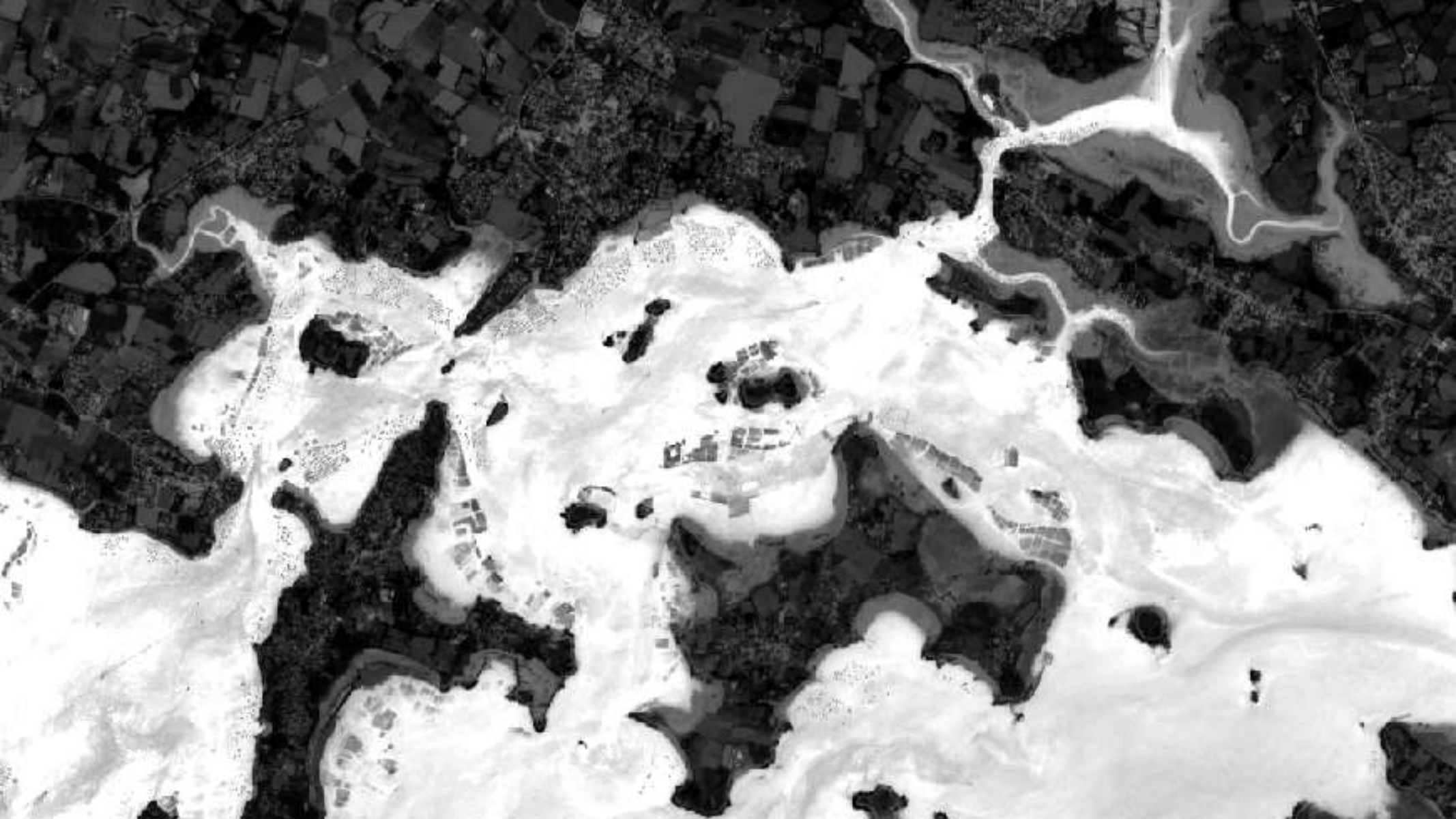
2nd step : compute a mask from the various NDWI images

