# INTEGRATED METHODOLOGY FOR SEGMENTATION OF LARGE OPTICAL SATELLITE IMAGES IN LAND APPLICATIONS OF REMOTE SENSING

Ron Schoenmakers

# Integrated Methodology for Segmentation of Large Optical Satellite Images in Land Applications of Remote Sensing

een wetenschappelijke proeve op het gebied
van de Wiskunde en de Informatica

## Proefschrift

ter verkrijging van de graad van doctor
aan de Katholieke Universiteit Nijmegen
volgens besluit van het college van decanen
in het openbaar te verdedigen op
13 september 1995, des namiddags te 3:30 uur precies

door

Ronald Peter Helena Maria Schoenmakers

geboren op 23 oktober 1961 te Brunssum (NL)

Promotores:

Prof. dr. ir. J. Vytopil

Prof. dr. ir. M. Molenaar
(Landbouw Universiteit Wageningen)

# Preface

The first time I heard from the Joint Research Centre (JRC) of the European Union, was when I fulfilled my military service at the Royal Dutch Navy. "Via via" I got Jacques Stakenborg's name and telephone number. He informed me that the group in which he was working mainly dealt with remote sensing and especially with image processing and classification. Furthermore he informed me that there were possibilities to be employed by the Centre, but the fastest way would be to apply for a Ph.D grant. I completed the application forms he had sent me together with a proposal on image processing and segmentation and one year and ten months(!) later I started as a "borsista" in Ispra. This was the second of October, 1989. It was decided following my proposal and Stakenborg's suggestions that the topic I would work on would be image processing with a focus on segmentation of remotely sensed optical imagery. For all the things Stakenborg did for me, I want to say something: Jack, you know how I appreciated the things you did for me and I "wanna" thank you for that. Staak, bedankt jonguuuhhhhhh.

Immediate after receiving the application forms I went to Theo Schouten and Jan Vytopil who had been my supervisors while carrying out my Master's thesis at the "Katholieke Universiteit" at Nijmegen (NL). They were interested in having a Ph.D student in Italy and so Jan Vytopil became my promotor. Theo was the man who would follow me directly. In the beginning communication with Theo was not so easy as we still did not have electronic mail at the JRC and so we had to do it via phone and fax. Of course Jacques Stakenborg was the man who had to supervise me inside the Centre. Therefore, Jan I want to thank you for being my promotor and Theo I want to thank you for the discussions we had and the time you spent supervising me.

As I was not very familiar with remote sensing I had to develop my knowledge of it. I had to read books about how satellites work, different sensor types, how images are composed, calibration and geometric correction of data needed etc. Discussion with Jacques showed that it was best to start with two segmentation approaches, edge detection and region growing. Being familiar with the Frei-Chen edge detection method I decided to work on this algorithm. As Jacques was a very busy man (he even once had attached a sheet of paper on his chair with "Do not disturb" written on it), he proposed me to change supervisor and to speak to Graeme Wilkinson who was also very familiar with these image processing techniques. Via Graeme I got in contact with Ioannis Kanellopoulos who had already worked on the region growing part and who had implemented the best merge. Via them I got the right knowledge about the best merge method with which I still was not very familiar. And thus, Graeme you know that without your help and support this dissertation would not be in its current form. Especially during writing up, had you not been eager to criticize my work: “Ron, I don’t like this chapter, leave it away”, “you are not writing a textbook”, and “this chapter is definitely too long”, etc. But thinking about your remarks and criticisms (which made me sleep badly), I found out that you were almost always right. Thanks for being hard on me. Ioannis I want to thank you for giving me your best merge implementation, supplying me with the data I needed for the ERS-1/SAR filtering, for carrying out the classification of the Lisbon image and for the good time we had in Tokyo.

The good contacts I created with Graeme and the good contact I had already with Theo, resulted in a lot of papers which have been presented at several conferences, - the time I couldn’t present my paper at Houston because it is outside the European Union and so my mission was not paid for, Theo organized it in such a way that the University paid for my mission - and finally resulted in the dissertation as presented here.

However, the contacts with Jacques had not stopped and collaboration continued resulting in the paper on the filtering.

During my grant I met a colleague student, Guido Lemoine, who was working on radar images within the institute of remote sensing applications. During the many scientific discussions we had he created my interest for radar and we did several small experiments on the usability of the segmentation algorithm under development for segmentation of microwave imagery. And therefore, “Lemoen” bedankt. Via him I met people from Wageningen Agricultural University, i.e. Hans van Leeuwen with whom a joint paper has been presented. Hans also thanks. I also met Martien Molenaar who was asked in a later stage to be my other promotor. Martien I want to thank you for the time you have put in me and for the discussions we had.

During my visits at the University I was placed in the same room with Louis Vuurpijl who was (i.e. still is) doing his Ph.D on neural networks. He is responsible for my interest on this subject. His interest in segmentation and my interest in neural networks resulted in a joint paper which has been submitted to the Igarss95 conference. When I needed some help on LaTeX, Unix and C, Louis was always available and often he resolved the problem by writing a “script-je” (awk, C-shell). “Loe” also without you this thesis would not be as is. You know that. Ook bedankt.

Theo arranged it so that he found students willing to do work on related topics with my research, i.e. Charles Hendriks and Rene Schreurs who implemented the filter, as developed by Jacques and me, on the universities transputer network. Freddy Verstraaten and Rene Schreurs did their Master’s on the edge linking subject and the best merge respectively. Discussion of their progress was done when I was at university, via e-mail or via ‘talk’. I want to thank them for their collaboration and their motivation in developing software. Jongens bedankt.

Thanks to Jean Meyer-Roux and Paul Vossen who supplied me with a desk and allowed me free use of the computer equipment. I know that my colleagues noticed when I was performing some segmentation (and/or debugging). The VAX often “stood-still” and had to be rebooted when swapper took over. My colleagues patience in these situations attributed to the already pleasant atmosphere at the MARS project. They all were a great stimulation in ending up this dissertation. I want to thank them all. Special thanks to my roommates Giampiero Genovese with whom I also did a joint paper and who brought me in contact with Telespazio, Konstantinos Lambiris, “Brother, che cosa dobbiamo fare?”, and Ian “Mr Cyberspace” Barnes, the first human being I segmented. Also the fiver-side soccer games we played, our group against other groups within the institute, where amusing. Especially in summer when we played at noon in the heat sometimes you had the feeling you were dying. “Lekker bikkelen”.

It was very interesting to work in an international environment and I learned a lot from it.

Thanks to Jim Tilton from GSFC, NASA for applying and adapting IPRG, so it used the same merging function as I did, to the test image I used. This made a comparison possible between his implementation and mine.

I also want to thank Edmond Nezry for filtering the ERS-1 images as presented in this dissertation.

Being back in Nijmegen I often stayed with Jack van Poll and company. Therefore I want to thank: Bart (Hanbect), Cristine (Miep), Jules (Lucien), Louis (Loe), Renate, Silvano (Bronko), Steef, and Jack (The zenman, de knast). Dank U boys.

In the beginning it was not easy to survive in a “strange country”. I didn’t speak the language which led to funny situations in the shops when I needed something. Sometimes I had to give up when I tried to explain what I needed. To integrate into Italian society I decided to start with fitness. I went to the first gym “palestra” (“Energy”, Ispra) I found and subscribed. Michele the owner is very fanatic and increased my motivation. During the 4 year period I trained there I met a lot of locals and had very pleasant moments though “pompare” was not always that easy. Michele grazie per avermi accettato alla tua palestra e per avermi fatto pompare e soffrire.

My interest for the mountains made me decide to become a member of the Italian Alpine Club (“Club Alpino Italiano (CAI)”), section of Sesto Calende. Via the free climbing course I did with people from the club brought me furthermore in contact with a lot of Italians. The “rifugio” of the CAI at Alpe Devero has been for the last years the place where Dutch friends and I passed the Christmas days. I love the place, the times we arrived and the road was blocked, we couldn’t get there, defrosting the water tubes, the toilet, heating up the place etc., really I felt to be on another world. Per questo voglio ringraziare la gente del CAI che è stata sempre gentile e che mi ha dato sempre, senza fare problemi, il rifugio ad Alpe Devero. Ragazzi del CAI grazie per tutto.

Voglio ringraziare la gente del Celebration bar (New Sele), buona la pizza, e Il Glicine per i momenti divertenti. Grazie Stefano e company per le cene, le risate, le ferie, le arrampicate......(“arrambocare”)

After all I have to admit that the “Lago Maggiore” is a very nice region with a nice climate and within two hours drive you can go skiing (even in summer); Louis knows all about it (haha). So Italy has become my second home. Grazie Italia.

Having contact with a lot of Italians caused that my Italian became “better and better”. I even can say “hello” in Italian now.

Grazie alla famiglia Roncolato che mi ha offerto la sua ospitalità durante un periodo difficile. Grazie a tutti.

I want to thank all the people who are not explicitly mentioned here but who contributed in some way in the completion of this thesis. Thank you all.

And last but noy least, I have to thank my parents who suffered when I was away. Sometimes they did not see me for months. Dank u wel vader en moeder.

Furthermore I want to thank SHELL for paying my trip to Tokyo. And last but not least of course I want to thank the European Union who supplied me with the grant.

This dissertation has been written using LaTeX on a UNIX workstation. Figures have been created in LaTeX and XFIG. Graphics have been created using GNUPLOT and images in ERDAS format have been converted into postscript using XV.

Ispra, May 3, 1995 Ron Schoenmakers

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# List of Images

# Summary

Segmentation is a process which groups pixels of an image of any type which conform to user defined characteristics by using the grey values of these pixels and their adjacency. In this dissertation, a segmentation method for optical remotely sensed imagery is presented which integrates newly developed and improved existing methods, i.e. edge detection, edge linking and region growing, into a hybrid one with the aim of augmenting classification accuracy in land use applications and especially in agriculture. Existing segmentation methods, those adapted and those developed are discussed and compared.

For two segmentation methods, edge detection and region growing, their advantages and disadvantages are described. For region growing it is shown that order dependence and region over-growing are the main problems whereas for the edge detection it is shown that the full boundaries of fields are not always detected. It has been indicated in the literature that a combination of both methods can increase segmentation accuracy. Therefore a hybrid method has been developed which can be briefly described as follows: edges are detected using a gradient based edge detector. The separate edge fragments are then linked by addition of lacking pixels. In this way closed polygons are created. Edge fragments not belonging to a closed polygon are removed. Within a closed polygon region growing is carried out. In a post-segmentation stage (detected) edges are added to regions found by the region growing. In a pre-segmentation stage, noise is removed using a new filter based on the morphology of the ordered grey levels in an image window around every pixel.

The "Lacroix" edge detection method used, had two drawbacks for which solutions needed to be found. The first was that it was developed for single band imagery only. The second was its decreasing sensitivity for high intensity values of the pixels.

Because of the fact that edge detection does not always find closed boundaries, an edge linking method was developed which adds edge pixels where single edge pixel gaps exist between edge fragments in order to create closed boundaries. Remaining disconnected edge fragments are removed. The edge linking process uses graph data structures to represent edge pixels and their relationship.

The best merge (Tilton and Beaulieu & Goldberg) region growing algorithm overcomes the order dependence problem by merging those pairs of regions at every iteration which obey some merging criterion minimizing the merging cost. But in this dissertation it is shown that when several pairs have the same lowest merging cost, also this algorithm is order dependent. A method has been developed to solve this. However, it is shown that this merging procedure can have an undesirable side effect in that image ramps can be inappropriately merged. However for remotely sensed imagery this does not appear to be a serious disadvantage. Furthermore a cost function has been developed which is independent of the size of the regions.

The algorithm proposed is capable of segmenting full scene imagery which can occupy between 38 Mb (SPOT-P) and about 250 Mb (Landsat-TM). To handle these amounts of data, the method developed uses a slicing technique in such a way that swapping from disk to main memory (RAM) and vice versa is done by the algorithm independently of the operating system.

The use of the edge detection and edge linking to divide the image into independent closed polygons enables region growing to be broken into seperate processes based on each polygon separately. This leads to an improvement in execution times as the time needed for region growing is proportional to the square of the number of pixels at the start. This makes an implementation on a parallel computer or a number of computers connected in a network straightforward. This is of importance as region growing takes the most time of the whole segmentation chain.

Results of the hybrid segmentation method are presented for a Landsat-TM and SPOT panchromatic scenes. The use of the segmentation method is illustrated in three remote sensing contexts. 1) The combination of segmentation of Landsat-TM and a multi-layer perceptron neural classifier for land cover inventories. 2) The use of segmentation of a Landsat-TM image for speckle filtering of a geo-referenced ERS-1/SAR image. 3) The use of segmentation of NOAA/AVHRR NDVI imagery for the extraction of vegetation profiles. n of vegetation profiles. 4) An evaluation has been given for the segmentation of combined optical and radar imagery. Ground truth information is used to evaluate to segmentation. It has been shown that the within field variance decreased after segmentation.

# Chapter 1

## Introduction: Remote Sensing and Image Processing

### 1.1 General introduction

Currently remote sensing (RS) is gaining more and more importance. Data are collected within RS using different types of sensors and are presented in the form of multi-spectral digital images. The elements of these images are called pixels[1]. Current applications of RS using these data transform the data in order to extract the enclosed information.

Segmentation groups pixels of an image of any type which conform to user defined characteristics by using the spectral values of these pixels and their adjacency. In this dissertation, a segmentation method for optical remotely sensed imagery is presented which improves and integrates existing methods into a hybrid one. The methods being integrated are edge detection and region growing. The former looks for discontinuity in spectral value between neighbouring pixels whereas the latter looks for uniformity in spectral value between such pixels. Both methods are seen as each other's complement. The strengths of each method are used to compensate for weaknesses in the other. The hybrid method developed also encompasses a filtering stage, for which a new filter type has been developed. The integration of the edge detection and region growing resulted in a method which is suited for a parallel implementation.

*Chapter 1* introduces the main problem addressed in this dissertation and discusses the problems appearing in segmenting optical remotely sensed imagery. In *chapter 2*, existing methods in segmentation are reviewed including methods for edge detection and for region growing. In *chapter 3*, edge detection is discussed. In *chapter 4*, an algorithm is presented which links edge fragments resulting from the initial edge detection process into meaningful closed polygons. Edge fragments not belonging to a closed polygon are deleted. The resulting closed polygons divide the image into areas. In *chapter 5*, the region growing method used is discussed based upon the so called 'best merge' approach. *Chapter 6* describes how the separate modules are integrated. Furthermore it describes how full scene imagery is segmented and the possibilities of implementing the new segmentation method for parallel computation are discussed. In *chapter 7* overall results of the method implemented are given. In *chapter 8* overall conclusions

[1]pixel is an acronym for picture element.

are drawn. In *appendix A* the spectral noise suppressing filter is described. In *appendix B* the automatic method for thresholding the edge magnitudes is described. In *appendix C* an introduction into artificial neural networks is given. Especially the multi-layer perceptron (MLP) with backpropagation learning, is highlighted.

## 1.2 Introduction to remote sensing

Lillesand & Kiefer [63] describe RS as the science and art of obtaining information about an object, area or phenomenon through the analysis of data acquired by a device that is not in contact with that object, area or phenomenon. (For a further introduction to the subject of RS, see e.g. Mather [67], Elachi [27], Belward & Valenzuela [8], Swain & Davis [104], Asrar [3], Schowengerdt [99]).

In this dissertation the focus will be the observation of areas, especially agricultural fields. The required pre-processed data are gathered from satellite or aerial images.

The concept of *electro-magnetic radiation* describes the way in which energy is transferred from one object to another through space, with the 'speed of light' (300.000 km/sec). Figure 1.1 shows the electro-magnetic spectrum.

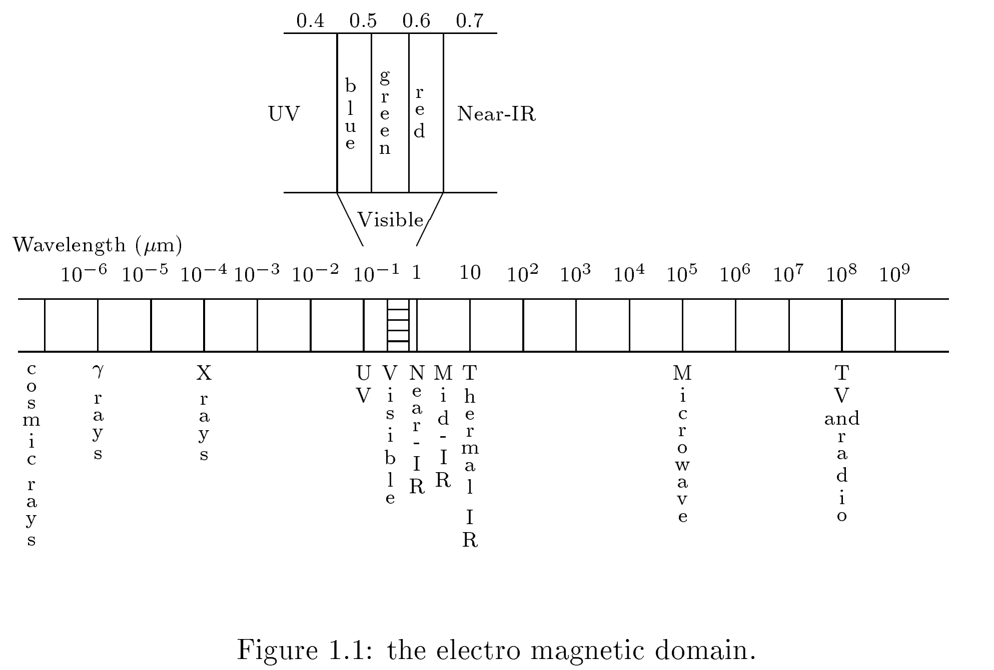



Figure 1.1: the electro magnetic domain.

Two main RS techniques can be categorized. *Active RS* describes the situation where the spaceborne or airborne system provides the source of illumination. *Passive RS* describes the situation where the natural source of object illumination is independent of the sensor.

The atmosphere influences profoundly the spectral composition and the intensity of radiation available to optical (i.e. passive) remotely sensing systems. These effects are caused principally through the mechanisms of *atmospheric scattering* and *absorption*. The former indicates interaction of the electro-magnetic radiation with particles within the atmosphere with different sizes. The latter indicates the loss of energy to atmospheric constituents (water vapor, $CO_2$ and ozone).

When electro-magnetic energy is incident ($E_I(\lambda)$, where $\lambda$ is the wavelength) on any given Earth surface feature, three fundamental energy interactions are possible.

1. $E_R(\lambda)$: reflected energy by the surface.
2. $E_A(\lambda)$: absorbed energy absorbed by the surface.
3. $E_T(\lambda)$: retransmitted energy to underlying Earth-layers.

By applying the principle of conservation of energy it follows that

$$E_I(\lambda) = E_R(\lambda) + E_A(\lambda) + E_T(\lambda) \; . \tag{1.1}$$

Two points concerning this relationship should be noted:

1. The proportion of energy reflected, absorbed and transmitted will vary for different Earth features, depending on their material type and condition.
2. the wavelength dependency means that the proportion of reflected, absorbed and transmitted energy will vary at different wavelengths.

The reflectance characteristics of surfaces may be quantified by measuring the portion of incident energy that is reflected and is called *spectral reflectance*. A graph of the spectral reflectance of an object as a function of the wavelength is termed a *spectral reflectance curve*. Figure 1.2 shows one for three different classes.

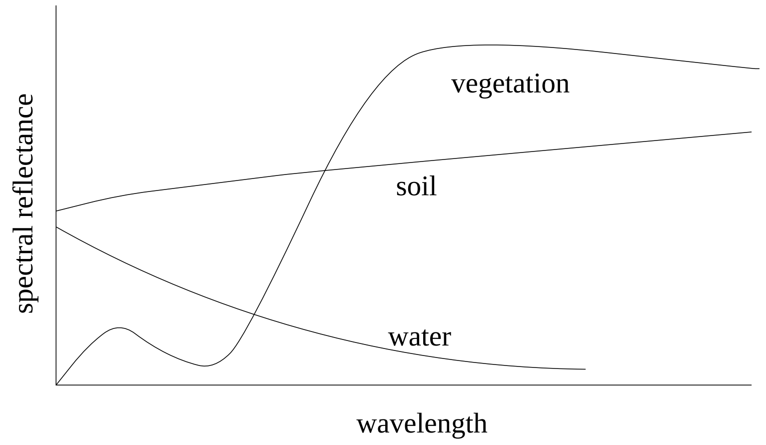



Figure 1.2: generalized spectral reflectance curves for 3 classes (after Schowengerdt [99]).

Imaging systems can be divided into three general categories: framing cameras[2], *scanning systems* and *pushbroom systems*.

Scanning systems use a scanning mirror which projects the surface onto the electronic detectors. The side to side scanning of the mirror provides an image

[2]not discussed here.

swath across the ground track of the satellite and the platform motion provides the along track imaging.

Pushbroom scanners need no scanning mechanism, but they use a linear array of detectors to cover all pixels in the across-track dimension.

The angle subtended by the geometrical projection of a single detector element at the Earth's surface defines the *instantaneous field of view* (IFOV). A scan line consists of contiguous IFOVs which constitute together the *field of view* (FOV). The ground area[3] represented by a pixel is determined by the altitude of the sensor system and its design parameters, particularly the IFOV. A pixel is completely characterized by three quantities:

- ▹ the linear dimension of the sensor's IFOV projected to the ground,
- ▹ the distance between consecutive IFOV samples,
- ▹ the number of bits representing the sample.

An overview of different platforms and sensor types used is given in table 1.1. Data from the following platforms are used mainly for mapping and monitoring applications: Landsat-TM[4], SPOT[5], ERS-1[6]. Raw data supplied by the sensors is subject to noise and distortions both in the spatial and in the radiometric domain. Therefore it needs to undergo geometric and radiometric correction. To correct errors introduced by the sensors these data have to be calibrated.

### 1.2.1 About imagery

Every sensor registers the analogue radiance signal and transforms it into a digital output signal. This is done for every pixel and therefore every pixel has as many spectral bands as there are sensors. A digital image $I$ consists of $N$ lines and $M$ columns. The image $I$ can be formally given as a matrix of pixels:

$$I = \begin{bmatrix} \vec{p}_{1,1} & \cdots & \vec{p}_{1,M} \\ \vdots & & \vdots \\ \vec{p}_{N,1} & \cdots & \vec{p}_{N,M} \end{bmatrix} \tag{1.2}$$

where $\vec{p}_{i,j}$ is a vector $(v_1, \ldots, v_c)$ where $c$ is the number of channels (bands) and the components $v_{j1}, j1 \leq c$ are the spectral values of each channel. A sliding window $W(i,j)$ is defined as that part of the image $I$ currently being used for (pre-)processing the data:

$$W(i,j) = \begin{bmatrix} \vec{p}_{i-k,j-k} & \cdots & \vec{p}_{i+k,j-k} \\ \vdots & & \vdots \\ \vec{p}_{i-k,j+k} & \cdots & \vec{p}_{i+k,j+k} \end{bmatrix} \tag{1.3}$$

where $k+1 \leq i \leq N-k$ and $k+1 \leq j \leq M-k$. Normally $k = 1, 2, \cdots$.

---

[3]Also know as ground resolution.

[4]The Landsat-TM image quality is examined in Wrigley *et al.* [121]. They examine the TM data quality in terms of band-to-band registration, periodic noise and spatial resolution. For all the bands (except the thermal one) the misregistrations are reported to be within half a pixel.

[5]Système Probabtoire pour l'Observation de la Terre.

[6]European Remote Sensing satellite.

| | | | |
|---|---|---|---|
| Satellite | Landsat 4-5 | SPOT 1-3 | ERS 1-2[7] |
| Organisation | NASA[8] | CNES[9] | ESA[10] |
| Bands used ($10^{-6}$m) | 1 0.45-0.52 Blue<br>2 0.52-0.60 Green<br>3 0.63-0.69 Red<br>4 0.76-0.90 Near-IR[12]<br>5 1.55-1.75 Mid-IR<br>6 10.4-12.5 Thermal IR<br>7 2.08-2.35 Mid-IR | XS1[11] 0.50-0.59 Green<br>XS2 0.61-0.68 Red<br>XS3 0.69-0.89 Near-IR<br>P[13]0.51-0.73 | 5.3 GHz C-Band polariza tion VV |
| Orbit | sun-synchroneous | idem | idem |
| Altitude (km) | 705 | 832 | 785 |
| Instruments | TM[14] | HRV[15] | AMI[16] |
| Sensing type | passive | passive | active |
| Sensors | scanning mirror | pushbroom CCD[17]array 3000 (XS)<br>6000 (P) | antenna |
| Ground resolution | 30m | 20m (XS) - 10m (P) | 25m[18] |
| Repeat cycle (days) | 16 | 26 | 3[19], 3[20], 35[21], 169[22] |
| Off-nadir viewing | No | Yes | No |
| Byte/pixel | 8 | 8 | 16 |

Table 1.1: different platform characteristics.

[7]ERS-2 to be launched early 1995.
[8]National Aeronautics and Space Administration.
[9]Centre National d'Etudes Spatiales.
[10]European Space Agency.
[11]Multi-Spectral.
[12]Infra-Red.
[13]Panchromatic.
[14]Thematic Mapper.
[15]Haute Resolution Visible.
[16]Active Microwave Instrument.
[17]Charged Coupled Device.
[18]for PRecision Image, PRI is one of the products available from ERS-1.
[19]during the initial commissioning phase.
[20]will operate for limited periods twice during the mission, to ensure highly repetitive coverage of ice zones during Arctic winters. The main limitations of a three day cycle are the restricted global coverage for the imaging SAR and the wide separation of the Radar Altimeter (The Radar Altimeter is a nadir-pointing pulse radar designed to measure the echoes from ocean and ice surfaces) tracks.
[21]enables SAR imaging of every part of the Earth's surface.
[22]employed late in the mission.

### 1.2.2 Image classification

Schowengerdt [99] defines classification as the decision-making technique which decides how to categorize a pixel from its spectral features (values). The goal of classification is to objectively map areas on the ground which have a similar *spectral signature*. This is used to denote the spectral response of a target. The labels assigned to the individual pixels, represent *(spectral) classes*. Because of the decision making process it may be that the spectral classes assigned do not correspond to the classes of ground objects. For example the class forest is composed of several land-cover types, each having a different spectral signature. Classification constitutes the following activities: i) the *training stage* ii) the *classification stage* and iii) the *output stage*.

The training procedure can be *supervised* or *unsupervised*. In the supervised approach the training process begins with the selection of training areas that are representative, homogeneous examples of each information category used in the classification stage. The range of variability for the category must be included. One important statistical aspect of selecting training data is that a sufficient number of pixels must be used to estimate the class signature properties accurately. (Typically 10 to 100 pixels per class per feature (i.e. spectral band)). Once the training data is selected the feature signatures of the pixels are calculated and used to recognize pixels with similar signatures throughout the image.

Unsupervised training involves algorithms which examine a large number of unknown pixels and divide them into a number of classes based on natural groupings present in the image values. The basic premise is that values within a given cover type should be close together in the measurement space, whereas data in different classes should be comparatively well separated. The classes that result from unsupervised classification are spectral classes, which are compared to some of the reference data. One of the more common methods is the K-means algorithm (or ISODATA algorithm [99], for a statistical description of the algorithm, see e.g. Devijver and Kittler [23]).

In the classification stage the pixels are assigned to the classes. Information gathered in the training stage may be used in the process. Different types of classifiers exist.

Maximum likelihood [99] is a widely spread and used classifier, while neural network classifiers are coming into use.

In the output stage the classification results are transformed into a user friendly format and often presented as a map.

However, classification of remotely sensed imagery is a difficult process, due to the presence of isolated and mixed pixels. Therefore, dealing with these pixels requires a special approach, e.g. Drake & Settle [24], Harrison *et al.* [40], klein Gebbink [57].

The effects of spatial resolution on the classification of TM data in comparison with MSS[23] is examined in Irons *et al.* [44]. They noted that the increase in spatial resolution from 80 (MSS data, $2^6$ grey levels) to 30 (TM data, $2^8$ grey levels) meters caused a 6.1% decrease in accuracy, when only 'pure' pixels were

[23]Multi-Spectral Scanner.

classified (with a maximum likelihood classifier). This decrease was attributed to increased within-class spectral variability at TM resolution. At the 'pure-plus-mixed-pixel' case almost no difference in accuracy is reported.

## 1.3 Applications of remote sensing

In the last decade the European Union (EU) has launched several continent-wide and global projects using RS, e.g. MARS[24], TREES[25], FIRS[26] etc. The MARS project, was started in 1987 as a 10 year pilot project for the application of RS in collecting agricultural statistics [71]. The main reason for launching the MARS project was that national systems of agricultural statistics differ in their approach to conventional surveys, and in the resources they have available. Furthermore the quantity and the quality of agricultural information gathered, especially on crop production, was not always adequate to its uses. Currently, the EU spends a large amount of money on agricultural subsidies. In order to control this cash flow (and to combat fraud on subsidy claims), the EU wants to have good insight into agricultural production. RS seems a promising technique for upgrading the agricultural statistics system.

The primary necessity is to distinguish, identify and measure the area of crops of significant importance. The second is to estimate production in time to make decisions. When such basic statistics are available, then forecasting can be considered.

In order to improve crop estimation required by the EU, image processing techniques like segmentation can be applied to pre-process the data. To extract information about crop yields or parcels, per pixel classifiers are widely used. A problem of this approach, is that irregularities within the spectral domain of the classes influence the performance of the classifier. For example pixels that are known to belong to the same 'thematic' class from ground surveys but having a slightly different spectral signature due to within-field variance, can be put into different classes. The result is that the final classification is very noisy with numerous isolated pixels. In order to improve the accuracy of the classification one of the possibilities is to perform a segmentation prior to the classification stage. The information resulting from the segmentation process can be used to improve the classification process (e.g. Janssen & van Amsterdam [46], Stakenborg [103]).

## 1.4 Existing segmentation approaches and definition

Segmentation has a long history in the image processing literature and many methods have been proposed, with various degrees of complexity. The existing methods for segmentation can be broadly divided into three categories, as given by Ballard & Brown [4]:

- ▷ Edge finding.
  In these methods discontinuities in the spectral domain of the image, strong

[24]Monitoring Agriculture with Remote Sensing techniques.
[25]TRopical Ecosystem Environment observations by Satellite.
[26]Forest Information from Remote Sensing.

enough to be detected, are searched for. These discontinuities are detected as edges, which are then composed by edge linking methods into a more elaborate structure representing a boundary of a homogeneous image class. Many types of operator have been developed to find edge fragments (e.g. Hueckel [42, 43], Lacroix [60], Roberts [85], Robinson [87], Canny [13]).

▷ Region growing.
In these methods an attempt is made to group pixels with similar characteristics into contiguous regions. The similarity property may be defined in terms of radiometric closeness or by some other criterion such as conformity with a local texture measure. Again several techniques exist for performing this (e.g. Chow & Kaneko [16], Kanellopoulos *et al.* [52], Zucker [123], Beaulieu & Goldberg [7], Tilton [107]).

▷ Map or knowledge based.
These methods use ancillary information to guide the segmentation procedure. This can be, for example, a digital map providing a background reference indicating where segment boundaries are likely to occur (e.g. Corr *et al.* [17], Janssen [45]). Such methods work reasonably well but are not generally usable in many parts of Europe. Knowledge based methods have also been used for controlling region growing or edge detection (e.g. Kanellopoulos *et al.* [52], Nazif & Levine [74], Janssen *et al.* [47], Ton *et al.* [110]).

The fact that much research is still devoted to this problem in RS, indicates that existing methods are still not providing the performance that application users of imagery require. To some extent it can be argued that the problem lies not with the methods but with the shortcomings of the image data combined with the lack of a definition of segmentation appropriate to RS application needs. To improve the performance of existing methods a segmentation method has been developed which integrates different types of algorithms.

### A definition of segmentation

A definition of segmentation has been given by Fu & Mui [34] (and others) as follows: $X$ is used to denote the grid of sample points of the image -i.e. the set of coordinate pairs $\{i, j\}$ where $i = 1, 2, 3, \ldots, N$ and $j = 1, 2, 3, \ldots, M$ (N, M are the numbers of pixels in the x and y directions respectively). Let $Y$ be a nonempty subset of $X$ consisting of contiguous 4(8)-connected pixels. Then a uniform predicate $P(Y)$ is defined which assigns the value TRUE or FALSE to $Y$ depending only on properties relating to the brightness matrix $f(i, j)$ (i.e. the spectral signature) for the points of set $Y$. Also $P$ has the property that if $Z$ is a non-empty subset of $Y$ then $P(Y) =$ TRUE implies that $P(Z)$ is always TRUE. A 'segmentation' of the grid $X$ for a uniformity predicate $P$ is then a partition of $X$ into disjoint non-empty subset $X_1, X_2, \ldots, X_n$ such that the following conditions hold:

▷ $\displaystyle\bigcup_{i=1}^{n} X_i = X$

▷ $X_i, \quad i = 1, 2, 3, \ldots, n$ is connected

▷ $P(X_i) = \text{TRUE}$ for $i = 1, 2, 3, \ldots, n$

▷ $P(X_i \cup X_j) = \text{FALSE}$ for $i \neq j$

where $X_i$ and $X_j$ are adjacent (4(8)-connected at some point).

## 1.5 Problems in segmentation

Some of the problems arising in segmentation of remotely sensed satellite imagery can be understood by reference to an artificial image sub-window, see figure 1.3. The image window can be divided into three separate areas: two segments (re-

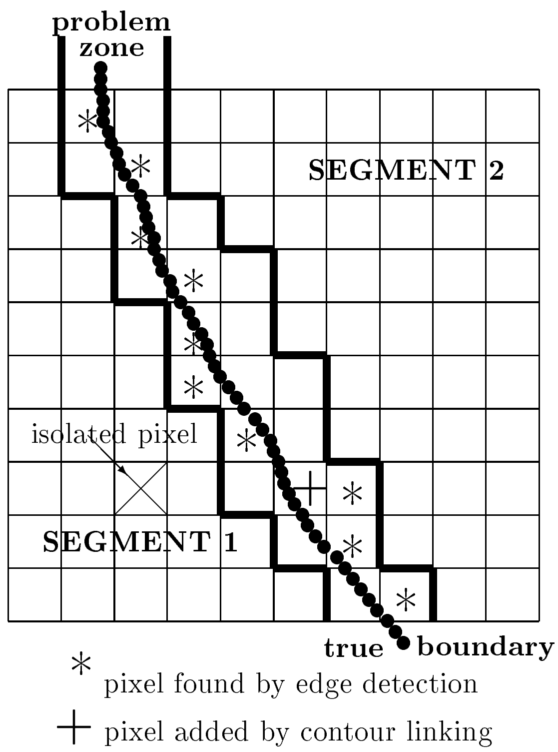



Figure 1.3: image window with segmentation problem zone.

spectively segment 1 and segment 2) which are true land cover segments in which the pixels are radiometrically very similar (though not necessarily equal) plus a 'problem zone' in which the pixels' radiances are impure and do not match either segment well. This is a common situation in satellite imagery where a transition zone between neighbouring land cover features is present because of the spectral mixing taking place in the border pixels which partially overlap more than one true radiometric class. These pixels may mix two land cover types on either side of the zone with features which are small (compared to the satellite resolution) separating them (e.g. road, river, fences, ditches, bare soil etc.). In the image sub-window the problem pixels account for a significant proportion of the total area.

