Sentinel-2 Agriculture

Design Definition File

Algorithm Theoretical Basis Document for L4 dynamic crop mask product

Milestone	Milestone 2
Version	1.0
Authors	CESBIO - Silvia Valero, David Morin UCL - Guadalupe Sepulcre, Nicolas Matton, Sophie Bontemps

This page is intentionally blank

Table of contents

List of figures

List of tables

1 Logical model

The classification process transform the L2A_R imagery into a 1/0 mask, 1 indicating the presence of cropland and 0 the presence of non-cropland.

The processing chain is run by tiles.

The processing chain can run in two different modes, depending if the user counts with field data for training the classifier or not:

- If the user has field data to train the algorithm, an extensive number of features related to the NDVI, the NDWI profiles and the brightness will be extracted from the imagery time series for feeding a supervised algorithm based on a Random Forest (RF) classifier;
- If there is no field data, training samples will be selected from an existing reference map (local/regional/global land cover map, crop mask, etc.) through an iterative trimming. These samples will be used to feed the same supervised algorithm than in the first case (i.e. based on a RF classifier) but the this classifier will here be fed by reflectance values coming from determinate moments of the growing cycle.

The principal steps of the classification process are schematized in Figure 1-1.

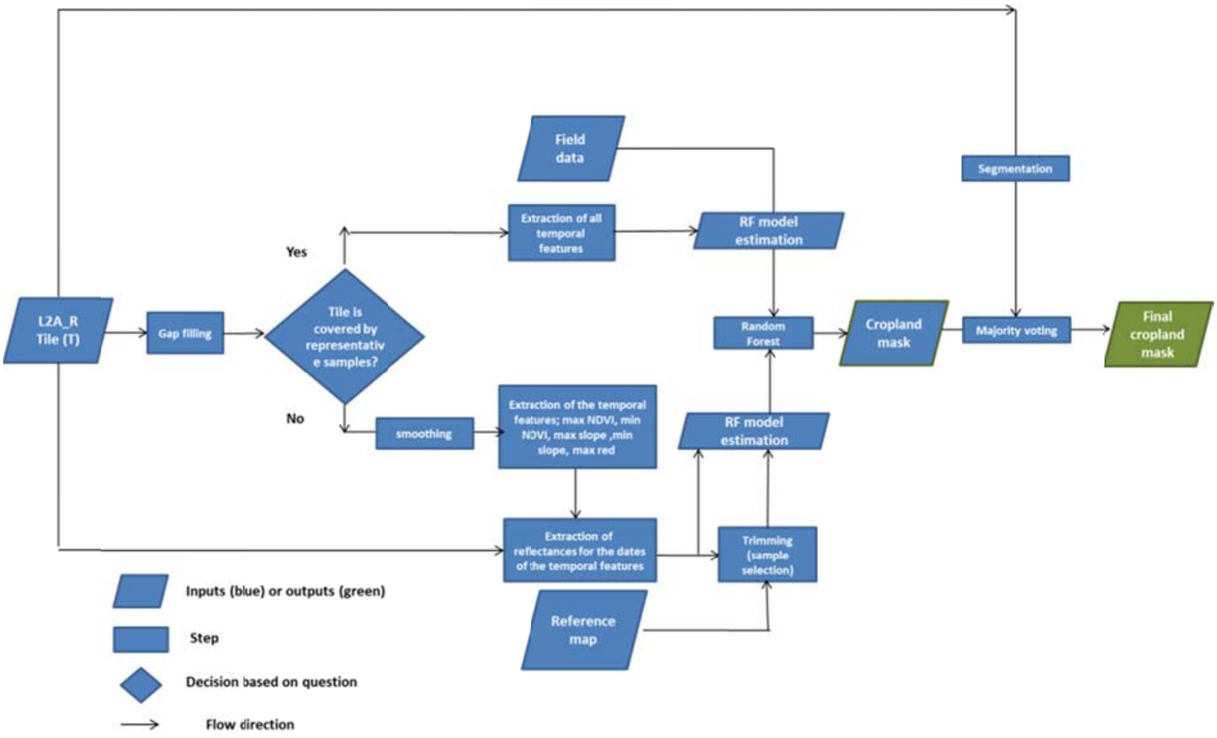

Field data
Segmentation
Extraction of all temporal features
RF model estimation
Yes
L2A_R Tile (T)
Gap filling
Tile is covered by representativ e samples?
Random Forest
Cropland mask
Majority voting
Final cropland mask
No
smoothing
Extraction of the temporal features; max NDVI, min NDVI, max slope ,min slope, max red
RF model estimation
Extraction of reflectances for the dates of the temporal features
Trimming (sample selection)
Inputs (blue) or outputs (green)
Step
Decision based on question
Flow direction
Reference map

Figure 1-1. Flowchart of the processing chain developed to generate the cropland mask

Several steps are common for the 2 modes:

- the gap filling, which uses a linear temporal interpolation of the pixels removed by the cloud mask;

- the supervised algorithm based on the RF classifier;
- the a posteriori filtering of the cropland mask.

The next sections of the document present in detail all the steps of the processing chain, distinguishing between the 2 modes and highlighting the common ones.

2 Common preliminary step: gap filling

The goal of the gap filling is to produce a reflectance image time series which is gap-filled with respect to missing data (identified thanks to the "validity masks" which contain clouds, cloud shadows and saturated pixels). The workflow of the gap filling is illustrated in Figure 3-3.

Original image time series
Gap filling
Gap filled image time series

Figure 2-1. Gap filling workflow

The gaps are filled by using the same linear interpolation method presented in Section 2.2.2 of the ATBD crop type document and is therefore no more recalled here. For this method, the cloud mask image times series needs to be used.

Input data: the original image times series

Output data: the gap filled image time series.

3 Mode based on in-situ data

The processing chain to run when field data exist is illustrated in Figure 3-1, which must be interpreted from left to right. This flowchart can be easily divided in three main tasks: the feature extraction, the classification and the post-filtering step. The post-filtering step is common with the "no field data module" and will thus be detailed later, in section 5. All other steps are detailed in this section.

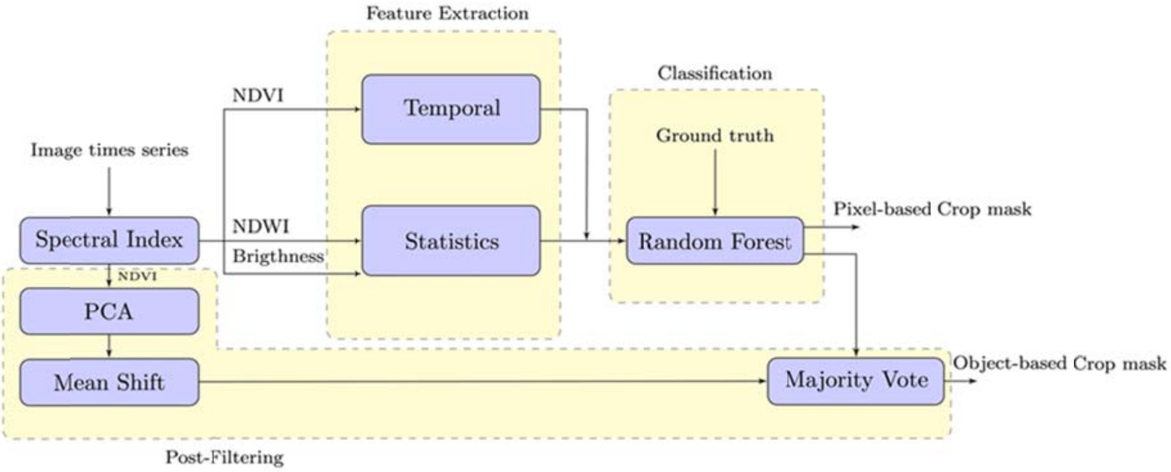


Figure 3-1. Flowchart describing the processing chain when field data exists

Input Data

The reflectance image times series that don't contain gaps (i.e. the output of the preliminary step detailed in 2).

As shown in Figure 3-1, the input data will be used in the first purple block whose name is *Spectral Index*. This task corresponds to the computation of three spectral indices: the NDVI, the NDWI and the Brightness index.

Output Data

The pixel-based crop mask product will be the output of the processing chain.

3.1 Pre-processing: spectral indices computation

The pre-processing workflow is illustrated in Figure 3-2, resulting in the three spectral indices: NDVI, NDWI and Brightness. This step is based on the algorithm presented in detail in Section 2.2.3 of the ATBD crop type document and is therefore no more recalled here.

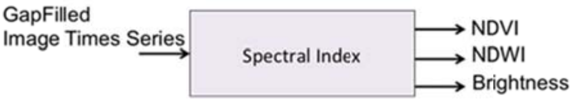


Figure 3-2. Pre-processing workflow, computing NDVI, NDWI and Brightness indices

3.2 Feature extraction step

Three different indices have been generated: the NDVI, the NDWI and the Brigthness. These indices are used in order to compute different features: temporal features (see sub-section 3.2.1) and statistic features (see sub-section 3.2.2).

17 temporal features are derived from the NDVI. Concerning the other two spectral indices, 5 statistics features are computed for each of them. In total, 27 features are computed, which are concatenated as a vector image (see sub-section 3.2.3) to be used in the subsequent classification step (that will be detailed in section 3.3).

3.2.1 Temporal features

The workflow developed to generate temporal features is illustrated in Figure 3-3.

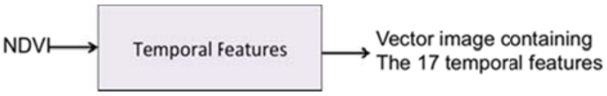


Figure 3-3. Temporal features generation workflow

<u>Input data:</u> the full gap filled NDVI times series

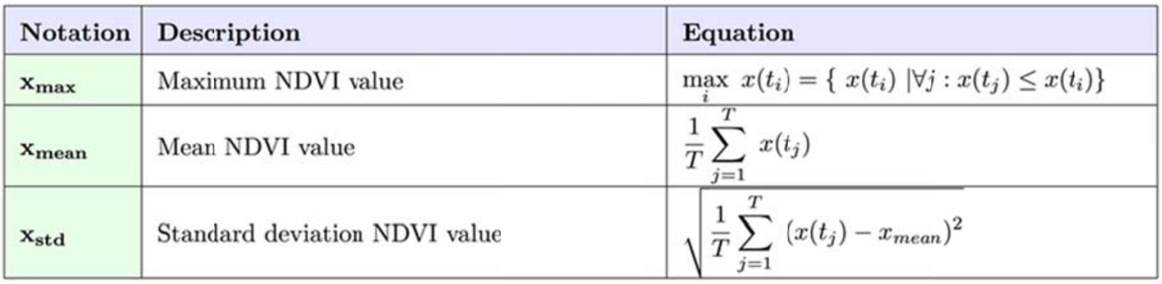
<u>Output data:</u> a vector image made of 17 channels, one for each temporal feature

<u>Available code</u>: a C++ code compatible with OTB has been developed.

The different features are described below.

- **Global features**

These features characterize the global trends on NDVI times series. The three features detailed Table 3-1 are computed.



Notation	Description	Equation
$\mathbf{x_{max}}$	Maximum NDVI value	$\max_i x(t_i) = \{ x(t_i) \mid \forall j : x(t_j) \leq x(t_i)\}$
$\mathbf{x_{mean}}$	Mean NDVI value	$\frac{1}{T}\sum_{j=1}^{T} x(t_j)$
$\mathbf{x_{std}}$	Standard deviation NDVI value	$\sqrt{\frac{1}{T}\sum_{j=1}^{T} (x(t_j) - x_{mean})^2}$

Table 3-1. Global features description

<u>Parameters:</u>

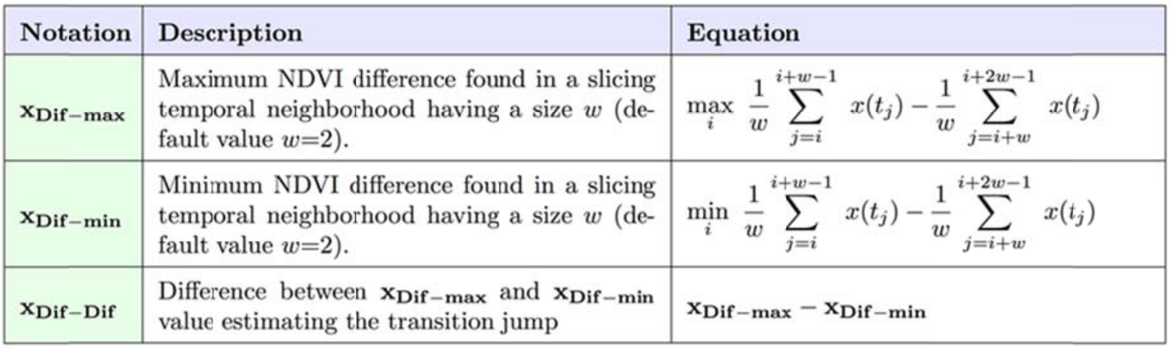
– x(ti) corresponds to the NDVI value at instant ti.