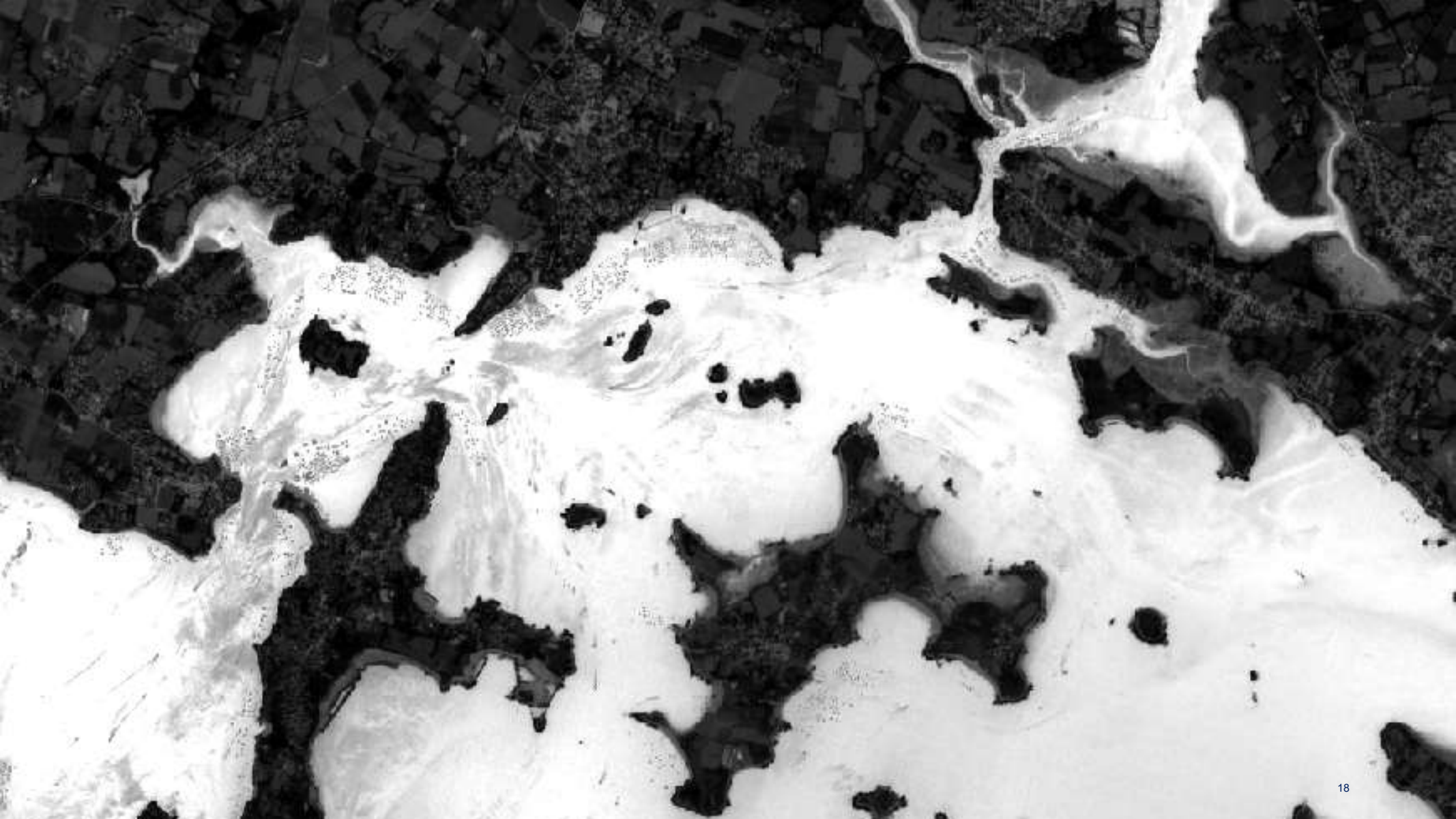
Have a look at the images ?