In practice with real imagery, the total proportion of problem pixels may be less than that shown in figure 1.3, although high proportions are still possible. For example in some of the Mediterranean countries of the EU a high proportion of fields have an area of only 1-2 ha., i.e. there are many fields which occupy only $5 \times 5$ SPOT or Landsat-TM pixels. With 1-2 mixed pixels around the boundaries of such fields the proportion of 'problem' pixels can easily be as high as 30% of the entire image. In applications requiring accurate area estimates of crop production, for example, an error level of the order of 30% is not acceptable.

Another problem appearing in remotely sensed imagery is the within-field variation. Due to natural causes e.g. wet spots, dry spots, different soil types, diseases etc., the spectral value of neighbouring pixels (belonging to the same field) are not necessarily similar. The isolated pixel in the figure might be such a pixel.

In the absence of background map information for map guided segmentation the two standard segmentation approaches that can be applied to such an image would be either region growing or edge detection (or a combination of the two e.g. Pavlidis & Liow [81], Schoenmakers *et al.* [95]). The problems of using either method for the segmentation of remotely sensed imagery are described in the following sections.

### 1.5.1 Problems of region growing

A region growing method looks for pixels with similar characteristics to adjacent segments and joins them onto these segments if the similarity criterion is satisfied. Every time a new pixel is joined to a segment the characteristics of that segment are re-defined. The addition of a pixel to a segment is therefore dependent upon an automatically or manually set similarity threshold, although the segment 'mean' which is used as the target of the threshold can drift as more pixels join the region and share their properties with it. The 'growing' of regions continues until the problem zone is reached where the pixel characteristics no longer conform to the criteria set for region membership, and the region growing stops. Pixels inside the problem zone may, however, either be grouped into regions themselves or not be assigned to any region at all, i.e. they remain as isolated pixels. In some places sensor noise or irregularities in land cover features (e.g. too much bare soil showing through a vegetation canopy in one small area of a field) also leaves isolated pixels in the middle of otherwise homogeneous segments. This problem occurs frequently in remotely sensed imagery.

The result of the region growing process is thus typically a non-optimal segmentation in which many uncertainties exist. If the similarity threshold is set too low the growing process will leave many pixels unassigned to segments. On the other hand if it is set too high, segments representing different land cover parcels will be incorrectly merged together.

### 1.5.2 Problems of edge finding

An edge finding method applied to figure 1.3 (e.g. Lacroix [60]) will calculate for every pixel a gradient magnitude and gradient direction. By also setting a threshold certain pixels are selected as edge candidates. Because the changes in pixel radiances are greatest in the problem zone it will find the edge candidates there. If a good threshold is used only one edge candidate pixel will be found along a transect cutting across two adjacent land cover parcels as shown in figure 1.3. However, it is difficult to set such a threshold and normally there are errors of commission (i.e. too many edge candidates) or alternatively errors of omission (i.e. too few edge candidates). Even if a good threshold can be found (possibly adaptively over the image) the edge candidates may not be contiguous and breaks will occur. For example in a typical situation the pixels marked '$*$' in figure 1.3 may be found by an edge detection and following procedure. However in order to form a segment boundary any remaining gaps will need to be bridged. At this stage, heuristic edge linking methods need to be utilized to form a closed boundary. This may permit the inclusion of pixel '$+$' allowing a continuous 8-connected boundary to be formed. Edge linking, however, can be difficult to implement. The use of 8-connectivity instead of 4-connectivity for edge following has the advantage of requiring less pixels to define the required physical boundary. Once the edge following procedure has been carried out the image is divided into separate areas which include not only the pure segment pixels but also some of the 'impure' problem zone pixels. In addition the edge pixels are taken as a separate image class and do not belong to any segment.

### 1.5.3 Comparison of the two approaches

For both methods described above it is necessary to reduce the number of uncertain pixels after segment definition. For the region growing approach, the difficulty is to assign pixels of the problem zone to the segments. In principle, they can be assigned so that the determined boundary is between pixels. This is a considerable advantage because no information is lost in having 'edge pixels' which are necessary in the edge finding approach. However the inclusion of problem zone pixels in the 'homogeneous' segments breaks the standard definition of segmentation. Assignment of the problem zone pixels is easier in the edge finding method because the segments are defined to start either side of the defined edge. However the edge pixels still remain unassigned.

## 1.6 What is a good segmentation method?

Answering this question is difficult. Until now different classes of problems called for different segmentation methods. If a segmentation operator, hereafter called segmentator, gives reasonable results for one type of image, e.g. an indoor image, it is not assured that it gives satisfying results when applied to another type of image, e.g. a remotely sensed image. Also the context in which the segmentator is applied has its importance, i.e. how is the final result you want to obtain, defined. This has led to a wide variety of segmentation algorithms. Ideally, one generic segmentation algorithm would exist for all kind of problems, making all existing methods superfluous. Another point is that judgment of a segmentation result in many cases is done by humans, because quantitive measures do not always exist. This means that for one interpreter the result is judged as 'good' and by a second one as 'poor'. The goal of this dissertation is not to develop a generic algorithm, but to develop an algorithm which will be applied to a certain class of problems: the segmentation of remotely sensed optical imagery. Also, within this class, certain distinctions should be made. Do we develop the segmentator for mapping, or is it developed for improving classification results? From this point of view and for this specific class of problems, a definition can be given for a 'good' segmentator.

The segmentator is defined to be 'good' if the results of the per field classifier are more accurate than a per pixel classification. This will be the measure used in a quantitive way. For mapping, the segmentator is defined to be good if clearly visible and distinctive objects within the image have not disappeared after segmentation. This somewhat vague qualitative measure means that rail tracks, roads and rivers etc. should still be visible and recognizable as separate objects after the segmentation. The segmentator developed in this dissertation tries to fulfill both criteria.

The process of segmentation can be improved if several segmentation methods are combined into a hybrid one which makes use of the strengths of either method (e.g. Pavlidis [80], Pavlidis & Liow [81], Haralick & Shapiro [39]). However this does not mean that the resulting algorithm is applicable to a broader class of problems with improved results.

## 1.7 Hybrid segmentation method developed

The aim of this project has been to develop a 'segmentation' approach for images of agricultural areas which not only combines region growing with edge detection but also uses an adaptable segmentation predicate using rules for uncertain pixel assignment. The basis of the approach is presented in figure 1.4. The different modules of the algorithm are explained briefly below.

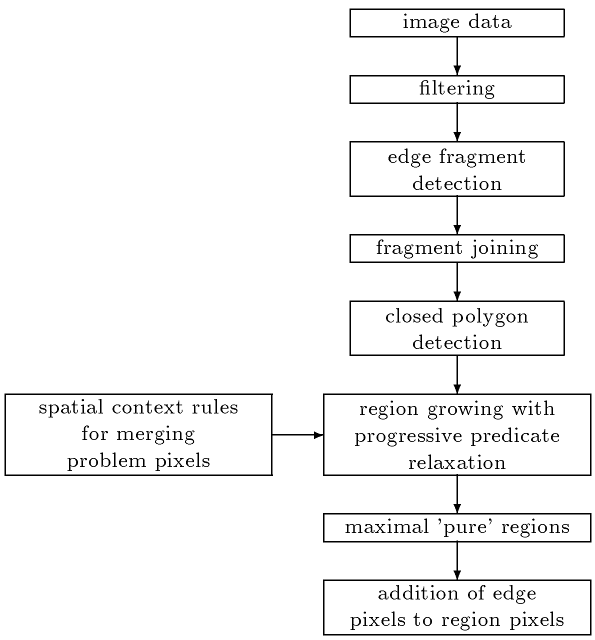



Figure 1.4: proposed segmentation methodology.

- ▹ Filtering can be performed in order to remove spectral noise.
- ▹ Edge finding is performed with the aim of forming single pixel wide 8-connected edge fragments.
- ▹ The separate edge fragments are connected into closed polygon boundaries (note that inside the polygon boundaries there should be some regions of relatively 'pure' pixels found by the region growing).
- ▹ Region growing is performed using a strict threshold to find 'pure' crop fields (although because of the use of a strict threshold isolated pixels may remain inside these regions, or due to the within-field variance several 'pure' crop fields (having the same crop) can be found).
- ▹ Spatial context rules are used to join remaining uncertain pixels (i.e. non-region, non-edge pixels, including isolated pixels) to segments. This needs to be done iteratively by progressively relaxing the segment uniformity predicate (although the segment means are not updated).

This approach enables us to capture as many pixels as possible within a meaningful crop parcel and to maximize the areal extent of the parcel even when the border zone contains pixels which are badly mixed. Genuine road or track features are also maintained. The method initially creates three categories of pixels. 'Pure' region pixels are defined by the strict constraints of the region growing uniformity predicate (P), 'edge pixels' (defined in terms of the edge finding algorithm) and remaining 'uncertain' pixels.

An important side effect of the production of 'pure' regions by our procedure is that a 'good' spectral signature can be obtained for the crop in the 'segment' using the properties of the segment mean. This spectral signature can then be analyzed to determine the crop state of health and has potential for quality checking and yield prediction.

Two other even more important side effects of first performing edge detection followed by edge linking and subsequent region growing as shown in figure 1.4, are that:

- ▹ over-growing of regions is normally overcome. This is because regions are not allowed to grow over an edge, which is a part of a closed polygon, and,
- ▹ edges not found by the edge detection can be found by the region growing. As the region growing is done within a closed polygon, the area within a closed polygon may be divided into separate regions, which are separated by edges (which were not detected by the edge detector).

This way the knife cuts at two 'edges'; region growing can not over-grow existing edges, and edges not detected by the edge detection/linking stage, may be detected at the region growing stage.

Moreover by dividing the image using the closed polygons into areas, the method is especially tailored for parallel implementation. Every area can be region grown on a separate processor.

## 1.8 Computing resource issues

A Landsat-TM image consists of 5760 rows, 6100 columns and 7 channels per pixel, resulting in a total of about 245 Megabytes of data. Currently available workstations have less than this amount of main memory (RAM[27]) It is therefore still impossible to store full scene images in memory. This means that data must be swapped from RAM to disk and vice versa. This swapping can be done by the operating system. But for portability reasons a flexible user defined swapping method is preferable.

Another problem not to be underestimated is the CPU[28] time needed. A method for segmentation can only be useful if the processing time remains within certain user defined limits. Because of the swapping of parts of the image from RAM to disk, the method is slowed down significantly. Optimisation in I/O[29] operations is needed, and there must be a tradeoff between RAM usage and disk-I/O.

In order to decrease computation time, parallel processing techniques can be employed. By dividing the problem into several smaller problems being computed in parallel on distinct processors, important speed-ups can be achieved.

The segmentation method treated in this dissertation is defined so that implementation is possible either on a one processor configuration or on a multi-processor configuration. This brings us to the two main constraints while developing the method:

1. The first constraint is that processing of an image on a one processor system is feasible within a reasonable time.
2. The second constraint is that the method also should offer possibilities for parallel implementation(s).

The developed algorithm fulfills both goals. Full scene images are segmented within reasonable time (i.e. a few hours) on one processor. The second goal follows directly from the manner in which the algorithm has been implemented.

[27]Random Access Memory.
[28]Central Processing Unit.
[29]Input Output.

# Chapter 2

## Existing Techniques for Image Segmentation

As mentioned in the first chapter several categories of segmentation methods exist. In this chapter two of these categories, and combinations of these, are discussed: region growing methods and edge detection methods. The former look for adjacent pixels with similar spectral signatures and try to group these into regions. The latter look for adjacent pixels whose spectral signatures differ significantly. It must be clear that these are complementary. The advantages and disadvantages of both categories have already been summarised. Surveys of segmentation techniques are given by Fu & Mui [34], Haralick & Shapiro [39], Zucker [123], and Pavlidis [80].

Pavlidis is very critical and negative in his examination. Some of his remarks are given here.

- ▹ A comparison of the results of the first papers (early 1960s) on the subject with the most recent literature (1986) suggests that only small improvements have been made.
- ▹ Probably the most urgent need is the integration of methodologies.

Especially the second remark is a reason to develop a method which integrates methodologies trying to gain larger improvements.

Pavlidis [80] states in his paper that edge detection is in some ways easier than detection of uniformity (i.e. region growing). For the latter a model of the image must be present. For the former it may suffice to detect only discontinuities in brightness. Of course the presence of noise makes detection of such discontinuities unreliable. Haralick & Shapiro [39] view segmentation as a clustering process.

In this chapter the following sub-division of segmentation techniques will be used.

1. Edge methods
2. Region methods
3. Hybrid methods

### 2.1 Edge methods

According to Haralick & Shapiro [39] edge operators assign a property vector to each pixel where the property vector depends on the $K \times K$ neighbourhood of the pixel. Similarity between pixels is established as a function of neighbouring pixel values.

Pavlidis [80] states that all techniques for edge detection face the following dilemma: If too large a neighbourhood is selected, more than one edge may be included, or an edge of complex shape so that their basic assumptions do not hold. If too small a neighbourhood is selected, then the estimation of the parameters is not reliable.

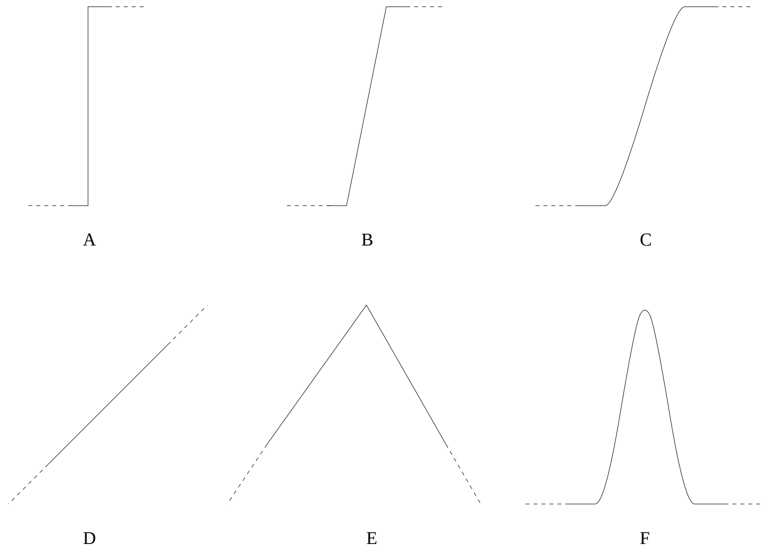



Figure 2.1: A) sudden step edge B) slanted step edge C) smooth step edge D) planar edge E) roof edge F) line edge.

Davies [19] differentiates several types of edges, see figure 2.1, where Fu & Mui [34] divide edge operators into: i) gradient based operators and ii) functional based operators.

### 2.1.1 Gradient based edge operators

Gradient based edge detectors detect within a given neighbourhood a (abrupt) change in pixel value. The difference between adjacent pixels in grey values is defined as the gradient.

Mathematical gradient operators are defined as:

$$\nabla f(x,y) = \left( \frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y) \right) \ , \tag{2.1}$$

where $\nabla f(x,y)$ is defined as the gradient at position $(x,y)$, $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$ are the first derivatives of the pixel values $f(x,y)$ with respect to the $x$ and $y$ directions.

The magnitude of the gradient $\nabla f(x,y)$ is defined as:

$$|\nabla f(x,y)| = \sqrt{\left( \frac{\partial f}{\partial x}(x,y) \right)^2 + \left( \frac{\partial f}{\partial y}(x,y) \right)^2} \ . \tag{2.2}$$

The direction of $\nabla f(x, y)$ is defined as:

$$\phi = \arctan\left(\frac{\frac{\partial f}{\partial x}(x,y)}{\frac{\partial f}{\partial y}(x,y)}\right) \quad . \tag{2.3}$$

The following methods are all based on the digital approximations of variations of equation (2.1) which will produce a high edge magnitude where there is an abrupt change in gray level and a low edge magnitude where there is little change in gray level.

The simplest form of an operator which detects abrupt changes in both $x$ and $y$ directions is the following gradient operator:

$$\frac{\partial f}{\partial x}(x, y) = f(x + 1, y) - f(x, y) \tag{2.4}$$

for detection in $x$ direction and

$$\frac{\partial f}{\partial y}(x, y) = f(x, y + 1) - f(x, y) \tag{2.5}$$

for detection in $y$ direction. This idea is generalized in defining two templates, as given in figure 2.2(a). By taking the convolution of these templates and the image window, where $(x, y)$ is the upper left corner in the window, the values for $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ can be computed, leading to the edge magnitude and edge direction.

$$\underset{\frac{\partial f}{\partial x}}{\begin{array}{|c|c|}\hline -1 & 1 \\ \hline 0 & 0 \\ \hline\end{array}} \quad \underset{\frac{\partial f}{\partial y}}{\begin{array}{|c|c|}\hline -1 & 0 \\ \hline 1 & 0 \\ \hline\end{array}} \qquad\qquad \underset{\frac{\partial f}{\partial x}}{\begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array}} \quad \underset{\frac{\partial f}{\partial y}}{\begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array}}$$

(a) (b)

Figure 2.2: (a) templates simplest form and (b) Roberts templates.

Roberts [85] uses a different version of this approach. He measures the changes in the $x$ and $y$ directions diagonally, resulting in the templates given in figure 2.2(b) for $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ computation. A disadvantage of both operators is that because of the small window size both operators are very susceptible to noise.

In order to be less noise dependent Sobel [26] (figure 2.3(a)), and Prewitt [84] (figure 2.3(b)), define $3 \times 3$ templates where $(x, y)$ is the central pixel in the window.

Kirsch [55] not explicitly computed the edge magnitude and edge direction using equations 2.2 and 2.3 but defined eight templates of which the in-product is taken with the image window as shown in figure 2.4. The eight templates can be

(a)

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$\frac{\partial f}{\partial x}$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

$\frac{\partial f}{\partial y}$

(b)

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

$\frac{\partial f}{\partial x}$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

$\frac{\partial f}{\partial y}$

Figure 2.3: (a) Sobel templates and (b) Prewitt templates.

simulated by the following operator, which reports the magnitude and direction associated with the maximum match, i.e. the highest output value, as follows:

$$\mathrm{MAX}\left[1, \max_{i=0}^{7} |5(a_i + a_{i+1} + a_{i+2}) - 3(a_{i+3} + \cdots + a_{i+7})|\right] \tag{2.6}$$

where $a_0, \ldots, a_7$ are the eight 8-connected neighbours of p, as shown in figure 2.4 and the subscripts are evaluated modulo eight.

| | | |
|---|---|---|
| $a_0$ | $a_1$ | $a_2$ |
| $a_7$ | p | $a_3$ |
| $a_6$ | $a_5$ | $a_4$ |

Figure 2.4: neighbours of pixel p.

Prewitt [84] also defined 8 templates which can be simulated by the following operator:

$$\max_{i=0}^{7} \{a_i + a_{i+1} + a_{i+2} + a_{i+3} + a_{i+7} - 2p - a_{i+4} - a_{i+5} - a_{i+6}\} \tag{2.7}$$

where $a_0, \ldots, a_7$ are the eight 8-connected neighbours of p, as shown in figure 2.4 and the subscripts are evaluated modulo eight.

Robinson [87] describes a system which is suitable for combining the detection and coding of visually significant edges in natural images. It uses $3 \times 3$ compass gradient masks. Furthermore he uses the concept of local connectivity of an edge direction map. The method works also for operators like Sobel [26] and Kirsch [55]. He uses 8 templates which can be simulated by the following operator:

$$\max_{i=0}^{7} \{a_i + b \cdot a_{i+1} + a_{i+2} - a_{i+4} - b \cdot a_{i+5} - a_{i+6}\} \tag{2.8}$$

where $b$ is 1 or 2. Robinson reports a better detection in diagonal direction for the operator with $b$=2. Moreover he notes that for the magnitude and direction the first four templates are sufficient, which means a reduction in computation time.

Frei & Chen [33] generalized the gradient idea and divided the 'problem' space into an edge sub-space, a line sub-space, and a point sub-space, where the edge and line sub-space are generated by a set of templates of four templates each. The point sub-space is generated by a single template. The total of nine templates $T_1, \ldots T_9$ form together the basis of the nine-dimensional space.

Equation (2.9) can be seen as the projection of the $3 \times 3$ image window $W$ into the nine basis functions $T_i$.

$$\Theta = \arccos \sqrt{\frac{\sum_{i=1}^{4}(W \cdot T_i)^2}{\sum_{i=1}^{9}(W \cdot T_i)^2}} \tag{2.9}$$

One way of measuring the 'edgeness' of a local area in the image is to measure the relative projection of the image into the edge basis functions. The relative projection into the particular 'edge sub-space' is given by equation 2.9. Thus if $\Theta$ is smaller than a user defined threshold, there is an edge to be reported.

For an exhaustive examination of the Frei-Chen operator see chapter 3. In that chapter the shortcomings of the Frei-Chen edge detector in their basic definition as noticed by Lacroix [60] are given. Park & Choi [77] have remarks on Lacroix's adaptations, see chapter 3. Furthermore they propose a fast implementation. Park [76] gives a normalization scheme for the nine templates and a Fourier implementation of the Frei-Chen masks. An improvement in edge magnitude computation for the Frei-Chen operator is also discussed in chapter 3.

Meer *et al.* [69] discuss existing template matching edge detectors. They developed a new method in which each mask responds to one ideal stimulus, and the confidence about the pixel is made based on a measure of the match to the ideal stimulus. So they define 8-stimulus masks. The $3 \times 3$ neighbourhood is defined as a 9-dimensional space (like Frei & Chen) and using the stimulus templates they deduce nine matching templates. Two confidence measures P and Q are defined. P gives the confidence in how well the edge fits with the model. Q gives the confidence for weak edges and can be used if these are to be rejected. A comparison with the Frei-Chen method [33] is given.

### 2.1.2 Functional approaches

Fu & Mui [34] consider edge detection as an approximation problem (i.e. models of edges developed and fitted with the image data).

Hueckel [43] defines $D$ to be a set of pixels, arranged so as to best approximate a disk. Hueckel defined constraints for optimal design of the operator:

1. Nullifying rounding errors on the periphery of the disk i.e. the error introduced by transformation of the continuous domain (the disk) to the discrete domain, the set of pixels $D$ should be minimal.

2. The weight of the input data decreases towards the disk's periphery.
3. An operator which locates edges needs not be sensitive to noise of high spatial frequencies.
4. the computing time for the operator should be minimal.

The question to be answered is: is the intensity of a pixel $(x, y)$ within $D$ the noisy form of an ideal edge which is characterised by a step function? Mero & Vassy [70] give a optimized version of Hueckel's filter.

According to Pavlidis [80] it sometimes makes sense to pre-process the image by a transformation that enhances the high contrast areas. The Laplacian is such a transformation which is defined as:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y) \tag{2.10}$$

where $\frac{\partial^2 f}{\partial x^2}(x,y)$ and $\frac{\partial^2 f}{\partial y^2}(x,y)$ are the second derivative in the $x$ and $y$ directions respectively. A simple approximation of $\nabla^2 f(x,y)$ is the function $L(x,y)$:

$$L(x,y) = f(x,y) - \frac{f(x,y+1) + f(x,y-1) + f(x+1,y) + f(x-1,y)}{4} \tag{2.11}$$

A disadvantage of this operator is that no directional information is given and moreover it is very susceptible to noise. A more common use of the Laplacian is the detection of zero crossing in the output values, representing the edges in the image.

A well-known transformation is the Marr-Hildreth operator [65]. Marr & Hildreth combine two physical considerations to determine the optimal smoothing filter:

1. Intensity changes are detected separately at different scales in the frequency domain, reducing the range of scales over which intensity changes take place. The filter's spectrum should be smooth and roughly band limited in the frequency domain and,
2. It should be optimally localized spatially with a small spatial variance.

The optimal filter which obeys both considerations is the second derivative of a Gaussian $D^2$. Intensity changes have corresponding peaks in the first directional derivative, or equivalently have a zero crossing in the second directional derivative $D^2$. So the detection of intensity changes can be reduced to finding the zero crossing of $D^2$, in the appropriate direction. The zero crossings are to be found of:

$$f(x,y) = D^2[G * I(x,y)], \tag{2.12}$$

where $I(x,y)$ is the image $I$, $G$ the Gaussian function, and $*$ is the convolution operator. The Gaussian $G$ is defined in one dimension as:

$$G(x) = \left[\frac{1}{\sigma}\sqrt{2\pi}\right] e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.13}$$

where $\sigma$ indicates the standard deviation and thus the width of the Gaussian distribution. Marr & Hildreth assume the mean $\mu$ to be zero. The above can be rewritten by the derivative rule for convolutions as:

$$f(x,y) = D^2 G * I(x,y). \tag{2.14}$$

The problem remaining is the orientation to be associated with $D^2$. The only orientation independent second order differential operator is the Laplacian $\nabla^2$. So the loci of zero crossings are detected by searching for the zero values of the convolution $\nabla^2 G * I$. In two dimensions

$$\nabla^2 G(r) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{r^2}{2\sigma^2}\right] e^{-\frac{r^2}{2\sigma^2}} \tag{2.15}$$

where $r^2 = x^2 + y^2$ (Again the mean $\mu$ is assumed to be zero). The zero crossings of the output correspond to edges, while areas where the output is positive correspond in general to brighter parts and areas where the output is negative correspond to darker parts.
Different, faster versions of the Laplacian of a Gaussian are found in the literature (e.g. Forshaw [32]).

Haralick [38] defines the facet model. Facets are defined as polynomials in $x$ and $y$ approximating $f(x,y)$.

Haralick assumes that in each neighbourhood of the image the underlying gray tone intensity function $f$, which is assumed to be linear, takes the parametric form of a polynomial in the row and column coordinates. The underlying functions from which the directional derivatives are computed are represented as linear combinations of the polynomials in any polynomial basis set. A polynomial basis set which permits the independent estimation of each coefficient would be easiest to use.

Given an index set $R$ and an observed data value $d(r)$ (the pixel value for $(0,0)$), the fitting problem is to determine coefficients $a_0, \ldots, a_{N-1}$, ($N$ is the number of elements of $R$), such that

$$d(r) = \sum_{n=0}^{N-1} a_n P_n(r). \tag{2.16}$$

The approximate fitting problem is to determine coefficients $a_0, \ldots, a_K$, $K \leq N-1$ such that

$$e^2 = \sum_{r \in R}\left[d(r) - \sum_{n=0}^{K} a_n P_n(r)\right]^2 \tag{2.17}$$

is minimized. From the polynomial the first and second partial derivatives can be determined. The first partial derivatives at the pixel centre determine the gradient direction. With the direction being fixed to be the gradient direction, the second partial determines the second directional derivative. If the gradient is high enough and, if in the gradient direction, the second directional derivative has a negatively sloped zero-crossing inside the area of the pixel, then an edge is declared in the centre pixel of the neighbourhood.

Canny's operator [13] is currently widely used and referenced to. Canny defines three criteria relevant to edge detector performance: i) low error rate, ii) edge points must be well localized and, iii) only one response to a single edge.

Canny developed a one-dimensional model for detecting step edges with random noise. First he defines the detection and localization criteria and second he defines criteria for eliminating multiple responses to a single edge.

Canny defines the detection and localization criteria as follows: Let the impulse response of a filter be $f(x)$ and denote the edge by $G(x)$. Assume the edge is centered at $x = 0$. The response of the filter to this edge at its centre $H_G$ is given by the convolution integral:

$$H_G = \int_{-W}^{W} G(-x) f(x) dx \tag{2.18}$$

assuming the filter has a finite impulse response bounded by $[-W, W]$. The root mean squared response to the noise $n(x)$ only, will be

$$H_n = n_0 \sqrt{\int_{-W}^{W} f^2(x) dx} \tag{2.19}$$

where $n_0^2$ is the mean squared noise amplitude per unit length. The first criterion, the output SNR, is defined as:

$$\mathrm{SNR} = \frac{H_G}{H_n} \tag{2.20}$$

Localization is defined as:

$$\mathrm{Localization} = \frac{\left| \int_{-W}^{W} G'(-x) f(x) dx \right|}{n_0 \sqrt{\int_{-W}^{W} f'^2(x) dx}} \tag{2.21}$$

where $G'$ is the first derivative. Equations (2.20) and (2.21) are mathematical forms for the first two criteria, and the design problem reduces to the maximization of both of these simultaneously.

Canny states that the optimal detector for step edges is a truncated step, or differences of boxes operator.

In order to eliminate multiple responses Canny defines an upper bound for both the SNR and the localization, the first two criteria can be maximized. Thus the optimal detector for step edges is a truncated step, or differences of boxes operator. Examining the output of a differences of boxes edge detector finds that the response to a noisy step is a roughly triangular peak with numerous sharp maxima in the vicinity of the edge. To minimise the probability of declaring more peaks as an edge, the number of peaks needs to be limited. Ideally this means making the distance between peaks in the noise response approximate the width of the response of the operator to a single step. This width will be some fraction of the operator width $W$. By doing so the response of the filter will be concentrated in a region of width $2W$

Having defined the mathematics for the three criteria Canny derives a detector for step edges. The step edge $G(x) = A\mu_{-1}(x)$ where $\mu_{-n}(x)$ is the n-th derivative of a delta function, and $A$ is the amplitude of the step

$$\mu_{-1}(x) = \begin{cases} 0 & \text{for } x < 0; \\ 1 & \text{for } x \geq 0; \end{cases} \tag{2.22}$$

Having derived the one-dimensional step edge detector mathematically, he gives as an efficient approximation of the operator the first derivative of a Gaussian $G'(x)$ where

$$G(x) = e^{-\frac{x^2}{2\sigma^2}}. \tag{2.23}$$

The one-dimensional operator is extended into two dimensions.

Deriche [22] discusses Canny's operator and proposes an improvement by giving a one-dimensional digital recursive implementation of Canny's optimal filter. The operator is said to give a substantial saving in computational effort.

Fleck [31] illustrates and explains artifacts in the output of five finite difference edge finders, among which the Canny and Deriche operator. The artifacts discussed include gaps in boundaries, spurious boundaries, and deformation of region shape. Fleck concludes that most algorithms deliver similar output, but that the differences between previous implementations seem to be due to differences in thinning, threshold settings, sub-pixel interpolation algorithms etc.

Tagare & deFigueiredo [105] discuss the one-dimensional formulation of the localization performance of edge detectors. They show that the derivative of the Gaussian is the optimal edge detector, discuss Canny's operator and give a simpler derivation of his operator.

## 2.2 Region methods

Many region operators are based on heuristics and intuition. Following Pavlidis [80] most region growing methods partition the image into a set of regions $S$, where $S = \{S_0, \ldots, S_n\}$. The elements $f(x, y)$ of $S_i$ satisfy a uniformity predicate $P$ of the following type:

1. An approximation to the data in a region $S_i$ is selected from a family of functions $F$, usually $F$ is that of $\{f(x, y) = \text{const}\}$, and then an error norm $e$ is computed over it. If $e$ is below a threshold, $P(S_i)$ is true, otherwise it is false.
2. The approximation is carried out over $S_i$ and some subsets of $S_i$. Then the parameters of the approximation are compared, and if they are close, $P(S_i)$ is true.

Most region growing algorithms use a two step approach. To yield an acceptable result, first a partition with a strict uniformity criterion is found. This typically yields a very large number of regions and a second less strict criterion is applied for eliminating the small regions. One difficulty with eliminating small regions is that the order in which regions are examined influences the final result (this is known as the order-dependence problem).

In the overview of region methods given by Haralick & Shapiro [39], they are divided into the following types:

- ▷ measurement space guided clustering
- ▷ region growing
- ▷ spatial clustering
- ▷ split and merge

### 2.2.1 Measurement space guided spatial clustering

A clustering using only measurement space data is performed to define a partition in measurement space. After the clustering is done each pixel is assigned the label of the cell in the measurement space partition to which it belongs. It can be debated if this procedure is a segmentation operation as no spatial information is used in the clustering process.

An example is 'histogram mode seeking' in which it is assumed that homogeneous objects on the image manifest themselves in measurement space. A disadvantage however is that no spatial information is used and furthermore if there are many different objects in the scene, then object separation might be difficult.

A well-known example of measurement space guided clustering is ISODATA (For a description of ISODATA (K-means) see, e.g. Devijver & Kittler [23], Mather [67]). In this algorithm it is assumed that the clusters present in the data are compact. If the compactness of a cluster does not conform to a compactness measure (e.g. set of standard deviations for the cluster measured separately for each axis of the feature space), then the cluster is elongated in the direction of the axis concerned (requiring that the clusters are well separated). If the elongated cluster is defined 'large', then the cluster is split in two and the process is repeated.

### 2.2.2 Region growing

In this technique the image is scanned in some pre-determined way. The value of a pixel is compared to the mean of an already existing but not necessarily completed neighbouring segment. If the pixel's value and the segment's mean are close enough the pixel is added to the segment and the segment mean is updated. If there is ambiguity in selection of a close segment in cases where there are more than one, then the closest is taken. However if there are two regions close enough then the regions are merged and the pixel is added to the new region. If there is no region close enough then a new region is started having this pixel as its first member.

A well-known example of region growing is the one proposed by Brice & Fennema [12]. They partition the image into initial segments of pixels having identical intensity. Two heuristics for merging sequentially pairs of adjacent regions are developed. The boundary surrounding an initial region is composed of a set of simply connected, closed sequences of boundary segments (vectors). Each vector is assigned a strength proportional to the magnitude of the difference in gray levels perpendicular to it. The first heuristic (The Phagocyte Heuristic) is that two adjacent regions are merged if the boundary between them is weak. The length of the weak part of the boundary between two regions, is the number of vectors having a strength less then some threshold. The second (The Weakness Heuristic") is that two regions are merged if the weak portion of their common boundary is smaller than some set percentage of their total shared boundary.

Other implementations of this idea are given by Kettig & Landgrebe [54], and Pavlidis [79]. Kettig & Landgrebe initially split the image into $2 \times 2$ cells which are tested on uniformity using different statistical approaches (supervised and unsupervised). In case they are not uniform they are split in to separate cells. then they are classified using different classifiers. The combination of a supervised segmentation and a maximum likelihood classifier is reported to provide the best results.

Haralick & Shapiro [39] state that there is a potential problem with region growing schemes, which is their inherent dependence on the order in which pixels and regions are merged. The same is also noted by Fu & Mui [34] in their survey.

