Satellite Image Segmentation and Classification

Xavier Gigandet

Diploma project

Assistant: Dr. Meritxell Bach Cuadra
Director: Prof. Dr. Jean-Philippe Thiran

Fall 2004

Abstract

The resolution of remote sensing images increases every day, raising the level of detail and the heterogeneity of the scenes. Most of the existing geographic information systems classification tools have used the same methods for years. With these new high resolution images basic classification methods do not provide satisfactory results.

In this study we developed a region-based classification method, consisting in two steps: a segmentation and a classification. The segmentation uses a Markov model to divide the image into several homogenous regions. Then follows the region-based classification performed either with the Mahalanobis distance or by the support vector machine classifier. This method was validated and a comparison between pixel-based and region-based classification was performed.

We demonstrated that this method provides better results comparing to the existing remote sensing classification tools, even if some work should be done to prove its robustness. We also proved that the prior segmentation significantly improves the results of classification, both from the quantitative and qualitative points of view.

Aknowledgements

This diploma project was carried out between September 2004 and January 2005 at the Signal Processing Institute of the Swiss Federal Institute of Technology in Lausanne, Switzerland.

We would like to thank the director of this diploma project, Professor Jean-Philippe Thiran, the assistant, Doctor Meritxell Bach Cuadra, as well as Leila Cammoun and Vlad Popovici from the Signal Processing Institute of the Swiss Federal Institute of Technology (EPFL) for the time spent to help us for this project. Thanks likewise to Abram Pointet, from the Geographical Information Systems Laboratory of the Swiss Federal Institute of Technology (EPFL) for his comments and advices.

Contents

Chapter 1

Introduction

1.1 Background

The resolution of images provided by the satellites increases every day. Some years ago, these images had a resolution of dozens of metres. The measured luminance of one pixel was representing the mean of luminance of several ground objects. Now the new satellites reach sixty centimetres of resolution, increasing the level of detail by a factor ten. With such images we can consider that each pixel is part of a single object. Thus, the heterogeneity of images has dramatically grown.

Satellite images are mainly used in geographic information systems (GIS). Their classification is very useful for cartography. With low resolution satellite images, the intensity of the pixels is enough to individually classify each of them. On the contrary, high resolution image classification is more difficult. The increasing complexity of the scenes raises the level of details. For example a tree in a field or the shadows of the objects are visible, and the contextual information of the pixels becomes essential for a good classification. The existing GIS classification softwares generally use the same methods for low and high resolution images. If satisfactory results can be obtained with low resolution shots, the effectiveness of these softwares for high resolution images is questionable. To ensure a good accuracy, manual classification is sometimes preferred to automatic methods.

The improvements of satellite imaging then require new classification methods. Some classifiers were recently developed for biomedical imagery or industry, but are still uncommon in remote sensing. Moreover in biomedical imagery a pre-processing step, the segmentation, is often added. Its aim is to divide the image into homogenous regions in order to extract contextual features. All these new methods are not fully exploited in remote sensing.

1.2 Objective

The objective of this project is to provide an effective classification tool for the Geographical Information System Laboratory (LaSIG) of the Swiss Federal Institute of Technology (EPFL). We then developed a method combining several algorithms to perform this task, including two different steps: segmentation and classification. In the first step, the aim is to divide the image into homogenous regions. No labels are attributed, this is only a preprocessing. In the second step, features such as intensity, texture or shape are extracted from these regions in order to perform a region-based classification. At the end of this step a label is attributed to each region, corresponding to the final classification.

In this study we will present the creation of this method. We will evaluate the need of the segmentation step before the classification. We will test several classifiers and discuss the best settings to ensure an effective classification. Finally the results will be compared with the existing methods provided by the current softwares.

1.3 Structure

In order to better explain the structure of this work, let us first present the diagram of the proposed method, denoted by *classification process*, Table 1.1.

Classification process
1 Segmentation step
2 Classification step

Table 1.1: Classification process diagram

This work is presented as follows. Chapter 2 presents some of the main algorithms used in the segmentation step. Then it describes the model that we used for our experiments. The algorithms for the classification step are discussed in chapter 3. It also presents some methods such as feature extraction and feature selection. In chapter 4 the testing images are shown and the methods used to validate our results are presented. The results of segmentation and classification steps are reported and validated in chapter 5 and 6 respectively, while chapter 7 evaluates the global classification process results. Chapter 8 presents and explains the created classification toolbox. Finally the conclusion takes place in chapter 9.

Chapter 2

Segmentation step

The segmentation is a process which extracts the outline of the ground objects by defining homogenous regions. Most of the methods only use the intensity of each pixel to define the regions, but produce very noisy segmentations, particularly with the high resolution satellite images. Some algorithms now include contextual information in the process to reduce the heterogeneity of the segmentations. In some of them textural information extracted from the image is also used.

The segmentation step is generally made up of three parts, Table 2.1. We first have the pre-processing, in which we select the features to use and eventually modify or re-scale the data. The second part is the initialization of the segmentation algorithm, if needed. Finally the third part is the segmentation itself.

The aim of this chapter is to answer the following question: which method and which features provide the best segmentation? The first section will present the pre-processing techniques. The second one will deal with the segmentation algorithms. As the initialization depends on the segmentation method, it is discussed inside this second section. Finally the last section will provide a summary of the settings to select to ensure an effective segmentation.

Classification process
1 Segmentation step
1.1 pre-processing
1.2 initialization
1.3 segmentation
2 Classification step

Table 2.1: Segmentation step diagram

2.1 Pre-processing

The pre-processing is very important for the following steps. The effectiveness of the segmentation depends on this stage. Moreover, if it is not properly performed, the segmentation may fail. This pre-processing is made up of two parts: feature selection and re-scaling methods. Each part will be discussed in this section.

2.1.1 Feature selection

State-of-the-art

Image intensity is the most used feature in the segmentation methods. Satellite images are usually multi-spectral, which means that the image is made up of several bands. For example a Quickbird image consists of four bands: blue, green, red and near infrared, Fig. 2.1. The intensity of each band can be extracted, we can then use four intensity features.

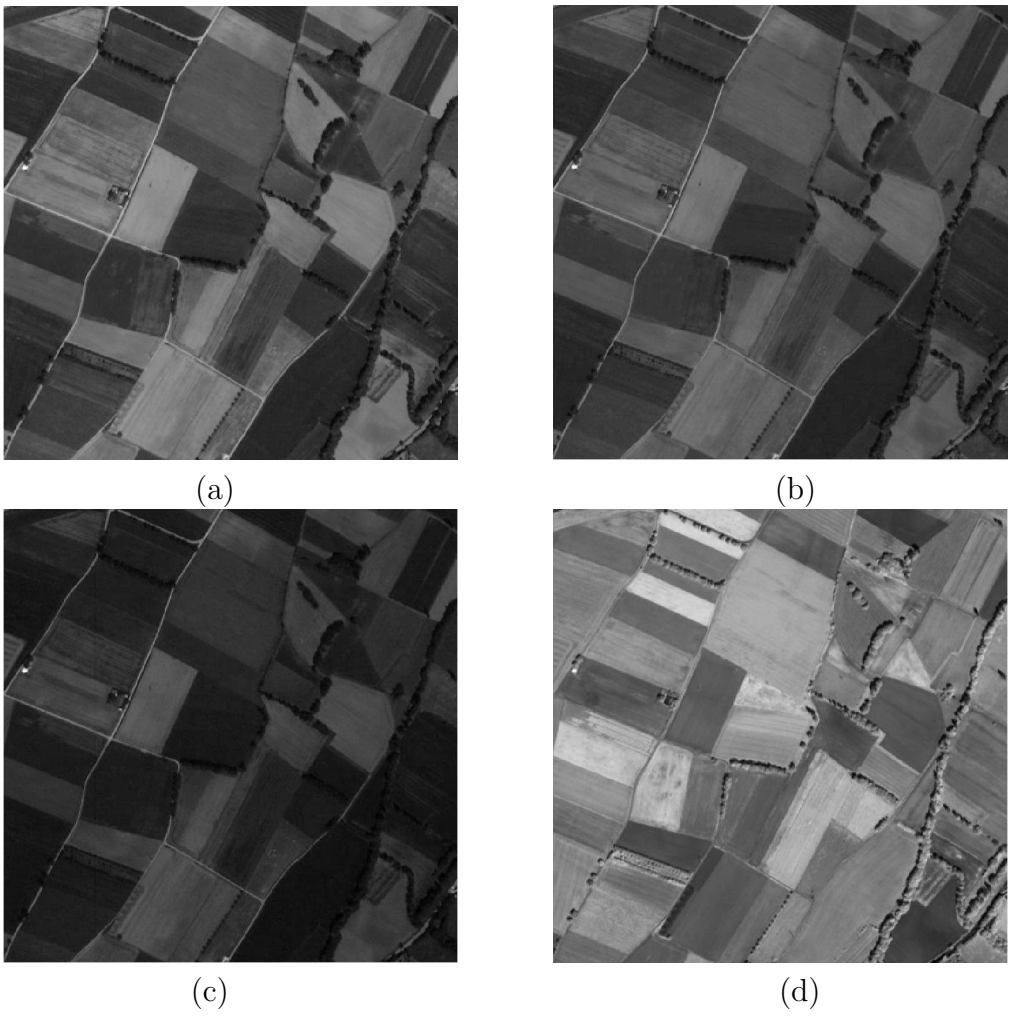
Some ground objects are better defined with their texture than with their intensity. For example, with an intensity-based segmentation a forest will be divided into several homogenous parts, which is not the case of a texture-based segmentation.

Although the texture is not directly available, it can be extracted from image intensity. Some of the methods we can use are explained in [1]. Let us summarize the main characteristics of each method:

Gray level co-occurrence matrix This method was first introduced by [2]. A matrix is defined by the distance d, the direction α, and the neighbourhood size Ng. The elements of this matrix, $p(i, j)$, represent the relative frequency by which two pixels with grey levels i and j, that are at a distance d in the direction α, are in the neighbourhood of size Ng of the pixel. Usually four different directions and a single distance are defined, so that each image has four matrices. Many features can then be extracted from these matrices, like contrast, uniformity, variance, inertia, etc.

Energy filters This method consists in applying filters to the original image, and then using the produced maps to compute the energy of the image.

Gabor filters A Gabor filter is characterized by its radial frequency, standard deviation and orientation. The resulting filtered image has variations within a limited range of frequencies and orientations. By using sev-



(a) (b)

(c) (d)

Figure 2.1: Quickbird sample, blue (a), green (b), red (c) and near infrared (d) bands

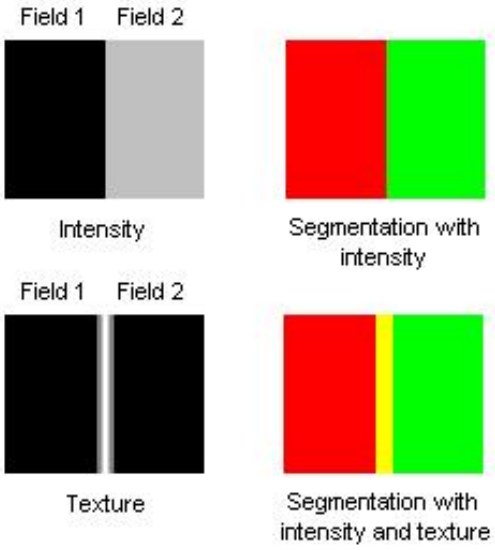


Figure 2.2: Border effect with textural features

eral filters it is possible to extract periodical structures in the image, which represent textures.

Wavelet decomposition The wavelet transform is a method to decompose the original image into a series of averaged images, called *trends*, and high frequencies images, called *fluctuations*, at different scales. The fluctuations are then used to extract some textural features like mean and variance [1, 3, 4].

In addition to these four methods, simple texture features can also be extracted from the neighbourhood of each pixel, like variance, contrast, etc [5].

Discussion

Although texture features provide important information in addition to intensity features, it is proven that their use do not necessarily improve the results of segmentation [1]. The example of Fig. 2.2 can demonstrate this affirmation.

We see with this example that adding a texture map to the intensity information provide bad segmentation because of the borders. Whatever method used for the texture extraction, the frontier between the two fields of Fig. 2.2 will be detected, because the intensity difference between them

is interpreted as a texture. By using the texture map in addition to the intensity, the pixels that are near the border are considered as a third region. The importance of this effect depends on the neighbourhood size used to extract the texture.

In [1] a quantitative comparison is made between textural and multi-spectral classification. The results show that without taking into account the border effect (inside the regions), the accuracy can be increased by around 20 percents with the addition of textural information. However, by considering the whole image the accuracy is decreased by 15 percents.

Moreover the aim of the segmentation is to divide the image into homogenous regions, which is in contradiction with the use of textural features. In view of these observations we think that textural information should not be included in the segmentation step. We will see in chapter 3 that this information is far more useful for the classification process.

2.1.2 Re-scaling methods

Depending on the type of the remote sensing images, the data can be quantified on eight, twelve or more bits. It is not rare to observe some *extreme* values in the distribution of intensity in a band of the image, obtaining sparse histograms such as Fig. 2.3. This is mainly due to the fact that sensors are calibrated to capture images of the whole Earth (showing dark and bright areas).

These extreme data can cause *divide by zero* errors in some segmentation algorithms, resulting in a segmentation failure. Different data re-scaling methods allow to avoid this problem. Some of them will be presented in what follows.

State-of-the-art

Dispersion intervals A new range is determined, so that most of the values (for example 98 percents) are contained within this range. The data whose value is lower or higher than the limits of the range are fixed to the lower, respectively the upper limit of the range. This operation is performed for each band.

Clustering The main aim of this method is to raise the histogram frequencies, especially for multi-spectral data. We first define some equally spaced clusters in the feature space. Then the nearest cluster is assigned to each data sample. After that, the new centres of the clusters are computed

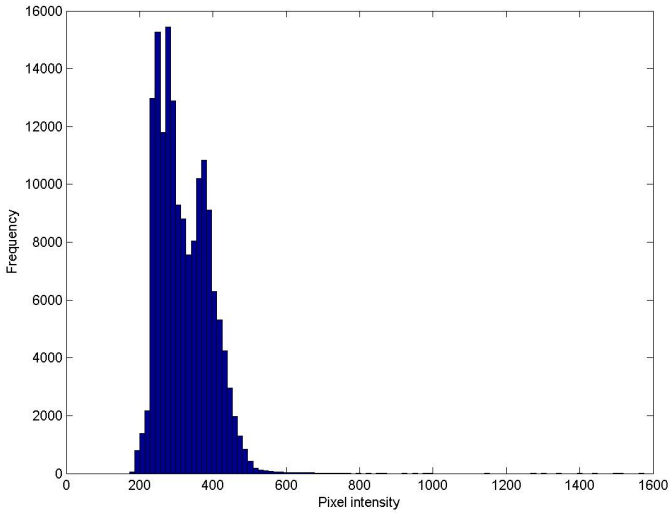


Figure 2.3: Example of pixel intensity distribution

using the assigned values while the empty clusters are deleted. This is a sort of *intelligent quantization*.