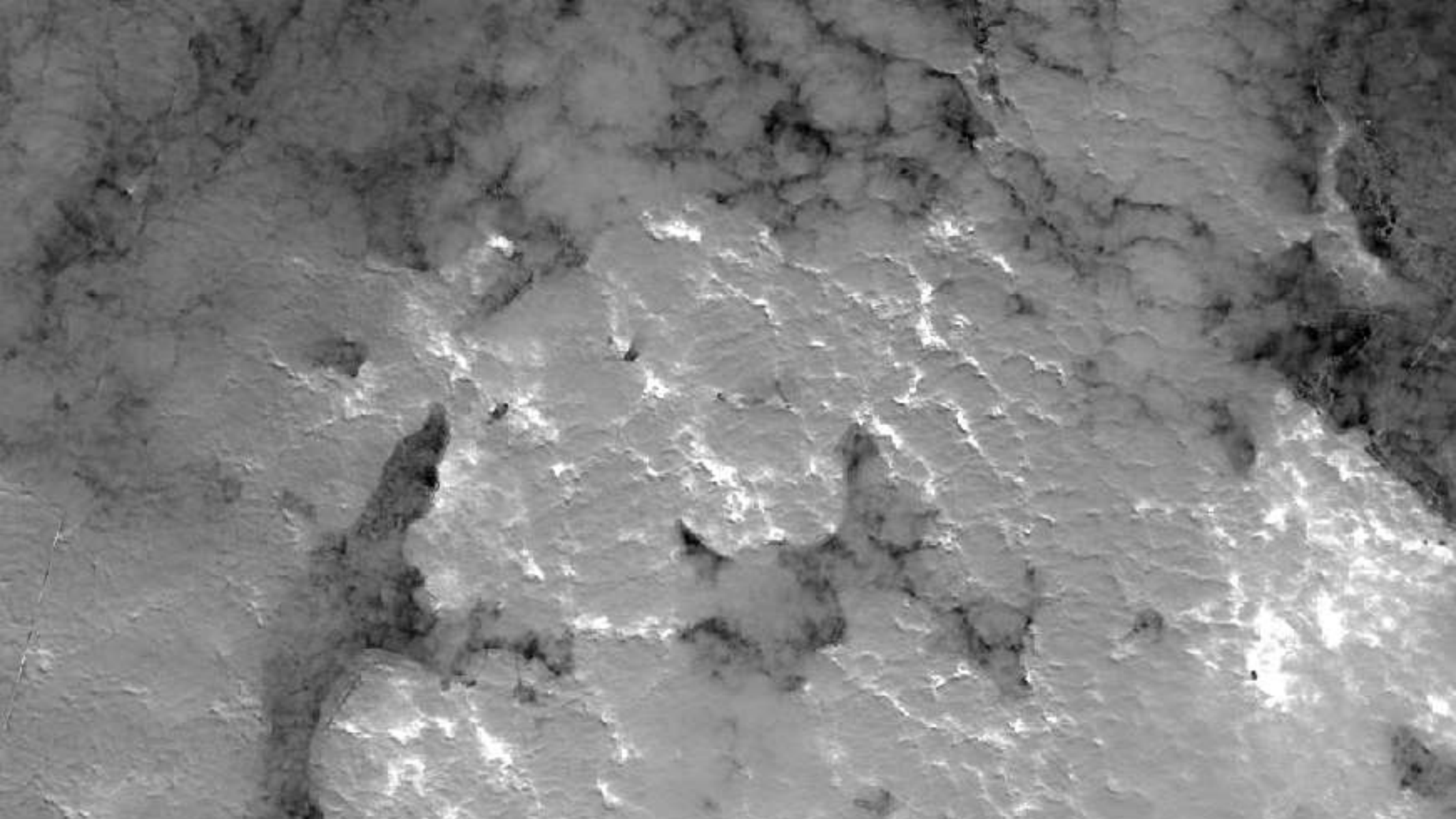
What do you observe ?

Are there some artifacts to deal with ?

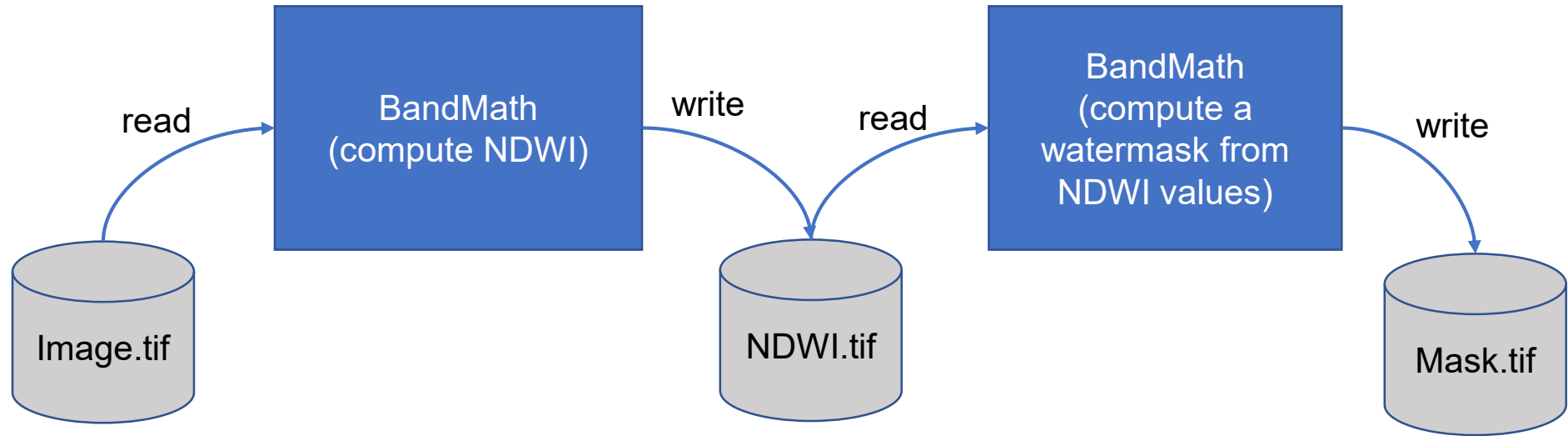
What kind of formula could we use to have a mask that maximize water extent (high tides) ?