– T is the length of the NDVI times series

– mean is the mean NDVI value computed by using the T values

- **Features detecting largest local NDVI transitions**

These features describe the difference between consecutive windowed NDVI values. They are presented in Table 3-2. The goal is to detect the largest transitions corresponding to the greenness and the senescence onset periods.



Notation	Description	Equation
$\mathbf{x}_{\mathbf{Dif-max}}$	Maximum NDVI difference found in a slicing temporal neighborhood having a size w (default value w=2).	$\max_i \frac{1}{w}\sum_{j=i}^{i+w-1} x(t_j) - \frac{1}{w}\sum_{j=i+w}^{i+2w-1} x(t_j)$
$\mathbf{x}_{\mathbf{Dif-min}}$	Minimum NDVI difference found in a slicing temporal neighborhood having a size w (default value w=2).	$\min_i \frac{1}{w}\sum_{j=i}^{i+w-1} x(t_j) - \frac{1}{w}\sum_{j=i+w}^{i+2w-1} x(t_j)$
$\mathbf{x}_{\mathbf{Dif-Dif}}$	Difference between $\mathbf{x}_{\mathbf{Dif-max}}$ and $\mathbf{x}_{\mathbf{Dif-min}}$ value estimating the transition jump	$\mathbf{x}_{\mathbf{Dif-max}} - \mathbf{x}_{\mathbf{Dif-min}}$

Table 3-2. Description of features detecting largest local NDVI transitions

Parameters:

- x(ti) and x(tj) correspond to the NDVI values at instants ti and tj.
- w is the size of the slicing temporal window used to compute the differences

- **mbiFeatures associated to the maximum NDVI value:**

These features are associated to the area containing the maximum NDVI value, as identified in Figure 3-4. The features are described in Table 3-3.

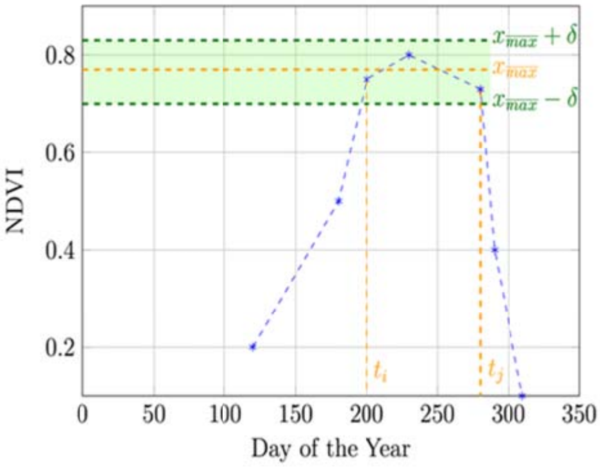
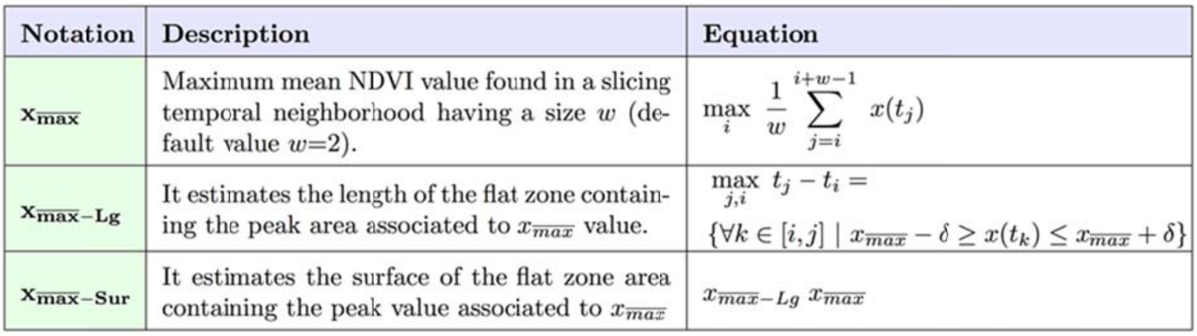


Figure 3-4. Maximum NDVI values identification from a full NDVI time series



Notation	Description	Equation
$\mathbf{x_{max}}$	Maximum mean NDVI value found in a slicing temporal neighborhood having a size w (default value w=2).	$\max\limits_{i} \frac{1}{w} \sum_{j=i}^{i+w-1} x(t_j)$
$\mathbf{x_{max-Lg}}$	It estimates the length of the flat zone containing the peak area associated to x_{max} value.	$\max\limits_{j,i} t_j - t_i =$ $\{\forall k \in [i,j] \mid x_{max} - \delta \geq x(t_k) \leq x_{max} + \delta\}$
$\mathbf{x_{max-Sur}}$	It estimates the surface of the flat zone area containing the peak value associated to x_{max}	$x_{max-Lg}\ x_{max}$

Table 3-3. Maximum NDVI features description

Parameters:

- x(ti) and x(tj) correspond to the NDVI values at instants ti and tj.
- w is the size of the slicing temporal window used to compute the differences
- delta δ parameter is a user threshold, whose default value is equal to 0.05. In order to estimate the surface, a rectangular shape has been considered.

- **Features associated to the greenness onset**

These features are associated to the largest increasing period of the NDVI function. As it can be seen in Figure 3-5, the greenness onset period is highlighted in orange. These temporal features are subject to the next constraint :

$$\{\forall k \in [i,j] \mid x'(t_k) \geq 0\}$$

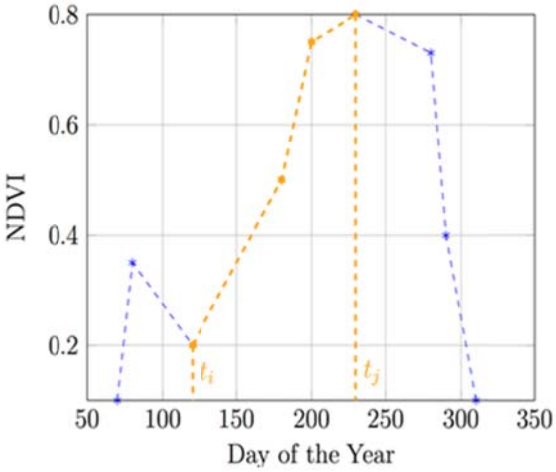
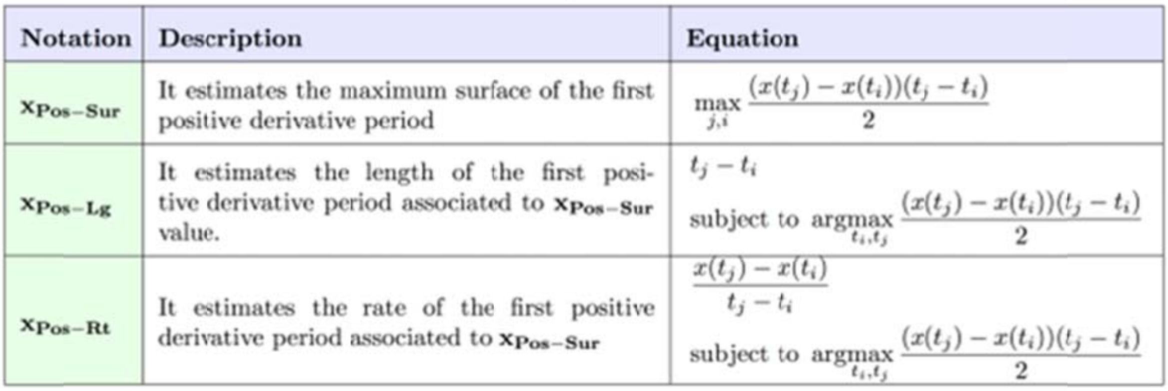


Figure 3-5. Identification of the largest increasing period in a full NDVI time series

Table 3-4 describes the features associated to the greenness period. In order to estimate the surface, a triangular shape has been considered.



Notation	Description	Equation
$\mathbf{x}_{\mathbf{Pos-Sur}}$	It estimates the maximum surface of the first positive derivative period	$\max\limits_{j,i} \frac{(x(t_j)-x(t_i))(t_j-t_i)}{2}$
$\mathbf{x}_{\mathbf{Pos-Lg}}$	It estimates the length of the first positive derivative period associated to $\mathbf{x}_{\mathbf{Pos-Sur}}$ value.	$t_j - t_i$ subject to $\underset{t_i,t_j}{\operatorname{argmax}} \frac{(x(t_j)-x(t_i))(t_j-t_i)}{2}$
$\mathbf{x}_{\mathbf{Pos-Rt}}$	It estimates the rate of the first positive derivative period associated to $\mathbf{x}_{\mathbf{Pos-Sur}}$	$\frac{x(t_j)-x(t_i)}{t_j-t_i}$ subject to $\underset{t_i,t_j}{\operatorname{argmax}} \frac{(x(t_j)-x(t_i))(t_j-t_i)}{2}$

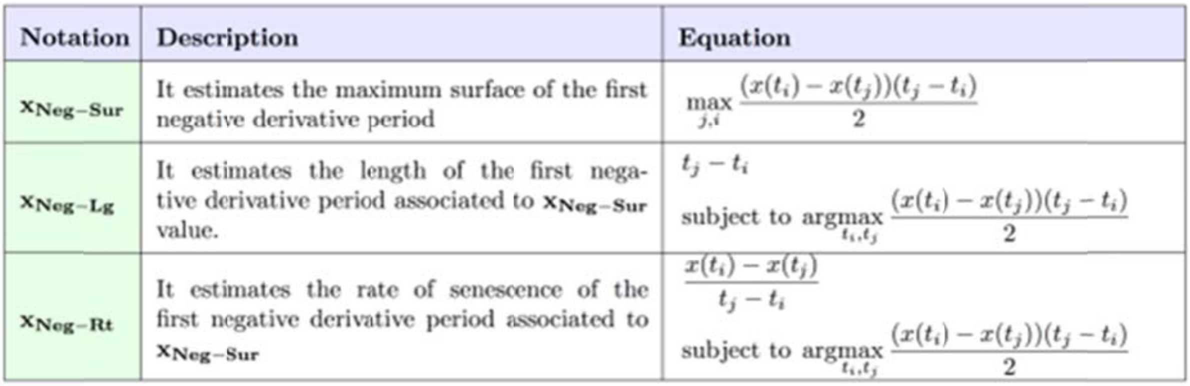
Table 3-4. Greenness period features description

Parameters:

– x(ti) and x(tj) correspond to the NDVI values at instants ti and tj.

- **Features associated to the senescence onset:**

These features are associated to the largest decreasing period of the NDVI function. They are presented in Table 3-5



Notation	Description	Equation
$\mathbf{x}_{\mathbf{Neg-Sur}}$	It estimates the maximum surface of the first negative derivative period	$\max\limits_{j,i} \frac{(x(t_i)-x(t_j))(t_j-t_i)}{2}$
$\mathbf{x}_{\mathbf{Neg-Lg}}$	It estimates the length of the first negative derivative period associated to $\mathbf{x}_{\mathbf{Neg-Sur}}$ value.	$t_j - t_i$ subject to $\underset{t_i,t_j}{\operatorname{argmax}} \frac{(x(t_i)-x(t_j))(t_j-t_i)}{2}$
$\mathbf{x}_{\mathbf{Neg-Rt}}$	It estimates the rate of senescence of the first negative derivative period associated to $\mathbf{x}_{\mathbf{Neg-Sur}}$	$\frac{x(t_i)-x(t_j)}{t_j-t_i}$ subject to $\underset{t_i,t_j}{\operatorname{argmax}} \frac{(x(t_i)-x(t_j))(t_j-t_i)}{2}$

Table 3-5. Senescence onset features description

Parameters:

– x(ti) and x(tj) correspond to the NDVI values at instants ti and tj.

- **Features characterizing bare soil transitions**

These features detect if there is a transition between the bare soil before the greenness onset or after the senescence onset. Bare soil transition is illustrated in Figure 3-6 and corresponding features are given in Table 3-6. The resulting value is 0 or 1.

