



# Ciências ULisboa

Faculdade  
de Ciências  
da Universidade  
de Lisboa

**DEGGE**

**MULTISPECTRAL REMOTE SENSING**

MASTER IN GEOSPATIAL ENGINEERING

2023/2024

**FREE AND OPEN-SOURCE SOFTWARE TO BE INSTALLED**

<p><b>Orfeo ToolBox (OTB) Version 9.0.0</b></p>	<p>Orfeo ToolBox (OTB) version 9.0.0, released on 2024-02-28, is an open-source project for state-of-the-art remote sensing (<a href="https://www.orfeo-toolbox.org/">https://www.orfeo-toolbox.org/</a>). 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.</p> <p>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 the applications in the toolbox. From resource-limited laptops to high-performance MPI clusters, OTB is available on Linux, macOS, and Windows. It is community-driven, extensible, and heavily documented. Orfeo ToolBox is not a black box (<a href="https://www.orfeo-toolbox.org/CookBook/">https://www.orfeo-toolbox.org/CookBook/</a>).</p>
<p><b>QGIS Version 3.36.0</b></p>	<p>QGIS is a professional GIS application that is built on top of and proud to be itself Free and Open Source Software (FOSS). The current version is QGIS 3.36.0 'Maidenhead' and was released on 2024-02-23 (<a href="https://qgis.org/en/site/forusers/download.html">https://qgis.org/en/site/forusers/download.html</a>). QGIS is available on Windows, macOS, Linux, and Android.</p> <p>QGIS has a lot of documentation. All documentation is in English, but documents such as the user guide are also available in other languages (<a href="https://qgis.org/en/docs/index.html">https://qgis.org/en/docs/index.html</a>).</p>
<p><b>SNAP Version 9.0.0</b></p>	<p>SeNtinel's Application Platform (SNAP) is an open-source common architecture for ESA Toolboxes ideal for the exploitation of Earth Observation data (<a href="https://step.esa.int/main/download/snap-download/">https://step.esa.int/main/download/snap-download/</a>). SNAP reunites all Sentinel Toolboxes to offer the most complex platform for this mission. The current version is 9.0.0 released on 2022-06-29.</p>

## PRACTICAL LESSON 1

### EXERCISE 1.1

Free and open access for both Landsat and Sentinel data products is available to all users through the following links, respectively:

#### **USGS WEBSITE**

The USGS Earth Explorer data portal is your one stop shop for obtaining geo-spatial datasets from our extensive collections. Users can navigate via interactive map or text search to obtain Landsat satellite imagery, Radar data, UAS data, digital line graphs, digital elevation model data, aerial photos, Sentinel satellite data, some commercial satellite imagery including IKONOS and OrbView3, land cover data, digital map data from the National Map, and many other datasets. Users can search by exact location via the interactive map or input specific coordinates to view what data types are available.

<https://www.usgs.gov/land-resources/nli/landsat/landsat-data-access>

<https://earthexplorer.usgs.gov/>

#### **CONVENTIONAL DATA ACCESS HUBS**

One of the access points to a wide range of Earth observation data from the Copernicus Sentinel missions managed by ESA is the **Copernicus Data Space Ecosystem** that provides tools for easy discovery, visualization, and download which will be continuously upgraded.

<https://www.copernicus.eu/en/access-data/conventional-data-access-hubs>

<https://dataspace.copernicus.eu/>

#### **DOWNLOAD THE DATA**

Search for the following two Sentinel-2 Level-2A data products ID in the ONDA platform at: <https://catalogue.onda-dias.eu/catalogue/> (login required, registration is free) and download them:

- S2A\_MSIL2A\_20210315T112111\_N0214\_R037\_T29SMC\_20210315T141433
- S2B\_MSIL2A\_20210817T112119\_N0301\_R037\_T29SMC\_20210817T130929

Once this process takes a while, download both images from the OneDrive platform (**folder DRM2024**).

The Level-2A processing includes a scene classification and an atmospheric correction applied to Top-Of-Atmosphere (TOA) Level-1C orthoimage products. Level-2A main output is an orthoimage Bottom-Of-Atmosphere (BOA) corrected reflectance product.

Additional outputs are an Aerosol Optical Thickness (AOT) map, a Water Vapour (WV) map, and a Scene Classification Map (SCM) together with Quality Indicators (QI) for cloud and snow probabilities at 60 m resolution. Level-2A output image products will be resampled and generated with an equal spatial resolution for all bands (10 m, 20 m or 60 m). Standard distributed products contain the envelope of all resolutions in three distinct folders:

**10 m:** containing spectral bands 2, 3, 4, 8, a True Colour Image (TCI) and an AOT and WV maps resampled from 20 m.

**20 m:** containing spectral bands 2 - 7, the bands 8A, 11 and 12, a True Colour Image (TCI), a Scene Classification map (SCL) and an AOT and WV map. The band B8 is omitted as B8A provides more precise spectral information.

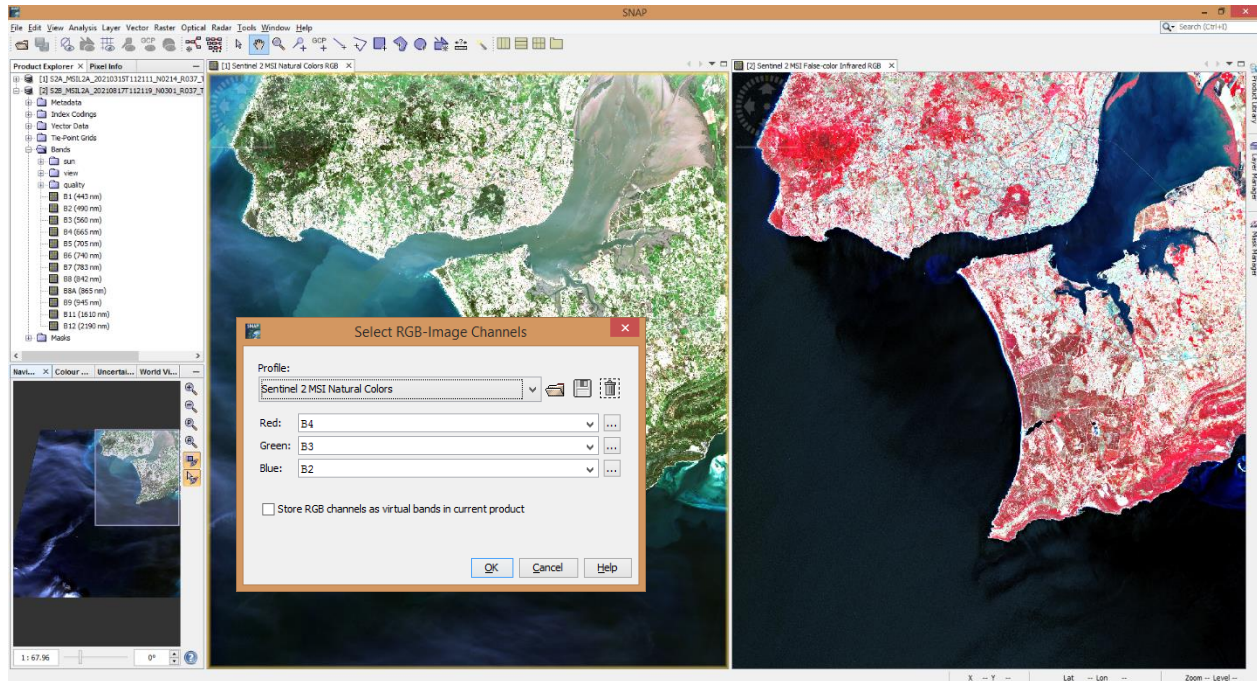
**60 m:** containing all components of the 20 m product resampled to 60 m and additionally the bands 1 and 9. *The cirrus band 10 is omitted, as it does not contain surface information.*

<b>SENTINEL-2</b>  <b>2A launched June 23, 2015</b>  <b>2B launched March 7, 2017</b>	<b>Bands</b>	<b>Central Wavelength (micrometers)</b>	<b>Resolution (meters)</b>
	Band 1 - Coastal Blue	0.443	60
	Band 2 - Blue	0.490	10
	Band 3 - Green	0.560	10
	Band 4 - Red	0.665	10
	Band 5 - Red Edge	0.705	20
	Band 6 - Red Edge	0.740	20
	Band 7 - Red Edge	0.783	20
	Band 8 - NIR	0.842	10
	Band 8A - Red Edge	0.865	20
	Band 9 - Water Vapor	0.945	60
	Band 10 – SWIR (Cirrus)	1.375	60
	Band 11 - SWIR	1.610	20
	Band 12 - SWIR	2.190	20

## OPEN THE PRODUCTS

In SNAP, use the Open Product button in the top toolbar (or **File > Open Product ...**) and browse for the location of the downloaded data. Select the two downloaded zip files and press Open Product.

Select an image and, using the right mouse button, click on “Open RGB Image Window”, and choose one of the available color composites: Natural Colors or False-color Infrared.

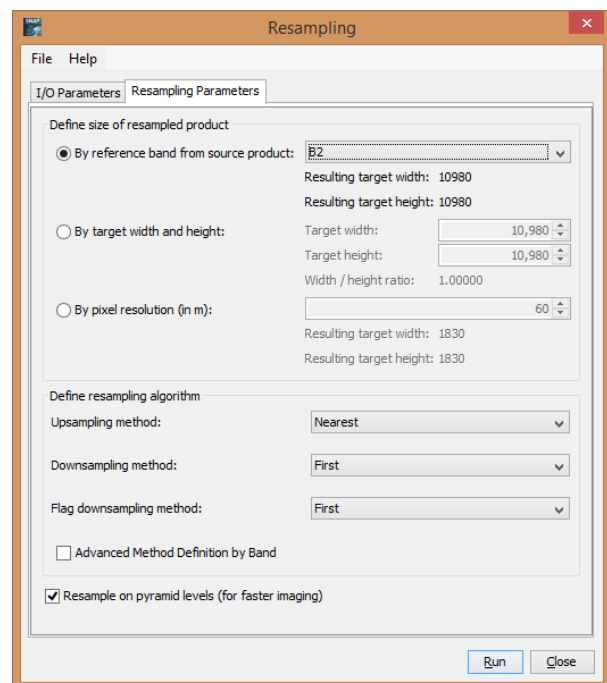
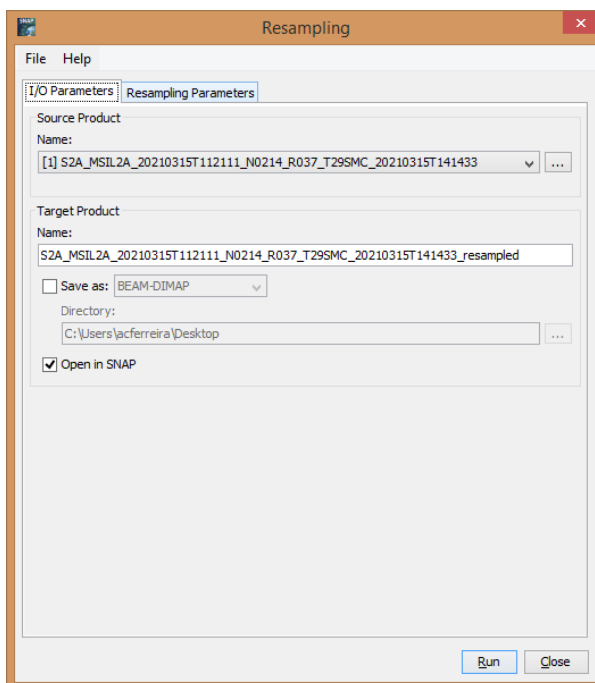


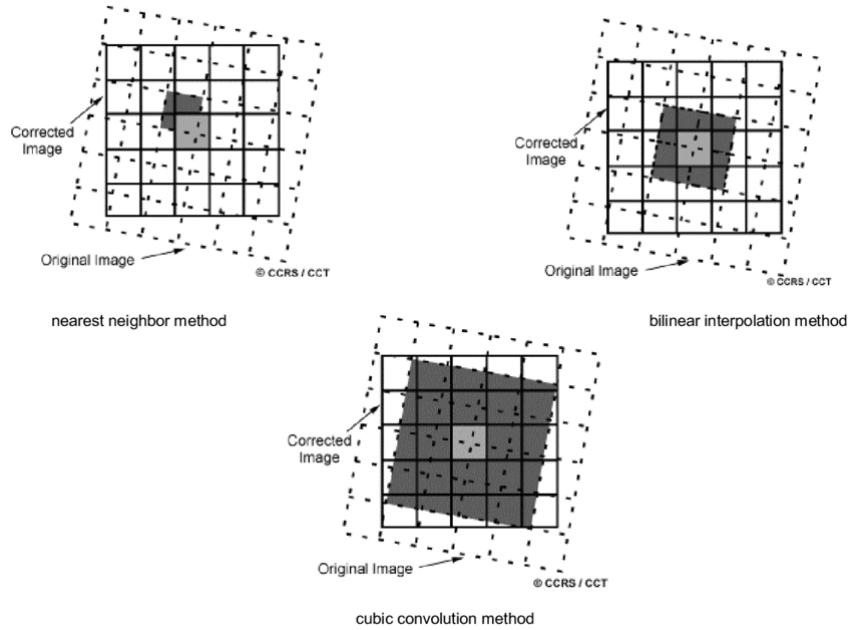
## EXERCISE 1.2

Pre-processing of satellite images might include some basic image operations, such as image resampling, extraction of a region of interest (ROI), band arithmetic operations, multispectral band merging, and multitemporal images collocation.

### RESAMPLE THE PRODUCTS

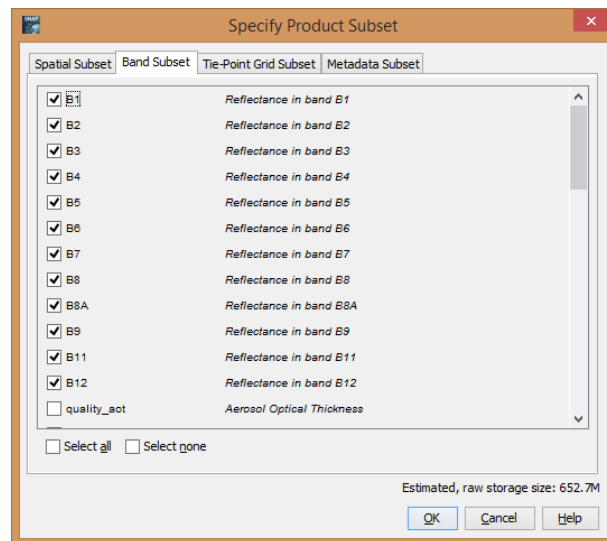
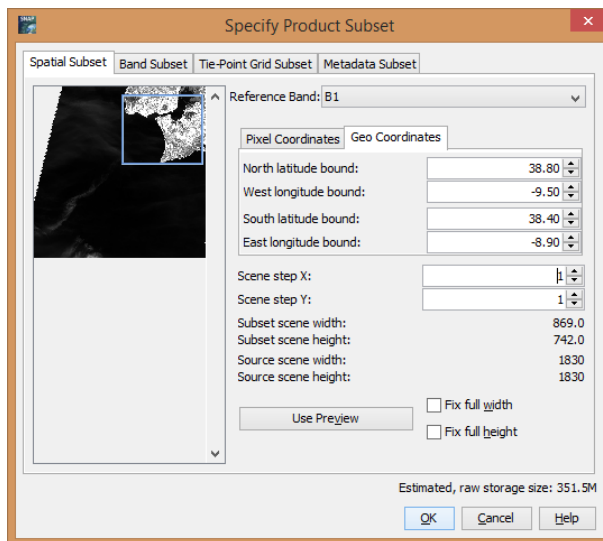
In SNAP, use **Raster > Geometric > Resampling** to resample a multi-size product to a single-size product. A multi-size product is a product in which bands are of different sizes and/or resolutions. This can be useful for those instances when a SNAP feature is not supported for a multi-size product.





## CREATE A SUBSET

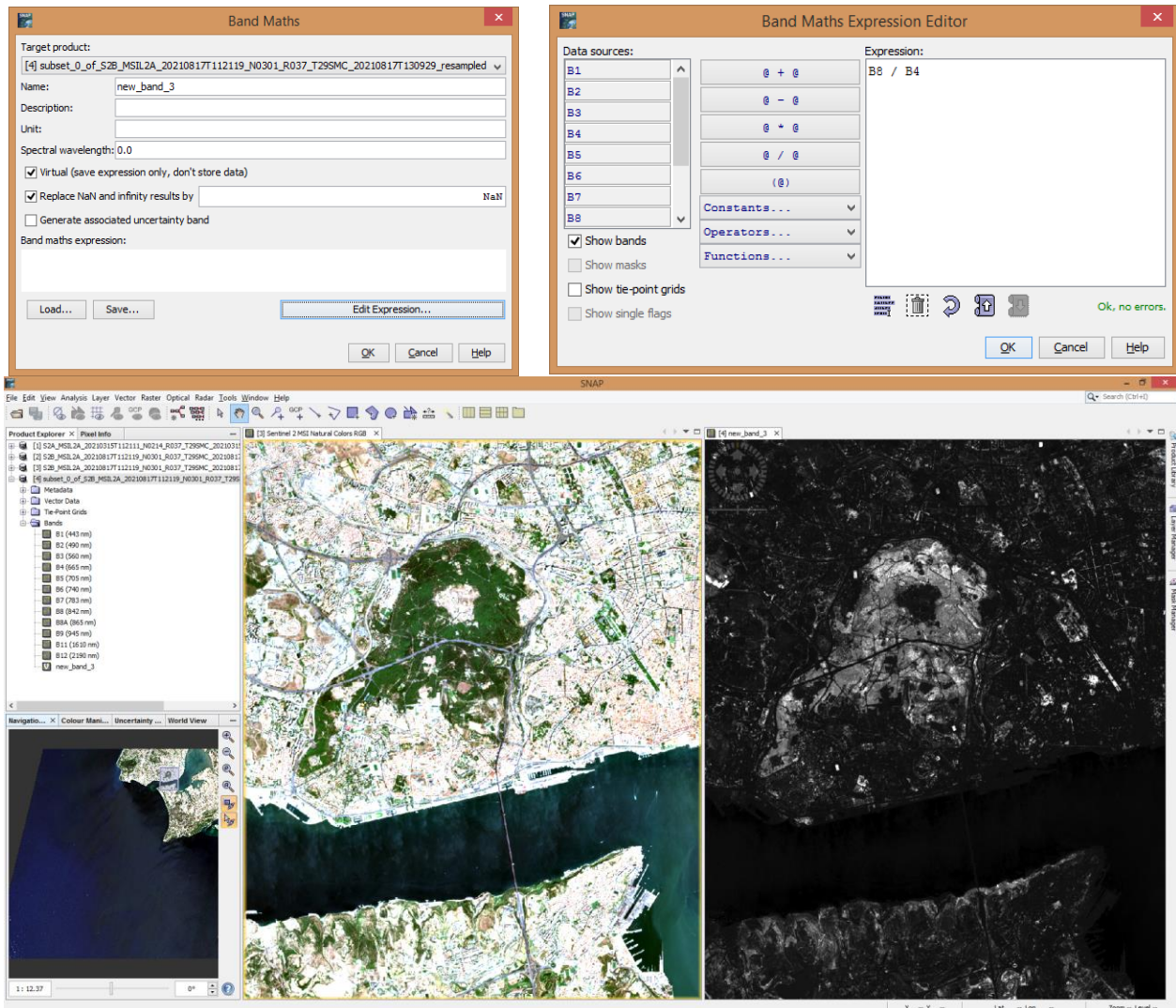
In SNAP, use **Raster > Subset** to create either spatial and/or spectral subsets of a data product. The spatial subset may be given by pixel positions or a geographical polygon.