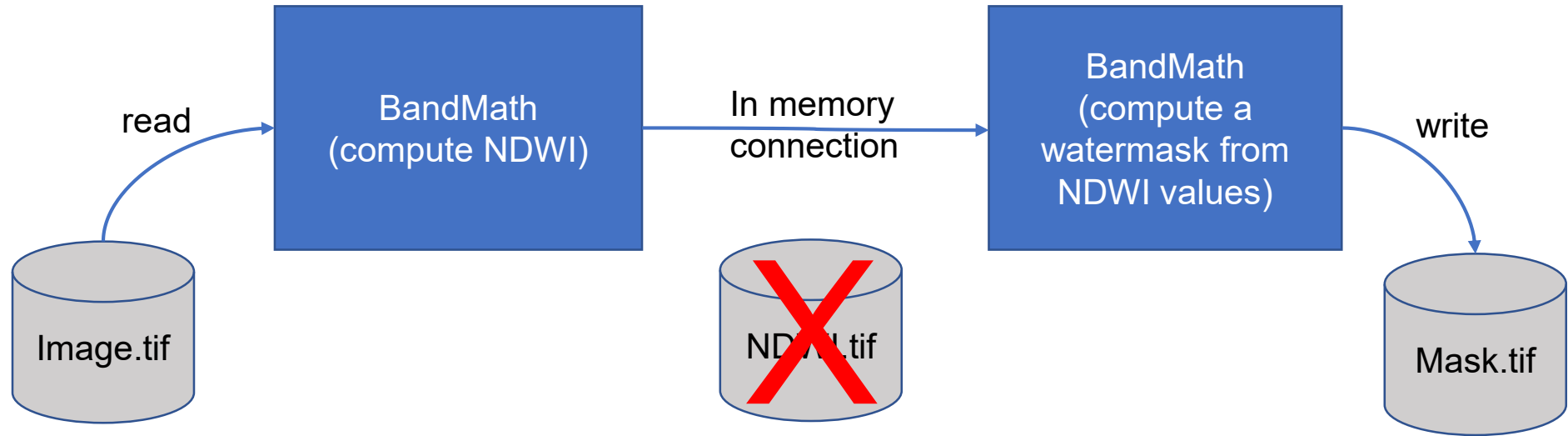




OTB Python API : Let's play with the pipeline !



OTB Python API : Let's play with the pipeline !



OTB Python API : Let's play with the pipeline !

[...]

```
app1.SetParameterString("out", output_image)
```

`app1.Execute()` → wire the pipeline but does not execute yet !!

[...]

```
app2.SetParameterInputImage("in", app1.GetParameterOutputImage("out"))
```

→ set the output of app1 as the input of app2

Or `app2.AddImageToParameterInputImageList("in", app1.GetParameterOutputImage("out"))`

→ same for an application that takes a list of input

`app2.ExecuteAndWriteOutput()` → resolve the pipeline, launch all processings and write output

OTB Python API : Let's play with the pipeline !

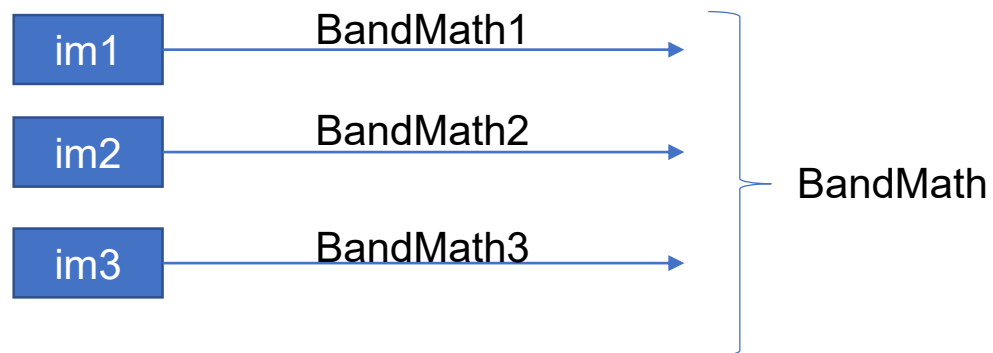


Image List : `il = [BandMath1.GetParameterOutputImage(« out »),
BandMath2.GetParameterOutputImage(« out »),
BandMath3.GetParameterOutputImage(« out »)]`

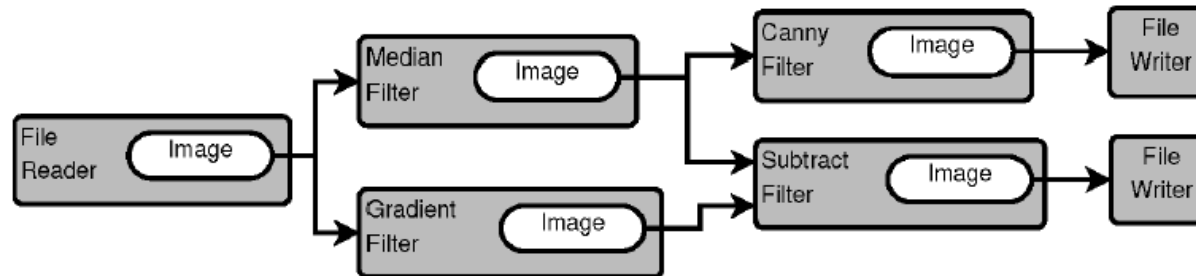
Expression : `exp = « f(im1b1, im2b1, im3b1) »`

Compute Radiometric Indices (NDWI2)...
.... But does not write file !