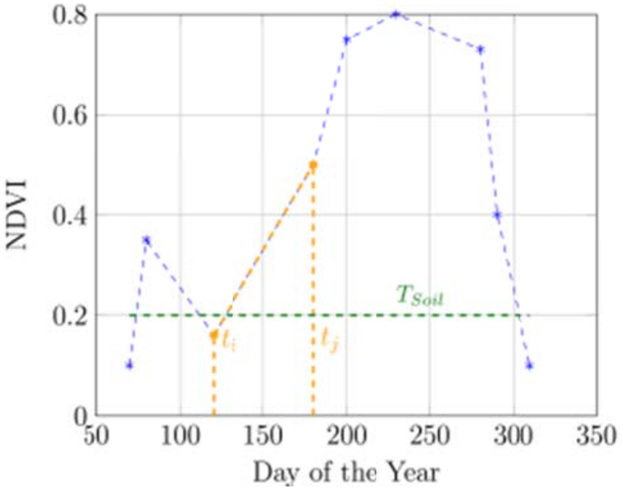
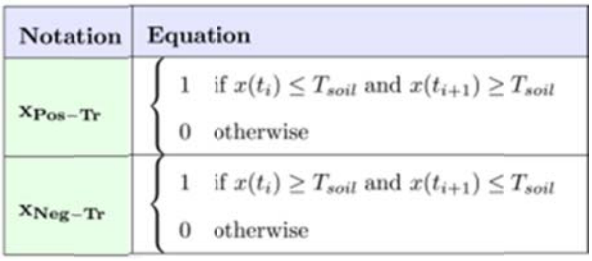


Figure 3-6. Identification of bare soil transitions in a full NDVI time series



Notation	Equation
$x_{\mathbf{Pos-Tr}}$	$\begin{cases} 1 & \text{if } x(t_i) \leq T_{soil} \text{ and } x(t_{i+1}) \geq T_{soil} \\ 0 & \text{otherwise} \end{cases}$
$x_{\mathbf{Neg-Tr}}$	$\begin{cases} 1 & \text{if } x(t_i) \geq T_{soil} \text{ and } x(t_{i+1}) \leq T_{soil} \\ 0 & \text{otherwise} \end{cases}$

Table 3-6. Bare soil transition features description

Parameters:

- x(ti) corresponds to the NDVI values at instant ti
- T_soil is a threshold defining non cultivated areas

3.2.2 Statistic features

The workflow developed to generate statistic features is illustrated in Figure 3-7.

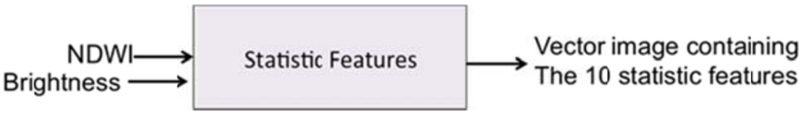


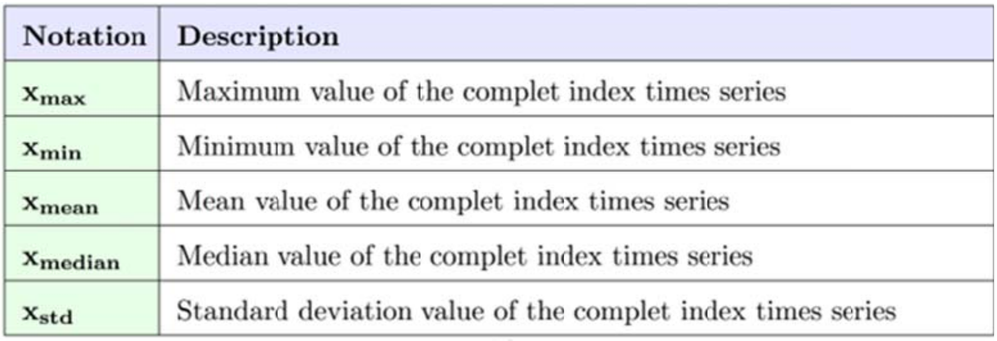
Figure 3-7. Statistic features generation workflow

Input data: the complete Brightness and NDWI image times series

Output data : a vector image made of 10 channels. These 10 channels correspond to 5 statistic features computed on Brightness and 5 statistic features computed on NDWI

Available code : A C++ code compatible with OTB has been developed.

As explained before, the statistic features correspond to a set of 5 global features. These 5 features are presented in Table 3-7. They are computed for Brightness and NDWI indices.



Notation	Description
x_{max}	Maximum value of the complet index times series
x_{min}	Minimum value of the complet index times series
x_{mean}	Mean value of the complet index times series
x_{median}	Median value of the complet index times series
x_{std}	Standard deviation value of the complet index times series

Table 3-7. Statistic features description

3.2.3 Feature image concatenation

The complete set of temporal and statistic features are then concatenated as shown in Figure 3-8. The resulting feature vector image is the output of the "feature extraction" step and will be used as a input for the next step : the classification.

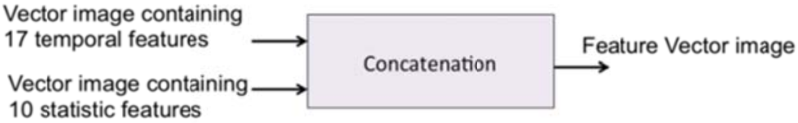


Figure 3-8. Workflow of the temporal and statistic features concatenation

Concatenation can be done using the next OTB application:

```
otbcli_ConcatenateImages -il Temporal_Features.tif Statistic_Features.tif
-out Feature_Concatenate_Images.tif
```

3.3 Classification step

The classification step can be divided in two subtasks:

1. The construction of the classification model
2. The use of the classification model in the complete image times series and its evaluation

3.3.1 Constructing the classification model

The workflow underlying the construction of the classification model is illustrated in Figure 3-9. It is made of 3 main tasks: splitting polygons for training and testing, randomly selecting training pixel samples and random forest training. They are detailed in the above sub-sections.

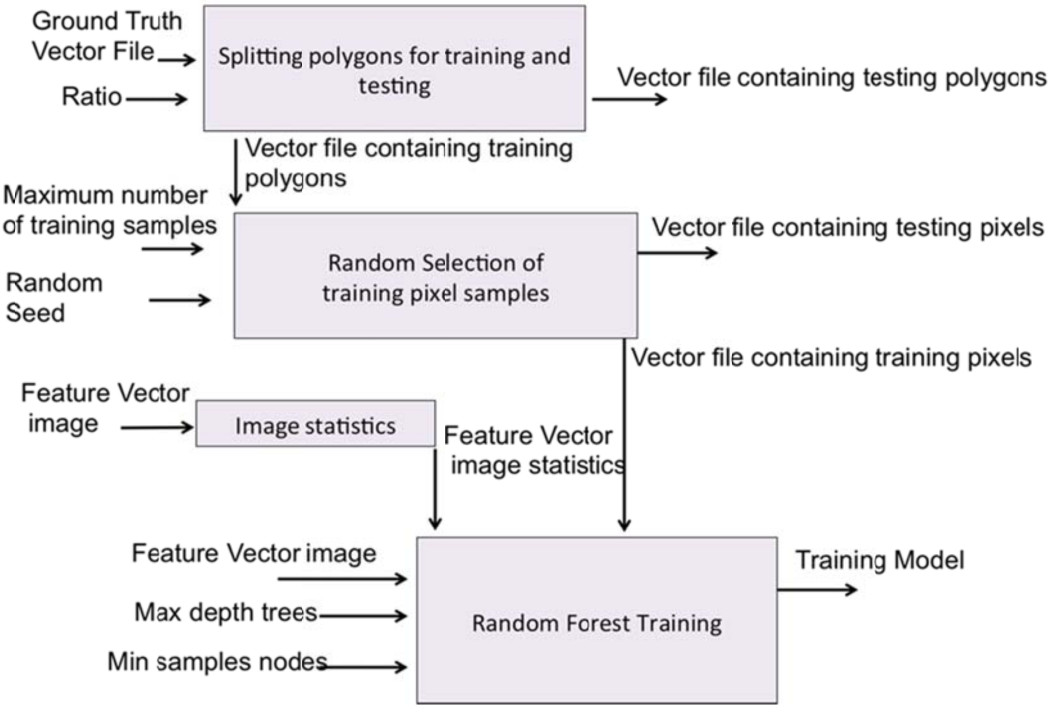

Ground Truth Vector File
Ratio
Splitting polygons for training and testing
Vector file containing testing polygons
Vector file containing training polygons
Maximum number of training samples
Random Seed
Random Selection of training pixel samples
Vector file containing testing pixels
Vector file containing training pixels
Feature Vector image
Image statistics
Feature Vector image statistics
Feature Vector image
Max depth trees
Min samples nodes
Random Forest Training
Training Model

Figure 3-9. Workflow of the classification model construction

3.3.1.1 Splitting polygons for training and testing

This first task consists in splitting the in-situ dataset into 2 disjoint subsets: the training set and the validation set (Figure 3-10).

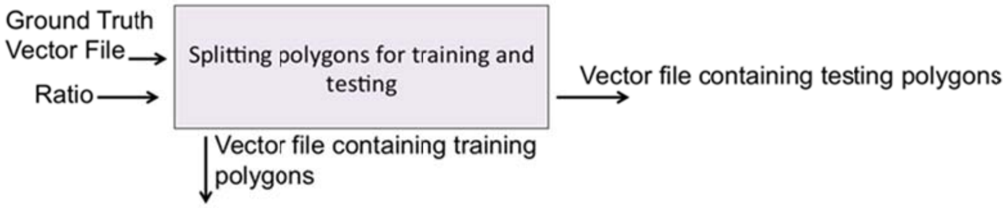

Ground Truth Vector File
Ratio
Splitting polygons for training and testing
Vector file containing testing polygons
Vector file containing training polygons

Figure 3-10. Workflow of the in-situ dataset splitting sub-task

Input data:

Ground Truth Vector File – vector file containing in-situ data

Ratio – parameter expressing the ratio between the number of training and validation polygons per class (corn, soybean, rice, no-crop, etc.)

Output data

Vector file containing training polygons – a vector file containing in-situ data for training.

Vector file containing testing polygons – a vector file containing in-situ data for validation.

The output sets are made of polygons and not individual pixels. The algorithm performing this task is explained in Section 2.2.1 of the ATBD crop type document and is therefore no more recalled here. An illustration is provided in Figure 3-11, with a ratio of 0.5 between the 2 subsets.

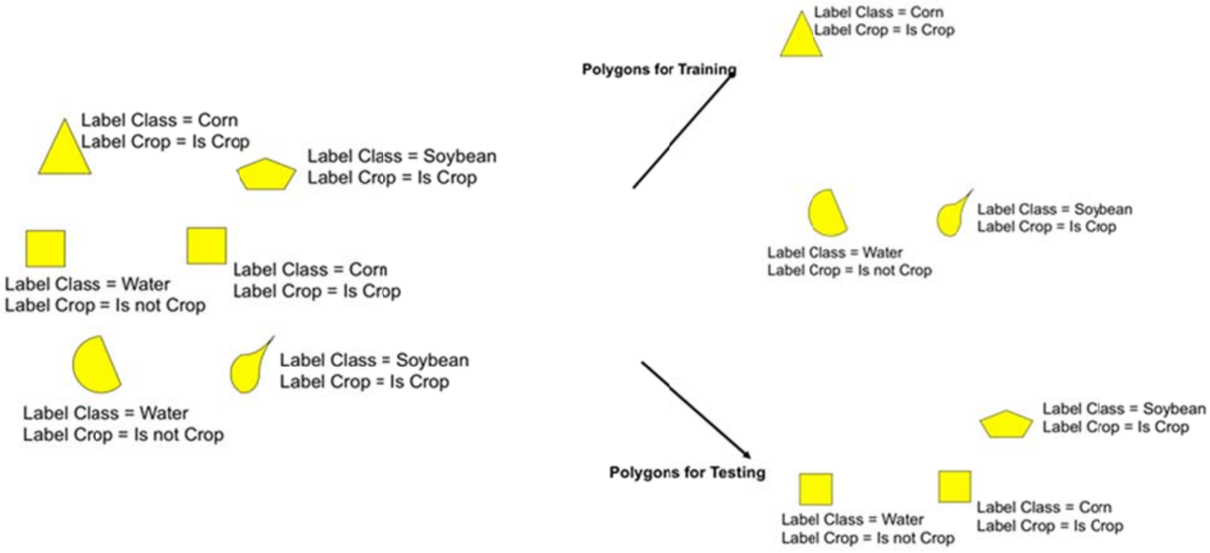
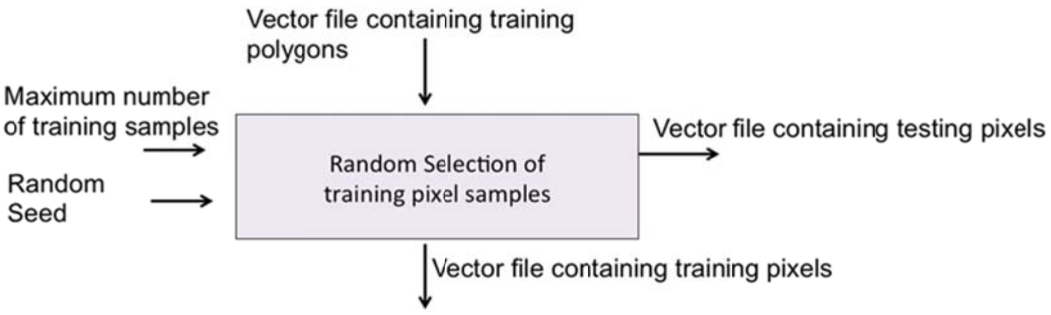


Figure 3-11. Illustration of the in-situ dataset split into two subsets for training and validation using a ration equal to 0.5

3.3.1.2 Random selection of training pixel samples : Crops and Not Crops

This step consists in randomly selecting a number of samples (i.e. pixels) for each class : “crop” and “no-crop”. The workflow is illustrated in Figure 3-12. The output selection will be class consistent, meaning that if 33% of crops belong to soybean class, 33% of the crop class training samples will also belong to the soybean class.

