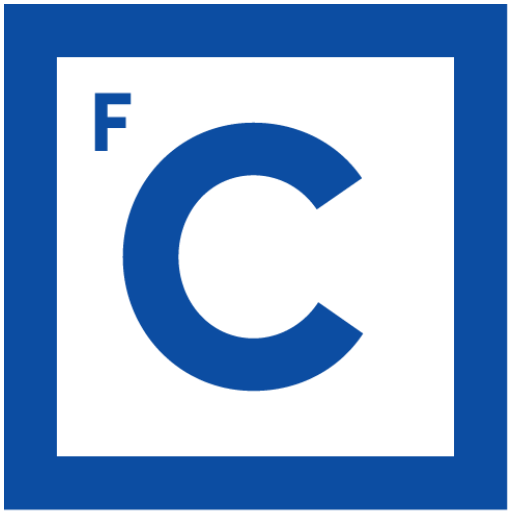


Ciências ULisboa

Faculdade de Ciências da Universidade de Lisboa

DEGGE

MULTISPECTRAL REMOTE SENSING

MASTER IN GEOSPATIAL ENGINEERING / MASTER IN GEOGRAPHIC INFORMATION SYSTEMS-TECHNOLOGIES AND APPLICATIONS

2019/2020

PRACTICAL LESSON 1

The software to be used in class for processing remote sensing images is **Orfeo ToolBox (OTB)** version 7.0.0, which is an open-source project for state-of-the-art remote sensing (https://www.orfeo-toolbox.org/). Built on the shoulders of the open-source geospatial community, it can process high resolution optical, multispectral and radar images at the terabyte scale. A wide variety of applications are available: from ortho-rectification or pansharpening, all the way to classification, SAR processing, and much more!

All of OTB's algorithms are accessible from Monteverdi, QGIS, Python, the command line or C++. Monteverdi is an easy to use visualization tool with an emphasis on hardware accelerated rendering for high resolution imagery (optical and SAR). With it, end-users can visualize huge raw imagery products and access all 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 (https://www.orfeo-toolbox.org/CookBook/)!

https://www.orfeo-toolbox.org/CookBook/Monteverdi.html

EXERCISE 1.1

Identify the main characteristics of the following optical satellite images by accessing their corresponding websites:

- Landsat-8 (https://www.usgs.gov/land-resources/nli/landsat/landsat-8?qt-science_support_page_related_con=0#qt-science_support_page_related_con)
- Sentinel-2 (https://sentinel.esa.int/web/sentinel/missions/sentinel-2)
- SPOT-7 (https://www.intelligence-airbusds.com/optical-and-radar-data/)
- Pleiades-1 (https://www.intelligence-airbusds.com/optical-and-radar-data/)
- WorldView-3 (https://www.digitalglobe.com/company/about-us)

Satellite Launch date/Mission status	Spatial resolution	Spectral resolution	Temporal resolution	Radiometric resolution	Swath Width
Landsat-8 11 February 2013 Operational					
Sentinel-2 2A:23 June 2015 2B: 7 March 2017 Operational					
SPOT-7 30 June 2014 Operational					
Pleiades-1 1A: 17 December 2011 1B: 1 December 2012 Operational					
WorldView-3 13 August 2014 Operational					

Spatial resolution is a measure of the fineness of detail of an image. For digital images, this refers to the ground area captured by a single pixel; because pixels are typically square, resolution is generally expressed as the side length of a pixel. For example, a 30-m pixel (i.e., the spatial resolution of Landsat sensors) stores one digital number per spectral band of information for any landscape feature smaller than 900 m^2, while WorldView-3 provides a spatial resolution of 1 m or less.

Spectral resolution, represented by the width of the wavelength interval and/or number of spectral channels (or bands) captured by a sensor, defines the storage of recorded electromagnetic energy and the sensor's ability to detect wavelength differences between objects or areas of interest. For example, the ETM+ sensor on board NASA's Landsat-7 satellite records data in eight bands, including one panchromatic band. Each of these bands is sensitive to different wavelengths of visible and infrared radiation. The sensor on the Pleiades satellites record data in four spectral bands targeted at the blue, green, red, and near-infrared portions of the electromagnetic spectrum, plus a panchromatic band.

Temporal resolution is the amount of time it takes a sensor to revisit a particular geographic location. Often, temporal resolution, or revisit time, is expressed in terms of days. For example, Landsat-7 has a 16-day orbit cycle, meaning that the satellite (and its ETM+ sensor) returns to a given location on Earth's surface every 16 days. Sentinel-2 has a high revisit time of 10 days at the equator with one satellite and 5 days with 2 satellites.

Radiometric resolution is the sensitivity of a sensor to brightness values (i.e., the smallest differences in intensity that it can detect). This metric is usually articulated in terms of binary bit-depth, which refers to number of grayscale levels at which data are recorded by a particular sensor (Jensen, 2005). The binary bit-depth is typically expressed in the following ranges of grayscale levels: 8-bit (0–255), 10-bit (0–1,023), 11-bit (0–2,047), 12-bit (0–4,095) and 16-bit (0–65,535). The WorldView-2 11-bit collection depth

represents a substantial improvement over the 8-bit resolution typically exhibited by predecessors such as the Landsat sensors.

EXERCISE 1.2

Access OneDrive (https://onedrive.live.com/about/pt-pt/) to download 3 image data sets (Pleiades-1B, Sentinel-2A and Landsat-8) for the same geographic location (northeast of Coruche, Santarém, Portugal).

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

Landsat data products held in the USGS archives can be searched and downloaded at no charge from a variety of sources. This page provides details about which data access portals may work best, based on the data desired. **EarthExplorer** is a Graphical interface used to define areas of interest by address, zip code, place name, or using the map. Queries can be applied to multiple collections simultaneously.

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

https://earthexplorer.usgs.gov/

The **Copernicus Open Access Hub** (previously known as Sentinels Scientific Data Hub) provides complete, free and open access to Sentinel-1, Sentinel-2, Sentinel-3 and Sentinel-5P user products, starting from the In-Orbit Commissioning Review (IOCR).

https://scihub.copernicus.eu/

EXERCISE 1.3

a) Open the Sentinel-2A image, downloaded from OneDrive, using its metadata file (MTD_MSIL2A.xml) inside the unzipped folder (S2A_MSIL2A_20190505T112121_N0211_R037_T29SND

_20190505T122038.SAFE).

File \ Open image(s) … (or Ctrl + O)

b) Generate two color composites (one True Color Composite and a False Color Composite).

When opening a metadata file, OTB places in the Color setup tab the following band's numbering in order to create a True Color Composite:

Red channel: Band 1 (Red, original band 4)

Green channel: Band 2 (Green, original band 3)

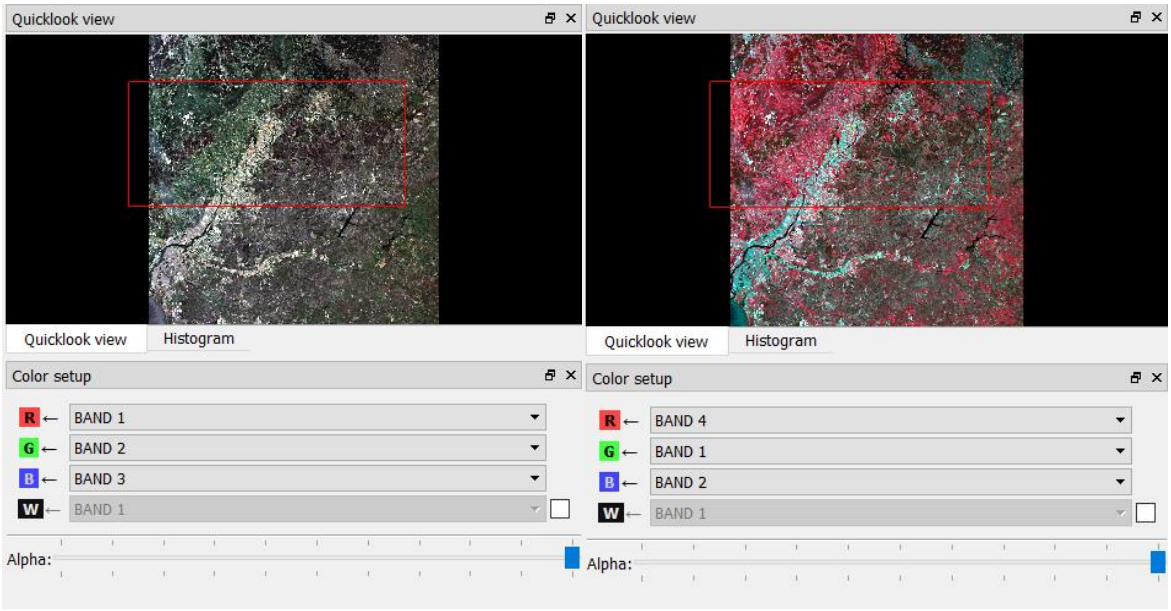
Blue channel: Band 3 (Blue, original band 2)

To create a common False Color Composite (described below) the following band's numbering should be considered:

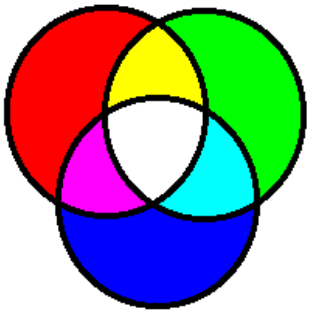
Red channel: Band 4 (NIR, original band 8)

Green channel: Band 1 (Red, original band 4)

Blue channel: Band 2 (Green, original band 3)



In displaying a color composite image, three primary colors (red, green and blue) are used. When these three colors are combined in various proportions, they produce different colors in the visible spectrum. Associating each spectral band (not necessarily a visible band) to a separate primary color results in a color composite image.



True Color Composite: If a multispectral image consists of the three visual primary color bands (red, green, blue), the three bands may be combined to produce a "true color" image (Figure 1.1a). For example, the bands 4 (red band), 3 (green band) and 2 (blue band) of a LANDSAT-8 OLI image can be assigned respectively to the R, G, and B colors for display. In this way, the colors of the resulting color composite image resemble closely what would be observed by the human eyes.

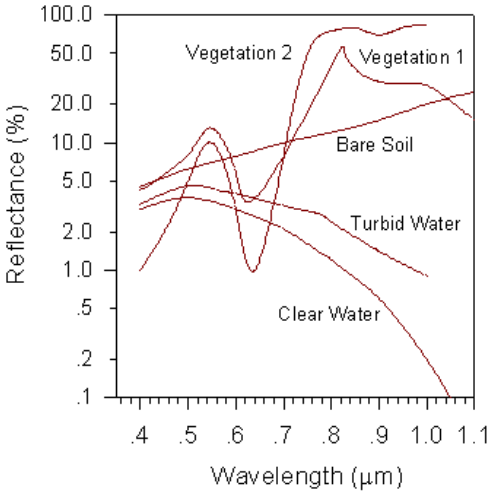
False Color Composite: The display color assignment for any band of a multispectral image can be done in an entirely arbitrary manner. In this case, the color of a target in the displayed image does not have any resemblance to its actual color. The resulting product is known as a false color composite image (Figure 1.1b). There are many possible schemes of producing false color composite images. However, some scheme may be more suitable for detecting certain objects in the image.

A very common false color composite scheme for displaying a multispectral image is shown below:

R = NIR band
G = red band
B = green band

This false color composite scheme allows vegetation to be detected readily in the image. In this type of false color composite images, vegetation appears in different shades of red depending on the types and conditions of the vegetation, since it has a high reflectance in the NIR band (as shown in the graph of spectral reflectance signatures on the right).

Clear water appears dark-bluish (higher green band reflectance), while turbid water appears cyan (higher red reflectance due to sediments) compared to clear water. Bare soils, roads and buildings may appear in various shades of blue, yellow or grey, depending on their composition.



(a)

(b)

Figure 1.1. Color composite image display. True Color Composite (a) and False Color Composite (b).

c) Get information about the image using the ReadImageInfo OTB application.

OTB- Applications Browser >> Image Manipulation >> ReadImageInfo
Get information about the image - Display information about the input image like: image size, origin, spacing, metadata, projections…
https://www.orfeo-toolbox.org/CookBook/Applications/app_ReadImageInfo.html

EXERCISE 1.4

a) Open the Pleiades-1A multispectral (MS) image, downloaded from OneDrive, using its metadata file (DIM_PHR1A_MS_201905051135509_PRJ_4076165101-2.XML) inside the folder (IMG_PHR1A_MS_002) and extract a region of interest (ROI) from the original Pleiades MS image using the **ExtractROI** OTB application.

OTB- Applications Browser >> Image Manipulation >> ExtractROI
Extract a Region Of Interest (ROI) with user parameters. There are four mode of extraction.
https://www.orfeo-toolbox.org/CookBook/Applications/app_ExtractROI.html

Note: Use **Extract mode: Standard**, considering the following values for Start X: 2480; Start Y: 1350; Size X: 2500, and Size Y: 2300 (it generates a new image with 2500 columns e 2300 rows).

b) Extract the same region of interest (ROI) from the original Pleiades panchromatic (P) image using again the **ExtractROI** OTB application.

Note: This time use **Extraction mode: Fit**, considering as **Reference image** the image extracted in the previous step.

EXERCISE 1.5

Open the Landsat-8 image downloaded from OneDrive, loading bands B2 to B7 (TIF files) separately[1] and then concatenate these 6 images of the same size into a single multi-channel image using the **ConcatenateImages** OTB application.

[1] *_MTL.txt file cannot be read by OTB (Error message: Probably unsupported format or incorrect filename extension).

OTB- Applications Browser >> Concatenation >> ConcatenateImages
Concatenate a list of images of the same size into a single multi-channel image.
https://www.orfeo-toolbox.org/CookBook/Applications/app_ConcatenateImages.html

Alternatively, SNAP[2] might be used to perform the same task. SNAP is able to recognise the *_MTL.txt file:

File \ Open Product …

Then, in order to create a new image only with bands B2 to B7.

Raster \ Subset …

In the Specify Product Subset window choose Band Subset and select only the bands that you are interested in.

To save this image as a TIFF file.

File \ Export \ GeoTIFF

EXERCISE 1.6

Enhance the contrast of the previous image using the **ContrastEnhancement** OTB application.

OTB- Applications Browser >> Image Filtering >> ContrastEnhancement
Enhance contrast in an image or to reduce the dynamic of the image without losing too much contrast. It offers several options as a nodata value, a contrast limitation factor, a local version of the algorithm and also a mode to equalize the luminance of the image.
https://www.orfeo-toolbox.org/CookBook/Applications/app_ContrastEnhancement.html https://www.orfeo-toolbox.org/CookBook/recipes/contrast_enhancement.html

[2] The **Sentinel Application Platform (SNAP)** is an open source common architecture for ESA Toolboxes ideal for the exploitation of Earth Observation data. SNAP reunites all Sentinel Toolboxes in order to offer the most complex platform for this mission. The basic functions include: opening a product, exploring the product components such as bands, masks and tie point grids. Navigation tools and pixel information functionality also represents some of the basic capabilities (https://step.esa.int/main/download/snap-download/).

After, compare the minimum and maximum pixel values in the original image and the corresponding values in the enhanced image using the Color dynamics tab on the right-side dock of the screen.