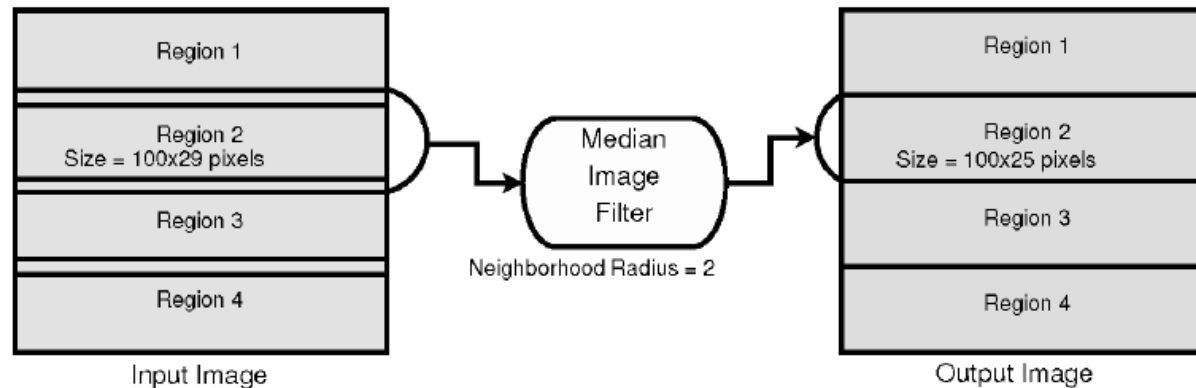
Makes a synthesis of the time serie of NDWI2...
... and writes the image !

A few words on the OTB pipeline (inherited from ITK)

Le modèle de *Pipeline*



Streaming



3rd step : finalize the watermark

From our NDWI2 synthesis, we want to compute a watermark.

Different solutions are available..

Easiest one : threshold on NDWI value

Alternative : use the segmentation framework in OTB. The Large Scale Mean Shift application is able to make very fine segmentation of high resolution satellite images.

We still have some artifacts :

- **we could use a Binary Morphological operator to erode a little bit the mask**
- **Or we can filter small features (ie : small submarine / dark areas)**









4th step : polygonize the raster using Gdal and filter islands

The last script will polygonize the raster file :

The different polygons are whether :

- Islands !
- The ocean or sea
- Main land
- Lakes / inner water bodies
- Small submarine areas, only visible on lowest tides

The script keeps shapes that intersects a shape of Morbihan gulf...

... and eliminates thoses whose area is bigger than « île aux moines » area, and smaller than 1 Ha (100m x 100m)



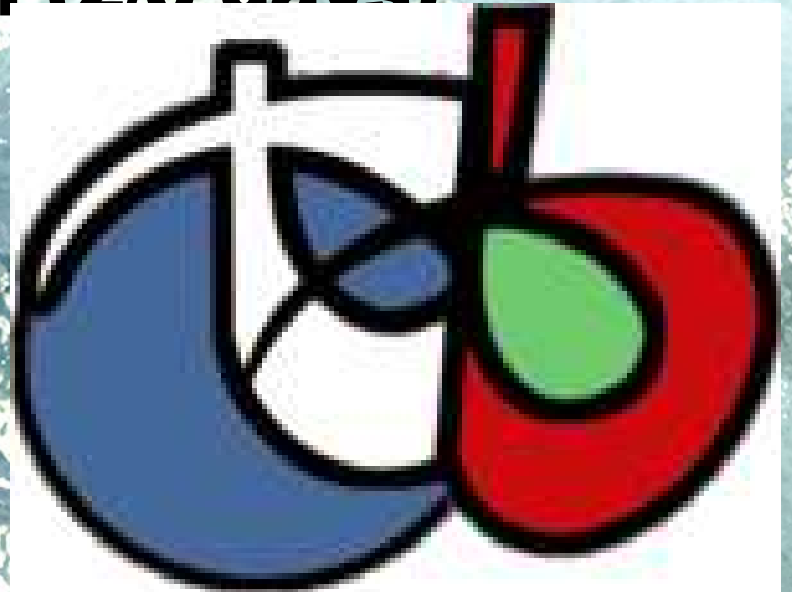


Any Question ?