This reduction of the training sample size is needed to reduce the time complexity of the RF classifier. If it is possible from the operational point of view, this step could be removed and the set of the training pixels samples will correspond to the set of the training polygon samples.



Vector file containing training polygons
Maximum number of training samples
Random Seed
Random Selection of training pixel samples
Vector file containing testing pixels
Vector file containing training pixels

Figure 3-12. Workflow for reducing the size of the training sample

Input data

Vector file containing training polygons – vector file containing in-situ data for training obtained by the previous splitting algorithm (section 3.3.1.1). It must be remarked that the polygons of this vector file as associated with two kinds of label: class label and crop label.

For instance, a polygon will be composed by the class label “corn” and the crop label “Is a crop”.

Maximum number of training samples – the second input data corresponds to the limited number of training samples that will be used for training. For instance, if this number is equal to 1000, it means that the output data will be made of 1000 crop and 1000 no crop samples.

Random seed – the third parameter is the random seed that will be used to do the random selection of the pixels. If the third parameter is not specified, a random seed will be generated automatically.

Output data

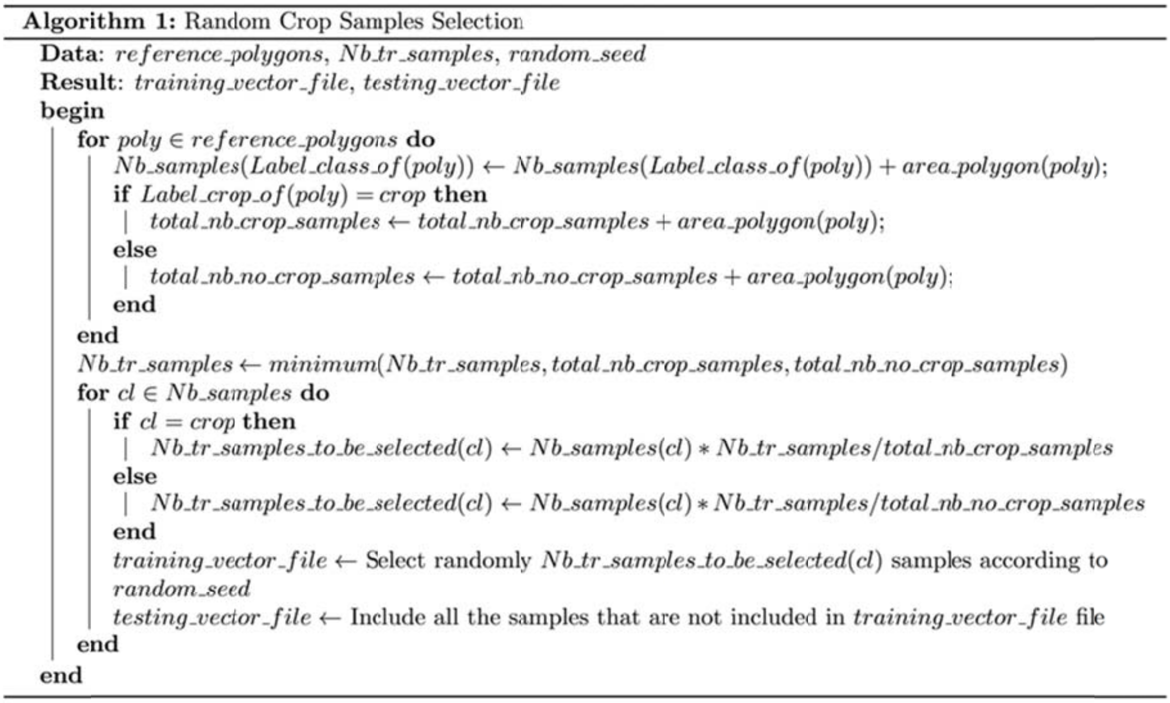
Vector file containing testing pixels – vector file containing the limited in-situ data for training. Polygons will be associated with one label: crop (with value 1) or no-crop (with value 0)

Vector file containing training pixels – vector file containing the data from Reference_polygons which are not included in the training_vector_file. Polygons will be associated with one label: crop (with value 1) or no-crop (with value 0)

Available code

A C++ code compatible with OTB has been developed. This code is inspired from the OTB class “*otbListSampleGenerator*”.

Algorithm



```
Algorithm 1: Random Crop Samples Selection
Data: $reference\_polygons$, $Nb\_tr\_samples$, $random\_seed$
Result: $training\_vector\_file$, $testing\_vector\_file$
begin
    for $poly \in reference\_polygons$ do
        $Nb\_samples(Label\_class\_of(poly)) \leftarrow Nb\_samples(Label\_class\_of(poly)) + area\_polygon(poly)$;
        if $Label\_crop\_of(poly) = crop$ then
            $total\_nb\_crop\_samples \leftarrow total\_nb\_crop\_samples + area\_polygon(poly)$;
        else
            $total\_nb\_no\_crop\_samples \leftarrow total\_nb\_no\_crop\_samples + area\_polygon(poly)$;
        end
    end
    $Nb\_tr\_samples \leftarrow minimum(Nb\_tr\_samples, total\_nb\_crop\_samples, total\_nb\_no\_crop\_samples)$
    for $cl \in Nb\_samples$ do
        if $cl = crop$ then
            $Nb\_tr\_samples\_to\_be\_selected(cl) \leftarrow Nb\_samples(cl) * Nb\_tr\_samples/total\_nb\_crop\_samples$
        else
            $Nb\_tr\_samples\_to\_be\_selected(cl) \leftarrow Nb\_samples(cl) * Nb\_tr\_samples/total\_nb\_no\_crop\_samples$
        end
        $training\_vector\_file \leftarrow$ Select randomly $Nb\_tr\_samples\_to\_be\_selected(cl)$ samples according to $random\_seed$
        $testing\_vector\_file \leftarrow$ Include all the samples that are not included in $training\_vector\_file$ file
    end
end
```

3.3.1.3 Random Forest Training step

The training of the classifier step is illustrated in Figure 3-13.

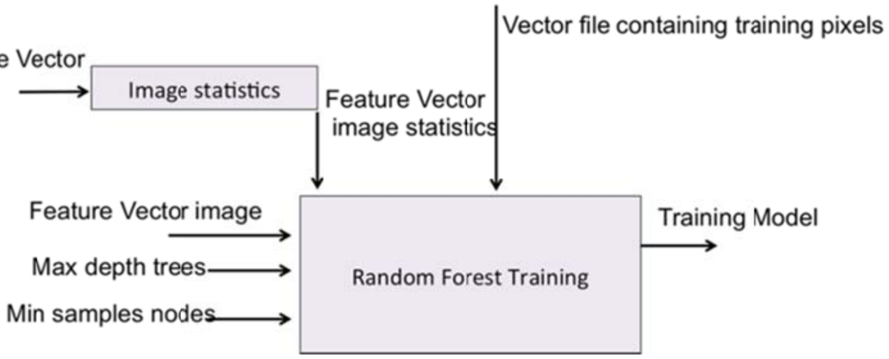


Figure 3-13. Workflow of the classifier training

This task has been developed on C++ and it is based on the "*otbcli_TrainImagesClassifier*" application. The OTB application is composed of two steps:

1. Random selection of training and testing samples;
2. Training of the classifier by using the previous training samples.

In our case, the splitting task is done by using the algorithm presented in the previous section (3.3.1.2). For this reason, the developed code implemented in C++ and compatible with OTB Library focuses on the second step.

Input data

Feature Vector Image – vector image containing the 27 temporal and statistic features

Feature Vector Image Statistics – the mean and the variance of each feature composing the feature vector image. This result can be obtained by applying the OTB application presented in 2.3.1 of the ATBD crop type document.

Max depth tree – maximum depth of the trees used for Random Forest classifier

Min samples nodes – minimum number of samples in each node used by the classifier.

Vector File containing training pixels – vector file containing the training pixels

Output data

Training model – resulting RF model to be used in the classification step.

3.3.2 Classification of the image times series and classifier evaluation

3.3.2.1 Random Forest classification step

This step consists in the classification of all the pixels of the image times series made of all the statistic and temporal features (see section 3.2). The general workflow is illustrated in Figure 3-14.

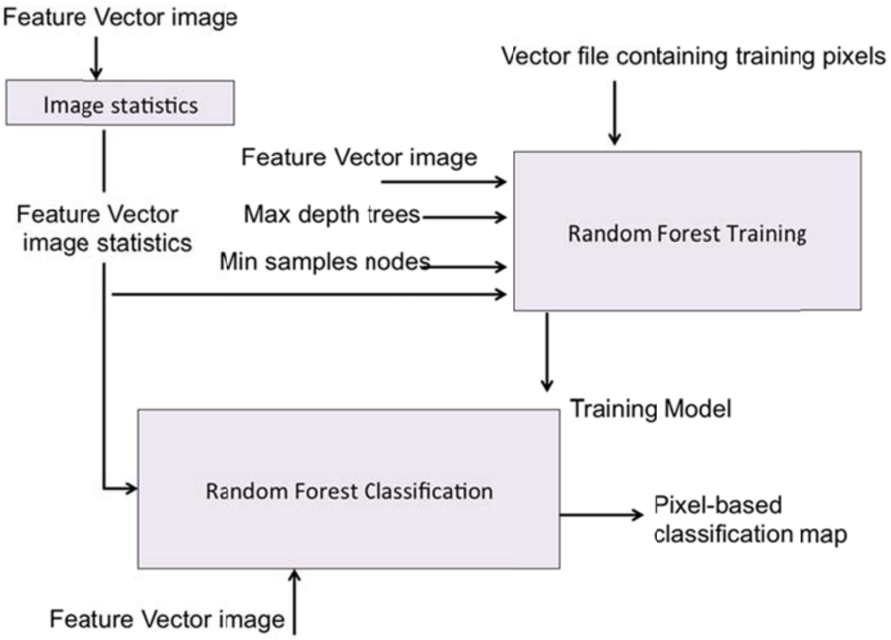


Figure 3-14. Workflow of the RF classification

<u>Input data</u>

Feature Vector Image – vector image containing the 27 temporal and statistic features

Feature Vector Image Statistics – the mean and the variance of each feature composing the feature vector image.

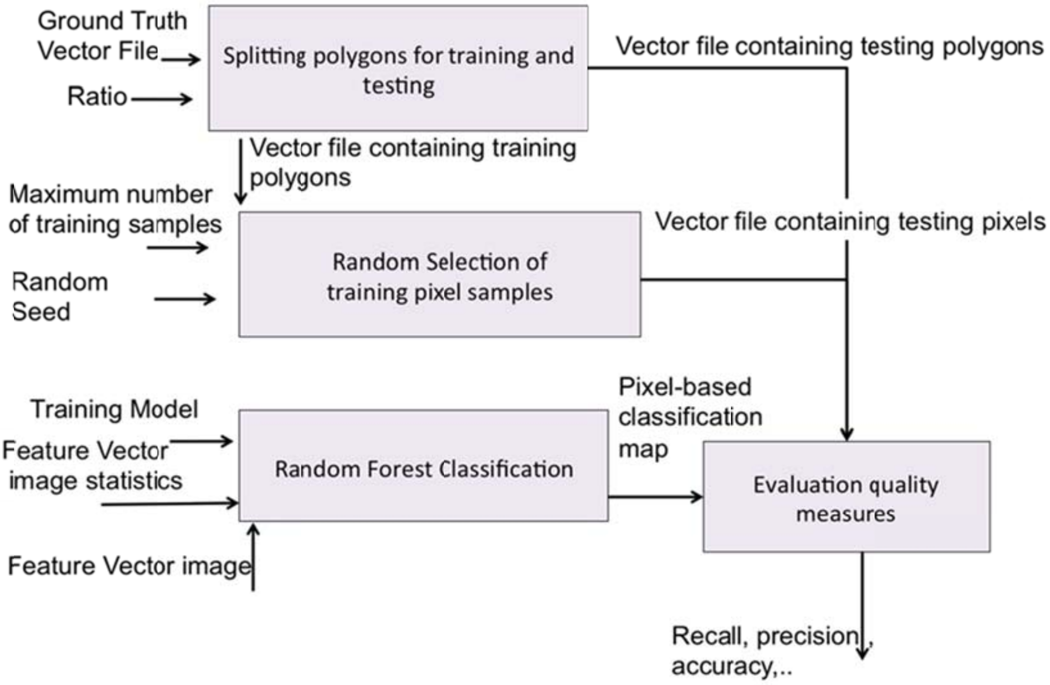
Training model – resulting Random Forest model obtaining from the training step.