### 2.2.3 Spatial clustering

In this technique clustering in measurement space and spatial region growing are combined. In many techniques the histogram of the spectral values is used to define 'seed-points' from which the region growing starts.

The region growing method used in this dissertation, the 'best merge' initially proposed by Beaulieu & Goldberg [7] and Tilton [107], falls also into this category. Tilton discusses the order dependence problem as noted by Fu & Mui [34], Haralick & Shapiro [39] and, Pavlidis [80]. Order dependence indicates that the final result of a segmentation depends on where the process is started and in what order it takes place. To be independent of where the process started, both Beaulieu & Goldberg, and Tilton examine at each iteration the whole image context and the (spatially adjacent) region pair having the lowest similarity value is merged into a new region. The basic algorithm is given in algorithm 2.1.

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

Algorithm 2.1: principle of the best merge.

Beaulieu & Goldberg as well as Tilton define a similarity criterion for computation of the similarity value between pairs of regions. A disadvantage of the merging process is its computational load. Efficient algorithms have to be developed to make processing of even small sized images feasible within a reasonable time. Tilton defines sub-images in which the merging takes place. The sub-images defined are merged in parallel. Beaulieu & Goldberg give an algorithm which examines only adjacent pairs of regions and give a sequential algorithm.

For a mathematical description of the best merge approaches see chapter 5.

In Kanellopoulos *et al.* [52] a spatial clustering scheme is described based on the best merge. They initially split the image into $2 \times 2$ cells and test their uniformity. Non uniform cells are split. Then they initially try to merge the remaining $2 \times 2$ cells. Following this remaining single pixel cells are merged to the regions. Finally by defining adjacency rules remaining isolated pixels are merged to existing regions.

In Benie & Thomson [9] a clustering scheme is defined based on the best merge. The algorithm consists of five stages. In the first stage initial cells are defined. A certain maximum number of cells is allowed. Cells are merged if they satisfy a merging criterion. In the second stage input data is used to calculate the description parameters of each initial segment (i.e. mean, variance, standard deviation, number of pixels, minimum and maximum values and a list of neighbours). In the third stage the merging costs (i.e. in measuring the homogeneity) of adjacent segments is computed. In stage four the actual merging scheme as defined for the best merge is applied. In stage five the user fixes the desired number of segments to cut the segments tree[1] and to create the final segmented image. An automatic computation scheme, based on a modified student-$t$ test, is proposed to determine the final number of segments. The student-$t$ test is used to evaluate the significance of the difference of two means[2].

In Woodcock & Harward [120] a segmentation technique is defined based on the best merge. They use a multi-pass approach that allows slow and careful growth of regions while inter-region distances are below a threshold. In a first phase of the algorithm a global criterion is applied to achieve a level of merging that eliminates (i.e. merges) adjacent regions, whose inter-region distance falls below the minimum expected variation between regions. In this phase multiple merges are allowed per iteration (i.e. to reduce computation time). After the global threshold halts merging, there tend to be many isolated small regions (often isolated pixels) left in the image. In the second phase, local criteria in the form of region size constraints are then combined with the previously applied global constraints to achieve a higher degree of merging.

In Ait Belaid *et al.* [1] the HSWO[3] algorithm of Beaulieu & Goldberg [7] is adapted so that cartographic information is added as a separate channel to the input data. Regions are not allowed to over-grow the edges determined by the cartographic channel. The algorithm is tested on two test-sites and an improved classification on these test-sites is reported.

[1]Every segment can be seen as a node. When two segments are merged, a new node is created having the two nodes as sons. This way a hierarchy in merging is created and the tree obtained is called segments tree.

[2]after G.P. Genovese, April 1994.

[3]Hierarchical Stepwise Optimization Algorithm

### 2.2.4 Split and/or merge

The principle of the split and/or merge algorithms is that the image is divided into sub-images (size power of two) which are then split and/or merged if for a sub-image to be split/merged a statistical uniformity criterion is fulfilled. The split method initially starts with the whole image as one region whereas the merge method starts with every pixel as a separate region.

One of the first split and merge algorithms is found in Muerle & Allen [72]. They defined a region to be any portion of a complex image in which the statistical distribution of the grey levels is reasonably uniform. Initially they divide the image into cells of $2 \times 2$, $4 \times 4$, $8 \times 8$ pixels. Take a cell and merge it with a neighbouring cell, if and only if they are similar enough following some statistical tests. This is repeated until no statistical similar neighbours are found. Then take another cell and restart.

In Robertson [86] and Klinger [58] the splitting method begins with the entire image as the initial segment. Then it successively splits each current segment into quarters if the segment is not homogeneous enough.

Cross *et al.* [18] propose a split and merge segmentation procedure based on the one given by Chen & Pavlidis [15]. It uses a quad-tree approach. An intermediate level in the quad-tree is used to start the segmentation. Nodes are merged/split using a uniformity test. Cross *et al.* compute a texture measure based on average grey value difference for every pixel. This value is then assigned to the pixel which serves then as input to the algorithm. Their algorithm is especially developed and tested for multi-spectral images.

Also the split and merge operator is sensitive to the order dependence problem, the final segmentation result might depend on the initial situation.

## 2.3 Hybrid operators

In these techniques the edge operators and region operators are combined in such a way that advantage is taken of their separate strengths. The strength of the edge detector is that boundaries are placed in a spatially accurate way. Its weakness is that not all edges are connected into closed boundaries and therefore too large regions may result. The strength of the region operators is their ability to place boundaries in weak gradient areas (Haralick & Shapiro [39]).

'The combined edge operator and region operator does the obvious thing. Region growing is only done for non-edge pixels, that is region growing is not permitted across edge pixels' (Haralick & Shapiro [39]).

Pavlidis & Liow [81] give an example of a method which integrates edge detection with split and merge. Pavlidis & Liow first perform a split and merge algorithm with the parameters set so that an over-segmented image results. The segment properties (i.e. mean, standard deviation etc.) are compared and in case of 'equality' between adjacent regions, boundaries are eliminated. In order to do so rules are defined which integrate contrast with boundary smoothness, variation of the image gradient along the boundary, and the presence of artifacts reflecting the data structure used. Finally boundaries are modified by defining a merit function which consists of the following terms which are functions of $W(t)$ where $W(t)$ is a point on the boundary: i) the magnitude of the image gradient at $W(t)$ which is a measure of the contrast along the contour, ii) the curvature evaluated along the points $W(t)$ of the contour, and iii) the first order derivative of the phase part of the image gradient at point $W(t)$.

Pavlidis & Liow report that their method gives good results on 'tool' images. The performance on aerial photograph images is less convincing. In their opinion the combination of the two methods yields far better results than those of a single method.

In Le Moigne & Tilton [61] a method is described which integrates an edge detector (Canny [13]) into the best merge as implemented by Tilton [107]. The implementation as defined by Tilton generates a tree of hierarchical regions. The fusion of the edge data with the region data is as follows: Utilizing simultaneously the results from the best merge, and the edge data the iteration of the best merge which best matches the edge image, i.e. the best iteration is found.

The method proposed in this dissertation obeys the constraint that edges may not be over-grown and combines spatial clustering with edge detection.

## 2.4 Concluding remarks

In this chapter existing region growing and edge detection algorithms have been discussed. The edge detection method used in this dissertation gives improved results over the operators proposed by Roberts [85], Kirsch [55], Robinson[87] as shown in Frei & Chen [33]. Lacroix [60] notes artifacts in the nine dimensional basis as defined by Frei & Chen and proposes a new basis. In Schoenmakers [93] a method is proposed for improving edge magnitude computation.

The operator as proposed by Haralick [38] is reported to be superior to the one defined by Marr & Hildreth [65], but in Pavlidis [80] it is noted that other authors notice the opposite with adapted versions of Marr & Hildreth [65]. In Grimson & Hildreth [37] it is said that the implementation used by Haralick of Marr & Hildreth's filter is different from the implementation given by Marr & Hildreth [65].

Canny [13] proposed a model for the detection of step edges and notes that the operator can be extended to detect corners. Concluding it must be said that the operator used in this dissertation can be replaced by others like Canny [13], Deriche [22], Haralick [38], Marr & Hildreth [65], and Meer [69]. Implementations of these operators should be compared on the same imagery in order to give a quantitive comparison.

For the region operators the conclusions can be more distinct. The only operator not suffering the order dependence problem is the best merge as proposed by Beaulieu & Goldberg [7], and Tilton [107] and therefore seems the one and only operator adequate to the region growing problem. However, the problem of region over-growing still remains.

Looking at the comments on hybrid techniques as given in Pavlidis [80], Pavlidis & Liow [81], and Haralick & Shapiro [39] it can be concluded that hybrid techniques offer better possibilities than one single operator alone. Therefore a combination of a selected edge operator and the best merge seems the only combination to improve the results of a segmentation as given by the best merge alone.

A problem remaining within RS is the presence of image noise. Noise might confound an edge detection and following process, and thus lead to the detection of false edges having no existence in reality. For region growing however noise might lead to creation of isolated pixels. In order to suppress this behaviour a noise suppression method is required and was implemented for this study. The method adopted is described in appendix A. It follows the approach of examining the morphology of the ordered spectral values in an image window. This approach has also been described in Schoenmakers & Stakenborg [94]. Of course other filters like [2, 10, 11, 20, 62, 68, 73, 83, 102, 115, 116] can be used. An overview of existing filters can be found in [25, 66].

# Chapter 3

## An Improved Edge Detection Method

### 3.1 Introduction

The edge detection and contour following algorithm adopted for this study was initially developed by Lacroix [60]. The edge magnitude computation is based on applying a local operator (in a $3 \times 3$ window) developed by Frei & Chen [33]. After thresholding the edge magnitudes, the edge direction and edge magnitude are used to compute 'a Likelihood of Being an Edge pixel' (LBE) for every pixel. Using these values and again the edge direction a non-maximum deletion algorithm is applied in order to select the final one pixel wide edges.

The algorithm for edge magnitude computation is not always reliable for optical remotely sensed multi-spectral imagery, i.e. many visible edges are not detected. In this chapter it is described how the edge magnitude is adapted in such a way that its sensitivity increases. Hence more edges are detected.

Lacroix describes the algorithm for single channel imagery. Remotely sensed imagery however, is (mostly) multi-channel and therefore requires an adaptation of the original edge magnitude and direction computations. In the herein described adapted method every pixel has for every channel an edge magnitude and edge direction value, and a procedure for the selection of the channel with the 'best' value is defined.

### 3.2 Edge magnitude and direction calculation

#### 3.2.1 The Frei-Chen operator

Frei & Chen [33] formulate the problem of boundary element detection as follows: Given an image sub-area $W$ in the continuous domain, determine whether the sub-area contains a boundary element between two regions of different homogeneous luminances, as shown in figure 3.1(a).

They define models of ideal boundary elements. In the continuous image domain they define an:

▷ ideal edge element,
as a straight boundary passing through the centre of $W$, separating two regions of constant luminances $\overline{l_1}$ and $\overline{l_2}$. Adopting the convention $\overline{l_1} > \overline{l_2}$, the direction $\phi_e$ of the edge element is uniquely determined with respect to any arbitrary fixed direction. The magnitude is defined as $|\overline{l_1} - \overline{l_2}|$ and

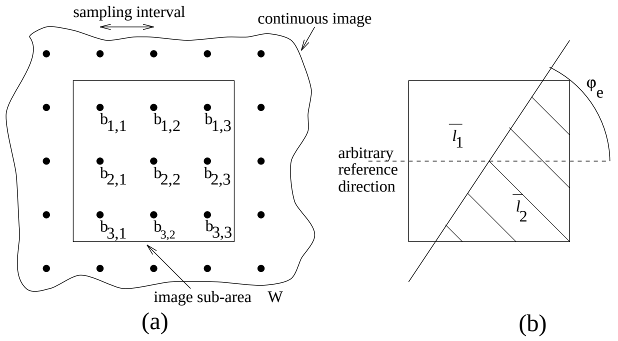



Figure 3.1: definition of image sub-area.

the orientation $\phi_e$ as $0 \leq \phi_e < 2\pi$. In figure 3.1(b) the boundary passing through the centre of $W$ (i.e. $b_{2,2}$), the angle $\phi_e$, the luminances $\overline{l_1}$, $\overline{l_2}$, and the arbitrary reference direction are shown.

▷ ideal line element,
as a straight strip of width approximately equal to one sampling interval, passing through the center of $W$, and of different $\overline{l_1}$ than its surrounding $\overline{l_2}$. It is therefore characterized by its magnitude $|\overline{l_1} - \overline{l_2}|$, its orientation $\phi_l, 0 \leq \phi_l < \pi$ and its polarity sign $(\overline{l_1} - \overline{l_2})$.

▷ ideal point,
as a point of brightness $\overline{l_1}$ different from the environment $\overline{l_2}$. It is therefore characterized by its magnitude $|\overline{l_1} - \overline{l_2}|$, and its polarity sign $(\overline{l_1} - \overline{l_2})$.

For the discrete case Frei & Chen first sample the image into a set of $N \times M$ samples (pixels).

For simplicity it is assumed that the number of channels is one, i.e. $c = 1$ in equation 1.2 and 1.3 and the vectors $\vec{p}_{i,j}$ are written as components $p_{i,j}$.

Frei & Chen consider the set of $n^2$ pixels $p_{i,j}$ of $W$ as an element of an $n^2$-dimensional vector space $\mathcal{B}$. The boundary element detection can now be formulated as: determine how well a vector $W$ of samples from an image sub-area 'fits' an ideal boundary element, not knowing its orientation a priori.

#### 3.2.1.1 Edge magnitude computation

Suppose the existence of an 'edge' subspace in $\mathcal{B}$. Find a set of $e$ orthogonal 'edge' basis vectors, $[T_1, \cdots, T_e]$ spanning that edge subspace, and expand that set with $n^2 - e$ 'non-edge' basis vectors to span $\mathcal{B}$. The orthogonal basis vectors $[T_1, \cdots, T_9]$ (for $n = 3$) as defined by Frei & Chen are given in figure 3.2.

Now consider the angle $\Theta$ (in $n^2$-space) between $W$ and the projection onto the edge-subspace, with

$$\Theta = \arccos \sqrt{\frac{\sum_{i=1}^{e} (W \cdot T_i)^2}{\sum_{i=1}^{n^2} (W \cdot T_i)^2}} \tag{3.1}$$

$$\begin{array}{cccc}
\begin{array}{|c|c|c|}\hline 1 & \sqrt{2} & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -\sqrt{2} & -1 \\ \hline \end{array} &
\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \sqrt{2} & 0 & -\sqrt{2} \\ \hline 1 & 0 & -1 \\ \hline \end{array} &
\begin{array}{|c|c|c|}\hline 0 & -1 & \sqrt{2} \\ \hline 1 & 0 & -1 \\ \hline -\sqrt{2} & 1 & 0 \\ \hline \end{array} &
\begin{array}{|c|c|c|}\hline \sqrt{2} & -1 & 0 \\ \hline -1 & 0 & 1 \\ \hline 0 & 1 & -\sqrt{2} \\ \hline \end{array} \\
T_1 & T_2 & T_3 & T_4
\end{array}$$

edge subspace

$$\begin{array}{cccc}
\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline -1 & 0 & -1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} &
\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & -1 \\ \hline \end{array} &
\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array} &
\begin{array}{|c|c|c|}\hline -2 & 1 & -2 \\ \hline 1 & 4 & 1 \\ \hline -2 & 1 & -2 \\ \hline \end{array} \\
T_5 & T_6 & T_7 & T_8
\end{array}$$

line subspace

$$\begin{array}{c}
\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \\
T_9
\end{array}$$

average

Figure 3.2: the 9 Frei-Chen templates.

where $e = 4$ for the edge subspace. The larger $\Theta$ $(0 \leq \Theta \leq \pi)$ the poorer the fit between $W$ and an element of the edge subspace. The decision strategy is to classify image sub-areas as containing an edge element if $\Theta$ is small. This can be done by thresholding equation (3.2)

$$r = \frac{1}{(W \cdot W)} \sum_{i=1}^{4} (W \cdot T_i)^2 \ . \tag{3.2}$$

The potential advantage of thresholding based on $\Theta$, is shown in figure 3.3. Two sub-area vectors $B_1$ and $B_2$ are shown, projected onto the 'edge' and 'non-edge' subspaces, respectively. $B_1$ has a large projection (and a large magnitude) onto the 'non-edge' sub-space and is rejected by Frei & Chen's thresholding, where it is accepted by a 'conventional' method. $B_2$ with its smaller magnitude but with a good fit onto the 'edge' sub-space is accepted by Frei & Chen's thresholding technique where it is rejected by the 'conventional' one.

The templates $T_1, \cdots, T_4$ of figure 3.2 define the edge subspace. The templates $T_4, \cdots, T_8$ of figure 3.2 define the line subspace. Template $T_9$ of figure 3.2 defines the average.

#### 3.2.1.2 Edge direction computation

Frei & Chen's method does not compute explicitly the edge direction. However, the edge direction $\alpha$ can be easily calculated using

$$\alpha = \arctan\left(\frac{W \cdot T_1}{W \cdot T_2}\right) . \tag{3.3}$$

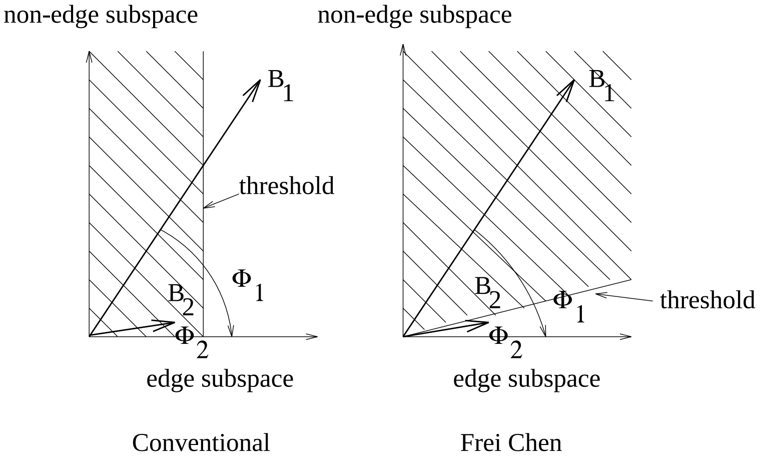



Figure 3.3: Frei-Chen boundary classification rule vs conventional.

In case $W \cdot T_1$ or $W \cdot T_2$ are zero, algorithm 3.1 is used, where $T_1$ and $T_2$ are as given in figure 3.2. Often the direction values $\alpha$ in $[0, 2\pi\rangle$[1] of $\mathbb{R}$ [2] are converted into [1,8] of $\mathbb{N}$ [3].

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

Algorithm 3.1: edge direction computation.

### 3.2.2 Lacroix's modifications of Frei & Chen's algorithm

Lacroix [60] defines a three module strategy to determine the edge pixels. In the first module she computes the edge magnitudes and directions. In the second module she thresholds them, defines and computes the LBE. In the third module she performs edge following based on the LBEs.

In this section we discuss the first module of Lacroix's algorithm. Her edge detection method is based on Frei-Chen's approach, but she first corrects a mistake in Frei-Chen's approach and then proposes a simplified basis equation.

Lacroix states that the denominator in equation (3.1) as proposed by Frei

[1] $[, \rangle$ define the boundaries of an interval, the former indicates inclusion of the left value, the latter indicates exclusion of the right value.

[2] $\mathbb{R}$ represents the real numbers.

[3] $\mathbb{N}$ indicates the natural numbers (0 included).

& Chen (i.e. $\sum_{j=1}^{n^2}(W \cdot T_j)^2$, is equal to the length of the pattern vector in the entire space if and only if the basis is normalized. The $\Theta$ (equation (3.1)) or $r$ (equation (3.2)) would remain unchanged if the elements of the basis had the same norm, as well as the elements of the non-edge basis. The $T_1, \cdots, T_e$ proposed by Frei & Chen have indeed the same norm but the remaining masks all have different norms.

In order to simplify equation (3.1), Lacroix proposes the use of the natural canonical basis made of 9 orthonormal masks and the masks $M_1, \ldots, M_4$, as given in figure 3.4, which generate a four dimensional edge space. However, the basis

$M_1$

| | | |
|---|---|---|
| -a | -1 | 0 |
| -1 | 0 | 1 |
| 0 | 1 | a |

$M_2$

| | | |
|---|---|---|
| -1 | -a | -1 |
| 0 | 0 | 0 |
| 1 | a | 1 |

$M_3$

| | | |
|---|---|---|
| 0 | -1 | -a |
| 1 | 0 | -1 |
| a | 1 | 0 |

$M_4$

| | | |
|---|---|---|
| 1 | 0 | -1 |
| a | 0 | -a |
| 1 | 0 | -1 |

$M_5 = -M_1$ $M_6 = -M_2$ $M_7 = -M_3$ $M_8 = -M_4$

Types of average: unweighted a = 1, weighted a = 2, isotropic a = $\sqrt{2}$

Figure 3.4: Some local operators for which eight gradient directions are available (after Lacroix [60]).

vectors $M_1$, $M_2$, $M_3$, $M_4$ are not orthogonal to each other. Therefore the basis is re-built into the orthogonal one as shown in figure 3.5. The canonical basis is

$B_1$

| | | |
|---|---|---|
| -1 | -a | -1 |
| 0 | 0 | 0 |
| 1 | a | 1 |

$B_2$

| | | |
|---|---|---|
| 1 | 0 | -1 |
| a | 0 | -a |
| 1 | 0 | -1 |

$B_3$

| | | |
|---|---|---|
| 0 | -1 | a |
| 1 | 0 | -1 |
| -a | 1 | 0 |

$B_4$

| | | |
|---|---|---|
| a | -1 | 0 |
| -1 | 0 | 1 |
| 0 | 1 | -a |

$B_5 = -B_1$ $B_6 = -B_2$ $B_7 = -B_3$ $B_8 = -B_4$

Types of average: unweighted a = 1, weighted a = 2, isotropic a = $\sqrt{2}$

Figure 3.5: Lacroix's final basis.

used in order to compute the norm of the pattern vector so that equation (3.1) becomes

$$r \propto \sqrt{\frac{\sum_{i=1}^{4}(W \cdot B_i)^2}{\left(w_{1,1}^2 + w_{2,1}^2 + \cdots + w_{3,3}^2\right)}} \quad . \tag{3.4}$$

The $B_i$ are not strictly normalized but have the same norm so that a 'proportional to' sign ($\propto$) is used.

#### 3.2.2.1 Park and Choi's remarks on Lacroix's algorithm

Park & Choi [77] noted that by setting 'a' in the templates $B_1, \ldots, B_4$ shown in figure 3.5 to $\sqrt{2}$, Lacroix's orthogonal basis is the same as the Frei-Chen edge subspace basis.

Furthermore they propose a fast computation of the ratio $r$ in equation (3.4) based on the normalized orthogonal basis as given in Gonzalez & Wintz [36]. Park & Choi rewrite equation (3.2) resulting in:

$$r = \frac{\frac{1}{2}\left((w_{11}-w_{33})^2+(w_{12}-w_{32})^2+(w_{13}-w_{31})^2+(w_{23}-w_{21})^2\right)}{(W \cdot W)}. \tag{3.5}$$

#### 3.2.2.2 Normalization of the Frei-Chen masks

Park [76] discusses the Frei-Chen edge masks in order to give a Fourier interpretation of the templates. He normalizes the nine orthogonal vectors $T_i$. The normalization factors are $2\sqrt{2}$ for $T_1, \cdots, T_4$; 2 for $T_5, T_6$; 6 for $T_7, T_8$ and 3 for $T_9$ and thus gets four new templates $T'_1, \cdots, T'_4$ as given in figure 3.6.

$T'_1$:

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

$T'_2$:

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

$T'_3$:

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

$T'_4$:

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

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

Figure 3.6: normalized masks.

So the thresholding equation as proposed by Frei & Chen, equation (3.2), becomes for this case

$$r = \frac{1}{(W \cdot W)} \sum_{i=1}^{4} (W \cdot T'_i)^2. \tag{3.6}$$

## 3.3 Improved magnitude calculation strategy

Given the windows 1 and 2 as shown in figure 3.7.
The ratio $r$ as defined in equations (3.2) (i.e. Frei & Chen), (3.5) (i.e. Park & Choi), and (3.6) (i.e. Park) is grey value dependent [93] as shown in table 3.1. Table 3.1 shows the ratio values for Frei-Chen, Park & Choi's fast implementation and Park's normalized templates for window 1 and window 2. From this table it follows that regions with the same absolute spectral difference have different ratios for all implemented algorithms.

This behaviour can cause problems while detecting edges and using a fixed threshold. Good detectable and well localized edges, but with a low relative difference between the surrounding regions having high spectral values, have a

| | | |
|---|---|---|
| 10 | 25 | 25 |
| 10 | 25 | 25 |
| 10 | 25 | 25 |

window 1

| | | |
|---|---|---|
| 200 | 215 | 215 |
| 200 | 215 | 215 |
| 200 | 215 | 215 |

window 2

Figure 3.7: two windows for ratio comparison.

| templates | window 1 | window 2 |
|---|---|---|
| Frei-Chen | $\frac{2700}{4050}$ | $\frac{2700}{397350}$ |
| Park-Choi | $\frac{337.5}{4050}$ | $\frac{337.5}{397350}$ |
| Park | $\frac{337.5}{4050}$ | $\frac{337.5}{397350}$ |

Table 3.1: different ratios.

low ratio. Edges having a high relative difference between the surrounding regions having low spectral values, have a higher ratio than in the first case. Due to global threshold setting the last edge might be detected, while the first one might not. However, such an edge should be detected too. Therefore it is desirable to have the ratio depending not on the relative difference between neighbouring regions, but on the absolute difference.

To avoid the above situation, a method is proposed here which overcomes this dependency in such a way that the ratio for two different windows (as shown in figure 3.7) will be the same, see algorithm 3.2.

Depending on the pixel values in the window $W$, equation (3.4) in combination with the templates $B_1, \ldots, B_4$ as given in figure 3.5, gives an edge magnitude ratio.

```
compute MinValue of window W;
IF (MinValue > 1)
THEN For all pixels in window W:
     PixelValue := PixelValue - MinValue;
ENDIF
```

Algorithm 3.2: subtraction of minimum value within the window $W$.

Now given two ratios $r_1$ and $r_2$ which are computed both by equation 3.4. Equation 3.4 results in for $r_1$:

$$r_1 \propto \sqrt{\frac{\sum_{i=1}^{4}(W \cdot B_i)^2}{\left(w_{1,1}^2 + w_{2,1}^2 + \cdots + w_{3,3}^2\right)}} \quad . \tag{3.7}$$

and for $r_2$:

$$r_2 \propto \sqrt{\frac{\sum_{i=1}^{4}(P \cdot B_i)^2}{\left(p_{1,1}^2 + p_{2,1}^2 + \cdots + p_{3,3}^2\right)}} \quad . \tag{3.8}$$

It is to be expected that for two image-windows $P$ and $W$ for which:

$$w_{i,j} = p_{i,j} + \min \tag{3.9}$$

the same edge magnitude (ratio $r$) for the pixel $w_{2,2}$ ($r_1$) and $p_{2,2}$ ($r_2$) will result, where $i, j = 1, \ldots, 3$, $\min > 0$ and $p_{i,j} > 0$.

On one hand it is shown that the numerator of equation (3.4) (i.e. adapted Frei & Chen) is independent of the spectral values if the absolute spectral difference between two adjacent regions is the same. On the other hand it is shown that the denominator is dependent on the spectral values, and that its value is higher as the spectral values of the pixels are high. This brings us to the following theorem:

<u>THEOREM</u> 3.1 *The ratio $r$ as defined by equation (3.7) is dependent on* min *and the larger* min *the smaller $r$.*

First two lemmas are given, which are used to prove the theorem.

<u>LEMMA</u> 3.1 *The numerator of equation (3.7) (i.e. $\sum_{i=1}^{4}(W \cdot B_i)^2$) is equal to $\sum_{i=1}^{4}(P \cdot B_i)^2$ and thus independent on min.*

<u>proof:</u>
Substitution of equation (3.9) for the templates of figure 3.5 in equation (3.7) gives for $WB_i$, $i = 1, \cdots, 4$

$$\begin{aligned} W \cdot B_1 = & -w_{1,1} - a \cdot w_{1,2} - w_{1,3} + w_{3,1} + a \cdot w_{3,2} + w_{3,3} = \\ & -(p_{1,1} + \min) - \mathrm{a} \cdot (\mathrm{p}_{1,2} + \min) - (\mathrm{p}_{1,3} + \min) + \\ & (p_{3,1} + \min) + \mathrm{a} \cdot (\mathrm{p}_{3,2} + \min) + (\mathrm{p}_{3,3} + \min) = \mathrm{P} \cdot \mathrm{B}_1 \end{aligned} \tag{3.10}$$

The proofs of $W \cdot B_2$, $W \cdot B_3$, $W \cdot B_4$ are similar.

As can been seen all $W \cdot B_i$'s $= P \cdot B_i$'s are independent of min and thus $\sum_{i=1}^{4}(W \cdot B_i)^2$ and $\sum_{i=1}^{4}(P \cdot B_i)^2$ are independent of min. $\square$

<u>LEMMA</u> 3.2 *The denominator of equation (3.7) (i.e.* $\left(w_{1,1}^2+\cdots+w_{3,3}^2\right)$*) is unequal to the denominator of equation (3.8) (i.e.* $\left(p_{1,1}^2+\cdots+p_{3,3}^2\right)$*) and thus dependent on min.*

<u>proof:</u>
The denominator results for $r_1$ with substitution of equation 3.9 in:

$$\left(w_{1,1}^2+w_{2,1}^2+\cdots+w_{3,3}^2\right)$$

$$\left((p_{1,1}+\min)^2+\cdots+(p_{3,3}+\min)^2\right)=$$

$$9\cdot\min^2+2\cdot\min\cdot(p_{1,1}+\cdots+p_{3,3})+(p_{1,1}^2+\cdots+p_{3,3}^2). \tag{3.11}$$

To be independent of min the above equation should be equal to:

$$\left(p_{1,1}^2+\cdots+p_{3,3}^2\right) \tag{3.12}$$

resulting in:

$$9\cdot\min+2(p_{1,1}+\cdots+p_{3,3})=0 \tag{3.13}$$

As can be seen immediate equation (3.13) has no solution for $\min>0$, $p_{i,j}>0$. Hence the denominator of $r_1$ is not equal to the denominator of $r_2$. The above implies that for $\min>0$ both denominators are dependent on min. □

Using lemma 3.1 and 3.2 the proof of theorem 3.1 is easy:

<u>proof:</u>
The dependency of $r$ on min follows directly from lemma 3.1 and lemma 3.2 The larger min, the higher the denominator and thus the smaller $r$. □

By applying algorithm 3.2 it is possible to eliminate the dependency on min in the calculation of the ratio $r$. Thus the ratios found for pixel values having intensities in the upper part of the spectral interval, i.e. [0,255] for byte values, will be the same as the ones found in the lower part.

### 3.3.1 Thresholding of the edge magnitudes

Having the edge magnitude values and their directions, the edge fragments must be determined. As all the edge-magnitudes are in $\mathbb{R}$ these must be converted to $\mathbb{N}$ (within [0,255] for byte values) in order to make automatic computation of a threshold, using the histogram of the edge magnitude values, possible. Thus pixels depending on the computed threshold are being classified as edge candidates or as non edge candidates. From the edge candidates, pixels have to be selected in order to construct one pixel wide edge fragments. How this is done is explained in section 3.5.1.

The conversion requires the minimum and maximum edge magnitudes and uses the difference to create 256 equidistant intervals into which the edge magnitude values are put depending upon their original edge magnitude values. It has to be remarked that the conversion of values from the continuous domain into the discrete domain, causes loss of information. The converted image can be thresholded in order to reject pixels not belonging to the edge pixel class.

An automatic global[4] threshold computation is included following Kapur *et al.* [53]. Their algorithm derives two probability distributions (in our case edge, non-edge pixels) from the original grey-level distribution. The maximum of the sum of the entropy of the edge pixels and the entropy of the non-edge pixels is taken as the threshold; see appendix B for a description of the algorithm. Of course other thresholding techniques can be used, e.g. [14, 21, 49, 56, 88, 91, 122].

Figure 3.8 shows the entropy histogram. For histogram calculation the computed edge magnitude values are used. From this histogram, the spectral value

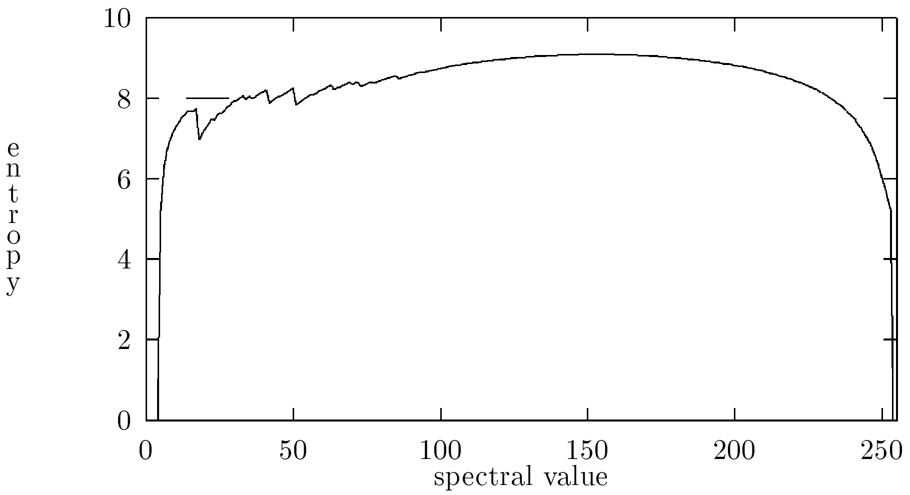



Figure 3.8: the entropy histogram for the edge magnitudes of image 3.10.

having the highest entropy is selected for the threshold (for image 3.10 it is 152, see a later section).

## 3.4 Edge detection in multi-channel imagery

None of the algorithms described so far defines what to do in the case of multi-channel imagery. For remotely sensed imagery this causes problems: it is necessary to decide either to use only one channel for segmentation or to use all channels separately and then selecting the 'best' one, subject to human interpretation.

[4]A global thresholding technique is one that thresholds the entire image with a single threshold value, whereas a local threshold technique is one that partitions a given image into sub-images and determines a threshold for each of these sub-images. Global techniques can be divided into point-dependent, i.e. the threshold is determined solely from the grey-value of the pixels, and region-dependent, i.e. the threshold is determined from the local property in the neighbourhood of the pixel [91]. The method as proposed by Kapur *et al.* is a global point-dependent method.