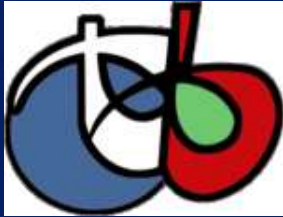


To go further with OTB :

- **Documentation :** [CookBook](#)
- [Full OTB training material](#)
- **OTB training session (2/3 days)**
- [User forum](#)
- [Gitlab](#)
- **OTB Days**



Your first steps with Orfeo Toolbox :



Focus on the classification framework

David YOUSSEFI – david.youssefi@cnes.fr
Yannick TANGUY – yannick.tanguy@cnes.fr

Objectives of this demo

Machine learning basics :

- Understand how to make a land cover map from a stack of several satellite images
- How to interpret results, to improve the classification ?

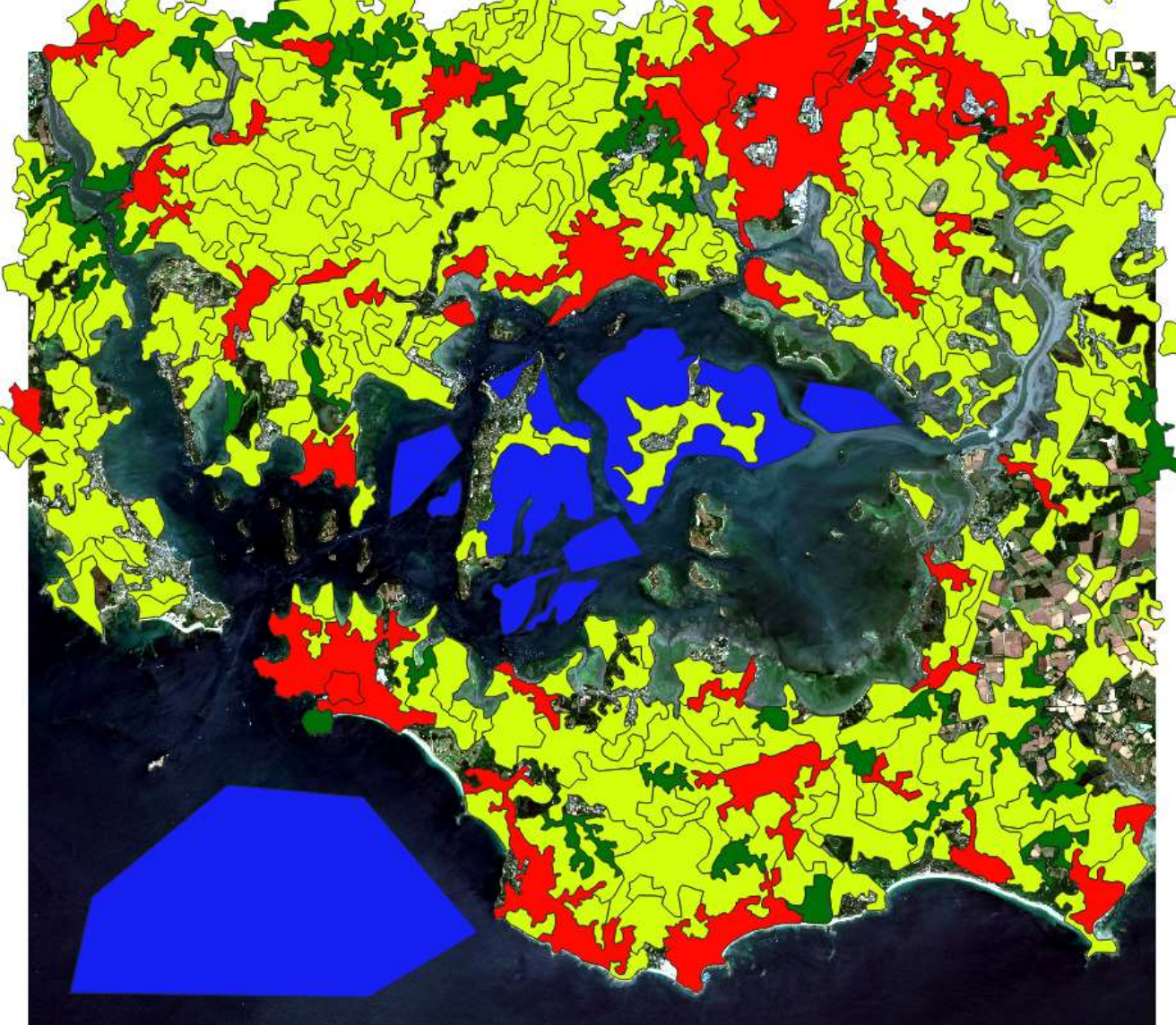
For this (very short !) demo :