Principal component analysis This method is a way to reduce the number of bands used for computation. This algorithm can concentrate almost all the information of several channels in only one or two bands [6]. The advantage of this method is to reduce computing time. Due to the strong correlation between the bands in remote sensing imaging, the two first components of the principal component analysis generally contain more than 95 percents of the information.

Logarithm The idea of this method is to take the logarithm of the intensity values in order to bring back the extreme values near to the centre of the distribution.

Discussion

The first method is the most used in statistical analysis. However, another problem occurs in our application: all the extreme values are fixed to the same upper limit. This creates a peak in the histogram, which can lead to the crash of some segmentation algorithms. The second method does not solve our problem, because the extreme values are still present. The

principal component analysis is very effective and reduces computing time, but the results are sometimes not as good as with the original data. The last method is also very effective and is the most suited approach in our case, although the logarithmic function slightly modifies the distribution of intensity. We then recommend to apply the logarithm method for all the bands whose distribution shows that there are extreme values.

2.2 Segmentation algorithms

This section will first discuss some of the most used segmentation algorithms. Then the algorithm used in our experiments will be presented.

2.2.1 State-of-the-art

Methods

Please note that some of the algorithms presented here can not be applied to multi-spectral data, in this case a single band must be selected for the segmentation.

Split and merge The principle of the split and merge algorithm is very simple. In the first step, called *split*, the original image is divided into four equal parts. For each iteration, each part is divided into four parts only if the initial part is not homogenous (depending on a confidence interval). The first step ends when all the parts are considered as homogenous. In the second step, called *merge*, two connected parts are merged if they have the same intensity level. This second step ends when all the parts with the same intensity level are not connected.

Watershed The intensity gradient is first computed. The regions start growing from the local minima of this gradient. This is called the zero level. Then the neighbouring pixels are merged, increasing the size of regions. If two different regions are going to merge a "wall" is constructed, preventing them to merge. The segmentation consists of the resulting boundaries.

K-means This algorithm starts with some clusters of pixels in the feature space, each of them defined by its centre. The first step consists in allocating each pixel to the nearest cluster. In the second step the new centres are computed with the new clusters. These two steps are repeated until convergence.

Finite gaussian mixture model (FGMM) The principle of this model is to approximate the distribution of the features with a mixture of probability density functions [7]. For each iteration the probability of the pixels to belong to each component of the mixture is computed, and the parameters of the new components are computed according to these probabilities.

Markov models Markov models are very interesting because they include contextual information. The Markov theory asserts that the total image information can merely consist in a local neighbourhood for each pixel. These models are similar to the FGMM, but they include the neighbourhood information when computing the probabilities. There are three different structures: chains, fields and trees. The structure represents the way the data are processed. There are three different models: classical, couple and triplet. These models use respectively one, two and three Markov random fields. Finally there are three estimation methods: EM, SEM and ICE. Any combination of these properties can be implemented, we then have 27 possible Markov models [8, 9, 10].

Discussion

We can not say that one method is better for all the images, because the results depend on the type, the resolution and even the content of the images. However we can say that some algorithms are generally better than the others in remote sensing. A study has been carried out to evaluate different segmentation softwares [11]. Some of them used clustering (K-means) and some others region merging (split and merge, watershed) algorithms, and the best software of this comparison was using watershed.

The region merging methods usually give better results than clustering. Among these algorithms we can say that watershed is more effective than split and merge, this last one being penalized because it can only create polygonal regions. One of the disadvantages of the watershed algorithm is the scaling parameter. When running this method the user has to choose the scale of the regions he wants to extract. This means that if the original image has small and big areas, either the big or the small regions will be well segmented, resulting either in a loss of information for the small ones or over-segmentation for the big ones.

In our last project we had compared the FGMM to one of the Markov models, *the gaussian hidden markov random field algorithm* (GHMRF) [12]. This study had shown that the GHMRF gives better segmentations, because

of the use of contextual information. We also had compared with watershed segmentation, and the GHMRF seems to give better results, since the scale, which is dependent on a single threshold parameter, strongly affects the quality of the segmentations.

If the effectiveness of the Markov models was demonstrated, the choice of the Markov model is less obvious. In [13] different Markov models are presented, and it is said that whatever model is used, the results of segmentation are always improved comparing to the FGMM, but that none of these models is better than the others. In addition to these observations it is important to keep in mind that the more complex the algorithm, the longer the computing time.

To conclude this discussion which only applies to satellite images, we can not say that one of the presented method provides the best segmentation results. However Markov models are very interesting because they include contextual information, we therefore strongly advise the use of a Markov model for remote sensing image segmentation. The algorithm we chose for our experiments is the GHMRF, which we had already used in [12].

2.2.2 The GHMRF algorithm

The following part is a short introduction to the GHMRF algorithm, please refer to [7, 14].

Application of the EM algorithm to image segmentation

Let us suppose that we have a density function $p(\boldsymbol{x}|\boldsymbol{\Theta})$ that is governed by the set of parameters $\boldsymbol{\Theta}$. In our case, we assume the following probabilistic model:

$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{k=1}^{M}(\alpha_k p_k(\boldsymbol{x}|\boldsymbol{\theta}_k)) \tag{2.1}$$

where the parameters are $\boldsymbol{\Theta} = (\alpha_1, ..., \alpha_M, \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M)$ such that $\sum_{k=1}^{M} \alpha_k = 1$ and each p_k is a probability density function characterized by $\boldsymbol{\theta}_k$. We then have M component densities weighted by the M coefficients α_k.

Let $\mathbf{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ be a set of data supposedly drawn from this distribution. The log-likelihood of the data is given by

$$\log(\mathcal{L}(\boldsymbol{\Theta}|\mathbf{X})) = \log \prod_{i=1}^{N}(p(\boldsymbol{x}_i|\boldsymbol{\Theta})) = \sum_{i=1}^{N} \log \sum_{k=1}^{M}(\alpha_k p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k)). \tag{2.2}$$

This likelihood is difficult to optimize because we have to compute the logarithm of a sum. However, by considering $\mathbf{X}$ as incomplete data and supposing that there is a set of data $\mathbf{Y} = \{y_i\}_{i=1}^N$ whose values indicate which probability density function p_k generated each $\boldsymbol{x}_i$, we can simplify this equation. Let us then suppose that $y_i \in 1, ..., M$ for each i, and that $y_i = k$ if the sample i was generated by the distribution k. Knowing the data $\mathbf{Y}$, the likelihood becomes

$$\log(\mathcal{L}(\boldsymbol{\Theta}|\mathbf{X},\mathbf{Y})) = \sum_{i=1}^{N} \log(P(\boldsymbol{x}_i|y_i)P(y_i)) = \sum_{i=1}^{N} \log(\alpha_{y_i} p_{y_i}(\boldsymbol{x}_i|\boldsymbol{\theta}_{y_i})) \quad (2.3)$$

which can be optimized given a particular form of the component densities. The problem is that we do not know the data set $\mathbf{Y}$. However, if we assume that $\mathbf{Y}$ is a random vector we can proceed. Let us first choose an arbitrary set of parameters $\boldsymbol{\Theta}^g = (\alpha_1^g, ..., \alpha_M^g, \boldsymbol{\theta}_1^g, ..., \boldsymbol{\theta}_M^g)$. Given $\boldsymbol{\Theta}^g$, we can easily compute $p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^g)$ for each i and k. The mixing parameters α_k can be thought of as prior probabilities of each mixture component. By using Bayes's rule, we can compute:

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{\Theta}^g) = \frac{\alpha_{y_i}^g p_{y_i}(\boldsymbol{x}_i|\boldsymbol{\theta}_{y_i}^g)}{p(\boldsymbol{x}_i|\boldsymbol{\Theta}^g)} = \frac{\alpha_{y_i}^g p_{y_i}(\boldsymbol{x}_i|\boldsymbol{\theta}_{y_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^g)}. \quad (2.4)$$

Markov random fields

The hidden Markov random fields theory asserts that the total image information can be reduced to the local neighbourhood information for each pixel. The local spatial information is then taken into account and is represented by $p(y_i|y_{N_i})$, which is the probability that the pixel i was generated by the component y_i knowing the classification of its neighbourhood y_{N_i} [14]. This information is used to replace the weights α_k previously defined as prior probabilities. This function can take several forms, we chose the most used and the more intuitive one, that is

$$p(y_i|y_{N_i}) = \frac{e^{\beta V_{i,y_i}}}{\sum_{k=1}^{M} e^{\beta V_{i,k}}}. \quad (2.5)$$

$V_{i,k}$ represents the number of neighbouring pixels generated from the component k. β is a constant which defines the importance of the neighbourhood. The neighbourhood size is defined by the user.

Implementation

This algorithm assumes that the data are generated by gaussian distributions, characterized by their centre $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

1. Initialization of parameters $\boldsymbol{\Theta}^g$, β
2. For each component k of the mixture, compute
   - $$p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g) = \frac{p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^g)p(k|k_{N_i})}{\sum_{k=1}^{M} p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^g)p(k|k_{N_i})} \tag{2.6}$$
   - $$\boldsymbol{\mu}_k^{new} = \frac{\sum_{i=1}^{N} \boldsymbol{x}_i p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g)}{\sum_{i=1}^{N} p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g)} \tag{2.7}$$
   - $$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{i=1}^{N} p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g)(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{new})^T}{\sum_{i=1}^{N} p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g)} \tag{2.8}$$
   - Update of $\boldsymbol{\Theta}^g$ with $\boldsymbol{\mu}_k^{new}$ and $\boldsymbol{\Sigma}_k^{new}$.
3. Repeat step 2 until convergence of $\boldsymbol{\Theta}^g$.

2.3 Summary of segmentation step settings

As we said before, a previous study on the segmentation with the GHMRF algorithm was carried out in [12], in which a discussion on the settings was provided. In what follows some important remarks about settings will be summarized.

2.3.1 Pre-processing

Feature selection Most of satellite or aerial images are multi-spectral, we then have several bands that we can use for the segmentation. Although the information between the bands are strongly correlated, especially between blue, green and red, it was shown that the more the bands, the best the segmentation results.

Re-scaling methods In order to ensure the effectiveness of the segmentation step we suggest to use the logarithm pre-processing whenever the distribution of intensity in a band is too sparse. The number of bands could be decreased by using the principal component analysis, but the results are sometimes slightly worse than with the original image. If the computing time is critical or if the number of bands is too high, this is an effective method to reduce the amount of data. However the principal component analysis was not used in this study.

2.3.2 Initialization

The initialization is very critical because the GHMRF algorithm is only able to converge to a local minimum. The quality of the segmentations can then be strongly affected by a wrong initialization. The most used method is to give aleatory values to the parameters of the components. However, to ensure the best segmentation results, the user has to run several times the algorithm and select the best experiment. We suggest another initialization method consisting of two steps: first the *k-means algorithm* to estimate the parameters of the components. Then, in order to speed up computing time, we suggest to run the FGMM with equal α weights.

2.3.3 Segmentation algorithm

When using the GHMRF algorithm, several parameters have to be defined. In what follows we will summarize the settings to use to ensure good segmentation results.

Number of components From all the settings, the number of components is by far the most critical one. The optimum depends on the image itself, and may vary depending on the desired segmentation level. Although user's expertise is essential in this choice, two criterions can be helpful: the *bayesian information criterion* (BIC) and the *normalized entropy criterion* (NEC). Both are computed without considering spatial prior (without neighbourhood), which speeds up the computation.

- The BIC criterion is a measure of the likelihood of the data, computed by

$$BIC_k = -2\mathcal{L}_k + v_k log(N) \tag{2.9}$$

with $\mathcal{L}_k$ the logarithm of the likelihood as defined above, and $v_k = k(1 + N + N^2)$, N being the size of the data and k the number of components. The first term of this equation decreases when k increases, since the global distribution is better approximated by a high number of gaussian components. The second term depends on the size of the data and is a complex solution penalization term. This criterion allows to determine from which k the improvement between k and $k+1$ is not significant comparing to the increasing complexity.

- The NEC criterion measures the capacity of the algorithm to provide well separated components, which is a measure of the noise of the segmentation. The numbers of components which correspond to local minima of this criterion give the less noisy segmentations. Lets divide the likelihood into two terms

$$\mathcal{L}_k = C_k + E_k \tag{2.10}$$

with

$$C_k = \sum_{k=1}^{M} \sum_{i=1}^{N} p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g) log(\alpha_k^g p_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^g)) \tag{2.11}$$

and

$$E_k = -\sum_{k=1}^{M} \sum_{i=1}^{N} p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g) log(p(k|\boldsymbol{x}_i, \boldsymbol{\theta}^g)). \tag{2.12}$$

The NEC criterion is then given by

$$NEC_k = \frac{E_k}{\mathcal{L}_k - \mathcal{L}_1}. \tag{2.13}$$

This criterion is not defined for a single component mixture.

Neighbourhood Different neighbourhood types were tested: direct (four nearest neighbours), indirect (8 nearest neighbours) or extended (24 nearest neighbours). The use of distance-dependant weights was also experimented. Apart from the direct neighbourhood which presents some artifacts, all the neighbourhood types give the same quality of segmentation. By default we suggest the indirect neighbourhood which is faster to compute than

the extended one. The choice of the β parameter is very important. This depends on the type of the image, as well as the scale of the segmentation. A small value will provide noisy segmentations, while a large one will damage the geometry of the objects. Usually β is experimentally chosen between 0.5 and 1.5, but user's expertise is essential.

Stopping criterion The stopping criterion is the maximum relative error of the centres of the distributions between two iterations. The choice of this value is not critical, we suggest 0.1 percent to ensure a good precision.

Chapter 3

Classification step

The basic principle of the classification is to assign a label to each instance, which is for remote sensing images either a pixel or a region. Each label corresponds to a class having its own properties. The algorithm that assigns these labels is called *classifier*. The classifier, which can be supervised or not, uses extracted features from the data to choose the labels.

In our case, regions are created in the segmentation step. The classification step is then made up of three parts, Table 3.1. In the first part, several features are extracted from the created regions. Then the second part consists in selecting the most appropriate features. Finally, the classification is performed.

The first section of this chapter will present the extracted features. Different feature selection methods will be discussed in the second section, while the last one will describe the classifiers used in this study.

Classification process
1 Segmentation step
2 Classification step
2.1 region-based feature extraction
2.2 feature selection
2.3 classification

Table 3.1: Classification step diagram

3.1 Region-based feature extraction

Each region must be characterized by some values to perform a classification, that is why we need to extract features from these regions. The following part will present the features that we extracted. Let r be a region of the segmented image and $P_r = p_{1,r}, ..., p_{K,r}$ be the set of pixels of size K belonging to the region r. $I_{p,b}$ is the intensity of the pixel p in the band b.

Intensity features The intensity is defined as the mean of the intensity of the pixels belonging to the same region. With multi-spectral data the mean can be computed on each band. The mean intensity for a region r in the band b is given by

$$I_{mean,r,b} = \frac{\sum_{i=1}^{K} I_{p_{i,r},b}}{K}. \tag{3.1}$$

Two features can also be interesting when using multi-spectral data: the red-infrared and red-green ratios. The first one is useful to detect the wooded areas, since vegetation has a low intensity in the red band but is very sensitive in the near infrared band. The second one can be helpful in the recognition of some farming areas.

The red-infrared ratio is computed with the following equation:

$$Ratio_{R,PIR,r} = \frac{I_{mean,r,R}}{I_{mean,r,PIR}} \tag{3.2}$$

and the red-green ratio is given by

$$Ratio_{R,G,r} = \frac{I_{mean,r,R}}{I_{mean,r,G}}. \tag{3.3}$$

Texture features The four methods extracting textural information previously described in section 2.1.1 (gray level co-occurrence matrix, energy filters, Gabor filters, wavelet decomposition) could also be applied to regions. However, these techniques need a lot of computing time. Another method is to evaluate some parameters on the regions from the intensity maps. For example the standard deviation of intensity can easily be computed.

The standard deviation of intensity is given by

$$I_{std,r,b} = \sqrt{\frac{\sum_{i=1}^{K} (I_{p_{i,r},b} - I_{mean,r,b})^2}{K}}. \tag{3.4}$$

Shape features The main shape features are the area and the perimeter. The compactness, which is the ratio between the area and the perimeter, gives an idea of the shape of the regions and can be very useful for image classification.

Let g be the resolution of the original image. The area of the region r is computed by

$$Area_r = Kg^2. \tag{3.5}$$

The perimeter is given by

$$Perimeter_r = g(\sum_{i=1}^{K} N_{p_{i,r}}) \tag{3.6}$$

with $N_{p_{i,r}}$ being the number of sides of pixel i belonging to the frontier of region r. We can then compute the compactness with the following equation:

$$Compactness_r = \frac{Perimeter_r}{4\sqrt{Area_r}}. \tag{3.7}$$

Discussion Note that the usefulness of these features is not discussed at this point, since feature selection is included in the next step.

When using distance computation in the classifiers all the features must have the same range, otherwise some of them can have more weight than the others. That is why the classifiers require scaled values to work properly. The presented features were then scaled in the range [0,1] after their extraction.

3.2 Feature selection

The most critical setting for the classification step is probably the choice of the input data, and more precisely the choice of the relevant features. Actually, for each class there are some relevant and some useless features. If the user chooses to perform the classification step with all the presented features, the results will probably not be satisfying. The problem is that the useless features reduce the importance of the relevant ones, resulting in a ineffective classification.

The feature selection is an essential step to ensure the best results. One of the possible methods is to let the user choose which features he wants to use. However, it is very difficult to know which features are relevant or not, because it is generally not observable.

In the following section we will propose a feature selection method combining two techniques, the *cross-validation* and the *sequential generation*.

3.2.1 Cross-validation

The principle of a v-fold cross-validation is to divide the training set into v subsets of equal size [15]. Next, each subset is sequentially tested using the $v-1$ other subsets to train the classifier. Each training sample is then predicted once and we can evaluate the accuracy of the classifier by computing the ratio of correctly classified samples.

Supposing that the created subsets are representative of the whole data set, we can approximatively predict the total accuracy of the classification. Moreover, the cross-validation may find the best internal parameters for the classifiers.

3.2.2 Feature selection by sequential generation

The cross-validation can evaluate any subset of features by giving its accuracy. Each existing subset of features could be tested to find the optima. However computing time is generally too critical to allow this complete search.

In [16] different feature selection methods are presented. Two of these techniques will be explained in what follows. Note that we suppose that we can know the score of a subset (with the cross-validation for example).

Sequential Forward Generation The analysis begins with an empty set of features. Then, among all the existing features, the one which individually provides the best score is selected and added to our empty set of features. Next, we combine our set of features with any unselected feature. Among all these combinations, the one which provides the best score is added to our set. The algorithm stops when there is no unselected features left. We then have the list of the added features, and we know each score. We can then select the best subset, according to the score or to a prior estimation of the size of the optimal subset.

Sequential Backward Generation The same method is applied, but backward. The analysis begins with the full set of features. Next, we try to remove each feature from the set. The least important feature is removed from this set. This step is repeated until the set of features is empty. We then have the historic of the removed features. We can then select the best subset.

These two methods are complementary, we then suggest to use both, and to select the subset which provides the best score. This method does not guarantee that the selected subset of features is the best one, but it is generally a good compromise between time cost and accuracy.

3.3 Classifiers

There is a wide range of classifiers. It is beyond the scope of this work to describe all of them in details. That is why we will first present an overview of some of the most known classifiers. Next, the advantages or disadvantages of each classifier will be discussed. Finally the classifiers that we used in our study will be presented.

3.3.1 State-of-the-art

Note that we make the difference between unsupervised classifiers, which only need samples to perform automatic classification, and supervised classifiers, which have to be trained with a set of samples whose labels are known, called the training set.

Unsupervised classifiers

K-means algorithm This algorithm starts with arbitrary clusters in the feature space, each of them defined by its centre. The first step consists in assigning the nearest cluster to each sample. In the second step the centres are recomputed with the new clusters. These two steps are repeated until convergence.

Finite gaussian mixture model The model described in the section 2.2.1 can also be applied to the classification step. In this case each component of the created mixture corresponds to a label.

Supervised classifiers

Finite gaussian mixture model This is the supervised version of the FGMM. In the first step, the distribution of the training instances for each class is approximated with a mixture of gaussian probability density functions. Then for each testing sample the best distribution in terms of maximum likelihood is selected and the corresponding label is assigned to the sample.

Mahalanobis distance For each class the mean and the covariance matrix are computed according to the training instances. Then for each testing sample we compute the Mahalanobis distance to each class (for details please see section 3.3.3). The nearest class is then assigned to each testing sample.

K-nearest neighbours For each testing sample we search the K nearest training instances and the most represented label among these instances is selected.

Neural network This is a classifier which tries to find non linear separating surfaces in the feature space. For details please refer to [17].

Support vector machine The basic principle of a support vector machine in a two-class case is to locate a linear hyperplane that maximizes the distance from the members of each class to the optimal hyperplane. As it is sometimes not possible to separate classes with a linear hyperplane without misclassification, the support vector machine tries to find the optimal non-linear transformation to apply to the data in order to find a linear separating plane without misclassification. Finally, each testing instance is transformed and the class is chosen depending on which side of the hyperplane this instance lies. This can be easily extended to multi-class cases [18, 19].

3.3.2 Discussion

The choice of the classifier is not obvious and strongly depends on the data. As we will see later, trained or supervised classifiers are generally more performant than unsupervised ones, because of the prior interpretation of the image by the user. Among the supervised classifiers, a study assessed that the support vector machine is generally more effective than neural network and maximum likelihood classifiers [18]. According to this study, support vector machine is often considered to be the best compromise between results and complexity. However we have to keep in mind that for neural network and support vector machine classifiers the choice of the internal parameters is essential to ensure good results and is sometimes very hard, while the Mahalanobis distance classifier for example does not need any parameter adjustment.

One of the main goals of this study is to evaluate the need of a prior segmentation for the classification. We found several remote sensing classification softwares, but those which include a prior segmentation are uncommon. Most of them only perform pixel-based classification.

The library of classifiers used in these softwares is almost always the same: unsupervised classifiers (K-means) and supervised classifiers (maximum likelihood, K-nearest neighbours, Mahalanobis distance, sometimes neural network). For supervised classification, testing areas are selected by the user,

or the specifications of the classes are chosen from a library of pre-defined values.

It is also important to note that some of these softwares allow to do different levels of classification. For example we can create a classifier with two levels, the first one separating water and land, and a second step only for land areas separating urban and forest regions.

3.3.3 Classifiers used in the study

Five different classifiers were assessed in our study. The first one is an unsupervised FGMM classifier. Its supervised version was also analysed. Moreover, we tested the Mahalanobis distance, the K-nearest neighbours and the support vector machine classifiers. In what follows a detailed description of their implementation will be presented.

Let us first introduce the notation used in the following parts. R denotes the number of regions, each region r being characterized by its feature vector $\boldsymbol{x}_r$. $\mathbf{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_R\}$ is the whole data set. The corresponding set of labels is given by $\mathbf{L} = \{l_1, ..., l_R\}$. For supervised classifiers we must provide the set of training samples $\mathbf{X}_{train} = \{\boldsymbol{x}_{train,1}, ..., \boldsymbol{x}_{train,TR}\}$ with their corresponding labels $\mathbf{L}_{train} = \{l_{train,1}, ..., l_{train,TR}\}$, and the set of testing instances $\mathbf{X}_{test} = \{\boldsymbol{x}_{test,1}, ..., \boldsymbol{x}_{test,TE}\}$ with their corresponding labels $\mathbf{L}_{test} = \{l_{test,1}, ..., l_{test,TE}\}$.

Unsupervised FGMM

The equations 2.1 to 2.4 defined in the section 2.2.2 can also be applied to the region data set $\mathbf{X}$. In this case, no Markov random fields are used, and the previously defined α weights simply represent the prior probabilities of each class. The FGMM assumes that the data are generated by gaussian distributions, each of it characterized by its centre $\boldsymbol{\mu}$, its covariance matrix $\boldsymbol{\Sigma}$ and its prior probability α. This algorithm is implemented as follows:

1. Initialization of parameters $\boldsymbol{\Theta}^g$
2. For each component k of the mixture, compute
   - $$p(k|\boldsymbol{x}_r, \boldsymbol{\theta}^g) = \frac{p_k(\boldsymbol{x}_r|\boldsymbol{\theta}_k^g)\alpha_k^g}{\sum_{k=1}^{M} p_k(\boldsymbol{x}_r|\boldsymbol{\theta}_k^g)\alpha_k^g} \tag{3.8}$$
   - $$\alpha_k^{new} = \frac{\sum_{r=1}^{R} p(k|\boldsymbol{x}_r, \boldsymbol{\theta}^g)}{R} \tag{3.9}$$

- $$\mu_k^{new} = \frac{\sum_{r=1}^{R} \boldsymbol{x}_r p(k|\boldsymbol{x}_r, \boldsymbol{\theta}^g)}{\sum_{r=1}^{R} p(k|\boldsymbol{x}_r, \boldsymbol{\theta}^g)} \tag{3.10}$$

- $$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{r=1}^{R} p(k|\boldsymbol{x}_r, \boldsymbol{\theta}^g)(\boldsymbol{x}_r - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_r - \boldsymbol{\mu}_k^{new})^T}{\sum_{r=1}^{R} p(k|\boldsymbol{x}_r, \boldsymbol{\theta}^g)} \tag{3.11}$$

- Update of $\boldsymbol{\Theta}^g$ with $\boldsymbol{\mu}_k^{new}$, $\boldsymbol{\Sigma}_k^{new}$ and α_k^{new}.

3. Repeat step 2 until convergence of $\boldsymbol{\Theta}^g$.

4. For each sample $\boldsymbol{x}_r$ compute the corresponding label l_r given by

$$l_r = argmax_k(p_k(\boldsymbol{x}_r|\boldsymbol{\theta}_k)). \tag{3.12}$$

For the classification step, the number of components is simply equal to the number of classes that the user needs to extract from the satellite image. As for the GHMRF algorithm used for the segmentation step, the initialization is very critical because the FGMM is only able to converge to a local minimum. We also suggest to use the *k-means algorithm* to approximate the components of the mixture. The stopping criterion is fixed around 0.1 percent.

Supervised finite gaussian mixture model

This model is similar to the unsupervised FGMM. The difference is that this one is a trained algorithm.

1. For each class c, use the FGMM to extract the mixture of distributions M_c, characterized by the set of parameters $\boldsymbol{\Theta}_c$, using the training samples $\mathbf{X}_{train}$.

2. For each testing sample $\boldsymbol{x}_{test,te}$ compute the corresponding label $l_{test,te}$ given by

$$l_{test,te} = argmax_c(p(\boldsymbol{x}_{test,te}|\boldsymbol{\Theta}_c)). \tag{3.13}$$

In a region-based classification the number of training samples is usually very small, it is then not possible to define several gaussian distributions on a reduced number of instances. That is why the number of components in our application can not exceed one. The initialization and the stopping criterion are the same as for the unsupervised FGMM.

Mahalanobis distance

This is a minimal distance classifier, which means that it tries to minimize the distance between each testing sample and the centre of its class, defined by the training samples. The classes are assumed to follow a gaussian distribution.

1. For each class c, compute the mean $\boldsymbol{\mu}_c$ and the covariance $\boldsymbol{\Sigma}_c$ using the set of training instances $\mathbf{X}_{train}$.

2. For each testing sample $\boldsymbol{x}_{test,te}$ and each class c, compute the Mahalanobis distance given by

$$d_{\boldsymbol{x}_{test,te},c} = ((\boldsymbol{x}_{test,te} - \boldsymbol{\mu}_c)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_{test,te} - \boldsymbol{\mu}_c)^T)^{1/2}. \tag{3.14}$$

3. For each testing instance $\boldsymbol{x}_{test,te}$, compute the corresponding label $l_{test,te}$ with the following equation:

$$l_{test,te} = argmin_c(d_{\boldsymbol{x}_{test,te},c}). \tag{3.15}$$

There are no settings to adjust for this algorithm.

K-nearest neighbours

This algorithm is very simple, and one of its properties is that it does not make hypothesis on the distribution of the classes. The implementation is given by

1. For each testing $\boldsymbol{x}_{test,te}$ and each training sample $\boldsymbol{x}_{train,tr}$, compute the euclidian distance

$$d_{\boldsymbol{x}_{test,te},\boldsymbol{x}_{train,tr}} = ||(\boldsymbol{x}_{test,te} - \boldsymbol{x}_{train,tr})||. \tag{3.16}$$

2. For each testing sample $\boldsymbol{x}_{test,te}$ we select the K nearest neighbours $\boldsymbol{x}_{train,te}$ by searching the minimal distance computed right before. We then have a subset of K training samples, $D_{\boldsymbol{x}_{test,te}}$, whose labels $L_{D,\boldsymbol{x}_{test,te}}$ are known .

3. For each test instance the corresponding label is given by the most represented class in the subset $L_{D,\boldsymbol{x}_{test,te}}$.

The number of neighbours is the only parameter to adjust. If it is too small, the number of neighbours is not representative of the real neighbourhood of an instance. If this number is too large, especially if it is greater than the number of training instances for each class, the neighbourhood is too extended and the results are not significant. The optimum value is then a compromise between these two aspects.

Support vector machine

Note that the following part is a very short introduction to SVM, for details please refer to [20, 19, 18].

Introduction to a linear SVM classifier Let us take a look at a two-class case. The aim of a support vector machine classifier is to locate a hyperplane that maximizes the distance from the instances of each class to this hyperplane.

Supposing that each training sample $\boldsymbol{x}_{train,tr}$ has the corresponding label $l_{train,tr} \in \{-1, +1\}$. The training data set $\mathbf{X}_{train}$ is *linearly separable* if we can define a vector $\boldsymbol{w}$ and a scalar b so that the following equations are satisfied

$$\boldsymbol{w} \cdot \boldsymbol{x}_{train,tr} + b \geq +1 \quad \forall \quad \{tr \mid l_{train,tr} = +1\} \tag{3.17}$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_{train,tr} + b \leq -1 \quad \forall \quad \{tr \mid l_{train,tr} = -1\}. \tag{3.18}$$

We want to find a hyperplane dividing the data so that all the instances with the same label are on the same side of this hyperplane. The aim is then to find $\boldsymbol{w}$ and b so that

$$l_{train,tr}(\boldsymbol{w} \cdot \boldsymbol{x}_{train,tr} + b) > 0 \quad \forall \quad tr \in \{1, ..., TR\}. \tag{3.19}$$

If a hyperplane that satisfies 3.19 exists, the two classes are *linerarly separable*. In this case, it is possible to scale $\boldsymbol{w}$ and b so that

$$l_{train,tr}(\boldsymbol{w} \cdot \boldsymbol{x}_{train,tr} + b) \geq 1 \quad \forall \quad tr \in \{1, ..., TR\}. \tag{3.20}$$

Such a hyperplane is said to be in its *canonical representation*. In this case the distance from the closest instance to the hyperplane is $1/||\boldsymbol{w}||$.

The optimal hyperplane can then be found by minimizing $||\boldsymbol{w}^2||$ under constraint 3.20. This optimization problem is solved by using *Lagrange multipliers*, which lead us to a maximization problem, λ_{tr} being the non-negative Lagrange multipliers associated with constraint 3.20:

$$\sum_{tr=1}^{TR} \lambda_{tr} - \frac{1}{2} \sum_{tr,tr'=1}^{TR} \lambda_{tr} \lambda_{tr'} l_{train,tr} l_{train,tr'} (\boldsymbol{x}_{train,tr} \cdot \boldsymbol{x}_{train,tr'}) \tag{3.21}$$

under constraints $\lambda_{tr} \geq 0 \quad \forall \quad tr \in \{1, ..., TR\}$.

If $\lambda^a = \{\lambda_1^a, ..., \lambda_{TR}^a\}$ is an optimal solution of the maximization problem 3.21, the optimal separating hyperplane is given by

$$\boldsymbol{w}^a = \sum_{tr=1}^{TR} l_{train,tr} \lambda_{tr}^a \boldsymbol{x}_{train,tr}. \tag{3.22}$$

If the data are not linearly separable, we can introduce a slack variable ξ_{tr} and equation 3.20 can be written as

$$l_{train,tr}(\boldsymbol{w} \cdot \boldsymbol{x}_{train,tr} + b) - 1 + \xi_{tr} \geq 0 \quad \forall \quad tr \in \{1, ..., TR\}. \tag{3.23}$$

and the conditions to find the optimal hyperplane are

$$\min_{\boldsymbol{w}, b, \xi_1, ..., \xi_{TR}} (\frac{1}{2}||\boldsymbol{w}^2|| + C \sum_{tr=1}^{TR} \xi_{tr}) \tag{3.24}$$

$$l_{train,tr}(\boldsymbol{w} \cdot \boldsymbol{x}_{train,tr} + b) - 1 + \xi_{tr} \geq 0 \tag{3.25}$$

$$\xi_{tr} \geq 0 \quad tr = 1, ..., TR. \tag{3.26}$$

We can see that 3.24 is the same equation as in the linearly separable case but with the addition of a second term, which controls the number of misclassified points. C is called the *penalty value*, and defines the trade-off between the number of misclassifications in the training data and the maximization of the margin (the minimal distance between the closest instance and the hyperplane).

Extension to a non-linear SVM classifier Sometimes it is not possible to find a linear separating hyperplane. In those cases, the solution is to map the data into a higher dimensional feature space through a non-linear transformation function, so that a linear separating plane can be found. Let $\boldsymbol{\Phi}(\boldsymbol{x})$ be the result of the transformation function applied to $\boldsymbol{x}$, equation 3.21 can be written as

$$\sum_{tr=1}^{TR} \lambda_{tr} - \frac{1}{2} \sum_{tr,tr'=1}^{TR} \lambda_{tr} \lambda_{tr'} l_{train,tr} l_{train,tr'} (\boldsymbol{\Phi}(\boldsymbol{x}_{train,tr)} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_{train,tr'})). \tag{3.27}$$

In order to simplify computation we introduce the *kernel function* K such that

$$K(\boldsymbol{x}_{train,tr}, \boldsymbol{x}_{train,tr'}) = \boldsymbol{\Phi}(\boldsymbol{x}_{train,tr}) \cdot \boldsymbol{\Phi}(\boldsymbol{x}_{train,tr'}). \tag{3.28}$$

Then, instead of computing $\boldsymbol{\Phi}(\boldsymbol{x})$ we only compute the kernel function, and the explicit form of the transformation function is not necessarily known.

There are different kernel functions, the most known have been reported in Table 3.2.

Kernel function	Definition	Parameter
Linear	$\boldsymbol{x} \cdot \boldsymbol{x}_i$	
Polynomial	$(\boldsymbol{x} \cdot \boldsymbol{x}_i + 1)^d$	d: positive integer
Radial Basis Function	$\exp(-\frac{\|\|\boldsymbol{x}-\boldsymbol{x}_i\|\|^2}{2\gamma})$	γ: user-defined value
Sigmoid	$\tanh(\kappa(\boldsymbol{x} \cdot \boldsymbol{x}_i) + \Theta)$	κ and Θ: user-defined values

Table 3.2: Most known kernel functions

Extension to multi-class classification There exist two main methods to extend support vector machine to a multi-class classification.

One against the rest classification For a M-class classification, M binary classifiers are created, each of them being trained to discriminate one class from the remaining $M - 1$ classes. During the testing phase, the instances are classified by computing the margin from the linear separating hyperplane. The label of each instance is determined by the maximal output of this margin computation.

Pairwise classification A classifier is created for each pair of classes, we then have $M(M-1)/2$ binary classifiers for a M-class classification. For each instance the label output of each classifier is stored. Then the corresponding label is selected by finding which one occurs the most.

Settings

Kernel function According to a recent study [19], the most robust and efficient kernel functions are the radial basis function and the polynomial with degree 2. We suggest then to use one of these functions, but we can not exclude some better results with other kernel functions.

Internal parameters Depending on the kernel function several parameters have to be chosen. All those parameters strongly vary depending on the input data, that is why these settings have to be user-defined, and no generalization can be done about the best values. The empirical approach seems to be the only method.

Multi-class method According to [19], pairwise seems to give better results than one against the rest classification. We then suggest to use pairwise classification to ensure good classification results.

Chapter 4

Validation Methods

The validation of the segmentation results will be made in two steps. First a qualitative evaluation will be performed on different type of images to assess the robustness of the GHMRF algorithm. Then, its effectiveness will be verified with a quantitative comparison. For the classification step, only the quantitative evaluation will be performed. In this chapter the validation methods will be presented.

4.1 Qualitative evaluation images

The first image used to test our algorithms is a sample of high resolution Quickbird satellite image. The resolution is 2.4 metres and its size is 500x500 pixels. It represents a scene mixing urban and rural areas. We dispose of four spectral bands: blue, green, red and near infrared. We can see in Fig. 4.1a the green band representation of this image.

The second test image is another sample of Quickbird satellite image with exactly the same properties except the size, 401x401 pixels. The scene does only contain rural areas. The green band is represented in Fig. 4.1b.

The next two images are samples of Aster satellite image, with a 15 metres resolution. Their size is 501x501 pixels, and the spectral bands are: green, red, infrared. The two images represent scenes mixing urban and rural areas as well as water. We can see the green band of each image in Fig. 4.2.

In order to complete this collection the last two images are aerial images. The first one has a resolution of 50 centimetres and has three spectral bands: green, red and near infrared, while the second one has a resolution of 30 centimetres and has red, blue and green spectral bands, Fig. 4.3. Their size is 501x501 pixels.

Figure 4.1: Quickbird 1 (a) and Quickbird 2 (b) images, green band

Figure 4.2: Aster 1 (a) and Aster 2 (b) images, green band

Figure 4.3: Aerial 1 (a) and Aerial 2 (b) images, green band

4.2 Quantitative evaluation tools

The quantitative validation is very important. Since test images are too heterogenous to allow visual validation, we have to define some statistical tools to evaluate our results. However, a ground truth representation of the image is necessary to compute statistical scores. The creation of a ground truth image is a heavy task, especially for images in which urban areas are present. That is why we chose the Quickbird 2 image as the validation image. The manual classification of this image was reported in Fig. 4.4. The following classes are reprsented: wooded area, farming type 1, farming type 2, open land, scarce vegetation area, urban area.

As a segmentation only consists in defining homogenous regions it is not possible to compare it directly with the manual classification. Each region was then manually classified, thus creating an ideally classified image, which could then be compared with the ground truth image. Note that the quantitative evaluation do not show the accuracy of the results since the ideal classifier does not exist, but it shows the error provided by the segmentation and that will not be corrected with the classification step. In other words, it shows the accuracy of the segmentation to keep the frontiers between the ground objects.

For the classification step the results can be compared to the ground truth image without any additional processing.

Before describing the tools used for the quantitative validation let us first introduce some notations inspired from [21]. Let n be the total num-

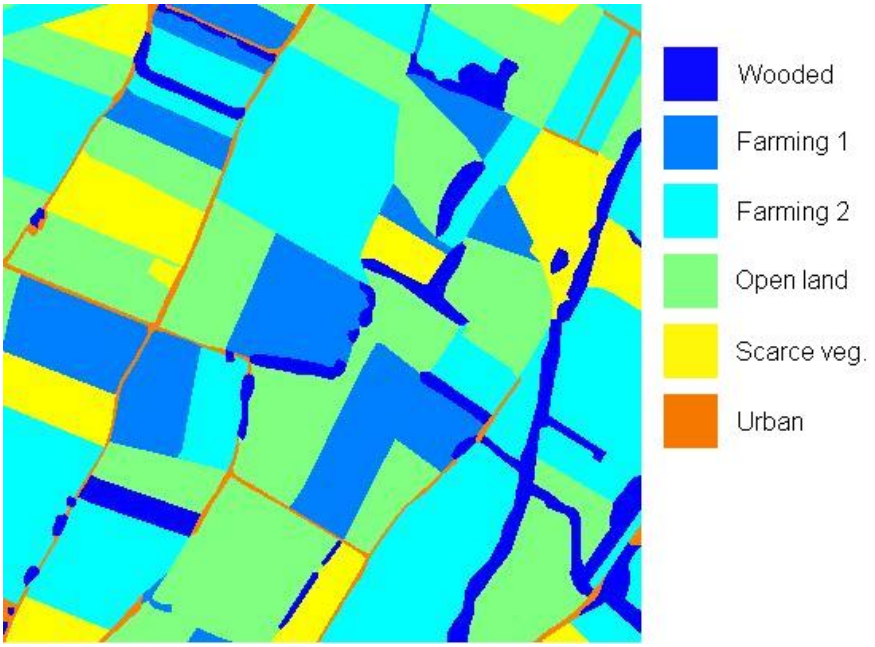


Figure 4.4: Quickbird 2 manual classification

ber of pixels, and n_{ij} be the number of pixels classified into class i in the experimental classification and into class j in the real classification. Let $n_{i+} = \sum_j n_{ij}$ be the number of samples classified into category i in the experiment, and $n_{+j} = \sum_i n_{ij}$ be the number of samples classified into class j in the reference data.

Overall accuracy This is the most used statistic for the validation. The overall accuracy is the percentage of misclassified pixels in the image. It is given by

$$p_o = \frac{\sum_i n_{ii}}{n}. \tag{4.1}$$

Kappa index This measure is very similar to the overall accuracy, but it introduces a chance agreement. A zero value would indicate that the classification agrees with the reference as bad as an aleatory classification. The Kappa index is computed by

$$K = \frac{n \sum_i n_{ii} - \sum_i n_{i+} n_{+i}}{n^2 - \sum_i n_{i+} n_{+i}}. \tag{4.2}$$

Producer's accuracy It is computed for each class. Producer's accuracy indicates the ratio of pixels of a class from the reference data that are well classified in the experimental classification. This is given by

$$producer's \quad accuracy_j = \frac{n_{jj}}{n_{+j}}. \tag{4.3}$$

User's accuracy Unlike producer's accuracy, it indicates the ratio of pixels of a class from the experiment data that are well classified. This can be computed by

$$user's \quad accuracy_i = \frac{n_{ii}}{n_{i+}}. \tag{4.4}$$

Dice Similarity index It is a measure of agreement between the regions of each class. Let $n\{R_k\}$ be the number of pixels belonging to class k in the reference data, $n\{C_k\}$ the number of pixels belonging to the same class k in the experimental classification. We can also define $n\{R_k \cap C_k\}$ the number of pixels which belong to the intersection of R_k and C_k. The dice similarity index is then given by

$$S_k = 2\frac{n\{R_k \cap C_k\}}{n\{R_k\} + n\{C_k\}}. \tag{4.5}$$

Chapter 5

Segmentation step assessment

All the results of segmentation presented in this chapter were performed with the GHMRF algorithm. The settings were those suggested in section 2.3. The β parameter was fixed to 1.0. The number of components was chosen experimentally by visual assessment. The BIC and NEC criterions were computed for each image in order to test their effectiveness.

In order to evaluate the results of our segmentation method, both qualitative and quantitative validations will be used. In the first part we will visually analyse our results on different type of images, in order to test the robustness. Then a quantitative validation will be performed to test the effectiveness of the suggested method. Next, a comparison with the existing softwares will take place. Finally a discussion on the segmentation step will be made in the last section.

5.1 Qualitative assessment

The first part will present the tests on the BIC and NEC criterions. Then, for each type of image, the best results will be reported and discussed.

BIC and NEC criterions

For the six images presented above we reported in Table 5.1 the number of components selected, for both low-scale and high-scale segmentations, in comparison with the local minima (only for $2 < k < 14$) of the NEC criterion.

We can see that most of the time the NEC criterion confirms the choices made by visual expertise. The visual interpretation of the user is essential because the less noisy results are not necessarily those which best keep the geometry of the image. However, this criterion seems to be an appropriate tool to help the expert in his choice of settings.

Image	Low-scale	High-scale	NEC minima
Quickbird 1	-	8	3,5,**8**,13
Quickbird 2	-	10,11	4,6,**10**,12
Aster 1	3	10	**3**,5,8,11
Aster 2	5	11	**5**,8,10
Aerial 1	4	-	3,8,10,12
Aerial 2	5	-	3,**5**,9,11,13

Table 5.1: Number of components selected vs. NEC criterion

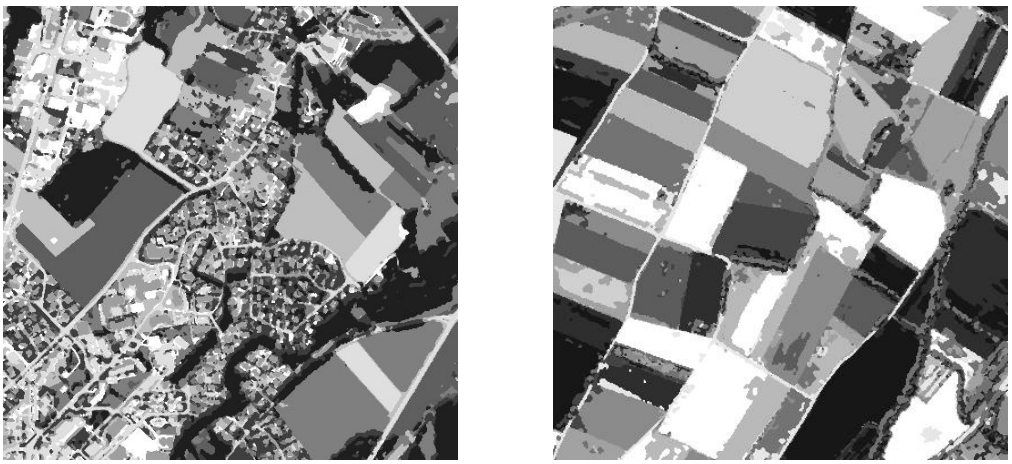


Figure 5.1: Quickbird 1 (k=8) and Quickbird 2 (k=10) segmentations

Quickbird images

We can see that the quality of the two segmentations of Fig. 5.1 is good. The large homogenous regions are well defined, while keeping the geometry in urban areas. These segmentations may seem slightly noisy, but this could be solved with the classification step or by a post-processing. Note that at this step of the classification process it is not necessary to obtain a perfect thematic partition of the territory. This will be achieved with the classification step. Remind that if an homogenous region is divided into two separated segments, this could be corrected with the classification step. On the contrary, two different regions grouped in the same segment will not be corrected. That is why it is sometimes better to do an over-segmentation to be sure not to loose regions.

In order to better evaluate the conservation of the geometry, Fig. 5.2 represents the false colours Quickbird 1 original image with the superposition

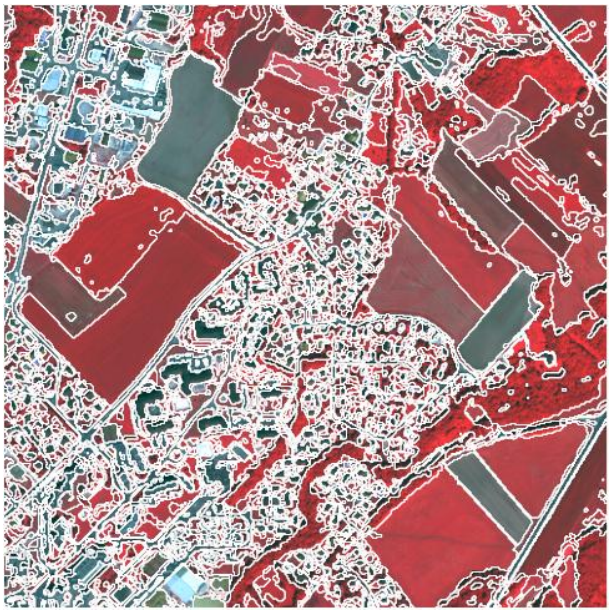


Figure 5.2: Segmentation superposed to Quickbird 1 original image

of the regions defined with the automatic segmentation. We can then confirm the ability of the model to separate simultaneously large and small areas.

Aster images

Due to the resolution of these images, two different segmentation scales can be used: the low-scale, able to separate global structures as water, urban, and farming areas, and the high-scale, which is more detailed and separates smaller objects.

The quality of the segmentations of Fig. 5.3 and 5.4 seems to be similar to the quality of the Quickbird ones, although it is difficult to compare them because of the different resolutions. The geometry is well kept, and there is very few noise. We can see that the lake is divided into two regions depending on its deepness, which could cause some problems for the classification step.

It is important to note that original images have a strong vertical noise. For the low-scale segmentations ($k < 8$) this artifact is suppressed by the GHMRF algorithm, but this is not the case when using a high number of components.

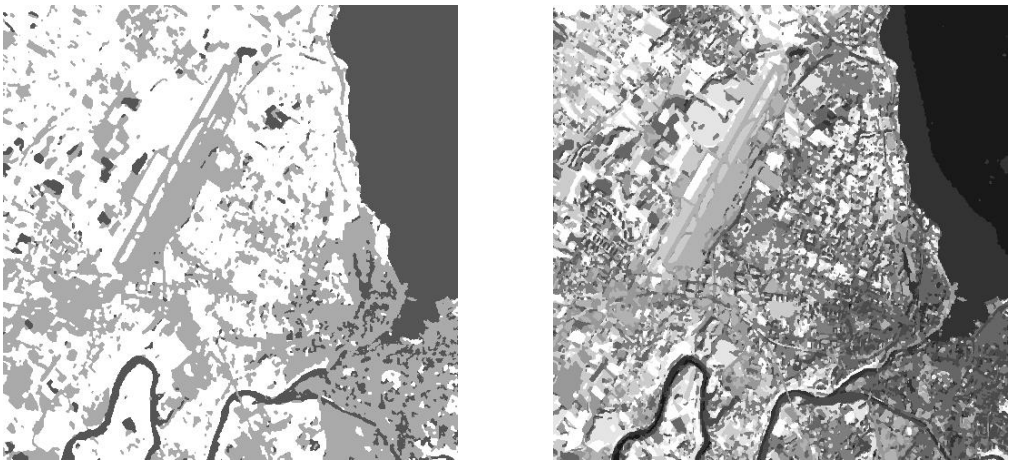


Figure 5.3: Aster 1 low-scale (k=3) and high-scale (k=10) segmentations

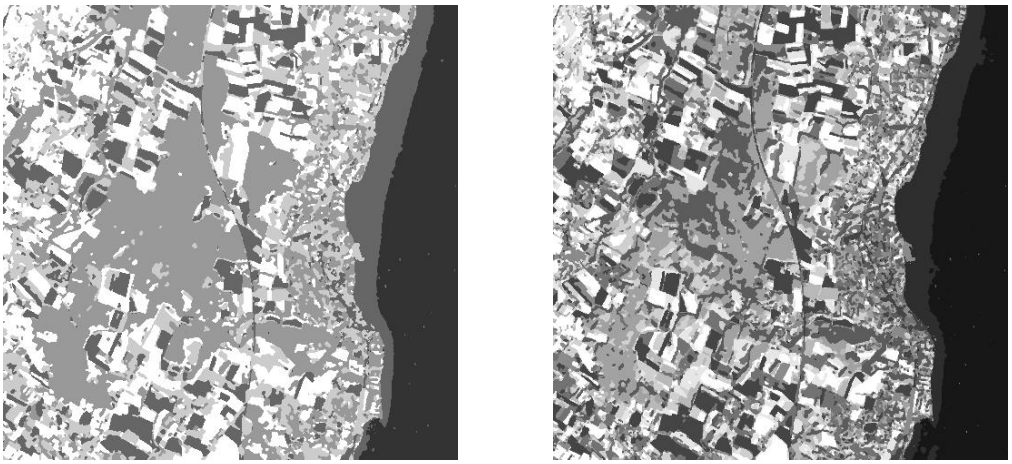


Figure 5.4: Aster 2 low-scale (k=5) and high scale (k=11) segmentations

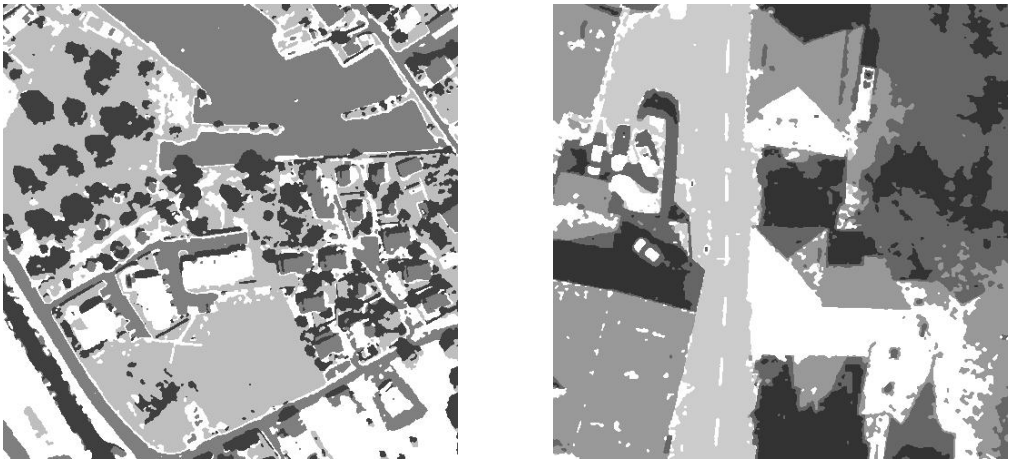


Figure 5.5: Aerial 1 (k=4) and Aerial 2 (k=5) low-scale segmentations

Aerial images

The segmentation of aerial images is more complex, due to the bad quality of these shots and the scale of the objects. For these images we do not suggest to use high-scale segmentation because of the increasing noise.

We can see on Fig. 5.5 (left) that some structures are well extracted from the original image, such as houses, trees, roads or water. The second image is harder to segment, due to the very high resolution (30 centimetres). The shadows of the objects can not be detected and treated as they should be.

The heterogeneity of aerial images is too high because of the high resolution. Moreover the information is poor since the bands are strongly correlated. The poor quality of the results does not show a weakness of the GHMRF algorithm, but rather the poor quality of aerial images.

5.2 Quantitative assessment

For this quantitative validation we computed the statistical scores of the Quickbird 2 segmented images, for each number of components between $2 \leq k \leq 13$. The overall accuracy and dice similarity index are shown in Fig. 5.6 and Table 5.2. The other statistics, the producer's and user's accuracies, as well as the kappa index, are reported in Appendix A.

We can see that the more the number of components, the better the accuracy, which is not surprising. Visually we had chosen $k = 10, 11$ as the best segmentations, and we can see that their accuracy is very good. this can

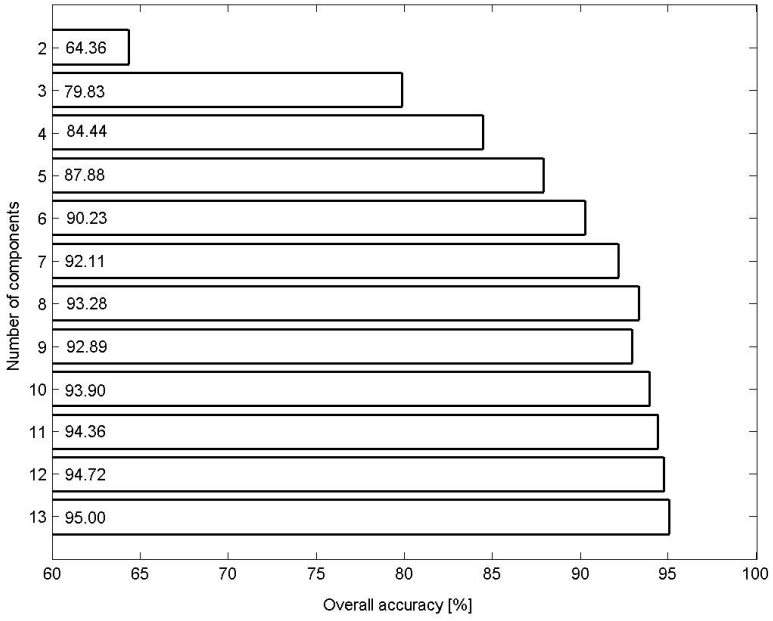


Figure 5.6: Segmentation: overall accuracy

also be shown by comparing the manual classification of this segmentation with the real classification map, Fig. 5.7.

We can see on Fig. 5.7 that some urban areas do not appear in our classification. These errors are confirmed in Table 5.2, showing the dice similarity index for each class.

Except for the low values of k that would visually be suppressed, we can see that farming, poor vegetation and open land areas have a very good dice similarity index. Urban and wooded regions are less accurate, and we can see that satisfying results are only obtained with high values of k ($k > 10$). This is probably due to the high heterogeneity of these regions. In order to ensure good results we then suggest to use a number of components higher or equal than 10. However it is important to keep in mind that if the accuracy is better with high values of k, the geometry is worse than with low values, resulting in a worse classification. This point will be discussed later in this project.

By representing the distribution of segmentation labels for each class, we can see that shape or textural features will be necessary to ensure a good classification since the intensity does not seem to be sufficient, Fig. 5.8.

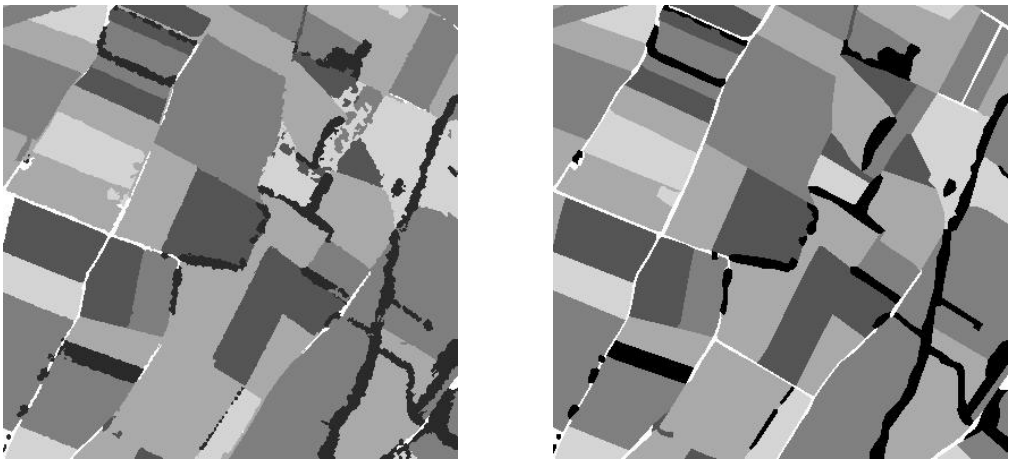


Figure 5.7: Manual classification of k=10 result and ground-truth map

	dice similarity index [%]					
k	wooded	farm. 1	farm. 2	op. land	poor veg.	urban
2	28.13	28.68	70.73	81.27	4.61	4.99
3	24.28	75.60	82.26	93.27	65.42	21.36
4	50.93	85.90	89.47	92.54	70.37	34.67
5	65.19	86.19	92.80	93.43	78.55	48.00
6	70.67	95.48	92.56	94.09	86.38	45.95
7	78.05	95.10	95.42	94.43	87.60	45.97
8	83.02	96.96	95.74	94.93	90.07	45.29
9	81.71	95.34	96.14	94.85	89.02	44.05
10	86.09	96.23	95.97	95.85	90.31	57.55
11	83.66	97.01	96.21	96.29	91.52	67.96
12	83.80	96.92	96.85	96.89	92.03	70.44
13	88.16	96.80	96.14	96.34	94.40	75.41

Table 5.2: Segmentation: dice similarity index

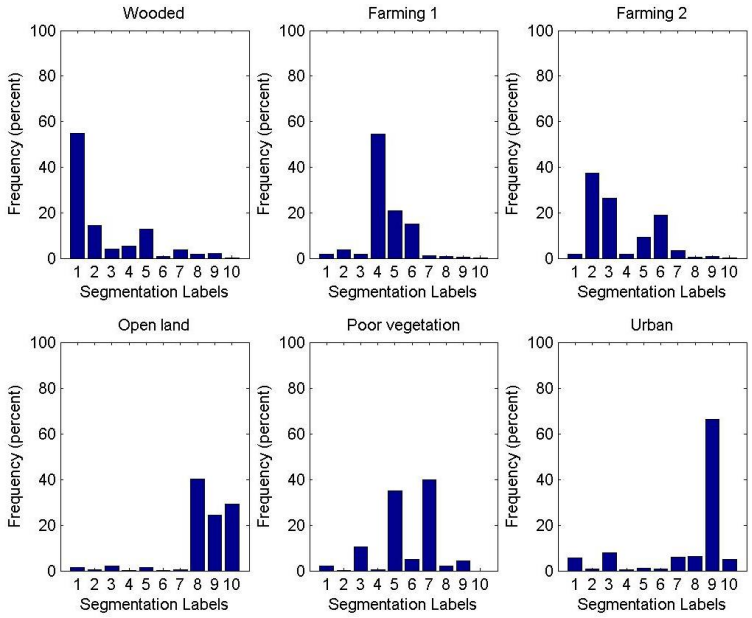


Figure 5.8: Distribution of segmentation labels (k=10)

5.3 Comparison with the existing softwares

In order to better analyse these results, we segmented the Quickbird 2 image with E-cognition, the reference software for remote sensing image classification, which supposedly uses a watershed approach for the segmentation. This image was manually classified and the same quantitative validation tools were applied. The results are shown in Table 5.3.

	E-cognition segmentation					
overall accuracy [%]	92.62					
dice similarity index [%]	78.37	94.05	94.18	95.71	91.72	67.52

Table 5.3: Segmentation: E-cognition results

These results are as good as our $k = 10, 11$ segmentations, we can then say that both segmentations are effective. The segmentation provided by E-cognition shows less noise, but the single aggregation level parameter that can be tuned with this software raises a scaling problem: it is not possible to segment simultaneously small and big regions. This is not the case of the GHMRF algorithm, which means that the segmentation step that we suggest provides a higher potential for the classification step, which could probably

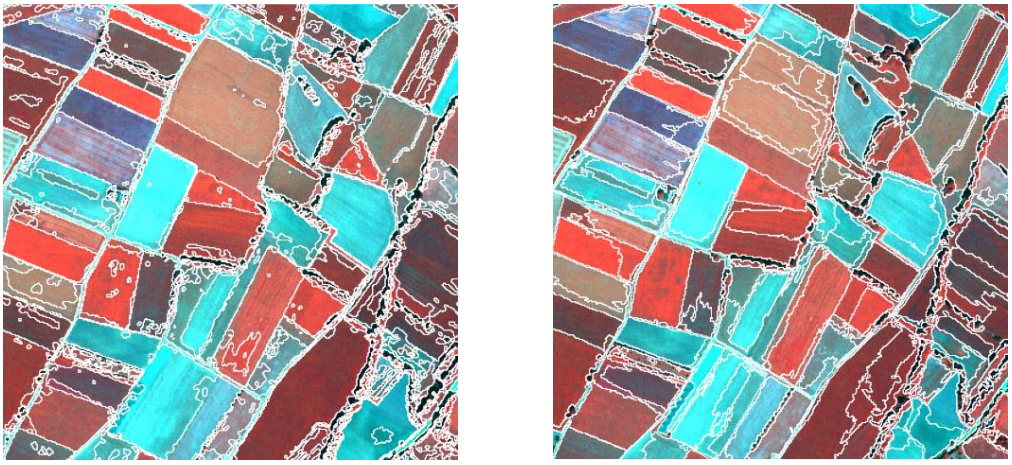


Figure 5.9: GHMRF $k = 10$ (left) and E-cognition (right) segmentations

be more powerful. This observation is confirmed by the comparison of the two segmentations superposed to the original image, Fig. 5.9.

5.4 Discussion

The results presented above allow us to do some conclusions on the segmentation step.

The GHMRF algorithm used for the segmentations seems to be an effective method. The qualitative evaluation shows that the results of segmentation are good and provide very few noise. The Markov random field is a good way to reduce the heterogeneity of these results. Moreover the basic principles of this method allow to segment simultaneously small and large areas, thus improving the conservation of the geometry. The quantitative evaluation proves that the error caused by the segmentation is very small, which confirms the visual observations.

Although quantitative evaluations were not performed on other type of images, we can suppose with the help of the visual observations that this method also works for Aster shots, and probably for other types too. However the effectiveness of the segmentation algorithms depends on the quality and the quantity of information of the image. That is why we were not able to find satisfactory results with aerial images.

The comparison of our segmentation results with an E-cognition segmentation demonstrated that the accuracy of both methods is quite equivalent. However, when performing a segmentation with E-cognition, the user must

select the size of the segments. This method then provides segments with equal size, which is not representative of the reality. We think that the GHMRF algorithm is a better approach for satellite image segmentation.

A parameter is essential to ensure good segmentation results: the number of components. From a qualitative point of view, user's expertise is necessary to ensure satisfactory results. We presented the NEC criterion, whose local minima correspond to the less noisy segmentations. This is a very useful help for the user, but can not replace it, since noise is not the only discriminating factor. Each image has its own optimal value, so the only way to find the best setting is the visual analysis. From a quantitative point of view, we showed that the more the number of components, the less the error. However, we must keep in mind that the more the number of components, the more the segments, which can lead to an over-segmentation. In other terms, the real regions of the image will be divided into several smaller parts, decreasing the information contained in the features extracted from these parts, and reducing the performance of the classification. It is then very important to find a compromise between geometry and accuracy to ensure the best classification results.

Finally, note that the GHMRF algorithm would not provide satisfactory results if the pre-processing and the initialization were not properly performed. The two prior steps presented for the segmentation are as important as the segmentation itself. By using the suggested method for pre-processing and initialization, the probability of finding good segmentation results is high.

Chapter 6

Classification step assessment

In this chapter we used the same image for all the experiments (Quickbird 2). We selected the prior segmentation with 11 components, which was presented above as the best compromise between geometry and accuracy. The feature extraction was performed, the extracted features are reported in Table 6.1. In order to simplify reading in the following tables, the features will be denoted by their number, so please refer to this table to know which features are used.

For the following experiments eight training sets of varying size were manually selected, Table 6.2. The map size is the percentage of the whole image included in the training set. The columns Wo., Fa.1, Fa.2, Op., Sc., Ur. are the number of regions in the training set of class "Wooded", "Farming 1 and 2", "Open land", "Scarce vegetation" and "Urban".

The image was always classified into six classes, in order to compare the results with the manually classified reference. The settings used were those suggested in section 3.3, except for the following ones:

Supervised FGMM We only used one component to characterize the reference distributions since the algorithm becomes unstable with a higher value. In this configuration, no initialization and stopping criterions are needed.

K-nearest neighbours The number of neighbours was experimentally set to 5, because of the poor number of training regions available for each class.

Support Vector Machine We chose the radial basis function as the kernel function. The internal parameters γ and C were chosen by a grid-search evaluation in the range $C = [2^{-20}, 2^{8}], \gamma = [2^{-10}, 2^{15}]$ for each experiment.

Number	Feature
1	Blue mean
2	Green mean
3	Red mean
4	Near infrared mean
5	Blue standard deviation
6	Green standard deviation
7	Red standard deviation
8	Near infrared standard deviation
9	Ratio red/infrared
10	Ratio red/green
11	Area
12	Perimeter
13	Compactness

Table 6.1: Features extracted from the segmentation

N	Nb regions	Wo.	Fa.1	Fa.2	Op.	Sc.	Ur.	Map size [%]
1	49	10	6	7	7	9	7	31.6
2	62	22	11	8	10	6	5	47.2
3	51	16	9	8	8	4	6	33.7
4	42	13	5	6	8	5	5	25.7
5	62	22	11	8	10	6	5	44.3
6	62	22	11	8	10	6	5	22.2
7	60	10	10	10	10	10	10	27.5
8	96	34	16	14	13	9	10	50.8

Table 6.2: Properties of training sets

In the first part we will compare the performance of the different classifiers that were used. The second part will present cross-validation results, while the third one will describe and test another method of classification. Finally a discussion on the classification step will take place.

6.1 Performance of the classifiers

In this first part our aim is to compare the performance of the classifiers. That is why an exhaustive search was performed to select the best combination of features for each classifier and for each training set. In order to evaluate the results, the accuracy of each classifier (which is the percentage of well-classified testing regions weighted by the area) is reported in Table 6.3 and Fig. 6.1.

	Training set							
Classifier	1	2	3	4	5	6	7	8
Unsup. FGMM	77.6							
Sup. FGMM	75.1	75.9	74.3	70.1	77.3	76.6	79.1	76.7
Mahalanobis	75.3	77.5	74.6	70.4	79.2	76.3	78.8	77.6
K-NN	54.3	72.0	72.0	65.8	69.9	68.3	69.6	67.6
SVM	78.2	79.5	85.2	79.9	81.8	83.5	83.5	80.8

Table 6.3: Performance of the classifiers: classifier accuracy [%]

We can see that the unsupervised classifier can provide good accuracy. However, urban areas generally disappear, which is not surprising since this classifier is not trained. This classifier is also unstable because of the arbitrary initialization. Therefore, we can conclude that the unsupervised FGMM classifier is not suitable for our purpose.

Among the supervised classifiers, we can see a slight advantage for the support vector machine, but all the tested classifiers present a good classifier accuracy. The K-nearest neighbours classifier seems to be less powerful than the others, its simplicity probably being a handicap. The Mahalanobis distance and supervised FGMM classifiers provide very similar results, which is not surprising because of their quite equivalent basic principles.

The optimal combination of features varies for each training set and each classifier. However we can find some similarities between the subsets:

- Shape features are generally less useful than the others. If we take a look at the manually classified image, we can see that forest and

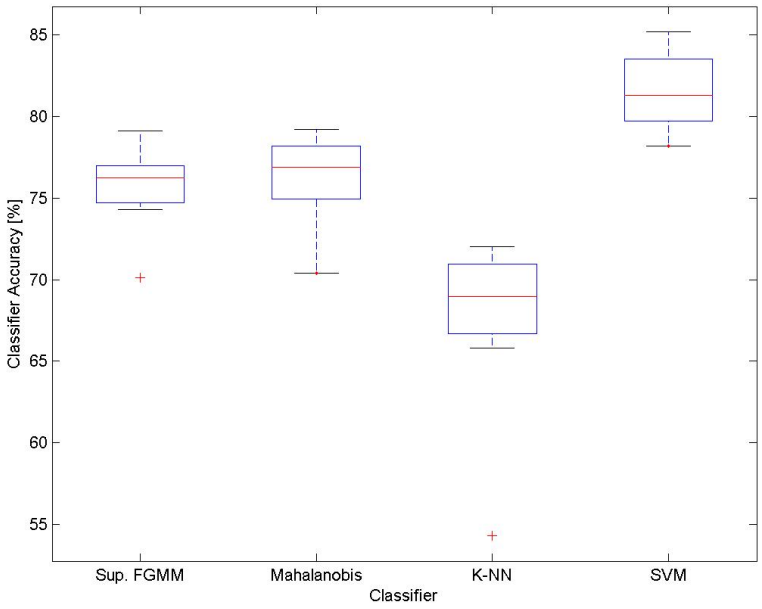


Figure 6.1: Performance of the classifiers

urban areas excepted, shape features do not seem to be significant. In other terms the classes have shape characteristics uniformly distributed. These two classes do not have enough influence to allow the inclusion of shape features in the classifier.

- The features 4 and 8, which correspond to near infrared mean and standard deviation, are very common in the subsets. This can be explained by the great quantity of information included in this band.

If supervised classifiers present almost the same performances, their stability is very different. In Fig. 6.2 we can see the distribution of the classifier accuracy for all the combinations of features and for each classifier.

The supervised FGMM is not robust at all, since the algorithm fails for more than 60 percents of the experiments. If the K-nearest neighbours and the Mahalanobis distance classifiers are quite equivalent in terms of stability, the support vector machine seems to be by far the most robust: 18 percents of the subsets of features provide more than 70 percents of classifier accuracy. Therefore, we can say that this classifier is the most stable of the tested supervised classifiers. We then strongly suggest to use it whenever possible,

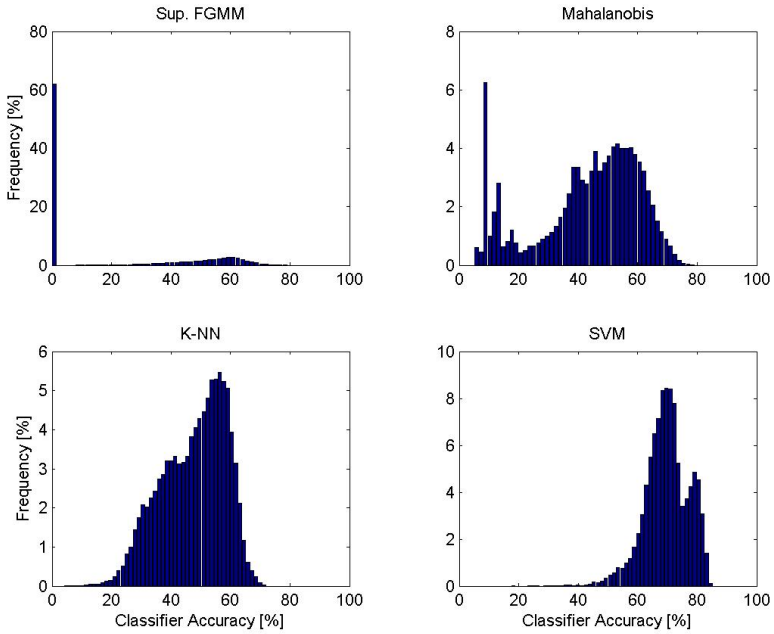


Figure 6.2: Distribution of classifier accuracies

despite the fact that the selection of internal parameters is very critical, requiring user's expertise.

6.2 Cross-validation assessment

The aim of this part is to test the effectiveness of the feature selection method presented in section 3.2, that is cross-validation combined with sequential generation. We used the Mahalanobis distance classifier with the settings chosen in the section 6.1, so that we could compare the results of cross-validation classification with the results previously reported in Table 6.3.

For these results we performed sequential forward generation for feature selection. Both sequential forward and backward generation are generally performed, and the best score is selected among these two methods. However, sequential backward generation did not give satisfactory results in our case. We then decided not to include sequential backward generation in our method.

When using cross-validation we can choose the number of folds, that is the number of training subsets that are used. After some tests this value was

N	C-Valid. clas. acc. [%]	Max clas. acc. [%]
1	56.4	75.3
2	72.5	77.5
3	52.8	74.6
4	57.1	70.4
5	72.2	79.2
6	68.0	76.3
7	69.7	78.8
8	63.8	77.6

Table 6.4: Cross-validation: classifier accuracy

set to 3, which seems to be the best compromise between computing time and accuracy.

The cross-validation is based on an arbitrary selection of training subsets, which means that the results strongly vary. Moreover, the training sets are generally small, and the created subsets are not representative of the whole data set. We then applied the following process, which gave us the best results: the cross-validation is repeated several times. Then for each feature the frequency of appearance in the subsets is computed. We then consider the three most frequent features as the best feature selection subset.

For each training set we computed the best subset of features. Then, we classified the regions by using this subset. The classifier accuracy is reported in Table 6.4. In order to simplify reading we also reported the maximum classifier accuracy obtained in the previous section.

Let us first note that as the cross-validation involves an arbitrary selection of training subsets, the results reported above may slightly vary.

If we compare these results with the maximum classifier accuracy we can see that the results of cross-validation are between 5 and 22 percents worse. As we previously said, automatic classification is not the best solution, and nothing can replace user's expertise. However, this method can be seen as a help for starting the evaluation of parameters.

This method was also applied to the K-nearest neighbours and supervised FGMM classifiers, and the results were nearly the same as for the Mahalanobis distance classifier. These results were then not reported in this study.

This cross-validation method was also evaluated for support vector machine classifier, but this did not give satisfactory results for two main reasons:

- For each subset of features we have to perform a grid-search to obtain the internal parameters of the classifier. The cross-validation is then

hundreds of time slower than for Mahalanobis distance. If this analysis is performed in around 30 seconds for the latter, the time spent for the former is too long to allow the use of this method.

- Even if we took the time to perform the whole analysis, the internal parameters chosen by cross-validation could not be used for the whole set, because they are very sensitive and dependant from the training set. That means that even after this long analysis the user should manually select the internal parameters to ensure satisfactory results.

We did then not found an effective method for automatic classification with the support vector machine classifier. However, as we said above, around 20 percents of the combinations of features give a classifier accuracy better than 70 percents with the support vector machine. It is then faster and probably more accurate to arbitrary select a combination of features and to manually optimize the internal parameters.

6.3 Multi-level classification

When we look at the manually classified image, it seems obvious that the classes do not have the same spectral and geometrical characteristics. For example, we can say that the compactness is relevant for urban areas, since these regions are very elongated. On the contrary, farming areas do not present singularities in shape features. We can then say that each class has its own set of relevant features.

The principle of the multi-level method is to divide the classification into several levels. Each level discriminates one class c from the remaining ones, and excludes the regions belonging to this class for the next level. Then the same method is applied to the remaining regions, until all the classes are determined. For each level, only the relevant features are used, so that each class is constructed with its best features.

By its definition the support vector machine is the most appropriate for one against the rest classification. We evaluated the potential of this method for the training set 2, Table 6.5.

We can see that shape features are essentially used for urban areas, as expected. The individual scores are very good, and by combining these classification levels in the order "Urban, Farming 1, Scarce vegetation, Wooded, Open land, Farming 2", we obtained a global classifier accuracy of 84.6 percents. The improvement comparing to the classical support vector machine classifier (79.5 percents) is not really significant. Moreover this method requires more time and there is a lot of parameters to estimate. Therefore, we

Class	Classifier acc. [%]	Features
Wooded	95.9	1,3,5,6,8
Farm.1	97.3	1,3,4
Farm.2	89.2	3,4,9,11
Open l.	94.9	1,2,5,8
Scarce v.	97.5	3,4,8,9,11
Urban	99.2	3,7,11,12,13

Table 6.5: Multi-level classification: classifier accuracy

do not suggest to use the multi-level classification for this image. However, this method can be useful in other cases, this possibility should then be let for the experimented users.

6.4 Discussion

According to the results presented above, we can conclude that the Mahalanobis distance and support vector machine classifiers are the two best tested classifiers. The support vector machine usually gives better performance, but the tuning of the internal settings is not easy, since no methods can estimate them. This classifier should then be let to experimented users. The Mahalanobis distance classifier is very simple to use, so we strongly suggest to begin image classification with this one. Both classifiers seem to be very robust to the training sets. Unfortunately we could not test these classification methods with other images, so it is difficult to extend our conclusions to other kind of images.

The feature selection is by far the most important part of the classification step to ensure a good accuracy. We presented a method combining sequential generation with cross-validation to perform this task. From these experiments we can conclude that this method does not work for the support vector machine because of the internal parameters. For the other classifiers, the results strongly vary depending on the training sets. Therefore, we can not say that this method is reliable. User's expertise is still essential, even if this feature selection method can be a good help for less experimented users.

The multi-level classification method may improve the results. However, this method requires the tuning of a lot of parameters. As it would be too long to perform an entire feature selection for each class, we only advise this method for experimented users, who well know the spectral and geometrical characteristics of the classes they want to extract.

Chapter 7

Classification process

In this chapter we will evaluate the effectiveness of the whole classification process presented above. We will first combine the results of both segmentation and classification steps to get the overall accuracy of our process. Next, we will compare these results with pixel-based classification, and we will evaluate the robustness of our method. Finally, a discussion about these experiments will be made.

7.1 Classification process assessment

7.1.1 Quantitative evaluation

We selected the best two tested classifiers, that is Mahalanobis distance and support vector machine. We then computed for each training set (with the best subset of features previously determined) the overall accuracy of the classification process. The results are reported in Table 7.1.

Globally we can see that these results are good for the two classifiers. The support vector machine provides better results, but as we previously said, the optimization of its internal parameters can discourage some users. We can also note that even if the overall accuracy is sensitive to the training set, the method is robust to the variation of the training sets, for all that

	Training set							
Classifier	1	2	3	4	5	6	7	8
Mahalanobis	79.4	84.3	80.2	74.3	85.7	79.3	81.7	85.3
SVM	81.5	86.2	86.3	81.5	86.6	83.3	83.5	80.8

Table 7.1: Classification process: overall accuracy [%]

Dice similarity index [%], training set 2, Mahal. classifier					
wooded	farm. 1	farm. 2	op. land	scarce veg.	urban
68.1	85.6	86.9	92.1	70.2	53.6

Table 7.2: Classification process: dice similarity index

they are chosen by a confirmed user.

The dice similarity index demonstrates that wooded and urban areas are less accurate than the others classes. We can see it on the following example, showing the dice similarity index for the training set 2 with the Mahalanobis distance classifier, Table 7.2. However, this is not surprising since this weakness was demonstrated in the segmentation step. We can then affirm that this problem comes from the segmentation, and that the classifier is effective for all the classes. Scarce vegetation areas are slightly less accurate than the others one, but the reason is probably the poor representation of this class in the test image.

7.1.2 Qualitative evaluation

In order to better evaluate the obtained results, Fig. 7.1 and 7.2 present respectively the Mahalanobis distance (overall accuracy: 84.3 percents) and the support vector machine (overall accuracy: 86.2 percents) classified images, with the training set 2, compared to the reference.

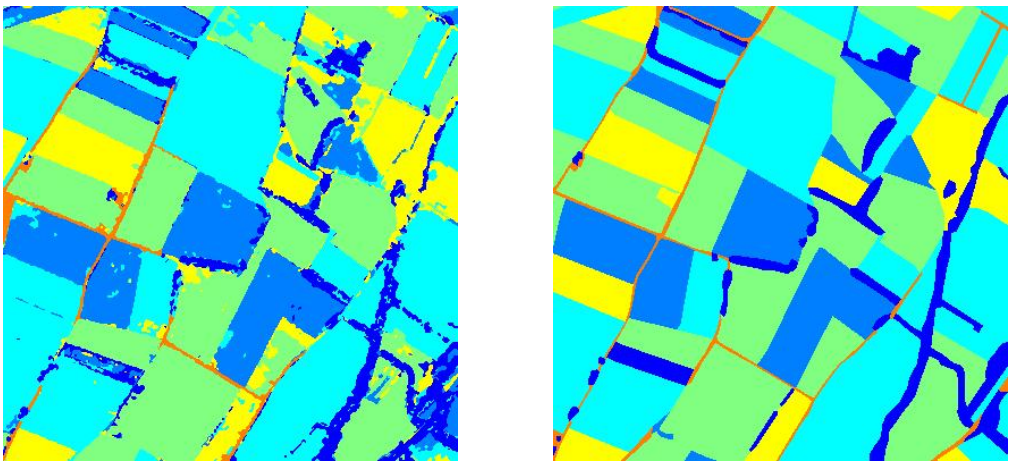


Figure 7.1: Mahalanobis distance classified image (left) and reference (right)

We can say that these classifications are good, since very few regions are lost comparing to the reference. The three main error factors are:

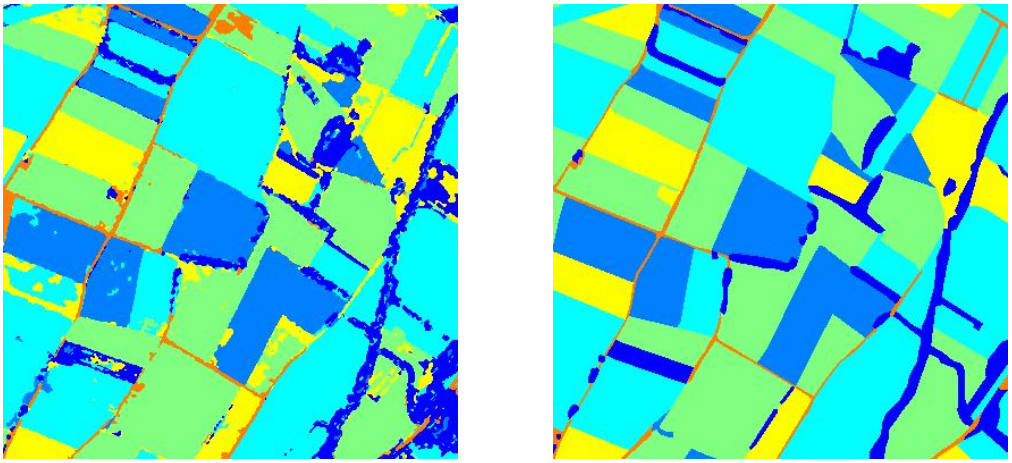


Figure 7.2: SVM classified image (left) and reference (right)

- The error of the segmentation, that means the regions that were not correctly extracted in the first step. This mainly affects urban areas.
- The noise: after the segmentation step, the image presents many small segments that do not correspond to real objects. The cause is a too important heterogeneity inside the regions. The classifier is not able to aggregate this noise to the real regions they belong to, because the spectral and geometrical characteristics are very different.
- The shadows: when performing automatic classification, taking into account the shadows of the objects is very difficult. Our process does not include algorithms that allow the treatment of these shadows. The real frontiers of the objects are then not necessarily exact.

The first error factor could be corrected by a better segmentation, but other artifacts would probably appear. The second one could be suppressed by a post-processing after the classification process. The third factor can hardly be corrected since this is a general problem of remote sensing images.

7.1.3 Comparison with E-cognition

As seen before, E-cognition is the reference software for satellite image classification. We then classified the same image with this software in order to compare both methods, Fig. 7.3. We can see that this image is more noisy, which explains its overall accuracy of 66.8 percents.

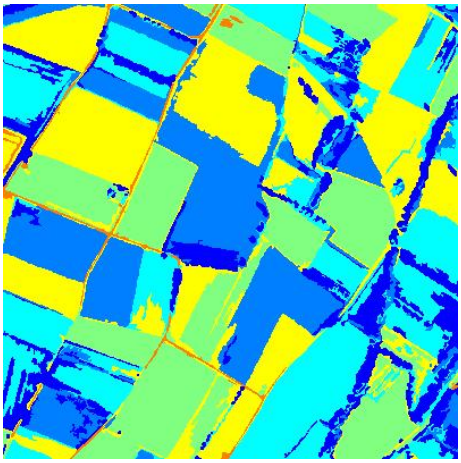


Figure 7.3: E-cognition classification

By extracting the prior segmentation of this software and using it with our classifier, we were able to obtain results of more than 75 percents of overall accuracy. Therefore, this software provides lower results for two main reasons:

- The method used by this software provides a single threshold parameter to tune the homogeneity of the segmentation. This threshold parameter acts both on the spectral and on the spatial dimension of the grown regions. Therefore all the segments will have a similar size, which does not reflect the territory reality, showing both small and large objects.
- No feature selection method is provided in E-cognition but the user selection. Therefore, a common use is to integrate all the features in the classification step. As mentioned above, this may interfere with an optimal classification, due to the weights of non-informative features.

7.2 Pixel vs. region-based classification

One of the most important point of our study was to evaluate the need of a prior segmentation. This part will try to answer to this question. The aim is twofold: to compare pixel-based classification versus region-based classification, and to compare the use of the different features.

We selected the Mahalanobis distance classifier, which is the best compromise between performance and computing time, to perform this comparison. For region-based classification we used the training set 2 previously

	Overall accuracy [%]	
Features	Pixel-based	Region-based
Intensity	72.5	80.0
Int.+Tex.	77.9	84.2
Int.+Shape	-	82.5
Int.+Tex.+Shape	-	84.3

Table 7.3: Pixel-based vs. region-based classification

described. We also needed a training set for pixel-based data. To ensure the best conditions, all the pixels belonging to the training set were considered as training pixels. The computation of the texture in the pixel-based classification was performed by computing the standard deviation of the intensity for each band in a 3 by 3 neighbourhood.

The results of this comparison are reported in Table 7.3. From these experiments we can make several remarks. The prior segmentation can improve the results of around 8 percents. Moreover there is less noise in the region-based classification, since we perform Markov random fields to create homogenous areas. Finally, training is easier when using region-based classification, since it is more practical for the user to select regions than to manually draw areas to train the classifier. We can then say that the prior segmentation is essential and necessary for a good classification.

Texture features By comparing the results with and without texture features, we can see that these features can improve the results of more than 4 percents, in both pixels and regions cases. However, textures features present a strong border effect in the pixel-based classification. Despite the improvement of the accuracy we do not suggest to use texture features in pixel-based classification. On the contrary, the use of texture features is strongly advised for region-based classification.

Shape features We can see that shape features improve the results of around 2 percents compared with an intensity-based classification. The difference between intensity-texture and intensity-texture-shape classification is not significant. For this classifier, these features do not seem to be useful, but it is difficult to generalize this conclusion to other cases. We then suggest to extract shape features, even if the results are not improved in some cases, since the feature selection method previously described can help selecting the most relevant features.

7.3 Influence of the prior segmentation

One of the most important settings for satellite image classification is the choice of the prior segmentation itself, since all the features are computed from the regions created by the segmentation.

From a theoretical point of view, we know that a too small number of components k when segmenting limits the maximum accuracy, since the geometry is not well respected. On the other hand, if k is too high, the real regions are over-segmented, which means that the features of the created areas are less significant, resulting in a harder task for the classifier. We then have to find the best compromise between segmentation and classification.

In this part, our aim is to determine which segmentation gives the best classification results. For this evaluation, we performed the same classification (with Mahalanobis distance) with different prior segmentations, that means different number of components. Of course, the training regions slightly vary between a segmentation to another, but we always tried to select nearly the same regions, so that the results of Fig. 7.4 could give us an idea of the effectiveness of the classification process.

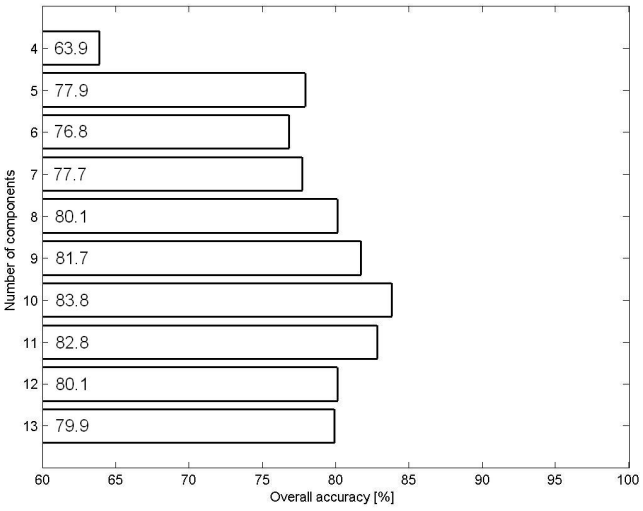


Figure 7.4: Influence of the prior segmentation

With $k = 2, 3$ it was not possible to perform a classification since these segmentations did not allow to select relevant training regions. We can see that from $k = 4$ to $k = 10$ the accuracy is increasing, principally because

of the improvement of the segmentations. From $k = 9$ to $k = 11$, the classification results are very good. From $k = 12$ we see that there is a slight decrease. This is not surprising, because these segmentations are visually very noisy. This confirms the fact that a too large number of components is not advised for classification, since the regions extracted from the over-segmented image are not significant.

We can not say that there is only one optimum value, since the overall accuracy depends on the training regions, we can then not affirm that $k = 10$ is the best segmentation for the following steps of the whole classification process. However, we can say that between $k = 9$ and $k = 11$ the classification will be very effective, which corresponds to the choice we made in the segmentation step. The results also prove the robustness of the classification step, since the choice of the prior segmentation is not critical. Finally we can say that the maximum accuracy of the segmentation is not very important. When the user has to select a segmentation, the first criterion must be the noise. This can be analysed visually and with the normalized entropy criterion.

7.4 Discussion

The results presented above show that the classification process is very effective. The results partially depend on the prior segmentation, a good choice of the number of components for the segmentation step is then essential. The segmentations that we visually selected as the best ones provide the best results in the classification process. We can then affirm that the choice of the prior segmentation is important, but not critical.

There are three types of errors in the resulting classification maps: the shadows, the prior segmentation errors and the noise in the homogenous parts. The first category of errors can hardly be corrected, since this is a problem that comes from the original image. The noise in the homogenous parts is due to an over-segmentation of some areas. This noise could be partially suppressed by a post-processing, but we also could decrease it with a better segmentation. Finally, prior segmentation errors can not be corrected in the classification step. We can then see that most of the inaccuracy comes from the segmentation step. However, this is not a weakness of the GHMRF algorithm, since this kind of errors appears in most of the segmentation methods.

If we compare our results with E-cognition classifications, we can see that our method is significantly more effective. The two main reasons are:

the advantages of the GHMRF model for the segmentation and the feature selection method which is not clearly explained in this software.

By comparing pixel-based with region-based classification, it seems obvious that the segmentation step added in our process increases the accuracy of our classification results. We can then affirm that the prior segmentation is very important for the classification of remote sensing images. Moreover the noise is strongly reduced with the Markov random fields.

Chapter 8

Conclusion

The aim of this project was to provide some classification tools for remote sensing images, and we can affirm that this aim was successfully achieved. For the segmentation step we demonstrated the effectiveness of the GHMRF algorithm, both from the qualitative and quantitative points of view. For the classification step, we proved the good performances of two classifiers, the support vector machine and the Mahalanobis distance. But beyond the simple considerations on the effectiveness of these algorithms, we presented a global process for remote sensing image classification.

The results show that this process seems to be very effective, exceeding the limits of the existing remote sensing classification softwares. However, due to the fact that ground-truth images are difficult to obtain, the robustness of this process was not fully demonstrated. Some further work should then be done in this direction.

Some improvements could also be added to this method. A post-processing could be inserted to suppress the noise in the homogenous regions. The segmentation step could also be improved in order to increase the potential of the classification step. However, due to the limits imposed by remote sensing acquisition systems, we do not think that exceptional progress could be provided to this classification process.

Bussigny, December 23, 2004.

X. Gigandet

List of Figures

List of Tables

Appendix A

Segmentation step evaluation: additional results

A.1 Kappa index

k	overall accuracy [%]	kappa index [%]
2	64.36	48.17
3	79.83	71.98
4	84.44	79.02
5	87.88	83.69
6	90.23	86.98
7	92.11	89.47
8	93.28	91.02
9	92.89	90.53
10	93.90	91.87
11	94.36	92.37
12	94.72	92.98
13	95.00	93.36

Table A.1: Segmentation: overall accuracy and kappa index

A.2 Producer's accuracy

	Producer's accuracy [%]					
k	wooded	farm. 1	farm. 2	op. land	scarce veg.	urban
2	17.48	17.50	89.29	97.53	2.36	2.60
3	14.21	68.78	96.12	95.36	59.81	12.15
4	48.79	85.16	91.47	96.22	65.20	23.16
5	60.45	86.12	95.05	95.45	76.50	34.27
6	79.38	97.07	92.09	95.43	78.88	35.12
7	72.94	94.49	94.43	96.66	94.05	34.63
8	81.57	96.14	95.66	96.76	92.74	33.10
9	83.32	92.77	95.48	96.13	95.78	30.11
10	85.75	95.11	94.85	97.41	94.92	46.59
11	80.13	96.08	95.67	96.34	96.01	71.69
12	79.96	95.93	96.67	96.83	93.66	82.81
13	86.86	96.69	95.24	96.95	94.39	83.77

Table A.2: Segmentation: producer's accuracy

A.3 User's accuracy

	User's accuracy [%]					
k	wooded	farm. 1	farm. 2	op. land	scarce veg.	urban
2	71.99	79.30	58.56	69.64	93.01	62.91
3	83.22	83.90	71.89	91.26	72.18	88.09
4	53.27	86.65	87.55	89.13	76.43	68.94
5	70.73	86.26	90.64	91.48	80.70	80.10
6	63.68	93.94	93.03	92.78	95.46	66.44
7	83.93	95.70	96.42	92.29	81.97	68.34
8	84.51	97.80	95.81	93.17	87.54	71.70
9	80.16	98.04	96.79	93.59	83.16	82.02
10	86.43	97.38	97.12	94.33	86.12	75.24
11	87.53	97.95	96.75	96.24	87.43	64.58
12	88.02	97.93	97.03	96.95	90.45	61.28
13	89.48	96.90	97.06	95.73	94.40	68.56

Table A.3: Segmentation: user's accuracy

Appendix B

Matlab Toolbox

The experiments were performed with the Matlab software. Several functions were created, which are briefly described in Table B.1.

`S2_pca.m`	Performs the principal component analysis
`S3_seg_kmeans.m`	K-means initialization for the segmentation
`S3_seg_ghmrf.m`	Runs the GHMRF segmentation algorithm
`S3_seg_criterions.m`	Computes the BIC and NEC criterions
`S3_seg_rsp.m`	Removes the single pixels from the segmentation
`S4_fex_feat_extract.m`	Extracts the features from the regions
`S6_class_fgmm.m`	Runs the FGMM classifier
`S6_class_knn.m`	Runs the K-nearest neighbours classifier
`S6_class_mahal.m`	Runs the Mahalanobis distance classifier
`S6_class_svm.m`	Runs the SVM classifier
`S6_class_crossvalid.m`	Performs the cross-validation
`S6_class_featsel_up.m`	Performs the sequential forward generation

Table B.1: Matlab functions

These functions are available on the additional CD-ROM *Satellite Image Segmentation and Classification: Matlab toolbox and user interface.*

Appendix C

User interface

A basic interface, based on the toolbox described in Appendix B, was created in order to simplify the task of the user. The main advantages are:

- Automatic data formatting: The user only needs to select the TIFF files and the data are ready to use.
- Easy tuning of the parameters for both segmentation and classification steps.
- Creation of the training set by user selection on the map of the segments.

This interface is available on the additional CD-ROM *Satellite Image Segmentation and Classification: Matlab toolbox and user interface.*

Bibliography

[1] L. A. Ruiz, A. Fdez-Sarria, and J. A. Recio, "Texture feature extraction for classification of remote sensing data using wavelet decomposition: A comparative study," *IAPRS*, vol. 35, 2002.

[2] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 610–621, November 1973.

[3] N. Sebe and M. S. Lew, "Wavelet based texture classification," *International Conference on Pattern Recognition*, vol. 3, pp. 947–950, 2000.

[4] S. Liapis, N. Alvertos, and G. Tziritas, "Maximum likelihood texture classification and bayesian texture segmentation using discrete wavelet frames," *International Conference in Digital Signal Processing DSP97*, 1997.

[5] V. Lakshmanan, V. DeBrunner, and R. Rabin, "Texture-based segmentation of satellite weather imagery," *International Conference on Image Processing*, vol. 2, pp. 732–735, 2000.

[6] B. R. Corner, R. M. Narayanan, and S. E. Reichenbach, "Application of principal component analysis to multisensor classification," *Proc. 27th Applied Imagery Pattern Recognition Workshop*, pp. 202–210, 1998.

[7] J. A. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixtures and hidden markov models," *Univeristy of Berkeley, ICSI-TR-97-021*, pp. 1–7, 1998.

[8] L. Amoura and W. Pieczynski, "Hierarchical markov fields and fuzzy image segmentation," *Proceedings Second IEEE International COnference on Intelligent Processing Systems*, pp. 154–158, 1998.

[9] P. Lanchantin and W. Pieczynski, "Unsupervised non stationary image segmentation using triplet markov chains," *Advanced Concepts for Intelligent Vision Systems ACVIS04*, 2004.

[10] J. Zhang and W. Modestino, "Maximum-likelihood parameter estimation for unsupervised stochastic model-based image segmentation," *IEEE Transactions on Image Processing*, vol. 3, pp. 404–420, July 1994.

[11] M. Neubert and G.Meinel, "Evaluation of segmentation programs for high resolution remote sensing applications," *Proceedings of the ISPRS Joint Workshop High Resolution Mapping from Space*, 2003.

[12] X. Gigandet, "Segmentation d'images satellites multispectrales par classification statistique," tech. rep., Swiss Federal Institute of Technology (Lausanne), 2004.

[13] W. Pieczynski, "Modèles de markov en traitement d'images," *Traitement du Signal, Presses Universitaires de Grenoble*, vol. 20, no. 3, pp. 255–277, 2003.

[14] M. B. Cuadra, *Atlas-based segmentation and classification of magnetic resonance brain images*. PhD thesis, Swiss Federal Institute of Technology (Lausanne), November 2003.

[15] C. J. L. C. W. Hsu, C. C. Chang, "A practical guide to support vector classification," 2003.

[16] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer, 1998.

[17] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford Clarendon Press, 1995.

[18] M. Pal and P. M. Mather, "Support vector classifiers for land cover classification," *Map India, Image Processing & Interpretation*, 2003.

[19] P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, "Evaluation of factors affectiong support vector machines for hyperspectral classification," *The American Society for Photogrammetry & Remote Sensing ASPRS*, 2004.

[20] A. Karlsson, "Classification of high resolution satellite images," tech. rep., Swiss Federal Institute of Technology (Lausanne), August 2003.

[21] R. G. Congalton and K. Green, *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*, ch. 5, pp. 45–50. Lewis, 1999.