## CREATE A RATIO BAND

In SNAP, use **Raster > Band Maths** to create new image sample values derived from existing bands, tie-point grids, and flags. The source data can originate from all currently open and spatially compatible input products. The source data is combined with arbitrary mathematical expressions to generate the target data. By default, a new image view is automatically opened for the new sample values.

In this case, consider a simple ratio between Band 8 (NIR) and Band 4 (Red). *A new band with the result is added to the product.*

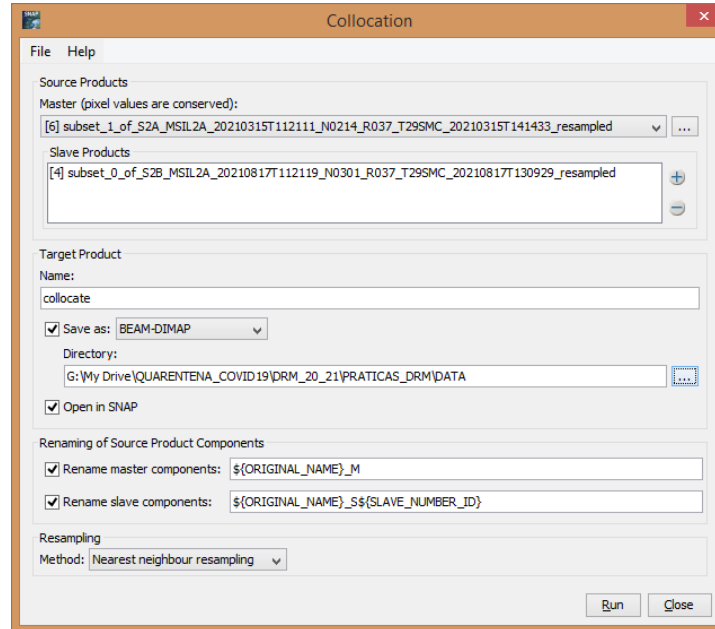




## COLLOCATE OVERLAPPING PRODUCTS

In SNAP, use **Raster > Geometric > Collocation** to collocate spatially overlapping products. Collocating products implies that the pixel values of slave products are resampled into the geographical raster of the master product.

To avoid naming conflicts, the Collocation Tool allows renaming both master and slave components such as bands, flag codings, and bitmask definitions according to a user-defined pattern.



Alternatively, one can use **Radar > Coregistration > Stack Tools > Create Stack** to collocate two spatially overlapping products. Collocating two products imply that the pixel values of one product (the secondary) are resampled into the geographical raster of the other (the reference). When two products are collocated, the band data of the secondary product is resampled into the geographical raster of the reference product.

The collocation algorithm requires accurate geopositioning information for both reference and secondary products (select Product Geolocation). The metadata for the stacked product is copied from the metadata of the reference product. *When the reference image (Master) extent is selected as the output extent for the collocated images, “None” must be used for the resampling type.*

### EXERCISE 1.3

Perform the operations in exercise 1.2 using the **Graph Builder** and **Batch Processing** tools.

Graphs can be created visually with the Graph Builder, processed directly from the DAT, and then saved as XML files. These saved graphs can then also be used as the input for the command line Graph Processing Tool (GPT) with a different set of input data products.

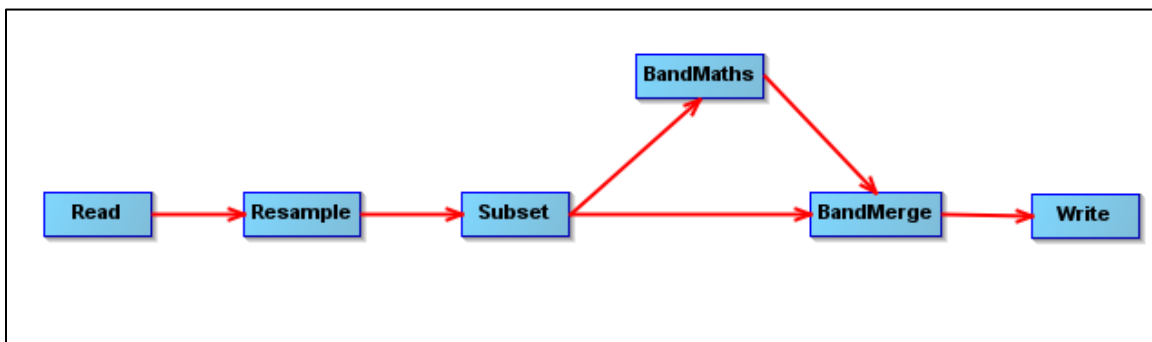
The Graph Builder allows the user to assemble graphs from a list of available operators and connect operator nodes to their sources.

The Batch Processing tool available via the DAT allows you to execute a single reader/writer graph for a set of products.

### CREATE A GRAPH

In SNAP, use **Tools > Graph Builder** to create a graph with a sequence of operators.

In this case, the operations are **Resample**, **Subset**, and **Band Maths**. *However, in Graph Builder, Band Maths only outputs the calculated band (instead of adding it to the original product, as mentioned in exercise 1.2), therefore another operator (Band Merge) has to be added to merge the new band and the original bands of the product.*



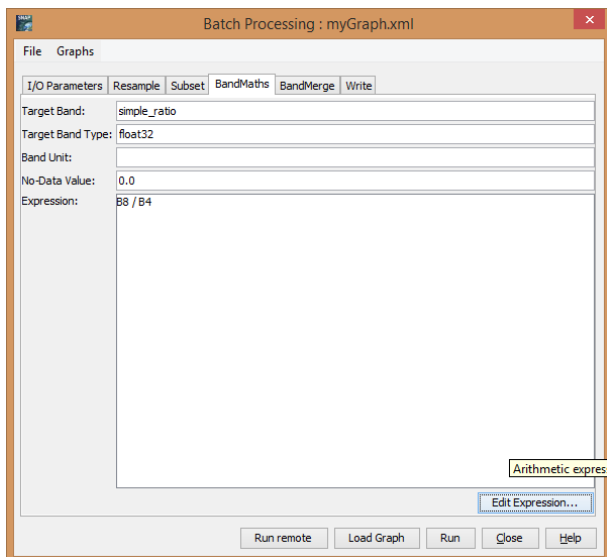
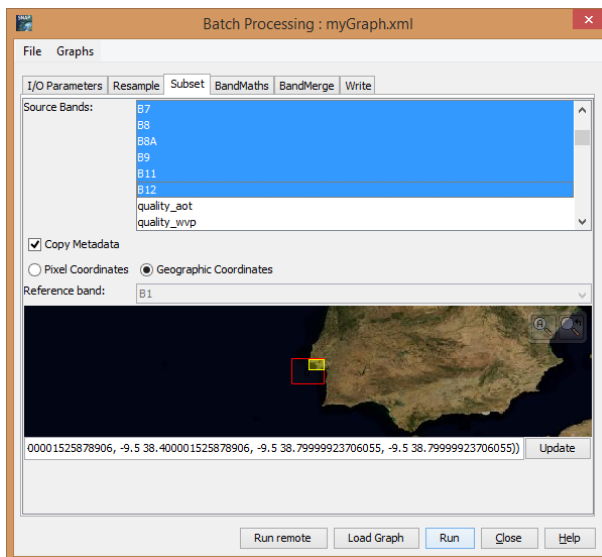
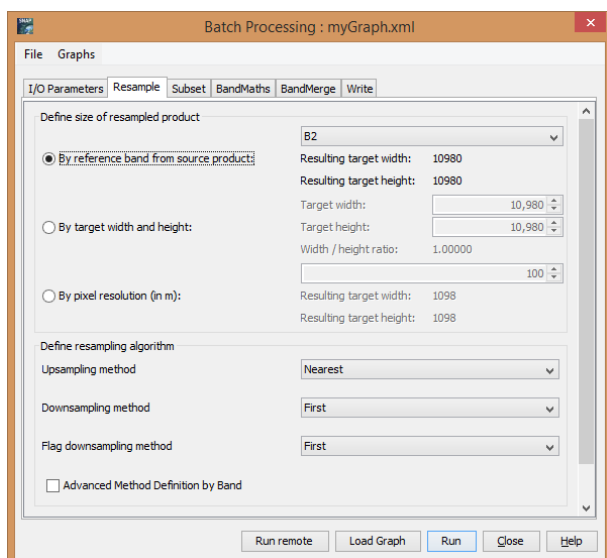
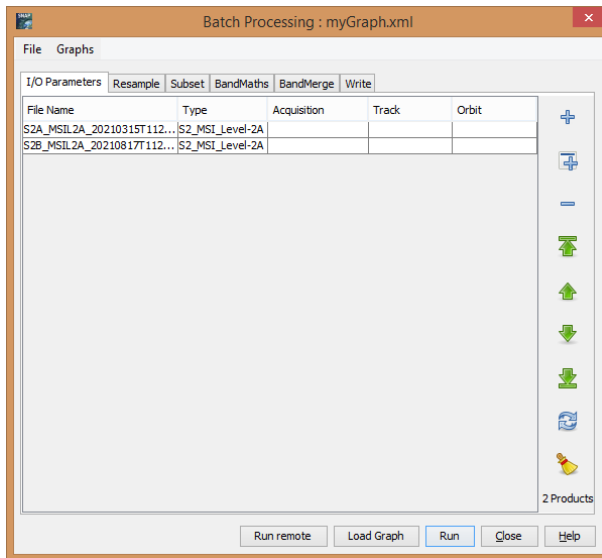
When using Graph Builder, the subset operation requires the insertion of the Region Of Interest (ROI) as a WKT String. For that purpose, use the shapefile ROI.shp that is in the OneDrive platform (**folder DRM2024**). Open this file in QGIS, select the polygon, and extract the WKT string using the WKT Plugin.

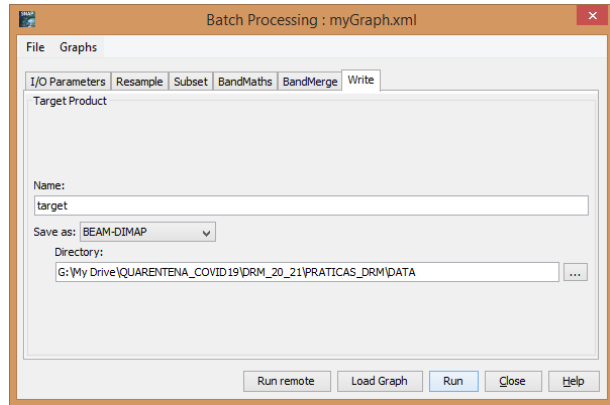
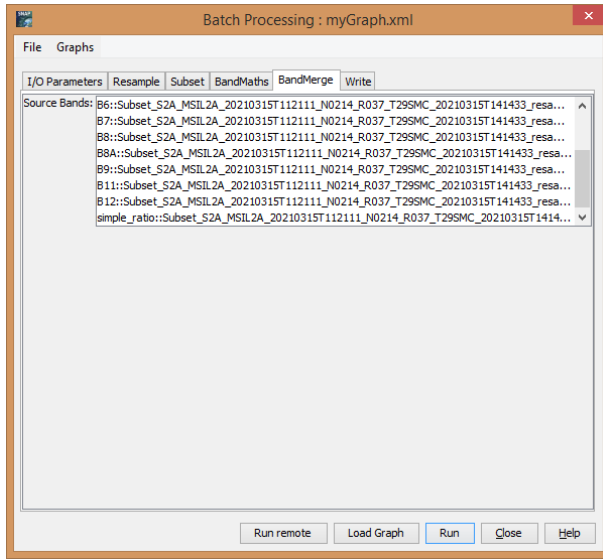
```
MultiPolygon (((-9.5 38.7999999999999716, -8.9000000000000036 38.7999999999999716, -8.9000000000000036 38.3999999999999858, -9.5 38.3999999999999858, -9.5 38.3999999999999858, -9.5 38.7999999999999716)))
```

## RUN THE GRAPH IN BATCH PROCESSING

In SNAP, use **Tools > Batch Processing** to execute a single reader/writer graph for a set of products. Press the "Load" button to browse for a previously saved graph. Next, add products in the IO tab by pressing the "Add" button or dragging and dropping a ProductSet or Products from the Project or Products views. Set the target folder where the output will be written to and then press "Run".

*To avoid all target products having the same name, in the Write tab replace the name of the Target Product with "target".*





## PRACTICAL LESSON 2

### EXERCISE 2.1

Apply several techniques for dimensionality reduction, such as spectral indices (vegetation, built-up, soil, and water indices, for example), principal component analysis, texture, and temporal metrics.

#### CALCULATE SPECTRAL INDICES

In SNAP, use **Raster > Band Maths** to create new image sample values derived from existing bands. Calculate the NDVI<sup>1</sup> (Normalized Difference Vegetation Index) band, the NDWI<sup>2</sup> (Normalized Difference Water Index) band, the SAVI<sup>3</sup> (Soil-Adjusted Vegetation Index), and the NDBI<sup>4</sup> (Normalized Difference Built-up Index).

$$NDVI = \frac{(NIR - Red)}{(NIR + Red)}$$

$$NDWI = \frac{(Green - NIR)}{(Green + NIR)}$$

$$SAVI = \frac{1.5 * (NIR - Red)}{(NIR + Red + 0.5)}$$

$$NDBI = \frac{(SWIR1 - NIR)}{(SWIR1 + NIR)}$$

Alternatively, use **Optical > Thematic Land Processing** to calculate several radiometric indices included in the:

- **Soil Radiometric Indices**
- **Vegetation Radiometric Indices** (NDVI and SAVI)
- **Water Radiometric Indices** (NDWI2)

#### CALCULATE THE PRINCIPAL COMPONENTS OF AN IMAGE

In SNAP, use **Raster > Image Analysis > Principal Component Analysis** to generate the principal component images from a stack of co-registered detected images. The Principal Component Analysis (PCA) consists of a remapping of the information of the input co-registered images into a new set of images. The output images are scaled to prevent negative pixel values.

<sup>1</sup> Rouse, J.W., Jr., Haas, R. H., Schell, J. A., & Deering, D. W. (1974). Monitoring Vegetation Systems in the Great Plains with Ert's, NASA Special Publication. *Proceedings of the Third Earth Resources Technology Satellite- 1 Symposium*, 309–317.

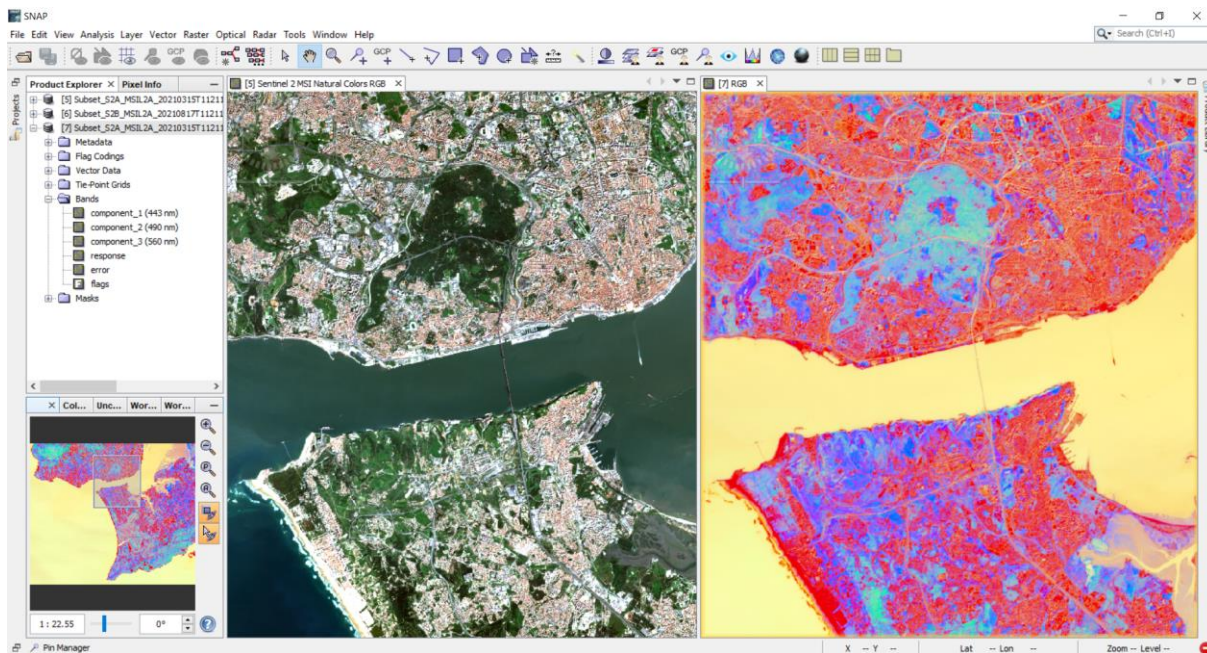
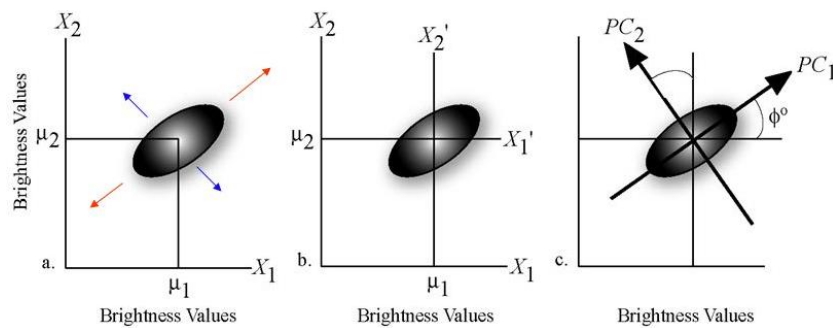
<sup>2</sup> McFeeters, S. K. (1996). The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features. *International Journal of Remote Sensing*, 17(7), 1425–1432. <https://doi.org/10.1080/01431169608948714>.

<sup>3</sup> Huete, A. R. (1988). A soil-adjusted vegetation index (SAVI). *Remote Sensing of Environment*, 25(3), 295–309. [https://doi.org/10.1016/0034-4257\(88\)90106-X](https://doi.org/10.1016/0034-4257(88)90106-X).

<sup>4</sup> Zha, Y., J. Gao, and S. Ni (2003). "Use of Normalized Difference Built-Up Index in Automatically Mapping Urban Areas from TM Imagery." *International Journal of Remote Sensing* 24, no. 3:,583-594.

The PCA operator consists of the following major steps:

- (1) Average the pixels across the input images to compute a mean image;
- (2) Subtract the mean value of each input image (or image from step 1) from itself to produce zero-mean images;
- (3) Compute the covariance matrix from the zero-mean images given in step 2;
- (4) Perform eigenvalue decomposition of the covariance matrix;
- (5) Compute PCA images by multiplying the eigenvector matrix by the zero-mean images given in step 2.



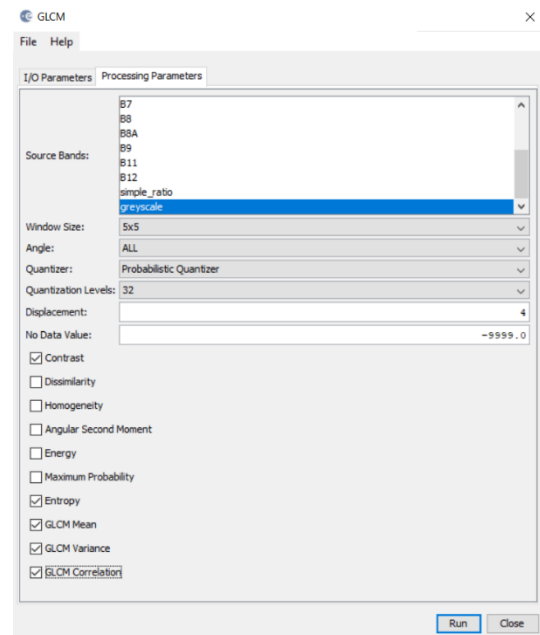
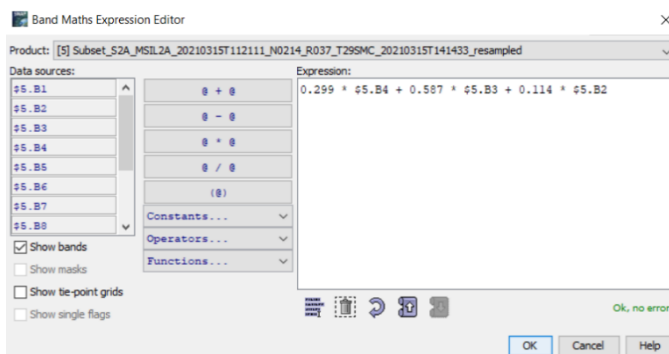
## CALCULATE THE TEXTURE OF A BAND

In SNAP, use **Raster > Image Analysis > Texture Analysis > Grey Level Co-occurrence Matrix** to produce GLCM<sup>5</sup> (Gray Level Co-occurrence Matrix) texture features by making use of spatial information inherent in the image. The texture is the pattern of intensity variations in an image and can be a valuable tool in improving land-cover classification accuracy. Texture information involves the information from neighboring pixels which is important to characterize the identified objects or regions of interest in an image.

The GLCM is one of the most widely used methods to compute second-order texture measures. Several texture features can be computed from the GLCM matrix, each modeling different properties of the statistical relation of pixels co-occurrence estimated within a given moving window and along predefined directions and inter-pixel distances. The GLCM is a measure of the probability of occurrence of two grey levels separated by a given distance in a given direction. The features can be categorized into three groups, i.e. contrast group (contrast, dissimilarity, and homogeneity), orderliness group (angular second moment, maximum probability, and entropy), and statistics group (mean, variance, and correlation).

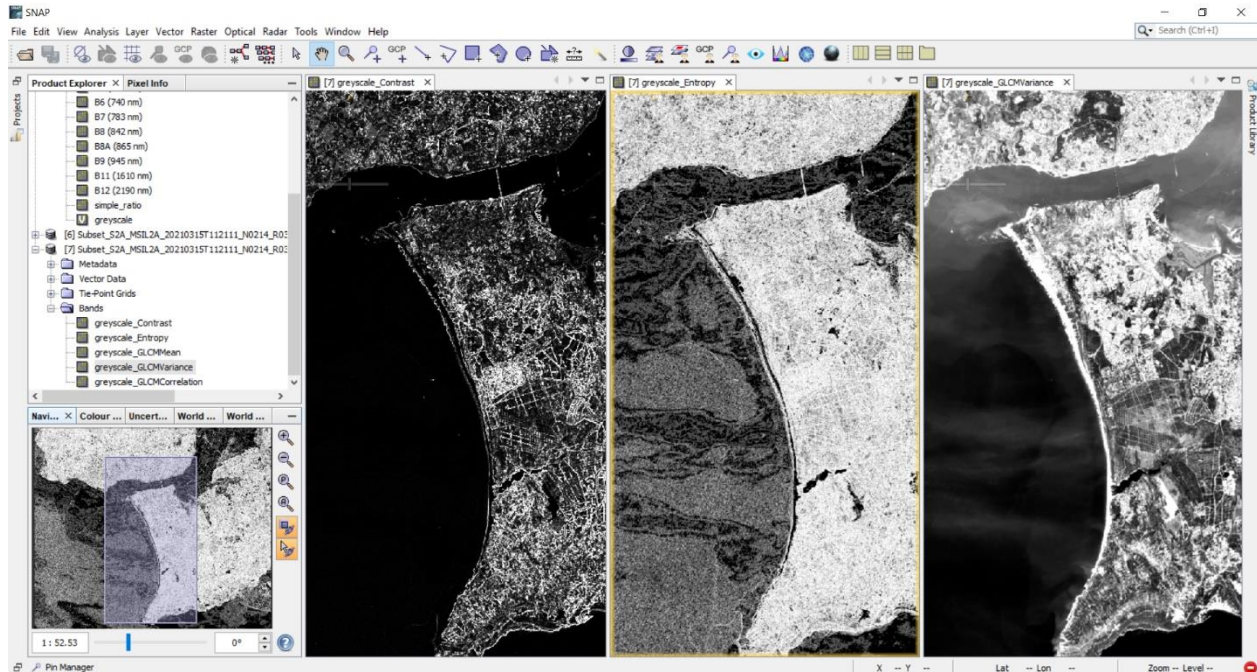
Before calculating the GLCM texture features, convert the RGB image to a grayscale image<sup>6</sup> using the following expression and **Raster > Band Maths**:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$



<sup>5</sup> Haralick, R. M., Dinstein, I., & Shanmugam, K. (1973). Textural Features for Image Classification. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(6), 610–621. <https://doi.org/10.1109/TSMC.1973.4309314>.

<sup>6</sup> [https://mmuratarat.github.io/2020-05-13/rgb\\_to\\_grayscale\\_formulas](https://mmuratarat.github.io/2020-05-13/rgb_to_grayscale_formulas)



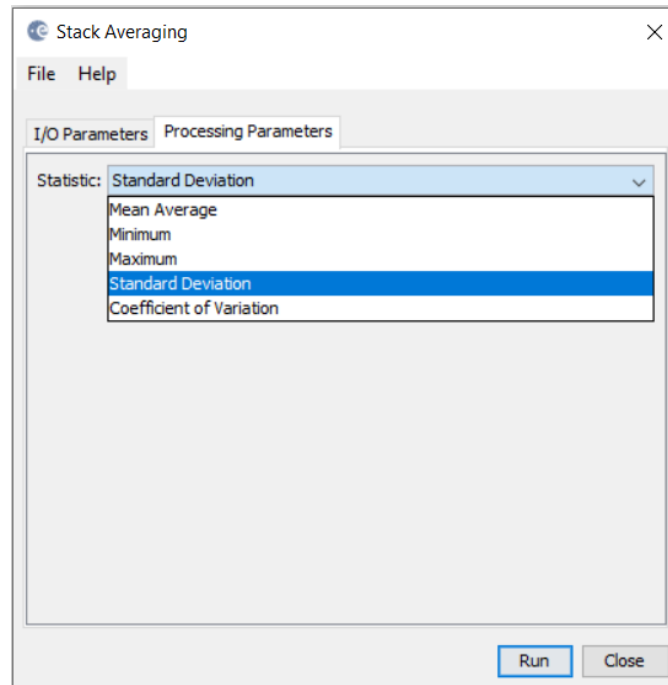
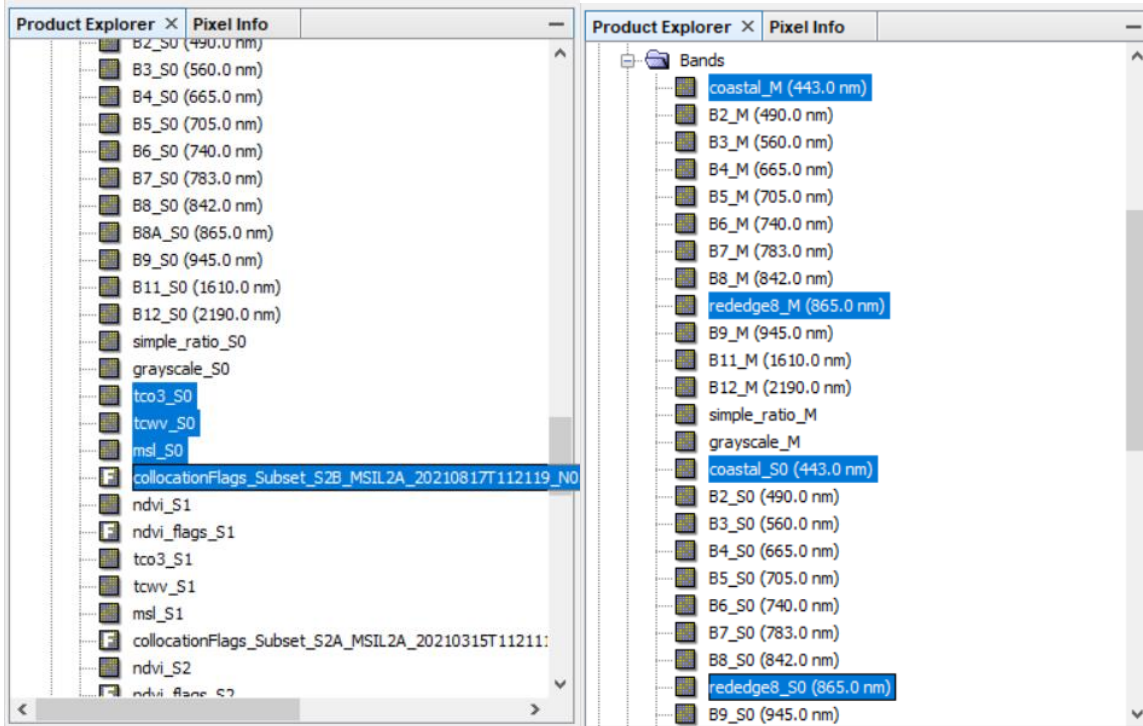
## CALCULATE TEMPORAL METRICS (from a time series of images)

In SNAP, use **Radar > Coregistration > Stack tools > Stack Averaging** to compute statistics (including the mean, minimum, maximum, standard deviation, and coefficient of variation) for the corresponding bands in the given stack product. For example, the mean of all "B2\_" bands.

*However, for Sentinel-2 products, bands that share the same name (before the last underscore), such as the "B1\_", "B11\_" and "B12\_" bands, and also "B8\_" and "B8A\_" bands, will be simultaneously (and incorrectly) considered for the mean (or other metric) calculation. To avoid this, rename the "B1\_" band as "coastal\_" band and "B8A\_" band as "rededge8\_" band.*

**NOTE:** Before calculating the temporal metrics, delete all the colocation flags and other bands, besides the original bands and the subproducts, from the time series generated with **Raster > Geometric > Collocation**.





## PRACTICAL LESSON 3

### EXERCISE 3.1

Perform change detection between two multispectral images using the MAD<sup>7</sup> (**Multivariate Alteration Detector**) algorithm implemented in the Orfeo Toolbox (OTB).

OTB performs change detection between two multispectral images using the MAD algorithm ([https://www.orfeo-toolbox.org/CookBook/Applications/app\\_MultivariateAlterationDetector.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_MultivariateAlterationDetector.html)). The MAD algorithm produces a set of N change maps (where N is the maximum number of bands in first and second input images), with the following properties:

- Change maps are differences between 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 that 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. Change maps are sorted by increasing correlation.

The application will also print the following information:

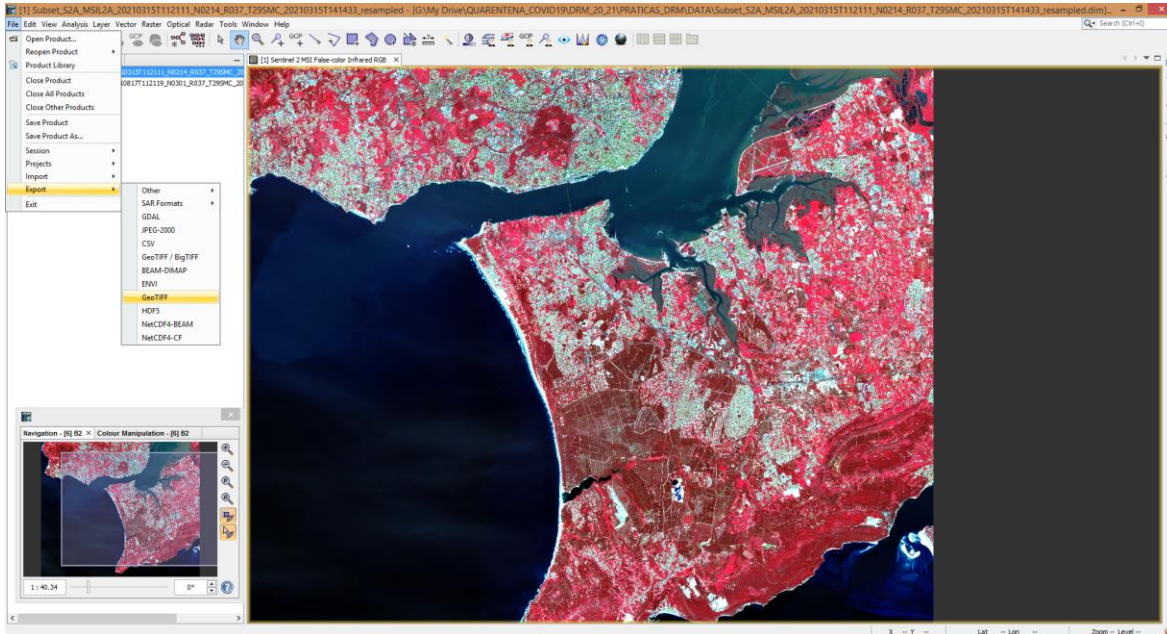
- Mean1 and Mean2 which are the mean values of bands for both input images,
- V1 and V2 which are the two linear transforms that are applied to input image 1 and input image 2 to build the change map,
- Rho, the vector of correlation associated with each change map.

### EXPORT MULTISPECTRAL IMAGES FROM BEAM-DIMAP FORMAT (SNAP) TO GeoTIFF FORMAT

In SNAP, use **File > Export** to export both images, generated with the graph in exercise 1.3, from BEAM\_DIMAP format (\*.dim) to GeoTIFF format (\*.tif), which is recognizable by the QGIS and OTB.

---

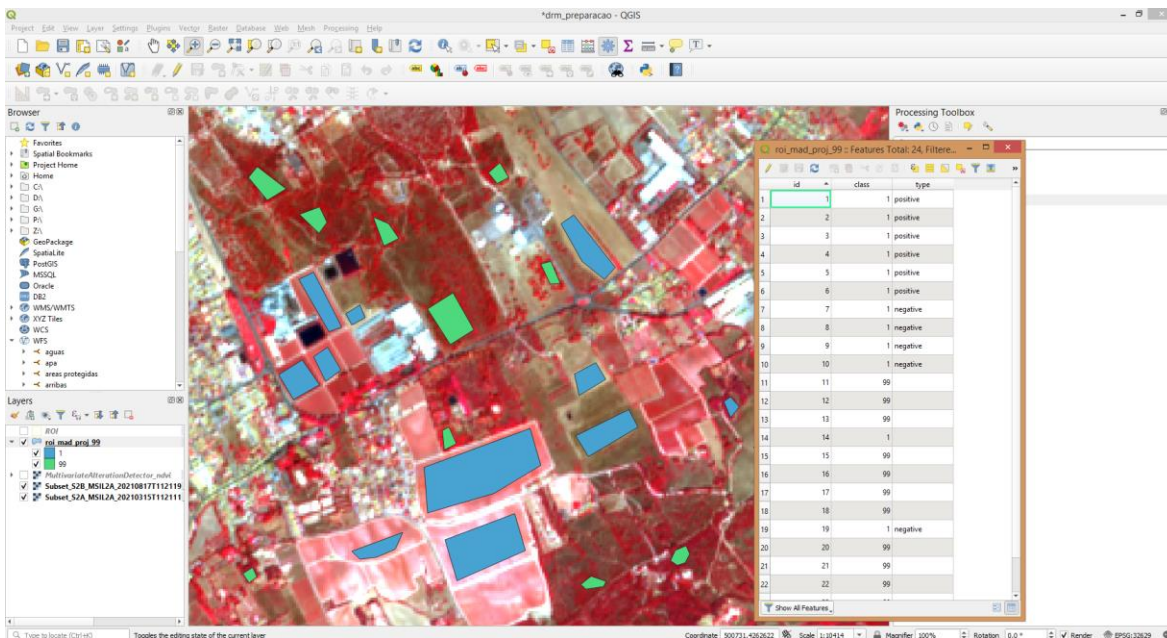
<sup>7</sup> Nielsen, A. A., & Conradsen, K. (1997). Multivariate alteration detection (MAD) in multispectral, bi-temporal image data: A new approach to change detection studies.



## CREATE A POLYGON VECTOR FILE WITH SAMPLES FOR CHANGE DETECTION VALIDATION

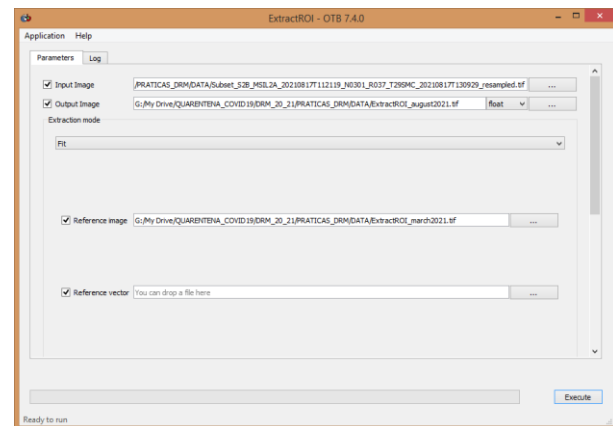
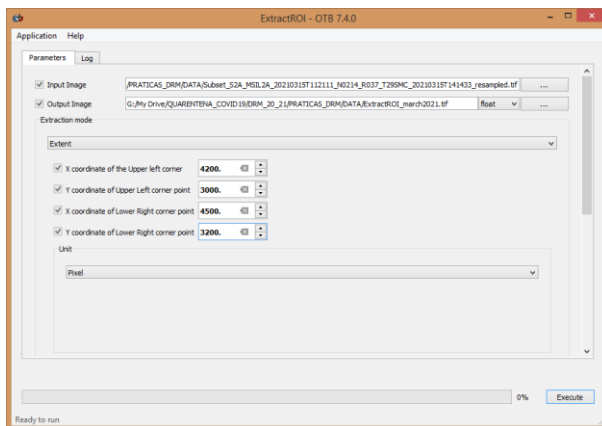
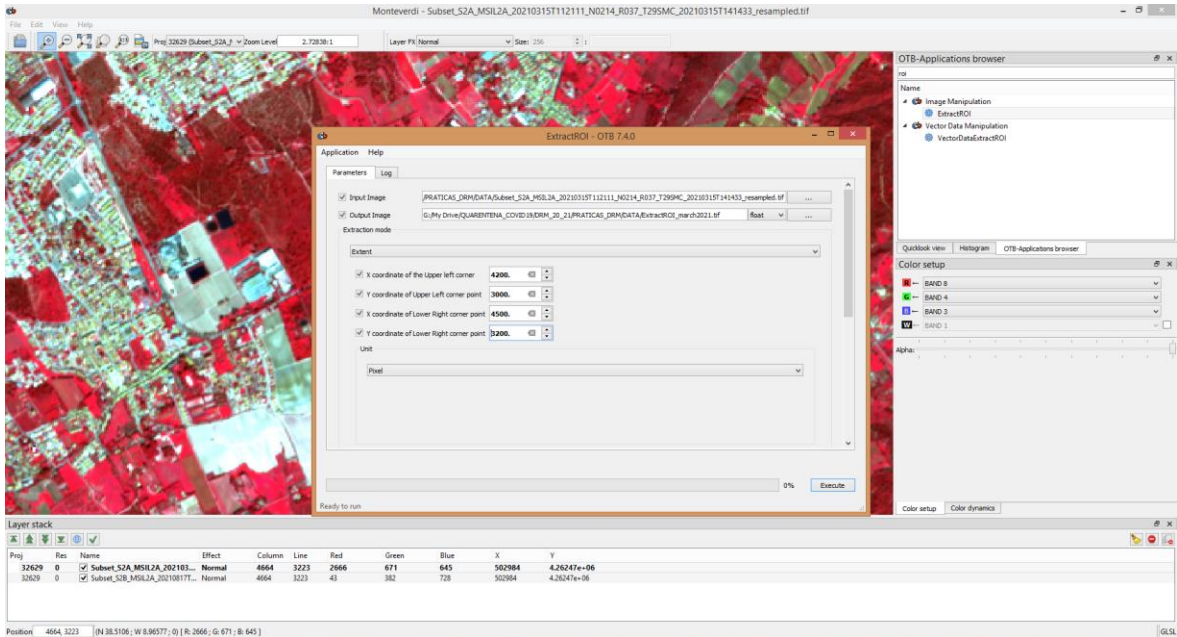
In QGIS, use **Layer > Create Layer > New Shapefile Layer** to create a polygon shapefile with change/no change areas for change detection validation. Use the following coordinate reference system (EPSG: 32629-WGS84/UTM zone 29N) and add a new field (“class”) to the attribute table, with the corresponding value for the two classes (change/no change): 1 for the “change” class and 99 for the “no change” class.

*Instead of 1 for the “change” class and 0 for the “no change” class, use 1 for the “change” class and 99 for “no change” class due to some limitations of the OTB application “ComputeConfusionMatrix” when dealing with the 0 value.*



## CREATE A SUBSET

In OTB, use **Image Manipulation > ExtractROI**<sup>8</sup> to extract an area of interest. First, use the Extraction mode “**Extent**” to subset, by pixel coordinates, one of the images. Then, use the Extraction mode “**Fit**” to subset the other image using a reference image (the one that was previously subset).



<sup>8</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ExtractROI.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ExtractROI.html)

## CALCULATE NDVI BANDS

In OTB, use **Feature Extraction > RadiometricIndices**<sup>9</sup> to compute radiometric indices (in this case, the **NDVI**) using the relevant channels of the input image.

*As Sentinel-2 Level-2A products provide values coded in JPEG2000 with the same quantification value of 10,000 as for Level-1C products, a factor of 1/10,000 needs to be applied to Level-2A digital numbers (DN) to retrieve physical surface reflectance (SR) values. The physical values range from 1 (minimum reflectance 10<sup>-4</sup>) to 10,000 (reflectance 1), but values higher than 1 can be observed in some cases due to specific angular reflectivity effects. The value 0 is reserved for “No Data”<sup>10</sup>.*

This is particularly necessary for indices that use factors, such as the SAVI, since those factors were estimated for physical reflectance values. So, before calculating the indices, in OTB, use **Image Manipulation > BandMathX**<sup>11</sup> to perform 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 library. Separating expressions by semicolons ( ; ) will concatenate their results into a unique multiband output image.

In this case, the expression to be used to convert, at the same time, all the bands of an image from DN to SR values should be:

Since we intend to convert all the bands at the same time, the expression should be:

$$\text{im1b1}/10000; \text{im1b2}/10000; \dots; \text{im1bn}/10000$$

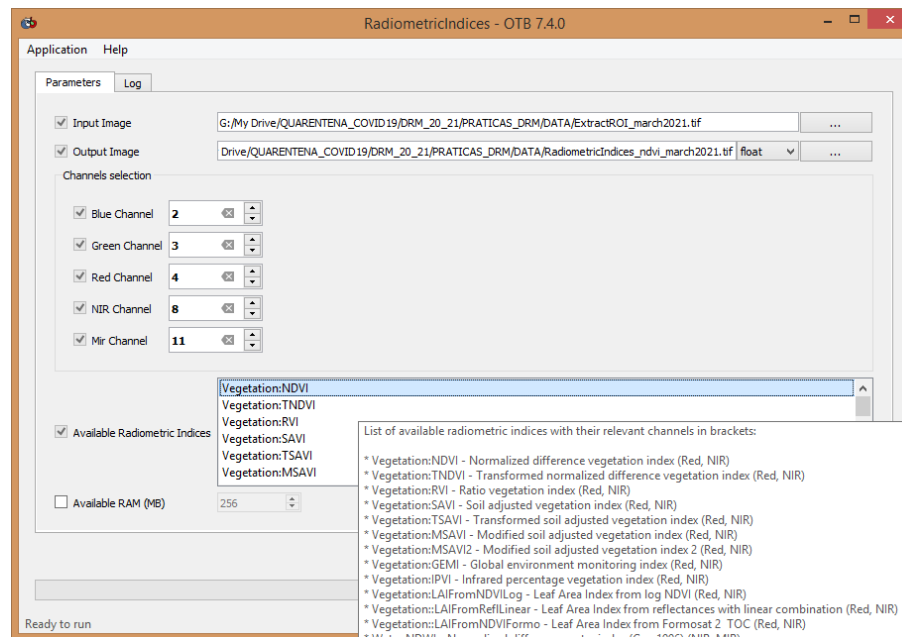
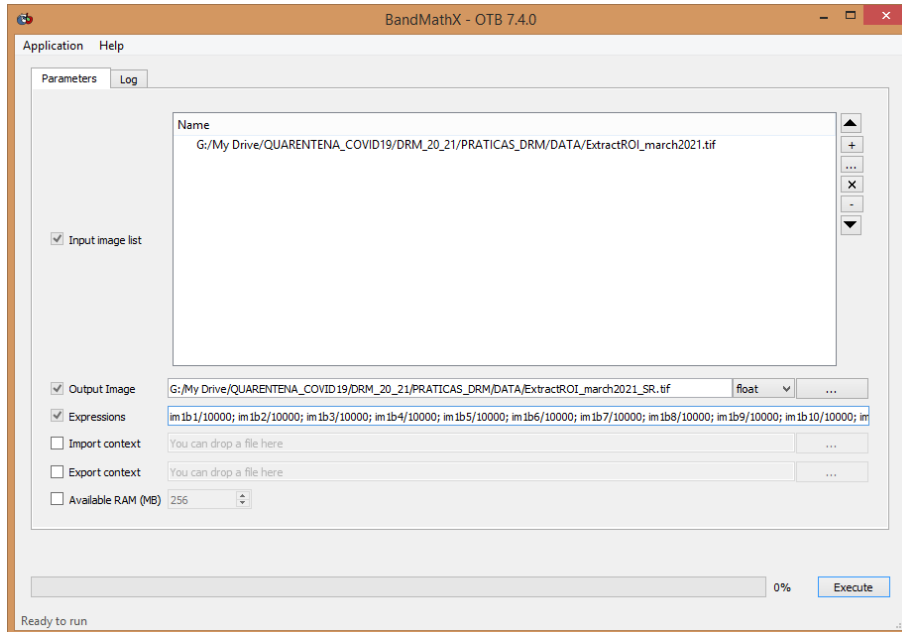
where **im** identifies the image, that in this case is unique, and **b** identifies the band, that in this case goes from 1 up to n.

DN (NIR band)= 4760	<b>NDVI= 0.906</b>	$NDVI = \frac{(NIR - Red)}{(NIR + Red)}$	SR (NIR band)= 0.4760	<b>NDVI= 0.906</b> (range from -1 to 1)
DN (Red band)= 235	<b>SAVI= 1.359</b>	$SAVI = \frac{1.5 * (NIR - Red)}{(NIR + Red + 0.5)}$	SR (Red band)= 0.0235	<b>SAVI= 0.679</b> (range from -1 to 1)

<sup>9</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_RadiometricIndices.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_RadiometricIndices.html)

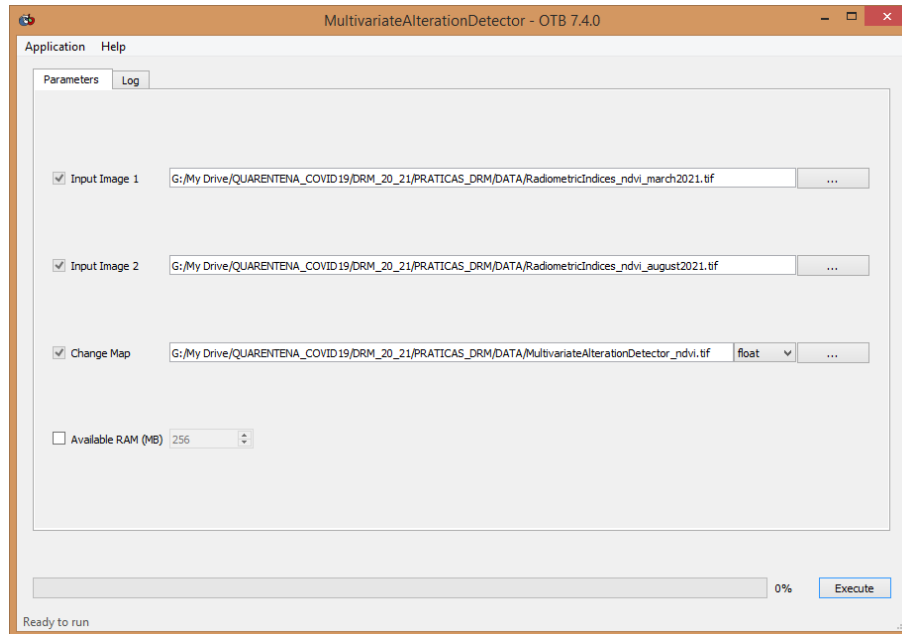
<sup>10</sup> <https://www.mdpi.com/2072-4292/9/6/584> (in section 3.5, page 12/81)

<sup>11</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_BandMathX.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_BandMathX.html)



## APPLY THE MULTIVARIATE ALTERATION DETECTOR (to a pair of images)

In OTB, use **Change Detection > MultivariateAlterationDetector**<sup>12</sup> to perform change detection between two NDVI images, created in the previous step, using the MAD algorithm.



## CREATE A BINARY MASK (CHANGE/NO CHANGE AREAS)

In OTB, use **Image Manipulation > BandMath**<sup>13</sup> to define a threshold condition to generate a binary mask with: 1 for the class “change” and 99 for the class “no change”, as defined previously for the attribute “class” of the validation shapefile.

After the definition of a threshold value by visual interpretation (as an example let’s consider that the threshold is the value 1.2), the expression (if-then-else operator: `(condition ? value_true : value_false)`) should be:

$$(im1b1 \geq 1.2 \text{ or } im1b1 \leq -1.2 ? 1 : 99)$$

where **im** identifies the image, that in this case is unique, and **b** identifies the band (also unique in this case), the symbol ( ? ) separates the condition from the true value, and the symbol ( : ) separates the else value. In this case, we have an operator ( or ), since the change might be positive or negative.

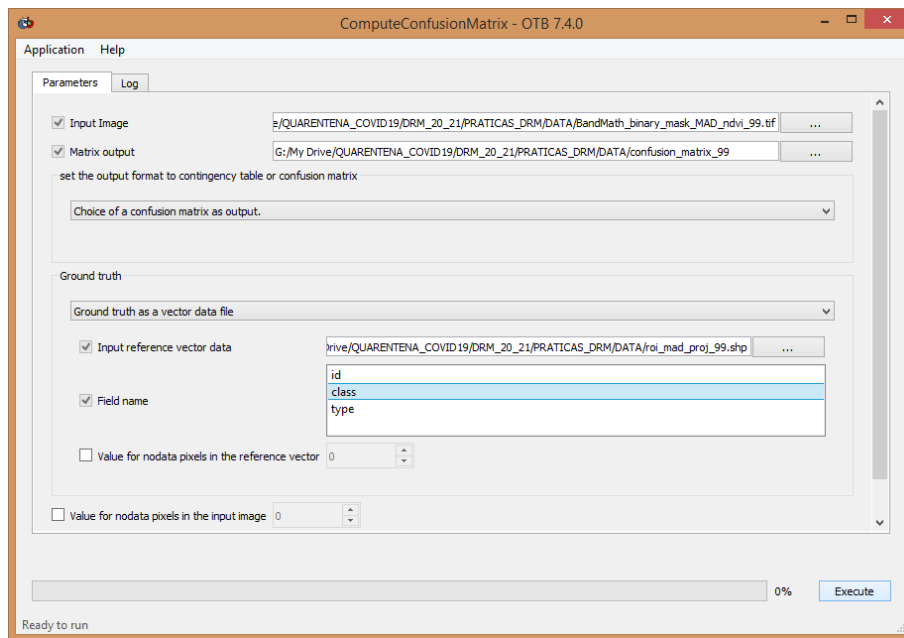
<sup>12</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_MultivariateAlterationDetector.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_MultivariateAlterationDetector.html)

<sup>13</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_BandMath.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_BandMath.html)

## COMPUTE A CONFUSION MATRIX

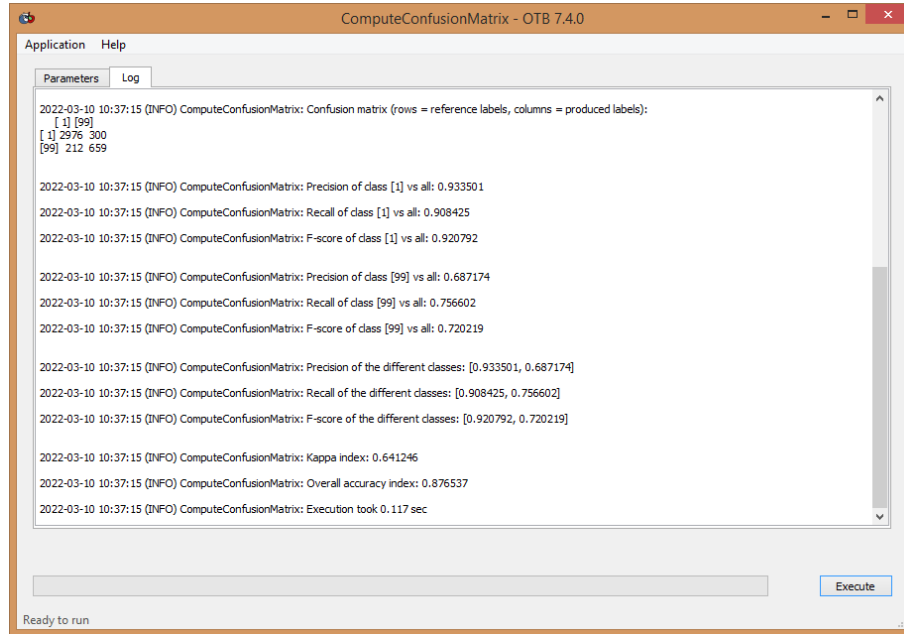
In OTB, use **Learning > ComputeConfusionMatrix**<sup>14</sup> to compute the confusion matrix of a classification.

This application computes the confusion matrix of a classification map relative to a ground truth dataset. The ground truth can be provided as either 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. *In OTB, 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.



<sup>14</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ComputeConfusionMatrix.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ComputeConfusionMatrix.html)





A confusion (or error) matrix compares information from reference samples with the corresponding information on the produced map the same samples. The matrix is a square array of numbers set out in rows and columns which express the labels of samples assigned to a particular category in one classification relative to the labels of samples assigned to a particular category in another classification.

One of the classifications, is assumed to be correct and is termed the reference data and is displayed in rows in OTB. Thus, OTB displays the map labels or classified data generated from the remotely sensed image in columns. Thus, two labels from each sample are compared to one another:

**Reference data labels:** The class label or value of the accuracy assessment site, which is derived from data collected that is assumed to be correct; and

**Classified data or map labels:** The class label or value of the accuracy assessment site derived from the map.

Error matrices are very effective representations of map accuracy because the individual accuracies of each map category are plainly described along with both the errors of inclusion (commission errors) and errors of exclusion (omission errors) present in the map. A commission error occurs when an area is included in an incorrect category. An omission error occurs when an area is excluded from the category to which it belongs. Every error on the map is an omission from the correct category and a commission to an incorrect category.

In addition to clearly showing errors of omission and commission, the error matrix can be used to compute not only the **overall accuracy**, but also the producer's accuracy or **recall** (the complement of the omission error) and the user's accuracy or **precision** (complement of the commission error). Overall accuracy is simply the sum of the major diagonal (i.e., the correctly classified sample units) divided by the total number of sample units in the error matrix. Recall and precision are ways of representing individual category accuracies instead of just the overall classification accuracy.

		PRODUCED LABELS				
		Change	No Change	Sum rows	Recall	F1-score
REFERENCE LABELS	Change	2976	300	3276	2979/3276= <b>0.91</b>	2x(0.93x0.91)/(0.93+0.91)= <b>0.92</b>
	No Change	212	659	871	659/871= <b>0.76</b>	2x(0.69x0.76)/(0.69+0.76)= <b>0.72</b>
Sum columns		3188	959	4147 (*)		
Precision		2976/3188= <b>0.93</b>	659/959= <b>0.69</b>			
<b>Overall Accuracy</b> = (2976+659)/4147= <b>0.88</b> <b>Kappa Coefficient</b> = (4147*(2976+659)-(3276x3188+871x959))/ (4147 <sup>2</sup> -(3276x3188+871x959))= <b>0.64</b>						

(\*) the sum of all column's totals must be equal to the sum of all rows totals.

The **Kappa** coefficient is a measure of overall agreement of a matrix. In contrast to the overall accuracy — the ratio of the sum of diagonal values to total number of cells counts in the matrix — the Kappa coefficient takes also non-diagonal elements into account. Therefore, the Kappa coefficient measures the proportion of agreement after chance agreements have been removed from considerations. A Kappa value of 1 represents perfect agreement while a value of 0 represents no agreement. Kappa has the following formulation:

$$K = \frac{N \sum_{i=1}^n m_{i,i} - \sum_{i=1}^n G_i C_i}{N^2 - \sum_{i=1}^n G_i C_i}$$

where  $i$  is the class number;  $N$  is the total number of classified pixels that are being compared to ground truth;  $m_{i,i}$  is the number of pixels belonging to the ground truth class  $i$ , that have also been classified with a class  $i$  (i.e., values found along the diagonal of the confusion matrix);  $C_i$  is the total number of classified pixels belonging to class  $i$ ; and  $G_i$  is the total number of ground truth pixels belonging to class  $i$ .

Besides, F1-score is also used in machine-learning. F1-score is the weighted average of precision and recall. Therefore, this score takes both commission and omission errors into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if commission and omission errors have similar cost. If the cost of commission and omission errors are very different, it's better to look at both precision and recall.

$$F1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

## PRACTICAL LESSON 4

### EXERCISE 4.1

Classify a composite with two Sentinel-2 images acquired in March and August 2021 (*subset\_0\_collocate.tif*) using the Random Forest classifier. The image composite has 26 bands, 13 for each acquisition date (B1-B8, B8A, B9, B11-B12, *simple\_ratio*).

The training and validation data are in a polygon shapefile created in QGIS (*reference\_samples.shp*), whose polygons correspond to six different classes adopted from the Level 1 of the COSsim (a Simplified Land Cover and Land Use Map for Portugal produced by DGT) nomenclature<sup>15</sup> adopted (1: Built-up Areas, 2: Agriculture; 3: Forest; 4: Bushes and spontaneous herbaceous vegetation; 5: Surfaces without vegetation; 6: Water and wetlands).

Both types of data can be downloaded from the OneDrive platform (**folder DRM2024**).

Orfeo ToolBox offers a set of applications to perform supervised or unsupervised pixel-based image classification<sup>16</sup>. This framework allows to learn from multiple images and using several-machine learning methods such as SVM, Bayes, *k*-NN, 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 the machine learning model from samples;
7. Perform the classification by applying the model.

### STEP 1: COMPUTE SAMPLES STATISTICS FOR EACH IMAGE (`PolygonClassStatistics`)

In OTB, use **Learning > PolygonClassStatistics**<sup>17</sup> to compute statistics on a training polygon set. Process a set of geometries intended for training (they should have a field giving the associated class). The geometries are analyzed against a support image to compute statistics.

<sup>15</sup> [https://geo2.dgterritorio.gov.pt/atom-dgt/COSsim/Nomenclatura\\_COSsim.pdf](https://geo2.dgterritorio.gov.pt/atom-dgt/COSsim/Nomenclatura_COSsim.pdf)

<sup>16</sup> <https://www.orfeo-toolbox.org/CookBook/recipes/pbclassif.html#pixel-based-classification>

<sup>17</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_PolygonClassStatistics.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_PolygonClassStatistics.html)

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 an 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:

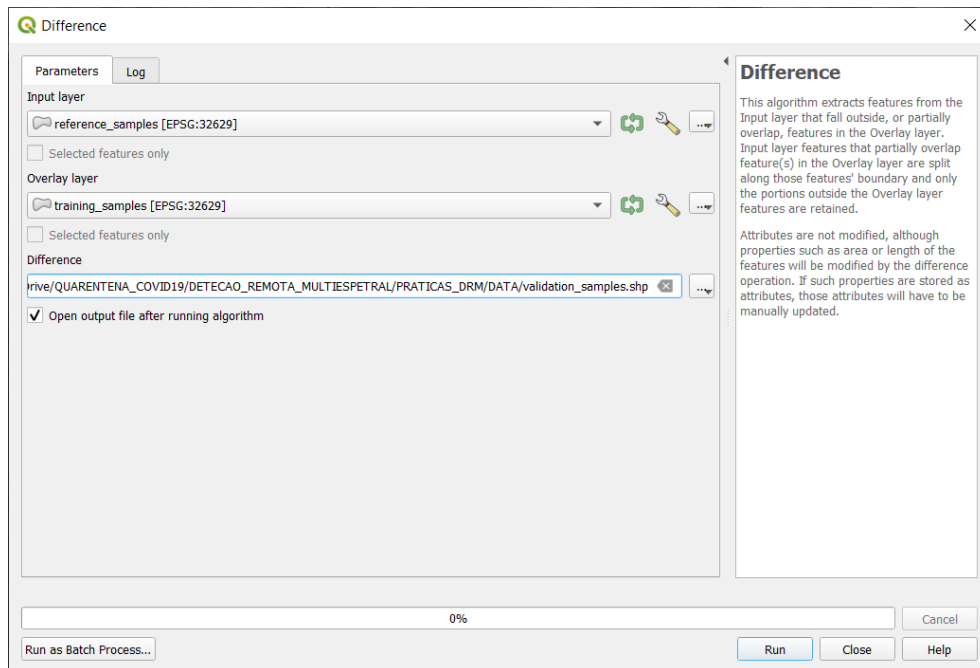
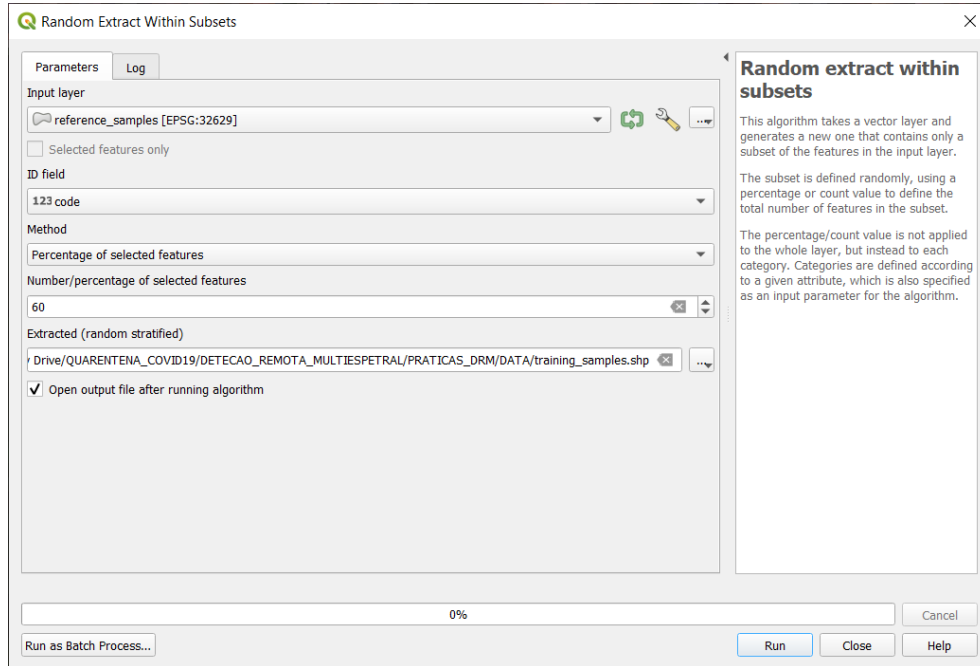
- polygons: select pixels whose center falls inside the polygon;
- lines: select pixels intersecting the line;
- points: select closest pixel to the provided point.

These training geometries must be, previously, created by the user in ArcMap or QGIS, adopting the reference system of the image to be classified. For this exercise, use the polygon vector file already created (named *reference\_samples.shp*) which contains 36 samples: 6 for Built-up Areas (Class 1); 7 for Agriculture (Class 2); 6 for Forest (Class 3); 7 for Bushes and spontaneous herbaceous vegetation (Class 4); 6 for Surfaces without vegetation (Class 5); and 4 for Water and wetlands (Class 6).

These samples are collected by digitizing, on screen, polygons corresponding to those land use classes, having the images in the background. Each polygon should contain the same spectral characteristics, and the polygons defined for a certain class must represent all the spectral variability of that class! *Add a field, named "code" for example, to the attribute table to add an integer value corresponding to each class. The values in this field shall be cast into integers. Only geometries with an integer field will be considered by OTB.* Note that, this is a very simplistic example just for you to know how this works! In a real application, the number of reference samples should be much higher, especially when working with machine-learning classification algorithms.

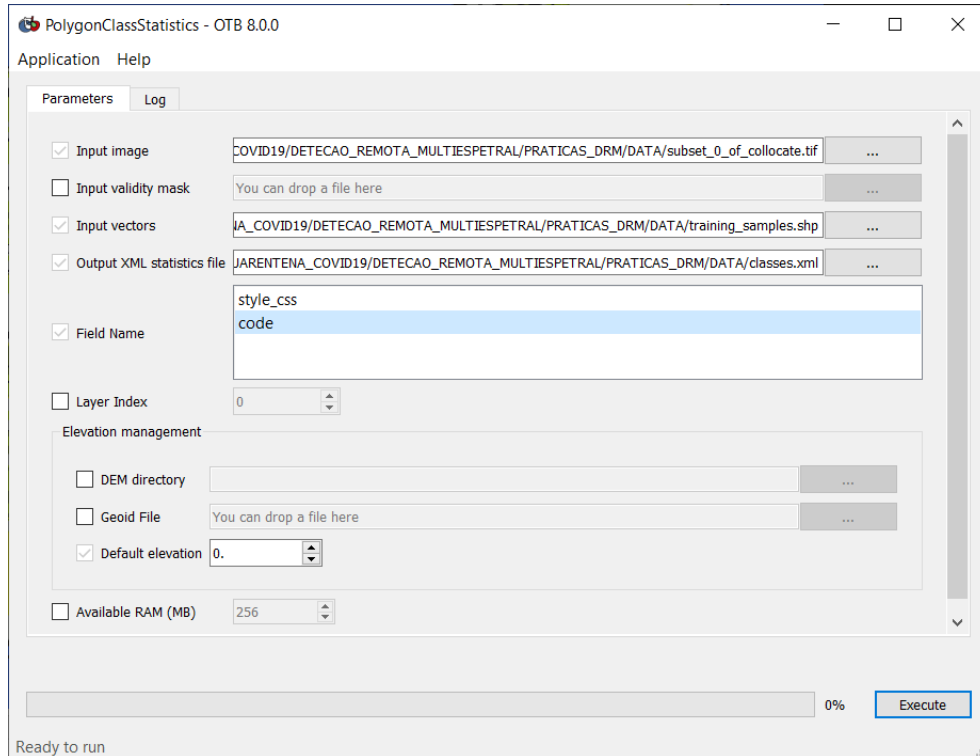
Using the QGIS tool **Vector Selection > Random extract within subsets**, generate a new vector layer (name this file as *training\_samples.shp*) containing only a random subset of the features in the original layer, using a percentage (in this case, 60%), for each class (in this case, using the attribute "code" with the number corresponding to each class).

Using the QGIS tool **Vector overlay > Difference**, extract features from the input layer (*reference\_samples.shp*) that fall outside features in the overlay layer (*training\_samples.shp*), to generate a new shapefile with the remaining 40% of the polygons in the original shapefile (name it *validation\_samples.shp*).



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`.

The `-field` parameter is the name of the field that corresponds to class labels in the input geometries (in this case, the attribute “code”). *As mentioned before, the values in this field shall be cast into integers.*



The Output XML statistics file (here named *class.xml*) looks like this (use WordPad to open the file):

```
<?xml version="1.0" ?>
<GeneralStatistics>
  <Statistic name="samplesPerClass">
    <StatisticMap key="1" value="579" />
    <StatisticMap key="2" value="378" />
    <StatisticMap key="3" value="2339" />
    <StatisticMap key="4" value="898" />
    <StatisticMap key="5" value="484" />
    <StatisticMap key="6" value="1897" />
  </Statistic>
  <Statistic name="samplesPerVector">
    <StatisticMap key="0" value="71" />
    <StatisticMap key="1" value="42" />
    <StatisticMap key="10" value="581" />
    <StatisticMap key="11" value="302" />
    <StatisticMap key="12" value="208" />
    <StatisticMap key="14" value="628" />
    <StatisticMap key="15" value="62" />
    <StatisticMap key="16" value="32" />
    <StatisticMap key="17" value="103" />
    <StatisticMap key="18" value="28" />
    <StatisticMap key="19" value="321" />
    <StatisticMap key="2" value="47" />
    <StatisticMap key="22" value="1012" />
    <StatisticMap key="23" value="885" />
    <StatisticMap key="3" value="218" />
    <StatisticMap key="4" value="88" />
    <StatisticMap key="5" value="132" />
  </Statistic>
</GeneralStatistics>
```

```
<StatisticMap key="6" value="168" />
<StatisticMap key="7" value="191" />
<StatisticMap key="8" value="641" />
<StatisticMap key="9" value="815" />
</Statistic>
</GeneralStatistics>
```

## STEP 2: SELECT SAMPLE POSITIONS FOR EACH IMAGE (SampleSelection)

In OTB, use **Learning > SampleSelection**<sup>18</sup> to select samples from a training vector data set. 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 have been previously 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 an input to this application (Input Statistics).

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 select the sample positions, there are two available sampling techniques:

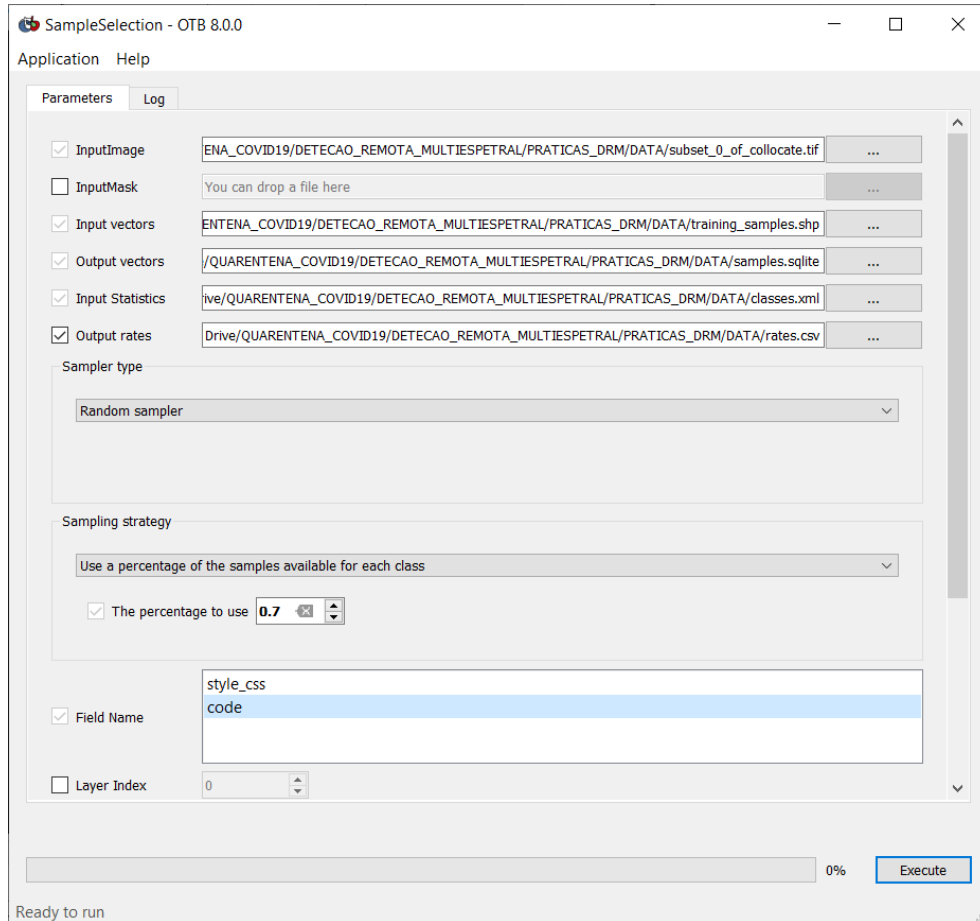
- **Random:** Randomly select samples while respecting the sampling rate.
- **Periodic:** Sample periodically using the sampling rate.

The application will make sure that samples span 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.

---

<sup>18</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_SampleSelection.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_SampleSelection.html)

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

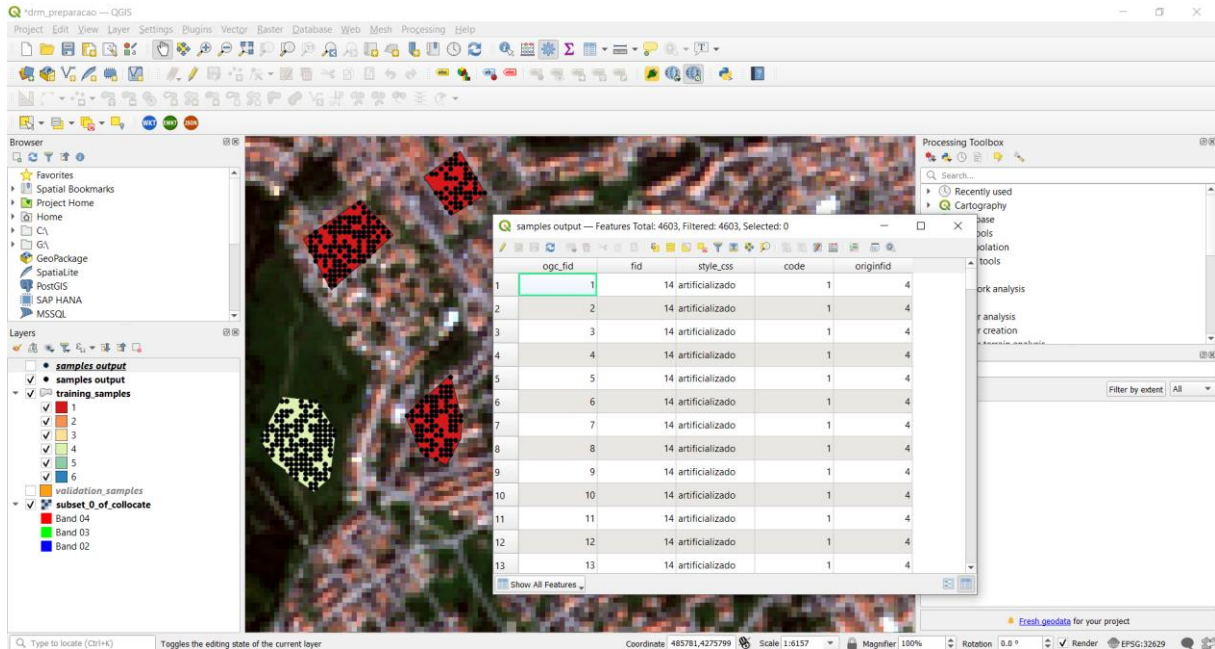


The Output rates file (here named *rates.csv*) written by the optional `-outrates` parameter sums-up what has been done during sample selection (use WordPad to open the file).

```
#className requiredSamples totalSamples rate
1      405      579      0.7
2      265      378      0.7
3     1637     2339      0.7
4      629      898      0.7
5      339      484      0.7
6     1328     1897      0.7
```

The Output vectors file (here named *samples.sqlite*) might be viewed using QGIS (open it as a vector file). The black dots show the samples that have been selected.





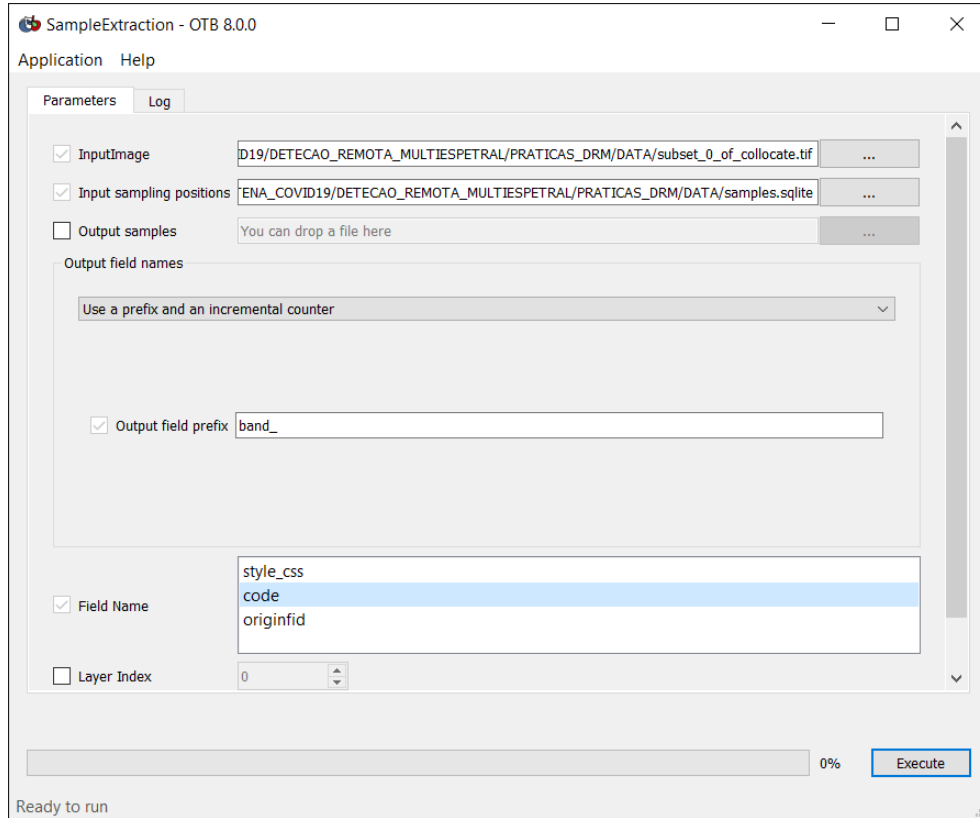
### STEP 3: EXTRACT SAMPLES MEASUREMENTS FOR EACH IMAGE (SampleExtraction)

In OTB, use **Learning > SampleExtraction**<sup>19</sup> to extract samples values from an image. The application extracts samples values from an image using positions contained in a vector data file.

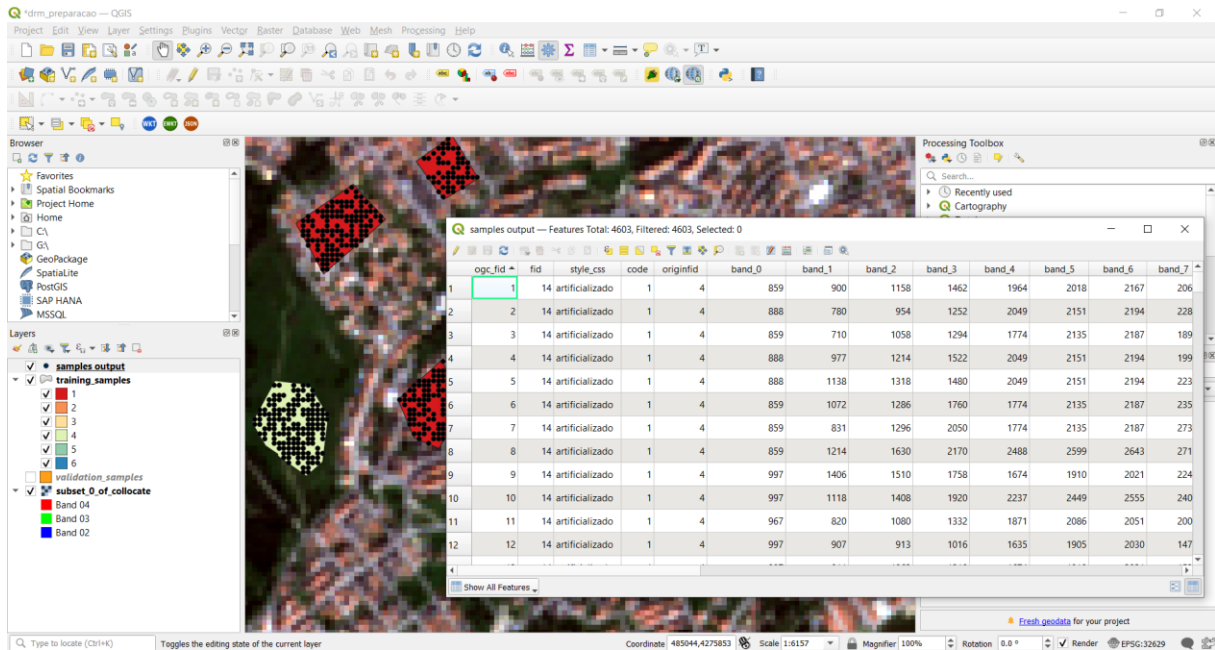
Now that the locations of the samples are selected, we will attach measurements to them. This is the purpose of the **SampleExtraction** application. It will walk through the list of samples and extract the underlying pixel values. If no **-out** parameter is given, the **SampleExtraction** application can work in update mode (updates the samples file – samples.sqlite – created in the previous step), 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.

<sup>19</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_SampleExtraction.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_SampleExtraction.html)



The updated samples file attributes table has as many more columns as the number of bands in the image (in this case, 26 bands):

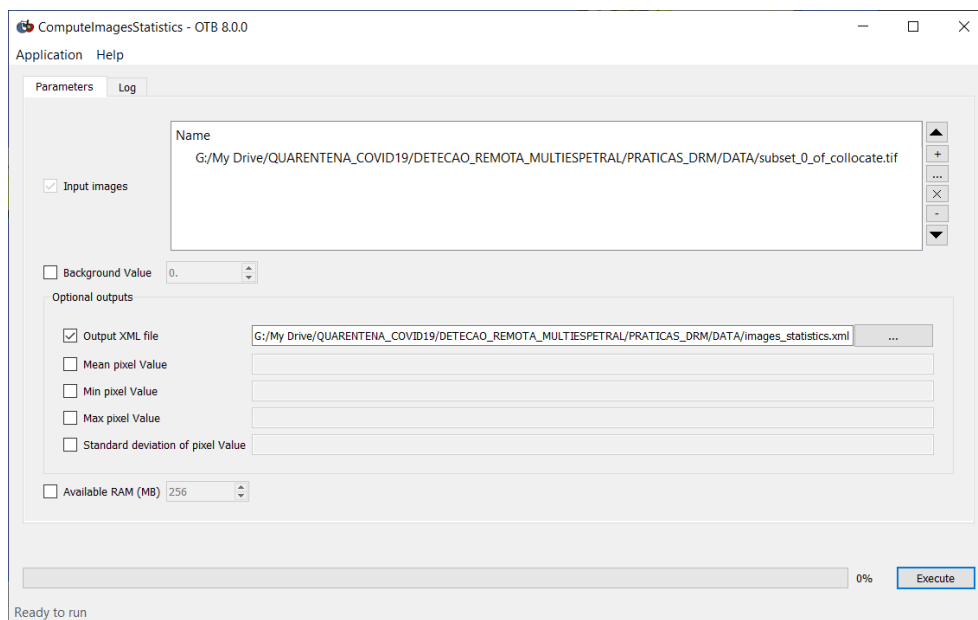


## STEP 4: COMPUTE IMAGES STATISTICS (ComputeImagesStatistics)

In OTB, use **Learning > ComputeImageStatistics**<sup>20</sup> to compute the global mean and standard deviation for each band from a set of images and optionally saves the results in an XML file. Each image of the set must contain the same bands as the others (i.e. same types, in the same order).

The output XML is intended to be used as 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.

Some machine learning algorithms converge faster if the range of features is [-1, 1] or [0, 1]. Others will be sensitive to relative ranges between features, 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 algorithms using euclidean distance at some point to compare features (such as the Support Vector Machines, SVM<sup>21</sup>). 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 the pooled variance of each band for one or several images.



The Output statistics file (here named *images\_statistics.xml*) can then be fed to the training and classification applications (use WordPad to open the file).

<sup>20</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ComputeImagesStatistics.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ComputeImagesStatistics.html)

<sup>21</sup> <https://medium.com/@meritshot/standardization-v-s-normalization-6f93225fbd84>

```

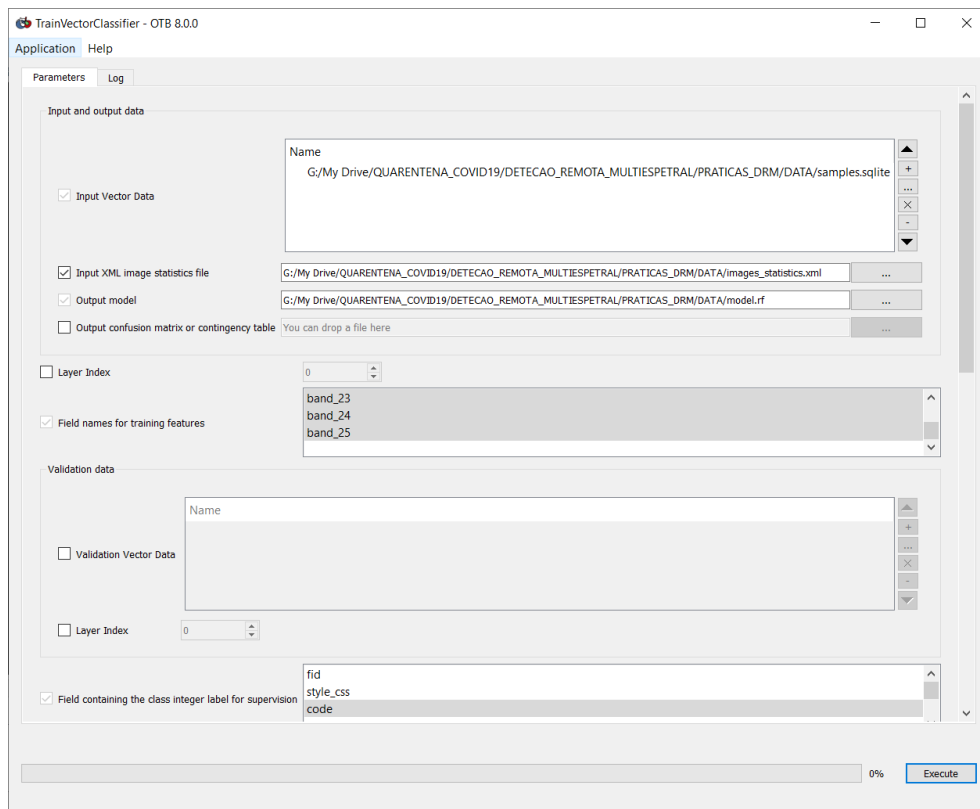
<?xml version="1.0" ?>
<FeatureStatistics>
  <Statistic name="mean">
    <StatisticVector value="641.594" />
    <StatisticVector value="688.629" />
    <StatisticVector value="914.817" />
    <StatisticVector value="889.033" />
    <StatisticVector value="1171.85" />
    <StatisticVector value="1708.58" />
    <StatisticVector value="1863.41" />
    <StatisticVector value="1963.26" />
    <StatisticVector value="1977.88" />
    <StatisticVector value="2023.28" />
    <StatisticVector value="1594.05" />
    <StatisticVector value="1203.18" />
    <StatisticVector value="2.62903" />
    <StatisticVector value="564.895" />
    <StatisticVector value="722.768" />
    <StatisticVector value="946.704" />
    <StatisticVector value="1084.5" />
    <StatisticVector value="1331.47" />
    <StatisticVector value="1660.54" />
    <StatisticVector value="1815.59" />
    <StatisticVector value="1919.3" />
    <StatisticVector value="1957.28" />
    <StatisticVector value="1955.03" />
    <StatisticVector value="2084.22" />
    <StatisticVector value="1632.36" />
    <StatisticVector value="1.95031" />
  </Statistic>
  <Statistic name="min">
    <StatisticVector value="105" />
    <StatisticVector value="19" />
    <StatisticVector value="102" />
    <StatisticVector value="52" />
    <StatisticVector value="1" />
    <StatisticVector value="1" />
    <StatisticVector value="1" />
    <StatisticVector value="1" />
    ...
  <Statistic name="stddev">
    <StatisticVector value="368.275" />
    <StatisticVector value="527.229" />
    <StatisticVector value="618.283" />
    <StatisticVector value="809.745" />
    <StatisticVector value="891.29" />
    <StatisticVector value="1228.62" />
    <StatisticVector value="1345.35" />
    <StatisticVector value="1447.53" />
    <StatisticVector value="1435.57" />
    <StatisticVector value="1396.4" />
    <StatisticVector value="1246.89" />
    <StatisticVector value="1040.09" />
    <StatisticVector value="2.57724" />
    <StatisticVector value="450.878" />
    <StatisticVector value="640.937" />
    <StatisticVector value="787.111" />
    <StatisticVector value="1011.39" />
    <StatisticVector value="1092.76" />
    <StatisticVector value="1263.84" />
    <StatisticVector value="1361.8" />
    <StatisticVector value="1456.61" />
    <StatisticVector value="1462.56" />
    <StatisticVector value="1440.06" />
    <StatisticVector value="1622.09" />
    <StatisticVector value="1349.36" />
    <StatisticVector value="2.75754" />
  </Statistic>
</FeatureStatistics>

```

## STEP 5: TRAIN THE MACHINE LEARNING MODEL (TrainVectorClassifier)

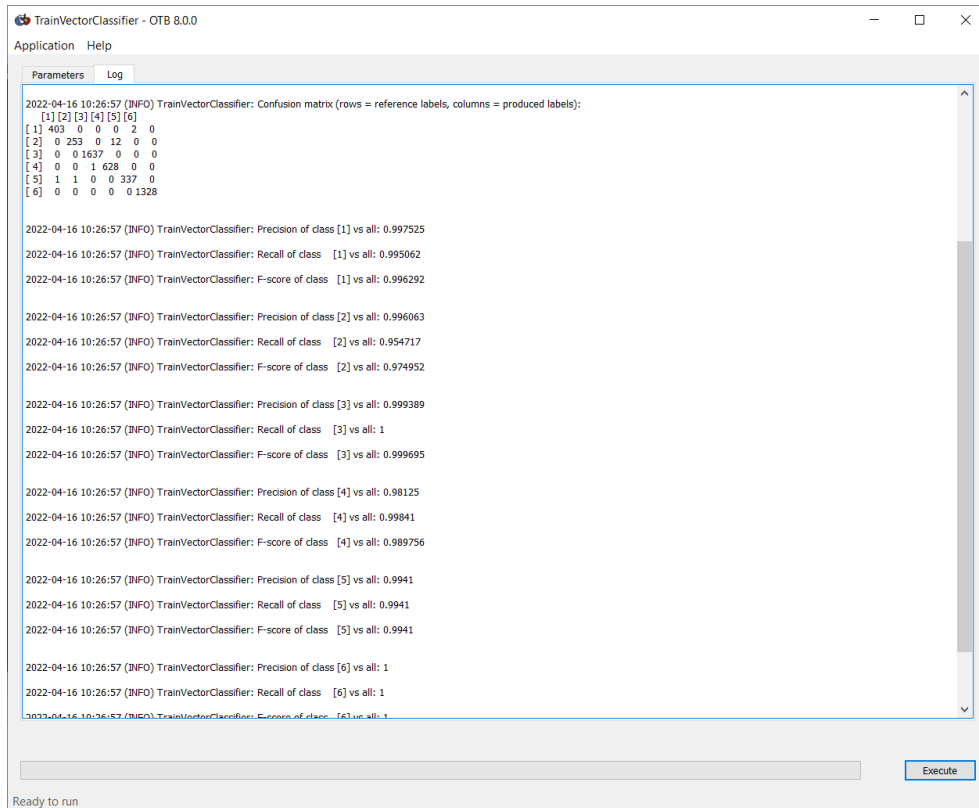
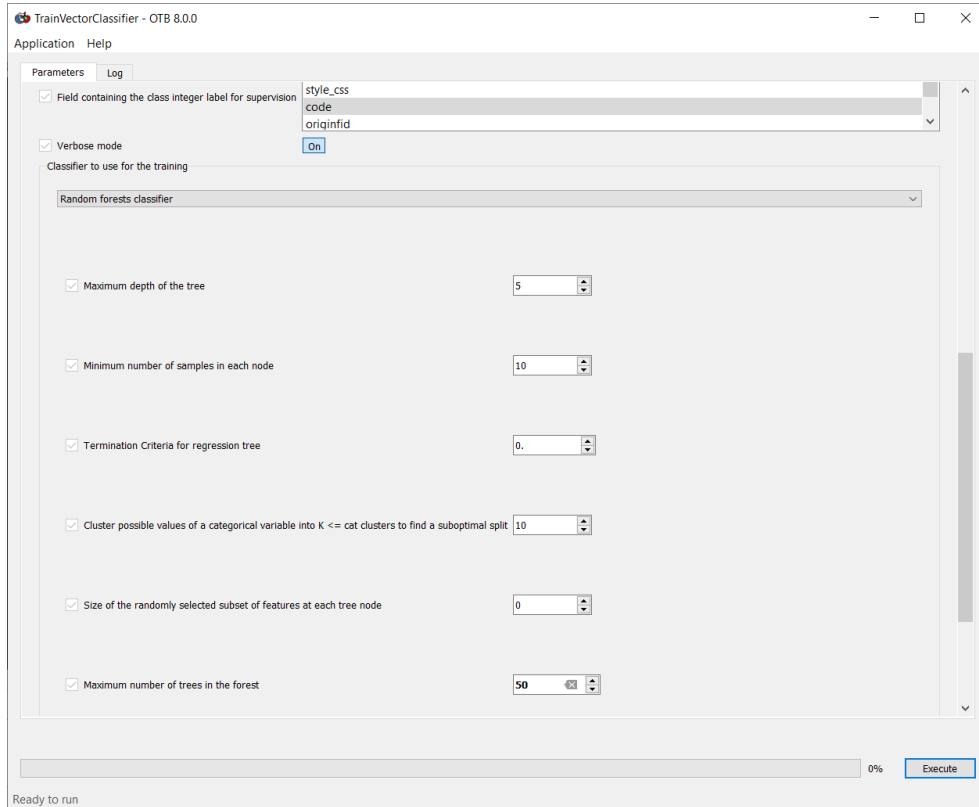
In OTB, use **Learning > TrainingVectorClassifier**<sup>22</sup> to train a classifier based on labeled geometries and a list of features to consider for classification. This application is based on LibSVM, OpenCV Machine Learning (2.3.1 and later), and Shark ML. The output of this application is a text model file, whose format corresponds to the ML model type chosen. There are no image or vector data outputs created.

Now that the training samples are ready, we can perform the learning using the **TrainVectorClassifier** application. In this case, use the **Random Forest algorithm** and change the number of trees from 100 to 50 (see input parameters and corresponding default values below).



...

<sup>22</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_TrainVectorClassifier.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_TrainVectorClassifier.html)



## RANDOM FOREST CLASSIFIER OPTIONS (default values)

(more details in [https://docs.opencv.org/2.4/modules/ml/doc/random\\_trees.html](https://docs.opencv.org/2.4/modules/ml/doc/random_trees.html))

**Maximum depth of the tree** `-classifier.rf.max int` DEFAULT VALUE: 5

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** `-classifier.rf.min int` DEFAULT VALUE: 10

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** `-classifier.rf.ra float` DEFAULT VALUE: 0

If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.

**Cluster possible values of a categorical variable into  $K \leq \text{cat}$  clusters to find a suboptimal split** `-classifier.rf.cat int` DEFAULT VALUE: 10

Cluster possible values of a categorical variable into  $K \leq \text{cat}$  clusters to find a suboptimal split.

**Size of the randomly selected subset of features at each tree node** `-classifier.rf.var int` DEFAULT VALUE: 0

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** `-classifier.rf.nbtrees int` DEFAULT VALUE: 100

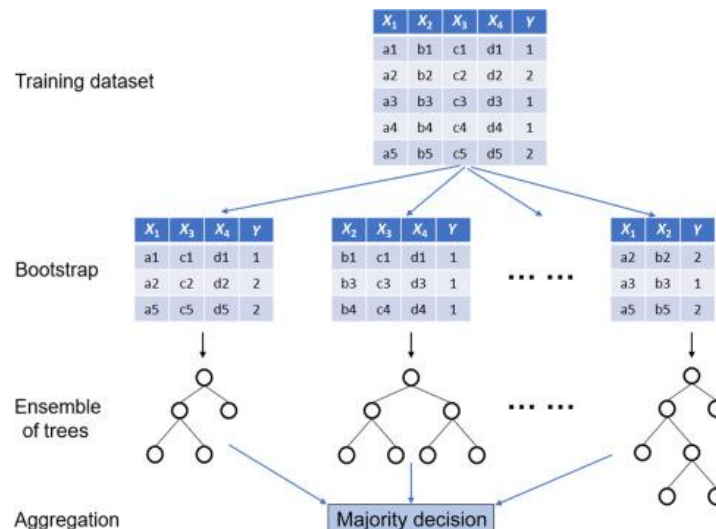
The maximum number of trees in the forest. Typically, the more trees you have, the better the accuracy. However, the improvement in accuracy generally diminishes and reaches an asymptote for a certain number of trees. Also keep in mind, that increasing the number of trees increases the prediction time linearly.

**Sufficient accuracy (OOB error)** `-classifier.rf.acc float` DEFAULT VALUE: 0.01

Sufficient accuracy (OOB error).

<https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710>

<https://www.sciencedirect.com/topics/engineering/random-forest>

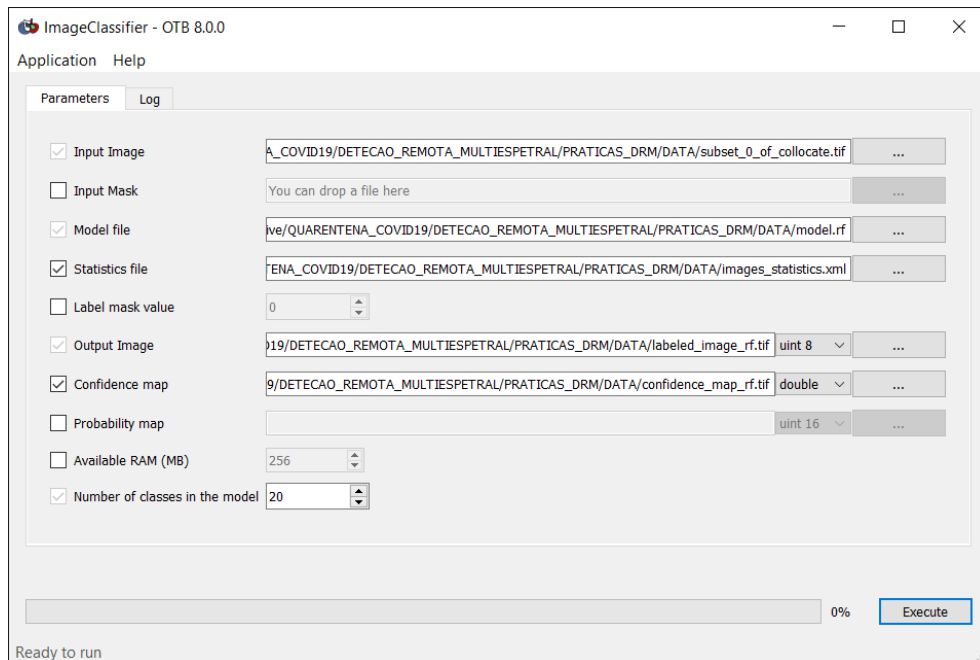


## STEP 6: PERFORM THE CLASSIFICATION BY APPLYING THE MODEL (ImageClassifier)

In OTB, use **Learning > ImageClassifier**<sup>23</sup> to perform a classification of the input image according to a model file. This application performs an image classification based on a model file produced by the **TrainImagesClassifier** or the **TrainVectorClassifier** 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.

Once the classifier has been trained, one can apply the model to classify pixels inside defined classes on a new image using the **ImageClassifier** application. A confidence map of the produced classification might be produced. The confidence index depends on the model: in the case of Random Forests, the confidence is the proportion of votes for the majority class.

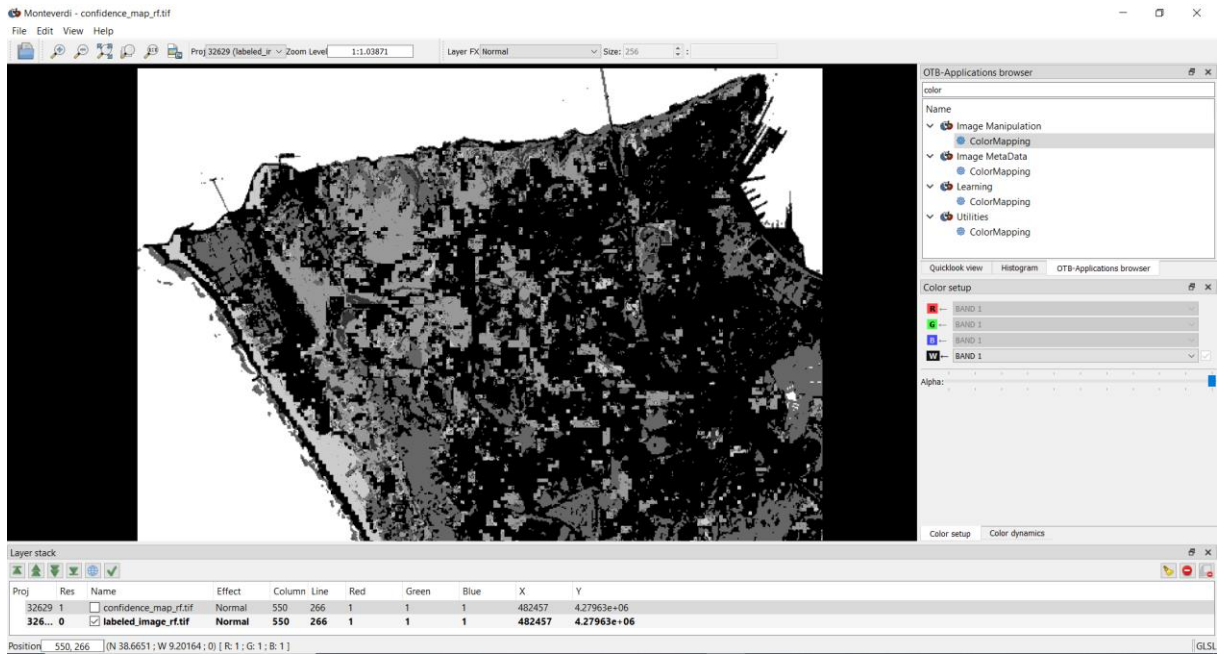
Note that, the probability map is only implemented for the Shark Random Forest classifier at this point.



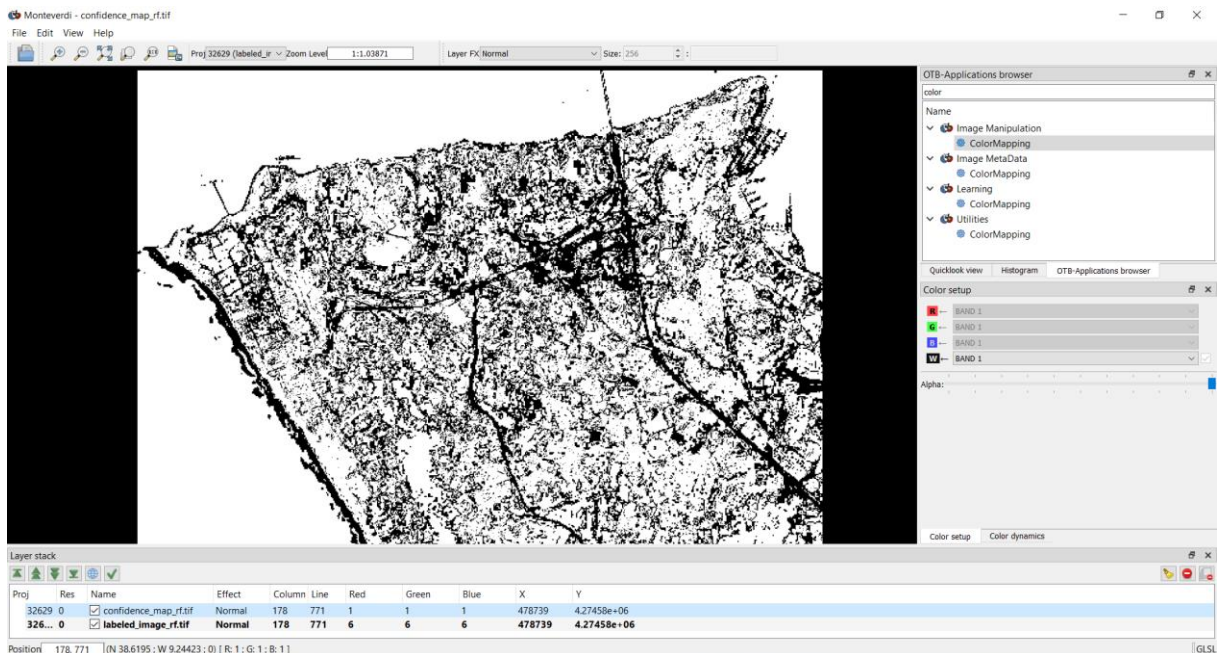
<sup>23</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ImageClassifier.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ImageClassifier.html)



## Labeled Image

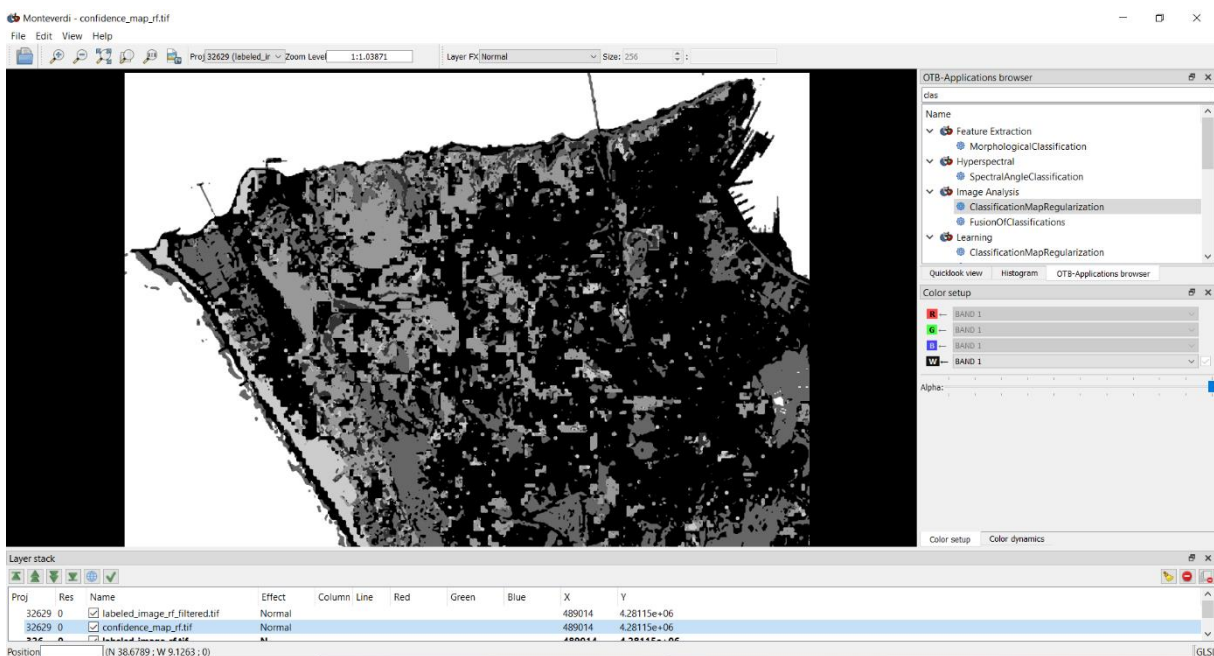
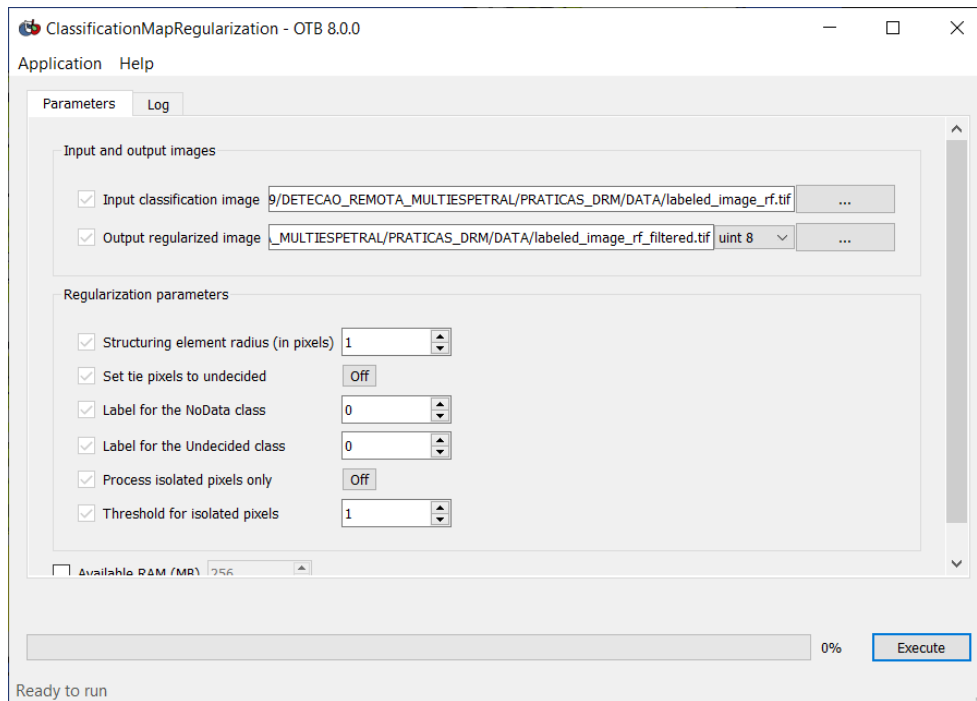


## Confidence map



## STEP 7: FILTER THE LABELED IMAGES (ClassificationMapRegularization)

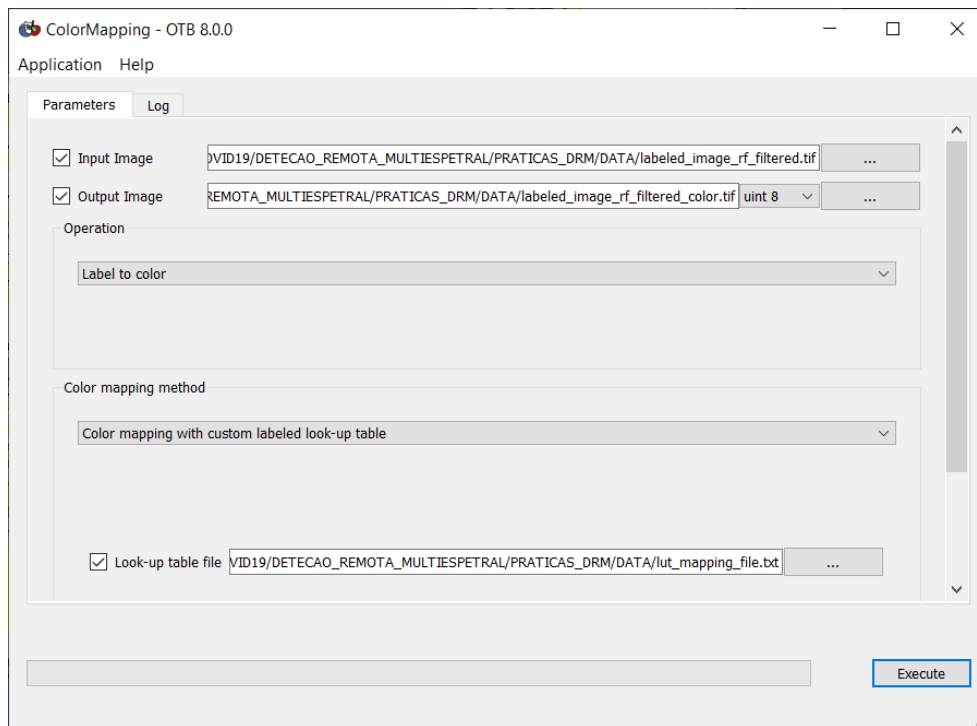
In OTB, use **Learning > ClassificationMapRegularization**<sup>24</sup> to filter the input labeled image (with a maximal class label = 65535) using Majority Voting in a ball-shaped neighborhood. Majority Voting takes the more representative value of all the pixels identified by the ball-shaped structuring element and then sets the center pixel to this majority label value.



<sup>24</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ClassificationMapRegularization.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ClassificationMapRegularization.html)

## STEP 8: MAP THE LABELED IMAGE TO AN 8-BITS IMAGE (ColorMapping)

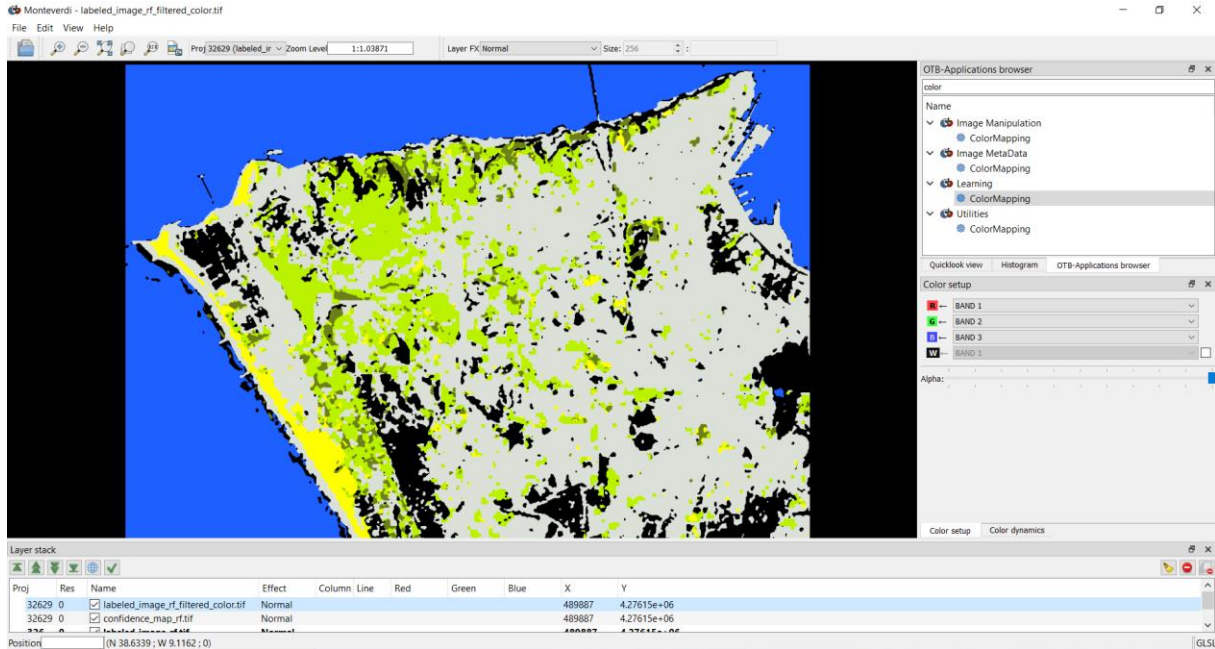
In OTB, use **Learning > ColorMapping**<sup>25</sup> to map a label image to an 8-bits RGB image (both ways) using different methods.



When using ColorMapping with custom labeled look-up table options, a Look-up table file `method.custom.lut filename [dtype]` is required. 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.

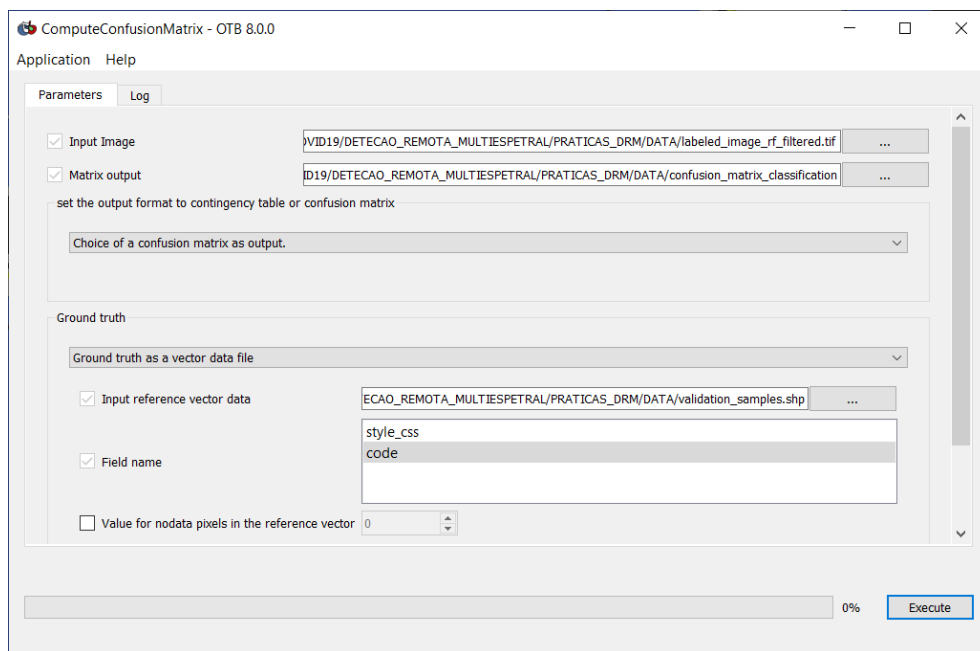
```
# Lines beginning with a # are ignored
1 192 192 192
2 96 169 23
3 0 138 0
4 164 196 0
5 227 200 0
6 27 161 226
```

<sup>25</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ColorMapping.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ColorMapping.html)

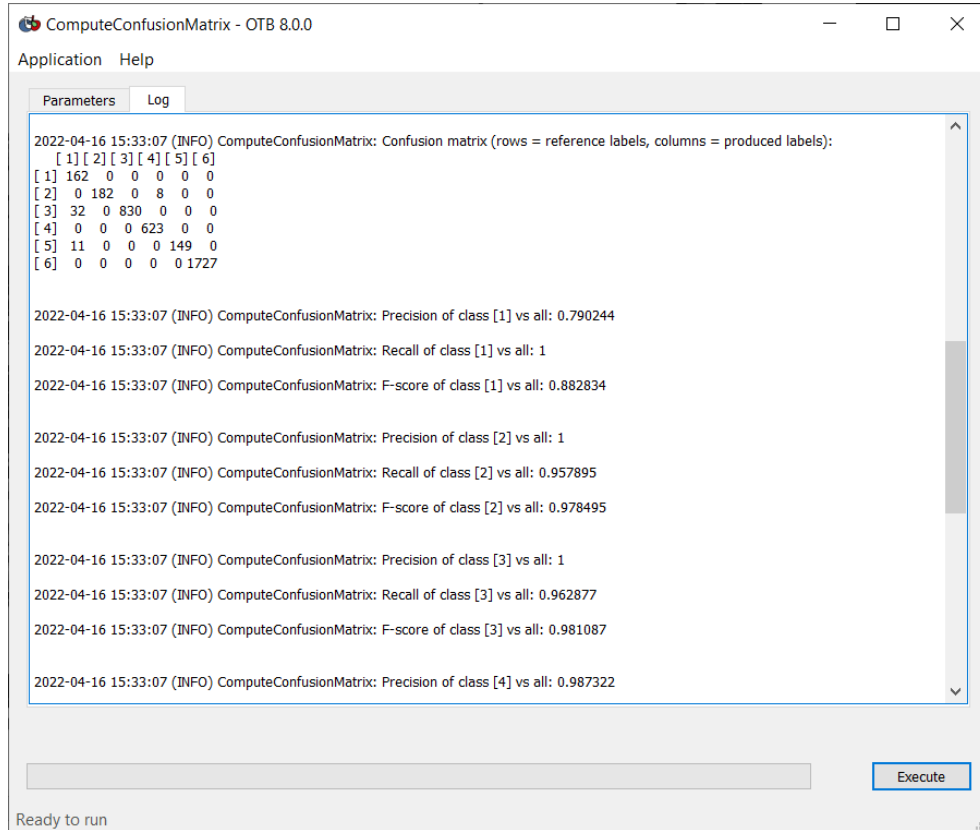


## STEP 9: COMPUTE A CONFUSION MATRIX (ComputeConfusionMatrix)

In OTB, use **Learning > ComputeConfusionMatrix<sup>26</sup>** to compute the confusion matrix for a classification (in this case, a Random Forest classification).



<sup>26</sup> [https://www.orfeo-toolbox.org/CookBook/Applications/app\\_ComputeConfusionMatrix.html](https://www.orfeo-toolbox.org/CookBook/Applications/app_ComputeConfusionMatrix.html)



**Produced Labels**

Reference Data

Classes	1	2	3	4	5	6	Row Total	Recall	F1-score
1	162						162	1	0.88
2		182		8			190	0.96	0.98
3	32		830				862	0.96	0.98
4				623			623	1	0.99
5	11				149		160	0.93	0.96
6						1727	1727	1	1
Column Total	205	182	830	631	149	1727	3724		
Precision	0.79	1	1	0.99	1	1			
<b>Overall Accuracy= 0.99</b> <b>Kappa Coefficient= 0.98</b>									

## FINAL PROJECT

Produce a land use/land cover map using a time series with at least four Sentinel-2 Level-2A images covering an entire year, one for each season. Choose a geographical region inside one of the UTM tiles covering Portugal's mainland (no more than a quarter of the image 50 km x 50 km). The adopted region of interest must have as many different land use/land cover classes as possible. Use the Level 1/Level 2 of the COSsim nomenclature<sup>27</sup> as examples to define the classes that will be represented in your final map.

### Nomenclatura da Carta de Ocupação do Solo Simplificada (COSsim)

A nomenclatura da COSsim é composta por três níveis de detalhe temático crescente, com 6, 9 e 15 classes nos níveis 1, 2 e 3, respetivamente (Tabela 1).

Tabela 1. Nomenclatura da COSsim.

Nível 1	Nível 2	Nível 3
1 – Artificializado	10 – Artificializado	100 – Artificializado
2 – Agricultura	21 – Agricultura	211 – Culturas anuais de outono/inverno*
		212 – Culturas anuais de primavera/verão*
		213 – Outras áreas agrícolas*
3 - Floresta	31 – Folhosas	311 – Sobreiro e Azinheira
		312 – Eucalipto
		313 – Outras folhosas
	32 - Resinosas	321 – Pinheiro bravo
		322 – Pinheiro manso
		323 – Outras resinosas
4 – Matos e vegetação herbácea espontânea	41 – Matos	410 – Matos
	42 – Vegetação herbácea espontânea	420 – Vegetação herbácea espontânea
5 – Superfícies sem vegetação	50 – Superfícies sem vegetação	500 – Superfícies sem vegetação
6 – Água e zonas húmidas	61 – Zonas húmidas	610 – Zonas húmidas
	62 – Água	620 – Água

\* A COSsim2018 não tem estas subclasses da Agricultura.

To help you define the training samples to train the classifier, download the COS2018 from the national spatial data infrastructure (*Sistema Nacional de Informação Geográfica, SNIG*<sup>28</sup>) but keep in mind that the COS nomenclature<sup>29</sup> is much more detailed than the COSsim nomenclature, so you will have to generalize.

The methodology for land use/land cover mapping may include different approaches:

- The combined use of subproducts derived from the original bands, such as spectral indices, principal components, texture, and temporal metrics;
- Different proportions of training/validation samples;
- Different machine learning algorithms: Random Forest, Support Vector Machines, k-Nearest Neighbors, and Artificial Neural Network.

<sup>27</sup> [https://geo2.dgterritorio.gov.pt/atom-dgt/COSsim/Nomenclatura\\_COSsim.pdf](https://geo2.dgterritorio.gov.pt/atom-dgt/COSsim/Nomenclatura_COSsim.pdf)

<sup>28</sup> <https://snig.dgterritorio.gov.pt/>

<sup>29</sup> [https://www.dgterritorio.gov.pt/sites/default/files/documentos-publicos/2019-12-26-11-47-32-0\\_ET-COS-2018\\_v1.pdf](https://www.dgterritorio.gov.pt/sites/default/files/documentos-publicos/2019-12-26-11-47-32-0_ET-COS-2018_v1.pdf)

To write your final report, use a pre-defined template that can be downloaded from: <https://www.mdpi.com/journal/remotesensing/instructions>. The document must have at least 14 pages and no more than 20 pages, including figures and tables. You can add more pages with extra details on your results (annexes).