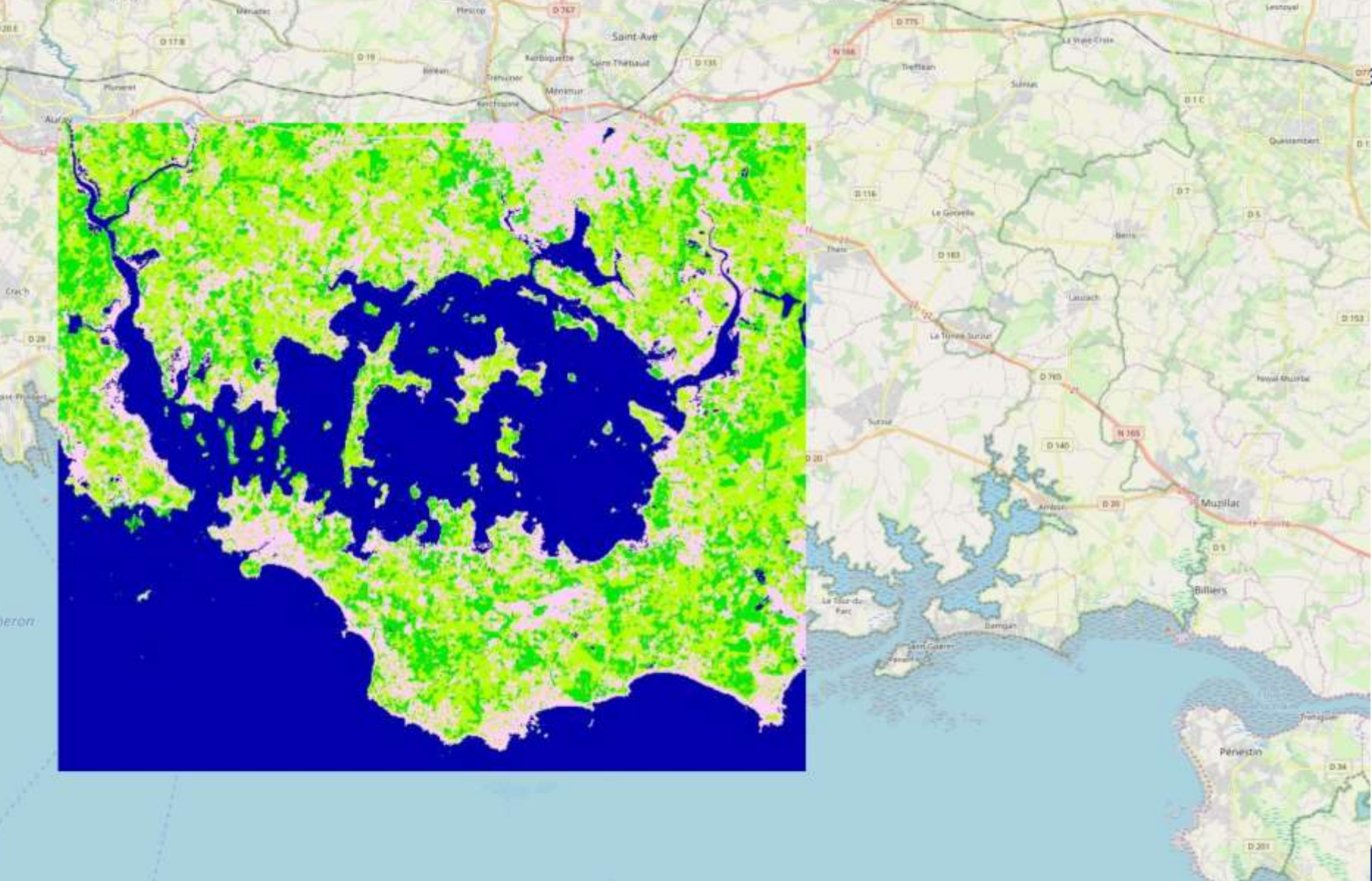
- Very small time series (3 images !) : usually, we use a time serie that covers the whole year (for example, on image every 10 days -> 36 images)
- Very simple classification : 4 classes (urban area, water/sea, forest, cultures)
- Only 4 bands (R, G, B + Near Infrared) instead of the 10 bands available in S2