In our implementation, for every channel the edge direction and edge magnitude is computed, resulting in an edge magnitude vector and an edge direction vector. It is clear that for a given pixel the different channels might result in different magnitudes and different directions. This means that in some way a unique direction computation has to take place. The solution to this problem which has been used in this study, is to simply select the direction of the channel having the strongest magnitude value. More complex schemes could be envisaged based on examining the directions indicated in the different channels and taking a consensus view.

## 3.5 One pixel wide contour creation

The pixels classified as edge candidate pixels do not provide a one pixel wide contour. In order to select the 'best' candidates for contour creation, Lacroix [60] first computes the LBE for every pixel. Using these values Lacroix applies a non-maximum deletion algorithm to do the edge following and to create the one pixel wide contours.

### 3.5.1 Lacroix's likelihood of being an edge

The idea behind the LBE computation is to postpone the deletion of some edge points until contextual information becomes available. Pixels assigned a LBE=0 are deleted immediately, pixels with LBE=1 are definitely considered as edge candidates, and pixels with $0 < \text{LBE} < 1$ have to pass the third module to know their status.

The LBE is computed as given hereafter. Each pixel has two counters $m$ and $v$; $m$ counts how many times a point is a local maximum; $v$ counts how many times a pixel is visited. While scanning the image, on each pixel a $3 \times 1$ window is placed such that the middle of the window coincides with the pixel and the other two pixels lie along the current gradient direction. All pixels in the window are visited during this process and their $v$ counters are incremented. But only the pixel(s) having the greatest gradient magnitude increment its/their $m$ counters. After the scan a LBE $= m/v$ is assigned to every pixel.

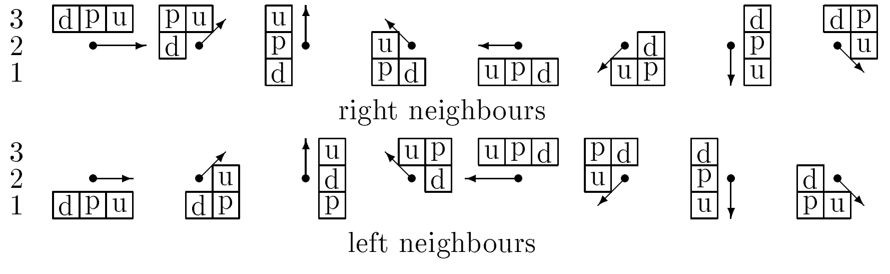



Figure 3.9: the next pixel of the contour.

Now a pixel with a local maximum will have LBE 1, while a pixel that is sometimes but not always a local maximum, depending on the position of the window, will have $0 < \text{LBE} < 1$, and a pixel which is never a local maximum will have LBE 0.

### 3.5.2 Lacroix's non-maximum deletion algorithm

Having all the LBE values for every pixel, Lacroix performs contour following based on Kunt's algorithm [59], a sequential process which uses the fact that edges in the real world are mostly continuous and perpendicular to the edge gradient.

```
mark current pixel;
IF (p has LBE = max)
THEN p is the next point of the contour
     IF LBE of u = 1 AND/OR LBE of d = 1
     THEN modify their LBE to 1/2
          {no new contour may start here}
     ENDIF
ELSE IF (LBE of u = max) XOR (LBE of d = max)
     THEN the one with LBE=max is the next point of the contour
     ENDIF
ELSE IF (LBE of u = max) AND (LBE of d = max)
     {current point is a junction}
     THEN choose the one with the closest gradient norm;
          set the LBE of the other to 1
     ENDIF
ENDIF
```

Algorithm 3.3: LBE following.

Lacroix's algorithm marks pixels when they are visited, so contours do not overlap; the contour following algorithm is given algorithm 3.3. In this algorithm let $p$, $u$, and $d$ be, respectively the pixel up, perpendicular or down depending on the direction of the current pixel and, let $max$ be the maximum of their LBE.

Scan the image until an un-marked pixel with LBE=1 is found. Start a left and right contour following from here. Depending on the current pixel's edge direction, match the window selected from figure 3.9 onto it. In this figure the endpoint of the arrow is the current pixel.

## 3.6 Results

Image 3.1 shows the SPOT test image from the Flevoland polder, The Netherlands, of July 4, 1991. The image size is $500 \times 500$ pixels by three channels. Image 3.2 shows channel 1, image 3.3 channel 2, and image 3.4 channel 3 of image 3.1. For clarity of display the images have been histogram equalized.

Image 3.1 shows presence of thin clouds. The separate fields are well recognizable. Image 3.2 and image 3.3 also show cloud presence. In image 3.4 however the clouds (visually) disappear. Also image 3.4 is brighter than the other two. Image 3.4 shows infield variance which is not visible in images 3.2 and 3.3. This is due to the changing chlorophyll contents of crops which may be caused by different soil moisture conditions or even crop disease.

| templates | with subtraction | without subtraction |
|---|---|---|
| image 3.1 | 152 | 76 |
| image 3.2 | 147 | 27 |
| image 3.3 | 150 | 65 |
| image 3.4 | 154 | 28 |

Table 3.2: thresholds computed.

Table 3.2 shows the computed threshold. The thresholds computed without subtraction of the minimum of the pixels of window $W$ before edge magnitude computation, are lower than the thresholds computed with the subtraction of the minimum.

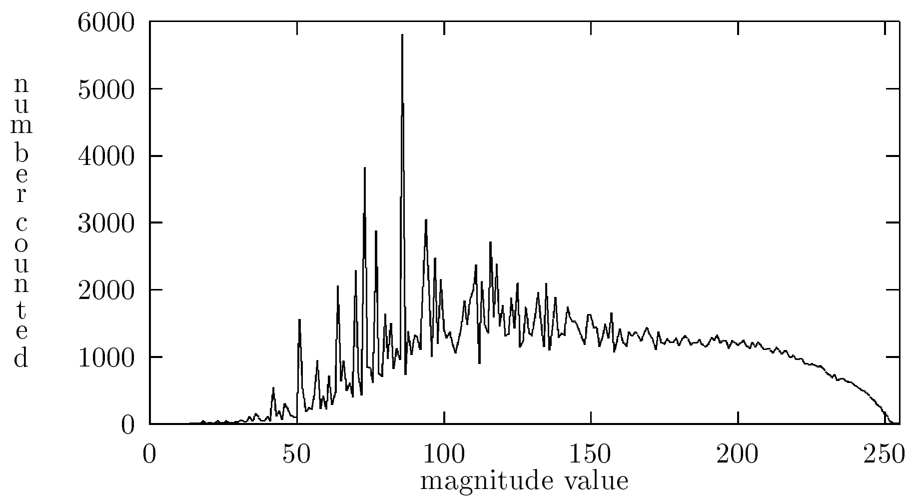



Figure 3.10: converted magnitudes for image 3.1 with subtraction of the minimum for window $W$

The edge-magnitude histogram for image 3.1 is given in figure 3.10. Note the peak for value 86.

The edge-magnitude histogram for image 3.1 without subtraction of the minimum is given in figure 3.11. Figures 3.10 and 3.11 show that the magnitudes in the first histogram are more distributed than in the second one. In figure 3.11 it can be seen that the edge magnitude values are concentrated around the value 5. Due to this concentration, thresholding might be complicated. The peaks in figure 3.10 can be caused by noise within the image. As indicated previously the subtraction of the minimum pixel value of the pixels of the window $W$, before computation of the edge magnitude, can emphasize the influence of noise pixels on the edge magnitude computation.

The reason that figure 3.10 and figure 3.11 are so different might be caused by the fact that many pixels have high intensity values, resulting in a relative low edge magnitude ratio. If the minimum pixel value is subtracted from the pixels in the window $W$, the intensity values become 'lower' and thus the edge magnitude ratio will be 'higher'. The better spread of the ratios is caused by the fact that the absolute difference between 'two edges' in the window $W$, plays a role. Low absolute differences result in a low edge magnitude ratio, while high absolute differences result in a high ratio.

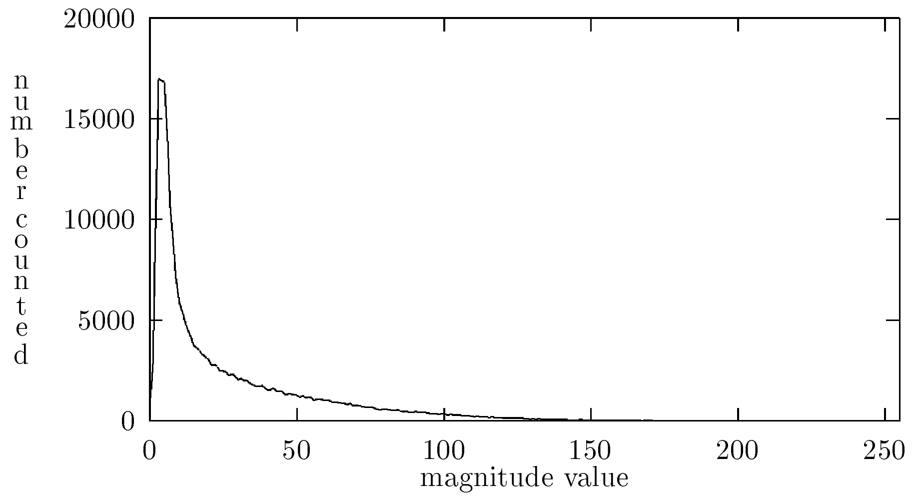



Figure 3.11: converted magnitudes for image 3.1 without normalization

Image 3.5 shows the resulting image after edge detection and following based on image 3.1. Images 3.6-3.8 are created by applying the edge detection to images 3.2-3.4 respectively. If we look at image 3.6 we see that the cloud presence is detected by the edge detector. For images 3.7 and 3.8 these are not so clearly visible. It can be seen that image 3.5 is more or less a composition of images 3.6, 3.7 and 3.8. Image 3.8 shows the most detail of the last mentioned, but it is also the noisiest one. Image 3.9 is the zoomed upper right part of image 3.5 where every edge pixel is labeled according to the channel which gave it the highest magnitude. Pixels having their origin from channel one are labeled red, pixels having their origin from channel two are labeled green, pixels having their origin from channel three are labeled blue. Of course combinations are possible as can be seen in the image.

Images 3.10 and 3.11 show the discrete edge magnitudes with and without subtraction of the minimum. The brighter pixels are pixels with a higher edge magnitude. Darker pixels are pixels where edge activity is low. It is clear that in image 3.10 far more pixels having a strong edge activity, are detected. From table 3.2 it follows that the pixel values for image 3.10 after thresholding are in the set $\{0, 152, 153, \ldots, 255\}$.

Image 3.12 shows image 3.11 but after thresholding. The computed threshold value for image 3.11 is 76. All pixels having an edge value below 76 are set to zero. Applying LBE computation and following on image 3.12 gives image 3.13.

If a comparison between image 3.5 and image 3.13 is made it is clear that image 3.5 shows more edges detected. This proves the effectiveness of subtraction of the minimum of the pixels of the input window $W$ before edge computation.

## 3.7 Concluding remarks

In this chapter it has been shown that edge detection can be made more sensitive to weak but well localized edges by subtracting the minimum pixel value of the input window $W$. A disadvantage however is that it is also more susceptible to noise. Due to noise in one channel, for a certain pixel, the intensity value of neighbouring pixels might differ significantly, resulting in an edge magnitude value higher than that of the other channels. This may cause a weak edge to be overruled by a noisy edge.

A weaker point is selection of the edge direction. In our implementation the final edge-direction is depending upon the strongest magnitude found. For example if a pixel consists of three channels and all channels give a comparable magnitude value but three different directions then the choice of the strongest might seem random.

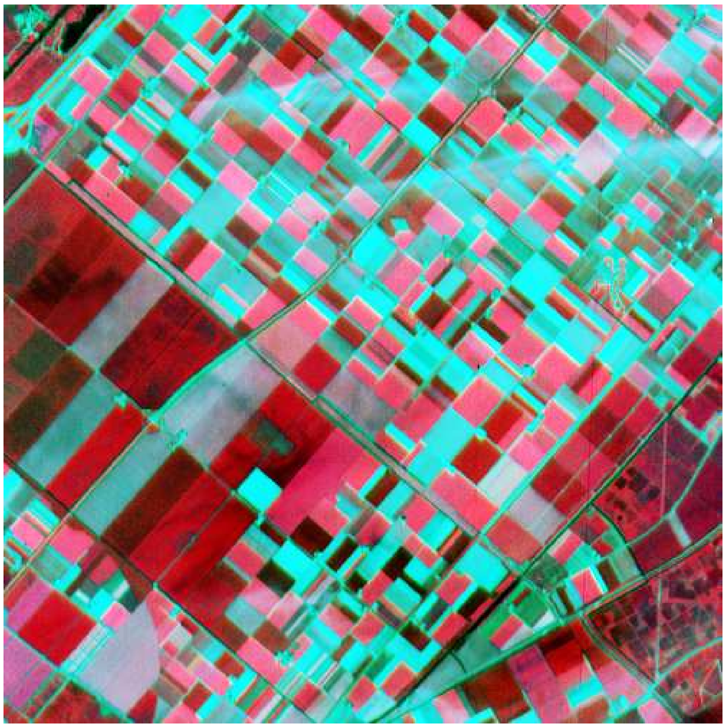


Image 3.1: SPOT $500 \times 500$ pixels, three band Flevopolder 91, original data ©SPOT-Image, channel 3,2,1 : R,G,B.

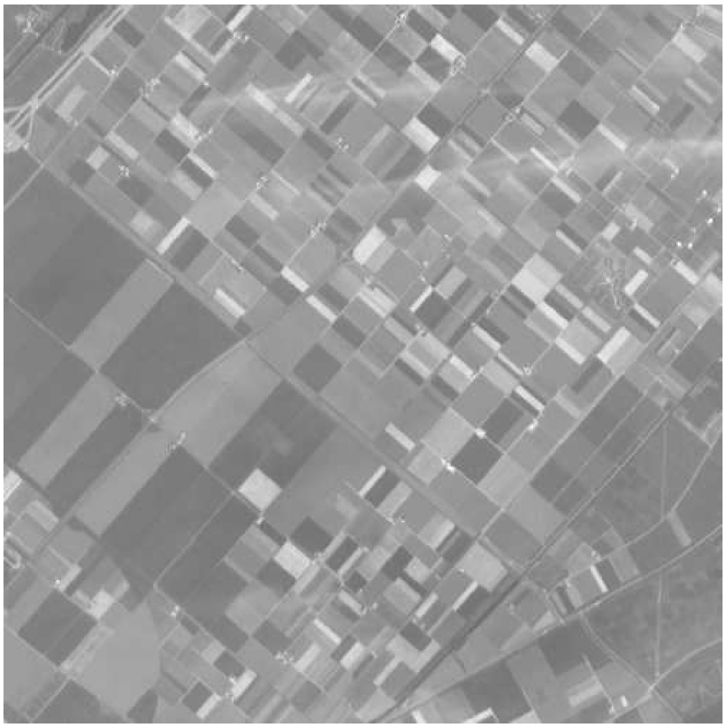


Image 3.2: channel one of image 3.1.

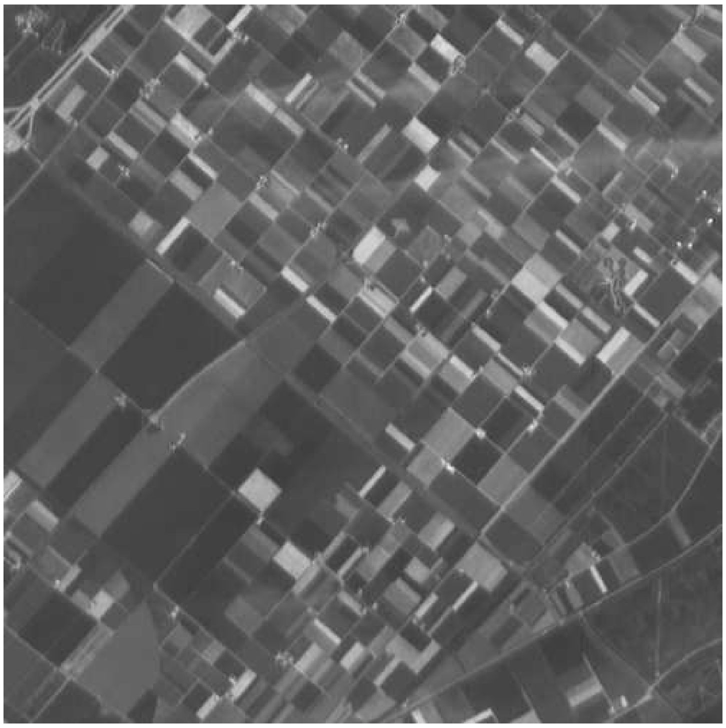


Image 3.3: channel two of image 3.1.

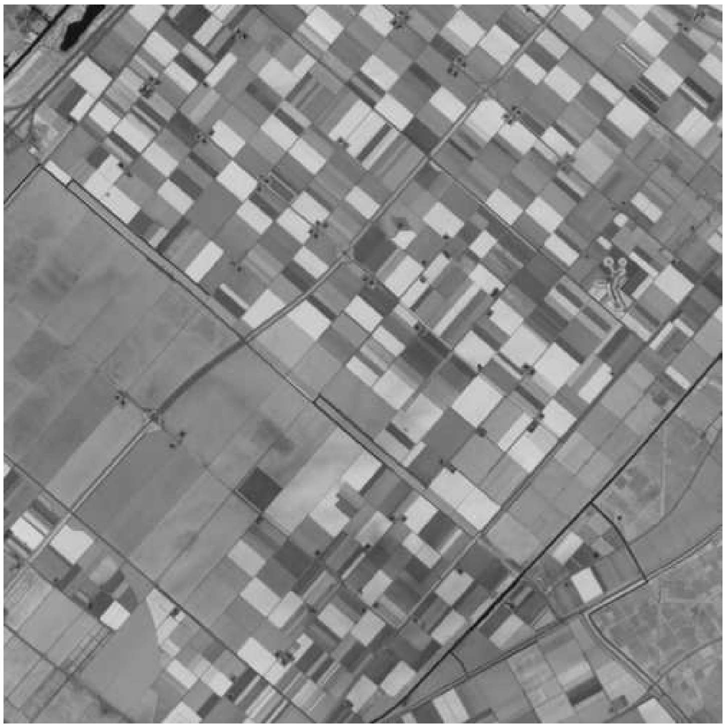


Image 3.4: channel three of image 3.1.

Image 3.5: edges detected in image 3.1.

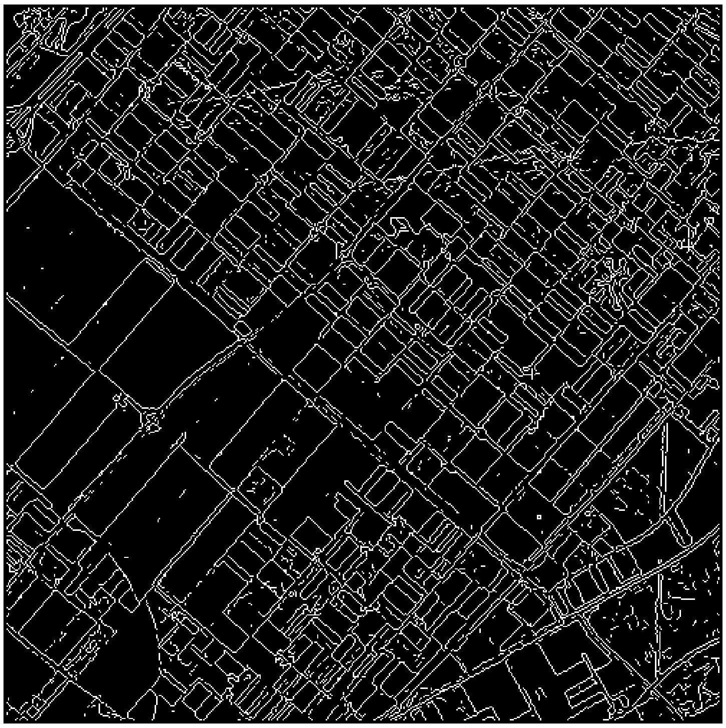


Image 3.6: edges detected in image 3.2.

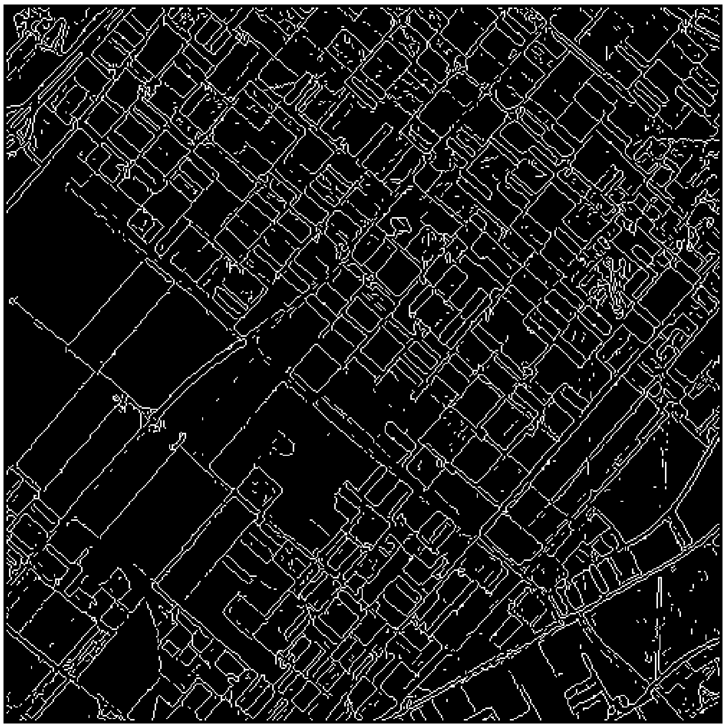


Image 3.7: edges detected in image 3.3.

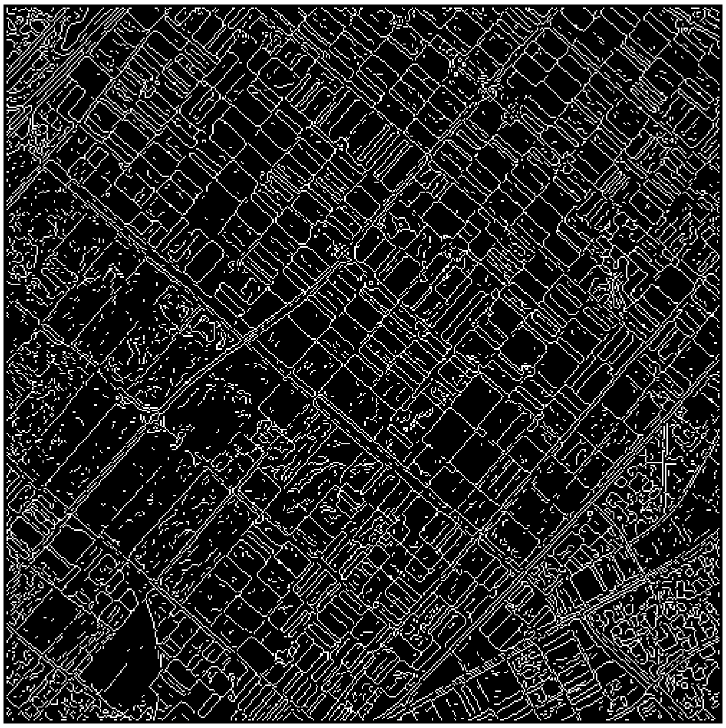


Image 3.8: edges detected in image 3.4.

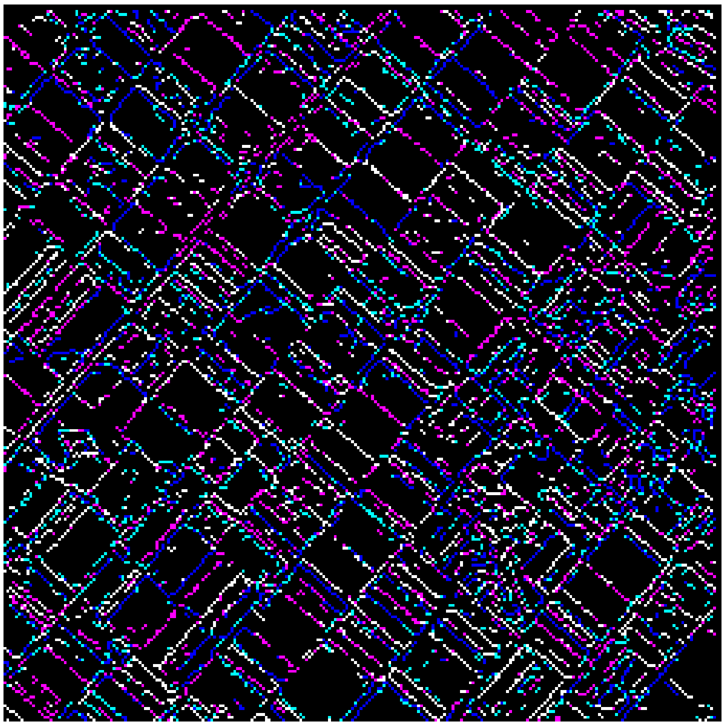


Image 3.9: zoomed upper right part of image 3.5. edges colored after channel origin, channel 1 green; channel 2 red; channel 3 blue; Flevopolder 91.

Image 3.10: converted edge magnitude image within interval [0,255] of image 3.1.

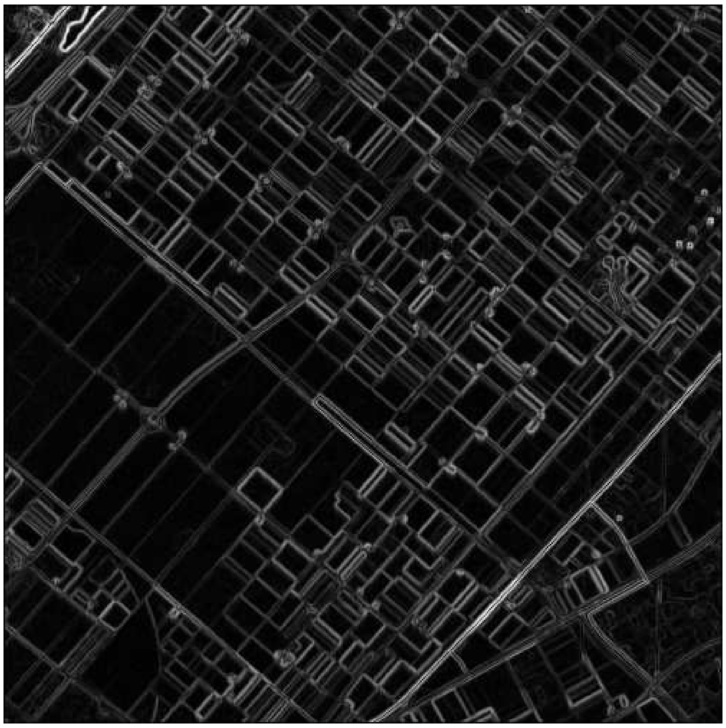


Image 3.11: discretised edge magnitudes of image 3.1 without normalization.

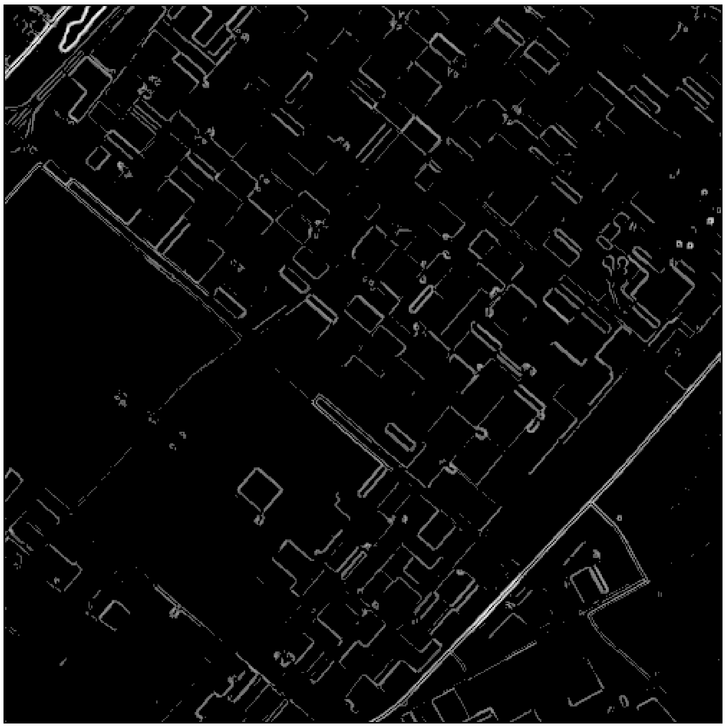


Image 3.12: image 3.11 thresholded with threshold 76.

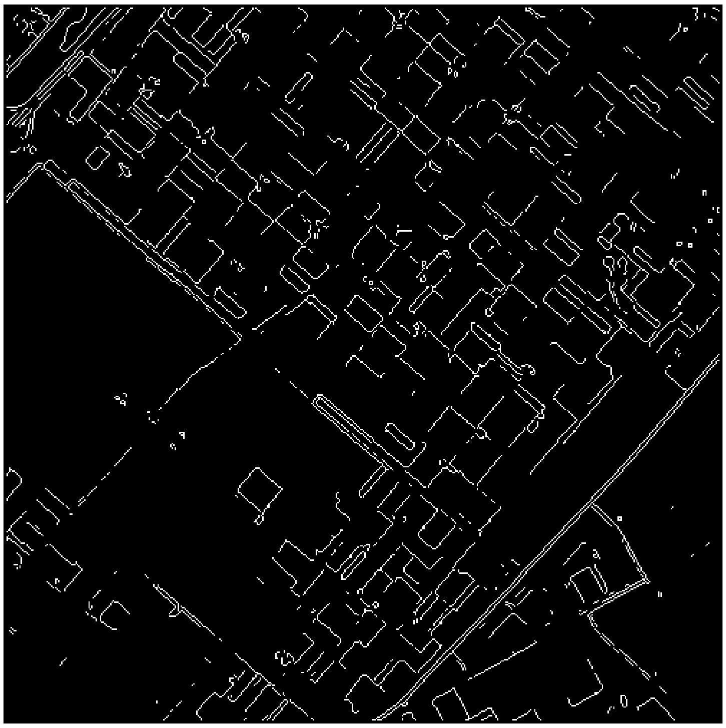


Image 3.13: final edges of image 3.1 without normalization.

# Chapter 4

## An Edge linking Method

### 4.1 Introduction

At the end of the edge detection procedure, every pixel is classified either as an edge pixel or as a non-edge pixel. Edge pixels which are 8-connected together form edge fragments. As stated in the introduction these fragments are normally not connected into closed structures (in the following referred to as closed polygons). In order to connect these fragments, pixels can be added which conform to simple rules.

In this chapter a newly developed method is described which connects edge fragments by adding pixels in such a way that closed polygons are formed. Fragments which cannot be joined into closed polygons will be deleted.

Graph theory is used to construct data structures which represent the connections between edge pixels. The edge following procedure looks for 8-connected un-marked edge pixels. These are then marked, and added as "sons" of the current pixel. This way "trees" are constructed together composing a "forest". By using the forest information, edge pixels are added in a pre-defined way in the image, closing gaps and therefore creating closed polygons.

The main goal is to connect most of the separate edge fragments into new fragments which together form closed polygons. These polygons divide the image into separate areas. Within a separate area some non-connected edge fragments can still be present.

In Perkins [82] an edge linking method is described which uses an expansion-contraction technique. First edge points are expanded, i.e. grown within their neighbourhood, to close gaps and then contracted after the separate uniform regions have been identified. The process is performed iteratively from small to large expansions.

### 4.2 Data structure definitions.

Definitions given here are taken from Jeurissen [48].

[GRAPH] A graph $G$ is a pair $\langle N, V \rangle$ where $N$ is a not empty set of points (nodes, vertices) and $V$ a set of not ordered pairs of elements of $N$. So if $(a, b) \in V$ then $(a, b) = (b, a)$. The elements of $V$ are the edges, the elements of $N$ are vertices, nodes. The *degree* of a point (node) $p$ in a graph is the number of neighbours of $p$, i.e. $(p, p') \in V$.

[PATH] A path in a graph $G$ is a tuple $x_0, x_1, \ldots, x_k$ with:
$k \geq 0$, $x_i \in N$ for $i = 0, 1, \ldots, k$ and
$(x_i, x_{i+1}) \in V$ for $i = 0, 1, \ldots, k-1$ and $x_i \neq x_j$ for $i \neq j$.
The *length* is $k$.

[CYCLE] A cycle in a graph $G$ is a path $x_0, \ldots, x_k$ with:
$x_0 = x_k, x_i \neq x_j$ for $0 < j - i < k$ and $k > 2$, $x_i \in N$.

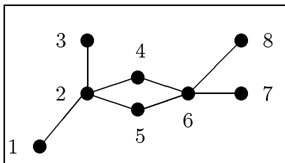
[TREE] A tree is a *connected graph* without cycles. Connected means there is a path for every pair of nodes $(a, b)$ from $a$ to $b$ (and vice versa).

[ROOT] A point of a tree having degree one.

[FOREST] A forest is a set of trees.

[DEGREE OF $G$] The degree of graph G is the total number of vertices of G.

[LEAF] A leaf is a node of degree 1 (thus the root can be a leaf and a leaf can be the root of the tree).




$G = \langle N, V \rangle$

$N = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$V = \{(1,2), (2,3), (2,4), (2,5), (4,6), (5,6), (6,7), (6,8)\}$

Figure 4.1: a graph example

The degree of node 2 of the graph $G$ in figure 4.1 is 4, i.e. node 2 has 4 neighbours, nodes 1,3,4,5. The degree of $G$ is 8. A cycle is 2,4,6,5,2. A path is 1,2,5,6,7 with length 4. To create a tree out of $G$ delete one of the edges (2,5), (2,4), (4,6), (5,6).

Here the input image $I$ is formulated in terms of a graph $G$ as a tuple $\langle N, V \rangle$, where:

- ▷ $N$ is the set of nodes of the graph. $N$ is defined so that every node $n$ represents a pixel $p$ of $I$ with $p$ being an edge pixel.
- ▷ $V$ is the set of edges $v = (p_1, p_2)$ of the graph, where $p_1$ and $p_2$ are 8-connected pixels of $I$. The edges $v$ of $V$ express the adjacency of two pixels within the image $I$.

Every node $n$ added to $N$ will have an unique index; the row and column positions of the corresponding pixel $p$ in the image $I$, denotated by the function $ind(n) = pos(p) = (x, y)$. Furthermore the function which determines $p$'s intensity value (label) as $lab(p) = l, l \in L$ is defined, where $L$ is a set of labels. Every pixel $p$ of $I$ is defined to be 8-connected, i.e. the pixel $p$ in figure 4.2 is connected with the pixels $1, \ldots, 8$, expressed by the arrows. Initially, pixels have an intensity value of 0 or 255 (i.e. edge, non-edge). While scanning the image pixels are marked for which values other than 0 or 255 are used.

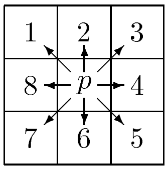



Figure 4.2: 8 connectivity.

The distance $d(p, q)$ between the centres of two pixels $p$ and $q$ is defined as:

$$d(p, q) = d(pos(p), pos(q)) = d((p_x, p_y), (q_x, q_y)) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{4.1}$$

where $p_x, p_y$ are the column, row index respectively for the pixel $p$ in the raster image $I$. This results in a distance for the pixel $p$ and the even numbered pixels of 1 in figure 4.2 and $\sqrt{2}$ for $p$ and the odd numbered pixels.

## 4.3 Forest construction

The basic idea behind the edge-following is a combination of depth-first and breadth-first search. Given a pixel $p$, every time a new un-marked pixel $q$ is found, with $d(p, q) \leq \sqrt{2}$, it is remembered, as in a breadth-first search, but the searching continues in a depth-first way. If more than one un-marked neighbour $q$ is found, then first the pixels having $d(p, q) = 1$ are added as sons of $p$. In a later stage, remaining neighbours with a distance $\sqrt{2}$ are added as sons of $p$. If, of course, no neighbours with distance 1 are present then immediate neighbours with distance $\sqrt{2}$ are added as sons of $p$.

The algorithm 4.1 constructs trees, together forming a forest, so that a projection from the image to the forest is defined. Algorithm 4.1 is nothing more then a projection of the input image on the forest $G$

$$proj : I \times I \rightarrow G$$

such that $\forall_{G_i \in G}$, $p, q \in I, lab(p) = E, lab(q) = E$, $N_i \cup r$, where $R$ is the root of the tree:

$$\begin{aligned}
proj(\emptyset, \emptyset) &= r, \;\; ind(r) = pos(\emptyset) = (-1, -1) \;\text{ and} \\
proj(p, p) &= n, \;\; n \in N_i, \;\; \forall_{p \in I} : ind(n) = pos(p) = (x, y) \text{ and} \\
proj(p, q) &= v, \left\{ \begin{array}{l} \left\{ \begin{array}{l} v \in V_i, \;\; proj(p,p) = n, \;\; proj(q,q) = m, \\ n, m \in N_i \;\text{ and }\; d(p,q) \leq \sqrt{2} \end{array} \right\} \\ \text{or} \\ \left\{ \begin{array}{l} v \notin V_i \;\; proj(p,p) = n, proj(q,q) = m, \\ d(p,q) \leq \sqrt{2}, \\ \exists_{n_1, \ldots, n_k \in N} : r, n_i, \ldots, n_k, n \;\; \wedge \\ \exists_{m_1, \ldots, m_l \in N} : r, m_i, \ldots, m_l, m \end{array} \right\} \end{array} \right\}
\end{aligned} \tag{4.2}$$

In words:
$proj(\emptyset, \emptyset) = r$, means the creation of a root node initialized with the $x, y$ values

$-1, -1$.
$proj(p,p) = n$, means that for every pixel $p$ a node $n \in N$ is created, with as index the pixel positions within the image.
$proj(p,q) = v$, expresses the connectivity of the graph $G_i$. The corresponding nodes $n, m$ of $p, q$ respectively form an edge within $G_i$, or there exists a path from the root $r$ to the node $n$, and to node $m$, where $p$ and $q$ are 8-connected.

For simplicity reasons $proj(p,p)$ is written as $proj(p)$.

```
WHILE (∃p ∈ I :  lab(p) = E) DO
  1. G_i=⟨N_i, V_i⟩, N_i:=∅ , V_i:=∅ .
  2. proj( ∅ )=r , proj(p)=n ,
     N_i:=N_i+{n, r}, V_i:=V_i+{(r, n)},
  3. lab(p):=M .
  4. build_tree(p, n) =
       (a) B(p)={q | d(p, q)=1 and lab(q)=E}.
       (b) ∀q ∈ B(p) :  proj(q)=m , N_i:=N_i+{m}, V_i:=V_i+{(n, m)},
           lab(q):=M .
       (c) ∀q ∈ B(p) :  build_tree(q, n)
       (d) B(p)={q | d(p, q) ≤ √2 and lab(q)=E}.
       (e) ∀q ∈ B(p) :  proj(q)=m , N_i:=N_i+{m}, V_i:=V_i+{(n, m)},
           lab(q):=M .
       (f) ∀q ∈ B(p) :  build_tree(q, n)
ENDWHILE
output G = {G_0, ..., G_NT}.
```

Algorithm 4.1: tree construction.

The backtrack algorithm for forest construction is given in algorithm 4.1. Given pixel $p$, with $(proj(p) = n)$, $p \in I, n \in N_i$, first it is examined to determine if $p$ has 4-connected neighbours labelled 'edge'. If present these are marked and added as sons of $n$. The construction continues for one of the 4-connected neighbours. If moreover, $p$ has also 8-connected neighbours these are not added as sons of $n$ (and thus not marked). Because of the backtrack principle of the algorithm their situation will be evaluated when no un-marked 4-connected pixels are present. When tracking back trough the tree constructed, for every marked pixel the 8-connected pixels are checked. If they are not marked these are added as sons and tree construction continues from here (for the 8-connected pixels). If however, $p$ has no 4-connected neighbours but only 8-connected ones, these will be added as sons of $n$.

LEMMA 4.1 *Algorithm 4.1 creates trees $G_i$'s in such a way that there is only one path from the root $r$ to a node $n$, $(proj(p) = n)$.*

proof:
Assume there are two (or more) paths $P_1, P_2$ from $r$ to $n$. When $P_1$ is created and $p$ is visited, $p$ is marked, and therefore as defined by the algorithm, cannot

be marked a second time when $P_2$ is being created. When an un-marked 8-connected neighbour $q$ ($proj(q) = m$) of $p$ is visited $q$ is checked on being marked and therefore $m$ can never be added as a son of $n$ and thus within $G_i$ there cannot be two paths from $r$ to $n$. □

LEMMA 4.2 *Given two pixels $p$ and $q \in I$, $proj(p) = n$, $proj(q) = m$. If $p$ and $q$ are 8-connected there is a path from $n$ to $m$ in the corresponding tree $G_i$.*

proof:

1. if$(n, m) \in V_i$ then there is a path $n, m$,
   or

2. $(n, m) \notin V_i$
   Because of the connectivity of a tree there are paths:
   $r, n_1, \ldots, n_j, \ldots, n_{k1}, n$ with $k1 \geq 1$ and
   $r, n_1, \ldots, n_j, m_1, \ldots, m_{k2}, m$ with $k2 \geq 1$
   so there is a path $n, n_k, n_{k1-1}, \ldots, n_{j+1}, n_j, m_1, \ldots, m_{k2}, m$.

□

LEMMA 4.3 *If situation 2 of lemma 4.2 occurs then there is a closed polygon in $I$ for that path.*

proof: Because of the paths $r, n_1, \ldots, n$ and $r, n_1, \ldots, m$ there are chains in $I$ from $s$, ($proj(s) = n_1$) to $p$ and $q$. Because $p$ and $q$ are 8-connected there is a chain from $s$ to $p$, $p$ to $q$ and, $q$ to $s$. □

LEMMA 4.4 *Every node can have at maximum three sons.*

proof:
Given figure 4.2 and $p$ being the current pixel. Two cases have to be differentiated, i) the father of $p$ is an even labelled pixel (e.g. 2) or ii) the father of $p$ is an odd labelled pixel (e.g. 1).

case i):
Pixel 2 is taken as the father of $p$. For pixels 4,6,8 being the father of $p$ the proof is similar.
Pixels 1 and 3 cannot be neighbours of $p$. If one of them would be an edge pixel, it would have been added as a son of 2 (as defined by the algorithm, 4-connected pixels are added first). If 8 is a son then 7 cannot be a son of $p$. If 6 is a son then 5 and 7 cannot be sons of $p$. If 4 is a son then 5 cannot be a son of $p$. If 7 is a son of $p$ then 6 and 8 cannot be sons of $p$ (i.e. they cannot be edge pixels). If 5 is a son of $p$ then 4 and 6 cannot be sons of $p$ (i.e. they cannot be edge pixels). By writing out all the combinations now it can be seen that $p$ can have at maximum three sons (pixels 4,6, and 8)!

case ii):
Pixel 1 is taken as the father of $p$. For pixels 3,5,7 being the father of $p$ the proof is similar.
Pixels 2 and 8 cannot be neighbours of $p$. If one of them would be an edge pixel, it would have been added as a son of 1 (as defined by the algorithm, 4-connected pixels are added first). If 3 is a son then 4 cannot be a son of $p$ (i.e. they cannot

be edge pixels). If 5 is a son then 4 and 6 cannot be sons of $p$ (i.e. they cannot be edge pixels). If 7 is a son then 6 cannot be a son of $p$ (i.e. they cannot be edge pixels). If 4 is a son then 3 and 5 cannot be sons of $p$. If 6 is a son then 5 and 7 cannot be sons of $p$. By writing out all the combinations it can be seen that $p$ can have at maximum three sons (pixels 3,5, and 7)! $\square$

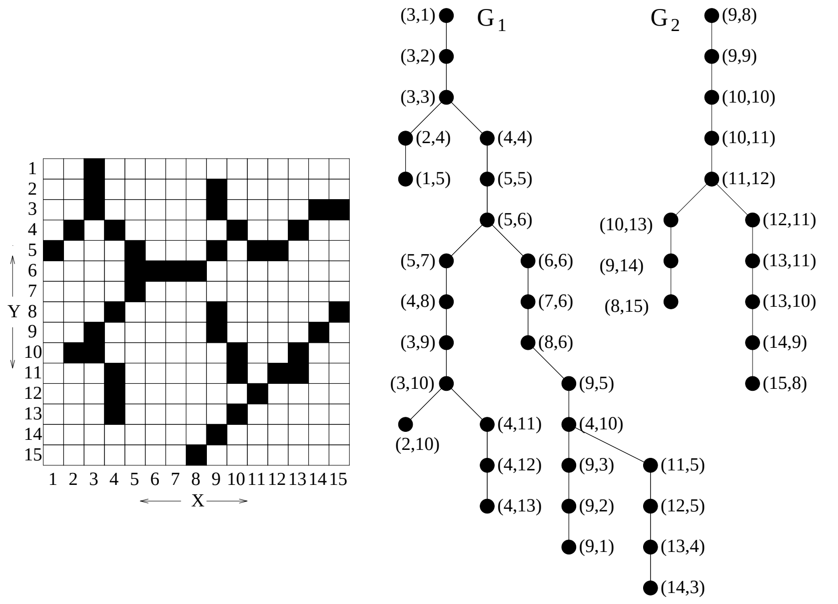



Figure 4.3: edge fragment to graph conversion.

An example of the image to forest conversion is given in figure 4.3. Pixel (5,5) is an example that first 4-connected nodes are added as sons. As can be seen in the corresponding graph only (5,6) is its son, and not also (6,6). An example of a pixel having 4-connected as well as 8-connected pixels as sons is pixel (3,10). When the 'following' algorithm has as current pixel (3,10) then first the 4-connected pixels are checked. This results in pixel (2,10) as the only son and the next pixel for contour following. As pixel (2,10) has no un-marked pixels within its vicinity, the following procedure stops here and the backtracking starts. The first pixel on the way back is pixel (3,10). Now the 8-connected pixels are checked, and pixel (4,11) is found and added as a son of (3,10).

What happens with structures in the image which are already closed while tree construction takes place? The algorithm is developed so that every pixel of an edge fragment appears once and only once as a node in the graph. This can be easily seen. If an 8-connected node is selected as the next point of a tree it is marked, and therefore can never be selected for addition to another node. Furthermore even if the current un-marked pixel has a marked neighbour, which has been marked some iterations before, no vertex is added to $V$, because only un-marked neighbours are registered as being neighbours of a pixel. This means however that for two pixels $p$ and $q$ which are 8-connected and thus appear in the same tree $G_i$, there is no vertex $(n, m)$ $proj(p) = n$, $proj(q) = m$ in $V$. Doing so precludes that cycles in $G_i$ are formed and that eventually searching within the graph might become impossible due to the circuit.

## 4.4 Determination of the leaves of the trees

By applying a depth-first algorithm using the back-track principle, the leaves $\Upsilon_G = \{\Upsilon_{G_0}, \ldots, \Upsilon_{G_{NT}}\}$ of the forest $G$, are determined easily. But not all the leaves $\Upsilon_{G_i}$, of the tree $G_i$ are leaf pixels in the image. Exceptions to be considered are:

1. pixels on the border of the image being a leaf of a tree, are not to be considered as leaf pixels within the image.
2. Given a leaf $v$ of $\Upsilon_{G_i}$ and another leaf $v'$ of $\Upsilon_{G_i}$, $proj(p) = v$ and $proj(q) = v'$ with $d(p,q) \leq \sqrt{2}$, then $p$ and $q$ are 8-connected and are not considered as leaf pixels within the image.
3. Given a leaf $v$ of $\Upsilon_{G_i}$ and a branch node $n \in N_i$ and $n \notin \Upsilon_{G_i}$, $proj(p) = v$ and $proj(q) = n$. If $d(p,q) \leq \sqrt{2}$ and $n$ is not the father of $v$ then $v$ has not to be removed from $\Upsilon_{G_i}$.

The above three exceptions indicate that leaves for which these cases are valid have to be removed from $\Upsilon_G$, resulting in $\Upsilon_G^* = \{\Upsilon_{G_1}^*, \ldots, \Upsilon_{G_{NT}}^*\}$.

<u>LEMMA</u> 4.5 *Given exception 2 or 3. Then the leaf $v$ belongs to a closed polygon.*

<u>proof:</u>
case 2.
Given path $r, n_1, \ldots, n_k, v$ and path $r, m_1, \ldots, m_l, v'$ and $(v, v') \in V_i$
Then there is a cycle $n_j, \ldots, v, v', m_l, \ldots, m_1$.

case 3.
as for case 2. □

Examples of the above three exceptions are given in figure 4.4. On the left we see the image and on the right the graph $G_1$ re-constructed from the image. In figure 4.4 pixel (4,1) is an example of a border pixel. It also is the root of the tree $G_1$. An example of the second case are the leaves (5,4) and (6,4) which are within a distance of $\sqrt{2}$ of each other. An example of the leaf-node case is pixel (6,10). All these leaves are not 'real' leaves and should be removed from $\Upsilon_G$. After determination of all the leaves $\Upsilon_G$ is: $\{(1,4),(5,4),(6,4),(6,10)\}$ After deletion of the non-valid leaves of $\Upsilon_G$, $\Upsilon_G^*$ is empty!

## 4.5 Edge fragment connection

Here we explicitly use the information gathered in the previous step. For every leaf of $\Upsilon_G^*$ we check for its corresponding pixel $p$ within $I$ if there are candidates for pixel addition. In order to determine the candidates for pixel connection, simple rules have been defined, which are:

- ▷ leaf-leaf connection (i.e. try to connect two leaves by adding edge pixels)
- ▷ leaf-branch connection (i.e. try to connect a leaf with a branch node by adding edge pixels).

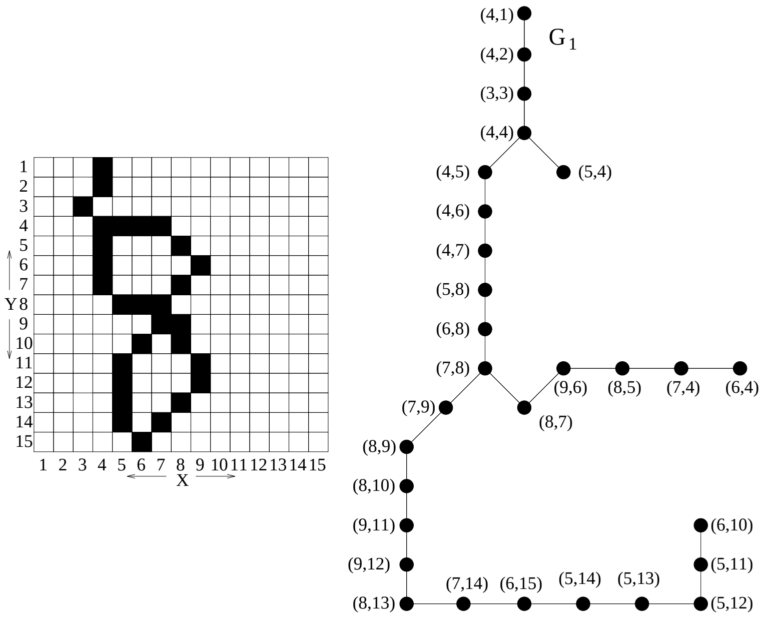



Figure 4.4: examples of leaves within a tree.

Both cases are restricted by the following constraints:

▷ leaves $v, v'$, $proj(p) = v$, $proj(q) = v'$ with a distance of $d(p, q) = 2, \sqrt{5}$ or $\sqrt{8}$

▷ leaf $v$, $proj(p) = v$ with node $n$, $proj(n) = q$ with a distance of $d(p, q) = 2, \sqrt{5}$ or $\sqrt{8}$

The idea of connecting only these two cases is generalized in algorithm 4.2.

| | | | | |
|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 |
| 2 | 1 | 1 | 1 | 2 |
| 2 | 1 | 0 | 1 | 2 |
| 2 | 1 | 1 | 1 | 2 |
| 2 | 2 | 2 | 2 | 2 |

(a)

| | | | | |
|---|---|---|---|---|
| A | B | C | D | E |
| P | 1 | 2 | 3 | F |
| O | 8 | 0 | 4 | G |
| N | 7 | 6 | 5 | H |
| M | L | K | J | I |

(b)

Figure 4.5: $5 \times 5$ matrices for pixel addition.

The set of leaves of $G$ is used for pixel addition in the image $I$. Given a leaf $v$ of $\Upsilon^*_G$ its corresponding pixel $p$ is centered within a $5 \times 5$ window. Looking at figure 4.5(a) we define the pixel labelled 0 (zero) in the centre of the window to be the layer zero pixel. The pixels labelled one are the layer one pixels and the pixels labelled 2 are the layer two pixels. The leaf-leaf and leaf-branch rules as defined above can now be re-formulated into the following: Given a layer zero edge-pixel, find a layer two edge pixel such that there is no layer one pixel connecting the layer zero and layer two pixel.

From figure 4.5(b) the following layer one triplets are formed:

$$TR_1 = \left\{ \begin{array}{l} \langle 1,2,3\rangle, \langle 2,3,4\rangle, \langle 3,4,5\rangle, \langle 4,5,6\rangle, \\ \langle 5,6,7\rangle, \langle 6,7,8\rangle, \langle 7,8,1\rangle, \langle 8,1,2\rangle \end{array} \right\}$$

and from layer two the following triplets

$$TR_2 = \left\{ \begin{array}{l} \langle B,C,D\rangle, \langle D,E,F\rangle, \langle F,G,H\rangle, \langle H,I,J\rangle, \\ \langle J,K,L\rangle, \langle L,M,N\rangle, \langle N,O,P\rangle, \langle P,A,B\rangle \end{array} \right\}$$

From both sets of triplets the following product is defined:

$$TR = \left\{ \begin{array}{l} \langle\langle 1,2,3\rangle, \langle B,C,D\rangle\rangle, \langle\langle 2,3,4\rangle, \langle D,E,F\rangle\rangle, \\ \langle\langle 3,4,5\rangle, \langle F,G,H\rangle\rangle, \langle\langle 4,5,6\rangle, \langle H,I,J\rangle\rangle, \\ \langle\langle 5,6,7\rangle, \langle J,K,L\rangle\rangle, \langle\langle 6,7,8\rangle, \langle L,M,N\rangle\rangle, \\ \langle\langle 7,8,1\rangle, \langle N,O,P\rangle\rangle, \langle\langle 8,1,2\rangle, \langle P,A,B\rangle\rangle \end{array} \right\}$$

Using fig 4.5(b) and the sets $TR$, $TR_1$, and $TR_2$ the connection can be defined as in algorithm 4.2.

For every leaf $\upsilon$ of $\Upsilon^*_G$, $proj(p)$=$\upsilon$:

1. Define $p$ to be the layer zero pixel.
2. Define $S$ = $\{2,6,8,4,3,1,5,7\}$, $i$:=1
3. $\forall$ $q$=$S(i) \in S$
   (a) IF one of the pixels $q'$ of $TR_1(i)$ has $lab(q')$ = $E$.
       THEN $i$:=$i$+1, next element of $S$
   (b) IF no one of the pixels $q'$ $TR_2(i)$ has $lab(q')$ = $E$.
       THEN $i$:=$i$+1, next element of $S$
   (c) $lab(q)$ := $E$
   (d) IF perform area test left AND
       perform area test right THEN $lab(q)$ := NOEDGE
   (e) $i$:=$i$+1

Algorithm 4.2: edge fragment connection

In algorithm 4.2 $TR(i)$ indicates the i-th element from $TR$ e.g. $TR_1(1) = \langle 1,2,3\rangle$ and $TR_1(8) = \langle 8,1,2\rangle$, $S(1) = 2$ and $S(8) = 7$.

The "area test" performed in section 3d of algorithm 4.2, is a simple test in the image which checks if the new polygon covers at least a minimum number of pixels (this number is user-defined). If the new polygon covers an area smaller than the minimum area, the added pixel is removed. This is done to overcome the problem of creation of many small polygons. The checking is done 'left' and 'right' because the closure of the gap introduces two 'separate areas' which are identified as left and right.

## 4.6 Pruning

In order to determine the closed polygons, superfluous information has to be deleted from the edge image. This means that edge-fragments not belonging to a closed polygon have to be removed. The edge fragment linking has added pixels in $I$ and, therefore it is not known which trees are connected within $G$. A simple solution to solve this is to re-construct the trees, resulting in a new graph $G' = \{G'_0, \ldots, G'_{NT'}\}$. From this graph the set of leaves is also determined, resulting in $\Upsilon_{G'} = \{\Upsilon_{G'_1}, \ldots, \Upsilon_{G'_{NT}}\}$. As before this set contains also leaves whose corresponding pixels are not to be removed from the image. These leaves are removed from $\Upsilon_{G'}$ resulting in $\Upsilon^*_{G'} = \Upsilon^*_{G'_i}, \ldots, \Upsilon^*_{G'_i}$. The leaves within this set are used as start points for edge-fragment removal from the image $I$.

In order to simplify the pruning the path from the root to a leaf $v'$ is remembered (by the backtracking). Having deleted the leaf $v'$ its predecessor is known, and can be tested for removal. This is continued until no predecessor can be found which can be removed.

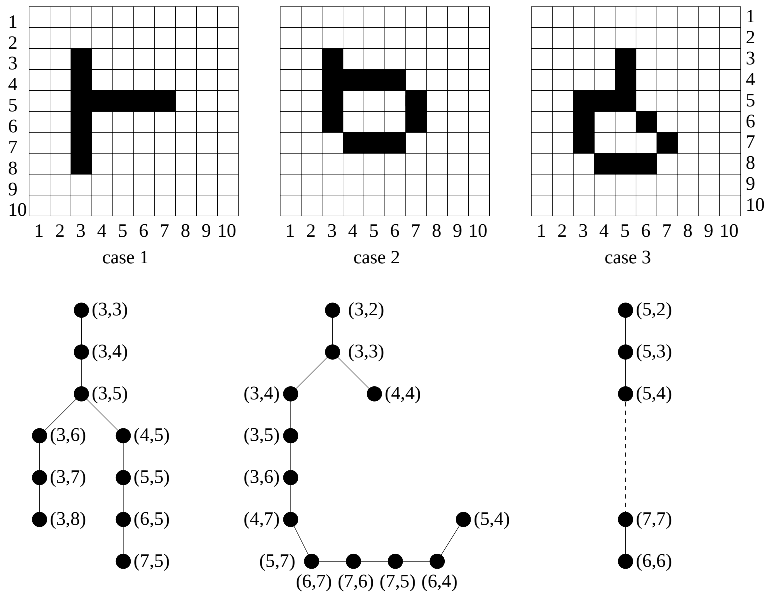



Figure 4.6: examples of pruning for the three cases.

For removal of leaves three cases are considered. Given a tree $G'_i$ and a leaf $v$, $v \in \Upsilon_{G'_i}$, $proj(p) = v$, then:

1. $v$ is the root of $G'_i$. Given a path $v = n_0, \ldots, n_k$ with $k \geq 1$, $proj(p_i) = n_i$ and $n_0, \ldots, n_{k-1}$ have degree 2, $n_0, n_k$ have degree 1. If for all $n_0, \ldots, n_{k-1}$ there is no node $n'$ with $proj(q) = n'$ for which $d(p_i, q) = \sqrt{2}$ and $n_i$ not the (grand)father of $n'$, then $n_0, \ldots, n_{k-1}$ can be removed from $G'_i$. If $n_k$ has degree one, remove $n_k$.

2. Given a path $n_0, \ldots, n_k = v$ with $k \geq 0$, $proj(p_i) = n_i$, and $n_0, \ldots, n_{k-1}$ have degree 2, $n_k$ has degree 1. Given a leaf $v' \neq v$ with $proj(q) = v'$. If for all $n_i$, $i = 0, \ldots, k$, $d(p_i, q) \geq \sqrt{2}$, then $n_0, \ldots, n_k$ can be removed from $G'_i$.

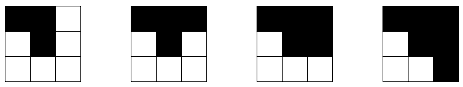
3. Given a path $n_0, \ldots, n_k = v$ with $k \geq 0$, $proj(p_i) = n_i$, and $n_0, \ldots, n_{k-1}$ have degree 2, $n_k$ has degree 1. Given a node $n' \notin \{n_0, \ldots, n_k\}$ with $proj(q) = n'$. If for all $n_i$, $i = 0, \ldots, k$, $d(p_i, q) \geq \sqrt{2}$, then $n_0, \ldots, n_k$ can be removed from $G'_i$.

Examples of the above three cases are given in figure 4.6. The first case states that for nodes from the root until the first node having degree two or more, sons can be removed.

<u>LEMMA</u> 4.6 *Given a path $w$: $v = n_0, \ldots, n_k$ If $\exists n' \notin \{n_0, \ldots, n_k : d(p_i, q') \leq \sqrt{2}$ } where $proj(p_i) = n_i$ and $proj(q') = n'$. Then $d(p_i, q') = \sqrt{2}$.*

<u>proof:</u>
Given a path $w$: $v = n_0, \ldots, n_k$ as defined. Now given a node $n' \notin \{n_0, \ldots, n_k\}$, $proj(q) = n'$. If $d(p_i, q) = 1$, $n'$ would be a son of $n_i$ and $p$ would have an degree of three and so there would be a shorter path $w'$: $n_0, \ldots, n_i$. □

The second and the third case need to be evaluated at the same time. If for a certain path $w$: $n_0, \ldots, n_k = v$ case two is valid then also case three has to be evaluated before removal of the path is considered, and of course vice versa.



A B C D

Figure 4.7: templates used for pruning.

The pruning algorithm is derived from above three cases and uses the leaves as start points. For every leaf of $\Upsilon^*_{G'}$ the situations as given in figure 4.7 are evaluated. The black squares are edge pixels. The central pixel of the templates is mapped onto the leaf. If situation 'A' or 'B' or 'C' or 'D' is valid then the leaf can be removed from the branch and its father becomes the new leaf. Of course for all the cases the templates are rotated in all eight directions. The algorithm is given in algorithm 4.3.

1. `Given a leaf` $v \in \Upsilon_{G'_i}$ `and path` $P$;
   $P$ : $r = p_0, \ldots, p_k = v$
2. $\forall p_i \in P$, $i = k, k-1, \ldots, 1, 0$
   - `template(A,B,C,D) is mapped upon` $p_i$;
   - `WHILE template(A,B,C,D) not rotated in all directions`
   - `DO`
     - `IF (template(A,B,C,D) fits edge pixels in` $I$`)`
     - `THEN remove` $p_i$ `from` $P$;
       - $k := k - 1$;
   - `ENDDO`

Algorithm 4.3: tree pruning.

## 4.7 Determination of closed polygons

The detection of closed polygons is done by a floodfill procedure, as given in Smith [101]. It assumes that the boundaries of regions are 8-connected. But the filling itself uses only 4-connectedness, i.e. in figure 4.2 for pixel 'p' only the pixels '1', '3', '5', and '7' are examined. It assumes that all the pixels within a region have the same label, which is different from the label of the boundary (edge) pixels. A stack is employed to pop and push seed-points.

The floodfill results in labelling all the pixels within a closed polygon. By scanning the surrounding rectangle (i.e. bounding box) of the closed polygon, all the pixels with a neighbour labelled as being an edge pixel are determined.

## 4.8 Results

One of the images for which we show results is image 3.5. The results of the edge fragment connection and pruning are given in image 4.1. If we only perform the pruning without the edge fragment connection image 4.2 is the output. To show the strength of this method we show the results of the edge detection as well as the edge fragment connection for another image, image 4.3. This Landsat-TM image is of the Slieve Bloom Mountains in Central Ireland and has been taken on June 13, 1988. It consists of 5 channels (TM bands 1-5). The results of the edge detection are given in image 4.4. As can be seen this is quite a noisy image, probably due to the small fields in the scene. The image after the edge fragment connection is given in image 4.5. This result is clearly well structured and already looks like a reasonably good segmentation.

## 4.9 Concluding remarks

Here an edge-fragment connection algorithm is described which adds pixels between edge fragments having a gap of one pixel, using a method which conforms to simple rules. At the end the pruning step removes edge fragments not belonging to closed polygons, so that the final result is a clean edge image composed only of closed polygons and nothing else. Different possibilities for the implementation of these algorithms are given in Verstraaten [111].

A disadvantage of this method is its ad hoc character. If for a certain leaf more than one connection is possible, the first found will be added.

Also information available within the vicinity of a leaf pixel is not considered. However its strongest point is its simplicity.

A possible improvement in the connection procedure would be to use the direction of the edge pixels. While following, the pixels would be labelled according to their direction (only four directions are needed). Given a leaf, an attempt would first be made to connect this leaf with another edge pixel laying in the leafs direction. If one is found these are connected and the connection as described before would not be used. There would be a (small) additional computaional overhead in doing this.

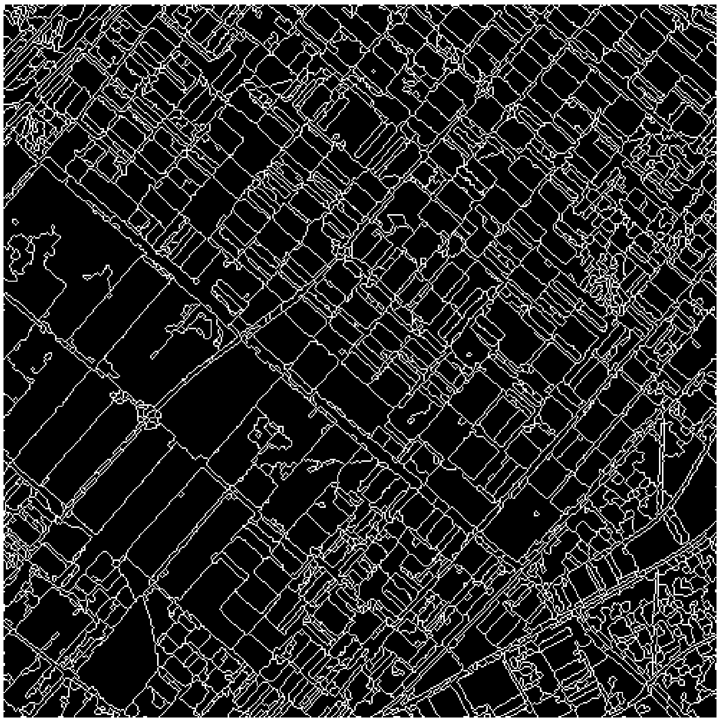


Image 4.1: closed polygons found in image 3.5 (Flevopolder image) with pixel connection.

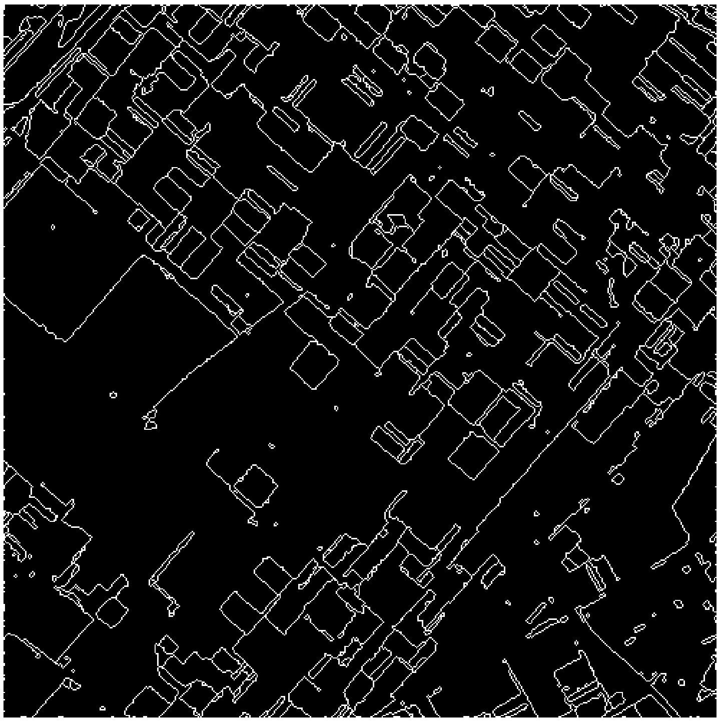


Image 4.2: closed polygons found in image 3.8 (Flevopolder image) without pixel connection.

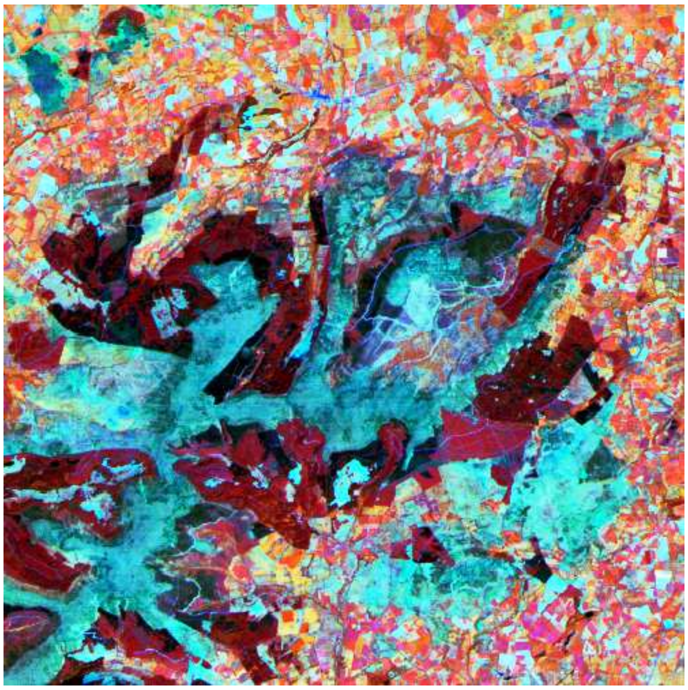


Image 4.3: Landsat-TM image, Slieve Bloom Mountains area, Central Ireland, June 13, 1988 (lsjun88.lan, 500 × 500 pixels 5 bands, channel 4,5,3 : R,G,B), Original satellite data ©Eurimage.

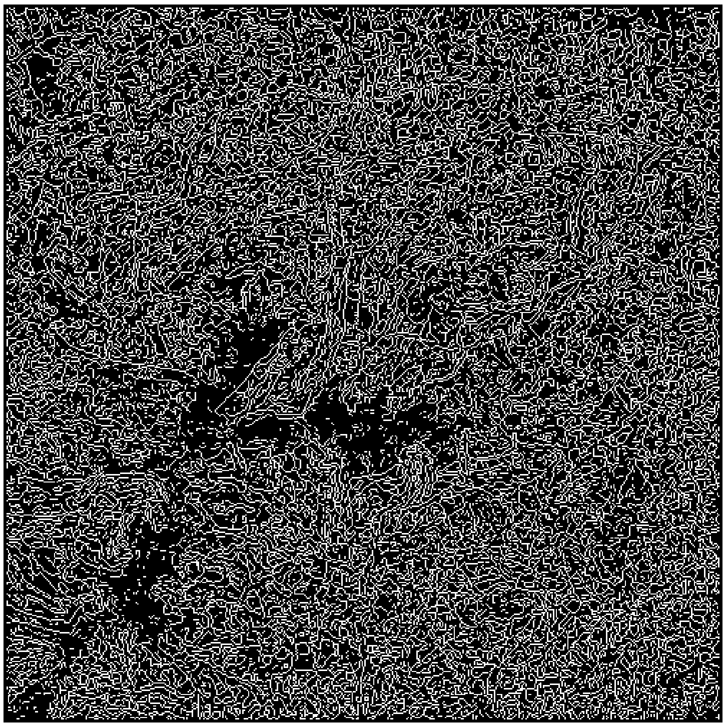


Image 4.4: edges found in Ireland image, lsjun88.lan.

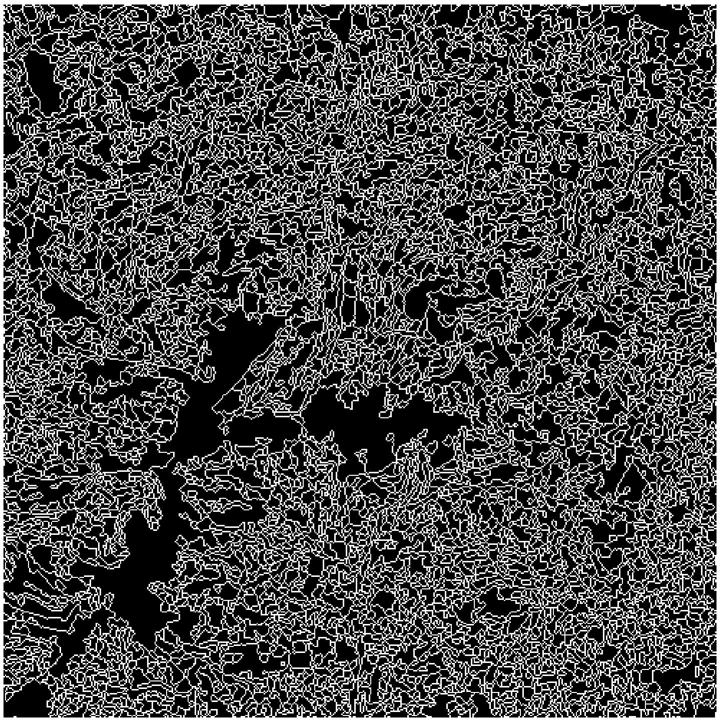


Image 4.5: closed polygons with pixel connection found in image 4.3.

# Chapter 5

## Order Independent Region Growing

### 5.1 Introduction

In this chapter the region growing method as proposed by Tilton [106, 107, 108, 109] and, Beaulieu & Goldberg [7] is examined and adapted for integration with the edge detection process. This so called 'best merge' algorithm examines, in every merging iteration, the whole image context and merges that pair of adjacent regions having minimum merging cost. In case of ambiguity, when several pairs with the same minimum merging cost are possible, both authors do not give a solution for selecting a pair. Randomly selecting a pair might violate the order independence claim, i.e. the final segmentation result might be different when a different pair is selected. In order to achieve complete order independence, in case of ambiguity it is proposed to merge all the pairs (with the same merging costs) simultaneously. This means that in one iteration several (adjacent) regions might be merged into one region, or that non adjacent pairs (with the same merging costs) somewhere in the image are merged simultaneously. To verify that it is allowed to merge 3 (or more) adjacent regions in one iteration into one region, a cost function for the merging of these 3 (or more) adjacent regions, is derived. It is proven that the cost of simultaneous merging is not higher than pairwise merging of regions.

### 5.2 Order Dependence

Most region growing methods are sequential processes, i.e. the output of process 'i' is used as input for process 'j' and are subject to the problem of order dependence. Zucker [123] defines order dependence as: 'information acquired in earlier stages of processing can (will) be used to influence the processing of later stages'. Thus the first image points processed may affect the choice of the points to be processed next. More importantly the order in which image points are processed may affect the final result. This means that selection of seed points is vital for a good segmentation result.

Figure 5.1 showing 'two sunny sides up eggs' explains the problem. Two initial regions A and B are shown. If first A is grown until its new border, the filled area, will belong to A. But if first B is grown the filled area will belong to region B.

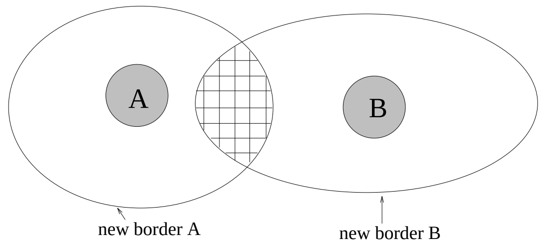



Figure 5.1: the egg model.

Here however we would like to overcome the order dependence problem as stated in Tilton [107] and Ait Belaid *et al.* [1]. The end result of the segmentation should be independent of the choice of the start points, and it should depend only on the spectral signatures of the pixels and their spatial relationship.

## 5.3 The best merge algorithm

To overcome the order dependence problem an ideal solution would be to merge only the most similar pair of spatially adjacent regions (pixels) in the entire image at each iteration as stated in Beaulieu & Goldberg [7] and, Tilton [109, 106, 107, 108]. This means that at every iteration the whole image context is examined, before the actual merging takes place. The algorithm in its basic form is given in algorithm 5.1. This approach is computationally intensive.

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

Algorithm 5.1: principle of the best merge.

Beaulieu & Goldberg [7] do not explicitly mention the order dependence problem but in Ait Belaid *et al.* [1] it is said that the order dependence problem can be overcome by using Beaulieu & Goldberg's algorithm.

Tilton speaks about performing a best merge at every iteration. A problem arises if two or more pairs have the same similarity value, or have a region in common.

Selection of the right pair in case of ambiguity is important, because it can affect the final segmentation result. In order to overcome this problem, a possibility is to merge all the pairs with the lowest similarity value simultaneously as stated in Schachter *et al.* [92]. The claim here is that complete order independence can be achieved if and only if pairs having the same similarity value are merged simultaneously. An algorithm will be proposed which performs the merging of a number of regions having the same merging cost, simultaneously (even on a sequential machine). However the total cost of the merging might be higher than if a pair is chosen randomly in case of ambiguity.

In order to compute a similarity value for adjacent pairs of regions, a similarity criterion has to be defined. The output of this criterion value for regions determines which regions are to be merged. It must be clear that for a different choice of a similarity criterion, different regions might be merged. Thus the selection of the similarity criterion might also influence the final result.

Tilton [106, 107] initially called his algorithm Spatially Constrained Clustering (SCC) which has been renamed since 1989 as Iterative Parallel Region Growing (IPRG) [107, 108]. Beaulieu & Goldberg call their algorithm the Hierarchical Stepwise Optimization Algorithm (HSWO).

## 5.3.1 The similarity criteria

### 5.3.1.1 Tilton's criterion

Tilton [107] defined a **dis**similarity criterion based on minimizing the increase in variance normalized mean square error. The mean square error ($MSE$) of the k-th band of a multi-band image is defined as:

$$MSE_k = E\left[(D_k - D_k^r)^2\right] \cong \frac{1}{N-1}\sum_{p=1}^{N}(D_{kp} - D_{kp}^r)^2 \tag{5.1}$$

where $D_k$ and $D_k^r$ are the data values of the k-th band of the original and region mean images, respectively; $D_{kp}$ and $D_{kp}^r$ are the values of the p-th pixel of the k-th band of the original and reconstructed images respectively; $E$ denotes the expected value; and $N$ is 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 band ($NMSE_k$) is defined as:

$$NMSE_k = \frac{MSE_k}{\sigma_k^2} \tag{5.2}$$

where $\sigma_k^2$ is the variance of the k-th band. For a particular pair of spatially adjacent regions, $\Delta NMSE_k$ is the change in $NMSE_k$ when the pair of regions is merged and the new region mean image is formed and compared to the original image. This change in $NMSE_k$ (i.e. $\Delta NMSE_k$) is calculated as follows:

$$\Delta NMSE_k = \frac{MSE_k^c - MSE_k}{\sigma_k^2} \tag{5.3}$$

where $MSE_k^c$ is the mean squared error when regions i and j are merged, while $MSE_k$ is the mean squared error before regions $i$ and $j$ are merged. Equation 5.3 can be rewritten as:

$$\Delta NMSE_k = \frac{n_i(\overline{D}_{ki} - \overline{D}_{kij})^2 + n_j(\overline{D}_{kj} - \overline{D}_{kij})^2}{(N-1)\cdot\sigma_k^2} \tag{5.4}$$

where $n_i$ and $n_j$ are the number of pixels in region $i$ and $j$ respectively, before merging. $\overline{D}_{ki}$ and $\overline{D}_{kj}$ are the mean values of band $k$ for regions $i$ and $j$ respectively, before merging. $\overline{D}_{kij}$ is the mean value of band $k$ for the region that would result from merging the regions $i$ and $j$.

In order to create a region for all the bands (i.e. not having different regions for different bands) Tilton takes the sum (or equivalently, the average) over the bands, or the maximum over the bands [1].

In Baraldi & Parmiggiani [5] it is shown that the change in NMSE is the result of three different effects: the first is related to the contrast existing between the two segments to be merged, while the second and the third are due, respectively, to the ratio (product) and to the sum of the two segment areas. The possible combinations among these effects may lead the IPRG algorithm to some unacceptable behaviour. Furthermore it is stated that the IPRG similarity criterion considers as the best single merge, the segment pair which produces the lowest increase in $\Delta NMSE$. However, this is a *post*-merge[2] heterogeneity control, when two segment feature vectors have been merged into one new parameter (which is in the IPRG case the $\Delta NMSE$ variable).

In general, the similarity measurement should be used as an *ante*-merge condition, whose objective is to compare two segments, each one described by its own feature vector, in order to assess their degree of match (similarity value). Applying the similarity measurement as a *post*-merge condition rather than as an *ante*-merge condition might be very misleading.

In their paper it is shown that the theoretical observations provided, show that the IPRG similarity criterion is not effective in pursuing spectrally homogeneous region segmentation by merging spectrally similar regions, each region being characterized by low-within segment variability. At the same time, in no way can the IPRG algorithm pursue texture homogeneous region segmentation [5].

But Tilton does not intend that his similarity criterion is connected with the IRPG algorithm: 'I have always thought of the similarity criterion that I have used as an "example" criterion just used for the development of the overall algorithm. I NEVER intended that the NMSE or Change in Entropy criterion should be intimately connected with the IPRG algorithm. They are just placeholders waiting for a demonstrably better replacement for a particular application'[3].

#### 5.3.1.2 Beaulieu and Goldberg's criterion

Beaulieu & Goldberg [7] regard a picture as a two dimensional function $f(x, y)$, where $(x, y) \in I, I$ being the image plane. A picture partition $P$ divides the picture plane $I$ into $n$ regions, $S_1, S_2, \ldots, S_n$. Let $f_i(x, y)$ designate the pixel values for the region $S_i$. Now $f_i(x, y)$ can be written:

$$f_i(x, y) \approx r_i(x, y) \tag{5.5}$$

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

[1]after a private communication, May, 1994.

[2]A *ante*-merge similarity condition, applied to a segment pair eligible for merging, means that the condition compares the statistics describing the first region of the pair with the second region. Whereas a *post*-merge similarity condition, refers to a segment pair eligible for merging, assesses the statistics of the third region resulting from the merging

[3]after a private communication, July, 1994.

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

$$H(S_i) = \sum_{(x,y)\in S_i} (f(x,y) - r_i(x,y))^2. \tag{5.6}$$

The increase of error while merging two segments must be minimized and is defined as a sequence of stepwise optimizations (i.e. for every merging iteration). Thus the goal of the stepwise optimization problem is therefore to find the two segments for which merging produces the smallest increase in error. Therefore the stepwise criterion is defined as:

$$C_{ij} = H(S_i \cup S_j) - H(S_i) - H(S_j). \tag{5.7}$$

By choosing

$$r_i(x,y) = \mu_i, \tag{5.8}$$

the segment approximation error or the segment cost is given as:

$$H(S_i) = \sum_{(x,y)\in S_i} (f(x,y) - \mu_i)^2, \tag{5.9}$$

where $\mu_i$ is the mean value of region $i$. The segment approximation error can be rewritten as:

$$C_{i,j} = \frac{N_i \cdot N_j}{N_i + N_j}(\mu_i - \mu_j)^2 \tag{5.10}$$

where $N_i$ and $N_j$ are the number of pixels in regions $i$ and $j$ respectively.

### 5.3.2 The stopping criteria

#### 5.3.2.1 Tilton's stopping criterion

Tilton defines in [107] a merging threshold. If a merge's (dis)similarity function is not better or equal to this threshold, no merging will be carried out. The threshold is defined as follows:

$$\text{Threshold} = \frac{ME - NMSE}{\frac{1}{2}NR \cdot FR} \tag{5.11}$$

where $ME$ is a user defined parameter, $NR$ the number of regions at the current iteration, $FR$ is the estimated fraction of regions participating in merges at each iteration and,

$$NMSE = \sum_{i=1}^{N1} \sum_{j=1}^{N2} \left( \max_{k=1}^{m}(NMSE_{kij}) \right) \tag{5.12}$$

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

In Baraldi & Parmiggiani [5] it is shown that there is need of an *ante*-merge mean vector comparison, followed by a *post*-merge homogeneity test, performed by a spectrally homogeneous region segmentation algorithm.

Tilton left the above criterion in his current version of the IPRG. 'The stopping criterion is a very difficult problem, since what is the "best" segmentation varies depending on the application. Also, you should find that, while the segmentation produced at a particular iteration may produce a generally "best" segmentation, in local areas this segmentation may be too coarse or too fine. Ultimately, one needs a locally adaptive stopping criterion, that can be adjusted for the needs of particular applications' [4].

#### 5.3.2.2 Beaulieu and Goldberg's stopping criterion

Beaulieu & Goldberg define as a stopping criterion either the number of regions merged as a percentage of the initial regions, or a value for the approximation error.

### 5.3.3 The algorithms

#### 5.3.3.1 Tilton's algorithm

In his papers Tilton [109, 106, 107] states that the approach as given in algorithm 5.1 becomes impractical for large images even on a massively parallel computer. In order to overcome the problem of the high computational load, Tilton approached the best merge by performing in parallel a merge of selected sets at each iteration, see algorithm 5.2.

```
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 5.2: principle of the 'Iterative Parallel Region Growing' (IPRG) algorithm.

To achieve this merging of similar pairs in parallel, Tilton defines a set of sub-images recursively with respect to a certain region. A level 0 sub-image is the empty set. A level 1 sub-image with respect to a region is the region itself. A level 2 sub-image, with respect to a region is the level 1 sub-image with respect to that sub-image and all regions that are spatially adjacent to the level 1 sub-image. A level n sub-image with respect to a region is the level n-1 sub-image, with respect to that region, plus all regions that are spatially adjacent to the level n-1 sub-image. In figure 5.2 the different levels can be recognized immediately. The region labelled 'R' represents a level 1 sub-image with respect to region 'R'.

[4] after a private communication, July, 1994.

The regions labelled 'R' and 2 represent a level 2 sub-image, and the regions labelled 'R', 2 and 3 represent a level 3 sub-image with respect to region 'R'.

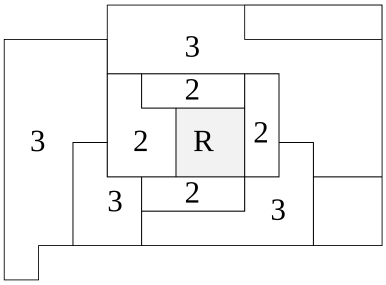



Figure 5.2: level 1,2, 3 subimages.

Tilton states that the following scheme closely approximates the one merge per iteration approach. At each iteration, divide the image into a set of, possibly, overlapping sub-images and then perform the best merge within each sub-image. After each iteration the sub-images will have to be re-defined appropriately.

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

#### 5.3.3.2 Beaulieu and Goldberg's algorithm

Beaulieu & Goldberg noticed that only the neighbourhood of previous merged adjacent region pairs has to be recalculated, already reducing computation time significantly. They employ variables to hold the state in which the segmentation is at the current iteration. The HSWO algorithm starts with an initial picture partition $P^0 = \{S_1, S_2, \ldots, S_n\}$ obtained by merging pixels with the same spectral values. At each iteration it merges two segments to yield a segment hierarchy. The algorithm is given in algorithm 5.3.
The variables involved in the algorithm are as follows:
1) $B_i$, the neighbourhood of segment $S_i$
2) $D_i$, the parameters that describe the segment $S_i$ e.g. the segment mean and size
3) $C_{i,j}, = C(D_i, D_j)$, the cost of merging segment $S_i$ with $S_j$, where $S_j$ is contained in $B_i$.
4) $k$ denotes the level-number, $m$ the segment number.
Step 2 iii) is added in algorithm 5.3 compared to [7] because in [7] $S_m$ was not defined. In contrast with Beaulieu & Goldberg we do not use the intersection and union notation, but for the sake of simplicity we use the '+' and '-' operators. The notation 'A+B' means that elements from set B are added to set A. The notation 'A-B' indicates the elements from set B are removed from set A.

1. Initialization

   i) $P^0 = \{S_1, S_2, \ldots, S_n\}$
   ii) $k = 0$ and $m = n$
   iii) calculate $D_i$ and $B_i$ for $\forall S_i \in P_0$
   iv) calculate $CS = \{C_{i,j} \mid S_j \in B_i$ and $i > j\}$

2. Merge the two most similar segments

   i) $k = k + 1$ and $m = m + 1$
   ii) $C_{u,v} = \mathrm{MINIMUM}\ (CS)$
   iii) $S_m = \{S_u, S_v\}$
   iv) $P^k = P^{k-1} + \{S_m\} - \{S_u, S_v\}$
   v) calculate $D_m$ from $D_u$ and $D_v$
   vi) $B_m = B_u + B_v - \{S_u, S_v\}$
   vii) $\forall_{S_j \in B_m}\ [B_j = B_j + \{S_m\} - \{S_u, S_v\}]$
   viii) $CS = CS + \{C_{m,j} \mid S_j \in B_m\} - \{C_{i,j} \mid i, j = u \text{ or } v\}$

3. Stopping condition

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

Algorithm 5.3: Beaulieu and Goldberg's algorithm.

## 5.4 Similar merging pairs

A problem arises if two or more pairs of regions (within a sub-image) have an equal similarity criterion value. Tilton states that he randomly selects a merging pair[5]. Doing so might violate the order independence claim. Lately Tilton adapted his IPRG: 'In the current version of IPRG, the merge constraint scheme DOES break ties (the lowest region label wins), making merge constraint level 1.0 totally equivalent to Willebeek-LeMair's [119] mutually best pairwise constraint[6].

However, this is not sufficient, if the image is rotated 90 degrees, then another region will have the lowest region number, as the regions are numbered differently.

Beaulieu & Goldberg give no solution for this case. It is assumed however, that they also select a pair randomly in case of ambiguity. In order to fulfill the order independence claim, it is necessary to merge the pairs having the same similarity criterion value [92], in one iteration in parallel (simultaneously).

[5]after a private communication, May, 1992.
[6]after a private communication, July, 1994.

## 5.5 A cost function for three regions

In their paper Beaulieu & Goldberg derived a cost function for merging that minimized the approximation error. Starting from the approximation error, equation (5.9) for one segment, using the stepwise criterion as given in equation (5.7), they derived the cost function for merging two segments $H(S_i)$ and $H(S_j)$ as given in equation (5.10).

Given three segments $S_i$, $S_j$, $S_k$ of a picture partition $P$, for which $C_{ij}$ and $C_{jk}$ are equal, then a cost function for the simultaneous merging of the three segments can be given. Redefine the stepwise criterion as:

$$C_{ijk} = H(S_i \cup S_j \cup S_k) - H(S_i) - H(S_j) - H(S_k)\,. \tag{5.13}$$

By using:

$$\sum_{(x,y)\in S_i} f(x,y) = N_i \mu_i \tag{5.14}$$

and

$$\mu_{i\cup j\cup k} = \frac{N_i\mu_i + N_j\mu_j + N_k\mu_k}{N_i + N_j + N_k} \tag{5.15}$$

we can derive the following cost function

$$C_{i,j,k} = \frac{N_i \cdot N_j(\mu_i - \mu_j)^2 + N_i \cdot N_k(\mu_i - \mu_k)^2 + N_j \cdot N_k(\mu_j - \mu_k)^2}{N_i + N_j + N_k}\,. \tag{5.16}$$

When more than 3 regions are involved in the merging process, the cost function can be generalized as follows:

$$C_{i_1,\cdots,i_{NR}} = \frac{\displaystyle\sum_{k=0}^{NR}\sum_{l=k+1}^{NR} N_{i_k}N_{i_l}(\mu_{i_k} - \mu_{i_l})^2}{\displaystyle\sum_{k=0}^{NR} N_{i_k}} \tag{5.17}$$

where $NR$ is the number of regions to be merged into one.

## 5.6 Cost comparison of pairwise or simultaneous merging

Here it is shown that the cost for merging three regions simultaneously in parallel is equal to merging in two consecutive steps two pairs from which one is in common.

LEMMA 5.1 *Given three regions $S_i$, $S_j$, $S_k$ for which $C_{ij} = C_{jk}$ then the sum of first merging $S_i$ and $S_j$ and secondly $(S_{i\cup j})$ and $S_k$ into $S_{(i\cup j)\cup k}$ is equal to the cost of merging in one step $S_i$, $S_j$ and $S_k$ into $S_{i\cup j\cup k}$.*

proof:

For the merging of three segments in parallel the cost function is $C_{i,j,k}$ as defined in equation (5.13). The merging of first $S_i$ and $S_j$ is given in equation (5.7) and then $S_{i\cup j}$ and $S_k$ into $S_{(i\cup j)\cup k}$ is given as follows:

$$C_{(i\cup j)k} = H(S_{i\cup j} \cup S_k) - H(S_{i\cup j}) - H(S_k) \; . \tag{5.18}$$

The total cost is now defined as:

$$C_{(i\cup j)k} + C_{ij}. \tag{5.19}$$

Given equation (5.19) and equation (5.13) it is sufficient to prove that

$$C_{ijk} = C_{(i\cup j)k} + C_{ij}, \tag{5.20}$$

resulting in:

$$\begin{aligned} &H(S_i \cup S_j \cup S_k) - H(S_i) - H(S_j) = \\ &H(S_{i\cup j} \cup S_k) - H(S_{i\cup j}) + H(S_i \cup S_j) - H(S_i) - H(S_j). \end{aligned} \tag{5.21}$$

Using

$$\mu_{i\cup j} = \frac{N_i \cdot \mu_i + N_j \cdot \mu_j}{N_i + N_j} \tag{5.22}$$

and equations (5.15) and (5.9) it is easy to proof that

$$H(S_{i\cup j} \cup S_k) = H(S_i \cup S_j \cup S_k) \tag{5.23}$$

and

$$H(S_{i\cup j}) = H(S_i \cup S_j) \tag{5.24}$$

and equation (5.20) follows. □

This means that subsequent merging of 2 regions which have a region in common and which have both minimum cost can be replaced by the simultaneous merging of three regions, indicating that the piecewise approximation of the regions and their merging as set by Beaulieu & Goldberg [7] is still valid.

**An example**

The following example illustrates the above, i.e. the merging of 3 regions simultaneously is as expensive as first merging two of them into a new one, and then merging this new one with the remaining one into the final region.

For the regions $S_1$, $S_2$, $S_5$ and $S_6$ in figure 5.4 the total cost when merging first $S_1$, $S_2$ into $S_{1,2}$ then, $S_5$, $S_6$ into $S_{5,6}$ and finally $S_{1,2}$ and $S_{5,6}$ into $S_{1,2,5,6}$ is $1.5 + 1.5 + 12 = 15$.

Simultaneous merging of $S_1$, $S_2$, $S_5$ and $S_6$ into $S_{1,2,5,6}$ gives also 15.

## 5.7 Order independent best merging

### 5.7.1 The similarity criterion

In contrast to both Tilton and Beaulieu & Goldberg our similarity criterion does not take the segment size into account. An advantage is that regions are merged depending only upon their spectral values and not on their size and or shape.

The merging criterion defined is the Euclidean spectral distance between the regions $i$ and $j$.

$$C_{i,j} = \sqrt{(\mu_{1i} - \mu_{1j})^2 + \cdots + (\mu_{ci} - \mu_{cj})^2} \tag{5.25}$$

where $c$ is the number of channels. If only one channel is involved equation 5.25 becomes simply

$$C_{i,j} = |\mu_i - \mu_j| \,. \tag{5.26}$$

How is the above merging criterion related to our aims? In Southern European countries the fields are normally only 1-2 ha. in area (about 10-20 Landsat-TM pixels). In table 5.1 a comparison between Northern and Southern European countries is given for farms holding less than 20 ha. To avoid merging of regions,

| country | percentage |
|---|---|
| UK | 44% |
| Denmark | 46% |
| Greece | 98% |
| Portugal | 98% |
| Spain | 88% |

Table 5.1: percentage of total farm holdings of less than 20ha. in area, source Eurostat [30].

depending upon their field size, the merging criterion which they defined, needs to be changed to one which is independent of the size of the fields. The Euclidean distance supplies a merging criterion with this property. In section 5.8, for the merging criterion as proposed by Beaulieu & Goldberg it is shown, how the field size influences the merging cost, and thus the sequence of the pairs to be merged, see table 5.3.

### 5.7.2 The stopping criterion

As our criterion is the Euclidean distance we simply set a threshold which is a 'derivative' of this distance (depending upon the number of channels in the image). A disadvantage is its ad hoc character. Several thresholds have to be tried interactively. However, the threshold is independent of the number of regions or their size, in contrast with Tilton and Beaulieu & Goldberg. The set threshold only depends on the differences in intensity within the image and ideally needs to be larger than the variations *within* the different ground cover types and smaller than the mean difference *between* the different ground cover classes. These values can be known a-priori (i.e. using ground truth data) or can be derived with statistical methods [98] for a given image.

### 5.7.3 The new order independent merging algorithm

The first objective in developing a new best merge algorithm is that pairs with the same similarity value are merged within the same iteration simultaneously. This will also be the main difference of our algorithm from Beaulieu & Goldberg's algorithm. A second difference will be that no neighbour lists $B_i$ are created. Only at the end of every iteration after the merging of the similar pairs will a neighbour list be created, in order to update the distance list.

Given a pair of adjacent regions to be merged all the neighbours are computed. How this is done is explained in a following section. If the similarity criterion value for all adjacent pairs is greater than the merging threshold then the merging stops. All pairs having a similarity criterion value below the threshold are stored in an array.

The algorithm which is given in algorithm 5.4 starts with an initial picture partition $S = \{S_1, S_2, \ldots, S_n\}$ and at each iteration merges two or more segments with an equal similarity criterion value. Initially every pixel is regarded as a separate element of $S$. The variables involved in the algorithm are defined as follows:

1. $M_i$, the mean intensity value of segment $S_i$
2. $N_i$, the size of $S_i$ (number of pixels contained in $S_i$).
3. $P_i = \{e_1, e_2, \ldots, e_{N_i}\}$, the set of pixels contained in $S_i$.
4. $C_{i,j}$, the cost of merging segment $S_i$ with $S_j$, where $S_i$ and $S_j$ are adjacent.
5. T, the merging threshold depending on function $C$.

In step 2 (i) the set of pairs of segments $S_u, S_v$ having minimum cost $C_{u,v}$ is calculated. In step 2 (ii) we determine for every pair $S_u, S_v$ their neighbours and their costs, if such a pair is a member of $CS$. In step 2 (iii) all segment pairs found in 2(i) or 2(ii) are deleted from $CS$ to create a new set $CS'$. In step 2 (iv) the actual merging is done. All pairs having minimum cost are merged in parallel. If more than one segment pair share a common segment then they are spatially contiguous and are merged into one. In step 2 (v) all segments are removed from $S$ which are contained in the segment pairs in $CS_{\min}$ since these will have been merged. In step 2 (vi) the newly created segments which are members of $S_{\text{new}}$ are added to $S$, In step 2 (vii) and 2 (viii) the properties of the newly created segments are computed. In steps 2 (ix) - 2 (xi) the new set of segment pairs is created where each pair has at least one element in $S_{\text{new}}$, the pairs are adjacent, and the cost of merging each pair is within the threshold. In step 2 (xii) the new segment pairs found in steps 2 (ix) - 2 (xi) are added to $CS'$. In step 2 (xiii) $S$ and $CS$ are updated to contain (a) the new set of segments making up the image and (b) the set of segment pairs which are within the merging threshold respectively.

Because of the consistent simultaneous merging in case of ambiguity we call our algorithm Simultaneous Order Independent Iterative Merging (SOIM).

1. Initialization
   - i) $S = \{S_1, S_2, \ldots, S_n\}$
   - ii) set value of threshold T
   - iii) merge $S_i$ and $S_j$ if $M_i = M_j$ and $S_i$ and $S_j$ are adjacent
   - iv) $CS = \{\langle S_i, S_j \rangle \mid S_i$ and $S_j$ adjacent and $i < j$ and $C_{i,j} < T\}$

2. Merge the most similar segments
   - (i) $CS_{\texttt{min}}$ = $\{\langle S_u, S_v \rangle \mid \langle S_u, S_v \rangle \in CS$ and $C_{u,v}$ = MINIMUM $\}$
   - (ii) $CS_{\texttt{nbr}} = \{\langle S_u, S_k \rangle, \langle S_v, S_l \rangle \mid \langle S_u, S_k \rangle \in CS$ or $\langle S_v, S_l \rangle \in CS\}$
   - (iii) $CS'$ = $CS$ - $CS_{\texttt{min}}$ - $CS_{\texttt{nbr}}$
   - (iv) simultaneously create a new set $S_{\texttt{new}}$ of all possible new segments $S_m$, each from $K$ segment pairs $\langle S_x, S_y \rangle$ such that:

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

     and the $K$ segment pairs are spatially contiguous.
   - (v) $S'$ := $S$ - $S_i$ where $S_i$ is one component of a pair $\langle S_p, S_q \rangle$ and $\langle S_p, S_q \rangle \in CS_{min}$
   - (vi) $S''$ := $S' \cup S_{\texttt{new}}$
   - (vii) $\forall S_i \in S_{\texttt{new}}$ : calculate $M_i$
   - (viii) $\forall S_i \in S_{\texttt{new}}$ : calculate $C_{i,j}$ where $S_j \in S''$ and $S_i, S_j$ adjacent
   - (ix) $CS_{\texttt{new1}} = \{\langle S_i, S_j \rangle \mid \forall S_i \in S_{\texttt{new}} : S_j \in S'', i < j, C_{i,j} < T$ and $S_i, S_j$ adjacent.$\}$
   - (x) $CS_{\texttt{new2}} = \{\langle S_i, S_j \rangle \mid \forall S_j \in S_{\texttt{new}} : S_i \in S'', i < j, C_{i,j} < T$ and $S_i, S_j$ adjacent.$\}$
   - (xi) $CS_{\texttt{new}} = CS_{\texttt{new1}} \cup CS_{\texttt{new2}}$
   - (xii) $CS'' = CS' \cup CS_{\texttt{new}}$
   - (xiii) $CS := CS''; S := S''$;
     Go back to 2(i) unless stopping criterion fulfilled.

3. Stopping condition

   $CS$ = ∅ .

Algorithm 5.4: Simultaneous Order Independent Iterative Merging (SOIM).

## 5.8 Two examples for the three different versions

Here two examples of the different similarity criteria as defined by the different authors on two windows are given. The first example is the one given in Beaulieu & Goldberg [7] and is an example of a 'nice' one. For both Tilton and, Beaulieu & Goldberg no ambiguity between pairs appears. The second example (see Schreurs [100]) slightly differs from the first one so that for all authors, ambiguity between pairs appears.

| | | | |
|---|---|---|---|
| 1 | 2 | 2 | 13 |
| 1 | 10 | 2 | 13 |
| 1 | 3 | 3 | 13 |
| 6 | 6 | 10 | 10 |

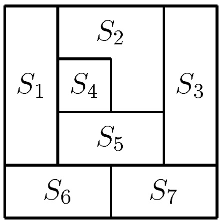



Figure 5.3: example 1.

**Example 1**

We list the criterion values for Beaulieu & Goldberg, Tilton and, our algorithm.

| pairs | | HSWO | IPRG | SOIM |
|---|---|---|---|---|
| $S_1$ | $S_2$ | 1.5 | 0.005 | $\boxed{1}$ |
| $S_1$ | $S_4$ | 60.7 | 0.193 | 9 |
| $S_1$ | $S_5$ | 4.8 | 0.015 | 2 |
| $S_1$ | $S_6$ | 30.0 | 0.095 | 5 |
| $S_2$ | $S_3$ | 181.5 | 0.576 | 11 |
| $S_2$ | $S_4$ | 48.0 | 0.152 | 8 |
| $S_2$ | $S_5$ | $\boxed{1.2}$ | $\boxed{0.004}$ | $\boxed{1}$ |
| $S_3$ | $S_5$ | 120.0 | 0.381 | 10 |
| $S_3$ | $S_7$ | 10.8 | 0.034 | 3 |
| $S_4$ | $S_5$ | 32.7 | 0.104 | 7 |
| $S_5$ | $S_6$ | 9.0 | 0.029 | 3 |
| $S_5$ | $S_7$ | 49.0 | 0.156 | 7 |
| $S_6$ | $S_7$ | 16.0 | 0.051 | 4 |

Table 5.2: merging costs for the 3 algorithms for example 1 for one iteration.

As can be seen from table 5.2 all three criteria result in the same pair with the lowest dissimilarity value (Tilton) and similarity value (Beaulieu & Goldberg) (pair $\langle 2,5\rangle$). Our method also returns the pair $\langle 1,2\rangle$. This means that the pairs

| | | $S_i$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 1 | 1/2 | 2/3 | 3/4 | 4/5 | 5/6 | 6/7 | 7/8 | 8/9 | 9/10 |
| | 2 | | 1 | 6/5 | 8/6 | 10/7 | 12/8 | 14/9 | 16/10 | 18/11 |
| | 3 | | | 3/2 | 12/7 | 15/8 | 2 | 21/10 | 24/11 | 27/12 |
| | 4 | | | | 2 | 20/9 | 24/10 | 28/11 | 32/12 | 36/13 |
| $S_j$ | 5 | | | | | 5/2 | 30/11 | 35/12 | 40/13 | 45/14 |
| | 6 | | | | | | 3 | 42/13 | 48/14 | 54/15 |
| | 7 | | | | | | | 7/2 | 56/15 | 63/16 |
| | 8 | | | | | | | | 4 | 72/17 |
| | 9 | | | | | | | | | 81/18 |

Table 5.3: weighting factor for Beaulieu & Goldberg $\frac{(N_i \cdot N_j)}{(N_i+N_j)}$ for two regions $S_i$ and $S_j$. On the x-axis the segment size for $S_i$, on the y-axis the segment size for $S_j$.

$\langle 1,2\rangle$ and $\langle 2,5\rangle$ must be merged simultaneously in order to compose the new region $\langle 1,2,5\rangle$. The new region mean for the new region $\langle 1,2,5\rangle$ will be:

$$\frac{3\cdot 1+3\cdot 2+2\cdot 3}{3+3+2}=1.875 \tag{5.27}$$

The mean value for the pairs $\langle 2,5\rangle$ and $\langle 1,2\rangle$ for both Tilton and Beaulieu & Goldberg will be also 1.875.

Another interesting fact is that Beaulieu & Goldberg's criterion favours the merging of smaller regions in case of equal difference in mean of two regions.

Table 5.3 shows the values of weighting factor $\frac{N_i \cdot N_j}{N_i+N_j}$ for segment sizes of regions $S_i$ and $S_j$ between 1 and 9. In case of ambiguity of two pairs of regions with the same difference in spectral value, smaller regions are favoured for merging over larger regions. Also the merging of a small region with a large region is favoured over the merging of two regions with a comparable but smaller size.

Tilton's criterion as given in equation 5.4 also favours the merging of smaller regions.

**Example 2**

In this example we have changed the pixel values in such a way that both Tilton as Beaulieu & Goldberg give the same similarity values for different pairs.

From table 5.4 it follows that the pairs $\langle 1,2\rangle$, $\langle 2,5\rangle$ and $\langle 5,6\rangle$ are all three candidates for merging for all three authors. For SOIM it is clear, all three pairs are merged into a new region $\langle 1,2,5,6\rangle$. For the HSWO and the IPRG algorithms randomly selecting a pair creates a different hierarchy for both cases, and order independence is violated.

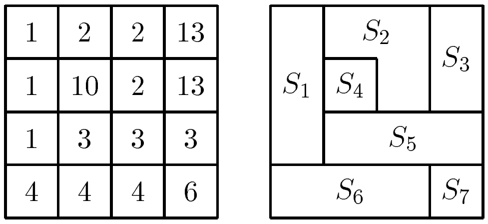



Figure 5.4: example 2.

| pairs | | HSWO | IPRG | SOIM |
|---|---|---|---|---|
| $S_1$ | $S_2$ | $\boxed{1.5}$ | $\boxed{0.007}$ | $\boxed{1}$ |
| $S_1$ | $S_4$ | 60.7 | 0.270 | 9 |
| $S_1$ | $S_5$ | 6 | 0.027 | 2 |
| $S_1$ | $S_6$ | 13.5 | 0.060 | 3 |
| $S_2$ | $S_3$ | 181.5 | 0.645 | 11 |
| $S_2$ | $S_4$ | 48.0 | 0.213 | 8 |
| $S_2$ | $S_5$ | $\boxed{1.5}$ | $\boxed{0.007}$ | $\boxed{1}$ |
| $S_3$ | $S_5$ | 120.0 | 0.533 | 10 |
| $S_4$ | $S_5$ | 36.7 | 0.163 | 7 |
| $S_5$ | $S_6$ | $\boxed{1.5}$ | $\boxed{0.007}$ | $\boxed{1}$ |
| $S_5$ | $S_7$ | 6.7 | 0.030 | 3 |
| $S_6$ | $S_7$ | 3 | 0.013 | 2 |

Table 5.4: merging costs for the 3 algorithms for example 2 for one iteration.

## 5.9 Merging isolated pixels

After merging, isolated pixels still remain within the image. These isolated pixels have spectral values significantly different from their surrounding neighbouring segments. This means that adding them to the closest neighbour and updating the segment parameters would weaken the constraint of computing 'pure' regions. In order to overcome the above, three rules (see algorithm 5.5) have been developed which try to add isolated pixels to existing regions, but without updating the segment mean [52]. Of course more rules could be developed and implemented. Note: in the rules of algorithm 5.5 a "segment" means a segment with size greater than 1. Adding pixels to segments diminishes the number of isolated pixels and the output image will be less noisy.

In figure 5.5 an example is given for every rule. For rule 1, $P$ is 4-connected to the four segments A,B,C,D. Assuming that $P$ is spectrally closest to A, then $P$ will be added to that segment. For rule 2, $P$ is at 3 places a 4-connected neighbour of segment A. Therefore $P$ is added to segment A. For rule 3, $P$ and $Q$ are 4-connected and $P$ is 4-connected with the segments A,B,C. Assuming that $P$ is closest to segment A, $P$ will be added to that segment.

1. **IF** $P$ `is surrounded by four different segments`
   **THEN** `assign` $P$ `to the spectrally nearest segment`
2. **IF** $P$ `is surrounded by three pixels belonging to one segment`
   **THEN** `assign` $P$ `to this segment`
3. **IF** $P$ `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 5.5: rules for merging isolated pixels based on 4-connectedness.

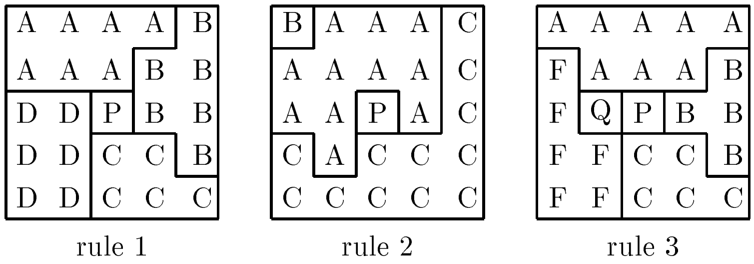



Figure 5.5: examples of the three context rules. P and Q are isolated pixels and A,B,C,D,F are regions with size > 1.

## 5.10 Disadvantages of SOIM

A disadvantage of the simultaneous merging however might be that the criterion as set by Beaulieu & Goldberg in minimizing the merging cost might be violated. The cost of merging regions A,B, and C in case of equal cost for (A,B) and (B,C), simultaneously, might be higher than separate merging of first A and B and then (in a later step) B and C.

### 5.10.1 Merging effects of SOIM

Due to the consistent simultaneous merging at every iteration, paths of connected pairs with the same cost function might be constructed.

A nice example for which this is the case is the following: an image consisting of 1 line by 256 columns, where the pixels in column 0 have spectral value zero, the pixels in column 1 have spectral value one, etc. For our algorithm this image generates the end result after only one iteration and merges all the pixels into one segment

Example 2 in the previous section also shows the consistent simultaneous merging. For SOIM, in the first iteration, immediately the regions $S_1$, $S_2$, $S_5$, and $S_6$ are merged into one new region, whereas for both Tilton and Beaulieu & Goldberg a pair is selected randomly.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | • • • | 253 | 254 | 255 |

Figure 5.6: an example of an image ramp.

Tilton, however, is concerned that with our algorithm, image data ramps as shown in figure 5.6 will be inappropriately merged[7]. However from experiments with remotely sensed imagery this "inappropriate" merging does not appear to occur very frequently.

## 5.11 Data structures used for HSWO and SOIM

### 5.11.1 Beaulieu's implementation of the HSWO algorithm

In [6] Beaulieu describes the implementation of the HSWO algorithm. Recalculation of the similarity value for all adjacent pairs can be avoided by updating only the values that are modified by a segment merger.

Referring to algorithm 5.3 Beaulieu uses a descriptive data structure $D_i$ containing region information of $S_i$, i.e. the segment mean and size. The criterion $C_{i,j}$ is calculated from $D_i$ and $D_j$. Each region $S_i$ also has a neighbour list $B_i$. For each region $S_i$ all its neighbours are examined, in order to find the one with the lowest criterion value, the best neighbour. The lowest criterion value is stored in a criterion list $Cr(i)$. A tree structure is employed to find the lowest value of the criterion list. Thus data structures are employed to organize image data and to reduce computing time, which is obtained by storing the parameters $D_i$ which don't needed to be re-computed each time information about segment $S_i$ is requested.

### 5.11.2 About the implementation of SOIM

In the current version of SOIM a data structure (known as cell) is used for bookkeeping information about a region. Every cell contains the mean spectral value of that region, its size, its members, and a flag if it is the main cell of a region. Knowing the number of channels in the image, memory can be allocated dynamically for the mean spectral values of the channels. The data structure uses separate arrays (i.e. for size/memberof, the mean and the members). Furthermore two compile time switches are created. While compiling the program two parameters can be set, determining the functionality of the program. One switch is for storing the bounding box of a cell or storing the members itself, and

[7]after a private communication, July, 1994.

the other one is for storing the neighbours or re-computing these every time they are needed. In the version used the members are stored and the neighbours are re-calculated every time they are needed (see below).

The current version of the best merge looks at every iteration the set of all pairs of regions for which the value of the cost function $C_{ij}$ is equal to the minimum value of all $C_{ij}$'s. It is assumed that the cost function is symmetric, i.e. $C_{ij} = C_{ji}$. This way it is sufficient to compare pairs $(i, j)$ for which $i > j$.

The two different implementations described in [100] differ in the structure containing the distance information. In the first implementation the data structure contains the distance, main cell, and the neighbour for every distance. In the second implementation however the variable "neighbour" is omitted and is computed every time it is needed. This means that for the first implementation 12 bytes are needed and for the second 8 bytes for each distance.

A significant difference of our implementation with Beaulieu's [6] implementation is that no neighbour list is maintained and this results in saving of memory.

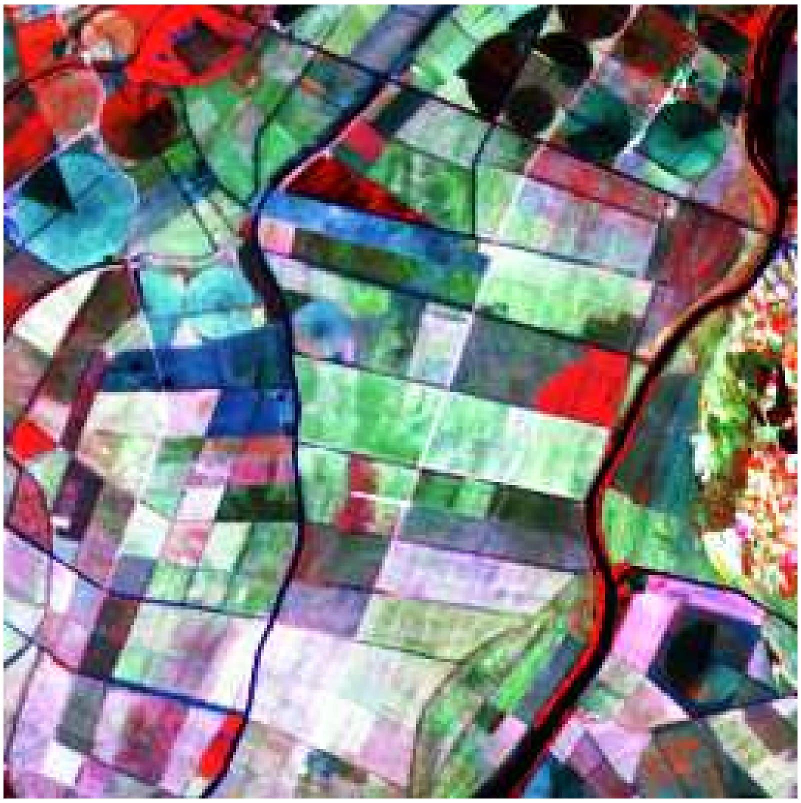


Image 5.1: Landsat-TM, June 24, 1991, $256 \times 256$ zoomed part of Lisbon area image, Portugal, 6 channels. For display, R,G,B = channels 4,5,3, Original satellite data ©Eurimage.

Every region's neighbours are computed every time they are needed for the pair of regions to be merged. This can slow down the algorithm, but updating the neighbour lists is also time consuming. Given a pair $(R_a, R_b)$ of adjacent regions to be merged, first the neighbours of $R_a$ and then of $R_b$ are calculated.

The pixels in the image are numbered from left to right and from top to bottom. This means that the (x,y) coordinates of a pixel can be easily determined. From the members (pixels) of $R_a$ we can compute the minimum and maximum values in $x$-direction as well as in the $y$-direction (4-connected), i.e. (minx,maxx) and (miny,maxy). Now fit a window of (minx-1,maxx+1) and (miny-1, maxy+1) and then check for every pixel within the window the pixel's membership of $R_a$. If it does not belong to $R_a$ and is not a member of $R_b$ and has a 4-connected member which is an element of $R_a$, then store its membership value in $R_a$'s neighbour list. The same is done for $R_b$.

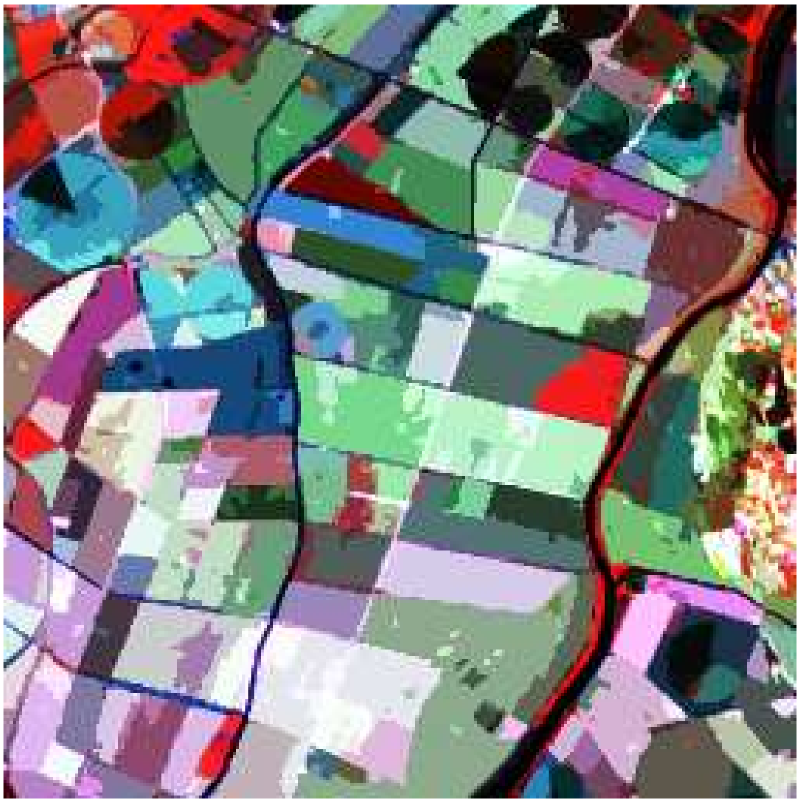


Image 5.2: image 5.1 merged with threshold 6 (R,G,B = channel 4,5,3).

## 5.12 Results

The region growing process has been tested on various images such as the agricultural area in the vicinity of Lisbon, Portugal, shown in image 5.1. This image is a $256 \times 256$ part of the Lisbon area, Landsat-TM scene of June 24, 1991. For displaying the image the RGB[8] mapping is on channels 4,5,3 respectively.

Easily recognizable are the irrigation areas (the circles) within the image. Furthermore a farm track is easily visible in white and vertically cuts the image more or less into two parts. Image 5.2 is the result of region growing with SOIM with a Euclidean spectral threshold of 6. All the fine structures are still visible

[8]Red Green Blue

but there are also many isolated pixels. Image 5.3 is the result after merging with a spectral threshold of 15. This image shows an example of over-growing of regions. The farm track has completely disappeared and many separate fields (possibly cultivated with the same crop) are merged into one field.

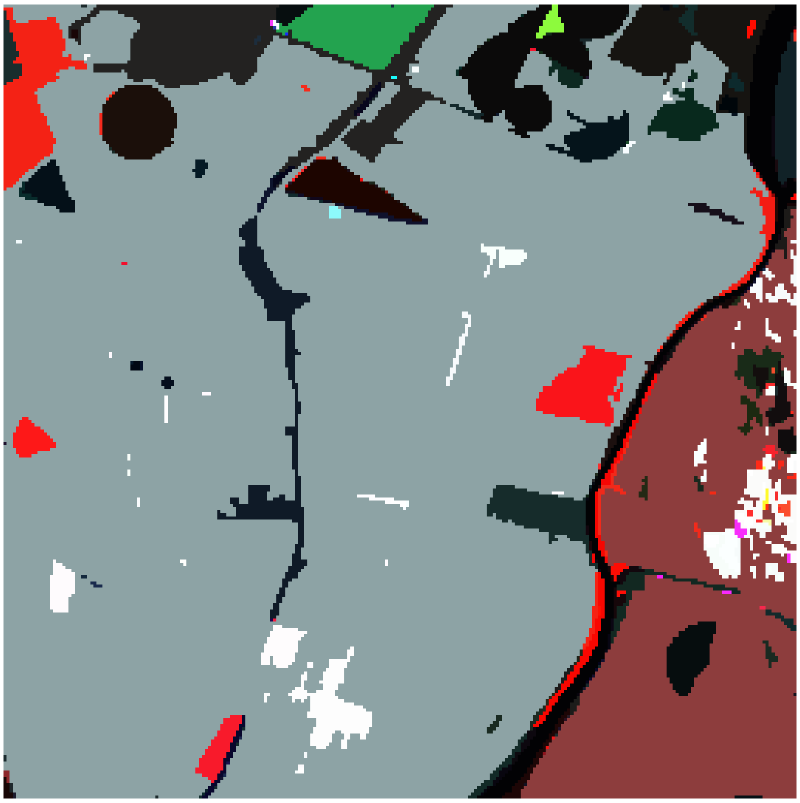


Image 5.3: image 5.1 merged with threshold 15 (R,G,B = channel 4,5,3).

## 5.13 Comparison of results with the IPRG

To make a comparison between the segmentation results of both SOIM and IPRG possible, Tilton implemented our cost function (equation 5.25) in the IPRG. The results of using IPRG in combination with equation 5.25 are given in images 5.4 and 5.5. As can be seen his results are very similar with our results for the image region grown with threshold 6. For both algorithms with threshold 15 it can be seen that most of the image is grown into one region. Remarkably structures detected with SOIM are not detected with IPRG and vice versa. This could be due to the fact that Tilton performs his merging within the separate sub-images in parallel.

As Tilton performs a best merge within every sub-image it is not necessary to merge regions with the lowest merging cost in the entire image. The pair merged within one sub-image can have a merging cost much higher than the pair having the lowest cost within another sub-image.

Our criterion to merge only pairs with the lowest costs reduces the set of pairs drastically. Therefore it can be concluded that our merging scheme effectively does mutual best pairwise merging, because the occurrence of non mutual best pairwise merging is rare.

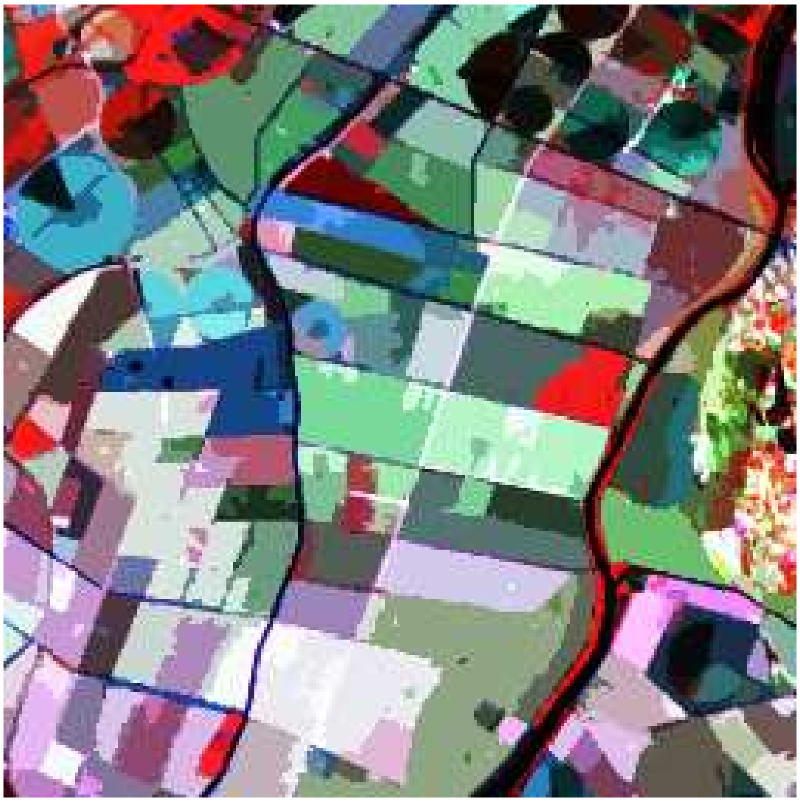


Image 5.4: image 5.1 merged with IPRG with threshold 6, cost function used equation 5.25 (R,G,B channel 4,5,3.).

## 5.14 Concluding remarks

In this chapter it has been shown that the order independence aim as set by Tilton and Beaulieu & Goldberg is violated in case of ambiguity in minimum similarity criteria values between pairs of regions for their algorithms. In this case a randomly selected pair is merged. In order to overcome this weakness it is proposed in case of ambiguity to merge these pairs in parallel.

The main question to be answered however, remains: How does the consistent iterative simultaneous merging influence the end result? On the images thus far merged with SOIM the above problem seemed to have no visible influence on the final result. But for the image ramp as proposed in section 5.10.1 it seemes better to randomly select a pair of regions each iteration instead of merging all the pairs simultaneously into one region at the first iteration.

Furthermore a data structure is proposed which contains all the region information needed, but which minimizes memory usage initially. Furthermore two different implementations for minimizing the memory needs for the distances are

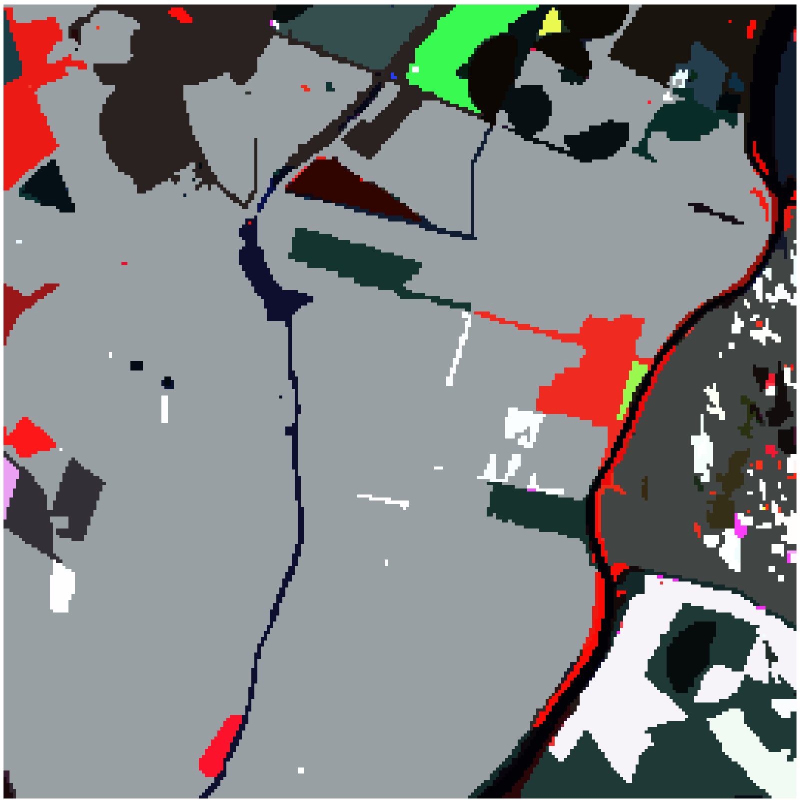


Image 5.5: image 5.1 merged with IPRG with threshold 15, cost function used equation 5.25 (R,G,B channel 4,5,3.).

given. A saving of 33% is gained but it must be noted that the second implementation is significantly slower.

Our stopping criterion is a threshold on the cost function which depends only on the spectral values of the pixels and not on the region size. This is in contrast to Beaulieu & Goldberg and Tilton where the region size influences the cost function. For their case however, the problem is that a priori knowledge of the image to be processed is required. A problem for our case however, is the setting of the threshold. It is difficult to set a good merging threshold and further research on this topic is merited.

# Chapter 6

# System Integration and Computing Requirements

## 6.1 Introduction

Landsat-TM full scenes occupy about 250Mb. Keeping full scene imagery in main memory is (until now) for most UNIX based workstations, due to the amount of RAM available, impossible. Furthermore also memory is needed for storing intermediate results (often 3-5 times the size of the input image). Because of this large memory need, 'intelligent' algorithms have to be developed which keeps that part of the image which is needed in memory and the rest on disk. For this purpose 'slicing' and 'windowing' techniques are used. In these techniques a trade off is made between RAM usage and the number of disk I/O operations needed.

The segmentation method as defined in the former chapters, is adapted for processing full scene images using the slicing technique (if needed). The algorithm is implemented in such a way that processing on a one CPU machine (a sequential machine), from the computation time point of view, becomes feasible. The slicing technique is used for all parts except for edge following and best merge.

## 6.2 The slicing technique and windowing technique

Data is expected to be in BIL[1] format. In the description of the slicing technique we use the following parameters: $N$ : the number of lines of the scene, $M$ : the number of columns of the scene, $C$ : the number of channels. The number of pixels within one image is therefore $N \times M \times C$.

### 6.2.1 The slicing technique

The slicing technique uses a sliding window, and its basic idea is shown in figure 6.2. A window is defined covering a certain number of lines $K$ with $K < N$. Here $K$ is set to 3. The window moves from the top of the image (line 1) to the bottom (line $N$). The image data under the window will be read into RAM,

[1]Band Inter-Leaved format, which indicates that data is registered in the following order, Line 1, band 1, line 1, band 2,..., line 1, band C; line 2, band 1, ... line 2, band C, line 3, band 1,..., line N, band C. Also other formats are possible, see e.g. Schowengerdt [99].

the data outside the window will be kept on disk. To have access to the data within the window a pointer array is defined. Every element of the array points to a separate line. Initially the first pointer points to the top-line of the window and the k-th pointer points to the bottom-line of the window. If the data to be processed is on the bottom part of the window (i.e. near line $K$), the top part, i.e. the first $L$ ($L < K$) lines, will be written to disk (i.e. the input/output file is updated). The image window will be moved $L$ lines towards line $N$ of the image. Within these $L$ lines new image data will be read. Thus information in the 'top' part of the window, i.e. the information just read, belongs to data towards the bottom of the image. To have the information in the window in the correct order, the pointers are rotated, so the current lines seem to be on the 'bot' part of the window. It is not important in which sequence the data is in the window, but it is important that the pointers are pointing to the correct lines within the buffer, as the data is always accessed via the buffers. The following example, figure 6.1 shows the principle; $L$ is set to one.

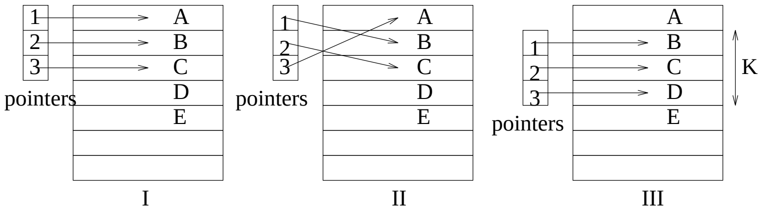



Figure 6.1: example of pointer rotation.

Situation I shows the initial situation. The lines processed and read into the buffer are lines A, B, and C. The pointers 1,2, and 3 are pointing to line A, B, and C respectively. After one iteration line A is not needed anymore, but line D is instead. The pointers are rotated so that pointer 1 points to line B, pointer 2 points to line C and pointer 3 points to line A (situation II). Line A (pointer 3) is written to disk, and line D is read into the buffer pointed to by pointer 3 (situation III), and the window is advanced by one line. The processing as described above is done on a per-channel basis, i.e all the channels are done sequentially.

This technique is used for the filtering and for the edge magnitude and direction calculation.

### 6.2.2 The windowing technique

The principle of windowing is displayed in figure 6.2. Two window are displayed, the current window (size $K$ lines), and the new window (size $K$ lines). If the current line to be processed comes within a certain range of bot (top) in figure 6.2, then the window is shifted $K/2$ lines down (up). The upper (lower) half of the window is written to disk, and the lower (upper) half of the window is copied to the upper (lower) half of the window. The new $K/2$ part of the image within the window is read in the lower (upper) part of the window. As can be seen in the figure no pointers are present, and only two variables 'top' and 'bot' are kept to remember the position of the window in the image. The size of the window can be given as an input parameter at program launch.

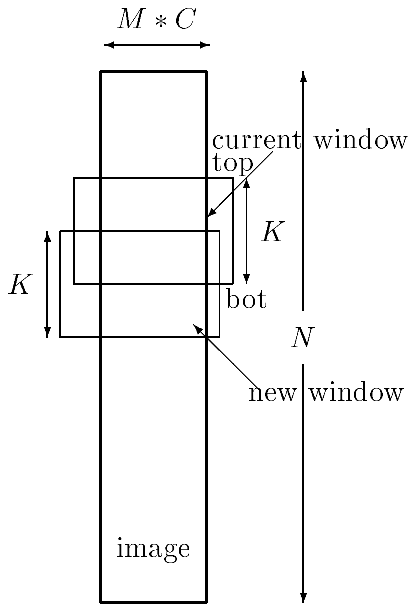



Figure 6.2: the windows over the image.

## 6.3 Data formats and utility programs

Images are stored on disk in the Erdas format [29], versions 7.2 and 7.4 and files written on SUN and VAX machines are supported by the programs. This is that files are in BIL format preceeded with a header of 128 bytes. The header for the Erdas 7.2 version is given in C below:

```
/* erdas 7.2 format */

typedef struct lan_header LanHeader;
typedef struct lan_header {
 Byte        hdword[6];   /* pos 1-6     */
 Ushort      ipack;       /* pos 7-8     */
 Ushort      nbands;      /* pos 9-10    */
 Byte        unused1[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 */
} Header;
```

The variable `hdword` is contains the Erdas-version, where `ipack` indicates the number of bits per pixel, `nbands` gives the number of channels in the image, `cols` and `rows` give the number of columns and lines of the image. The other variables are not used in the implementations.

Region information is stored in the developed REGINF format, consisting of a header followed by data for each region. The header is given in the C data structure below:

```
struct infheader {
  Byte hdword[6];
  Ushort byteorder;
  Ushort rows;
  Ushort cols;
  Ushort channels;
  Ushort accuracy;
  int numregions;
  Ushort spare[6];
} infheader;
```

The variable `accuracy` is the same as the variable `ipack` in the previous datastructure, where `numregions` gives the total number of regions in the image.

The data per region consist of the average intensity values of the region and the coordinates of the pixels belonging to the region. The coordinates are stored in Run Length Encoded[2] (RLE) format to save disk space.

Furthermore software utility programs have been developed which extract region information from the REGINF files and which convert files in REGINF into Erdas format.

Public domain software 'XV' has been adapted so that Erdas format can be read and displayed.

A shell script has been written which performs the separate programs of the chain sequentially, with default parameter settings.

## 6.4 Filtering complete images

As the filter as defined in appendix A needs a $3 \times 3$ window, only three lines at a time are used for computation and thus $K$ can be set to three. While filtering (a channel of) line $i$, lines $i$-1 and $i$+1 are needed. If line $i$ has been filtered, then line $i$-1 is not needed anymore and can be written to disk. Because of the rotation of the pointers, the current line becomes line $i$+1 and line $i$+2 has to be read from memory.

## 6.5 Edge detection on large images

The slicing method for computation of the edge values, threshold and subsequent LBE values is in principle the same as for the filter. A problem arises however when the actual edge following has to take place. Edges may appear as chains through the whole image, and thus the slicing technique has to be adapted.

The computation of the edges over the image requires four stages. In the first stage the edge magnitude and edge direction are computed. Furthermore the edge magnitude minimum and maximum are computed. In the second stage

[2] In RLE format the pixel coordinates are written together with the number of pixels having the same intensity values in x (or y)-direction.

using the magnitude minimum and maximum, the edge values are scaled into the interval $[0, 255]$ and a threshold is computed. In the third stage the LBE values are computed using the edge magnitudes, edge directions and the threshold. Edge following takes place from the fourth stage and is more difficult than the previous stages. The problem is that edge fragments may form chains over the whole image. In order to overcome this problem, while following, the windowing technique as described in section 6.2.2 is used, as the following moves along the current contour. If the contour following is terminated for the pixel from which it started, the window is reset upon that pixel and the scanning of the image for a un-marked pixel continues.

In the implementation of the sliding window used here, the window is set up depending on the computer system used, and there is a trade off between the number of users on the system, and RAM available.

The calculation of the edge magnitudes gives values in the real domain. In the current version of the edge detection module, in order to save disk space, the output values are scaled so that they fit in two bytes.

Every stage needs information from earlier stages so no information can be thrown away. This means that some intermediate files (shown in figure 6.3) must be kept on disk in order to be available as input for later stages.

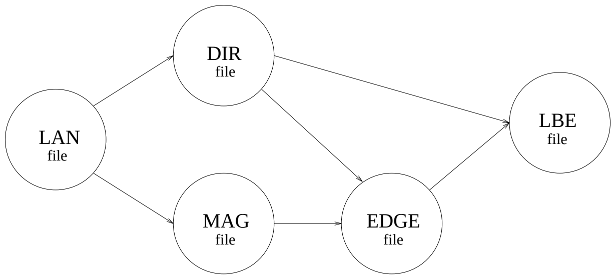



Figure 6.3: data flow for edge detection.

The edge detection method outputs a number of files.

1. the magnitude file (MAG, 2 bytes per pixel)
2. the edge file (EDGE, 1 byte per pixel)
3. the direction file (DIR, 1 byte per pixel)
4. the LBE file (LBE, 1 byte per pixel)

This can cause problems due to the huge memory needs. For a $6000 \times 6000$ image (1 channel), $36 \times 5$Mb is needed. But the magnitude file may be deleted after the edge file is made and before the LBE file is created (for LBE computation only the direction and edge file are needed, see figure 6.3).

## 6.6 Edge linking on large images

The sliding window technique is not used because there could be intensive bookkeeping, e.g. reading the image, reading/writing existing trees from/to disk. As stated in chapter 4 a node can have only three sons, resulting in the following ‘C’ data structure, and thus occupying (2+2+4+4+4) bytes.