<u>Output data</u>

Pixel-based classification map – resulting crop mask product.

3.3.2.2 Classifier Evaluation: Computing evaluation quality measures

The classification result will be evaluated using all the vector files which contain the testing pixels and polygons which have not been used for the training (Figure 3-15).



Ground Truth Vector File
Ratio
Splitting polygons for training and testing
Vector file containing testing polygons
Vector file containing training polygons
Maximum number of training samples
Random Seed
Random Selection of training pixel samples
Vector file containing testing pixels
Training Model
Feature Vector image statistics
Random Forest Classification
Feature Vector image
Pixel-based classification map
Evaluation quality measures
Recall, precision, accuracy,..

Figure 3-15. Workflow of the classification assessment

The evaluation quality indices (recall, precision, accuracy, etc.) that are computed are detailed in crop mask chapter of the Design Justification File.

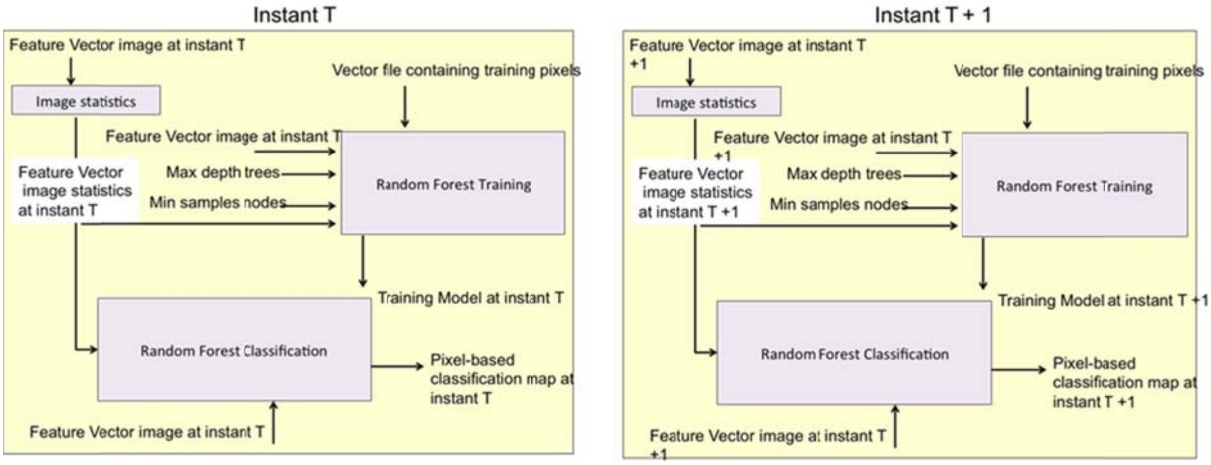
All these indices are derived from a confusion matrix result. This confusion matrix can be computed by using the OTB application "*otbcli_ComputeConfusionMatrix*", as it is done for the crop type product (see Section 2.4.2 of the ATBD crop type document).

3.4 Processing chain working in the iterative operational model

The processing chain detailed in the section 3.3 (i.e. the classification step) will have to run in an operational mode in which the cropland mask will be obtained iteratively. Two different cases must be taking into account regarding the satellite acquisitions: the first year of Sentinel-2 images and the other years. These two situations are analyzed here after.

3.4.1 The first year of Sentinel-2 images

The "in-situ processing chain" works if it exists ground truth data from the current year. Satellite data captured from the 6 first months will be necessary to produce the first cropland mask product. Considering that the first cropland map will be constructed at instant T, Figure 3-16 shows how the cropland product will be computed in an iterative way for the following instants of time.



Instant T
Feature Vector image at instant T
Vector file containing training pixels
Image statistics
Feature Vector image at instant T
Feature Vector image statistics at instant T
Max depth trees
Min samples nodes
Random Forest Training
Training Model at instant T
Random Forest Classification
Pixel-based classification map at instant T
Feature Vector image at instant T
Instant T + 1
Feature Vector image at instant T +1
Vector file containing training pixels
Image statistics
Feature Vector image at instant T +1
Feature Vector image statistics at instant T +1
Max depth trees
Min samples nodes
Random Forest Training
Training Model at instant T +1
Random Forest Classification
Pixel-based classification map at instant T +1
Feature Vector image at instant T +1

Figure 3-16. “In-situ” classification chain running at instants T and T+1, when we are during the first S2 year

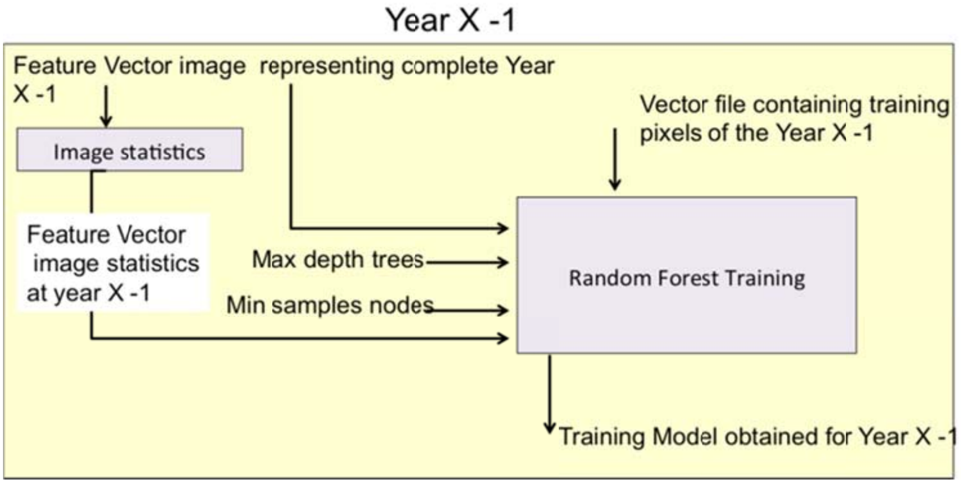
The right figure shows how since the instant T, the feature image will be recomputed at each instant of time. Besides, this image will be used: (i) to compute the image statistics, (ii) to train the classifier and (iii) to classify the complete image.

3.4.2 The rest of the years

For the rest of the years, some considerations have been done:

1. A complete year of data acquisitions exists from the previous year;
2. The ground truth in-situ data exists from the previous year;
3. The satellite data of the current year have been interpolated, in order to have the same acquisition dates than during the previous year.

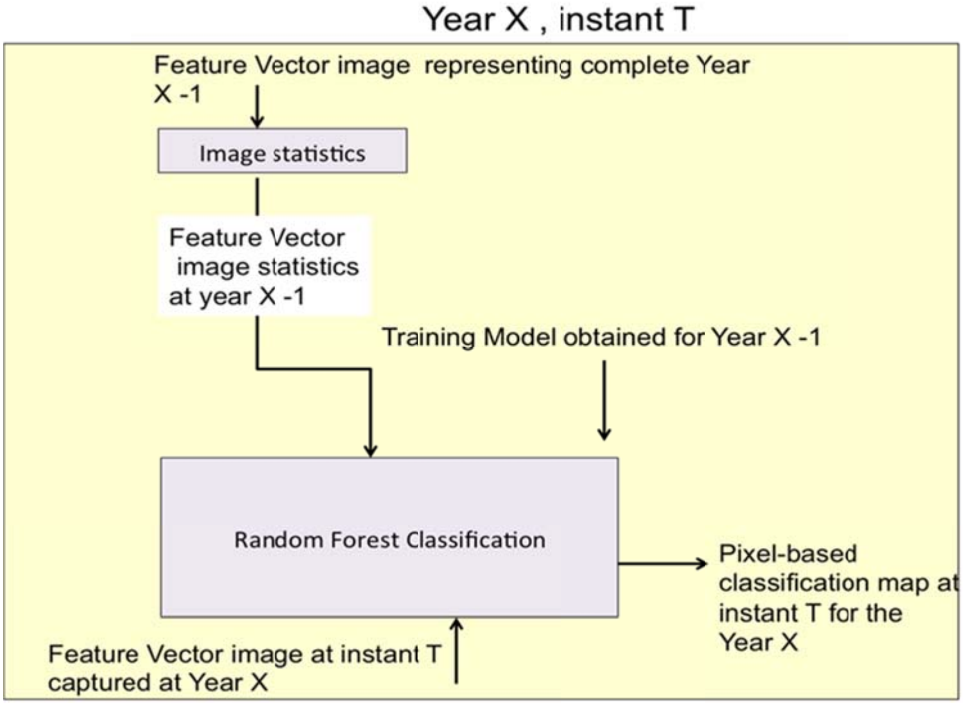
According to these last assumptions, a classifier model is computed for the previous year. This allows developing the system presented in Figure 3-17 and Figure 3-18. In these figures, Year X-1 corresponds to the previous year, whereas Year X corresponds to the current year.



Year X -1
Feature Vector image representing complete Year X -1
Vector file containing training pixels of the Year X -1
Image statistics
Feature Vector image statistics at year X -1
Max depth trees
Min samples nodes
Random Forest Training
Training Model obtained for Year X -1

Figure 3-17. “In-situ” classification chain running in Year X-1, when we are not during the first S2 year

By using the classifier model obtained for the previous year (i.e. built on the in-situ data and the images from the previous years), the classification of the current year will be done, after each new acquisition captured at instant **T**, as illustrated in Figure 3-18.

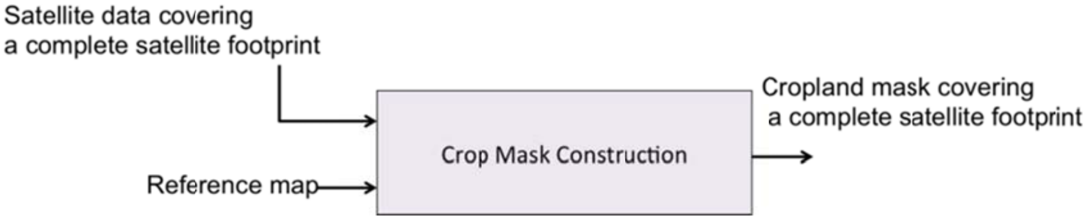


Year X , instant T
Feature Vector image representing complete Year X -1
Image statistics
Feature Vector image statistics at year X -1
Training Model obtained for Year X -1
Random Forest Classification
Pixel-based classification map at instant T for the Year X
Feature Vector image at instant T captured at Year X

Figure 3-18. “In-situ” classification chain running in Year X, when we are not during the first S2 year

3.5 Processing chain model working for a large area covering several satellite footprints

When working large areas, as shown in Figure 3-19, the construction of the crop mask will need an additional input variable, which has been namely *Reference map*. This map corresponds to an image covering the satellite footprint. In it, the values of the pixels correspond to the classifier model that must be used to classify the satellite footprint.



Satellite data covering a complete satellite footprint
Reference map
Crop Mask Construction
Cropland mask covering a complete satellite footprint

Figure 3-19. Crop mask production logical workflow when working over large areas

For instance, if all pixels of the reference map have a value equal to 1, it means that the classifier model number 1 will be used in order to classify the complete footprint. The classifier model number 1 can be for example a classifier model obtained from a neighborhood footprint. When the reference map has pixels associated with different values, it means that different models will be used for classifying the footprint.

It can be noted that this approach can also contribute to solve the problem where in-situ data is only available in one satellite footprint.

4 Mode without in-situ data

4.1 Preliminary step: data smoothing

A Whitaker filter is applied to the time series, both on the reflectance bands and on the NDVI (Figure 4-1). The filter is applied using the R function “*Whit1*”. Whit1 is a function from the “ptw” R package, which is a weighted Whittaker smoothing with a first order finite difference penalty. The manual can be found at http://cran.r-project.org/web/packages/ptw/ptw.pdf.