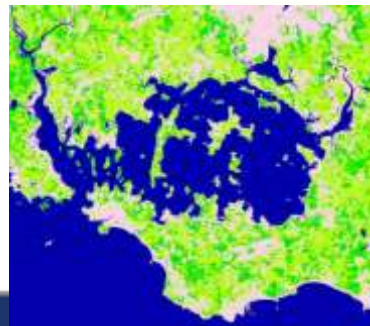
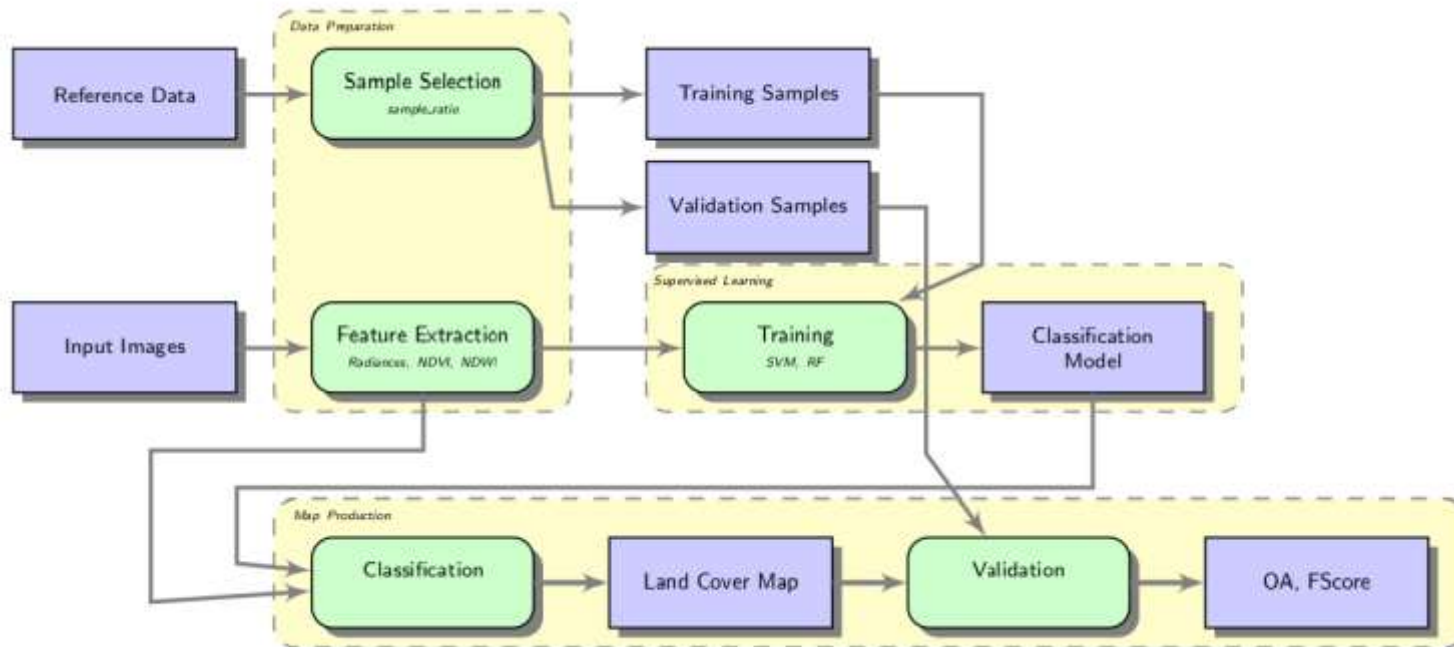
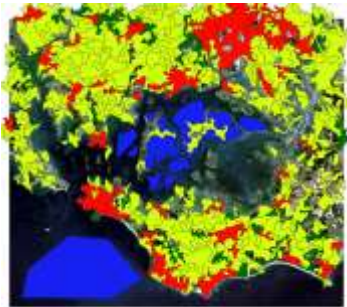












Let's produce a landcover map of Morbihan area

- **Open a terminal :**
 - `$> cd otb-guided-tour`
 - `$> jupyter notebook`
- **Open the « `land_cover_classification` » notebook and follow the instructions !**

Main steps

Step	Purpose of that step	OTB application
Make an image stack from our S2 images	The image stack will contain 12 bands : 3 dates * (R, G, B, Near IR)	ConcatenateImages
Compute statistics on our ground truth polygons	Evaluate how many pixels are covered by each class	PolygonClassStatistics
Select samples	Choose a nb of samples for each class : creates a shapefile with all the samples.	SampleSelection
Extract features	Each sample (1 pixel) will be filled up with the 12 bands values ; the samples shapefile is updated	SampleExtraction
Train a model	A random forest algorithm will learn from our samples	TrainVectorClassifier
Classification step	Each pixel of our image stack will be classified : it returns an integer image. Each pixel has a code (1, 2, 3, 4) depending on its class.	ImageClassifier, [+ ColorMapping]


More information on the CookBook

Classification — OTB Cookbook

[←](#)
[→](#)
[↺](#)
[⌂](#)

<https://www.orfeo-toolbox.org/CookBook/recipes/pbclassf.html#pixel-based-classification>

OTB Cookbook



2.8.1

Welcome to Orfeo Toolbox

Installation

A brief tour of OTB Applications

QGIS interface

Monteverdi

Advanced Use

⊖ Recipes

From raw image to calibrated product

SAR processing

Residual registration

Image processing and information extraction

BandMathImageFilterX (based on muParserX)

Enhance: local contrast

⊖ Classification

⊖ Feature classification and training

⊖ Pixel based classification

Sample statistics estimation

Sample selection

Sample extraction

Working with several images

Sample statistics estimation

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: