Quality measures for the allocation of mixed pixels.

October 15, 1998

Ron Grazier
Thesis no. 447

Preface

This thesis is the result of a project I have carried out in order to complete my study for the department of Experimental Informatics for Technical Applications (EITA) at the 'Katholieke Universiteit Nijmegen' and obtain to my master's degree in computer science.

The project assigned to me is involved with remote sensing, an area in which the department, in cooperation with the Joint Research Centre in Ispra (Italy), previously has done lots of work. The goal of this project is to measure how well mixed pixels are detected by a segmentation program that was applied to an artificial satellite image.

I would like to thank my supervisor Theo Schouten for the time he put into helping me, Maurice Klein Gebbinck for guiding me through the first period of working on my thesis and Tricky and all the others who offered any kind of help or support. But special thanks go out to my parents who supported me during all these years during my study.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

In 1992 the European Commission (EC) introduced the McSharry subsidy system that was related to the surface areas of crops grown. From then on farmers could file a request for subsidy by filling in applications containing information about the size of areas and about what was grown on these areas. The people who made requests for subsidy could however "accidently" make errors in the application files, by writing down a larger surface area or by saying that a kind of crop is grown on a parcel that receives more subsidy than the one actually grown in that area, thus resulting in more subsidy than would be appropriate. To detect this kind of fraud the EC is forced to check a fraction (in practice 5 percent) of the requests for crop subsidies. The controlling of data given by subsidy applicants is a time consuming task if it is done by humans, therefore it is easier if it is done by computer. For checking the applications one must have access to precise information about where the different surface areas (segments) are located and of which ground cover type these areas consist. Therefore an automated system is applied that uses satellite images and divides the surface area shown into parcels having their own specific characteristics. To achieve this final step a segmentation program is used.

This report describes a method called Mixed Pixel Allocation Quality (MPAQ) for determining a quantitative value for the quality of detecting mixed pixels during a automated segmentation process of an image obtained by remote sensing. An image consists of picture elements called pixels. When these elements cover more than one object they are called mixed pixels else they are called pure. For this report we are especially interested in satellite images of agricultural fields, but MPAQ is also usable for other kind of images. The segmentation program used here is the best merge algorithm designed by Ron Schoenmakers [4].

To determine the quality of the resulting segmentation one has to have a correct segmentation for comparison. A segmentation made by human experts is considered to be correct. In spite of the segmentation process being laborious and very costly the resulting segmentation has a high probability of containing errors. The reason for this is that humans remain fallible and a segmentation made by different people will result in different segmentations making it hard to tell which man-made segmentation is correct. In this thesis a correct segmentation is obtained during the process of creating an artificial image by a method developed by Theo Schouten and Maurice Klein Gebbinck [9]. A possible comparison alone however is not enough, one also has to have a specific method for determining the correctness of the segmentation. The used methods for evaluating the accuracy of an image are the percentage correct and the kappa coefficient [5, 6].

This chapter describes remote sensing and its applications. Information about segmentation of an image and existing segmentation methods is given. Finally some information is given about the quantitative method in use at present. In chapter 2 the generation of an artificial image with realistic characteristics is described. In chapter 3 the best merge segmentation method is described. Chapter 4 shows how the edges, which indicate the boundaries between the different fields, are detected. The developed measure for determining the quality of the detection of mixed pixels and the implementation and evaluation of this method are shown in chapters 5, 6 and 7 respectively. Finally the conclusions about the evaluation of the developed method and possible future work are given.

1.1 Remote sensing

Retrieving information of an object without direct contact is called remote sensing. Examples of remote sensing imagery are images of the earth's surface by airplane or satellite, x-ray images made by a x-ray machine in a hospital, or radar images of the ocean floor. In this thesis remote sensing is applied to view areas of the earth's surface from an overhead perspective by means of satellites. These satellites present the information acquired in the electro-magnetic domain in the form of images.

There exist numerous definitions for remote sensing. A suitable definition for remote sensing as given in [5] is:

> Remote sensing is the practice of deriving information about the earth's land and water surfaces using images acquired from an overhead perspective, using electro-magnetic radiation in one or more regions of the electro-magnetic spectrum, reflected or emitted from the earth's surface.

This definition does not include the whole spectrum of remote sensing. It does however match the remote sensing applications viewed in this report.

1.2 Applications of remote sensing

There are numerous areas of interest for which remote sensing can be used. The applications vary from the more obvious areas such as geology or cartography to more surprising areas like archaeology.

A project set up by the European Community is the MARS project started in 1987 in order to develop an application for remote sensing in agricultural statistics. This project was set up in order to fulfill two goals. The first goal was to distinguish, identify and measure the surface of the different crops. This results in the possibility of estimating the crop production. The second objective is being able to make predictions about crop production estimates in time.

An other series of projects was started in 1972 by the launching of Landsat-1, a satellite orbiting the earth, depicting large areas of its surface in several areas of the electro-magnetic spectrum. The advantage of Landsat is that the images of scanned areas are stored digitally resulting in an increasing number of digital analysis applications of the made images. The images of the more recent Landsat-4 and -5 show the earth's surface in seven spectral bands as given in Table 1.1. The images used in this thesis are of the same format as those provided by these Landsat satellites.

Band	Wavelength (μm)	Description
1	0.45 - 0.52	Blue
2	0.52 - 0.60	Green
3	0.60 - 0.69	Red
4	0.75 - 0.90	Near-Infrared
5	1.55 - 1.75	Mid-Infrared
6	10.40 - 12.50	Thermal-Infrared
7	2.05 - 2.35	Mid-Infrared

Table 1.1: The Landsat spectral bands.

1.3 Classification and segmentation

To interpret an image one must be able to identify the different areas present. This can be done by classification of the satellite image. During the classification process the individual pixels are labeled with a ground cover class according to their spectral properties. Pixels with identical labels belong to the same ground cover class but do not necessarily belong to the same field. They may be spread over several different fields in the image belonging to the same kind of surface.

Due to internal variations in intensity within fields pixels belonging to the same ground cover class may differ in the spectral domain. The classification process can therefore label them differently. This leads to a final classification with a lot of noise and isolated pixels. A way of improving the classification is by performing segmentation before classification.

With segmentation an image is divided in its different elements (segments). This happens by grouping the individual pixels into regions which have similar spectral characteristics. The pixels grouped together not only have corresponding spectral characteristics but must be connected to the same area also. Areas of the same ground cover class are interpreted as separate segments. The result of segmentation is an image subdivided into regions without specifying the ground cover class to which each region belongs.

1.4 Existing segmentation methods

There are several methods for segmenting an image. The first method is examination of the images by human experts. The people who perform the segmentation must have a great amount of knowledge. The segmentation process also is very laborious and therefore costly. The problem in this case is that no person is infallible. Also segmentations of the same image made by different people will result in different segmentations which makes it difficult to determine which segmentation is correct. This means that the resulting segmentation very probably contains errors.

A possibility of reducing the chance of making errors is by checking on location to determine the features of the photographed area. The mapping of the measured ground truth data onto the image however is a difficult process to perform accurately. These tasks however still have to be performed by humans, and of course humans remain fallible and the results of the last mentioned tasks may still contain errors. Another problem with this method is that several areas may be difficult or even impossible for people to reach.

One may conclude that the manual segmentation methods are not reliable because of the possibility of errors being made by the people involved. This results in the conclusion that only automated segmentation systems may result in correct segmentations. Also the automated methods are less laborious and less costly.

The present automated segmentation methods can be divided into the following categories:

- Edge detection methods
- Region growing methods
- Map based or knowledge based methods
- Hybrid methods

Edge detection methods

Edge detecting methods locate strong discontinuities in the spectral domain between adjacent pixels in the image. These discontinuities are then compared with a predefined threshold value. If it is larger it will be considered to be an edge pixel representing part of a region boundary. The difficulty with the threshold is that if it is too high, little of the real edges will be found. If the threshold however is too low, small discontinuities, that are result of variations in fields, will be wrongly seen as edges and thus too many edges will be found.

By applying an edge thinning algorithm one pixel wide edges can be created. The edge pixels found can be joined together by edge linking methods to create closed structures. During edge linking the edge fragments that are no part of these closed structures are removed. This results in a structure where the individual segments are separated by one pixel wide boundaries. A well known example of edge detection is the one developed by Lacroix [17].

Region growing methods

The region growing method checks for similarity between adjacent pixels or groups of pixels. The compared pixels are then joined together in a group if they satisfy a similarity criterion, which is based on a threshold. The trouble with this threshold is that if this threshold is too high over-growing may occur. In this case fields differing in reality are wrongly merged into one region. On the other hand if the threshold is too low fields may be split in several region resulting in a fractured image containing many isolated pixels. Several methods for region growing have been developed (e.g. Beaulieu & Goldberg [14], Tilton [15]).

Map based or knowledge based methods

This method uses ancillary information to guide the segmentation process. This information can, for instance, be a digital map providing a reference about where segment boundaries probably occur. These map based methods have been used for both edge finding and region growing.

Hybrid methods

A hybrid segmentation method has been developed by Schoenmakers [4]. This method combines edge detection with region growing. Using an edge detection method based on the one developed by Lacroix edges in the image are detected. With an edge linking process the found edge fragments are connected together into closed polygons. Each polygon may comprise one or more segments of the image. Within each separate polygon region growing is performed to find the different segments and the edges separating those segments included in that polygon.

1.5 Problems in segmentation

A general problem in all the segmentation methods mentioned above is that all of them may result in errors. The obvious solution is to use the method which generally results in the best segmentation. The difficulty lies in determining which method this is.

To determine which method is the best under a given set of circumstances, one has to determine the quality of the segmentation. The quantitative measure itself can only be determined if one knows exactly what the correct segmentation looks like. This can never be achieved by using a real image because the segmentation of this image itself has to be determined one way or the other. And as said before there is no guarantee that the resulting segmentation is correct. This problem can only be solved if an artificial image, of which you know the true segmentation, is designed which can be processed by the various segmentation programs. Finally the quality of a segmentation can be determined by comparing the true segmentation with the resulting segmentation.

1.6 Current solutions

One of the methods for determining the segmentation quality is based on the method developed by Hubert Voogd for his masters thesis [1, 10]. In this method an artificial satellite image, from which a true segmentation can be derived, is first drawn. The artificial image is processed by the best merge segmentation program resulting in a test segmentation.

The program QMeasure is then used to determine the quality of the test segmentation made by comparing it with the true segmentation. The quality measure results in two different types of segmentation quality:

- Split quality: Splitting of a true segment into several segments.
- Merge quality: Merging of parts of different true segments into one segment.

The true segments can be split into several test segments. One method of calculating the split quality is determining the test segment with the most pixels of the true segment and calculating the covered area.

$$QSplitcov_i = \frac{ncov_{ik}}{nin_i}$$

Where:

$ncov_{ik}$: number of pixels that true segment i has in common with the found test segment k having the largest number of pixels in common with true segment i.

nin_i: number of pixels in true segment i.

If the true segment is not split the quality is 1. When the true segment is split heavily the quality approaches $\frac{1}{nin_i}$. This results in the following domain:

$$\frac{1}{nin_i} \leq QSplitcov_i \leq 1$$

An other method to calculate the quality is by using the entropy.

$$QSplitent_i = \sum_j p_{ij} \times log(p_{ij}) \text{ with } p_{ij} = \frac{ncov_{ij}}{nin_i}$$

The split entropy is 0 if the true segment is not split. When the true segment is heavily split the split entropy approaches $log(nin_i)$. This results in the fact that:

$$0 \leq QSplitent \leq \log(nin_i)$$

The split quality for the complete image can be calculated by using the quality per segment. This is done by taking the average of the quality per segment weighted by the number of pixels in the segment. This results in the following functions:

$$QOverallSplitcov = \sum_i \frac{QSplitcov_i}{nin_i}$$

$$QOverallSplitent = \sum_i \frac{QSplitent_i}{nin_i}$$

Pixels belonging to different true segments can be incorrectly merged into one test segment. In this case the roles of the true and test segmentation are reversed compared with their roles during splitting. This results in the following quality measure:

$$QMergecov_i = \frac{ncov'_{ik}}{nin'_i}$$

Where:

$ncov'_{ik}$: number of pixels test segment i has in common with true segment k which has the largest number of pixels in common with test segment i.

nin'_i: number of pixels in test segment i.

If no true segments are merged together to a test segment the merge quality is 1. If the test segment is a merging of numerous true segments the merge quality approaches $\frac{1}{nin'_i}$

$$\frac{1}{nin'_i} \leq QMergecov_i \leq 1$$

The merge quality also can be calculated by using the entropy.

$$QMergeent_i = \sum_j p_{ij} \times log(p_{ij}) \text{ with } p_{ij} = \frac{ncov'_{ij}}{nin'_i}$$

The merge entropy is 0 if the test segment is not the result of merging two or more true segments. If the test segment contains parts of many true segments the merge quality approaches $\log(nin'_i)$.

$$0 \leq QMergeent_i \leq \log(nin'_i)$$

The overall merge quality can be calculated by means of a method similar to the methods used in determining the overall split quality.

$$QOverallMergecov = \sum_i \frac{QMergecov_i}{nin_i}$$

$$QOverallMergeent = \sum_i \frac{QMergeent_i}{nin_i}$$

The overall quality of the segmentation is calculated by taking the average of the overall split and overall merge quality. Here both qualities are equally important.

$$Overallcov = \frac{QOverallSplitcov + QOverallMergecov}{2}$$

$$Overallent = \frac{QOverallSplitent + QOverallMergeent}{2}$$

Images obtained via remote sensing are built up by pixels whose sizes are large compared to the field sizes. As a result of this, many pixels partially cover more than one segment. The method described however does not take into account the existence of these mixed pixels. Because of this a significant part of the quality measurement of the image segmentation is ignored.

1.7 A new application

In the method developed by Hubert Voogd the quality of segmentation only takes into account which pure pixels are assigned to which segments. In most images however mixed pixels exist which consist of parts of two or more ground cover classes. This kind of pixels is not considered presently resulting in all kind of errors. Errors that may occur for example are the assignment of mixed pixels to one of its classes resulting in the incorrect assignment to a segment of the mixed pixel. The mixed pixel may also be assigned to a class not present in the pixel at all resulting in wrong assignment of a complete pixel and the image containing segments that do not exist in reality. Using knowledge about mixed pixels and their location could however result in the possibility of making segmentations up to sub-pixel level resulting in an segmentation with a more precise information about the fields shown in the image decreasing the made errors.

To include mixed pixels a new process must be constructed that determines the quality of allocation of the mixed pixels. This however is not possible working with best merge which, while segmenting the image, considers all pixels to be pure pixels. In order to still consider mixed pixels one can determine which pixels in the segmentation, being the result of the merging process, are likely to be mixed pixels.

Finally qualitative measures for comparing the location of the mixed pixels in the segmentation with those in the true segmentation have to be determined. By using these quality measures one obtains a quality value of the segmentation made.

Chapter 2

Constructing an artificial image

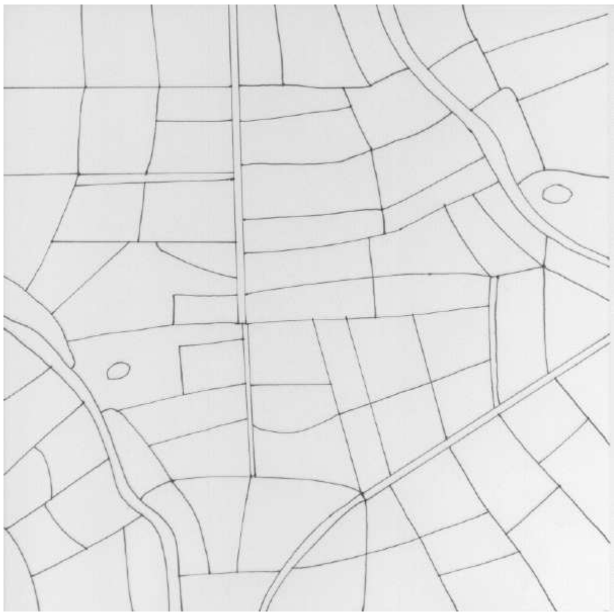
To determine the quality of detecting mixed pixels in an image using a segmentation method the found segmentation must be compared with the image's true segmentation. To get a true segmentation one has to have exact knowledge of how an area is subdivided. As explained in the previous chapter this can not been done by segmentation of a real image. It also is superfluous because if such a method already existed one would not have to put any effort into determining the segmentation quality of any method because a perfect method already exists. Therefore an artificial satellite image together with its true segmentation, containing information down to sub-pixel level, is created.

2.1 Generating an artificial image

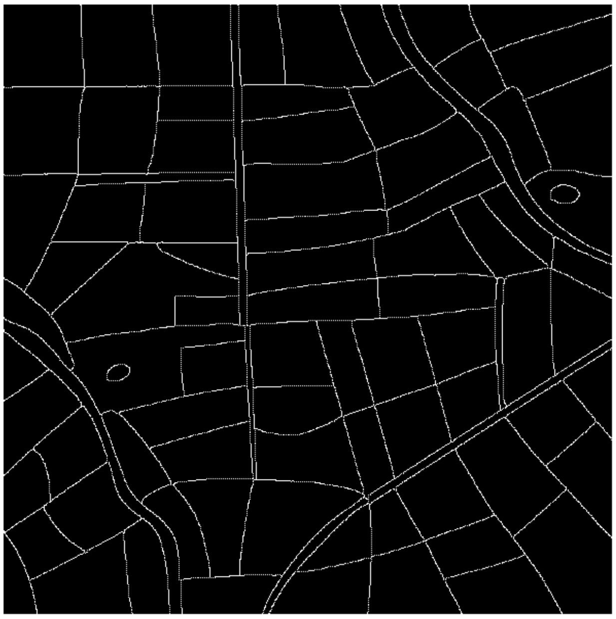
The first step in creating an artificial image is manually drawing an image on a white piece of paper using a black marker. In this drawing only the borders between the present fields, such as roads or agricultural fields, are shown. This drawing must then be scanned with a scanner, at a resolution of 100 dpi with 256 gray levels, and saved as a GIF image. With XV version 3.10 this GIF image is converted to PBM (raw), grayscale (see Figure 2.1(a)).

The image must then be converted to an edge image in which the different fields and field boundaries are obtained. This is done by thresholding and results in an binary image in which the fields are shown in black and the edges in white. It is saved in a ERDAS74 file format (see Appendix A.1) which also is the format of the images provided by the Landsat satellites. Pixels on the outer edge of the image are then removed. This is done to make sure that all the edges reach the perimeter of the image.

By edge linking one pixel wide gaps between edges are filled. This is done to provide closed polygons. All the edge fragments that are not part of these polygons are removed (pruned). The polygons with a size smaller than a minimum polygon size, given by the user, are removed also. Edge thinning finally is used to obtain an image with one pixel wide edges represented by closed polygons. During the process of edge linking a two pixel wide edge is created on the outside of the image to be sure that all the regions on the image's edge are inclosed by closed polygons as well (see Figure 2.1(b)).



(a) Scanned drawing.



(b) Linked image.

Figure 2.1: Scanned drawing and Linked edge file.

The linked edge file is converted into a REGINF file containing one channel. Blob coloring of the black pixels belonging to segments is then used to label each individual pixel with a field number indicating the segment it is part of. This results in a segmentation file in which a unique region number is assigned to each segment. This file is saved in a REGINF file format (see Appendix A.2).

A map file must then be made manually. This map file contains information about which region, indicated by its region number, contains which ground cover class. Each ground cover class is indicated by a specific number. Not only the regions can be assigned to a ground cover class, edges can be assigned to a ground cover class as well. The map file is an ASCII file consisting of two columns. The first containing the field number and the second the number of the related ground cover class.

Before the generation of the artificial image, templates are constructed for each ground cover class (see Schouten and Klein Gebbinck [10] for a detailed description of their construction). These templates are small images, of 100 by 100 pixels, that reflect the texture and statistics, such as mean and covariance, of real fields taken from satellite images. By using textures of the used ground cover classes one generates a more realistic image.

The image is then reduced in size by using a predefined resolution. Using the resolution r, blocks of r by r pixels are used to create one pixel in the artificial image. These created pixels may cover several ground cover classes making them mixed pixels. The intensity values of each artificial created pixel is taken as a linear mixture of the intensity values of the ground cover classes contained in it. The intensity can be written as:

$$\overline{int}_{x,y} = \sum_c fr_{x,y}(c) int_{x \ mod \ xd, y \ mod \ yd}(c)$$

where $\overline{int}_{x,y}$ is the intensity of the new pixel on position (x, y), $fr_{x,y}(c)$ is the fraction of pixel (x, y) covered by ground cover class c, xd and yd are the dimensions of the templates and $int_{x,y}(c)$ is the intensity of the of the pixel (x, y) of the template of class c. If the edges were not assigned to a ground cover class the edge pixels are ignored during calculation of these values.

The image is reduced in size before coloring instead of the other way around. This is done to obtain an image with identical color variation as the templates and thus as real surface areas. The effect of this method of reducing and coloring is that pixels in a larger resolution do not simply have an intensity value that is the average of pixels in a larger resolution located on the same position.

The true segmentation, constructed during the reduction of the image's resolution, does not only contain information of which pure pixels belong to which region but also information about mixed pixels that may have formed during resizing and is saved in the REGIN3 file format (see Appendix A.3). An example of a resulting artificial image is shown in Figure 2.2.

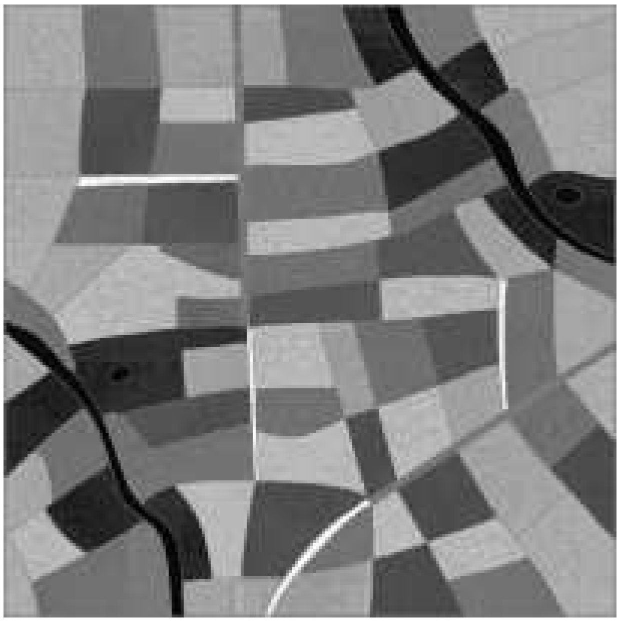
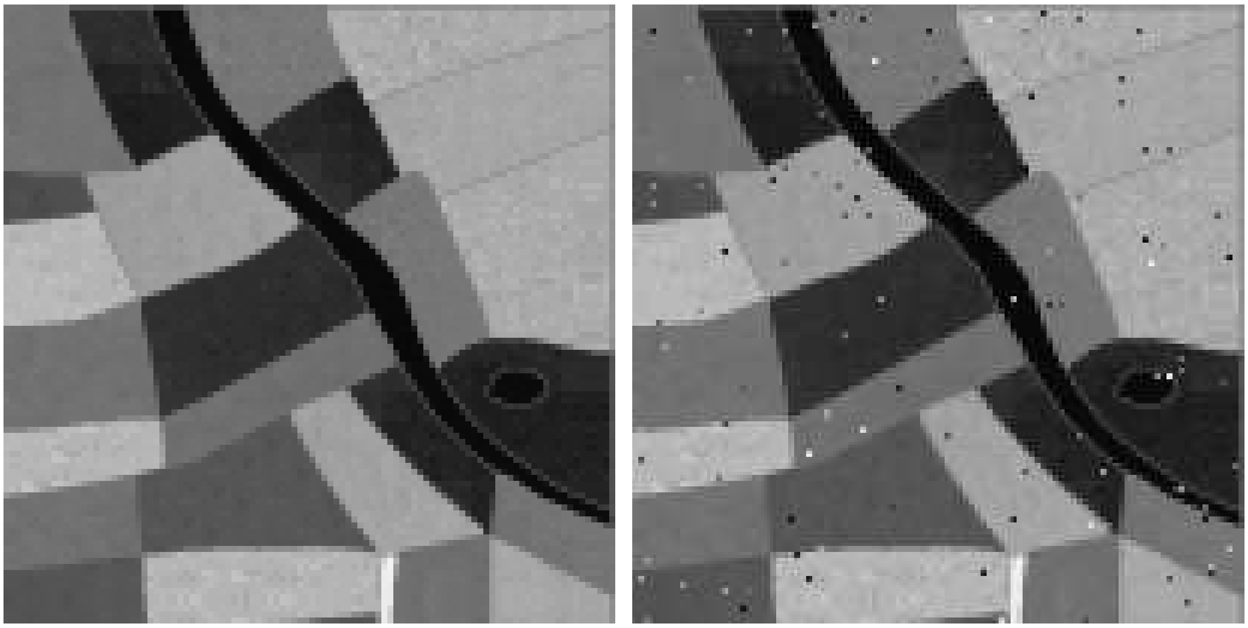


Figure 2.2: Artificial image

In this final step a number of isolated mixed pixels can also be added. Therefore a number of randomly chosen pixels are replaced by pixels partly existing of the ground cover class of an adjacent field and, for at least 75 percent, of a ground cover class differing from all its neighbors (see Figure 2.3).



(a) Top right artificial image.

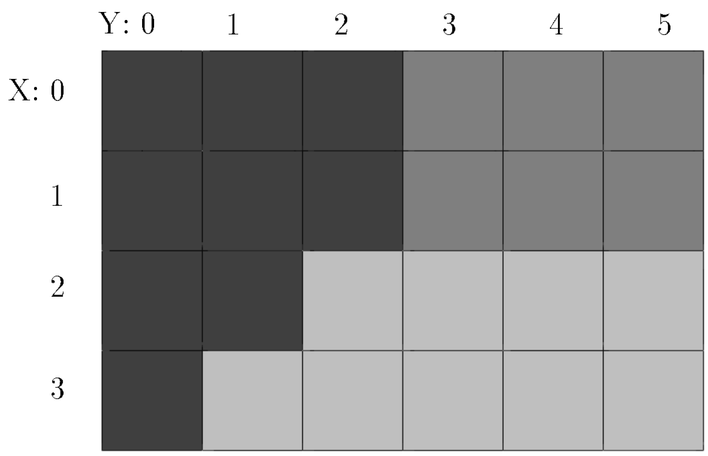
(b) Top right artificial image with 0.1 % isolated pixels.

Figure 2.3: Addition of isolated pixels.

2.2 Used file types

During the process of creating an artificial image several file types are used. For the edge images and the artificial image the ERDAS74 file type is used and for the segmentation files the REGINF or REGIN3 file type. Both file types are described in more detail in Appendix A.

In the ERDAS74 file type the actual image information is saved per individual pixel. Each pixel's intensity value is saved per row contained in the image. Per row the information is saved per channel. An example is given in Figure 2.4 where each color gray is a different kind of surface and the image contains two channels. In this example $v_{x,y,c}$ stands for the intensity value the pixel located on position (x, y) has in channel c.

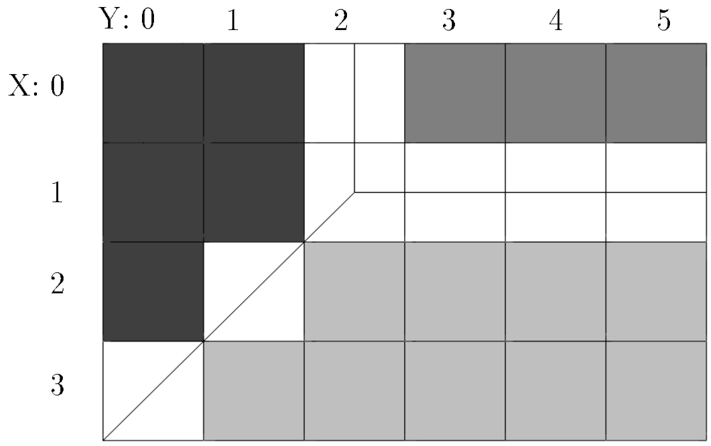


$$
\begin{array}{cccccc}
v_{0,0,1} & v_{0,1,1} & v_{0,2,1} & v_{0,3,1} & v_{0,4,1} & v_{0,5,1} \\
v_{0,0,2} & v_{0,1,2} & v_{0,2,2} & v_{0,3,2} & v_{0,4,2} & v_{0,5,2} \\
v_{1,0,1} & v_{1,1,1} & v_{1,2,1} & v_{1,3,1} & v_{1,4,1} & v_{1,5,1} \\
v_{1,0,2} & v_{1,1,2} & v_{1,2,2} & v_{1,3,2} & v_{1,4,2} & v_{1,5,2} \\
v_{2,0,1} & v_{2,1,1} & v_{2,2,1} & v_{2,3,1} & v_{2,4,1} & v_{2,5,1} \\
v_{2,0,2} & v_{2,1,2} & v_{2,2,2} & v_{2,3,2} & v_{2,4,2} & v_{2,5,2} \\
v_{3,0,1} & v_{3,1,1} & v_{3,2,1} & v_{3,3,1} & v_{3,4,1} & v_{3,5,1} \\
v_{3,0,2} & v_{3,1,2} & v_{3,2,2} & v_{3,3,2} & v_{3,4,2} & v_{3,5,2}
\end{array}
$$

Figure 2.4: Example ERDAS74 information

Two kind of region information files are used. The REGINF format that can only handle pure pixels and the REGIN3 format that can handle both, pure and mixed pixels. The data in a REGIN3 file is shown per region. First the list of pure pixels contained in the region is shown by means of run length encoding. In run length encoding each adjacent area of pixels on a row belonging to the segment is saved by writing the coordinates of the first pixel and the number of pixels belonging to that area. The end of the list of pure pixels in a region is marked by the value $(ushort-1)$. Thereafter the mixed pixels, covering part of the region, are shown by writing the coordinates of the mixed pixels together with the fraction of the pixel belonging to the region. In this case however first the y and then the x coordinate is written. The end of the list of mixed pixels is indicated by the value $(ushort-2)$.

The main difference between REGINF and REGIN3 is the lack of information about mixed pixels in the REGINF file type. Therefore only an example of a REGIN3 file is given in Figure 2.5 where each color gray is a different kind of surface and the mixed pixels are white.



$(0,0,2)$ $(1,0,2)$ $(2,0,1)$ $(ushort-1)$
$(2,0,f)$ $(2,1,f)$ $(1,2,f)$ $(0,3,f)$ $(ushort-2)$
$(0,3,3)$ $(ushort-1)$
$(2,0,f)$ $(2,1,f)$ $(3,1,f)$ $(4,1,f)$ $(5,1,f)$ $(ushort-2)$
$(2,2,4)$ $(3,1,5)$ $(ushort-1)$
$(2,1,f)$ $(3,1,f)$ $(4,1,f)$ $(5,1,f)$ $(1,2,f)$ $(0,3,f)$ $(ushort-2)$

Figure 2.5: Example REGIN3

Chapter 3

Best merge

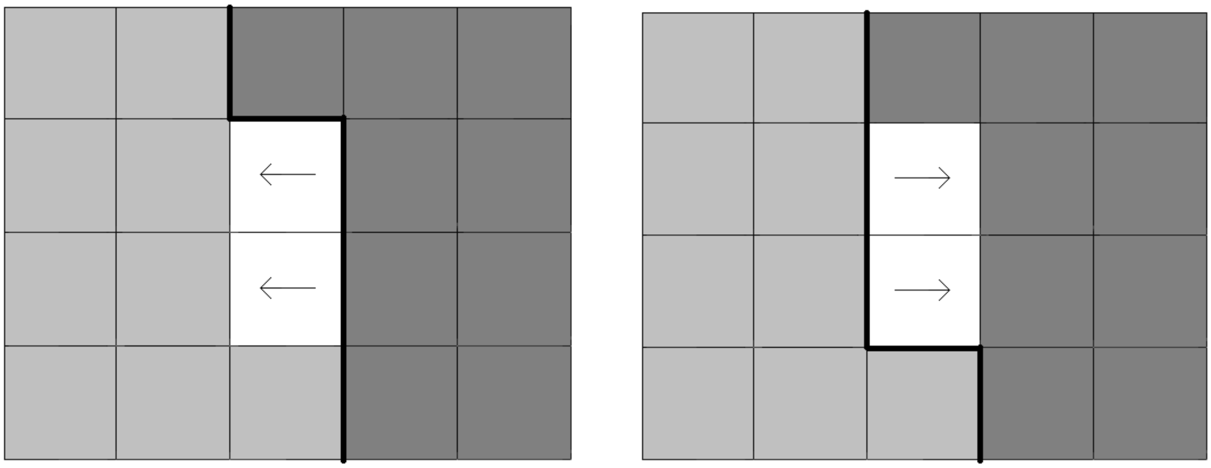
In this chapter the segmentation method best merge is described. The best merge algorithm performs region growing starting with all pixels considered as an individual segment. First neighboring segments with the same spectral value are merged. The merging costs or merging distance for adjacent segments are calculated by using a predefined cost function or similarity criterion. The segments with the lowest merging costs are merged. This continues until the minimal merging distance does not satisfy a stopping criterion. The stopping criterion indicates the maximal merging distance for which merging may be carried out.

3.1 Order dependence

The process of merging is an iterative process in which the output of an iteration will be used as input for the next iteration. This makes it possible that the processing of segments in one phase of the merging process influences the process in the next phases. The merging process is therefore susceptible to the problem of order dependence. Order dependence is defined by Zucker [16] as:

> Information acquired in earlier stages of processing can (will) be used to influence the processing of later stages.

This implies that the choice of the first segments to be merged influences the segments that will be merged next. Thus the order in which segments are merged may have an affect on the final outcome of the merging process. An example of the order dependence problem is shown in Figure 3.1. If the left region is grown first the central pixels will be merged with the left segment (see Figure 3.1(a)). If however the right segment is chosen to be merged first then the central area will be merged with with right segment (see Figure 3.1(b)).



(a) Left area merged first. (b) Right area merged first.

Figure 3.1: Example of order dependence.

3.2 Best merge algorithms

To avoid the order dependence problem only the adjacent segments with the greatest similarity are merged together. In each iteration the best merge is thus performed. A problem in this case however is when more than one segment pairs have the same minimal merging cost. To solve this case of order dependence all the segments that are involved are merged simultaneously.

Several algorithms for best merge exist. Two algorithms are developed by Tilton [15] and Beaulieu & Goldberg [14]. A method that is derived from those two algorithms is the best merge algorithm designed by Ron Schoenmakers [5]. A definition of the best merge algorithm is given in Algorithm 3.1.

1. Define a (dis)similarity criterion for pairs of regions
2. Define a stopping criterion
3. While stopping criterion is not met Do
   (a) Compute for all adjacent pairs of regions their similarity value
   (b) From these merge the single most similar pair

Algorithm 3.1: Best merge principle

3.2.1 Tiltons algorithm

Tiltons algorithm is called the Iterative Parallel Region Growing algorithm (IPRG) (see Algorithm 3.2). According Tilton, Algorithm 3.1 results in an large computational load. Tilton approaches the best merge algorithm by merging sets of segments in parallel at each iteration.

1. Initialize with one pixel per segment.

2. Compute a similarity measure for each pair of spatially adjacent segments.

3. Apply a merge control to all pairs of spatially adjacent regions based on merge constraints exploiting the similarity value as well.

4. Merge pairs of regions which have passed the merge control.

5. Check for convergence. If convergence is not reached, the procedure is iterated.

Algorithm 3.2: Tiltons algorithm

Tilton recursively defines sets of sub-images in relation with a specific region. The level 0 sub-image is the empty set. A level 1 sub-image related to a region is the region itself. A level 2 sub-image related to a region is the level 1 sub-image related to that region and all the adjacent regions. A level n sub-image related to a region is a level (n-1) sub-image related to that region and all the regions adjacent to that sub-image. In Figure 3.2 an example is shown. The region labeled R is a level 1 sub-image related to itself. The regions labeled R, and 2 form the level 2 sub-image related to region R, and the regions labeled R, 2 and 3 form the level 3 sub-image related to region R.

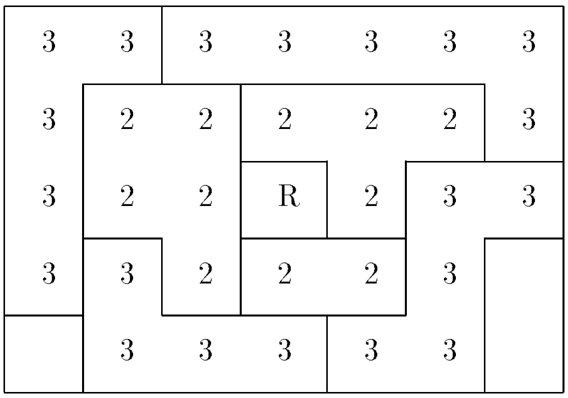


Figure 3.2: level n sub-images

Tilton states that the following scheme approximates the one merge per iteration approach. At each iteration the image is divided into a set of possibly overlapping sub-images.

The best merge algorithm is then performed within each sub-image. After each iteration the sub-images are defined again.

Tilton defines merge constraint levels as follows. Constraints level n signifies that merging is constrained to the best merge within the union of the level n sub-images with respect to each of the potential merging pairs.

3.2.2 Beaulieu & Goldbergs algorithm

Beaulieu & Goldberg call their best merge algorithm the Hierarchical Stepwise Optimization Algorithm (HSWO). The algorithm is given in Algorithm 3.3. It begins by making an initial partition $P_0 = \{S_1, S_2, ..., S_n\}$ by merging pixels with identical spectral values. For each segment S_i their neighbors B_i and the parameters D_i describing S_i, for example the segment mean intensity and segment size, are calculated. And for each segment S_i the merging costs with their neighbors are calculated.

In step 2(b) the segment pair with the minimum merging cost $C_{u,v}$ is determined. In step 2(c) the new segment that is result of merging segment S_u and S_v is calculated. In step 2(d) the new segment S_m is added to, and the merged segments are removed from the old partition resulting in a new partition. In steps 2(e) and step 2(f) the parameters D_i describing the new segment S_i and its neighbors are calculated. In step 2(g) for each neighbor of the new segment their neighbors are determined again. Finally in step 2(h) the merging costs of all the neighboring segments are updated. If segment pairs remain that can be merged the algorithm continues in step 2(a) else the algorithm stops.

3.2.3 Schoenmakers algorithm

In the algorithms created by Tilton and Beaulieu & Goldberg a problem arises when more than one group of segments have the same minimum merging cost. The problem with this is deciding which segments should be merged together. In Schoenmakers algorithm the groups of segments with the same minimum merging cost are all merged together at the same time. This merging however does not necessarily result in one segment but may result in two or more new segments.

The algorithm (see Algorithm 3.4) starts with an initial segmentation $S = \{S_1, S_2, ..., S_n\}$ where each individual pixel is considered to be a separate segment. The threshold value T is set and the segments with the same intensity value M_i are merged together. Finally the pairs of adjacent segments with a merging cost smaller than the threshold are determined where $C_{i,j}$ is the merging cost of segments S_i and S_j and CS the found pairs.

In step 2(a) the segment pairs S_u, S_v with minimum merging costs $C_{u,v}$ are calculated. In step 2(b) for every pair S_u, S_v found in step 2(a) the neighbors are determined. In step 2(c) all the pairs found in step 2(a) and 2(b) are removed resulting in a new set of segment pairs CS'. In step 2(d) all pairs having minimum costs are merged together in parallel. If segment pairs have segments in common then they are merged together into one segment. In step 2(e) the segments that were merged together are removed from S resulting in S'. In step 2(f) the segments created by merging are new segments (S_{new}) and these are added to

1. Initialization
   (a) $P_0 = \{S_1, S_2, ..., S_n\}$
   (b) $k = 0$ and $m = n$
   (c) calculate D_i and B_i, $\forall S_i \in P_0$
   (d) calculate $CS = \{C_{i,j} | S_j \in B_i \text{ and } i > j\}$
2. Merge the two most similar segments
   (a) $k = k + 1$ and $m = m + 1$
   (b) $C_{u,v} = MINIMUM(CS)$
   (c) $S_m = \{S_u, S_v\}$
   (d) $P_k = P_{k-1} + \{S_m\} - \{S_u, S_v\}$
   (e) calculate D_m from D_u and D_v
   (f) $B_m = B_u + B_v - \{S_u, S_v\}$
   (g) $\forall_{S_j \in B_m} [B_j = B_j + \{S_m\} - \{S_u, S_v\}$
   (h) $CS = CS + \{C_{m,j} | S_j \in B_m\} - \{C_{i,j} | i, j = u \text{ or } v\}$
3. Stopping condition

   Stop if no mergers are required. Otherwise go to step 2.

Algorithm 3.3: Beaulieu & Goldbergs algorithm

S' resulting in set S''. Then the mean intensities of the new segments are calculated in step 2(g) and the merging costs of new segments with their neighbors is determined in step 2(h). In steps 2(i) to 2(k) the new pairs of segments are determined where a new segment is part of these pairs and the merging cost of that pair is smaller than the threshold. In step 2(l) the new segment pairs CS_{new} are added to the current set of segment pairs CS'. In step 2(m) S and CS are updated to contain the new set of segments building up the image and the segment pairs that satisfy the merging threshold. If pairs exist that satisfy the merging threshold the algorithm continues at step 2(a) else the algorithm is ready. Because of the consistent simultaneous merging in case of ambiguity the algorithm is called Simultaneous Order Independent Iterative Merging (SOIM).

3.2.4 Merging of isolated pixels

Isolated pixels still exist in the image after the merging process. These isolated pixels have spectral values different from their neighbors. Adding the isolated pixels to one of its neighbors weakens the constraint of computing pure pixels. To overcome this problem three knowledge rules (see Algorithm 3.5) were developed in which an attempt is made to add isolated pixels to a region. In this algorithm regions are considered as neighbors if they are 4-connected. A pixel is 4-connected to pixel p if it is vertically or horizontally adjacent to p. If a pixel is vertically, horizontally or diagonally adjacent to pixel p it is 8-connected to p.

In Figure 3.3 examples are shown for every knowledge rule. For rule 1, isolated pixel P is 4-connected by segments A,B,C and D. When segment A is the spectrally nearest segment to P, then P will be merged with that segment. For rule 2, pixel P is 4-connected on three places with segment A and therefore is merged with that segment. For rule 3, pixels P and Q are 4-connected and P is 4-connected with segments A,B,C and D. When A is the spectrally nearest segment to P, with isolated pixel Q viewed as a small segment, then P will be merged with A.

3.3 Cost functions

In order to calculate the merging costs for adjacent pairs of regions a cost function must be defined.

3.3.1 Tiltons function

Tilton defined a dissimilarity criterion. The region pairs with the lowest dissimilarity are merged together. The dissimilarity criterion is based on minimizing the increment in variance normalized mean square error. The mean square error of channel k is:

$$MSE_k = E[(D_k - D_k^r)^2] \cong \frac{1}{N-1}\sum_{p=1}^{N}(D_{kp} - D_{kp}^r)^2 \qquad (3.1)$$

1. Initialization
   (a) $S = \{S_1, S_2, ..., S_n\}$
   (b) set threshold value T
   (c) merge S_i and S_j if $M_i = M_j$ and S_i and S_j are adjacent.
   (d) $CS = \{\langle S_i, S_j\rangle | S_i$ and S_j adjacent and $i < j$ and $C_{i,j} < T\}$
2. Merge the most similar segments
   (a) $CS_{min} = \{\langle S_u, S_v\rangle | \langle S_u, S_v\rangle \in CS$ and $C_{u,v} = MINIMUM\}$
   (b) $CS_{nbr} = \{\langle S_u, S_k\rangle | (\langle S_u, S_v\rangle \in CS_{min}$ or $\langle S_v, S_u\rangle \in CS_{min})$ and $\langle S_u, S_k\rangle \in CS$ and $k \neq v\} \cup \{\langle S_k, S_u\rangle | (\langle S_u, S_v\rangle \in CS_{min}$ or $\langle S_v, S_u\rangle \in CS_{min})$ and $\langle S_k, S_u\rangle \in CS$ and $k \neq v\}$
   (c) $CS' = CS - CS_{min} - CS_{nbr}$
   (d) Simultaneously create a new set S_{new} of all possible new segments S_m, each from K segment pairs $\langle S_x, S_y\rangle$ such that:

   $$P_m : \{e | e \in \bigcup_{1}^{K} [P_x \cup P_y] \text{ such that } \langle S_x, S_y\rangle \in CS_{min}\}$$

   where $P_i = \{e_1, ..., e_n\}$ is the set of pixels contained in S_i and the K segment pairs are spatially contiguous.
   (e) $S' := S - S_i$ where S_i is one component of a pair $\langle S_p, S_q\rangle$ and $\langle S_p, S_q\rangle \in CS_{min}$
   (f) $S'' := S' \cup S_{new}$
   (g) $\forall S_i \in S_{new}$: calculate M_i
   (h) $\forall S_i \in S_{new}$: calculate $C_{i,j}$ where $S_j \in S''$ and S_i, S_j adjacent.
   (i) $CS_{new1} = \{\langle S_i, S_j\rangle | \forall S_i \in S_{new} : S_j \in S'', i < j, C_{i,j} < T$ and S_i, Sj adjacent $\}$
   (j) $CS_{new2} = \{\langle S_i, S_j\rangle | \forall S_j \in S_{new} : S_i \in S'', i < j, C_{i,j} < T$ and S_i, Sj adjacent $\}$
   (k) $CS_{new} = CS_{new1} \cup CS_{new2}$
   (l) $CS'' = CS' \cup CS_{new}$
   (m) $CS := CS''; S := S'';$
   Repeat until stopping criteria fulfilled.
3. Stopping condition

   $CS = \emptyset$

Algorithm 3.4: Schoenmakers SOIM algorithm

1. If isolated pixel P is surrounded by four different segments then assign P to the spectrally nearest segment.
2. If isolated pixel P is surrounded by three pixels belonging to one segment then assign P to this segment.
3. If isolated pixel P is surrounded by three different segments and the spectral distance between P and the remaining isolated pixel Q is greater than the minimum distance between P and one of the adjacent segments then assign the pixel to that segment.

Algorithm 3.5: Knowledge rules for merging isolated pixels based on 4-connectedness

A	A	B	B	B
A	A	B	B	B
A	A	P	C	C
D	D	D	C	C
D	D	D	C	C

(a) Rule 1

A	A	A	B	B
A	A	A	B	B
A	A	P	C	C
A	A	A	C	C
D	D	D	C	C

(b) Rule 2.

A	A	B	B	B
A	A	B	B	B
A	A	P	C	C
D	D	Q	C	C
D	D	D	C	C

(c) Rule 3.

Figure 3.3: Example of the knowledge rules

where D_k and D_k^r are the data values of the k-th channel of the original and region mean images. D_{kp} and D_{kp}^r are the values of the p-th pixel of the k-th channel of the original and reconstructed image. E is the expected value and N the total number of pixels in the image. A region mean image is formed by substituting the mean vector of each region for the multi-spectral radiance values of each pixel in the region.

The variance normalized mean squared error for the k-th channel ($NMSE_k$) is:

$$NMSE_k = \frac{MSE_k}{\sigma_k^2} \tag{3.2}$$

where σ_k^2 is the variance of the k-th band. For a pair of adjacent regions $\Delta NMSE_k$ is the difference in $NMSE_k$ after merging the region pair and the new region mean image is formed and compared with the original image. The difference in $NMSE_k$ is defined as:

$$\Delta NMSE_k = \frac{MSE_k^c - MSE_k}{\sigma_k^2} \tag{3.3}$$

where MSE_k^c is the mean squared error when regions i and j are merged, and MSE_k the mean squared error before regions i and j are merged. This can be rewritten as:

$$\Delta NMSE_k = \frac{n_i(\overline{D}_{ki} - \overline{D}_{kij})^2 + n_j(\overline{D}_{kj} - \overline{D}_{kij})^2}{(N-1) \cdot \sigma_k^2} \tag{3.4}$$

where n_i and n_j are the number of pixels in region i and j before merging. N is the number of pixels in the image. $\overline{D}_{ki}$ and $\overline{D}_{kj}$ are the mean values of channel k for regions i and j before merging. $\overline{D}_{kij}$ is the mean value of channel k for region resulting after merging regions i and j.

3.3.2 Beaulieu & Goldbergs function

Beaulieu & Goldberg consider an image as a two dimensional function $f(x,y)$ where $(x,y) \in I$ with I the image dimensions. A partition P divides the image into n regions, $S_1, S_2, ..., S_n$. $f_i(x,y)$ indicates the pixel values for region S_i. Now $f_i(x,y)$ can be written as:

$$f_i(x,y) \approx r_i(x,y) \tag{3.5}$$

where $r_i(x,y)$ is a polynomial approximation of the pixel function $f(x,y)$.

The approximation error for each region can now be calculated by the sum of the squared deviations:

$$H(S_i) = \sum_{(x,y)\in s_i} (f(x,y) - r_i(x,y))^2 \tag{3.6}$$

By defining $r_i(x,y) = \mu_i$, where μ_i is the mean value of region i, the segment approximation error of the segment can be rewritten as:

$$H(S_i) = \sum_{(x,y)\in s_i} (f(x,y) - \mu_i)^2 \tag{3.7}$$

The increment of error during the merging process must be minimized. The increment can be calculated by using the stepwise criterion defined as:

$$C_{ij} = H(S_i \cup S_j) - H(S_i) - H(S_j) \tag{3.8}$$

The segment approximation error can be written as:

$$C_{i,j} = \frac{N_i \cdot N_j}{N_i + N_j}(\mu_i - \mu_j)^2 \tag{3.9}$$

where N_i is the number of pixels in region i.

3.3.3 Schoenmakers function

The cost function of Schoenmakers does not take the size of the regions into account. This leads to the advantage that the regions are only merged as result of their spectral values and not of their size.

The cost function used in this algorithm is the Euclidean distance between adjacent regions i and j.

$$C_{i,j} = \sqrt{(\mu_{1i} - \mu_{1j})^2 + ... + (\mu_{ci} - \mu_{cj})^2} \tag{3.10}$$

where c is the number of channels and μ_{aj} the spectral value of channel a in segment j. If only one channel exists the equation is:

$$C_{i,j} = |\mu_i - \mu_j| \tag{3.11}$$

where μ_j the average spectral value of segment j.

3.4 Stop criteria

To stop the merging process a stop criterion must be defined. Tilton and Beaulieu & Goldberg both define a stop criterion indicating when the merging process has finished.

3.4.1 Tiltons criterion

Tilton defines a merging threshold indicating the maximal dissimilarity value for which continuance is allowed. If the dissimilarity value is higher than the threshold the merging process will stop. The threshold value is defined as:

$$Tresh = \frac{ME - NMSE}{\frac{1}{2}NR \cdot FR} \tag{3.12}$$

where ME is a user defined value, NR the number of current regions, FR the estimated fraction of region participated in merges at each iteration and

$$NMSE = \sum_{i=1}^{N1} \sum_{j=1}^{N2} (max_{k=1}^{m}(NMSE_{kij})) \tag{3.13}$$

where $N1$ is the number of rows, $N2$ the number of columns and m the number of channels.

3.4.2 Beaulieu & Goldbergs criterion

Beaulieu & Goldberg define the stop criterion as the number of merged regions in relation to the initial number regions. Another criterion used is a set value for the maximal approximation error.

3.4.3 Schoenmakers criterion

The SOIM-algorithm also continues while the stopping criterion is not met. This stopping criterion is an upper limit for the merge cost. The value of this threshold is entered as a value given by the user.

3.5 Evaluation of the best merge methods

A major problem that arises in the best merge methods is the problem of order dependence stating that when a different order of processing of pixels is chosen the final segmentation may not always be the same (see Section 3.1).

For the merging methods described in this chapter the order dependence problem is solved by merging the segment pair with the lowest merging costs. When however several region pairs with the same minimum merging costs exist the methods proposed by Tilton and Beaulieu & Goldberg do not give a solution to this problem. In this case the region pair to be merged is selected randomly. This violates the demand of order independence because the resulting segmentation still may differ to the one being the result of selecting a different pair. A different attempt in avoiding the order dependence problem is selecting the pair with the lowest region number. This still is not satisfactory, if the same image for example is rotated a different segment pair might be selected resulting in a different outcome. The SOIM algorithm merges all the pairs having the same minimum merging costs at the same time fulfilling the order independence claim. Therefore this best merge method is the one used in the remainder of this thesis.

Chapter 4

Edge detection

Another method for image segmentation is finding the boundaries between regions by using edge detection methods. Edge detection methods search for large discontinuities in the image's intensity. Edge detection can be combined with region growing methods. First the image is divided into segments inclosed by closed polygons formed by found edge structures. Each found segment can be segmented further by using the region growing method. Because the size of regions on which region growing is applied are much smaller the execution time is reduced significantly. Each inclosed region is merged independently which can be done in parallel by using a cluster of workstations. We are especially interested in the area of mixed pixels. And because the found edge pixels are located on region boundaries, edge detection can be used to allocate the mixed pixels that are situated on the boundaries of the regions.

4.1 Edge detection

By calculating the edge direction and edge magnitude for each pixel a likelihood of being an edge pixel (LBE) can be computed. Using this information an image with one pixel wide edge structures can be constructed.

The edge detection used in this thesis is an edge detection method based on that developed by Frei & Chen [18] and modified by Schoenmakers [4]. The calculation of the edge direction and magnitude of an image sub-area W, that is considered as a set of n^2 pixels (here $n = 3$), is calculated per separate channel using windows of n by n pixels. Each window can be seen as a vector in the n^2-dimensional vector space. Frei & Chen defined the nine orthogonal basis vectors $[T_1, ..., T_9]$ of which the vectors $[T_1, ..., T_4]$ span the edge subspace (see Figure 4.1).

The angle Θ in n^2-space between W and the projection onto the edge subspace can be considered as:

$$\Theta = arccos\sqrt{\frac{\sum_{i=1}^{4}(W \cdot T_i)^2}{\sum_{i=1}^{n^2}(W \cdot T_i)^2}} \tag{4.1}$$

The larger Θ is the poorer the fit is between W and an element of the edge subspace. To

1	$\sqrt{2}$	1
0	0	0
-1	-$\sqrt{2}$	-1

T1

1	0	-1
$\sqrt{2}$	0	-$\sqrt{2}$
1	0	-1

T2

0	-1	$\sqrt{2}$
1	0	-1
-$\sqrt{2}$	1	0

T3

$\sqrt{2}$	-1	0
-1	0	1
0	1	-$\sqrt{2}$

T4

Figure 4.1: The Frei-Chen basis vectors.

determine whether an image sub-area contains an edge element, the following thresholding equation is applied:

$$r = \frac{\sum_{i=1}^{4}(W \cdot T_i)^2}{W \cdot W} \tag{4.2}$$

The image sub-areas contains an edge element if angle Θ is smaller than the calculated threshold value.

The advantage of thresholding based on Θ instead of the length of the projection of W on the edge subspace is shown in Figure 4.2. Two sub-areas W_1 and W_2 are shown with their projection on the edge subspace. If only the magnitudes are considered then W_1 with a large projection is classified as an area containing an edge element. While W_2 has a small projection and would not be classified as containing an edge element. When the threshold however is based on Θ, W_1 with a large angle is not and W_2 with a small angle is classified as an area containing an edge element.

The edge direction α, indicating the direction of the largest difference in intensity, can be calculated using:

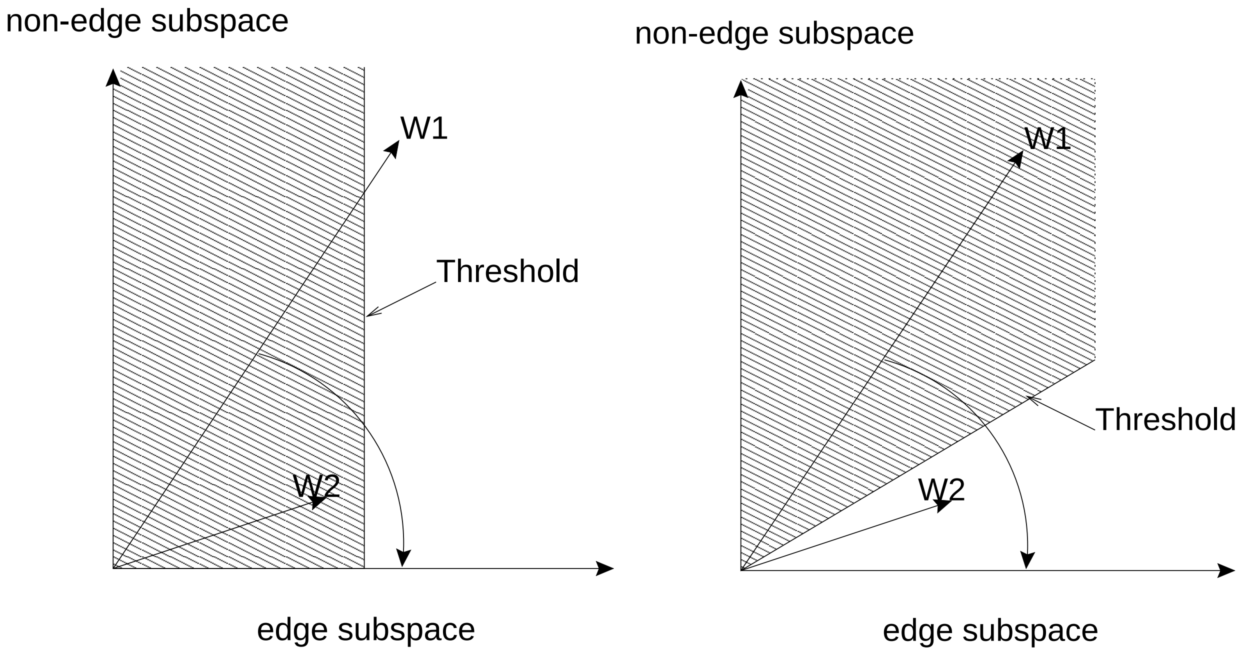
$$\alpha = arctan(\frac{W \cdot T_1}{W \cdot T_2}) \tag{4.3}$$

If $W \cdot T_1$ or $W \cdot T_2$ is equal to zero then algorithm 4.1 is used instead. The direction values α in $[0, 2\pi\rangle$ of $\mathbb{R}$, indicating the direction in degrees, are often converted into $[1, 8]$ of $\mathbb{N}$, indicating one of the adjacent pixels of the pixels located in the center of the sub-area.

Lacroix [17] states that the templates defined by Frei-Chen are not orthonormal to each other. Therefore he proposes normalized and orthogonal templates as shown in Figure 4.1.

Schoenmakers created an improved calculation strategy resulting in a better edge detection. The current (relative) calculation might detect edges in areas with low values and low difference with their adjacent regions, but edges with high values having the same difference might not be detected. To avoid this occurring the window values are set to their absolute values by subtracting the minimum value of the window.

Now one has many pixels classified as possible edge pixels but these do not form one pixel wide contours. The method for edge thinning used is the one developed by Lacroix. She calculates the LBE values for all pixels using 3 by 1 windows placed on the pixels such that the window used coincides with the pixel and its edge direction. The LBE finally is equal to $\frac{m}{v}$ where m how many times a pixel is a local maximum, and v how often a pixel



(a) Conventional threshold. (b) Frei-Chen threshold.

Figure 4.2: Thresholding for edge detection.

IF $(W \cdot T_1 = 0)$ AND $(W \cdot T_2 \geq 0)$ RETURN $\alpha = 0$

IF $(W \cdot T_1 = 0)$ AND $(W \cdot T_2 < 0)$ RETURN $\alpha = \pi$

IF $(W \cdot T_2 = 0)$ AND $(W \cdot T_1 > 0)$ RETURN $\alpha = \frac{\pi}{2}$

IF $(W \cdot T_2 = 0)$ AND $(W \cdot T_1 < 0)$ RETURN $\alpha = 1.5\pi$

IF $(\alpha > 0)$ AND $(W \cdot T_1 > 0)$ RETURN α

IF $(\alpha > 0)$ AND $(W \cdot T_1 < 0)$ RETURN $\alpha + \pi$

IF $(W \cdot T_1 < 0)$ RETURN $\alpha = \alpha + 2\pi$

RETURN $\alpha = \alpha + 2\pi$

Algorithm 4.1: Edge direction computation.

a	b	a
0	0	0
-a	-b	-a

T1

a	0	-a
b	0	-b
a	0	-a

T2

0	-a	b
a	0	-a
-b	a	0

T3

b	-a	0
-a	0	a
0	a	-b

T4

Where: $a = \frac{1}{2\sqrt{2}}$, $b = \frac{1}{2}$

Figure 4.3: Normalized templates.

is visited. The found LBE's are used in a contour following algorithm starting from a pixel with LBE=1 creating one pixel wide edge structures.

4.2 Edge linking

After conclusion of the edge detecting process each pixel is labeled as an edge pixel or as a 'normal' pixel. The edge pixels that are 8-connected with each other form edge fragments. These edge fragments do not necessarily form closed fragments. To achieve that the edge fragments form closed polygons a edge linking program is used that first fills in the small gaps between edge fragments The edge fragments not belonging to closed polygons after edge linking are then removed. (For a detailed description see [4].)

The edge linking process itself is based on graph theory. First the image is scanned and a graph consisting of the edge pixels is constructed. For all the leaves in the graph is checked if it is a candidate for connection. For connection of the leaves a few cases are considered:

- connection of a leaf with another leaf.
- connection of a leaf with a branch.

In both cases it must apply that the distance between the leaf and the other leaf or branch is 2, $\sqrt{5}$ or $\sqrt{8}$. This is only the case if a one pixel wide gap exists. If the leaf is candidate for connection the pixel that will be changed into an edge pixel will be determined taking into account if the pixel to change will form new polygons who all are larger than a user defined number of pixels. During the search process of the new added edge pixel the first found possibility that satisfies the demands is chosen.

After edge linking the closed polygons are detected using a floodfill procedure. Then the edges not belonging to any closed polygon are removed. This is done by recreating the trees and creating a new graph. The leaves in this graph can be determined, and these leaves can be used as starting points for the removal of the loose edge fragments. This is done by first removing the leaves, checking if its predecessor can be removed and if so do it. This continues for all leaves until no predecessor is found that can be removed.

4.3 Edges related to mixed pixels

The edges found in an image coincide with region boundaries, where many mixed pixels are located. Therefore the found edges can be useful for allocating mixed pixels. Ordinary edge detection does not only find edges on boundaries of segments but also edges in the segments themselves. The latter edge segments can be found due to variation within the fields and results in incorrectly selecting many pixels as edge pixels and thus as candidate mixed pixels. Also small parts of the real edges can be missed resulting in different regions being merged, and what is more important in this case fails to allocate a number of possible mixed pixels.

The edge linking process first fills in the small gaps in the found edge structures. Thus different segments will not be merged and more mixed pixels can be allocated. After this, edge fragments not belonging to any closed polygon will be removed as edges. Now the resulting edges do not only form closed polygons dividing the image, but many pixels that were incorrectly seen as candidate mixed pixels are now disregarded. A problem may arise due to the ad-hoc character of the linking process connecting the first found connection being possible if more options are present. This could result in the incorrect assignment of a pixel as edge pixel. But as a whole one can say that linked edges probably have higher agreement with the locations of mixed pixels than ordinary edge pixels.

Chapter 5

Determining the quality of detecting mixed pixels

To determine a quantitative value for the quality of detecting mixed pixels in an image, the mixed pixels found in the resulting segmentation (the allotted segmentation) are compared with those in the true segmentation (the reference segmentation). Determining the accuracy is done in a site-specific method by a point-by-point comparison. The different types of results possible are:

- the true pixel is a mixed pixel and the allotted pixel is a mixed pixel
- the true pixel is a pure pixel and the allotted pixel is a pure pixel

which are correct test results. The incorrect test results are:

- the true pixel is a mixed pixel and the allotted pixel is a pure pixel
- the true pixel is a pure pixel and the allotted pixel is a mixed pixel

To perform the comparison one has to determine where the mixed pixels are located in both the found and reference segmentation. After this a method for determining the accuracy of the found segmentation, resulting in a quality value, is applied.

5.1 Locating mixed pixels

To determine the quality of the allocation of mixed pixels one has to compare the location of the mixed pixels in the true segmentation with those in the allotted segmentation. The true segmentation is saved in the REGIN3 file format containing a description of the mixed pixels it contains. Any kind of information, such as location and decomposition, about the mixed pixels is acquired easily. However the allotted segmentations, resulting after best merge, are saved in the REGINF file format which does not contain any kind of information about mixed pixels. This would bring about that the accuracy of allocation of mixed pixels can not be calculated. However, one can assume that mixed pixels exist in specific areas of an image. The pixels that have a high probability of being mixed pixels are the isolated pixels, pixels in small regions and the edge pixels.

The isolated pixels may contain small structures in the image. For these structures, such as houses or trees in grass areas, it is unlikely that their size is equal to that of a pixel. It is very plausible that they have a size smaller than one pixel. Thus the majority of the isolated pixels consist of a very small structure and parts of its adjacent areas and therefore are mixed pixels.

The larger regions in the image are likely to consist only of pure pixels while smaller regions will mainly consist of mixed pixels. Including regions under a certain size into the category of mixed pixels will be appropriate up to a certain value under which the majority of the pixels contained in these regions indeed are mixed. Though above a certain size most of the pixels will be pure. Therefore it is important to choose a suitable maximum region size.

Edge pixels are pixels that indicate boundaries between different areas. It would be ideal if these pixels would be fully contained in one of the adjacent segments, in other words if the boundaries between two regions would be exactly situated on the edges of pixels. This would imply that the edge pixels consist only of one ground cover class and would be pure pixels. It is unlikely that these pixels are completely situated in one of the adjacent areas. It is very likely that the edge pixels cover part of more than one of the adjacent areas and therefore are mixed pixels.

5.2 Accuracy of a segmentation

Determining the quality of the detection of the mixed pixels is done by comparing the positions of the candidate mixed pixels in the allotted segmentation with those of the mixed pixels in the true segmentation. This is done in a point-by-point approach during which the agreement between two sets of pixels at specific locations is determined.

Selected sample points are used to determine the accuracy of the segmentation. These sample points may cover part of the full image or include the whole image. Using the entire image as sampling points results in an exact reproduction of the agreement between the reference and allotted segmentation. Comparing part of the image approaches the agreement, it however does not give an exact outline of the correspondence between both segmentations. Therefore all the pixels in the segmentation are considered to be sampling points during calculation of accuracy.

5.2.1 Error matrix

The error matrix is used to show the site-specific errors (see Figure 5.1). It exists of an $n \times n$ matrix with n being the number of ground cover classes. The columns show the classes of the true segmentation and the rows those of the allotted segmentation. The matrix value on location (x, y) shows how many of the pixels belonging to class x in the true segment are classified as class y in the classified segmentation. The values in the total row and column show how many elements of that class exist in the true and allotted segmentation respectively.

Reference Data

Evaluated Image		Class 1	Class 2	Class 3	Class 4	Totals
	Class 1	150	7	17	30	204
	Class 2	3	83	8	3	97
	Class 3	23	4	350	13	390
	Class 4	39	3	11	115	168
	Totals	215	97	386	161	698

Figure 5.1: Error matrix

Two kinds of errors can be distinguished for each ground cover class. The first is the error of omission where pixels belonging to a specific class are wrongly assigned to a different class. The error of omission of a class is calculated by dividing the number of omitted pixels by the number of pixels originally being of that class. The other type of error is the error of commission where pixels of classes are wrongly assigned to a different class. The error of commission of a class is calculated by dividing the number of committed pixels by the number of pixels assigned to that class. For example, in Figure 5.1 there are 386 pixels of Class 3. Of those 350 were correctly assigned as Class 3. The error of omission of Class 3 is the sum of the omitted pixels by the number of pixels originally being of Class 3 is $\frac{17+8+11}{386}$. The error of commission of Class 3 is the sum of the committed pixels by the number of pixels assigned to Class 3 is $\frac{23+4+13}{390}$. The overall error of omission (commission) if calculated by dividing the sum of omitted (committed) pixels of all classes by the total number of pixels. The overall error of omission is the same as the overall error of commission because the pixels omitted from a class will be committed to another class.

In the case of determining the accuracy of allocating the mixed pixels only two classes have to be considered namely the class of mixed pixels and that of pure pixels. A two dimensional version of the error matrix (see Figure 5.2) is applied for cases like these.

Reference Data

Evaluated Image		Mixed	Pure	Totals
	Mixed	M	C	M+C
	Pure	O	P	O+P
	Totals	M+O	C+P	M+P

Figure 5.2: Error matrix with two ground cover classes

5.3 Accuracy measures

Two types of accuracy measures are widely used. These are the percentage correct and the kappa coefficient [5, 6].

5.3.1 The percentage correct

The percentage correct returns the proportion of correctly assigned pixels of all the pixels used to build the error matrix. The percentage correct for each individual class is calculated by dividing the number of pixels of that class assigned correctly by the number of pixels of that class present in the true image. The overall percentage correct is calculated by all the correct assigned pixels by the total number of pixels. For Figure 5.1 the overall percentage correct is $\frac{698}{859} = 81.2573\%$.

5.3.2 The kappa coefficient

The kappa coefficient is a measure of difference of the observed agreement and the agreement that would be the result of a random matching of the two segmentations. The kappa coefficient is estimated by:

$$\hat{\kappa} = \frac{Observed - Expected}{1 - Expected}$$

Observed is the value of the overall percentage correct described before. This is the sum of the diagonal values divided by the total number of sample points. Expected is the correct classifications that can be anticipated by chance agreement of the two data sets. Expected is calculated by using the column and row totals (the marginals). Products of the column and row totals provide an estimate of pixels assigned to each cell of the error matrix. Expected is calculated by adding the diagonals of the marginal product matrix and then dividing the result by the total of products of column and rows. The later equals the square of the total number of sample points or in this case the square of the number of pixels in the image. Figure 5.3 shows an example with Figure 5.1 used as error matrix.

$\hat{\kappa}$ adjusts the percentage correct by subtracting the change agreement. Thus $\hat{\kappa}$ is 0.727 would imply that the classification results in an accuracy that is 72.7% better than those of a random assignment of pixels to categories. $\hat{\kappa}$ approaches 1 if the classification is almost perfect. When $\hat{\kappa}$ is 0 the classification is as good as those of random assignment of pixels and is below zero if it has a worse classification.

5.4 Global overview

The whole system constructed to determine the quality of allocating mixed pixels by segmentation programs can be subdivided in three phases. In the first phase an artificial image is created together with its related correct segmentation. In the second phase this

$Observed = 698/859$

Marginal product matrix

43860	19788	78744	32844
20855	9409	37442	15617
83850	37830	150540	62790
36120	16296	64848	27048

$SumOfDiagonals = 230857$

$TotalOfProducts = 859 * 859 = 737881$

$Expected = 230857/737881$

$\hat{\kappa} = (\frac{698}{859} - \frac{230857}{737881})/(1 - \frac{230857}{737881}) = 0.727$

Figure 5.3: Example calculating kappa coefficient

artificial image is segmented using the segmentation algorithm to be judged. This results in a REGINF file showing how the image is subdivided according the algorithm.

The third and final phase determines the quality of the segmentation being the result of the segmentation algorithm. Therefore first the pixels are determined which are mixed pixels in the allotted segmentation. By comparing the locations of the mixed pixels in the allotted segmentation with those in the reference segmentation an error matrix showing the site-specific errors is constructed. This error matrix is used by an accuracy measurement to calculate the quality value.

Chapter 6

Implementation

The complete system can be divided into three steps as shown in Figure 6.1. First the artificial image is constructed resulting in an artificial satellite image and its true segmentation. The resulting artificial image is applied to a segmentation program resulting in a allotted segmentation. The final step, being the point of interest, is checking the quality of allocating the mixed pixels by comparing the true with the allotted segmentation.

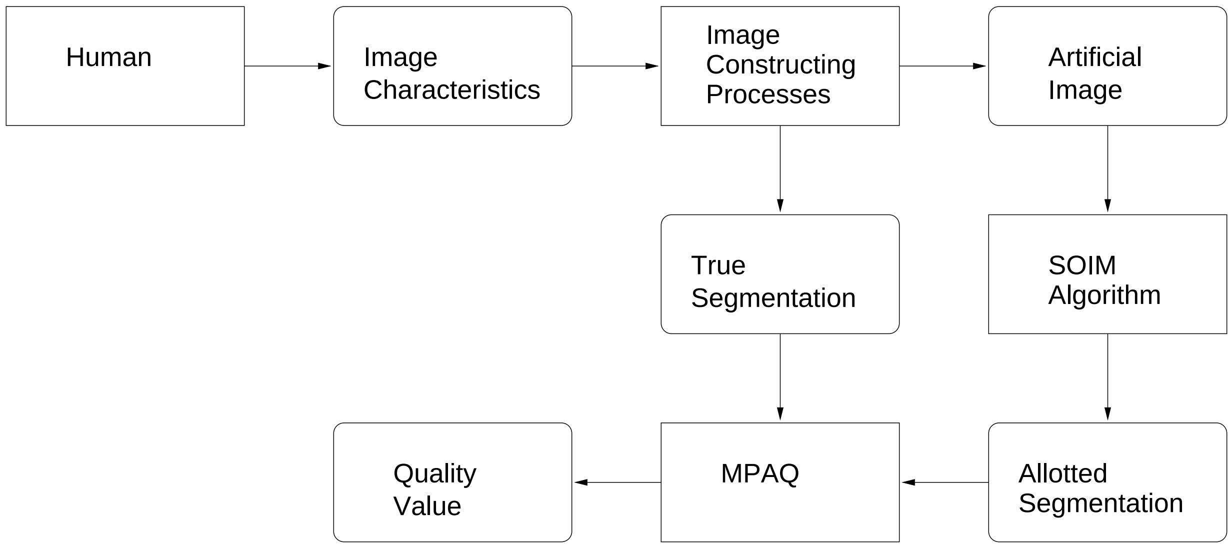


Figure 6.1: Complete system

A part of the used programs was implemented before starting this thesis. Therefore a short description of the existing parts that are of concern is given first. This is followed by a description of the implementation and a section describing the usage of MPAQ.

6.1 Existing programs

In this thesis several already existing programs where used to fulfill a few tasks. The first task is the construction of an artificial image. The second is the execution of the best merge segmentation program. MPAQ may use information related to the location of the edge pixels in the artificial image. For determining where these edges are situated the program doedge is used.

6.1.1 Constructing an artificial image

Constructing an artificial image is done in several stages (see Chapter 2). First an image is drawn by hand and scanned to produce an image in PBM format. The program pgmtolan is then used to convert the resulting image into an edge file. The program its command structure is:

pgmtolan [s|v] [t threshold] pgm-file lan-file

Where:

s|v: Sun or Vax output format
t threshold : threshold to edge file
default: s, no thresholding

After creating an edge file by using pgmtolan, the pixels on the perimeter of the image are removed by partlan and saved as a ERDAS file. After starting partlan the following is displayed on the screen:

partlan selects part of a lan file
1: change inputfile name
2: change outputfile name
3: 0 start line
4: 0 start element
5: 0 number of lines
6: 0 number of elements
10: doit
11: quit

(Instead of starting by only typing typing 'partlan' one can type 'partlan inputfile outputfile' after which both the file names are displayed in the information shown above.) The user now is able to change some of the shown information before he decides to create a new ERDAS file containing information about the selected part of the input file.

Contour is used to perform edge linking in the resulting edge image afile.lan to obtain closed polygons. This is done by filling in small gaps between found edges and removing edge fragments not belonging to closed polygons. The result is returned in an ERDAS file afile.lan_out. The command structure of contour is:

contour minpolysize input-edge-file

Where:

minpolysize: minimum size of polygons, smaller ones are removed

To create an region file with only one channel from the edge file, previously created by doedge, mdisp is used in the following way:

mdisp infile outfile

Where:

infile: an ERDAS file with extension lbe (1 channel, values of 0 and 255)
outfile: a REGINF file with 1 channel with the polygon pixels

Laninf is then used to determine the polygon number for each polygon present in the region file returned by mdisp.

laninf [s|v] [a|f|t] [m xxx] reginf-file output-file

Where:

s|v : Sun or Vax output format
a|f|t : Average of regions or Flag coloring or a Text output
m xxx: take xxx extra bytes of memory for buffers,
if xxx $\leq$ 1024 then it is MBytes
default: v, a, 2 Mbyte

Finally the REGINF file being the result of laninf is used by lansim for constructing an ERDAS file, being the artificial image, and the REGINF file being the artificial image's true segmentation. Using texture images for the ground cover classes available in the image and a man made map file, indicating which regions in the image are of which ground cover type, the segments are colored.

lansim scale frac mapfile prefix regionfile file1 file2

Where:

scale: integer size reduction in output file
frac: fraction isolated pixels ($0 \leq \text{frac} < 0.25$)
map-file: region number to class
prefix: prefix for class images
regionfile: input regionfile
file1: output erdasfile
file2: output regionfile

6.1.2 The best merge program

The best merge algorithm is a segmentation method that performs region growing.

bmerge inputfile threshold ppp [pinf_file] kr cf [maxcells]

Where:

inputfile : input file name
threshold : the threshold (float)
ppp : 'y' for performance per polygon,
'n' for whole image
pinf_file : file with the polygon info
kr : 'y' to perform knowledge rules on isolated pixels 'n' otherwise
cf : the cost-function to use
's' for Schoenmakers
't' for semi-Tilton
'b' for Beaulieu & Goldberg
maxcells : make 1 region of polygons larger than maxcells
default value 1,000,000,000

The cost function created by R Schoenmakers is used throughout this thesis, in this case the segmentations are always performed with the cf set on s.

6.1.3 Allocating edge pixels

In MPAQ the option exists to consider edges or linked edges, present in the image, as mixed pixels. The edge pixels in the image are found by using doedge and returns an ERDAS file with the extension lbe.

doedge [m xxx] input.lan

Where:

m xxx: take xxx extra bytes of memory for buffers
if xxx $\leq$ 1024 then it is MBytes
input.lan: input erdasfile

If the linked edges are being considered as mixed pixels the edge fragments found by doedge are connected by using contour as done by the construction of an image.

6.2 Implementation of MPAQ

The new application MPAQ is built up of three parts. The main module mpaq interpreting the command prompt and reading basic information about the true and allotted segmentation, the module mixpix determining which pixels in the allotted segmentation probably are mixed pixels, and finally the module qfuncts calculating the accuracy value (see Figure 6.2).

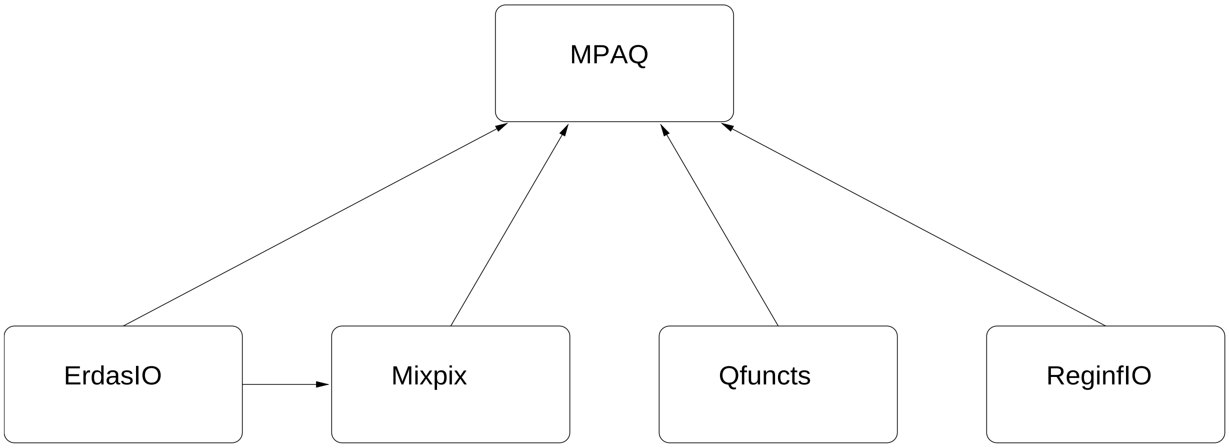


Figure 6.2: Module structure.

6.2.1 Existing modules

For the ERDAS and REGINF file types (see Appendix A) there already existed modules able to read or write information from or to files. For this thesis it was only necessary to read information from the files. Therefore only the procedures for reading information are discussed.

The ERDAS format:

In the module erdasio, written by Theo Schouten, the functions are implemented for handling files of the ERDAS72 and ERDAS74 format. The functions that are relevant are the only ones discussed here. Beside these functions three important static integers are also declared in this module. These integers are:

- VaxFormat: Indicates if the file is constructed on a Sun or a Vax.
- erdas_version: Indicates the version of the erdas file.
- ipack: Indicates if the ERDAS file uses 1 or 2 bytes per pixel.

void read_erdas_header(FILE *fpa, int *elements, int *lines, int *channels):

This function checks if the file pointed to by fpa is a correct ERDAS file. If the file is a correct ERDAS file the static integers VaxFormat, erdas_version and ipack are set before returning how many columns (elements), rows (lines) and channels the ERDAS file contains.

void get_erdas_info(int *fileformat, int *fileversion, int *filepack):

This function returns information about the version of the erdas file (fileversion), if it is constructed on a Sun or Vax (fileformat), and if there are one or two bytes used for saving one pixel (filepack). The information about these three items is extracted from the three declared static integers that were appointed values by read_erdas_header().

void read_erdas_line(Byte *scanline, FILE *fpa, int elts, int nr_chan):

This function reads the data of a line segment, consisting of elts number of pixels and nr_chan channels, of the picture pointed to by fpa, and stores it in scanline.

The REGINF format:

The routines for handling files of the REGINF or REGIN3 format are located in the module reginfio. This is a variation of the module regionio written by M. Klein Gebbinck.

struct inf_header read_reg_header(FILE *file, char *filename):

Here the header of the region file pointed to by file (for its structure see Appendix A.2) is read. After checking if the file is a correct REGINF or REGIN3 file the whole read file header is returned.

void read_header_values(struct inf_header *infheader, int *version, Ushort *format, Ushort *rows, Ushort *cols, Ushort *channels, Ushort *accuracy, int *numregions, Ushort *globregs):

This function dissects the header inf_header of the region file into its basic elements (for description of the header elements see Appendix A.2). All these elements are returned if the file header is a correct file header of a REGINF or REGIN3 file.

void read_reg_mixpixs(FILE *file, int byteorder, ushort *pix[]):

The function determines which pixels in the artificial image are mixed or pure pixels. The input in this module consists of two items. The first item is a REGIN3 file, being the true segmentation belonging to the artificial image, pointed to by file. The second item is the byteorder indicating whether the file is constructed on a Sun or a Vax.

The output it returns is the matrix pix indicating which pixels in the artificial image are mixed and which pixels are pure. The matrix itself is of the same dimensions as the artificial image. If the value of coordinate (x, y) in the matrix is 0 then the pixel on position (x, y) in the artificial image is a pure pixel, if the value is 1 it is a mixed pixel.

6.2.2 The main module

The main module mpaq first determines which type of quality value has to be calculated by evaluating the command prompt. This is done by the function getargs() which determines all the parameters in the given command. In case of using the wrong options the usable options will be shown by usage() and the process will be aborted. Currently two quality values can be calculated. These are the kappa coefficient and the percentage correct calculated by the functions kappa() and percentage() respectively.

void kappa(char *file1, char *file2):

The function first reads the header of the true segmentation pointed to by file1, by using read_reg_header(), after which the elements this file header contains are read by read_header_values(). Then memory is allocated for the matrix mixpix1, having the same dimensions as file1, which contains information about which pixels in file1 are pure or mixed after applying read_reg_mixpixs().

Thereafter, the file header of the allotted segmentation, pointed to by file2, and the header elements are read by read_reg_header() and read_header_values(). Then memory is allocated for matrix mixpix2, having the same dimensions as file2. After applying read_bmerge_mixpix() this matrix contains information about which pixels in file2 are mixed and which are pure.

If (linked) edges are also considered to be mixed pixels, the header and its elements of the edge file are read by using read_erdas_header() and get_erdas_info(). Then memory is allocated for a matrix mixpix3, having the same dimensions as the edge file, which contains information about which pixels in the edge file are edges and thus mixed pixels after performing read_edge_mixpix(). Thereafter mixpix3 is added to mixpix2 by add_poss_mixpix() resulting in a new mixpix2.

Finally the kappa coefficient for mixpix1 and mixpix2 is calculated by calculate_kappa().

void percentage(char *file1, char *file2):

This function is equal to that of kappa() except that in the final step the percentage correct is calculated by calculate_percentage() instead of the kappa coefficient being calculated by calculate_kappa().

6.2.3 Determine mixed pixels

Determining which pixels in the best-merged segmentation are mixed pixels is done in the module mixpix. In order to be able to determine which pixels are mixed it must be known how large the maximum segment size of the segments completely consisting of mixed pixels is. Also knowledge about the fact whether (linked) edges should be included must be available.

The main functions in this module are:

- read_bmerge_mixpix()
- read_edge_mixpix()
- add_poss_mixpix()

void read_bmerge_mixpix(FILE *file, int format, ushort *pix[], int size):

The function determines which pixels in the allotted segmentation file are mixed or pure pixels. The input in this function consists of two items. The first is the REGINF file, pointed to by file, belonging to the allotted segmentation. The second is the maximum region size indicating that all regions with a smaller size consist of mixed pixels.

The output it returns is a matrix pix indicating which pixels in the resulting segmentation are mixed and which are pure. The matrix itself is of the same dimensions as the used image. If the value of coordinate (x, y) in the matrix is 0 then the pixel on location (x, y) in the allotted segmentation is a pure pixel, if the value is 1 it is a mixed pixel.

void read_edge_mixpix(FILE *file, ushort *edgpix[], int rows, int cols, int channels, int format, int ipack):

If the (linked) edges in the artificial image have to be considered as being mixed pixels then (doedge followed by contour) doedge is used to return an edge file (see Paragraph 6.1.3). Function read_edge_mixpix() determines which pixels in this edge file are edge pixels and thus are mixed pixels. The input consists of the edge file, being of the ERDAS74 file type, and the information about this file concerning dimensions and number of channels.

The output is a matrix edgpix indicating which pixels in the artificial image are considered to be pure or to be mixed pixels. The matrix itself is of the same dimensions as the used edge file. If the value of coordinate (x, y) in the matrix is 0 then the pixel on location (x, y) in the allotted segmentation is a pure pixel, if the value is 1 it is a mixed pixel.

void add_poss_mixpix(ushort *pix[], ushort *edgpix[], int cols, int rows):

The function adds the mixed pixels found after considering the (linked) edges as mixed pixels to the mixed pixels found in the allotted segmentation. The input consists of the matrixes being the result of read_bmerge_mixpixs() and read_edge_mixpix().

The output is a matrix edgpix showing which pixels are mixed pixels in the matrix related with the edge file or with the allotted segmentation. If the value of coordinate (x, y) in the matrix is 0 then the pixel on location (x, y) in the allotted segmentation is considered to be a pure pixel, if it is 1 it is considered to be mixed pixel.

6.2.4 Accuracy measures

Calculating the accuracy is done in the module qfuncts. Two methods of calculating the accuracy have been realized till now. These are the percentage correct which is calculated by calculate_percentage() and the kappa coefficient calculated by calculate_kappa().

The main functions in this module are:

- calculate_kappa()
- calculate_percentage()

- error_matrix()
- kappa_coefficient()
- percentage_correct()

double calculate_kappa(ushort *mixpix[], ushort *pix[], int rows, int cols, int absolute):

This function calculates the kappa value. The input to this function consists of the matrixes mixpix and pix showing which pixels in the reference segmentation and allotted segmentation are pure or mixed pixels. The input further consists of information about the dimensions of the image and whether the quality value must be calculated by using absolute or relative values.

First memory is allocated for a matrix errormatrix. After performing error_matrix() this matrix will contain the error matrix with its row and column totals. After calculating the percentage correct, by using the resulting matrix in the function percentage_correct(), the kappa coefficient finally is calculated by kappa_coefficient().

The output calculate_kappa() returns is the calculated value of the kappa coefficient.

double calculate_percentage(ushort *mixpix[], ushort *pix[], int rows, int cols, int absolute):

The function calculates the percentage correct. The input to this function consists of the matrixes showing which pixels in the reference segmentation and allotted segmentation are pure or mixed pixels, the dimensions of the artificial image and if the quality value must be calculated by using absolute or relative values.

First memory is allocated for a matrix errormatrix. After performing error_matrix() this matrix will contain the error matrix with its row and column totals. Finally the percentage correct is calculated by using the function percentage_correct().

The output calculate_percentage() returns is the value of the percentage correct.

void error_matrix(ushort *mixpix[], ushort *pix[], double *errormatrix[], int rows, int cols, int absolute):

The function calculates the error matrix related to the artificial segmentation and the allotted segmentation.

The input consists of a value showing showing if the quality value must be calculated by using absolute or relative values and two matrixes mixpix and pix. In this case mixpix indicates which pixels in the artificial image really are mixed or pure pixels, and pix which pixels are seen as being mixed or pure pixels after segmentation (eventually including found edge pixels as being mixed pixels). If the value of coordinate (x, y) in the matrix is 0 then the pixel on location (x, y) is considered to be pure, if it is 1 the pixel is considered to be mixed.

The error matrix is calculated by comparing the type of pixel of all the pixels in mixpix with the pixels located on the same location in pix.

The output is a matrix errormatrix, consisting of 3 rows and 3 columns, containing the values of the error matrix together with its column and row totals.

void kappa_coefficient(double *errormatrix[], double pcorrect, double *kappa):

The function kappa_coefficient() calculates the value of the kappa coefficient. The input is the error matrix returned by error_matrix() and the percentage correct calculated by percentage_correct(). The output it returns is the value of the kappa coefficient.

void percentage_correct(double *errormatrix[], double *pcorrect):

The function percentage_correct() calculates the value of percentage correct. The input is the error matrix returned by error_matrix() and the output is the value of percentage correct.

6.3 How to use MPAQ

After implementation and compilation the program MPAQ is created that is able to determine the quality of detecting mixed pixels in a segmentation program. The input files of MPAQ are the artificial segmentation in the REGIN3 file format and the segmentation made by the SOIM algorithm in the REGINF file format. The output shows the accuracy in a predefined way. At the present time the accuracy can be shown in percentage correct or as kappa coefficient. The command structure of MPAQ is as follows:

mpaq quality-function [s x] [edge-info] arti-file bmerged-file:

Where:

quality-functions:
- k: kappa function. (using relative values)
- ka: kappa function. (using absolute values)
- p: percentage-correct. (using relative values)
- pa: percentage-correct. (using absolute values)

s x: The maximum size of regions fully consisting of mixed pixels
- Default: s 1.

edge-info:
- n: no usage of edge information
- e edge_file: use edge information in the edge file edge_file
- Default: n.

arti-file: REGIN3 file of artificial image.

bmerged-file: REGINF file of bmerged artificial image.

Chapter 7

Results

In this chapter the method for determining a quantitative value for the quality of detecting mixed pixels during a segmentation process of images is put to the test by using an artificial image. An artificial image is used instead of a real image to obtain a true segmentation that can be compared with the found segmentation. The SOIM segmentation method is applied for making segmentations at several threshold values ranging from 0 up to 40. Finally the quality of the resulting segmentations is determined using the MPAQ algorithm.

7.1 Test cases

Each test consists of three steps. The first step is the creation of an artificial image together with its true segmentation. Then the segmentation method is applied on the artificial image resulting in a test segmentation. Finally the test segmentation is compared with the true segmentation by labeling each individual pixel as pure or mixed and comparing the pixels in both the segmentations.

The different test cases are influenced by the different characteristics the artificial image may contain, by the different parameters used by the SOIM algorithm and by the developed quality program. In the artificial image the most important characteristics to be considered are:

- Assignment of a ground cover type to the edges
- Presence of isolated pixels
- The resolution of the artificial image

In the SOIM algorithm several parameters can be set. The threshold value, which is used in all the test cases, and the indication if the knowledge rules will be applied, are the most significant parameters that have to be set. The developed program also influences the outcome of the quality value. The parameters of importance that have to be set here are:

- The maximum size of regions seen as mixed pixels

- Absolute versus relative number of errors
- Using percentage correct or kappa coefficient

You can calculate the quality value by using the absolute or the relative number of errors in the allocation of pure and mixed pixels. If you use the absolute number of pixels however, the pixel type with the higher number of occurrence has greater influence in the outcome than the less occurring pixel type. Therefore all the test cases are shown with their relative values.

Until now you can use two methods of calculating the overall accuracy of the final segmentation. These are the percentage correct and the kappa coefficient. However, the percentage correct is not influenced by the errors not placed on the diagonal of the error matrix. In this chapter therefore the kappa coefficient is used because it takes into account the whole matrix.

At each experiment three cases are considered. The first with the constructed image, thereafter the artificial image with its edge pixels seen as mixed pixels and finally the image with its linked edges seen as mixed pixels.

7.2 Initial Experiment

The initial artificial image is created from an image drawn by Theo Schouten (see Figure 2.1(a)). It is converted into an edge image by using an threshold value of 200. Edge linking is performed with the minimum size of the polygons set at 10. The mapping process assigns the edge pixels to the ground cover class soil. Finally the image is reduced in size by using resolution 4, and 1 percent of the pixels in the image is replaced by isolated mixed pixels. The accuracy is calculated using the kappa coefficient with the region size set on the default value of 1. Figure 7.1 shows channel 5 of the resulting image.

In Figure 7.2 the calculated values for the kappa coefficient are shown on the y-axis and the SOIM thresholds on the x-axis. The value of $\hat{\kappa}$ first increases to an optimum and then decreases to a constant value. This is the result of the merging process. With a small SOIM threshold few segments are merged together and thus a segmentation that is very fragmented is formed. With increasing threshold values more and more segments are merged together incorrectly. This continues until the point is reached that incrementing the threshold no longer influences the outcome.

Without edges $\hat{\kappa}$ is higher than with considering (linked) edges as mixed pixels. This is due to the relatively high number of isolated pixels in the image. These pixels are dissimilar enough in relation with their surroundings to be seen as edge pixels. And because of their high number of occurrence extra edge structures are formed. Therefore far too many pixels are seen as mixed pixels. With higher SOIM threshold values the opposite occurs and without edges $\hat{\kappa}$ is smaller than with considering edges as mixed pixels. Due to the high threshold too many adjacent segments are merged together just as many isolated pixels are also merged with their surroundings. By considering (linked) edges as mixed pixels more of the mixed pixels present in the true segmentation are found.

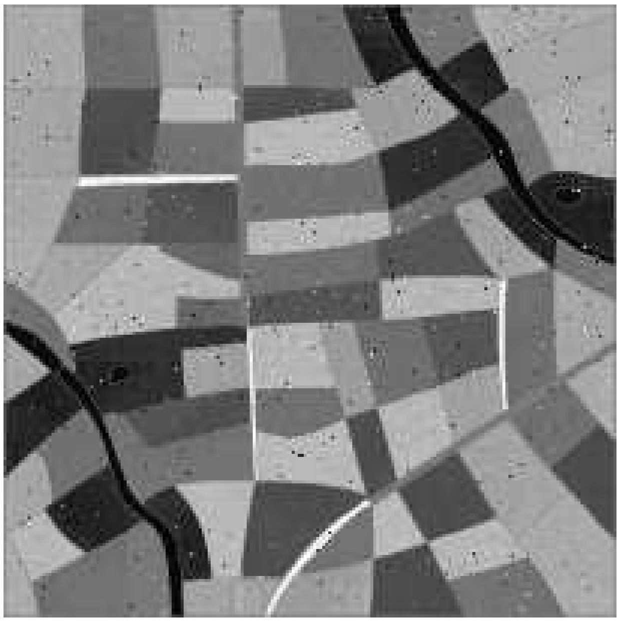


Figure 7.1: Initial artificial image

The value of $\hat{\kappa}$ is higher considering linked edges as mixed pixels than when you consider edges as mixed pixels. This is result of the addition of edge pixels to form closed polygons and the removal of the edge fragments that are not part of closed polygons. The deleted edge fragments mostly are pixels seen as edges due to variations in the regions. While the pixels added to form closed polygons mainly are pixels that really are edge pixels but were disregarded.

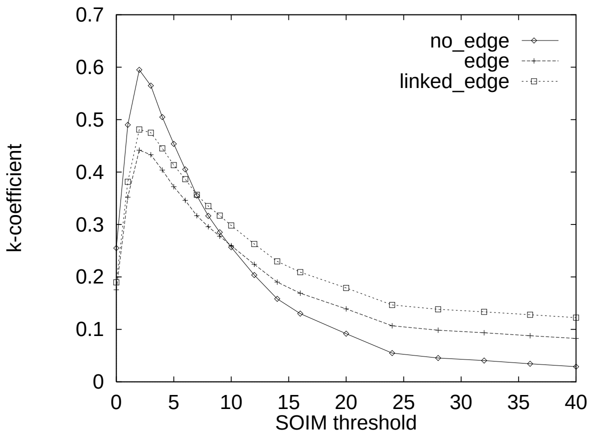


Figure 7.2: κ-coefficient of the initial image.

7.3 Absence of isolated pixels

Not adding isolated pixels during the process of creating an artificial image is a different case (see Figure 2.2). Due to the absence of isolated pixels the number of mixed pixels present in the image has decreased. Now $\hat{\kappa}$ is higher with than without considering the (linked) edges as mixed pixels. This is due to the fact that many of the pixels near the edges of segments are mixed pixels, and more of these mixed pixels are found when the found (linked) edge pixels are labeled as mixed pixels.

From Figure 7.3 it appears that the $\hat{\kappa}$ values are smaller with addition of isolated pixels than without addition of isolated pixels if (linked) edges are considered. This is the result of the large structures formed by the added isolated mixed pixels who are incorrectly seen as edge pixels. This results in many pixels wrongly being seen as mixed pixels in the case of addition of isolated pixels.

The $\hat{\kappa}$ values are larger with isolated pixels if edges are not considered. This is result of finding most of the added isolated pixels during the segmentation process.

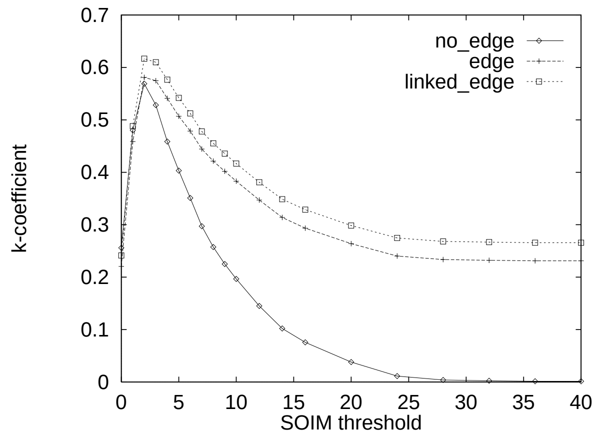
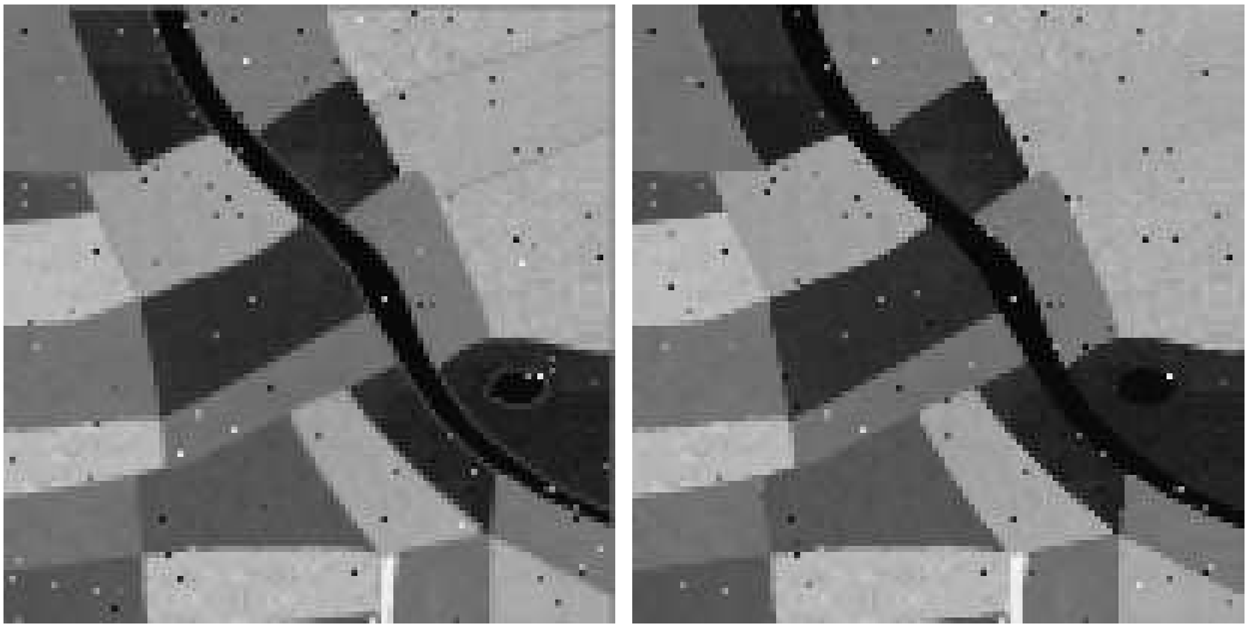


Figure 7.3: κ-coefficient image without isolated pixels.

7.4 Appointing soil class to edges

The next artificial image used is very similar to the initial image. The only difference is that no ground cover type has been assigned to the edges (see Figure 7.4). To see the difference in results of (not) assigning a ground cover class to edges the kappa values of Figure 7.2 and Figure 7.5 are compared.

When no ground cover type is assigned to the edges, less of the artificial pixels become mixed pixels and more become pure. The edge fragments being part of the artificial pixels do not influence the pixel values because their values were ignored during construction of the image. This is opposite to the case in which a ground cover type is assigned and the intensity of the edge pixels does influence the values of the artificial pixels and more edge pixels in the artificial image become mixed pixels. In the artificial pixels the percentage of



(a) Top right of image with assigned edges. (b) Top right of image without assigned edges.

Figure 7.4: Assignment of ground cover type to edges.

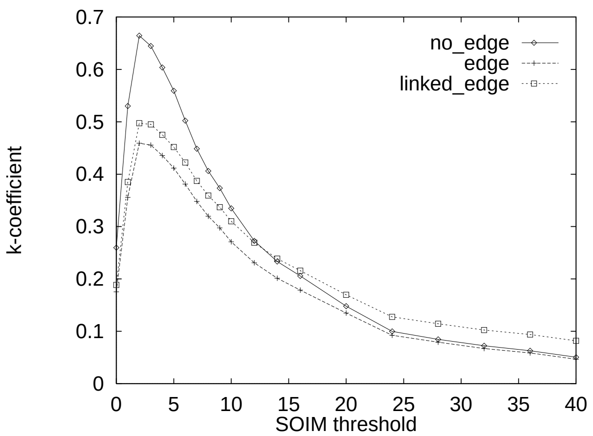


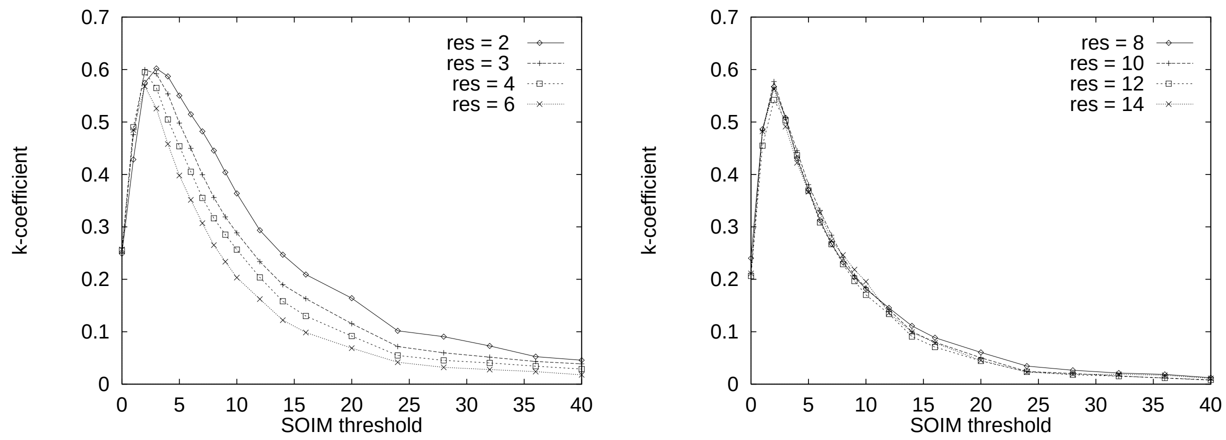
Figure 7.5: κ-coefficient image without edge assignment.

sub-pixels being edge pixels is however generally low. This results in pixels whose intensity does not differ a lot from those without any ground cover class being assigned to the edges.

The $\hat{\kappa}$ values are smaller with assigning of a ground cover class than without. This is the result of the number of artificial pixels being mixed pixels. With the assignment of a ground cover class to the edges all the pure pixels containing edge fragments, thus only consisting of one other ground cover type next to the edge fragments, become mixed. This results in an increment of the total number of mixed pixels. A new mixed pixel however hardly differs from a pure pixel consisting of the same ground cover type as its remaining sub-pixels. These mixed pixels therefore are hard to detect which leads to incorrectly considering many new mixed pixels as being pure.

7.5 Resolution of the image

During the process of constructing an artificial image one decides in which resolution the image has to be created in relation to the image drawn. By creating several artificial images with characteristics similar to the initial image (see Paragraph 7.2) differing only in resolution, several different cases can be considered. Because the kappa values of images with the present characteristics have the best result while not using information about the (linked) edges only information about this case is viewed (see Figures 7.6(a) and 7.6(b)).



(a) κ-coefficient image resolution 2-6.

(b) κ-coefficient image resolution 8-14.

Figure 7.6: κ-coefficient with several resolutions.

Due to the higher resolution[1] the artificial pixels cover a larger amount of sub-pixels. An artificial image with a higher resolution consists of a decreasing amount of pixels, however the percentage of pixel being mixed increases. The mixed pixels will mostly be built up of

[1]Normally a higher resolution means that an image shows more detail. Here a higher resolution means that one artificial pixel is build up out of more sub-pixels.

a specific ground cover type with small parts of other types and thus are less mixed. The intensity of these mixed pixels therefore will not differ much compared with the intensity of pure pixels completely built up of the sub-pixels being the majority of the mixed pixel. With increasing resolution the mixed pixels thus become less mixed and have a higher similarity to pure pixels resulting in the mixed pixels being more difficult to detect. As a result of this the kappa values are worse with increasing resolution.

It however is not possible to draw more conclusions about the different resolutions because of the method of constructing the artificial images. In the final step of constructing the artificial image the image first is reduced in size and then is then filled in using the constructed templates. In the obtained artificial images with high resolutions, the pixels therefore do not simply have the average intensity of pixels, located exactly in the area covered by the first mentioned pixels, contained in an image with a smaller resolution.

7.6 Usage of knowledge rules

During the best merge process one can use knowledge rules for merging the isolated pixels present in the image (see Paragraph 3.2.4). Due to the knowledge rules a lot of the found isolated pixels, which very likely are mixed pixels, are merged with their surrounding segments. Many of the pixels that are mixed are now seen as pure pixels. The $\hat{\kappa}$ values therefore are smaller with usage of knowledge rules than without knowledge rules (see Figure 7.7).

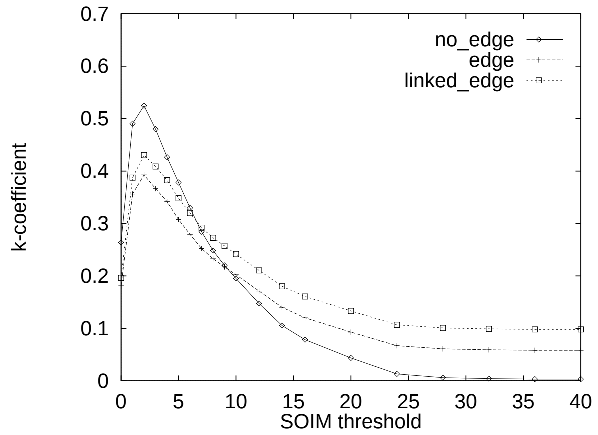


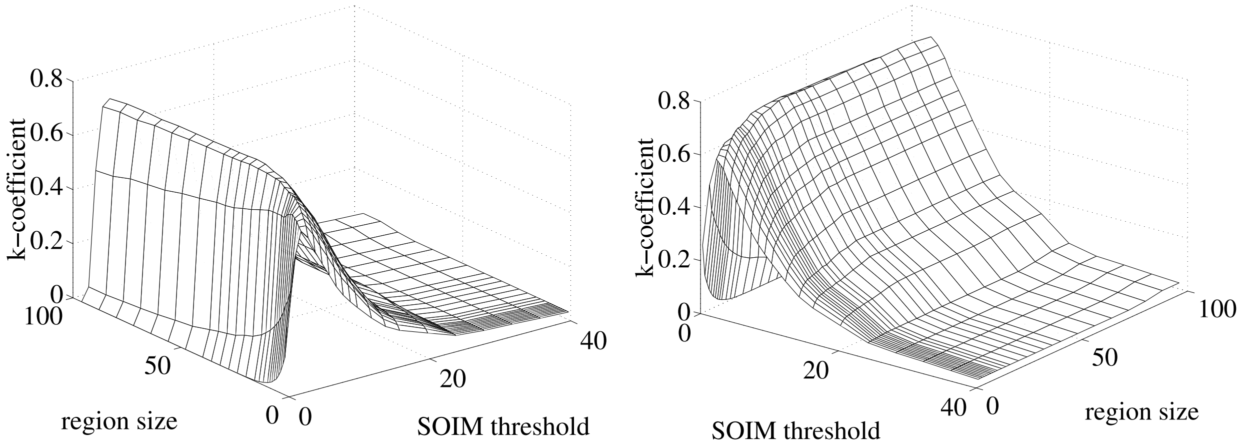
Figure 7.7: κ-coefficient with usage of knowledge rules.

7.7 Size of regions completely consisting of mixed pixels

You can assign the maximum size of segments whose pixels are all considered to be mixed pixels. In order to determine the effect of the maximum region size on the quality value, the initial image is applied using several increasing region sizes. The initial image contains a high percentage of isolated pixels. In this case the optimum $\hat{\kappa}$ is smaller with than without considering the (linked) edges to be mixed pixels (see Paragraph 7.2). Therefore the $\hat{\kappa}$ values are only shown for the latter case using several region sizes.

With increasing region size the $\hat{\kappa}$ values also increase. This happens because of the fact that the majority of the pixels contained in regions with larger sizes are mixed pixels. This is valid up to a certain region size above which the regions consist of a larger amount of pure rather than of mixed pixels, after which the $\hat{\kappa}$ values will decrease (see Figure 7.8).

The SOIM threshold value is still of great influence on the resulting kappa coefficient with variable region size. Therefore it also is shown how the maximum region size and threshold values relate to each other. Figure 7.8 shows that the threshold value has more influence on the outcome of the $\hat{\kappa}$ values than the region size has. This is the result of the SOIM threshold influencing the sizes of the regions present in the segmentation. With a low threshold the image is fragmented and many regions are incorrectly seen as consisting of mixed pixels when a small region size is used. This results in many pure pixels incorrectly being seen as mixed pixels. With a large threshold the image is divided in a few large segments which are only seen as mixed pixels when using a very large region size. With a low region size the mixed pixels contained in these large regions are seen as pure pixels thus incorrectly labeling many mixed pixels as pure. With a high region size many pure pixels embedded in the large segments are incorrectly seen as mixed pixels.



(a) κ-coefficient (view 1). (b) κ-coefficient (view 2).

Figure 7.8: Kappa values

For each region size the maximum value of the κ-coefficient and the optimal threshold resulting in that maximum value is shown in Figure 7.9. The best $\hat{\kappa}$ values are reached when the region size is 28. In Figure 7.10 the maximum κ-coefficient is shown for the different threshold values together with the optimal region size resulting in that maximum $\hat{\kappa}$ value. Here the best result is reached using a threshold of 4. By combining the Figures

7.9 and 7.10 you can again see that the setting of the SOIM threshold, which almost always returns the best results with a threshold value 3 or 4, is of more importance than the setting of the region size.

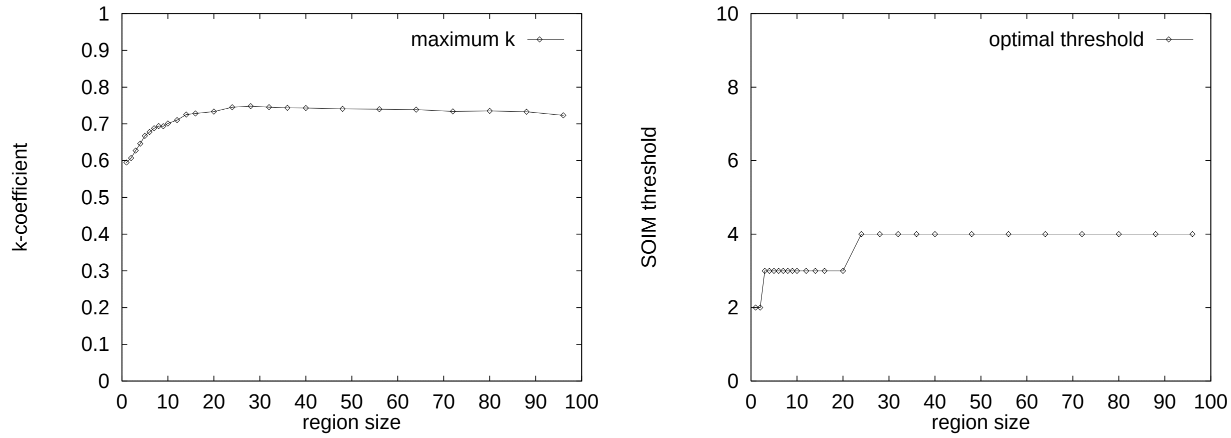


Figure 7.9: Optimal κ values per region size

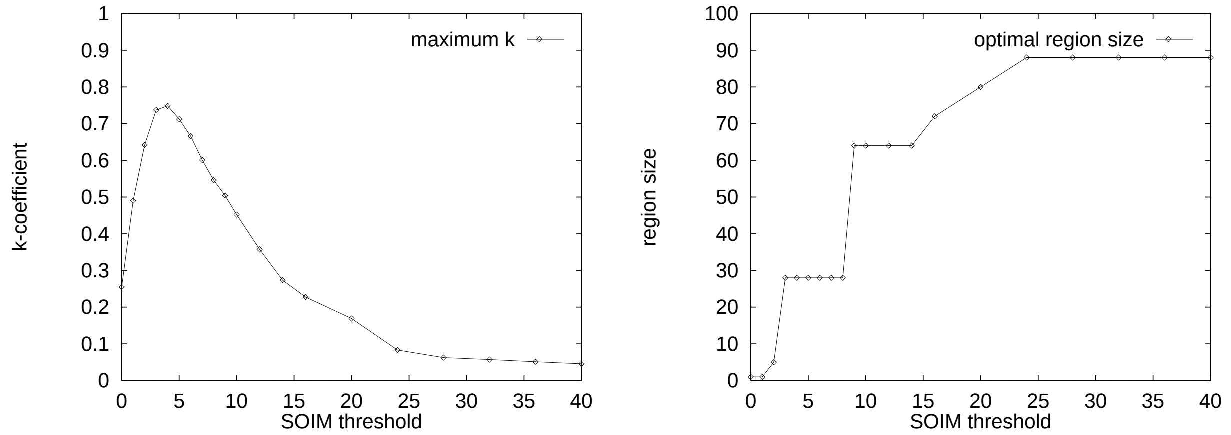


Figure 7.10: Optimal κ values per SOIM threshold

7.8 Summary and conclusions

In this chapter several tests have been performed in which the quality of detecting mixed pixels, during a segmentation process that is applied on an artificial image, is calculated. These tests cover the possible differences that may be set in artificial images and the options that can be set in the applied segmentation and quality program.

Reviewing the tests that where executed one sees that per individual test case the setting of the SOIM threshold is the factor that has the highest influence on the outcome of the result. Also in the majority of the test cases, when the SOIM threshold is chosen in such a way that the kappa coefficient approaches its maximum, not using information about (linked) edges leads to the best results. The only exception is when the image contains few isolated pixels in which case using information about linked edges results in the best quality values.

Chapter 8

Conclusions

In this thesis a algorithm called MPAQ has been developed that determines the quality of detecting mixed pixels. This is done by using an artificial image that was created together with its true segmentation. The reason for using an artificial image instead of a real image is that information about the ground truth data has to be available in order to determine a quantitative value of a made segmentation. This image is first segmented using a segmentation program resulting in a segmentation that has no kind of information about the locations or characteristics of the mixed pixels which are present in the image. However it assigns a ground cover type to each pixel thus considering all pixels as being pure. Therefore the algorithm first has to determine which pixels in the segmentation made by the segmentation program are mixed or pure. During this step it is also possible to regard found (linked) edges as consisting of mixed pixels. Thereafter, by using an accuracy measure, how well the detection of mixed pixels has occurred is determined by comparing the segmentation with the true segmentation resulting in a qualitative value.

8.1 Quality measurement results

In Chapter 7 several settings of MPAQ were put to the test using various different artificial images. The initial artificial image contains a high percentage of isolated pixels. Regarding found (linked) edges as being mixed pixels will lead to a worse result than without regarding edges as being mixed. This is due to the fact that many edge pixels are incorrectly found due to within field variation or high appearance of isolated mixed pixels. When the image however hardly contains isolated pixels and the fields in the image hardly contain within field variation, the majority of the mixed pixels are located near the edges of segments. Considering linked edges thus will lead to a better result.

In the artificial image a ground cover type can be assigned to the sub-pixels which are edges. This can be compared to real images in which most of the fields are divided by separations that are very small in relation to the pixels in the image. In this case many edge pixels will become mixed pixels whose intensities however hardly will differ from the intensities of the pixels in the case of not assigning a ground cover type. Most of these new mixed pixels will therefore not be found.

One may use images of different resolutions. Having a higher resolution results in the mixed pixels being less mixed and therefore harder to find. Therefore the results of images with higher resolutions will lead to worse results.

The SOIM threshold and maximum region size both influence the quality of the accuracy of detecting mixed pixels. When the threshold value increases the maximum region size also has to increase in order to achieve the maximum accuracy. The quality of detecting mixed pixels however, is influenced more by the setting of the SOIM threshold than that of the maximum region size.

8.2 Possible future work.

The used segmentation program only segments images down to the level of pure pixels. Therefore MPAQ first has to determine which pixels in the resulting segmentation are mixed pixels. Then only information is known about which pixels in the image are being regarded as mixed pixels but nothing is known about the composition of the mixed pixels.

A new segmentation program can be built that not only detects mixed pixels, for example by integrating the part of MPAQ that determines the possible mixed pixels into the best merge algorithm, but also determines the fraction of the ground cover types that the individual mixed pixels consist of (see [9, 11]).

MPAQ is only able to calculate two kind of quality values indicating the quality of localizing the mixed pixels namely the percentage correct and the kappa coefficient. The number of accuracy measures can be extended in new versions of MPAQ.

The artificial images used so far are small in relation to Landsat images of approximately 250Mb. Memory is generally too small to hold images of this size. Therefore a sliding window algorithm can be used that reads a fraction of large files into memory and keeps the whole file on disk.

The quality determining algorithm described in this thesis only focuses on the detection of mixed pixels. During this process nothing is determined about the localization of pure pixels or about the characteristics of the mixed pixels. Another option for future work therefore is the creation of an algorithm that returns a quantitative value indicating how well the whole image is segmented by comparing the complete estimate areas covered by the segments present in the image. Here the composition of the mixed pixels is also taken into account, resulting in a better area estimation.

A different point of interest for future work is the generation of more realistic images. Presently the segments are filled in by using templates that reflect the texture and statistics of real fields (see Paragraph 2.1). The method of calculating the intensity of the pixels present in a segment however results in a repetition of structure when this segment is larger than the templates used.

Appendix A

File Types

A.1 The ERDAS74 file format

The segmentation application is applied using an satellite image. These images are in the ERDAS file format of which there exist two versions, the ERDAS72 and ERDAS74 file format. The only difference between these versions is that in the ERDAS72 version floats are being used instead of integers as used in the ERDAS74. Because of this minor difference only the ERDAS74 file format will be described.
The ERDAS74 image is composed of two parts. The first is the header describing the map statistics. The latter is the body containing the actual image information itself.

The header is 128 bytes long and its structure is shown below:

```
struct
{
   byte hdword[6] /* pos 1-6. */
   ushort ipack   /* pos 7-8. */
   ushort nbands  /* pos 9-10. */
   byte unused[6] /* pos 11-16. */
   float cols     /* pos 17-20. */
   float rows     /* pos 21-24. */
   float rx       /* pos 25-28. */
   float ry       /* pos 29-32. */
   byte dummy[88] /* pos 33-120. */
   float xcell    /* pos 121-124. */
   float ycell    /* pos 125-128. */
}
```

The variables in the header are:

- hdword: Specifies the ERDAS file format. If hdword is HEADER the file is in the ERDAS72 format if hdword is HEAD74 in the ERDAS74 format.

- ipack: Indicates if there are 1 or 2 bytes used per pixel.
- nbands: Specifies the number of channels.
- cols: Specifies the number of columns.
- rows: Specifies the number of rows.

The body structure contains the information of the pixel values. It indicates the pixel values ordered by rows of which the image is built up. And per row per column the information is given per channel. The number of bytes used to represent a pixel is determined by ipack.

```
cols pixel values for line 1, channel 0
cols pixel values for line 1, channel 1
   :                            :
cols pixel values for line 1, channel nbands-1
cols pixel values for line 2, channel 0
   :                            :
cols pixel values for line rows, channel nbands-1
```

A.2 The REGINF file format

A segmented image is divided into its regions. The region information file contains the information about the individual regions. The REGINF file consists of a header and a body.

The headers structure is:

```
struct
{
   char format      /* pos 1-6 */
   ushort byteorder /* pos 7-8 */
   ushort nrows     /* pos 9-10 */
   ushort ncols     /* pos 11-12 */
   ushort nbands    /* pos 13-14 */
   ushort accuracy  /* pos 15-16 */
   int nregs        /* pos 17-20 */
   ushort spare[6]  /* pos 21-32 */
}
```

The variables in the header are:

- format: Specifies the file format. The format is REGINF or REGIN3. In this case format is REGINF.

- byteorder: Indicates if the file is constructed on a Sun or Vax.
- nrows: Specifies the number of rows.
- ncols: Specifies the number of columns.
- nbands: Specifies the number of channels.
- accuracy: Indicates if there 2 or 4 bytes used per mean value.
- nregs: Specifies the number of regions.
- spare[6]: Is for future use.

The body contains the actual information about of the regions. The first part contains information of the mean spectral values per channel multiplied by 100. The used data type for the mean values, ushort or int, depends on the accuracy defined in the header.

- ushort or int (depending on accuracy) per channel mean value * 100

The second part shows the information of the pure pixels. This is done by means of run length encoding in which each adjacent area of pixels on a row belonging to the segment is saved by writing the coordinates of the first pixel and the number of pixels belonging to that area. The information consists of:

- ushort y: start y-position
- ushort x: start x-position
- ushort n: area length

This continues until the end of the region is indicated by and end of region. This is an integer with value (ushort -1)

A.3 The REGIN3 file format

A segmented image is divided into its regions. The region information file contains the information about the regions. The REGIN3 file consists of a header and a body.

The headers structure is:

```
struct
{
   char format      /* pos 1-6 */
   ushort byteorder /* pos 7-8 */
   ushort nrows     /* pos 9-10 */
   ushort ncols     /* pos 11-12 */
   ushort nbands    /* pos 13-14 */
```

```
    ushort accuracy  /* pos 15-16 */
    int nregs        /* pos 17-20 */
    ushort spare[6]  /* pos 21-32 */
}
```

The variables in the header are:

- format: Specifies the file format. The format is REGINF or REGIN3. In this case format is REGIN3.
- byteorder: Indicates if the file is constructed on a Sun or Vax.
- nrows: Specifies the number of rows.
- ncols: Specifies the number of columns.
- nbands: Specifies the number of channels.
- accuracy: Indicates if there 2 or 4 bytes used per mean value.
- nregs: Specifies the number of regions.
- spare[6]: Is for future use.

The headers structure of this version is similar to the REGINF format. The only difference is that the spare[0] field is used to show the number of global regions.

The body contains the actual information about of the regions. The first part contains information of its ground cover class by means of a unsigned short. Then the number of pure pixels in the region are noted in an integer. After this information of the mean spectral values per channel multiplied by 100 is given. The used data type for the mean values, ushort or int, depends on the accuracy defined in the header. Finally the variance-covariance matrix is saved with each value multiplied by 100 and represented in an integer.

The second part shows the information of the pure and mixed pixels. This is done by means of run length encoding. The information consists of:

- ushort y: start y-position
- ushort x: start x-position
- ushort n: area length

This continues until the end of the region is indicated by and end of region. This is an integer with value (ushort -1)

Then the information of the mixed pixels is shown. This is done by giving the location and fraction of this segment in the mixed pixel. This is done for all the mixed pixels present in the segment.

- ushort y: start y-position
- ushort x: start x-position
- ushort f: fraction of this region * 100

This is repeated until the end of the region is indicated by and end of mixed pixels. This is an integer with value (ushort -2)

Appendix B

Utility Programs

During realization of this thesis several existing and developed programs are used. All is done on a UNIX workstation. In this chapter a short description about the used programs is given.

B.1 Written programs.

The programs written on a UNIX workstation in the programming language C before existance of but used during creation of this thesis are:

XV3: XV is an public domain image display program which can display, manipulate and convert from and to a range image formats. It however is not able to view images of the ERDAS file type. For this case the XV3 is developed so one can view and manipulate ERDAS files.

B.2 Unix applications.

The UNIX applications used by creation of this report are:

Gnuplot: Gnuplot is an interactive plotting program. It plots functions or data files. The user can define x and y ranges, label the axes or define functions.

Latex: This thesis is written using the LaTeX text editor.

Matlab: Matlab is an interactive program to help you with numeric computation and data visualization.

Ups: Ups is a graphical source level debugger for the C programming language running under the X11 window enviroment.

XV: XV is an public domain interactive image display program for the X11 window enviroment. It can display images and part of images, manipulate images and their color and convert from and to a range of image formats on all types of known X terminals.

Bibliography

[1] Hubert F.J.M Voogd, Error measurement for segmentation techniques. Masters thesis, Faculty of computer science, Catholic University of Nijmegen, The Netherlands, December 1994. Draft version.

[2] R Schreurs, Correctness and efficiency of the best-merge's region-growing. Masters thesis, Faculty of computer science, Catholic University of Nijmegen, The Netherlands, November 1993.

[3] Freddy Verstaaten, Segmentation of large satellite-images. Masters thesis, Faculty of computer science, Catholic University of Nijmegen, The Netherlands, September 1993.

[4] Ronald P.H.M. Schoenmakers, Integrated methodology for segmentation of large optical satellite images in land application of remote sensing. PhD thesis, Faculty of computer science, Catholic University of Nijmegen, The Netherlands, 1995.

[5] James B. Campbell, Introduction to remote sensing, Taylor & Francis Ltd, 2nd edition, London 1996.

[6] Andrew B. Peterson, A land cover classification of Greater Boca Raton, Florida using SPOT imagery, November 1994.

[7] HM Horwitz, RF Nalepka, PD Hyde, and JP Morgenstern. Estimating the proportion of objects within a single resolution element of a multispectral scanner. In *Proceedings of the 7th international symposium on remote sensing of environment*, pages 1307-1320, 1971. Ann Arbor, Michigan.

[8] Theo Schouten, PRELIMINARY Segmentation of full size Landsat TM and Spot. March 1994.

[9] Theo E Schouten, Maurice Klein Gebbinck. Decomposition of mixed pixels. In J Desachy, editor, *Image and signal processing for remote sensing II*, pages 104-115. SPIE 2579, 1995.

[10] Theo E Schouten, Maurice Klein Gebbinck. Quality measures for image segmentation using generated images. In J Desachy, editor, *Image and signal processing for remote sensing II*, pages 411-422. SPIE 2579, 1995.

[11] MS Klein Gebbinck. Fractional assignation of mixed pixels to regions. February 8, 1996.

[12] Theo E Schouten,MS Klein Gebbinck. A neural network approach to spectral mixture analysis.

[14] J-M. Beaulieu and M. Goldberg. Hierarchy in picture segmentation: A stepwise optimization approach. *IEEE Transactions on pattern analysis and machine intelligence*, 11(2):150-163, 1989.

[15] J.C Tilton. Image segmentation by iterative parallel region growing and splitting. *Proceedings of the international geoscience and remote sensing symposium (IGARSS89)*, pages 2235-2238, Vancouver, 1989.

[16] S.W. Zucker. Region growing: Childhood and adolescence. *Computer graphics and image processing*, 5(3):382-399, 1976.

[17] V. Lacroix. A three-module strategy for edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 10(6):803-810, 1988.

[18] W. Frei & C.C. Chen. Fast Boundary Detection; a Generalization and a New Algorithm. *IEEE Transactions on Computers*, C-26(10):988-998,1977.

Index