PRACTICAL LESSON 2

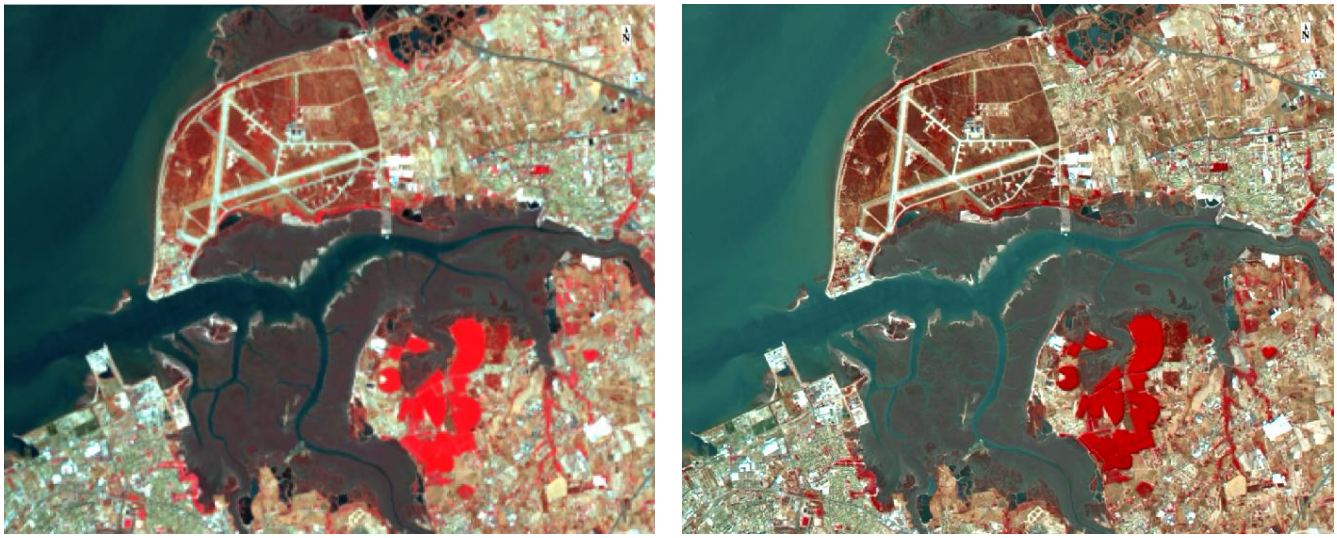
EXERCISE 2.1

Open the Pleiades-1A images, downloaded from OneDrive. The Pleiades-1A satellite features four multispectral bands (blue, green, red, and NIR), with a spatial resolution of 2 m, and a panchromatic band, with a spatial resolution of 0.5 m.

	Bands	Wavelength (micrometers)	Resolution (meters)
Pleiades-1A **HiRI (High Resolution Optical Imager)** **Launched December 17, 2011**	Band 1 - Blue	0.43 - 0.55	2
	Band 2 - Green	0.49 - 0.61	2
	Band 3 - Red	0.60 - 0.72	2
	Band 4 - Near Infrared (NIR)	0.75 - 0.95	2
	Pan	0.48 - 0.83	0.5

Like most high-resolution satellites, Pleiades panchromatic and multispectral data provide the opportunity to create multispectral pan-sharpened images. Although the user can purchase Pleiades pan-sharpened (PMS) product directly, in some cases the user may want to apply their own pan-sharpening.

Pan Sharpening is an image fusion method in which high-resolution panchromatic data is fused with lower resolution multispectral data to create a colorized high-resolution dataset (Figure 2.1). The resulting product should only serve as an aid to literal analysis and not for further spectral analysis.



(a) (b)

Figure 2.1. Pansharpening. Before Pansharp (a) and After Pansharp (b).

Perform the Pleiades 1-A image P+XS pansharpening using the **BundleToPerfectSensor** OTB application.

OTB- Applications Browser >> Geometry >> BundleToPerfectSensor
Performs P+XS pansharpening. The default mode use Pan and XS sensor models to estimate the transformation to superimpose XS over Pan before the fusion ("default mode"). The application provides also a PHR mode for Pleiades images which does not use sensor models as Pan and XS products are already coregistered but only estimate an affine transformation to superimpose XS over the Pan. Note that this option is automatically activated in case Pleiades images are detected as input.
https://www.orfeo-toolbox.org/CookBook/Applications/app_BundleToPerfectSensor.html **https://www.orfeo-toolbox.org/CookBook/recipes/optpreproc.html** **https://www.orfeo-toolbox.org/CookBook/C++/UserGuide.html?highlight=bundle**

Pan Sharpening is an image fusion method in which high-resolution panchromatic data is fused with lower resolution multispectral data to create a colorized high-resolution dataset. The resulting product should only serve as an aid to literal analysis and not for further spectral analysis.

There are three algorithms available in the applications are: RCS (simple Relative Component Substitution), BDF (Bayesian data fusion) and LMVM (Local Mean and Variance Matching).

RCS[3] method is a simple way to merge the images that considers that at the same resolution, the panchromatic image is the sum of the channels of the multispectral image. Once the images have been resampled at the same resolution, the fusion can be carried out: the idea is to apply a low-pass filter to the panchromatic channel image to give it a spectral content (in the Fourier domain) close to the multispectral image (Figure 2.2). Then, the multispectral image is normalised by the filtered panchromatic image and multiplied with the original panchromatic image. The only parameter is the low-pass filter radius.

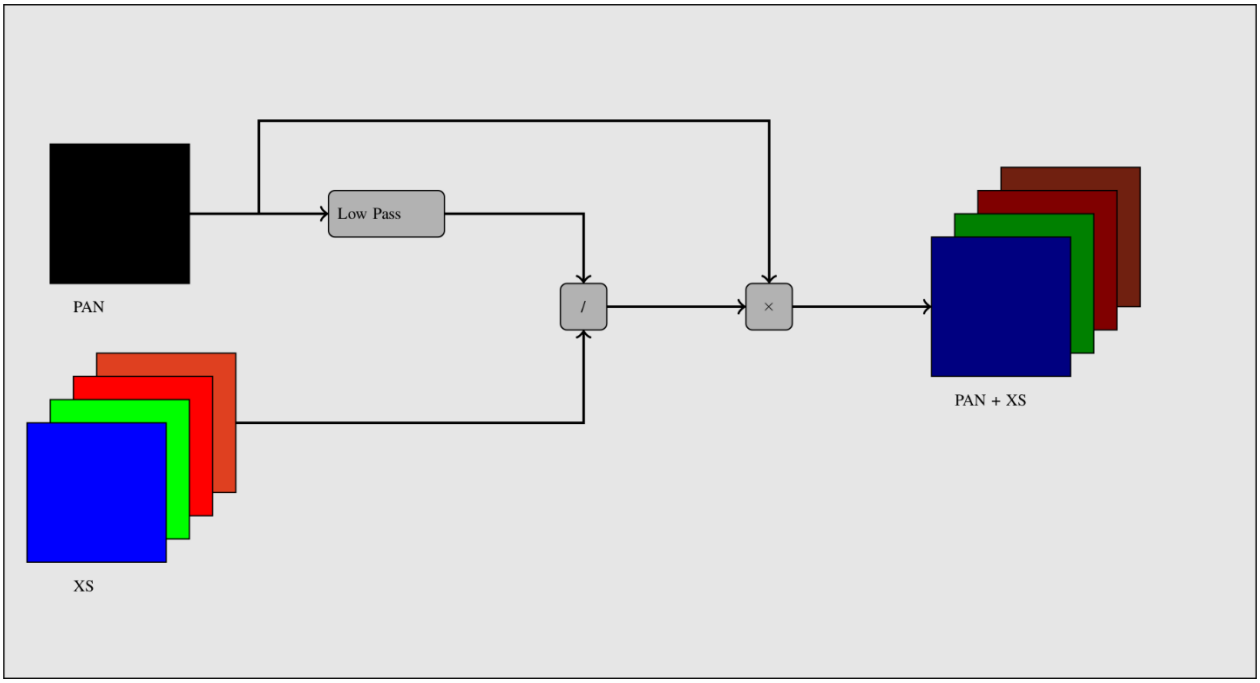


Figure 2.2. Simple pan-sharpening procedure.

BDF[3] method is based on a statistical relationship between the spectral bands and the panchromatic channel. It is better to process the region of interest, rather than the whole image. This method enables spatial information weighting with respect to the spectral information, making it possible to adapt its use according to the need, for example, giving more weight to the information in the panchromatic channel.

LMVM[4] method is a local intensity matching filter that adjusts both local means and variances. The general LMVM algorithm integrates two images, a high-resolution image into a low-resolution channel resampled to the same size. This algorithm will produce a simulated high spatial resolution image pertaining the spectral characteristics of the low-resolution channel. How well the spectral values are preserved will depend on the size of the filtering window. Small window sizes produce the least distortion. Larger filtering

[3] https://books.google.pt/books?id=vppNDwAAQBAJ&pg=PA216&lpg=PA216&dq=rcs+component+substitution+pansharp&source=bl&ots=4nhKBWnSNV&sig=ACfU3U0EKXps1w4w7s5uJ0tM2cJUtMg5GA&hl=pt-PT&sa=X&ved=2ahUKEwiVsteQjPznAhWFPOwKHaFUB7QQ6AEwEHoECAsQAQ#v=onepage&q=rcs%20component%20substitution%20pansharp&f=false

[4] https://www.researchgate.net/publication/2800867_Adaptive_Intensity_Matching_Filters_A_New_Tool_For_Multi-Resolution_Data_Fusion

windows incorporate more structural information from the high resolution image, but with more distortion of the spectral values.

EXERCISE 2.2

Bio-physical applications require conversion of raw digital count numbers (DN) to physical units. Current physical units include radiance[5] (at-Ground or at-Top Of Atmosphere), reflectance[6], or albedo (Figure 2.3).

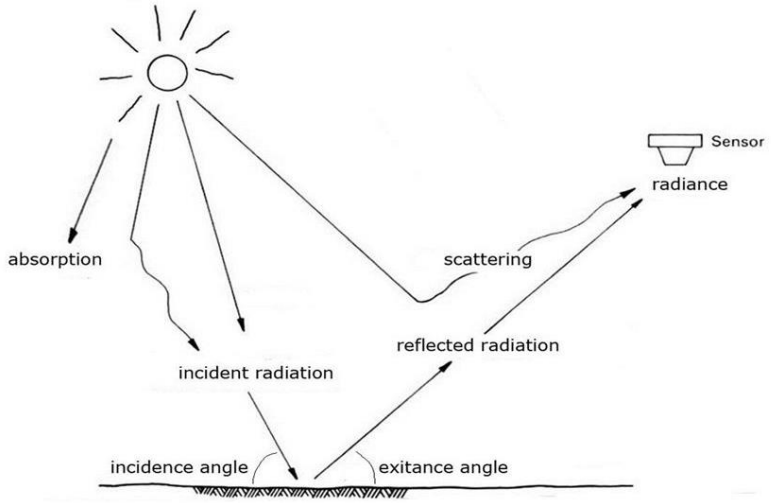


Figure 2.3. Atmospheric effects that influence the measurement of reflected energy by remote sensors.

Convert Pleiades MS pixel values from DN (for Digital Numbers) to surface reflectance (Top Of Canopy) using the **OpticalCalibration** OTB application.

OTB- Applications Browser >> Calibration >> OpticalCalibration
Convert pixel values from DN (for Digital Numbers) to reflectance. Calibrated values are called surface reflectivity and its values lie in the range [0, 1]. The first level is called Top Of Atmosphere (TOA) reflectivity. It takes into account the sensor gain, sensor spectral response and the solar illuminations. The second level is called Top Of Canopy (TOC) reflectivity. In addition to sensor gain and solar illuminations, it takes into account the optical thickness of the atmosphere, the

[5] Radiance is the radiant flux emitted, reflected, transmitted or received by a given surface, per unit solid angle per unit projected area. Radiance is the variable directly measured by remote sensing instruments. Radiance includes radiation reflected from the surface, bounced in from neighbouring pixels, and reflected from clouds above the area of the pixel. Atmosphere also absorb light, which will also decrease the observed radiance. Radiance most often has units of W m^{-2} sr^{-1}. Spectral radiance is the radiance of a surface per unit wavelength (W m^{-2} sr^{-1} μm^{-1}).

[6] Reflectance is the ratio of the amount of light leaving a target to the amount of light striking the target. Reflectance has no units. Top-of-atmosphere reflectance (or TOA reflectance) is the reflectance measured by a space-based sensor flying higher than the earth's atmosphere. These reflectance values will include contributions from clouds and atmospheric aerosols and gases.

atmospheric pressure, the water vapour amount, the ozone amount, as well as the composition and amount of aerosol gasses.
https://www.orfeo-toolbox.org/CookBook/Applications/app_OpticalCalibration.html

For this purpose, the following information (that can be found in the metadata file) is required:

- minute, hour, day, month and year of acquisition (<TIME> in the DIM*.xml file);
- sun elevation angle (<SUN_ELEVATION> in the DIM*.xml file);
- sun azimuth angle (<SUN_AZIMUTH> in the DIM*.xml file);
- viewing elevation angle[7] (calculated using <INCIDENCE_ANGLE> in the DIM*.xml file);
- viewing azimuth angle[8] (calculated using <AZIMUTH_ANGLE>, <INCIDENCE_ANGLE_ACROSS> and <INCIDENCE_ANGLE_ALONG> in the DIM*.xml file);
- gains and biases, one pair of values for each band (passed by a file) (<GAIN> and <BIAS> in the DIM*.xml file);
- solar illuminations, one value for each band (passed by a file) (<VALUE> followed by </Band_Solar_Irradiance> in the DIM*.xml file).

To generate both the Gain and biases and the Solar illumination files, use the following link:

https://www.orfeo-toolbox.org/CookBook/Applications/app_OpticalCalibration.html

Below, we give two examples of txt files containing information about gains/biases and solar illuminations :

- gainbias.txt :

Gain values for each band. Each value must be separated with colons (:), with eventual spaces. Blank lines not allowed. 10.4416 : 9.529 : 8.5175 : 14.0063 # Bias values for each band. 0.0 : 0.0 : 0.0 : 0.0

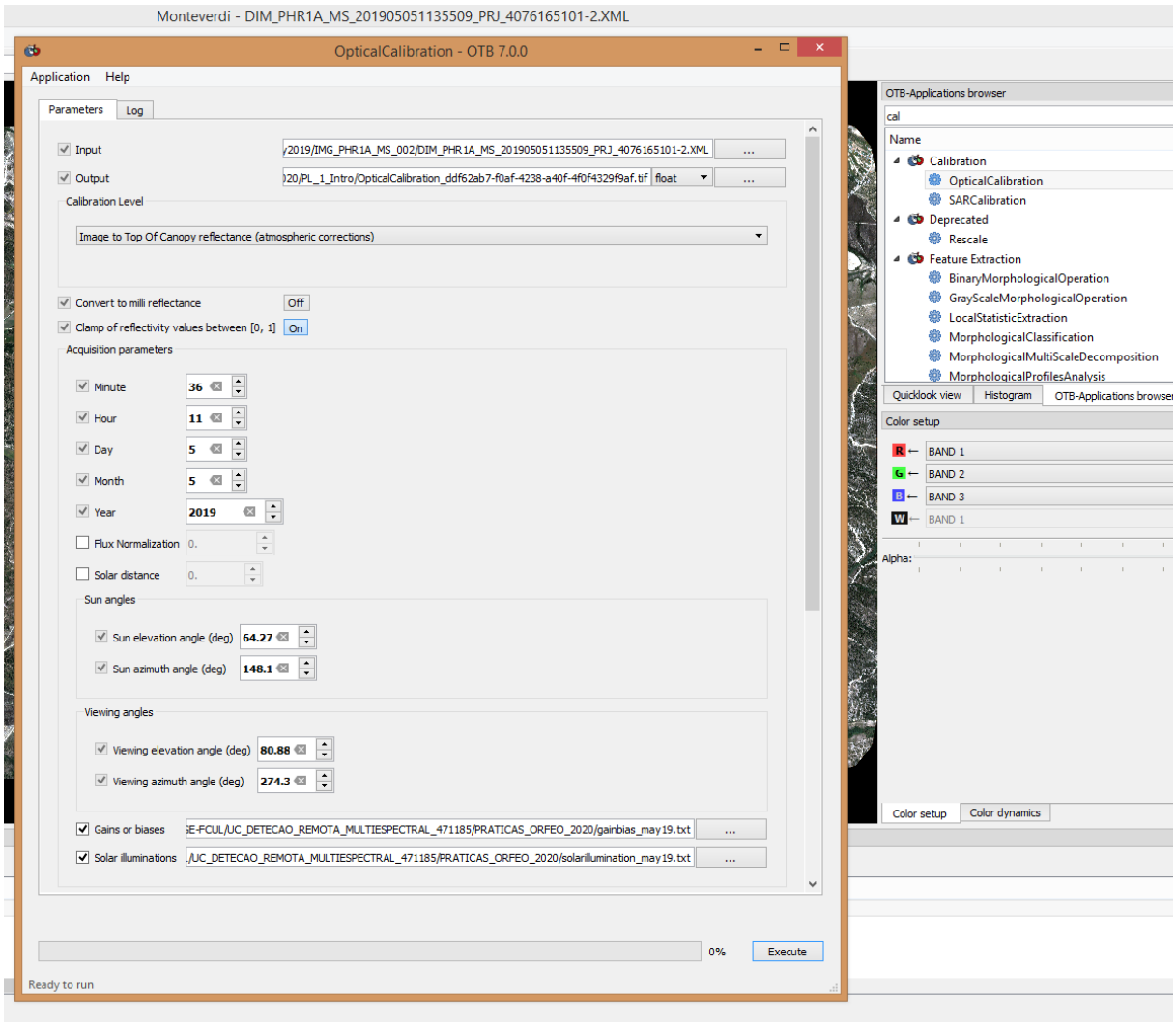
- solarillumination.txt :

Solar illumination values in watt/m2/micron ('micron' means actually 'for each band'). # Each value must be separated with colons (:), with eventual spaces. Blank lines not allowed. 1540.494123 : 1826.087443 : 1982.671954 : 1094.747446

[7] http://www.engesat.com.br/wp-content/uploads/PleiadesUserGuide-17062019.pdf (pages 85-86)

[8] http://www.engesat.com.br/wp-content/uploads/PleiadesUserGuide-17062019.pdf (pages 85-86)

PLEIADES		may/2019*
TIME	**Minute**	**36**
	Hour	**11**
	Day	**5**
	Month	**5**
	Year	**2019**
	SUN_ELEVATION	**64.27903349**
	SUN_AZIMUTH	**148.1768756**
	INCIDENCE_ANGLE	9.1140345
required for TOC in OTB	**Viewing Elevation Angle**	**80.8860**
	AZIMUTH_ANGLE (Azi)	179.938025
	INCIDENCE_ANGLE_ACROSS(Bx)	-9.088054
	INCIDENCE_ANGLE_ALONG (By)	-0.699363
required for TOC in OTB	**Viewing Azimuth Angle (Azsat)**	**274.3019**
	***all values are for center**	
	Gain/bias File	
	B1(Blue)	8.96 / 0
	B2 (Green)	9.02 / 0
	B3 (Red)	10.11 / 0
	B4 (NIR)	15.31 / 0
Band_Solar_Irradiance	**Solar illuminations File**	
	B1(Blue)	1915
	B2 (Green)	1831
	B3 (Red)	1594
	B4 (NIR)	1060


Monteverdi - DIM_PHR1A_MS_201905051135509_PRJ_4076165101-2.XML
OpticalCalibration - OTB 7.0.0
Application
Help
Parameters
Log
Input
Output
Calibration Level
Image to Top Of Canopy reflectance (atmospheric corrections)
Convert to milli reflectance
Off
Clamp of reflectivity values between [0, 1]
On
Acquisition parameters
Minute
36
Hour
11
Day
5
Month
5
Year
2019
Flux Normalization
Solar distance
Sun angles
Sun elevation angle (deg)
64.27
Sun azimuth angle (deg)
148.1
Viewing angles
Viewing elevation angle (deg)
80.88
Viewing azimuth angle (deg)
274.3
Gains or biases
Solar illuminations
0%
Execute
Ready to run
OTB-Applications browser
Name
Calibration
OpticalCalibration
SARCalibration
Deprecated
Rescale
Feature Extraction
BinaryMorphologicalOperation
GrayScaleMorphologicalOperation
LocalStatisticExtraction
MorphologicalClassification
MorphologicalMultiScaleDecomposition
MorphologicalProfilesAnalysis
Quicklook view
Histogram
Color setup
BAND 1
BAND 2
BAND 3
Alpha:
Color setup
Color dynamics

PRACTICAL LESSON 3

EXERCISE 3.1

Open the Sentinel-2 image downloaded from OneDrive, using SNAP (open the image using its metadata file – MTD_MSIL2A.xml).

File \ Open Product …

	Bands	Central Wavelength (micrometers)	Resolution (meters)
Sentinel-2 **2A launched** **June 23, 2015** **2B launched** **March 7, 2017**	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 Vapour	0.945	60
	Band 10 – SWIR (Cirrus)	1.375	60
	Band 11 - SWIR	1.610	20
	Band 12 - SWIR	2.190	20

Then, in order to create a new image with bands 2-8 and bands 11-12, and since you are using images with different spatial resolutions, first you must resample all the images to 10 m of spatial resolution.

Raster \ Geometric Operations \ Resampling

In the Resampling window choose Defining Resampling Parameters and define a band with 10 m of spatial resolution and the resampling algorithm as Nearest (Nearest neighbour).

There are 3 resampling algorithms available: Nearest, Bilinear and Bicubic (Figure 3.1).

- **Nearest Neighbor**: Uses the nearest pixel without any interpolation to create the warped image.
- **Bilinear**: Performs a linear interpolation using four pixels to resample the warped image.
- **Cubic Convolution**: Uses 16 pixels to approximate the sinc[9] function using cubic polynomials to resample the image. Cubic convolution resampling is significantly slower than the other methods.

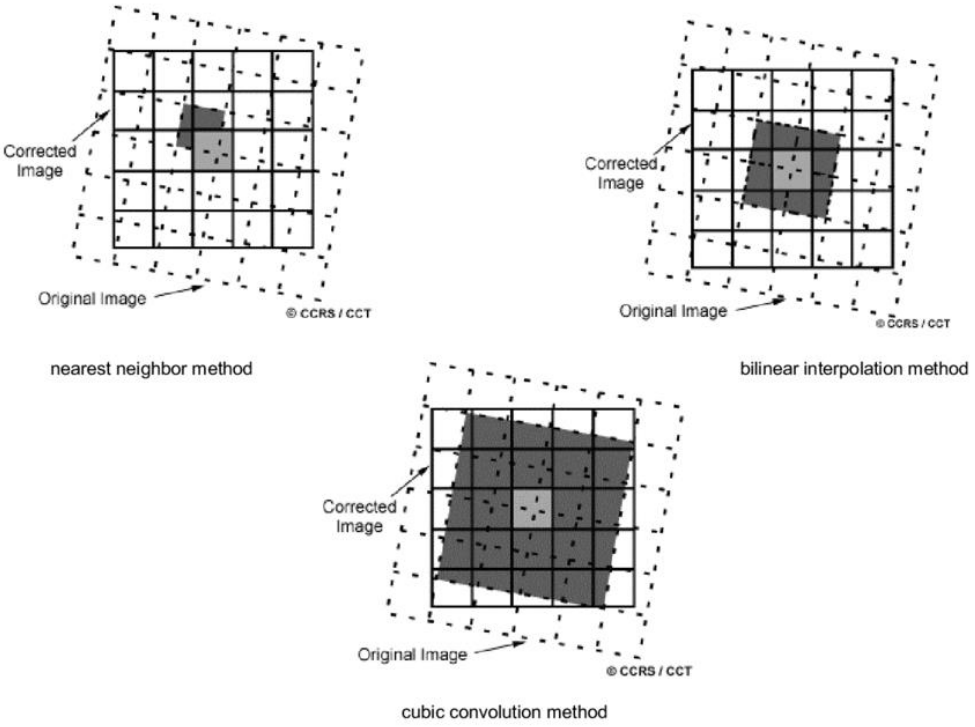


Figure 3.1. Resampling algorithms: Nearest neighbour, Bilinear interpolation and Cubic convolution.

After having all the bands with the same spatial resolution, you can subset the original image considering only Bands 2 to 8 and Bands 11 and 12. Besides, consider a ROI corresponding to the lower left corner of the image in order to reduce its size.

Raster \ Subset …

In the Specify Product Subset window choose first Spatial subset to reduce the image and then Band Subset to select only the bands that you are interested in.

[9] The sinc function, also called the "sampling function," is a function that arises frequently in signal processing and the theory of Fourier transforms. The full name of the function is "sine cardinal," but it is commonly referred to by its abbreviation, "sinc." https://mathworld.wolfram.com/SincFunction.html

To save this image as a TIFF file.

File \ Export \ GeoTIFF

EXERCISE 3.2

Sentinel-2 images can be downloaded as Level-1C and Level-2A products.

Level-1C products provide Top-Of-Atmosphere (TOA) normalized reflectance for each spectral band, coded as integers on 15 bits. The physical values range from 1 (minimum reflectance 10^{-4}) to 10000 (reflectance 1), but values higher than 1 can be observed in some cases due to specific angular reflectivity effects[10]. The value 0 is reserved for "No Data".

Level-2A products provide Bottom-Of-Atmosphere (BOA) reflectance generated with Sen2Cor processor whose main purpose is to correct single-date Sentinel-2 Level-1C products from the effects of the atmosphere.

The surface reflectance values are coded in JPEG2000 with the same quantification value of 10000 as for Level-1C products, i.e. a factor of 1/10000 needs to be applied to Level-2A digital numbers (DN) to retrieve physical surface reflectance values.

a) Apply a factor of 1/10000 to Level-2A original reflectance values to obtain the corresponding physical surface reflectance values using the **BandMathX** OTB application.

OTB- Applications Browser >> Image Manipulation >> BandMathX
Performs a mathematical operation on several multi-band images and outputs the result into an image (multi- or mono-band, as opposed to the BandMath OTB-application). The mathematical formula is done by the muParserX library. Separating expressions by semicolons (;) will concatenate their results into a unique multiband output image.
https://www.orfeo-toolbox.org/CookBook/Applications/app_BandMathX.html

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

im1b1/10000; im1b2/10000; …; im1b9/10000

[10] http://step.esa.int/thirdparties/sen2cor/2.5.5/docs/S2-PDGS-MPC-L2A-PDD-V2.5.5.pdf (page 36)

where **im** identifies the image, that in this case is unique, and **b** identifies the band, that is this case goes from 1 up to 9.

b) Calculate the vegetation indices (VIs) listed in the following table using the image converted to physical surface reflectance values (exercise 3.2a) and the **RadiometricIndices** OTB application.

OTB- Applications Browser >> FeatureExtraction >> RadiometricIndices
Computes radiometric indices using the relevant channels of the input image. The output is a multi-band image into which each channel is one of the selected indices.
https://www.orfeo-toolbox.org/CookBook/Applications/app_RadiometricIndices.html

Vegetation indices (VIs) obtained from remote sensing data are quite simple and effective algorithms for quantitative and qualitative evaluations of vegetation cover, vigor, and growth dynamics, among other applications. These indices have been widely implemented within Remote Sensing (RS) applications using different airborne and satellite platforms with recent advances using Unmanned Aerial Vehicles (UAV).

Vegetation Index	Equation	Reference
NDVI Normalized Difference Vegetation Index	$NDVI = \frac{(NIR - RED)}{(NIR + RED)}$	Rouse *et al.*, 1973[11]
SAVI Soil Adjusted Vegetation Index	$SAVI = \frac{1.5 \cdot (NIR - RED)}{(NIR + RED + 0.5)}$	Huete, 1988[12]
NDWI Normalized Difference Water Index	$NDWI = \frac{(NIR - SWIR^{*})}{(NIR + SWIR)}$ (*) a SWIR band between 1.55-1.75 µm.	Gao, 1996[13]

[11] Rouse, J. W.; Haas, R. H.; Schell, J. A.; Deeering, D. Monitoring Vegetation Systems in the Great Plains with ERTS (Earth Resources Technology Satellite). In Third Earth Resources Technology Satellite-1 Symposium; Freden, S. C., Mercanti, E. P., Becker, M. A., Eds.; Nasa: Washington, 1973; pp 309–317.

[12] Huete, A. R. A Soil-Adjusted Vegetation Index (SAVI). Remote Sens. Environ., 1988. https://doi.org/10.1016/0034-4257(88)90106-X.

[13] Gao, B. C. NDWI - A Normalized Difference Water Index for Remote Sensing of Vegetation Liquid Water from Space. Remote Sens. Environ., 1996. https://doi.org/10.1016/S0034-4257(96)00067-3.

CI Red-edge Chlorophyll Index	$CI = \frac{(\mathrm{Red}Edge6)}{(\mathrm{Red}Edge5)}$	Zarco-Tejada *et al.*, 2001[14]

Before computing spectral indices, raw pixel values (also called digital numbers or DN values) must be calibrated into physically meaningful units. The three most common radiometric corrections are radiance, top-of-atmosphere (TOA) reflectance and apparent surface reflectance. Some literature suggests that spectral indices computed from any of these data types are technically correct, although each will yield different index results for the same surface conditions.

This issue does not apply to the NDVI, but that is critically important for other vegetation indices (VI). However, if you do correct the data for atmospheric effects, that is likely to decrease values in the red and increase values in the NIR, leading to higher NDVI values calculated with reflectance, by comparison with those calculated with DNs.

Some VI incorporate numerical constants, typically determined using reflectance data. For example, the Soil Adjusted Vegetation Index includes the value 0.5 as a factor in the denominator of the equation, and the factor 1.5 as a multiplier. These values are scaled assuming that the red and NIR spectral data are measured in reflectance units, scaled 0-1. If DN values (e.g. 0-255) are used instead, the soil adjustment will be totally ineffective. The same applies to various other VI.

NDVI is a measure of healthy, green vegetation. The combination of its normalized difference formulation and use of the highest absorption and reflectance regions of chlorophyll make it robust over a wide range of conditions. It can, however, saturate in dense vegetation conditions when LAI (Leaf Area Index) becomes high. The value of this index ranges from -1 to 1. The common range for green vegetation is 0.2 to 0.8.

SAVI is similar to NDVI, but it suppresses the effects of soil pixels. It uses a canopy background adjustment factor, L, which is a function of vegetation density and often requires prior knowledge of vegetation amounts. Huete (1988) suggests an optimal value of L=0.5 to account for first-order soil background variations. This index is best used in areas with relatively sparse vegetation where soil is visible through the canopy.

NDWI reflects moisture content in plants and soil and is determined by analogy with NDVI. Instead of using the red range, the reflection intensity in which is determined by the presence of chlorophyll, a short-wave near-infrared (SWIR) is used in which high absorption of light by water occurs. A wider range of 1500-1750 nm is possible. The value of this index ranges from -1 to 1, high NDWI values correspond to high plant water content and coating of high plant fraction, while low NDWI values correspond to low vegetation content and cover with low vegetation. During periods of water stress the NDWI rate will decrease.

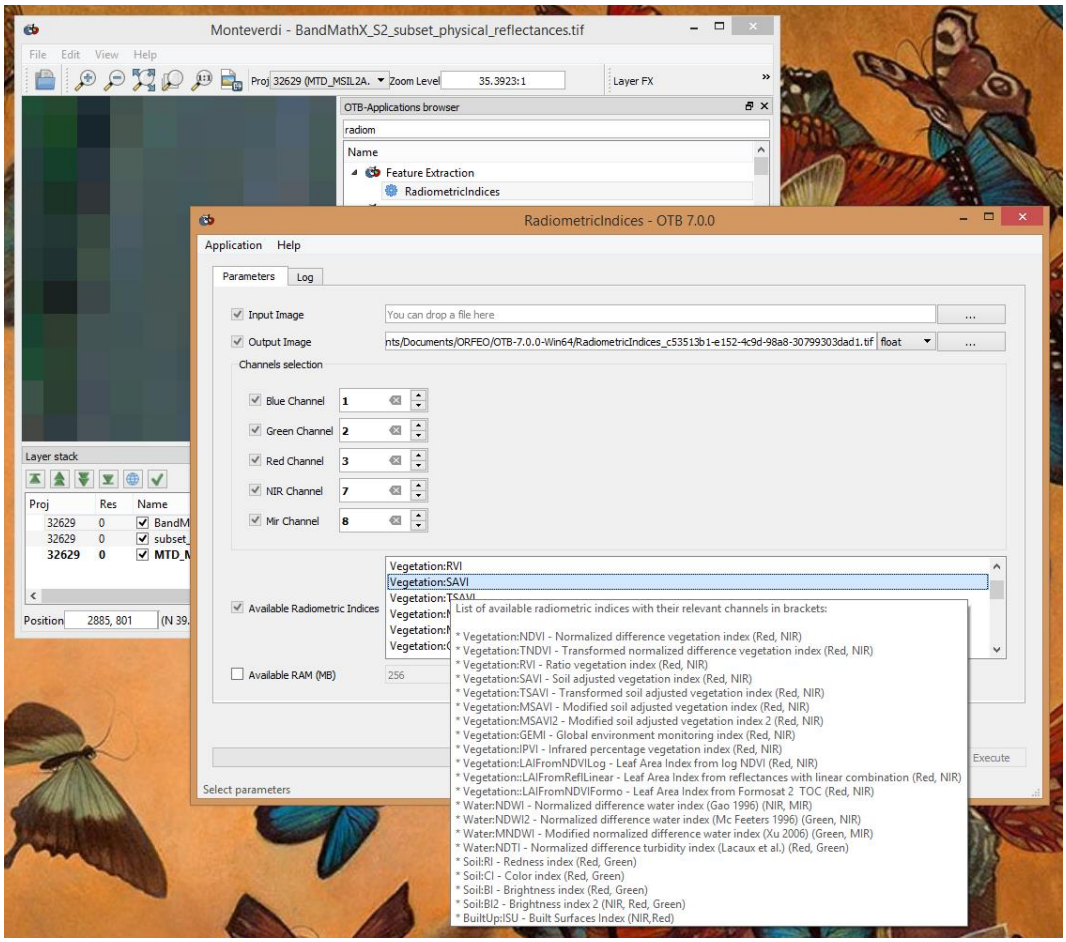
CI enables the estimation of the chlorophyll content of leaves based on reflectance in narrow red edge spectral bands. According to the typical spectral absorption characteristics of chlorophyll pigments, red and near-infrared (NIR) spectral bands are primarily used to build chlorophyll content indices. The red-

[14] Zarco-Tejada, P. J.; Miller, J. R.; Noland, T. L.; Mohammed, G. H.; Sampson, P. H. Scaling-up and Model Inversion Methods with Narrowband Optical Indices for Chlorophyll Content Estimation in Closed Forest Canopies with Hyperspectral Data. IEEE Trans. Geosci. Remote Sens., 2001. https://doi.org/10.1109/36.934080.

edge spectrum has received much attention for many years for monitoring chlorophyll content, and the red edge has been identified to be more sensitive to chlorophyll contents than the red part of the spectrum. The “red edge” refers to the steep part between the chlorophyll absorption valley in the red band and the high reflection shoulder in the NIR band. The value of this index is higher than 0, showing higher for healthy trees than for declining trees.

Be aware that, your image subset has now only 9 bands:

- Former Band 2 (Blue) is now Band 1;
- Former Band 3 (Green) is now Band 2;
- Former Band 4 (Red) is now Band 3;
- Former Band 5 (Red-edge1) is now Band 4;
- Former Band 6 (Red-edge2) is now Band 5;
- Former Band 7 (Red-edge3) is now Band 6;
- Former Band 8 (NIR) is now Band 7;
- Former Band 11 (SWIR1) is now Band 8;
- Former Band 12 (SWIR2) is now Band 9.



All the vegetation indices described above, except the Red-edge Chlorophyll Index, are implemented in **RadiometricIndices** OTB application.

c) So, to calculate this latter index you must use the **BandMath** OTB application instead.

OTB- Applications Browser >> Image Manipulation >> BandMath
Performs a mathematical operation on several multi-band images and outputs the result into a monoband image. The given expression is computed at each pixel position. Evaluation of the mathematical formula is done by the muParser library.
https://www.orfeo-toolbox.org/CookBook/Applications/app_BandMath.html

In this case the equation is the ratio between original bands 6 (numerator) and 5 (denominator), so the expression should be:

im1b5/im1b4

where **im** identifies the image and **b** identifies the band.

d) Considering the NDVI results, identify the threshold that enables the discrimination between vegetation and non-vegetation to generate a binary mask (raster file) to be used as ancillary data for a classification process for instance.

As mentioned previously, NDVI values range between 0.2 and 0.8 for green vegetation, yet you can refine the threshold value by identifying NDVI values common for healthy and dry vegetation in the Layer Stack window (Monteverdi footer window). After that, to create a binary mask use again the **BandMath** OTB application.

In this case, it is necessary to use an if-then-else operator: `(condition ? value_true : value_false)`

(im1b1 >= 0.2 ? 1 : 0)

using this condition, an image where value 1 corresponds to vegetated area whilst value 0 corresponds to non-vegetated areas is obtained.

PRACTICAL LESSON 4

EXERCISE 4.1

Dimension reduction is a statistical process, which concentrates the amount of information in multivariate data into a fewer number of variables (or dimensions). Though there are plenty of non-linear methods in the literature, OTB provides only linear dimension reduction techniques[15] applied to images for now: PCA (Principal Component Analysis); NA-PCA (Noise Adjusted PCA); MAF (Maximum Autocorrelation Factor), and ICA (Independent Component Analysis using a stabilized fixed point FastICA algorithm).

Usually, linear dimension-reduction algorithms try to find a set of linear combinations of the input image bands that maximise a given criterion, often chosen so that image information concentrates on the first components. Algorithms differs by the criterion to optimise and, also, by their handling of the signal or image noise.

In remote-sensing images processing, dimension reduction algorithms are of great interest for denoising, or as a preliminary processing for classification of feature images or unmixing of hyperspectral images. In addition to the denoising effect, the advantage of dimension reduction in the two latter is that it lowers the size of the data to be analysed, and as such, speeds up the processing time without too much loss of accuracy.

PCA uses an efficient method based on the inner product in order to compute the covariance matrix. The only parameter needed for the PCA is the number of principal components required as output (components are ordered by decreasing eigenvalues). Principal components are linear combination of input components (here the input image bands), which are selected using Singular Value Decomposition eigenvectors sorted by eigenvalue. We can choose to get less Principal Components than the number of input bands.

The **NA-PCA** transform is a sequence of two Principal Component Analysis transforms. The first transform is based on an estimated covariance matrix of the noise and intends to whiten the input image (noise with unit variance and no correlation between bands). The second Principal Component Analysis is then applied to the noise-whitened image, giving the Maximum Noise Fraction transform. Applying PCA on noise-whitened image consists in ranking Principal Components according to signal to noise ratio.

Like PCA, **MAF** (that performs a Maximum Autocorrelation Factor transform) tries to find a set of orthogonal linear transform, but the criterion to maximize is the spatial autocorrelation rather than the variance. Autocorrelation is the correlation between the component and a unitary shifted version of the component.

Again, identical to PCA, **ICA** computes a set of orthogonal linear combinations, but the criterion of Fast ICA is different: instead of maximizing variance, it tries to maximize statistical independence between components. In the Fast ICA algorithm, statistical independence is measured by evaluating non-Gaussianity of the components, and the maximization is done in an iterative way.

[15] https://www.orfeo-toolbox.org/SoftwareGuide/SoftwareGuidech18.html

a) Perform a Principal Component Analysis on two Sentinel-2 image's subsets (subset_S2A_L2A_4junho17 and subset_S2A_L2A_4julho17), available at OneDrive, using the **DimensionalityReduction** OTB application. Both images have only 10 bands (all bands except B1, B9 and B10).

OTB- Applications Browser >> Image Filtering >> DimensionalityReduction
Performs dimensionality reduction on input image. PCA, NA-PCA, MAF, ICA methods are available. It is also possible to compute the inverse transform to reconstruct the image and to optionally export the transformation matrix to a text file.
https://www.orfeo-toolbox.org/CookBook/Applications/app_DimensionalityReduction.html

Principal Components Analysis is used to produce uncorrelated output bands, to segregate noise components, and to reduce the dimensionality of data sets. Because multispectral data bands are often highly correlated, the principal components (PC) transformation is used to produce uncorrelated output bands. This is done by finding a new set of orthogonal axes that have their origin at the data mean and that are rotated so the data variance is maximized (Figure 4.1).

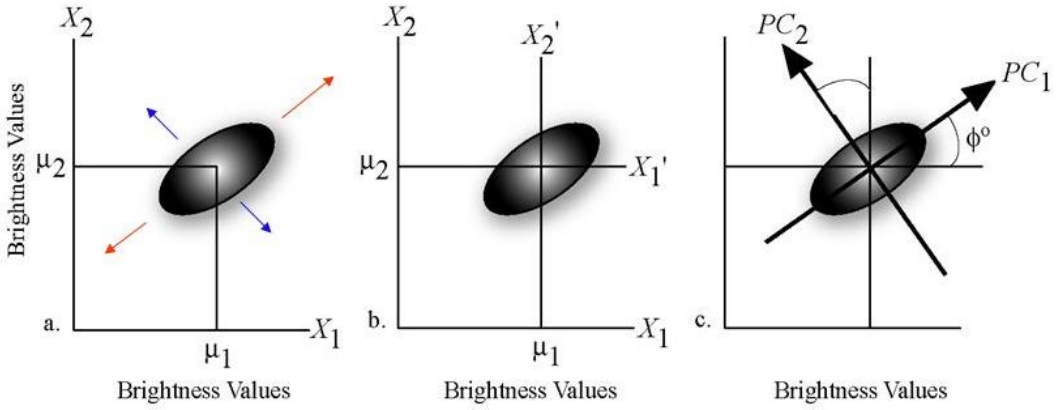
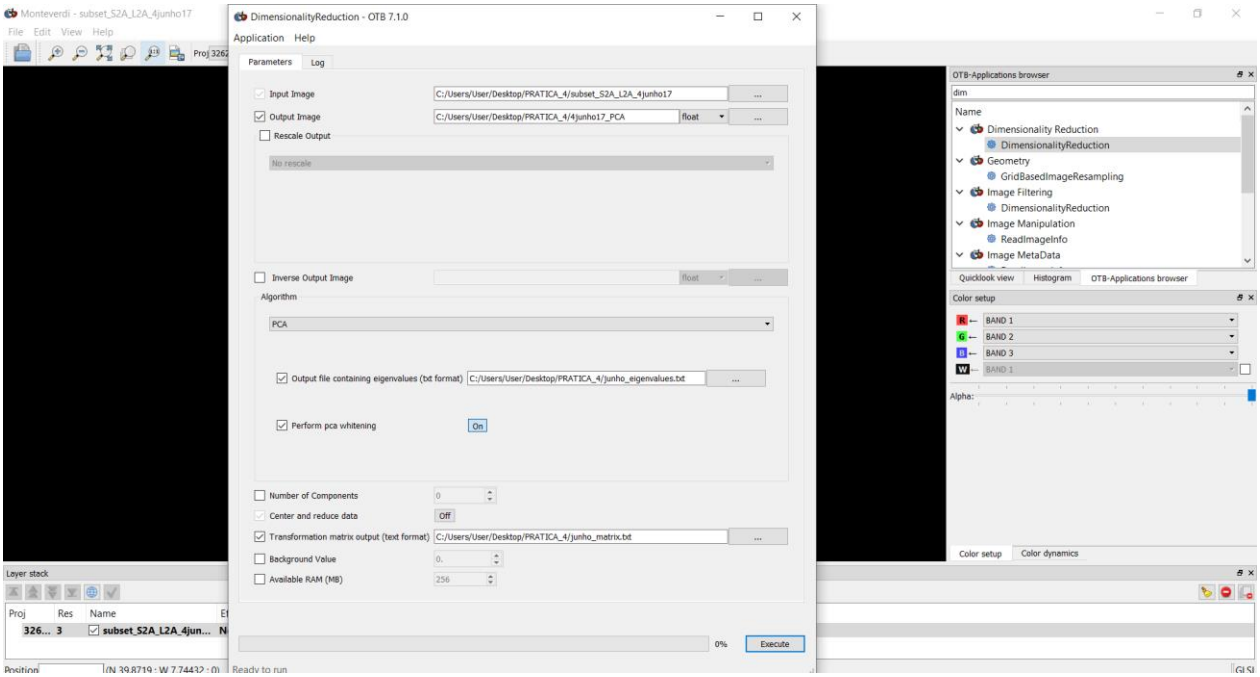


Figure 4.1. Principal Component Analysis (PCA).

PC bands are linear combinations of the original spectral bands and are uncorrelated. You can calculate the same number of output PC bands as input spectral bands. The first PC band contains the largest percentage of data variance and the second PC band contains the second largest data variance, and so on. The last PC bands appear noisy because they contain very little variance, much of which is due to noise in the original spectral data. The first two or three components, usually, contain over 90% of the information from the original bands. PC bands produce more colorful color composite images than spectral color composite images because the data is uncorrelated. OTB can complete forward and reverse PC rotations (Figure 4.2).

Figure 4.2. Result of applying a forward and reverse PCA transformation to a Worldview-2 image. From left to right: original image, color composition with first three principal components and output of the inverse mode (the input bands).



If the number of relevant components is not defined, by default all components are kept. In the output image the components are ordered by decreasing eigenvalues.

The output file containing eigenvalues (txt format) will look like this (use WordPad to open the junho_eigenvalues.txt file).

```
[3558877.4176377784, 1229399.0263627854, 161440.8846256116,
48623.7726765070, 15700.8762517341, 11980.1853732957,
6987.5082997192, 6186.0795512886, 3519.3698348313,
1980.7562758227]
```

EXERCISE 4.2

Change detection techniques try to detect and locate areas which have changed between two or more observations of the same scene. These changes can be of different types, with different origins and of different temporal length. This allows to distinguish different kinds of applications:

- land use monitoring, which corresponds to the characterization of the evolution of the vegetation, or its seasonal changes;
- natural resources management, which corresponds mainly to the characterization of the evolution of the urban areas, the evolution of the deforestation, etc.
- damage mapping, which corresponds to the location of damages caused by natural or industrial disasters.

From the point of view of the observed phenomena, one can distinguish two types of changes whose nature is rather different: the abrupt changes and the progressive changes, which can eventually be periodic. From the data point of view, one can have:

- Image pairs before and after the event. The applications are mainly the abrupt changes.
- Multi-temporal image series on which two types of changes may appear: (1) slow changes like for instance the erosion, vegetation evolution, etc, and (2) abrupt changes may pose different kinds of problems depending on whether the date of the change is known in the image series or not (in this case, the problem has a higher difficulty).

The problem of detecting abrupt changes between a pair of images is the following: Let I_1 and I_2 be two images acquired at different dates t_1 and t_2; we aim at producing a thematic map which shows the areas where changes have taken place.

Two main categories of methods exist:

- Strategy 1: Post Classification Comparison: the principle of this approach is two obtain two land-use maps independently for each date and comparing them.
- Strategy 2: Image Comparison: this approach consists in producing an image of change likelihood (by differences, ratios or any other approach) and thresholding it in order to produce the change map.

a) Detect abrupt changes in a Sentinel-2 image pair, acquired before and after the major wildfire disaster in *Pedrogão Grande* in June 2017, and establish a threshold value in order to identify the fire perimeter using the **BandMath** OTB application. Because of its simplicity and its low computation overhead, this strategy is the chosen one for the processing requested in this exercise. First, two NDVI images should be generated based on two Sentinel-2 image's subsets (subset_S2A_L2A_4junho17 and subset_S2A_L2A_4julho17) that are available at OneDrive. Both images have only 10 bands excluding bands B1, B9 and B10, so when calculating NDVI you should use band 3 as RED and band 7 as NIR.

In this case, the equation is the difference between the NDVI images in both dates (July - June), so the expression should be (when loading images into **BandMath**, the June image should be the first while the July image should be the second):

im2b1 - im1b1

where **im** identifies the image and **b** identifies the band. In this case, burned areas will exhibit values lower than 0, values around 0 mean no change and values higher than 0 for vegetated areas with an increase in the NDVI.

Identify the threshold that enables the discrimination between burned and non-burned vegetation to generate a binary mask (raster file) with the wildfire perimeter in order to calculate the total area affected by the fire. Again, you must identify typical values for burned and non-burned vegetation areas in the Layer Stack window (Monteverdi footer window). After that, to create a binary mask use again the **BandMath** OTB application.

In this case, it is necessary to use an if-then-else operator: (condition ? value_true : value_false)

(im1b1 <= *threshold value* ? 1 : 0)

using this condition, an image where value 1 corresponds to burned area whilst value 0 corresponds to non-burned areas is obtained.

b) Repeat the previous exercise using, instead of NDVI, the first two Principal Components (PC1) obtained in exercise 4.1.

c) To calculate the total area affected by the fire you should use QGIS.

First, convert the raster file into a polygon vector file

Raster \ Conversion \ Poligonize (Raster to Vector) …

This might take a while ….

Then, start editing the new shapefile, to delete all non-burned area polygons (DN=0) and all burned area polygons, that were misclassified (DN=1), with an area less than 2ha (20 000 m^2). You might have to delete some other polygons outside the fire perimeter. Finally, the sum of the areas of all the remaining polygons corresponds to the total burned area.

PRACTICAL LESSON 5

Machine-learning offers the potential for effective and efficient classification of remotely sensed imagery. The strengths of machine learning include the capacity to handle data of high dimensionality, to map classes with very complex characteristics, to accept a variety of input predictor data, and to not make assumptions about the data distribution (i.e. are nonparametric). Nevertheless, implementing a machine-learning classification is not straightforward, and the literature provides conflicting advice regarding many key issues.

Despite the increasing acceptance of machine-learning classifiers, parametric methods appear still to be commonly used in application articles and remain one of the major standards for benchmarking classification experiments. For example, the parametric maximum likelihood (ML) classifier has been the most used method, even though machine-learning methods were routinely found to have notably higher accuracies than ML. This results from the fact that ML is widely available in conventional remote-sensing image-processing software packages and to uncertainties regarding how to use and implement machine-learning techniques effectively. Relatively mature methods of machine-learning are support vector machines (SVM), single decision trees (DTs), Random Forests (RF), boosted DTs (Boosted DTs), artificial neural networks (ANN), and k-nearest neighbors (*k*-NN).

Orfeo ToolBox ships with a set of application to perform supervised or unsupervised pixel-based image classification[16]. This framework allows to learn from multiple images, and using several-machine learning method 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.

EXERCISE 5.1

Classify a subset of a Sentinel-2 image (subset_2_of_S2B_MSIL1C_20180326_resampled.tif), with the Random Forest method, using a polygon shapefile with the training samples (parcels_test.shp). Both types of data can be downloaded from OneDrive.

[16] https://www.orfeo-toolbox.org/CookBook/recipes/pbclassif.html#pixel-based-classification

STEP 1: Compute samples statistics for each image – PolygonClassStatistics

OTB- Applications Browser >> Learning >> PolygonClassStatistics
Computes 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.
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:

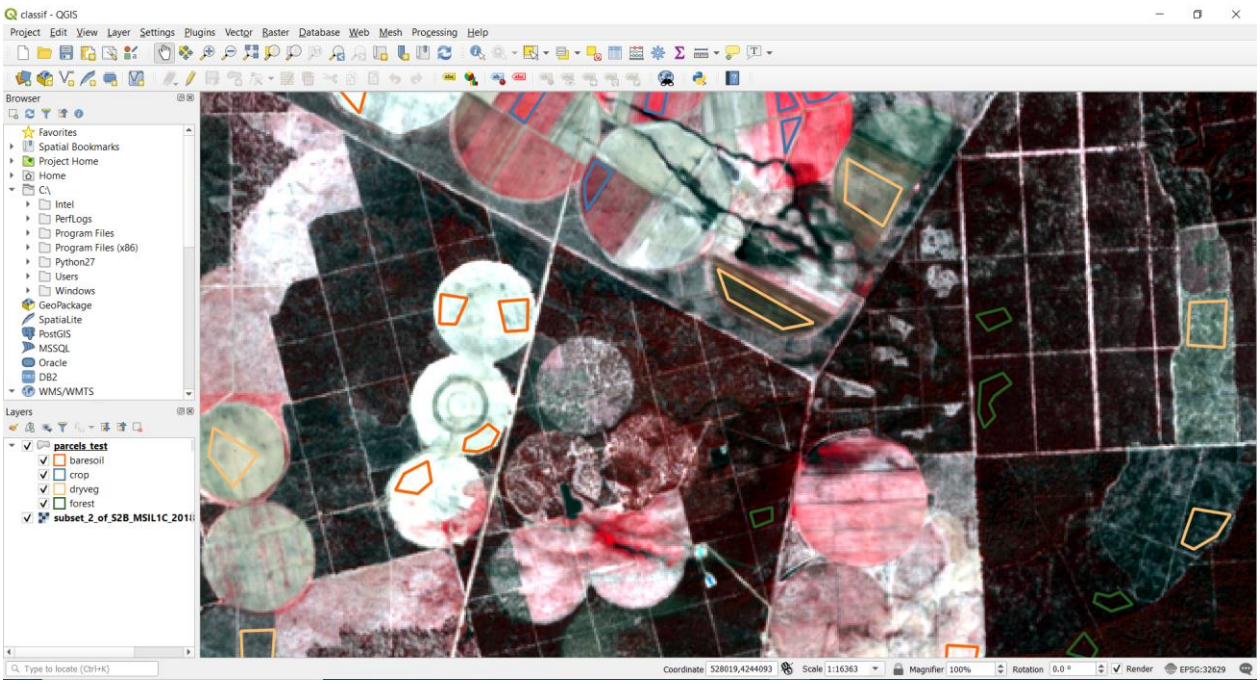
- number of samples per class;
- number of samples per geometry.

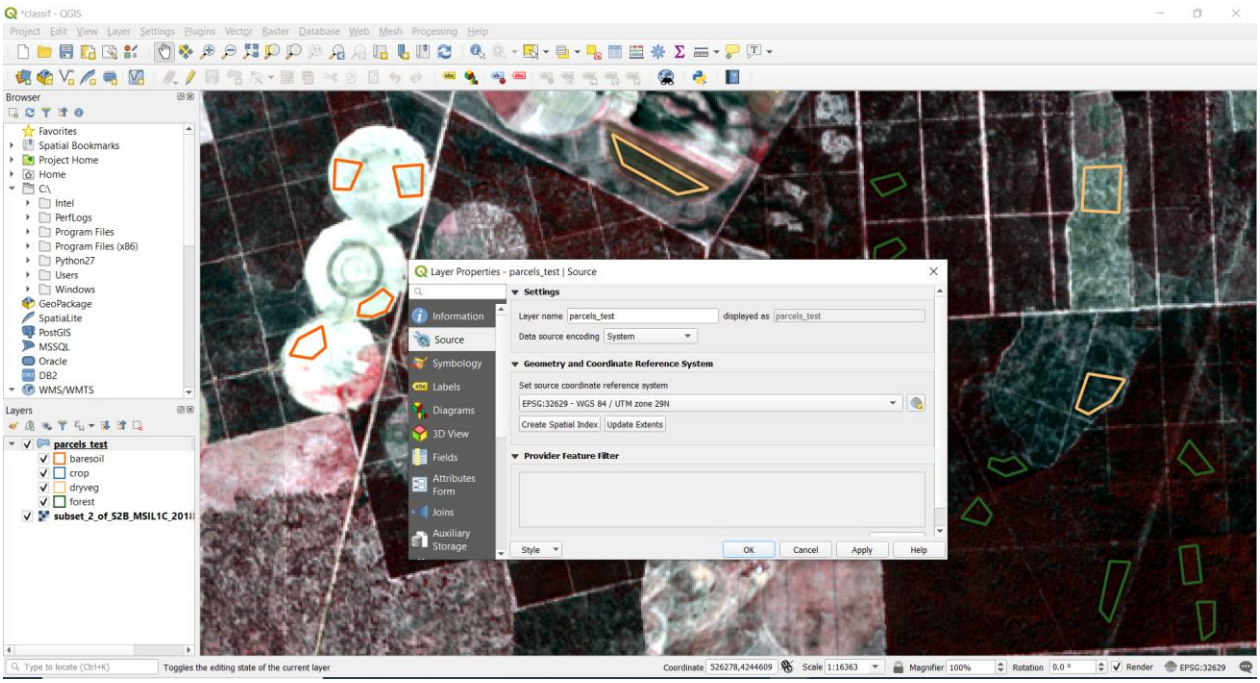
Supported geometries are polygons, lines and points. Depending on the geometry type, this application behaves differently:

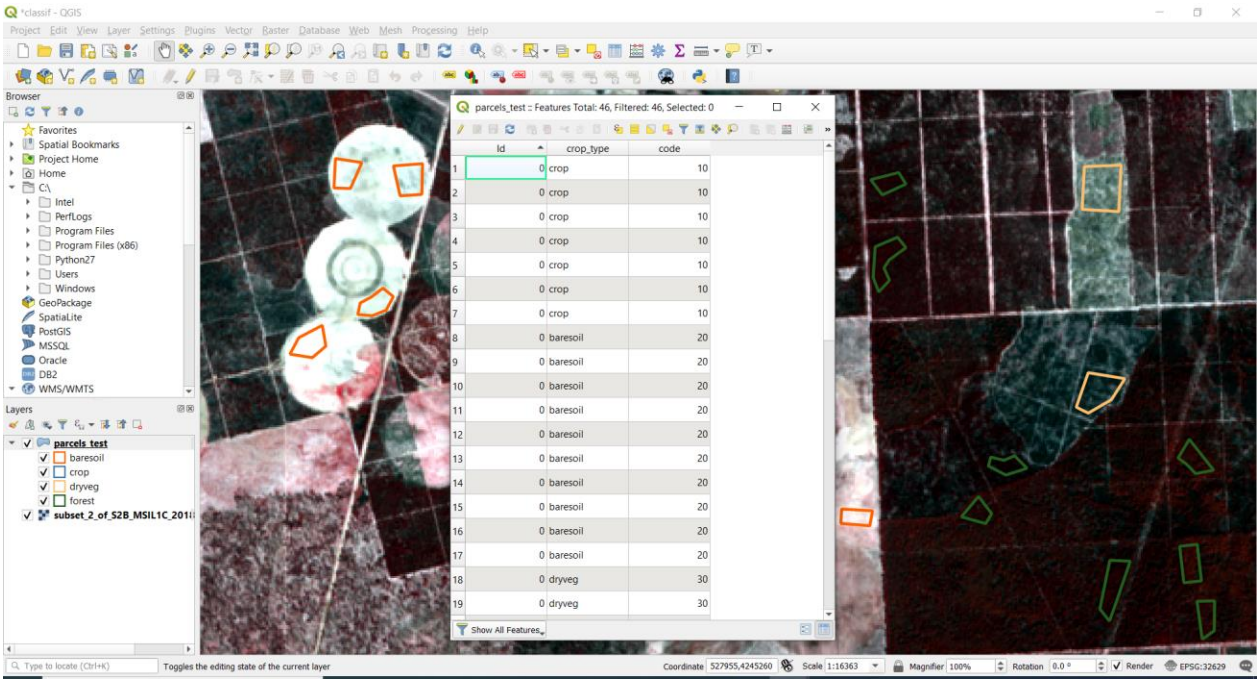
- polygons: select pixels whose center falls inside the polygon;
- lines: select pixels intersecting the line;
- points: select closest pixel to the provided point.

These training geometries must be, previously, defined by the user, using ArcMap or QGIS (the same reference system of the image to be classified must be adopted for this file). In this case, we are going to use a vector file with several polygons (training areas) created for each of the 4 land use/cover classes considered (crop, bare soil, dry vegetation and forest). The shapefile has 46 samples, 7 for crop (code 10), 10 for bare soil (code 20), 13 for dry vegetation (code 30) and 16 for forest (code 40). This is done by digitizing, on screen, polygons corresponding to those land use classes, having the image in the background. Try to define areas with the same spectral characteristics! 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 this field available will be considered.

Note that, this is a very simplistic example just for you to know how this works! In a real application, the number of training areas should be much higher, specially when working with machine-learning classification algorithms, as well as the size of the image that is usually much larger.

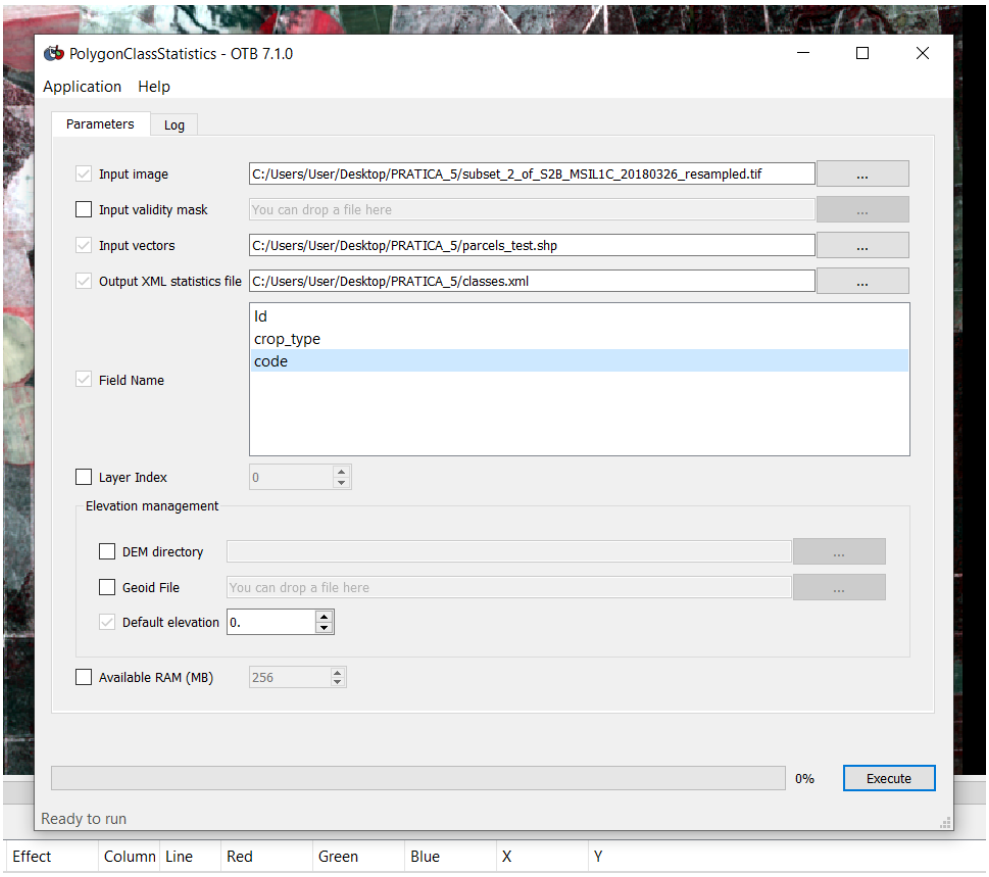

classif - QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help
Browser
Favorites
Spatial Bookmarks
Project Home
Home
C:\
Intel
PerfLogs
Program Files
Program Files (x86)
Python27
Users
Windows
GeoPackage
SpatiaLite
PostGIS
MSSQL
Oracle
DB2
WMS/WMTS
Layers
parcels test
baresoil
crop
dryveg
forest
subset_2_of_S2B_MSIL1C_201
Type to locate (Ctrl+K)
Coordinate 528019,4244093
Scale 1:16363
Magnifier 100%
Rotation 0.0 °
Render
EPSG:32629


*classif - QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help
Browser
Favorites
Spatial Bookmarks
Project Home
Home
C:\
Intel
PerfLogs
Program Files
Program Files (x86)
Python27
Users
Windows
GeoPackage
SpatiaLite
PostGIS
MSSQL
Oracle
DB2
WMS/WMTS
Layers
parcels test
baresoil
crop
dryveg
forest
subset_2_of_S2B_MSIL1C_201
Layer Properties - parcels_test | Source
Information
Source
Symbology
Labels
Diagrams
3D View
Fields
Attributes Form
Joins
Auxiliary Storage
Settings
Layer name parcels_test
displayed as parcels_test
Data source encoding System
Geometry and Coordinate Reference System
Set source coordinate reference system
EPSG:32629 - WGS 84 / UTM zone 29N
Create Spatial Index
Update Extents
Provider Feature Filter
Style
OK
Cancel
Apply
Help
Type to locate (Ctrl+K)
Toggles the editing state of the current layer
Coordinate 526278,4244609
Scale 1:16363
Magnifier 100%
Rotation 0.0 °
Render
EPSG:32629



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. As mentioned before, the values in this field shall be cast into integers.



The output XML file will look like this (use WordPad to open the classes.xml file).

```
<?xml version="1.0" ?>
<GeneralStatistics>
    <Statistic name="samplesPerClass">
        <StatisticMap key="10" value="1866" />
        <StatisticMap key="20" value="2886" />
        <StatisticMap key="30" value="5554" />
        <StatisticMap key="40" value="2643" />
    </Statistic>
    <Statistic name="samplesPerVector">
        <StatisticMap key="0" value="280" />
        <StatisticMap key="1" value="195" />
        <StatisticMap key="10" value="78" />
        <StatisticMap key="11" value="253" />
        <StatisticMap key="12" value="148" />
        <StatisticMap key="13" value="122" />
        <StatisticMap key="14" value="57" />
        <StatisticMap key="15" value="124" />
        <StatisticMap key="16" value="124" />
        <StatisticMap key="17" value="229" />
        <StatisticMap key="18" value="192" />
        <StatisticMap key="19" value="162" />
        <StatisticMap key="2" value="213" />
        <StatisticMap key="20" value="173" />
        <StatisticMap key="21" value="578" />
        <StatisticMap key="22" value="690" />
        <StatisticMap key="23" value="368" />
        <StatisticMap key="24" value="219" />
        <StatisticMap key="25" value="151" />
        <StatisticMap key="26" value="135" />
        <StatisticMap key="27" value="400" />
        <StatisticMap key="28" value="601" />
```

STEP 2: Compute sampling rates for each image – MultiImageSamplingRate

Note: Skip this step once we are working with only one image!

OTB- Applications Browser >> Learning >> MultiImageSamplingRate
Computes the sampling rate for an input set of images. Before calling this application, each pair of image and training vectors has to be analysed with the application **PolygonClassStatistics**.
https://www.orfeo-toolbox.org/CookBook/Applications/app_MultiImageSamplingRate.html

If the training set spans several images, the MultiImageSamplingRate application allows to compute the appropriate sampling rates per image and per class, in order to get samples that span the entire extents of the images.

It is first required to run the PolygonClassStatistics application on each image of the set separately. The MultiImageSamplingRate application will then read all the produced statistics XML files and derive the sampling rates according the sampling strategy.

STEP 3: Select samples positions for each image – SampleSelection

OTB- Applications Browser >> Learning >> SampleSelection
Selects 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 be analyzed by the **PolygonClassStatistics** application to compute statistics about the geometries, which are summarized in an XML file. Then, this XML file must be given as an input to this application (parameter instats).
https://www.orfeo-toolbox.org/CookBook/Applications/app_SampleSelection.html

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.

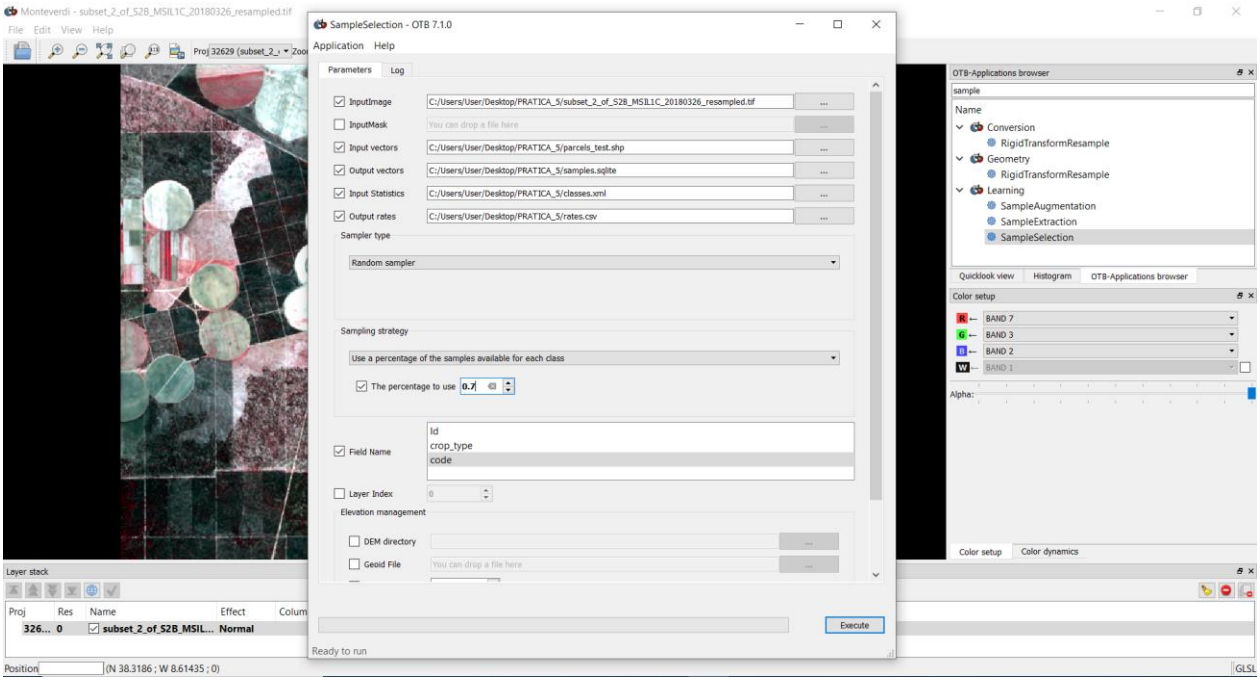
- **Smallest class strategy:** The class with the least number of samples will be fully sampled. All other classes will be sampled with the same number of samples.
- **Percent strategy:** Each class will be sampled with a user-defined percentage (same value for all classes) of samples available in this 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 spans the whole training set extent by adjusting the sampling rate. Depending on the strategy to determine the sampling rate, some geometries of the training set may not be sampled.

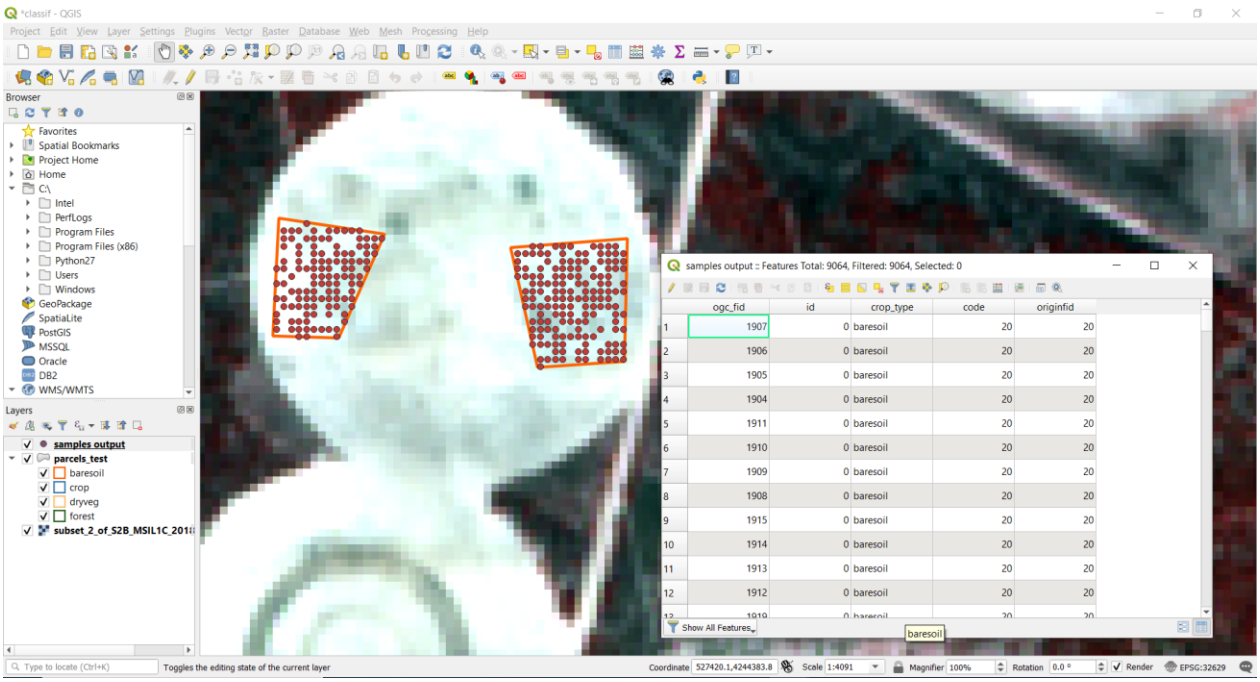
The application will accept as input the input image and training geometries, as well class statistics XML file computed during the previous step. It will output a vector file containing point geometries which indicate the location of the samples.



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

```
#className requiredSamples totalSamples rate
10    1306  1866  0.7
20    2020  2886  0.7
30    3888  5554  0.7
40    1850  2643  0.7
```

The samples.sqlite file might be viewed using QGIS (open it as a vector file). The red dots show the samples that have been selected.

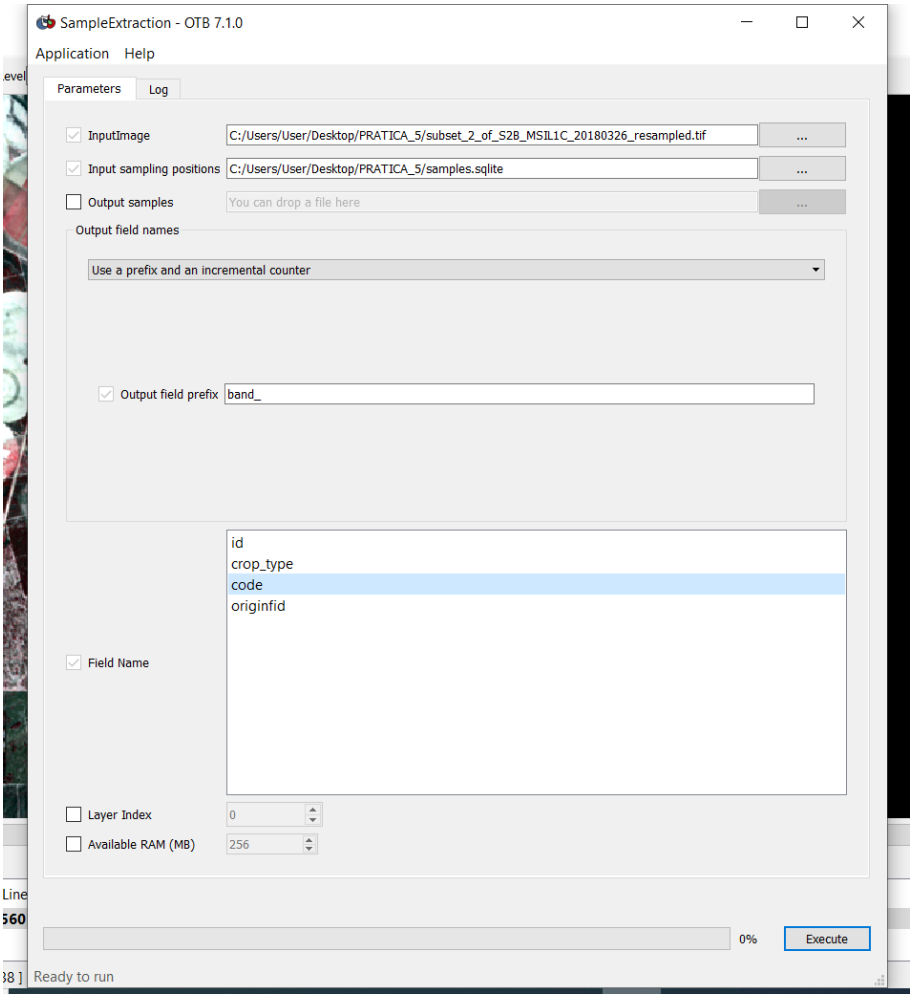


STEP 4: Extract samples measurements for each image – SampleExtraction

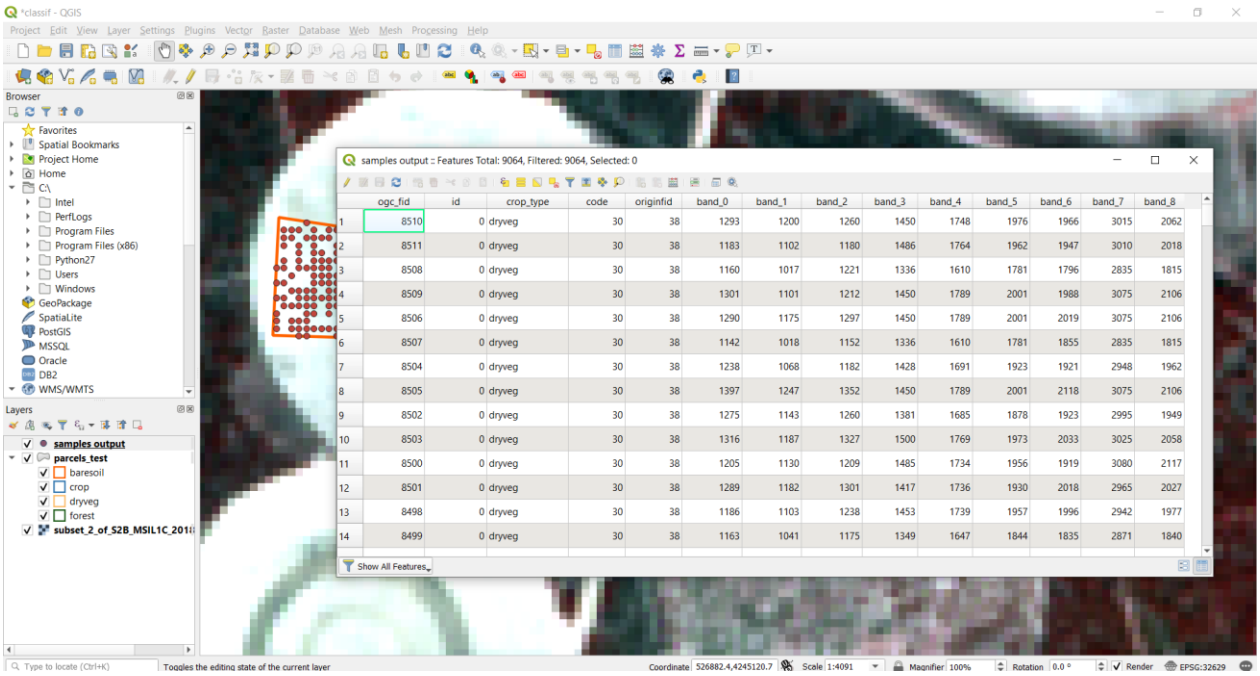
OTB- Applications Browser >> Learning >> SampleExtraction
Extracts samples values from an image. The application extracts samples values from an image using positions contained in a vector data file.
https://www.orfeo-toolbox.org/CookBook/Applications/app_SampleExtraction.html

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.



Here is the attributes table of the updated samples file.



STEP 5: Compute images statistics – ComputeImagesStatistics

OTB- Applications Browser >> Learning >> ComputeImagesStatistics

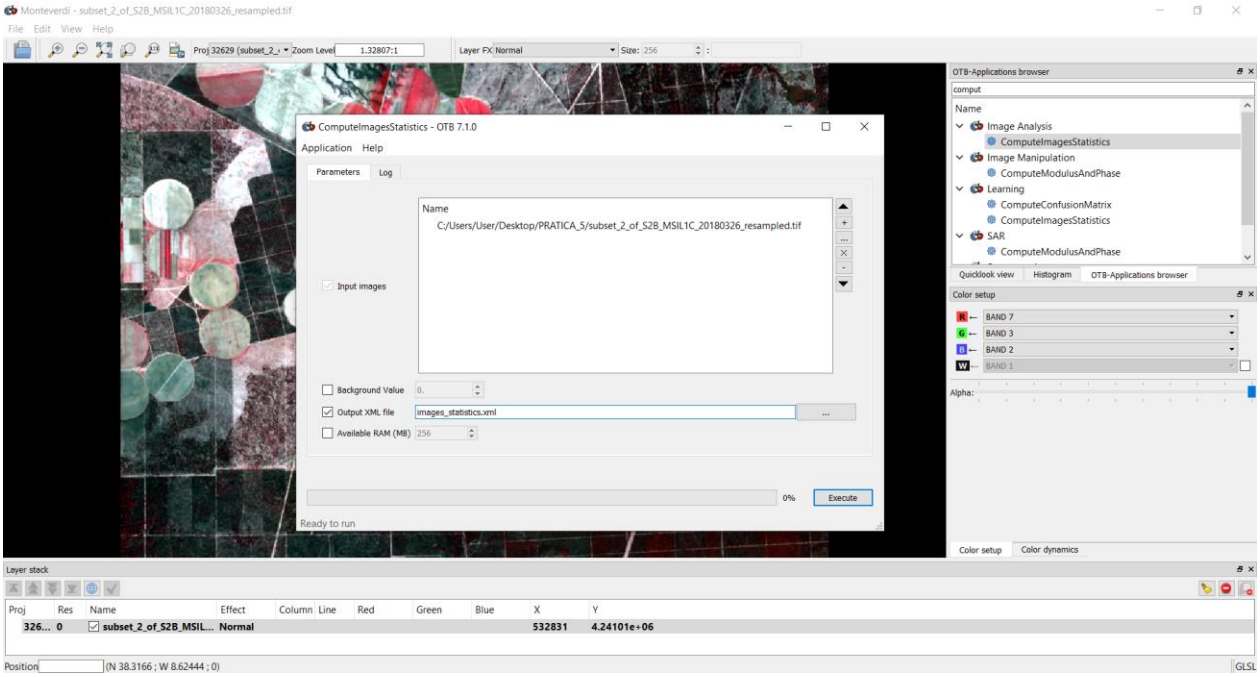
Computes global mean and standard deviation for each band from a set of images and optionally saves the results in an XML file. Each image of the set must contain the same bands as the others (i.e. same types, in the same order).

This application computes a global mean and standard deviation for each band of a set of images and optionally saves the results in an XML file. The output XML is intended to be used as an input for the **TrainImagesClassifier** application to normalize samples before learning. You can also normalize the image with the XML file in the **ImageClassifier** application.

https://www.orfeo-toolbox.org/CookBook/Applications/app_ComputeImagesStatistics.html

Some machine learning algorithms converge faster if the range of features is [-1, 1] or [0, 1]. Other will be sensitive to relative ranges between feature, e.g. a feature with a larger range might have more weight in the final decision. This is for instance the case for machine learning algorithm using euclidean distance at some point to compare features (such as Support Vector Machines – SVM). In those cases, it is advised to normalize all features to the range [-1, 1] before performing the learning. For this purpose, the

ComputeImageStatistics application allows to compute and output to an XML file the mean and standard deviation based on pooled variance of each band for one or several images.



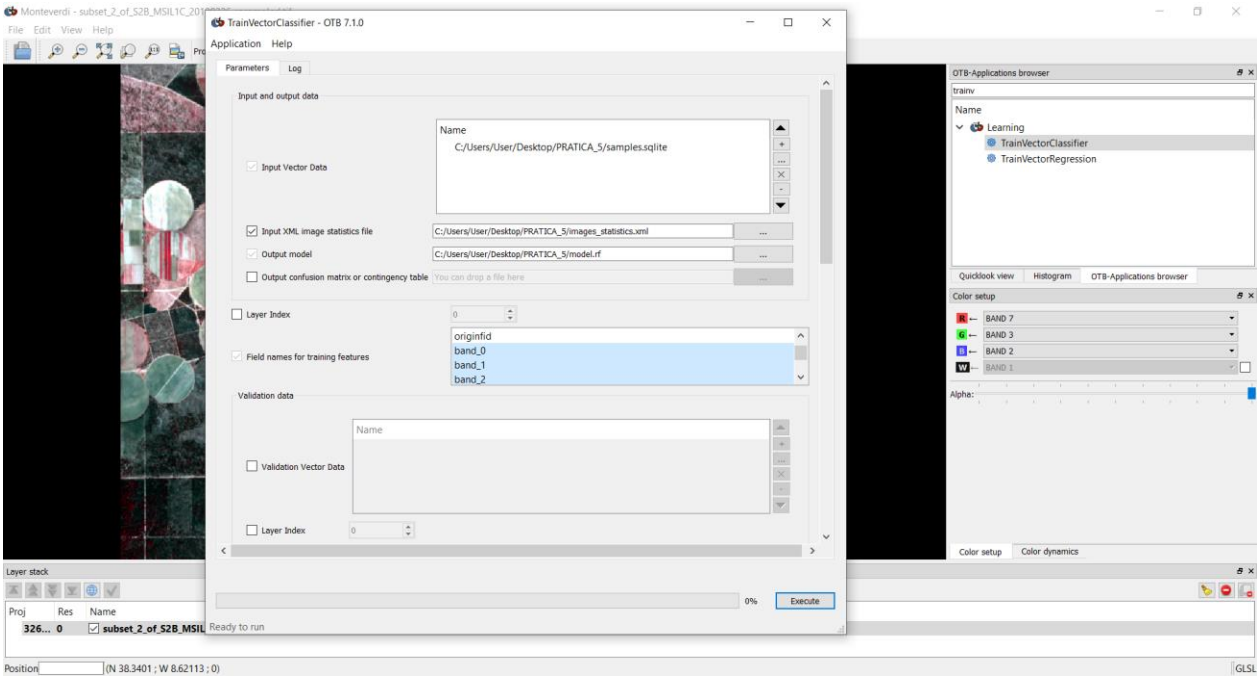
The output statistics file can then be fed to the training and classification applications (use WordPad to open the images_statistics.xml file).

```
<?xml version="1.0" ?>
<FeatureStatistics>
    <Statistic name="mean">
        <StatisticVector value="1372.69" />
        <StatisticVector value="1287.56" />
        <StatisticVector value="1372.95" />
        <StatisticVector value="1579.76" />
        <StatisticVector value="2087.46" />
        <StatisticVector value="2349.99" />
        <StatisticVector value="2355.58" />
        <StatisticVector value="2933.31" />
        <StatisticVector value="2200.63" />
    </Statistic>
    <Statistic name="stddev">
        <StatisticVector value="454.981" />
        <StatisticVector value="546.184" />
        <StatisticVector value="728.434" />
        <StatisticVector value="736.93" />
        <StatisticVector value="737.503" />
        <StatisticVector value="777.094" />
        <StatisticVector value="774.865" />
        <StatisticVector value="1092.49" />
        <StatisticVector value="1113.34" />
    </Statistic>
</FeatureStatistics>
```

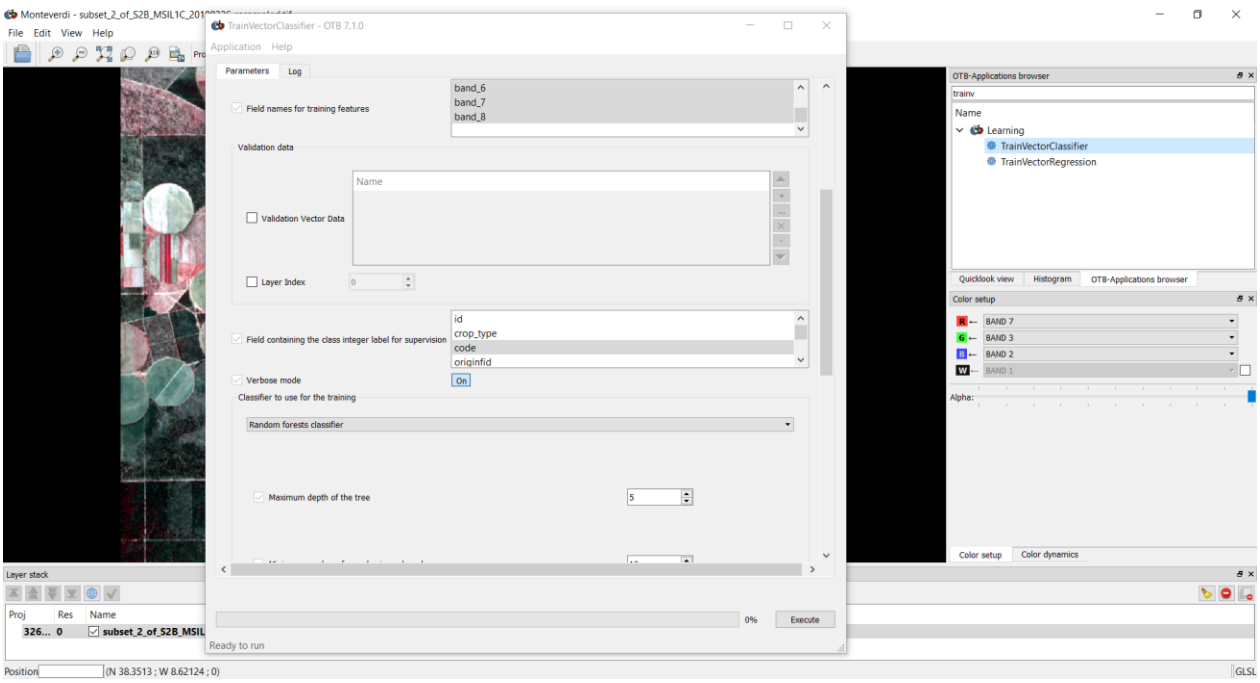
STEP 6: Train the machine learning model – TrainVectorClassifier

OTB- Applications Browser >> Learning >> TrainVectorClassifier
Trains 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.
https://www.orfeo-toolbox.org/CookBook/Applications/app_TrainVectorClassifier.html

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



...



RANDOM FOREST CLASSIFIER OPTIONS (in this case, use the default values)

(more details in 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 <= cat clusters to find a suboptimal split

-classifier.rf.cat int DEFAULT VALUE: 10

Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split.

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

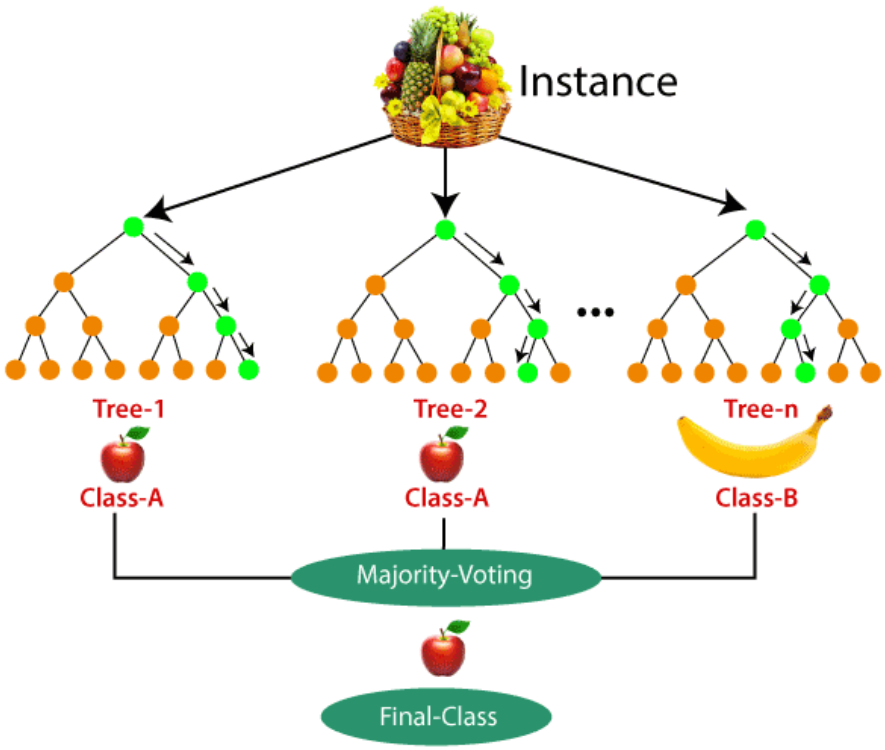
The size of the subset of features, randomly selected at each tree node, that are used to find the best split(s). If you set it to 0, then the size will be set to the square root of the total number of features.

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

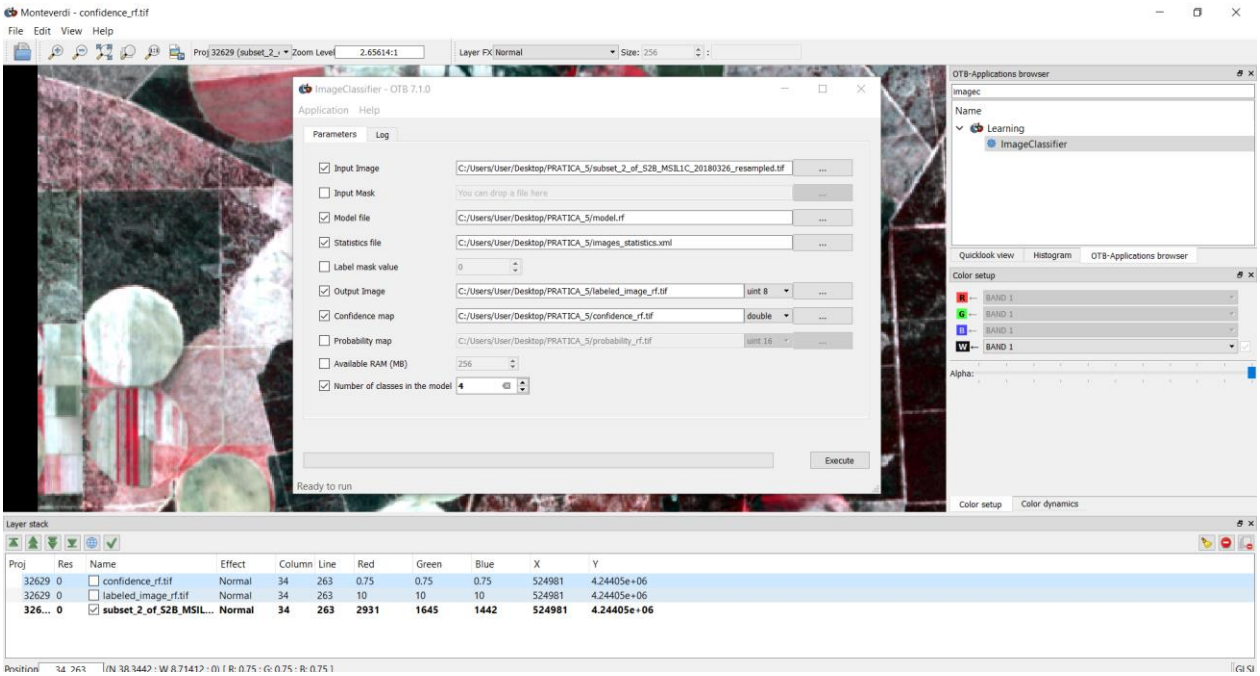
Sufficient accuracy (OOB error).



STEP 7: Perform the classification by applying the model – ImageClassifier

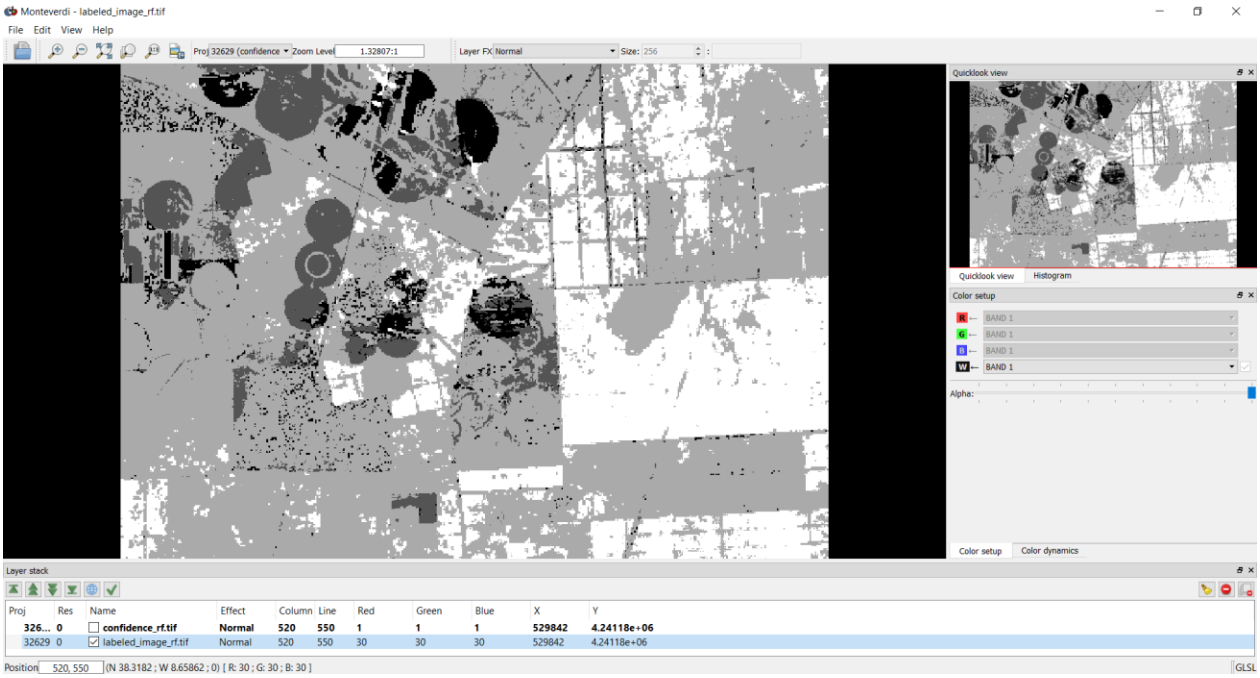
OTB- Applications Browser >> Learning >> ImageClassifier
Performs a classification of the input image according to a model file. This application performs an image classification based on a model file produced by the **TrainImagesClassifier** or the **TrainVectorClassifier** application. Pixels of the 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.
https://www.orfeo-toolbox.org/CookBook/Applications/app_ImageClassifier.html

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

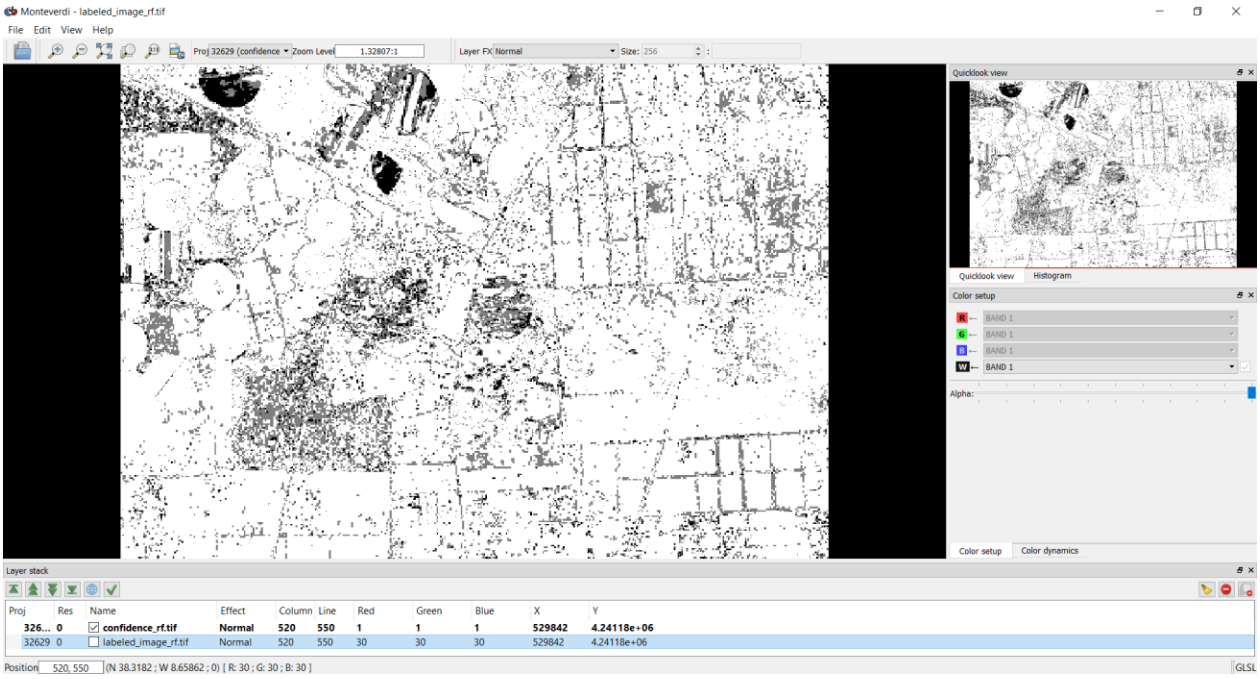


Note that, the probability map is only implemented for the Shark Random Forest classifier at this point.

Classified image with 4 classes



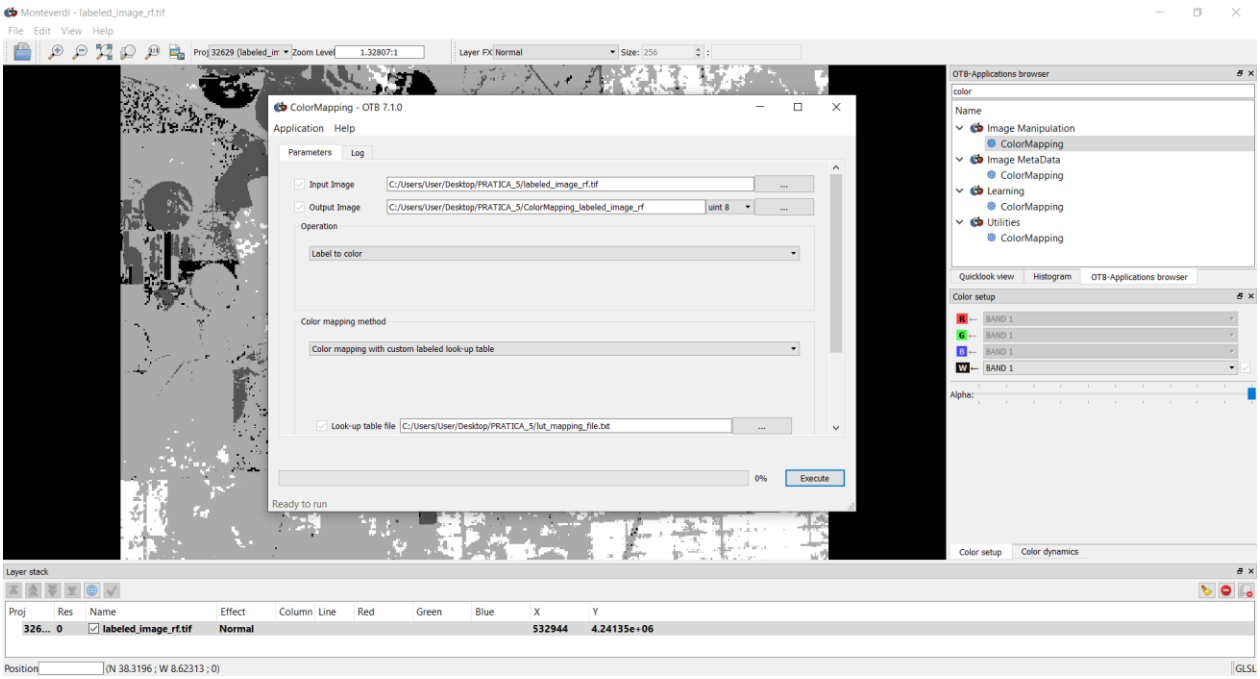
Confidence map



EXERCISE 5.2

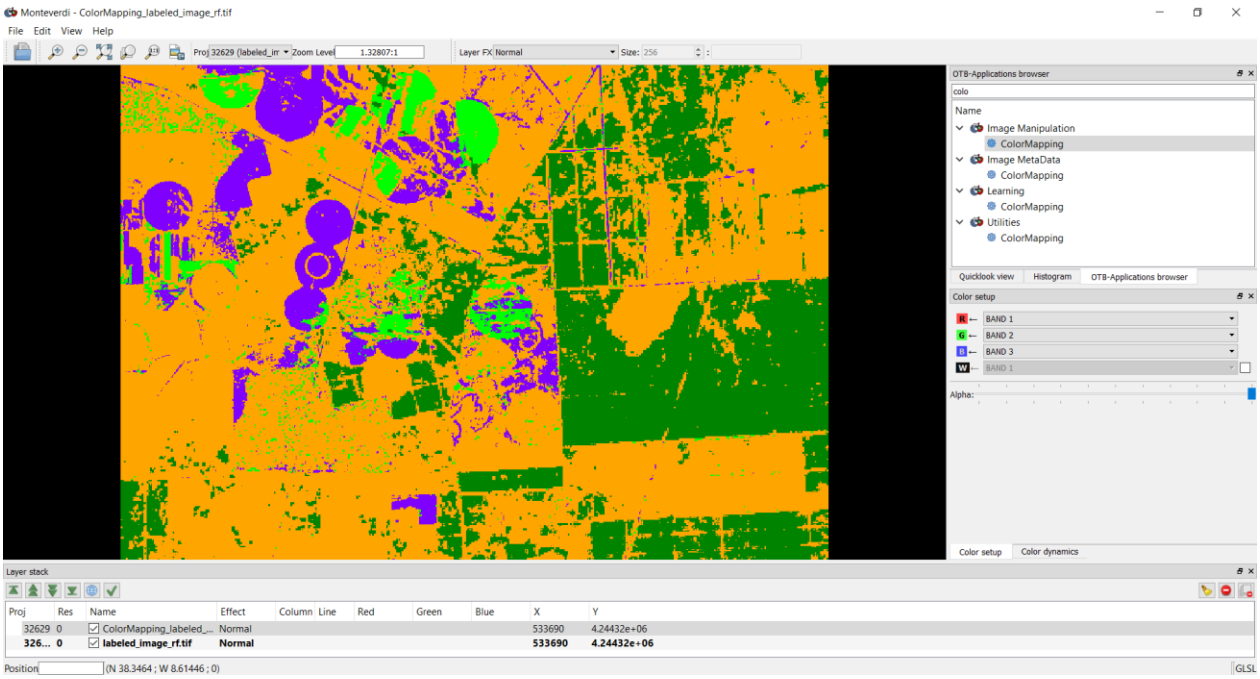
Map the labelled image, produced in the previous exercise, to an 8-bits image using the **ColorMapping** OTB application.

OTB- Applications Browser >> Image Manipulation >> ColorMapping
Maps a label image to an 8-bits RGB image (both ways) using different methods.
https://www.orfeo-toolbox.org/CookBook/Applications/app_ColorMapping.html



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
10 0 255 0
20 100 100 50
30 200 200 0
40 0 180 0
0 0 0 0
```


Monteverdi - ColorMapping_labeled_image_rf.tif
File Edit View Help
OTB-Applications browser
Name
Image Manipulation
ColorMapping
Image MetaData
ColorMapping
Learning
ColorMapping
Utilities
ColorMapping
Quicklook view
Histogram
OTB-Applications browser
Color setup
BAND 1
BAND 2
BAND 3
Alpha:
Color setup
Color dynamics
Layer stack
Proj
Res
Name
Effect
Column
Line
Red
Green
Blue
X
Y
32629 0 ColorMapping_labeled_... Normal 533690 4.24432e+06
326... 0 labeled_image_rf.tif Normal 533690 4.24432e+06
Position
(N 38.3464 ; W 8.61446 ; 0)
GLSL

CONFUSION (ERROR) MATRIX

An error matrix compares information from reference sites to information on the map for a number of sample areas. 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, usually the columns, is assumed to be correct and is termed the reference data[17]. The rows are usually used to display the map labels or classified data generated from the remotely sensed image. 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.

[17] In the **ComputeConfusionMatrix** OTB application, the confusion matrix is organized the following way: rows = reference labels, columns = produced (map) labels.

		Reference Data						
	Classes	**A**	**B**	**C**	**D**	**Row Total**	**Precision**	**F1-score**
Classified Data	**A**	**65**	4	22	24	115	65/115= 0.565	0.684
	B	6	**81**	5	8	100	81/100= 0.810	0.798
	C	0	11	**85**	19	115	85/115= 0.739	0.739
	D	4	7	3	**90**	104	90/104= 0.865	0.735
	Column Total	75	103	115	141	**434**(*)		
	Recall	65/75= 0.867	81/103= 0.786	85/115= 0.739	90/141= 0.638			
	Overall Accuracy= 0.74 **Kappa**= 0.654							

(*) the sum of all columns 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.

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

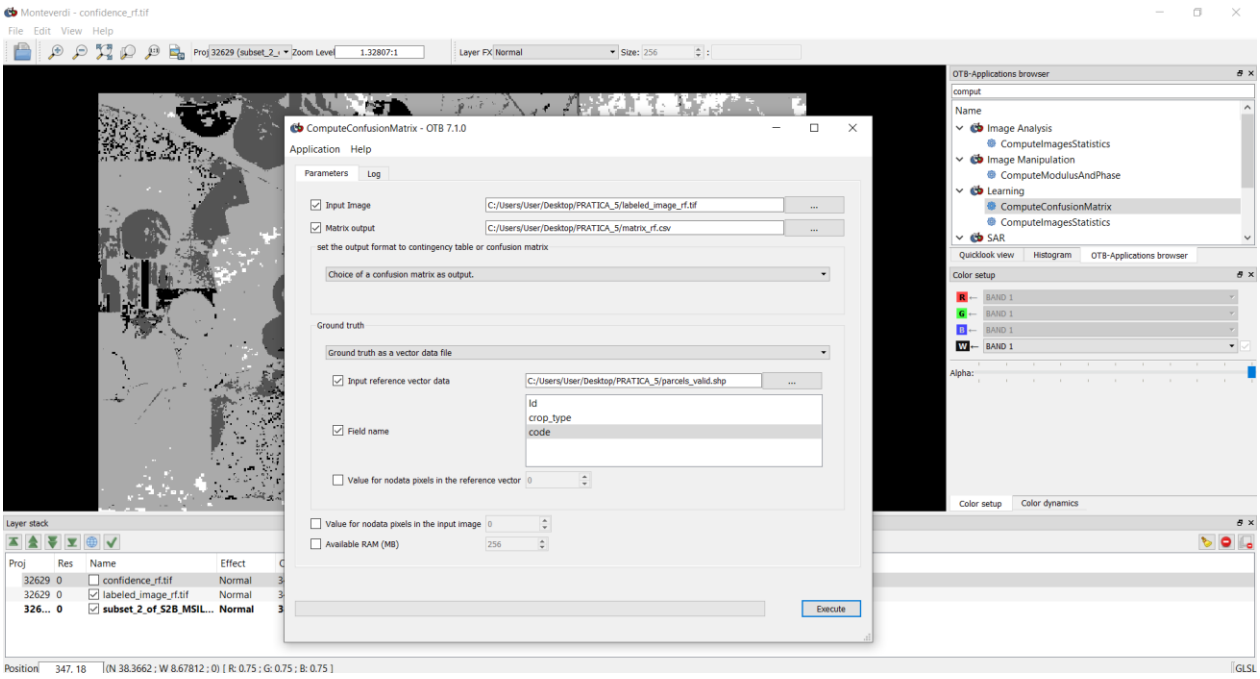
EXERCISE 6.1

a) Compute the confusion matrix of the results of the random forest classification performed on the previous practical lesson, using a polygon shapefile with the validation samples (parcels_valid.shp). This file contains sample corresponding to the same classes considered for the classification but at different locations. Download this file from OneDrive.

OTB- Applications Browser >> Learning >> ComputeConfusionMatrix
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. 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.
https://www.orfeo-toolbox.org/CookBook/Applications/app_ComputeConfusionMatrix.html

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

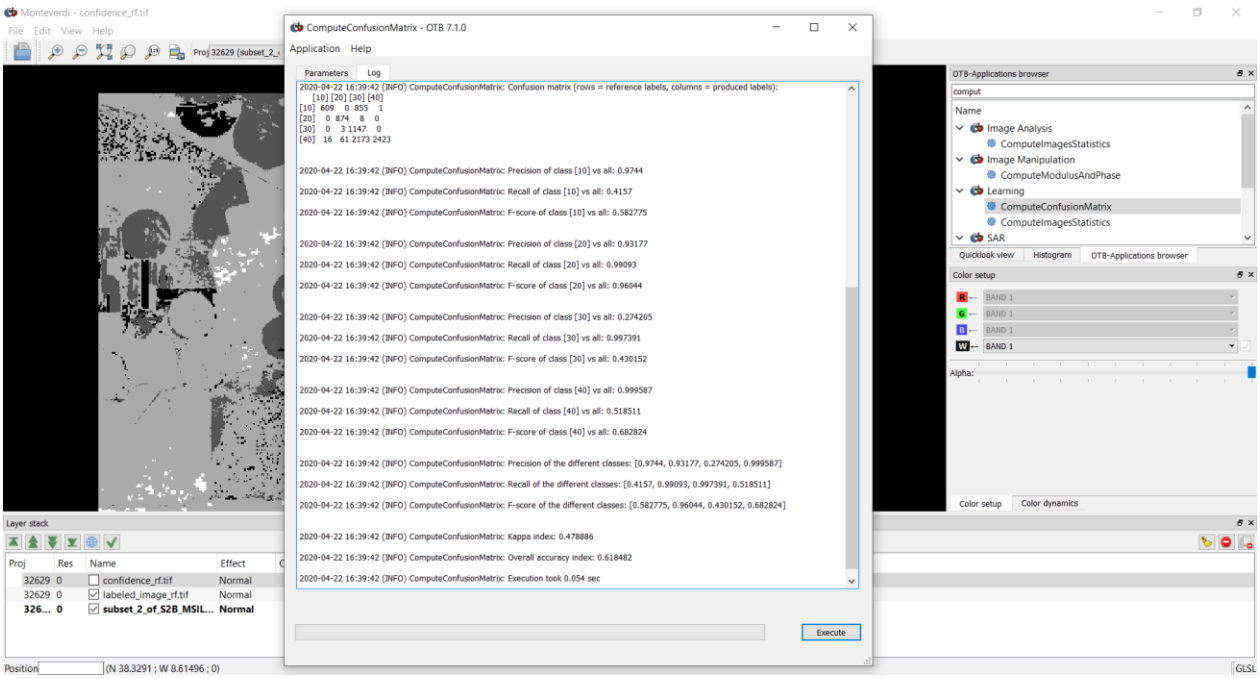
If you have made an unsupervised classification, it must be specified to the ComputeConfusionMatrix application. In this case, a contingency table has to be created rather than a confusion matrix.



The csv file where the output matrix is stored (use WordPad to open the matrix_rf.csv file).

```
#Reference labels (rows):10,20,30,40
#Produced labels (columns):10,20,30,40
609,0,855,1
0,874,8,0
0,3,1147,0
16,61,2173,2423
```

The confusion matrix as well as the overall accuracy, the kappa index, and the precision, recall and F1-score of each class are available in the Log window.



b) Fill the confusion matrix with the values in the Log window and in the matrix_rf.csv file.

Produced (map) labels

	Classes	10	20	30	40	Row Total	Recall	F1-score
Reference labels	**10**							
	20							
	30							
	40							
	Column Total							
	Precision							

Overall Accuracy=

Kappa=

10 – crop; 20 – bare soil; 30 – dry vegetation; 40 – forest