Figure 4-1. Workflow of the smoothing preliminary step

Input

There are the outputs of the previous gap filling step (section 2):

band<n>_gapfilled – gap filled image time series (reflectance values) - output of preliminary step (section 2)

NDVI_gapfilled – gap filled NDVI time series

From the first raster, the layers corresponding to each spectral band are selected and put together. This information, as well as the NDVI raster, will be the input of the smoothing algorithm.

The smoothing is applied separately for each time series (4 spectral channels and NDVI). The time series is given as a vector.

Output

band<n>_smoothed – raster files including the smoothed reflectance values for all the dates

NDVI_smoothed – raster files including the smoothed NDVI values for all the dates

Each raster file will have a number of bands equal to the number of image acquisition dates.

Algorithm parameters

The parameters of the Whit1 function are given in Table 4-1.

Table 4-1. Parameters of the Whit1 function.

Parameters	Description	Value
Lambda	Smoothing parameter (larger values lead to more smoothing)	2
w	Weights (a vector of same length as the time series to be smoothed). Default weights are equal to one signal to be smoothed	1

Pseudo-code representation[1]

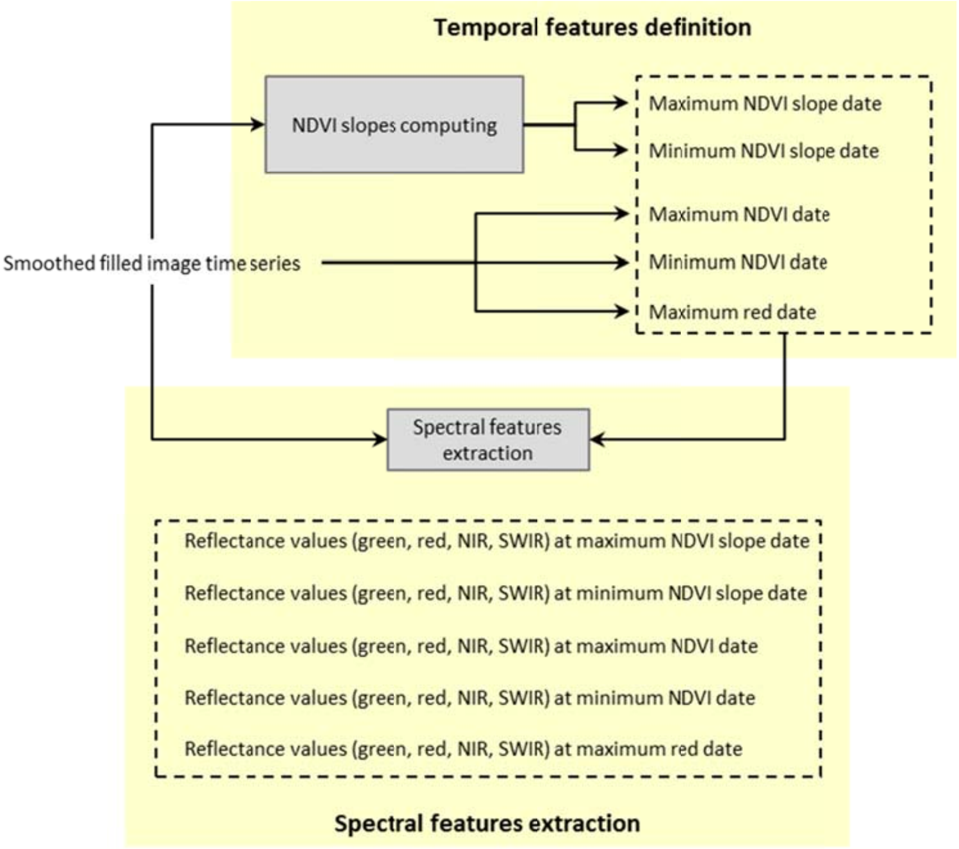
```
Algorithm "smoothing" is

Temporary output:
n raster files, n corresponding to the different spectral bands
        band_<n>

## Creating separated files per spectral band ####
For each of the n spectral bands:
        select the layers in the file band<n>_gapfilled corresponding to the spectral band n
        store and save these layers in band<n>
## Creating the smoothed temporal series of the different spectral bands####
For each of the n files corresponding to the n spectral bands:
        Create the smooth reflectance values, "out_band", by applying the R whit1 function
        with the following parameter:
                whit1(band<n>, 2)
        Save the output "out_band" in  band<n>_smoothed

## Creating a smoothed temporal series of NDVI ####
Create the smooth NDVI values, "outW_ndvi", by applying the R whit1 function with the
following parameter:
        whit1(NDVI_smoothed, 2)
Save the output "outW_ndvi" in  NDVI_smoothed
```

Algorithm 4-1. Time series smoothing

4.2 Features extraction

This step is organized in two consecutive sub-tasks: the definition of temporal features and the spectral features extraction (Figure 4-2). They are detailed in the two sub-sections 4.2.1 and 4.2.2.

[1] In the pseudo-code of section 4, the following color convention is used: green for input data, red for output data and brown for temporary files



Temporal features definition
NDVI slopes computing
Maximum NDVI slope date
Minimum NDVI slope date
Maximum NDVI date
Minimum NDVI date
Maximum red date
Smoothed filled image time series
Spectral features extraction
Reflectance values (green, red, NIR, SWIR) at maximum NDVI slope date
Reflectance values (green, red, NIR, SWIR) at minimum NDVI slope date
Reflectance values (green, red, NIR, SWIR) at maximum NDVI date
Reflectance values (green, red, NIR, SWIR) at minimum NDVI date
Reflectance values (green, red, NIR, SWIR) at maximum red date
Spectral features extraction

Figure 4-2. Workflow of the temporal and spectral features extraction process

4.2.1 Temporal features definition

This step identifies key dates during the growing cycle, based on the reflectance values and NDVI time series.

First, it computes the slopes of the "NDVI *vs* time" curve using the OTB function "*otbcli_BandMath*" (https://www.orfeo-toolbox.org/CookBook/CookBooksu132.html).

Then, it identifies on a per-pixel basis the date when the maximum and minimum slopes are achieved and the maximum and minimum NDVI are achieved. The algorithm also identifies when the maximum value of the red band is achieved. These processes are based on the R functions "*which.max*", "*which.min*", "*fwhmax*" and "*fwhmin*".

The dates are stored in 5 separate raster (one by date).

The algorithm runs by tile.

Input

band<n>_smoothed – raster files including the smoothed reflectance values for all the dates (output of Algorithm 4-1)

NDVI_smoothed – raster files including the smoothed NDVI values for all the dates (output of Algorithm 4-1)

Image_dates – a text file giving the acquisition dates of each image within the time series

Output

date_max_slope – raster file which stores, for each pixel, the date corresponding to the maximum slope of the “NDVI *vs* time” curve

date_min_slope – raster file which stores, for each pixel, the date corresponding to the minimum slope of the “NDVI *vs* time” curve

date_max_NDVI – raster file which stores, for each pixel, the date corresponding to the maximum NDVI value

date_min_NDVI – raster file which stores, for each pixel, the date corresponding to the minimum NDVI value

date_max_red – raster file which stores, for each pixel, the date corresponding to the maximum red value

Equations

The slope of the NDVI vs time curve is calculated according to the Equation 4-1.

$$Slope_t = \frac{\left(NDVI_{t+1} + NDVI_{t\_1}\right)}{\left(t_{t+1} - t_{t-1}\right)} \qquad \text{(Eq. 4-1)}$$

Algorithm parameters

The parameters of the OTB function “*otbcli_BandMath*” are given in Table 4-2.

Table 4-2. Parameters of the otbcli_BandMath function.

Parameters	Description	Value
Exp	Mathematical expression	E.g. (im1b3-im1b1)/difft

The R functions “*which.max*”, “*which.min*”, “*fwhmax*” and “*fwhmin*” don’t have parameters.

Pseudo-code representation

Algorithm “temporal features identification” **is**

Temporary output:
d raster (Tif) files, *d* corresponding to the different dates of acquisition corresponding to the slope of the curve “NDVI *vs* time”.

```
        NDVI_slope_<d>

 ## Computing the slopes of NDVI####
Read the image dates in the Image_dates text file

For d in 3:(length(dates)) (//The loop starts with the 3rd date, because the equation needs three followed dates to be computed)
        Compute “diffdate” as : dates[d] - dates[d-2]
        Run the otbcli_BandMath function:
```

```
        system( otbcli_BandMath -il   NDVI_smoothed -out NDVI_slope_<d> -exp
        '(im1b[d-2]-im1b[d])/diffdate')
End of For

## Identification of the temporal features ####
    ## Look for maximum and minimum NDVI slope####
    List the files which contain dates of slopes (output of previous steps):
        inFileList <-list of files including all NDVI_slope_<d>
    In the list, identify the maximum and minimum slopes using the which.max and
    which.min R functions:
        whmax = which.max(inFileList)
        whmin = which.min(inFileList)
    save "whmax" as a raster in  date_max_slope
    save "whmin" as a raster in  date_min_slope

     ## Look for max/min ndvi value####
    Identify the maximum and minimum NDVI values using the fwhmax and fwhmin R
    functions:
        whMm = fwhmax(NDVI_smoothed)
        whmm = fwhmin(NDVI_smoothed)
    save "whMm" as a raster in  date_max_NDVI
    save "whmm" as a raster in  date_min_NDVI

    ## Look for max red value####
    Identify the maximum Red value using the fwhmax R function:
        whmax = fwhmax (band2_smoothed)
    save "whmax" as a raster in date_max_red

## Correction of dates to correspond with smoothed data (since we cannot compute the slope
for the first & last dates) => offset=+1  ####
Run the otbcli_BandMath function:
system( otbcli_BandMath -il date_max_slope -out date_max_slope -exp 'im1b1+1')
system( otbcli_BandMath -il date_min_slope -out date_min_slope -exp 'im1b1+1')
```

Algorithm 4-2. Identification of temporal features

4.2.2 Spectral features extraction

This step consists in associated the key dates to their spectral values: for each pixel, the spectral information corresponding to the dates identified for the 5 key dates of the growing cycle (see Algorithm 4-2) are identified and stored in a new raster.

This is achieved thanks to a C++ program, named "*selectInList*".

Input

band<n>_smoothed – raster files including the smoothed reflectance values for all the dates (output of Algorithm 4-1)

date_max_slope – raster file which stores, for each pixel, the date corresponding to the maximum slope of the "NDVI *vs* time" curve

date_min_slope – raster file which stores, for each pixel, the date corresponding to the minimum slope of the "NDVI *vs* time" curve

date_max_NDVI – raster file which stores, for each pixel, the date corresponding to the maximum NDVI value

date_min_NDVI – raster file which stores, for each pixel, the date corresponding to the minimum NDVI value

date_max_red – raster file which stores, for each pixel, the date corresponding to the maximum red value

Output

ref_max_slope – raster file which stores, for each pixel, the reflectance values corresponding to the maximum slope date

ref_min_slope – raster file which stores, for each pixel, the reflectance values corresponding to the minimum slope date

ref_max_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the maximum NDVI date

ref_min_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the minimum NDVI date

ref_max_red – raster file which stores, for each pixel, the reflectance values corresponding to the maximum red date

Each raster file contains n layers (n being the number of spectral band).

Algorithm parameters

The parameters of the "*selectInList*" program are given in Table 4-3.

Table 4-3. Parameters of the SelectInList program

Parameters	Description	Value
inbands	Vector of the different raster files corresponding to each of the smoothed spectral bands	E.g. ('band1_smo.tif', 'band2_smo.tif', 'band3_smo.tif, 'band4_smo.tif')

Pseudo-code representation

Algorithm “spectral features extraction” **is**

```
## Extraction of reflectance values in each spectral band for the 2 dates corresponding to the maximum and minimum slope ####
define the parameters:
        infile<- date_max_slope
        inbands<-(band1_smoothed, band2_smoothed, band3_smoothed, band4_smoothed)
        outfile<- ref_max_slope

run the selectInList  function:
        system (selectInList  outfile  infile inbands)

define the parameters:
        infile<- date_min_slope
        inbands<-(band1_smoothed, band2_smoothed, band3_smoothed, band4_smoothed)
        outfile<- ref_min_slope
run the selectInList  function:
        system (selectInList  outfile  infile inbands)

## Extraction of reflectance values in each spectral band for the 2 dates corresponding to the maximum and minimum NDVI ####
define the parameters:
        infile<- date_max_NDVI
        inbands<-(band1_smoothed, band2_smoothed, band3_smoothed, band4_smoothed)
        outfile<- ref_max_NDVI
run the selectInList  function:
        system(selectInList  outfile  infile inbands)

define the parameters:
        infile<- date_min_NDVI
        inbands<-(band1_smoothed, band2_smoothed, band3_smoothed, band4_smoothed)
        outfile<- ref_min_NDVI
run the selectInList  function:
        system (selectInList  outfile  infile inbands)

## Extraction reflectance values in each spectral band for the date corresponding to the maximum Red ####
define the parameters:
        infile<- date_max_red
        inbands<-(band1_smoothed, band2_smoothed, band3_smoothed, band4_smoothed)
        outfile<- ref_max_red
run the selectInList  function:
       system( selectInList outfile  infile  inbands)
```

Algorithm 4-3. Extraction of spectral features.

```
Program SelectInList is:
for each pixel : (// loop over all the pixels)
        id = dateID  (// dateID is the file containing the key date to select (date_max_slope, date_min_slope, date_max_NDVI, date_min_NDVI, date_max_red))
        for each input file i:
          output(i) = pixelMB(i)(id) (// pixelMB is a list of multiband files (band<n>_smoothed) with the first index for list item and the second for band)
```

Algorithm 4-4. SelectInList program

4.3 Reference map preparation

The reference is the map that will be used to define the training samples that will feed the coming classification algorithm. This is the key dataset that allows generating a cropland mask in the absence of field data.

This step consists in removing, from the reference map, the pixels that are at the border of the different classes (or said differently, that are between two classes) as they have a higher probability to be wrongly classified (Figure 4-3).

Reference
Erosion
Eroded reference

Figure 4-3. Workflow of the reference map preparation

The process is based on a function named **classErosion**, written in python during the benchmarking and that relies on the python **erosion** function from the "*skimage.morphology*" module.

Input

Reference – raster file containing the original reference map, with the same extent and projection than the image time series.

Output

Eroded_reference – raster file containing the eroded reference map, with the same extent and projection than the image time series. Removed pixels are coded with NaN values.

Algorithm parameters

The parameters of the python **erosion** function are given in Table 4-4

Table 4-4. Parameters of the erosion function

Parameters	Description	Value
Image	Image array corresponding to each class	classinRst
selem	Binary array made of 0 and 1 values	sq(3)

Pseudo-code representation

Function classErosion **is**

```
def classErosion(Reference, Eroded_reference, size=3):

## Reading the input ####
Reference is opened and stored in “inRst”

## Identifying the different classes included in the reference ####
The different classes of “inRst” are identified and included in “uniqueClass” (where the classes are identified from 1 to u)

## Defining the size of the erosion ####
selem = sq(size) (where size is “3” according to Table 4-4 => selem is squared matrix of 3x3 ‘ones’)

## Erosion of the different classes ####
for u in “uniqueClass”:
        all the areas where “inRst” == u are saved in “classinRst”
        “eroded” = erosion(classinRst, selem)
        the resulting eroded areas are saved in “outRst”

## Saving the output file ####
An array is written compiling the “outRst” coming from each class
The array is saved in Eroded_reference
```

Algorithm 4-5. Reference erosion.

4.4 Trimming

Trimming consists in truncating a distribution from its least probable values that behave like outliers. The common purpose of this procedure is to reduce the sensitivity to outliers for many parameter estimates, such as the sample mean and variance.

Trimming, coupled with the previous reference map erosion, are two key steps that give robustness to the classification process. They will allow obtaining, from an inexact pre-existing reference map data that will be “clean” enough to train a classifier. They are therefore the core of the “without in-situ mode”.

The spectral distribution corresponding to each class of the eroded reference is iteratively trimmed in order to eliminate the pixels that behave as “outliers” from their Gaussian distribution point of view.

The trimming will not be performed on all spectral channels, but on the ones that are the most characteristic for crops – no crops. The selected channels are the red and NIR reflectance values at the minimum NDVI date as well as the green, red and NIR reflectance values at the maximum NDVI date. This selection could be updated when S2 data will be available to better use the S2 spectral richness.

The trimming is made using a "*MVG_IT*" function specifically created in R that computes iteratively the outliers of a Gaussian distribution. "*MGV_IT*" uses, in turn, two functions from the base core of R, which are the "*qchisq*" and "*mahalanobis*" ones.

From the trimming output (i.e. the identification of clean pixels in the reference map), a selection of samples is done randomly. This selection will constitute the training dataset for the subsequent RF classifier.

It shall be noted that even if the trimming was done on a selection of spectral channels, the RF classifier will run with all available spectral information: red, NIR and SWIR reflectance values from the dates corresponding to the maximum red, minimum and maximum NDVI and minimum and maximum slope.

The workflow of the trimming process is presented in Figure 4-4.

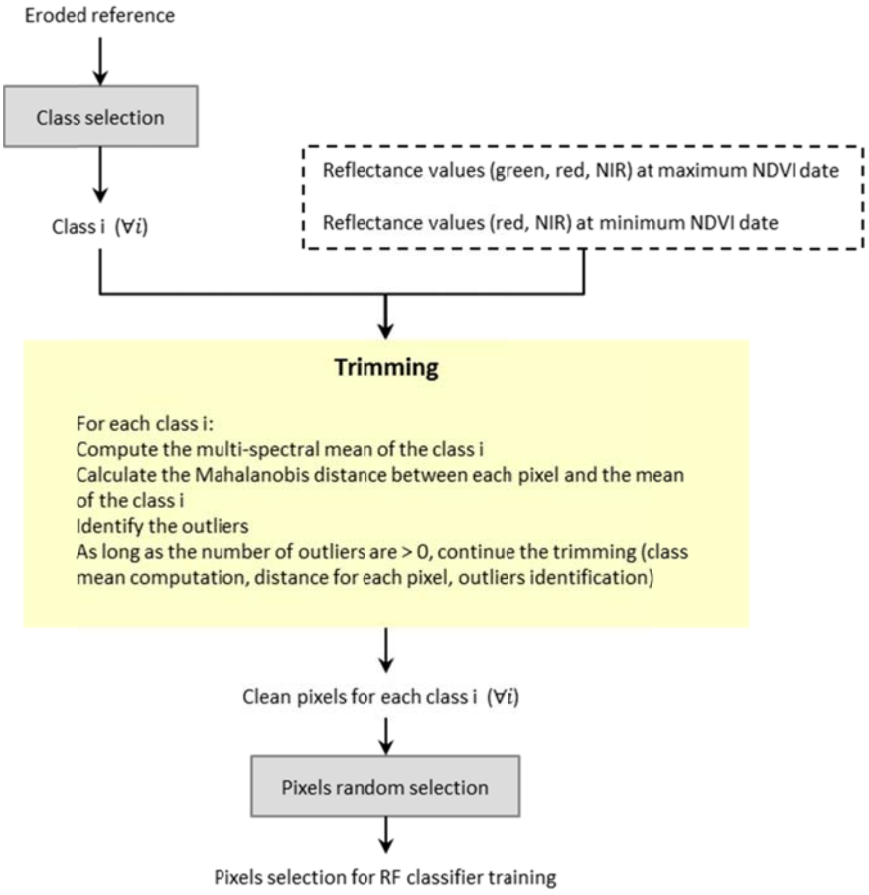

Eroded reference
Class selection
Class i (∀i)
Reflectance values (green, red, NIR) at maximum NDVI date
Reflectance values (red, NIR) at minimum NDVI date
Trimming
For each class i:
Compute the multi-spectral mean of the class i
Calculate the Mahalanobis distance between each pixel and the mean of the class i
Identify the outliers
As long as the number of outliers are > 0, continue the trimming (class mean computation, distance for each pixel, outliers identification)
Clean pixels for each class i (∀i)
Pixels random selection
Pixels selection for RF classifier training

Figure 4-4. Workflow of the trimming process

Input

Eroded_reference – raster file containing the eroded reference map, with the same extent and projection than the image time series. Removed pixels are coded with NaN values.

ref_max_slope – raster file which stores, for each pixel, the reflectance values corresponding to the maximum slope date *ref_min_slope* – raster file which stores, for each pixel, the reflectance values corresponding to the minimum slope date

ref_max_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the maximum NDVI date

ref_min_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the minimum NDVI date

ref_max_red – raster file which stores, for each pixel, the reflectance values corresponding to the maximum red date

Output

Training sample – data frame which contains the samples that will be used for the training.

Algorithm parameters

The parameters of the “*MGV_IT*” function are given in Table 4-5. Those of the “*qchisq*” and “*mahalanobis*” functions are described in Table 4-6 and Table 4-7.

Table 4-5. Parameters of the MGV_IT function

Parameters	Description	Value
x	Vector or matrix of a specific class for which the Gaussian model will be fitted	E.g. subset(trimRst,LC_class == class.i)
alpha	Confidence interval used to define what is an “outlier”	0.01
TrimBands	Selection of spectral bands that will be used for the trimming	Red and NIR from the minimum NDVI date // green, red and NIR from the maximum NDVI date

Table 4-6. Parameters of the qchisq function

Parameters	Description	Value
x	Vector of quantile values	(1-alpha), where alpha is 0.01
df	Degrees of freedom	NCOL(x)

Table 4-7. Parameters of the mahalanobis function

Parameters	Description	Value
x	Vector or matrix of data	x.trim
center	mean vector of the distribution	colMeans(x.trim)
cov	covariance matrix of the distribution	cov(x.trim)

Pseudo-code representation

```
Algorithm Trimming is

## Define input working files ####
Store Eroded_Reference in "labelRst"
Store the spectral features (ref_max_slope, ref_min_slope, ref_max_ndvi, ref_min_ndvi, ref_max_red) in "featRst"
Define the number of samples needed for the RF training in "ntrain"

## Convert input working files in a data frame ####
Store "featRst" as data frame in "metricRst"
Store "labelRst" as data frame in "LCstack"
Store "metricRst" and "LCstack" as data frame in "Rststack"

## Select reflectance values to use in the trimming (i.e. reflectance values whose Gaussian distribution will be cleaned) ###
Select the bands TrimBands (see Table 4-5 for this parameter) from "Rststack"
Select the class information corresponding to "LCstack" (class information) from "Rststack"
Store this information in "trimRst"

## Create a list of classes  ###
List all classes ID in a list "classList"

## Run the trimming, with the function MVG_IT ###
Create the list "trimList" that will receive the results of the trimming for the different classes

For each class 'i '  from the "classList"
        class.i <- classList[i]
        X  <- selection of  trimRst corresponding to class.i
        X.NP_IT  <- MVG_IT(X)
        trimList[i]<- including the result X.NP_IT for the class 'i'

## From the trimming output trimList, make a random selection for each class of n pixels that will be used to train the RF ####

For each class 'i '  from the "classList"
        subset.idx<- selects from "trimList[i]" a random sample of "ntrain" pixels
        out.id<-list of ones and zeros with the size of "trimList", '1' corresponding to the selected pixels (subset.idx)
        trimmed_values.sbs<-subset of "trimList" corresponding to out.id==1
        Training_Sample <- trimmed_values.sbs
```

Algorithm 4-6. Trimming

```
Function MVG_IT is

MVG_IT <- function(x, alpha = 0.01){
        chi <- qchisq((1-alpha), df=NCOL(x))
        (computes the Chi-Squared distribution and selects the threshold corresponding to the
        alpha p-value of the distribution per class)
        x.trim <- x
        For every class of the legend
                md <- as.matrix(mahalanobis(x.trim, colMeans(x.trim), cov(x.trim)))
                (Computing the Mahalanobis distance to the mean for each pixel)
                x.trim <- x.trim which md is lower or equal to chi
                (Selecting the "clean" pixels, i.e. those pixels whose mahalanobis distance to
                the mean is lower than the threshold)
        }
        it returns x.trim as output of the function
}
```

Algorithm 4-7. MVG_IT function

4.5 Random forest classifier

The classification is a supervised method relying on a RF classifier, just like in the "in-situ" mode. The RF classifier is applied to the reflectance values associated with the key dates (maximum and minimum NDVI slopes, maximum and minimum NDVI, maximum red). The RF classifier uses as training data the "clean" pixels selected by the trimming.

This step follows exactly the same logical than the RF classification used for the "in-situ" mode. The presentation is thus rather brief.

4.5.1 RF model estimation

The process follows the same logical than for the "in-situ" mode.

Input

Feature Vector Image – vector image containing the spectral features:

ref_max_slope – raster file which stores, for each pixel, the reflectance values corresponding to the maximum slope date

ref_min_slope – raster file which stores, for each pixel, the reflectance values corresponding to the minimum slope date

ref_max_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the maximum NDVI date

ref_min_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the minimum NDVI date

ref_max_red – raster file which stores, for each pixel, the reflectance values corresponding to the maximum red date

Feature Vector Image Statistics – the mean and the variance of each feature composing the feature vector image. This result can be obtained by applying the OTB application presented in 2.3.1 of the ATBD crop type document.

Vector File containing training pixels – vector file containing the training pixels

Output

Training model – resulting RF model to be used in the classification step.

Algorithm parameters

Parameters of the RF classifier are given in Table 4-8.

Table 4-8. Parameter for the RF classifier

Parameters	Description	Value
rf.nbtrees	Number of decision trees	100
rf.min	Minimum number of samples per nodes	5
rf.max	Maximum depth of each tree	25

Pseudo-code representation

The RF algorithm has been presented in detail in Section 2.3.2 of the ATBD crop type document and is therefore no more recalled here.

4.5.2 Classification

The classification process applies the RF model built in the previous step using the training dataset on the image time series in order to generate the cropland mask.

Input data

Feature Vector Image – vector image containing the spectral features:

ref_max_slope – raster file which stores, for each pixel, the reflectance values corresponding to the maximum slope date

ref_min_slope – raster file which stores, for each pixel, the reflectance values corresponding to the minimum slope date

ref_max_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the maximum NDVI date

ref_min_NDVI – raster file which stores, for each pixel, the reflectance values corresponding to the minimum NDVI date

ref_max_red – raster file which stores, for each pixel, the reflectance values corresponding to the maximum red date

Feature Vector Image Statistics – the mean and the variance of each feature composing the feature vector image. This result can be obtained by applying the OTB application presented in 2.3.1 of the ATBD crop type document.

Training model – resulting RF model to be used in the classification step.

Output data

Pixel-based classification map – resulting crop mask product.

Algorithm

The RF algorithm has been presented in detail in Section 2.4.1 of the ATBD crop type document and is therefore no more recalled here.

5 Common final step: crop mask post-filtering

As presented in Figure 3-1, the post-filtering task is made of 3 steps: (i) the Principal Component Analysis (PCA), (ii) the mean-shift algorithm and (iii) the majority vote. These steps are detailed in Figure 5-1.

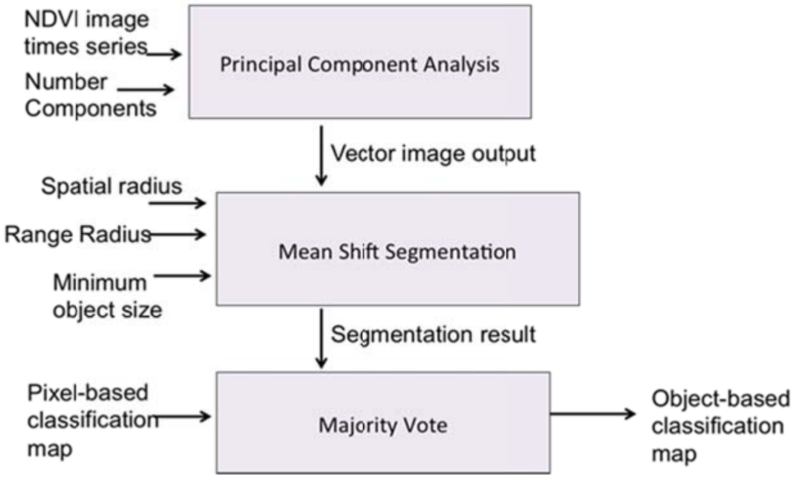


Figure 5-1. Workflow of the post-filtering step

5.1 Principal Component Analysis

Input data

NDVI image times series – vector image containing the complete NDVI image times series.

Number of Components – number of principal components composing the output image.

Output data

Vector image – vector image containing the "*Number of Components*" principal component images.

Available code

A C++ code compatible with OTB has been developed, this code uses the otb filter class "*otb::PCAImageFilter*".

5.2 Mean-Shift segmentation algorithm

Input data

Vector image output – vector image containing the result of the PCA

Spatial radius – spatial radius of the neighborhood

Range radius – range radius defining the radius (expressed in radiometry unit) in the multispectral space

Minimum object size – Minimum size of a region (in pixel unit) in segmentation. Smaller clusters will be merged to the neighboring cluster with the closest radiometry. If set to 0 no pruning is done.

Output data

Segmentation result – segmentation result image.

Available code

A C++ code compatible with OTB has been developed, this code uses the otb filter class "*otb::MeanShiftVectorImageFilter*"

5.3 Majority vote

Input data

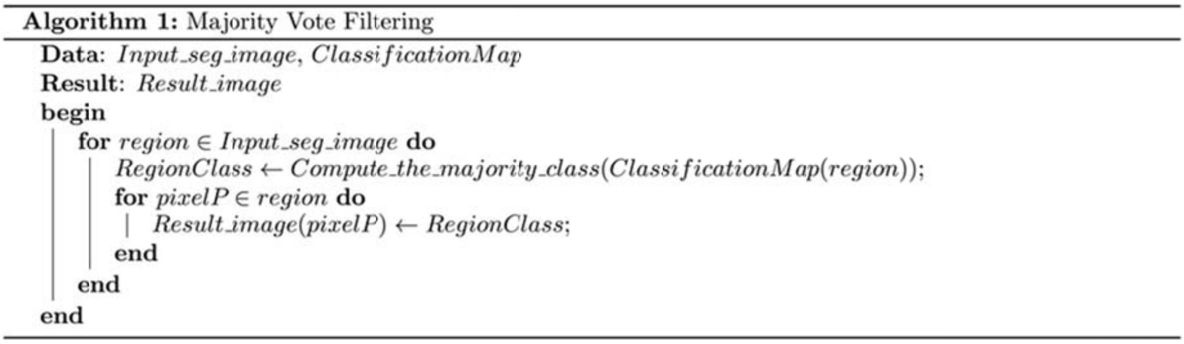
Segmentation result – segmentation image obtained by the mean-shift segmentation algorithm

Pixel-based classification map – pixel-based classification results obtained by the RF classifier (coming from the "in-situ" mode or not).

Output data

Object-based classification map – object-based classification map result

Algorithm



```
Algorithm 1: Majority Vote Filtering
  Data: $Input\_seg\_image$, $ClassificationMap$
  Result: $Result\_image$
  begin
      for $region \in Input\_seg\_image$ do
          $RegionClass \leftarrow Compute\_the\_majority\_class(ClassificationMap(region))$;
          for $pixelP \in region$ do
              $Result\_image(pixelP) \leftarrow RegionClass$;
          end
      end
  end
```

5.4 Operational context

It must be remembered that in the operational context, the post-filtering task must also be done following an iterative procedure. This iterative approach will be possible from the first satellite data acquisitions. This can be obtained by following the next flowcharts, Figure 5-2 and Figure 5-3.

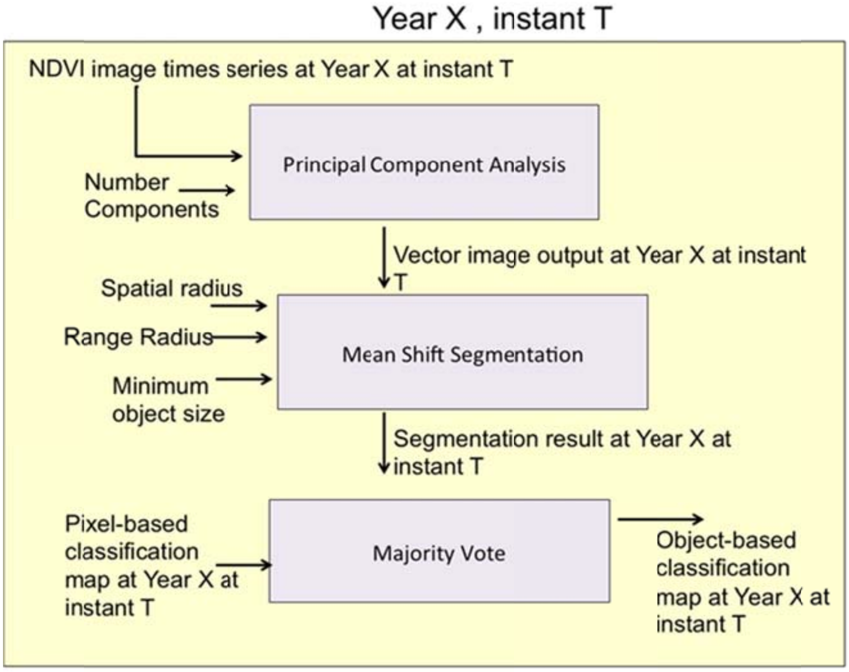

Year X , instant T
NDVI image times series at Year X at instant T
Principal Component Analysis
Number Components
Vector image output at Year X at instant T
Spatial radius
Range Radius
Minimum object size
Mean Shift Segmentation
Segmentation result at Year X at instant T
Pixel-based classification map at Year X at instant T
Majority Vote
Object-based classification map at Year X at instant T

Figure 5-2. Post-filtering workflow, at instant T

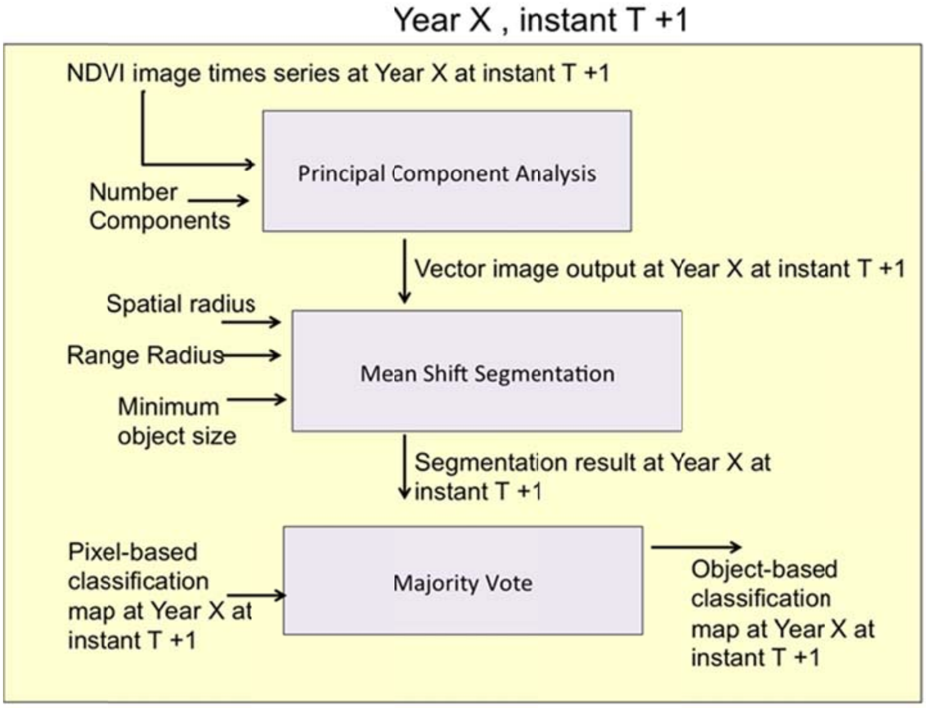

Year X , instant T +1
NDVI image times series at Year X at instant T +1
Principal Component Analysis
Number Components
Vector image output at Year X at instant T +1
Spatial radius
Range Radius
Minimum object size
Mean Shift Segmentation
Segmentation result at Year X at instant T +1
Pixel-based classification map at Year X at instant T +1
Majority Vote
Object-based classification map at Year X at instant T +1

Figure 5-3. Post-filtering workflow, at instant T+1

6 Appendix - Additional information on the codes of the « in-situ » processing chain

The code, associated with the « in-situ » mode (including the common steps with the "no in-situ" mode) is available on: http://tully.ups-tlse.fr/valeros/s2agri_cropmask

Codes from the OTB project are the following ones:

- **NDVI feature computation task:**

Includes/NdviParameterExtraction.hpp

tests/IndicesComputation/NdviParameterExtractionApplication.cxx

- **Statistics feature computation task:**

includes/IndexParameterExtraction.hpp

tests/IndicesComputation/IndexParameterExtractionApplication.cxx

- **Random selection of training pixel samples task:**

includes/ShpFileSampleGeneratorFast.hpp

src/ShpFileSampleGeneratorFast.cxx

- **Random Forest Training Step task:**

test/sClassification/ ShpFileMRFClassificationFast.cxx

- **Classifiication of the complete image times series task:**

tests/Classification/SVMClassificationMap.cxx

- **Principal Component Analysis task:**

tests/PostFiltering/ACP.cxx

- **Mean Shift Segmentation task:**

tests/PostFiltering/SegmentationACP.cxx

Two algorithms were coded in Python:

- Quality measures (how to evaluate the crop masks - section 3.3.2.2)
- Majority vote (how to apply the majority vote - section 5.3)