```
typedef struct node *Gnode;
typedef struct node {
        short x;
        short y;
        Gnode next1;
        Gnode next2;
        Gnode next3;
        } Graph.
```

In order to overcome the exhaustive bookkeeping, an approach has been chosen which is straightforward. In the current version, the image is divided into squares with size $X$ and on each sub-image independently edge linking, and closed polygon detection, as shown in figure 6.4 are performed. As can be seen from the figure, sub-images on the right and the bottom need not necessarily be of size $X$. The edge (LBE) input image contains two pixels along the border which do

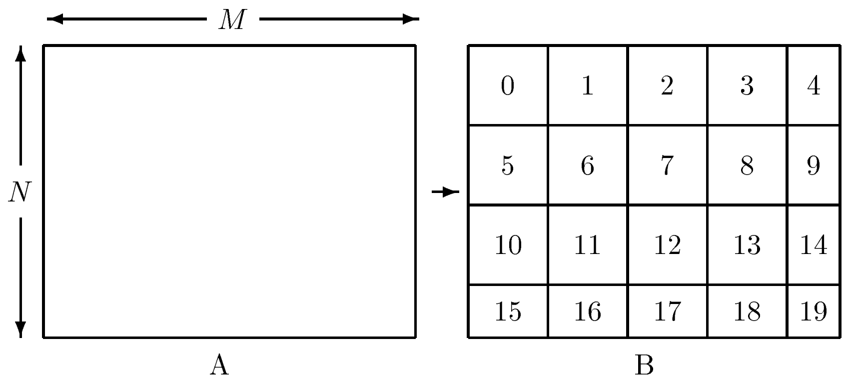



Figure 6.4: image division into $X \times X$ square sub-images.

not contain any information at all, i.e. the pixels all have value 0. For a correct operation of the algorithm these pixels are removed. The image processed thus contains $(N-4) \times (M-4)$ pixels. When the output image is reconstructed these pixels are added and the image will have its original size again [111].

A weak point of this implementation is that between the borders of two sub-images no pixel addition and thus no edge fragment connection takes place. This might lead to the loss of information (i.e. polygons). At the border it is checked only if two pixels within two different sub-images are connected and thus may not be pruned. After every sub-image has been ‘edge linked’ the pixels on the border have to be checked to see if they are edge valid pixels or not. Valid edge pixels have to remain within the sub-image and non-valid pixels have to be pruned.

The ‘parallel’ version consists of a master process which starts up several sub-processes. The master is in charge of the slaves and prepares the input for the slaves. The separate actions undertaken by the master and the slaves are given in algorithm 6.1. (In the current version the slaves are started on the same machine. To overcome resource competition (causing swapping etc.) only two

```
1. (MASTER)
   Divide image into NR sub-images.
   For every sub-image:  get borderpixels.
   Start slaves up for pruning and connection.

2. (SLAVES)
   Start connection and pruning on sub-images.

3. (MASTER)
   wait until all slaves ready.

4. (MASTER)
   while borderpixels have changed:
   (SLAVES)
   pruning of sub-image.

5. (MASTER)
   wait until all slaves ready.

6. (MASTER)
   reconstruct image.
```

Algorithm 6.1: division of the image data and activities over the master and the slaves for the edge linking module.

slaves are allowed at the same time.) This can be overcome by starting the slaves on different machines (which can be implemented easily).

First the image border (2 pixels wide) is removed and the image is divided into sub-images. For every sub-image the border pixels are determined in the following way (see figure 6.5), starting from the north-west corner. For sub-

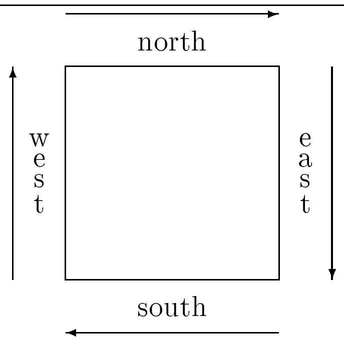



Figure 6.5: scanning direction border pixels.

images situated in the original image on the periphery, border pixels are not determined. Thus if the north side of a sub-image is originally on the northern periphery then its north border pixels are not determined. The border pixels are saved for each sub-image in separate files. For every sub-image a separate process is started performing edge linking and pruning (border pixels are not pruned). When a slave is ready it signals the master it is ready. The master waits until all the slaves are ready. Whenever, due to an error, a slave process is not able to terminate in a correct way, the master will wait forever. Because of the error the slave is not able to 'signal' the master that it is ready and the master will keep waiting for the slave to 'tell' him it is ready. The master checks for every sub-image the validity of the border pixels. Valid means that every border pixel is checked if it has an 8-connected neighbour in its corresponding neighbouring sub-image, and that the border pixel will correspond to one of the templates A,B,C, or D (rotated in all eight directions of course) of figure 4.6. The corner points are special cases, neighbour pixels have to be fetched from three neighbour sub-images, instead of one.

The method of master-slave communication described above, is very simple. In practice it would be better to have the master asking the slaves if they are ready. If after a certain time the slave does not reply, the master would terminate the slave's process and return while supplying a message to the user.

## 6.7 Region growing on complete scenes

By using the closed polygons found by the edge linking process, the image is divided into a set of independent areas, where it is assumed the pixels (from different closed polygons) do not have any physical relationship. Doing so reduces the problem of best merging a complete image into best merging an independent set of areas, where every area has a size $\ll N \times M$.

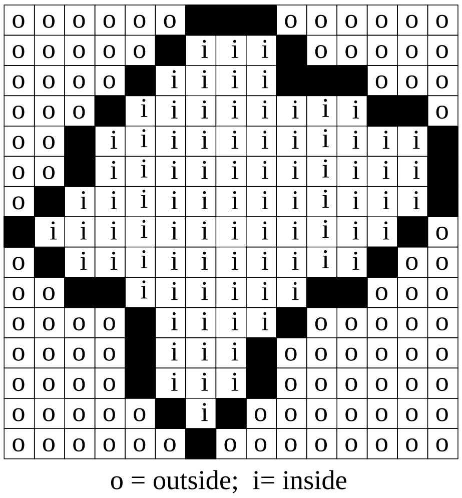



Figure 6.6: bounding box around a region.

In order to process only pixels within a closed polygon, pixels outside that polygon are labelled as being "outside". Only pixels inside the polygons are then considered for merging. This implies that the SOIM algorithm has to be adapted. A bounding box is put around the polygon, as shown in figure 6.6. Pixels outside the polygon are labelled 'o' and pixels inside the polygon are labelled 'i'. Edge (polygon) pixels are black. All the 'i'- pixels are initially regions and the merging cost for all the adjacent pairs can be computed. If for a certain region one of its neighbour pixels is an edge pixel then the edge pixel will be excluded from the merging process, and therefore not considered as a neighbour.

The region growing within a polygon works as given in algorithm 6.2. One problem however, remains in the current implementation. If a polygon exists covering a large area, due to the memory needed to store a pixel, the total number of pixels within the bounding box might require an amount of RAM which is larger than the RAM available. Then the program 'collapses', i.e. a message is given to the user informing him that the amount of memory needed for the program is larger than the amount available, and execution is stopped.

```
1. Read the polygon

2. From the polygon pixels, compute the minimum and maximum
   x and in y-directions, which define a bounding box.

3. Read the image for this bounding box.

4. Mark the polygon pixels as edge pixels (label EDGE)
   in the bounding box.

5. Perform a floodfill [101] in order to determine the pixels
   which are within the polygon.  Mark the pixels outside the
   polygon as OUTSIDE.

6. Perform the SOIM on the pixels not marked (EDGE, OUTSIDE)

7. Write the pixels within the polygon to the output file,
   i.e.  edge pixels and outside pixels are not written.
   This is done in order to prevent already merged data
   being overwritten.
```

Algorithm 6.2: applying the best merge within closed polygons.

As stated in Haralick and Shapiro [39] algorithm 6.2 does the obvious thing. Region growing is done only for non-edge pixels, i.e. the growing process is not allowed to merge pixels outside a closed polygon to the region. Edge detection and region growing are combined in such a way that advantage is taken of their different strengths. The strength of edge detection is that boundaries are placed in a spatially accurate way. Its weakness is that not all edges are connected into closed contours and therefore too large regions may result. The strength of the region growing algorithm is its ability to place boundaries in weak gradient areas.

A disadvantage of this method is that edge pixels are not considered in the region growing process and thus remain isolated. Because of the ground resolution of the satellites used (typically 10-30m) this would lead to loss of information. Pixels seen as an edge pixel by the algorithm can belong in reality to a field. Considering these pixels as separate regions (i.e. isolated pixels, because no region growing has been applied), without examining if they belong to neighbouring regions, could influence the accuracy of land cover inventories.

This could be solved by assigning 'edge' pixels fractionally to neighbouring segments on the basis of a spectral linear mixture model or else leaving them as linear/edge features if they have a signature incompatible with a linear mixture of their neighbours. The former can result in multiple segment membership for one pixel. The latter creates separate entities for pixels which can be classified as roads, rivers etc. Ideally edge pixels should be assigned fractionally to all classes which they mix, though this may be difficult in all but the simplest cases.

A method which adds these edges is described in Schouten [97], see the next section.

### 6.7.1 Adding edges to regions

In the current version of the segmentation method edges are not added to the regions. In the following a method is described which adds edges and isolated pixels to regions.

There are three passes made over the image to assign isolated pixels to a region. A pixel can be isolated because it was an edge pixel and was not considered in "bmerge" or because "bmerge" has put it into a region containing only that pixel. In the latter case that region is effectively deleted.

In each pass a serial scan over the image is made and each encountered isolated pixel $p$ is compared with each of its 8 neighbour pixels using the cost function of "bmerge". Let $n$ be the neigbouring pixel with minimal cost value $C_n$. Then the following actions are taken ($T$ is the input threshold parameter to "addedge"), see algorithm 6.3. After "addedge" all pixels are assigned to regions (some of

```
in pass 1:
  IF (C_n ≤ T and n belongs to a region with more than 1 pixel)
    add pixel p to the region to which n belongs
  ELSE
    donothing

in pass 2:
  IF (C_n ≤ T * √(2))
    IF (n belongs to a region with more than 1 pixel)
      add pixel p to the region to which n belongs
    ELSE
      create a new region containing pixels p and n
  ELSE
    donothing

in pass 3:
  IF (C_n ≤ T * 2)
    IF (n belongs to a region with more than 1 pixel)
      add pixel p to the region to which n belongs
    ELSE
      create a new region containing pixels p and n
  ELSE
    create a new region containing only pixel p

Then the scan over the image is continued.
```

Algorithm 6.3: adding edges and isolated pixels to region.

them with one 1 pixel)

Pixels on the border of the image ( the first 2 and last 2 pixels on a scanline and the first 2 and last 2 scanlines) are never assigned to regions.

An example of this program applied to the zoomed Lisbon area image, image 5.1 is given in image 6.1. Compare this image with image 5.3, the segmented zoomed Lisbon area image, without using polygons.

Of course this is an initial attempt in order to add isolated pixels (and thus edge pixels) to already existing regions (size larger than one). Further study is required in order to get an optimal edge to region assignment.

Image 6.1: Segmented image 5.1, image sj256.lan with polygons and edge addition, R,G,B channel 4,5,3, threshold for merging set to 15.

## 6.8 About a parallel implementation

The segmentation method as defined, has been developed so that a parallel implementation is relatively straightforward.

The slicing technique as used in the filter and the edge detection can be easily replaced by dividing the image into parts, where every part is processed separately on a different processor.

### 6.8.1 Two parallel implementations of the pre-segmentation filter

For the filter (appendix A) a parallel implementation is given by Hendriks & Schreurs [41] on a T800 transputer network. Their implementation of the filter consists mainly of two parts. The master and the slaves. The master distributes the image over the slaves which perform the actual filtering on their received part of the image. The filtered parts are sent back to the master when the filtering process finishes. The master merges the filtered parts into the output image. Hendriks & Schreurs distinguish four strategies for implementation.

1. The memory available on one slave for storing one channel data is insufficient. If the whole image (i.e. all the channels) fits within RAM of all the slaves together, then every slave gets a part of the image. No further distribution is needed.

2. The memory available on one slave for storing one channel data is insufficient. The whole image (i.e. all the channels) does not fit within RAM of all the slaves together. If one channel data fits within RAM of all the slaves, then every slave gets a part of the channel.

3. The memory available on one slave for storing one channel data is insufficient. The whole image (i.e. all the channels) does not fit within RAM of all the slaves together. One channel data does not fit within RAM of all the slaves. This strategy has not been implemented. The program reports this to the user and terminates.

4. The memory available on one slave for storing one channel data is sufficient. The number of passes used is equal to the number of channels divided by the number of slaves. In each pass each slave filters a channel. If the number of channels is greater than the number of slaves an extra pass is made.

The master runs the number of passes calculated for every strategy, sending in every pass to each slave a part of the image. After all the slaves have received their part, the master starts his handler which is a function for communication with the slaves. The handler continues until all the slaves have returned their part of the image. The slaves have a handler similar to that of the master, only these are started as parallel processes on the slaves. The slaves go into a loop, receiving data, filtering data and sending it back to the master, until the master tells the slaves to stop.

If comparison is made between the sequential version of the filter and the parallel version, it is reported that for large images ($> 1000 \times 1000$ pixels), the parallel version is faster. For smaller images the sequential version is faster.

Furthermore they report that the topology[3] plays an important role in the communication overhead. It was found that a tertiary tree[4] worked well, but a pipeline[5] was less satisfactory.

A second parallel implementation of the filter on a network of Sun-Sparc stations is reported by Elkardoudi & Janssen [28]. They use a similar approach as Hendriks & Schreurs. They also conclude that a parallel implementation gives an increase in speed for large images.

[3] Topology describes the way the separate processors are inter-connected.

[4] A topology form in which every processor has three sons.

[5] A topology form in which all the processors are connected in a line.

### 6.8.2 The other modules

For the edge detection part an approach similar to the filter can be used. Only the edge following part (after the LBE calculation) needs special treatment and a method will have to be developed for that. While for edge linking the algorithm for a parallel implementation is already given in section 6.6.

The best merge method as described above is perfectly suited for implementation on a parallel computer. Giving control of the segmentation to a master process, the master supplies the slaves with different data (i.e. polygons). Every slave performs the SOIM independently on the data supplied by the master. If a slave is ready it signals the master which then reads back the merged data and writes it to disk, while in the mean time supplying the slave with new data (if available). Of course this requires an ordering of the polygons (i.e. sequence, routing, etc.).

## 6.9 About memory usage

The method is developed so that it runs on a workstation with a certain amount of memory. If the user knows the available memory then a runtime parameter can be set controlling the amount of data read into RAM and therefore diminishing the amount of swapping needed. This option is implemented for the edge detection (LBE-following) and already resulted in a decrease of CPU-time needed. Implicitly this option is also implemented in the edge linking modules for defining how large the sub-images can be (depending on the RAM available of course). Using small sub-images increases run time.

## 6.10 About speed

All the computations reported here were carried out on a Sun-Sparc 10-41 workstation with 64Mb of RAM. The CPU-times as given in the tables (see below) are acquired with the Unix '/usr/5bin/time' command. The data is on a disk mounted via 'NFS' to the workstation.

### 6.10.1 Segmentation of the Flevopolder image

In this section the performance of the full processing chain on the Flevopolder (NL) image (image 3.1, 500 lines, 500 columns and 3 channels) is shown. The amount of RAM required for storing is less than the amount available on our Sun and therefore the number of I/O operations is put to a minimum, except for the parallel edge linking where the number of I/O operations is higher due to the reading/writing of the sub-images from/to disk. Therefore it is useful to compare CPU-times, while this is less useful for the images not fitting in RAM and requiring the special algorithms as described before. However these are given to give an indication of the CPU requirements of the separate algorithms.

The Flevoland image can be used to show the difference between region growing within the closed polygons found by edge detection, and region growing alone applied to the full image. Furthermore the difference between edge linking without division into sub-images (i.e. the sequential version) and the parallel edge linking, can be illustrated.

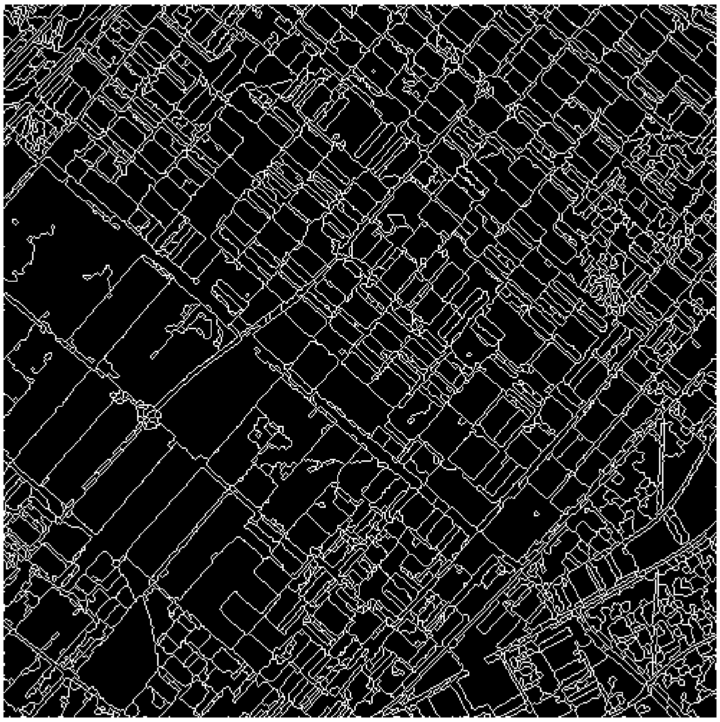


Image 6.2: edges detected using parallel edge linking in Flevopolder, The Netherlands, image, image 3.1, size of sub-images set to 170.

Image 3.5 shows the Flevopolder image after edge detection. Image 4.1 shows the edge detected Flevopolder image after sequential edge linking. The parallel edge linked version of the edge detected Flevopolder image, is shown in image 6.2. The size of the sub-images is set to 170. One can see that some edges have disappeared (left side of the image). This is due to the fact that on the borders of the sub-images no edge linking takes place.

| | elapsed time HH:MM:SS.S | CPU-time HH:MM:SS.S |
|---|---|---|
| edge detection | 53.3 | 12.9 |
| edge linking | 13.3 | 5.2 |
| parallel edge linking | 1:18.3 | 2.6 |
| polygon determination | 6.2 | 0.5 |
| best merge with polygons | 21.8 | 10.9 |
| best merge without poygons | 58:25.6 | 57:59.6 |

Table 6.1: time elapsed and CPU-time needed for Flevopolder image.

The CPU-times needed for the separate modules and the elapsed times for the separate modules are shown in table 6.1. The elapsed time gives the overall time to complete the program. It should be noted that the elapsed time increases as more processes are running at the same time: as the CPU has to share its time between all the processes. The figures given here however refer to tests when the segmentation program was the only user program in execution on the machine. The CPU-time indicates the time needed by the CPU to complete the process. This time is not influenced by the fact that there are more processes running simultaneously. For both the parallel edge linking and the sequential edge linking the CPU-times are shown.

[It has to be noted that the parallel version has no optimized code. Many improvements (in I/O) could be be implemented.] For this experiment the size for the sub-images in the parallel version was set to 170 (i.e. the image is divided into 9 sub-images).

The time needed to complete the edge linking for the parallel version was significantly higher than for the sequential version. This is due to the initial division (i.e. I/O operations) and the border pixel checking. Furthermore every time a border cycle is carried out, the subimage and border pixels are read from disk. The low CPU-time for the parallel edge linking is probably the time needed by the master, as the slaves are started as separate programs and thus as separate processes. To complete one border cycle, 12 seconds were needed. The sub-processes are costly to start up. Three border cycles were needed to complete the edge linking. This means that the slaves have been started up $3 \times 9 = 27$ times. If nine different workstations were available, every subimage could be given to a different workstation and a decrease in computation time for completion of a border cycle could be expected with a factor of about 9. Even on five machines a decrease with a factor of about four is to be expected as two processes run on one machine.

If the size of the subimages is set to 85 then a little more detail disappears (image not shown). But the elapsed time to complete the program is about three times as long as for the version with the sub-image size set to 170. Six border cycles were needed, indicating that the slaves have been started up $6 \times 36 = 216$ times. It can thus be concluded that the sub-image size should be set as large as possible in order to complete execution as soon as possible.

This loss in time when a comparison is made between the sequential and the parallel version, is re-gained when performing the best merge within polygons or on the full image. The sum of the elapsed times for parallel linking and region growing within polygons was less than two minutes, whereas the elapsed time for the region growing alone was almost one hour. The threshold was set for both cases to 8.

The difference in performing region growing on the Flevopolder image by the two approaches is shown in images 6.3 (showing the Flevoland image after best merging inside the polygons resulting from sequential edge linking) and 6.4 (showing the effect of performing only best merge, i.e. without use of polygons found by edge detection). In image 6.4 can clearly be seen the effect of over-merging the regions.

Examination of image 6.3 indeed shows that performing region growing within a polygon finds different fields. This proves the assumption that boundaries not found by the edge detection process can be detected by the region growing process.

Image 6.3: segmented with best merge inside polygons [SPOT, three bands, Flevopolder 91, with merging threshold 8] (R,G,B=channels 1,2,3).

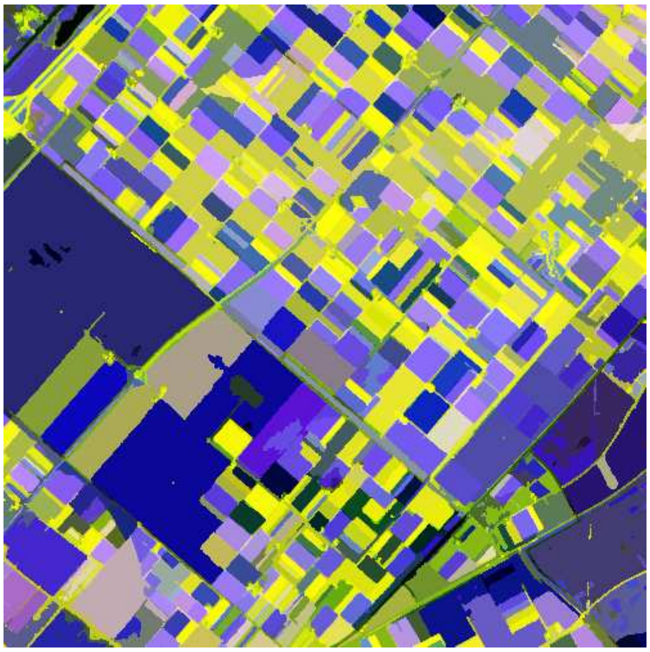


Image 6.4: segmented with best merge applied to the full image without polygons, [SPOT, three bands Flevopolder 91 with merging threshold 8] (R,G,B=channels 1,2,3).

A zoom of the lower right part is given in image 6.5. Here it is clearly visible that the region growing process found two separate regions in one polygon.

### 6.10.2 Segmentation of quarter scene Landsat-TM

The Landsat scene shown in image 6.6 shows the Lisbon area of Portugal taken on the 24-th of June, 1991. The image can be divided into a land area and a water area. The water area is spread within the image over 4 non-connected areas (i.e. different polygons). The dark bay area in the centre of the picture, is divided from the sea area, at the bottom left, by a bridge (which is detected by the edge detector), which is not visible in image 6.6. The third and fourth "water" areas can be seen at the bottom part of the image. Table 6.2 shows the difference in computation times when region growing was carried out for all polygons, and when it was carried out for the land area polygons only (since for land applications sea areas do not need to be segmented). Because the image size ($1953 \times 1801$ pixels) was larger than the user defined 800 pixels, the image had to be divided into parts.

After the initial pruning the border pixels of the sub-images were processed 6 times. A border pixel cycle needed about 2 minutes to complete. As can be seen from table 6.2, the region growing, using all the polygons (including the four water polygons) took about 59 hours, which is far too much for operational purposes. If the maximum polygon size is set to 75000 (i.e. excluding water from merging, the four polygons are given in table 6.3) then region growing on the remaining polygons only takes a few minutes. The four un-merged water

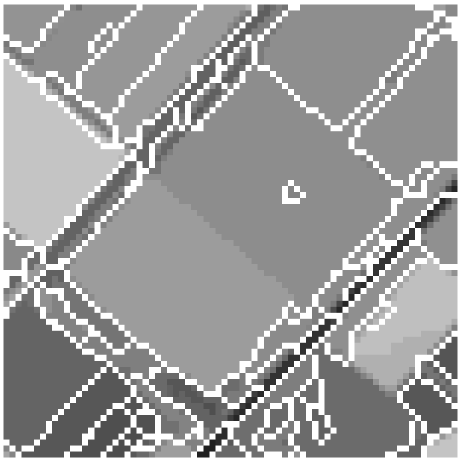


Image 6.5: zoomed (lower right corner, coordinates (x,y) (325,325), size 75 × 75 pixels) segmented with polygons Flevopolder 91 image (image 6.3, with merging threshold 8, (channel 3 displayed).

| | elapsed time HH:MM:SS.S | CPU-time HH:MM:SS.S |
|---|---:|---:|
| edge detection | 7:50.7 | 5:56.6 |
| parallel edge linking | 16:56.4 | 50.4 |
| polygon determination | 32.7 | 9.6 |
| best merge with polygons1 | 58:16:14.8 | 55:20:06.7 |
| best merge with polygons2 | 18:40.0 | 10:50.0 |

Table 6.2: time elapsed and CPU-time needed for Lisbon image; best merge with polygons1 indicates that for ALL the polygons a best merge has been carried out, with polygons2 indicates the times when the four largest polygons (> 75000 pixels) have been excluded.

polygons are considered as 4 separate regions, and the means of these polygons are taken as the new spectral values for every pixel within them.

### 6.10.3 Segmentation of SPOT Panchromatic

A SPOT panchromatic scene from August 10th, 1990 showing the Pelopónnisos area in Greece is shown in image 6.7. This is a another large image file with 6700 colums and 5980 rows of pixels. The image can be divided into an area where no data exists (spectral value 0, the wedges at the sides) and a scanned area which can be divided into a land area and a water area. The water area is composed of 2 un-connected zones (i.e. different polygons). In the images clouds can be recognized as well a few cities (i.e Piros - bottom left of the image and Amalias - to the left upwards from Piros). It is a mountainous area with many small fields.

The number of border cycles required is 9, where every border cycle required

Image 6.6: Landsat-TM image, $1953 \times 1801$ pixels, Lisbon, Portugal, titled 'subjune'. [Original satellite data ©Eurimage.]

about 21 minutes to complete. The two polygons excluded from the best merge had a size larger than 500000, see table 6.4. The threshold for merging was set to 4.

A zoom is made for display of a $100 \times 100$ pixel area. The original data is shown in image 6.8(a). Some small fields can be seen. The segmented result is shown in image 6.8(b).

In table 6.5 the merging is shown only for region growing on all except the two largest (water) areas.

## 6.11 Concluding remarks

In this chapter the segmentation method as presented in the former chapters has been integrated into a hybrid one capable of segmenting large scenes within a "reasonable" time. Segmentation results have been presented for different image

| polygon | #cells | win_n | win_w |
|---|---|---|---|
| 10993 | 1016595 | 154 | 318 |
| 70663 | 275796 | 1106 | 1 |
| 99850 | 94017 | 1589 | 1385 |
| 110111 | 93034 | 1771 | 880 |

Table 6.3: The four polygons not considered for best merging in the Lisbon image when best merge with polygons2 is performed. The first column gives the polygon number as in the polygon file. The second column gives the number of pixels within that polygon. In the last two columns, the image raster coordinates of the polygons (north-west, i.e. top left) are given.

| polygon | #cells | win_n | win_w |
|---|---|---|---|
| 1 | 19308940 | 1 | 1 |
| 863 | 4471544 | 1 | 5951 |

Table 6.4: Polygons not considered for best merging.

sizes and different types of satellite data. For small images, $500 \times 500$ pixels $\times 3$ channels (SPOT) already the difference is clear between using the integrated method and the region growing alone; [merging with polygons took 11 CPU-seconds while merging without polygons took almost one hour (see table 6.1)].

The merging alone on large areas in the Lisbon image (total pixels: $1016595 + 275796 + 940147 + 93034 = 1479442$, see table 6.3) took about 55 hours. The total area of the four polygons covers about 40% of the image. Therefore (if it is assumed that merging time increases linearly with the number of pixels) the merging time for the total image would be 135 hours! The gain in computation time clearly improves if large polygons are omitted from the merging process, as they consume the most CPU-time.

The problem however with omitting large polygons from the merging process is that a-priori knowledge about the image is required and that a comparison of the edge image with the original image needs to be made. The comparison can determine which polygons can be omitted from the region growing process. This

| | elapsed time HH:MM:SS.S | CPU-time HH:MM:SS.S |
|---|---|---|
| edge detection | 1:30:35.6 | 23:25.1 |
| parallel edge linking | 4:23:01.6 | 10:36.4 |
| polygon determination | 11:16.7 | 1:46.7 |
| best merge with polygons | 1:59:22.0 | 1:01:42.7 |

Table 6.5: time elapsed and CPU-time needed for SPOT panchromatic image.

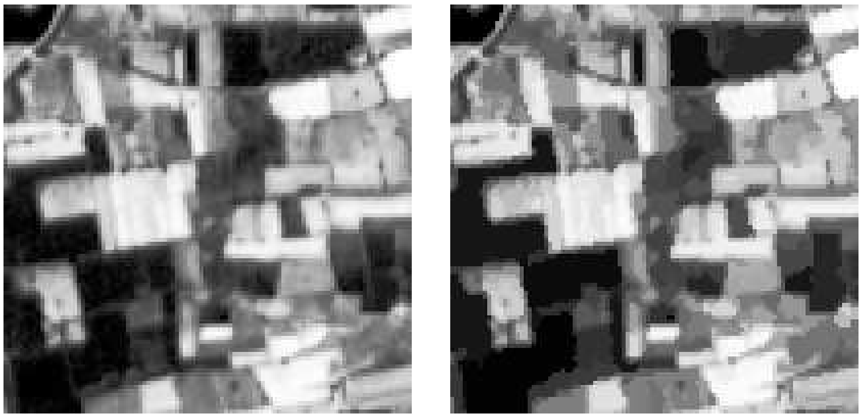
Image 6.7: Pangre SPOT-P image, Greece, 6700 × 5780 pixels. [Original satellite data ©SPOT-Image.]

can be done for large water areas, but when nothing is known about the class of the polygon, this approach can not be applied and therefore merging needs to be applied to the polygon resulting in excessive computation times and probably causing a collapse of the merge program when the polygon is too large.

Another problem with omitting the large polygons from the merging process is that pixels at the border of the polygon could belong to different classes. The edge detection and edge linking did not detect the different land cover types and therefore an incorrect merging of these pixels occurs.

However, homogeneity tests could give information about the distribution of the spectral values (i.e. mean, variance, standard deviation etc.) of the pixels in the polygon area.

From the above it can be concluded that the integration of edge detection and region growing results in a hybrid method which can be many times faster than the region growing method by itself. Although for large images (about 1000×1000 pixels), it is necessary to eliminate some polygons from region growing if computing time is to be kept inside reasonable bounds.



(a) (b)

Image 6.8: (a) Zoomed Greece SPOT-P image ($100 \times 100$) pixels Upper-left (x,y)-coordinates of Greece image (1025,3145) (b) segmented result.

# Chapter 7

## Applications of the Method

### 7.1 Introduction

In this chapter several applications of the segmentation algorithm are described. First a comparison is given of the classification by a neural network of a Landsat-TM image on a per-pixel basis and on the basis of the segmented scene. Secondly the use of the segmentation of optical imagery for speckle filtering radar imagery is given. Thirdly the use of segmentation in crop-profile extraction using NOAA-AVHRR NDVI imagery, is described.

### 7.2 Classification of a segmented image by a neural network

The Lisbon area of Portugal is a typical Southern European area with a very mixed landscape containing dense urban and industrial areas, agricultural zones, a variety of forests, vineyard and fruit cultivation areas and coastal areas (see the Lisbon area image, image 6.6).

For comparison, both the original Lisbon and the segmented Lisbon images were classified with a Multi-Layer Perceptron (MLP) Neural Network (NN) with backpropagation learning. A description of this MLP neural network is given in Rumelhart & McClelland [90] and Wilkinson & Kanellopoulos [118]. Such networks consist of layers of 'neuron' processing elements connected by 'weights'. The use of the method for the classification of satellite imagery is described in Kanellopoulos *et al.* [50, 51]. The network architecture used in the tests consisted of an input layer with six neurons (the number of bands), a hidden layer with 28 neurons and an output layer with 16 neurons (the number of classes). See appendix C for an introduction to neural networks.

| | | | |
|---|---|---|---|
| 1 tiled/concrete | 2 sand | 3 soil | 4 seawater |
| 5 freshwater | 6 estuary/lagoon | 7 wheat | 8 barley |
| 9 maize | 10 rice | 11 vineyards | 12 weeds |
| 13 garrigue | 14 grassland | 15 deciduous forest | 16 coniferous forest |

Table 7.1: the 16 classes for the Lisbon image.

The neural network was trained to recognize 16 classes in the Lisbon image using a detailed ground truth data set collected in a field survey in June 1991. The ground truth set was split into two separate parts, one of 8047 pixels for training the network and one of 4286 pixels for independently verifying the classification accuracy. The training was done for the 16 classes which are given in table 7.1. [Note: there was considerable variation in the number of pixels available for training and testing for each class on account of the differences in the landscape coverage of different classes and the amount of ground truth data available from a limited survey].

The confusion matrix for the classification of the non-segmented Lisbon image is given in table 7.2. The columns give the ground truth class of the pixels. The rows give the classification of the pixels. Thus for example, given ground truth class 1, the total of pixels known within this class is $67+1+2+1+8+14 = 93$. Of these 93, 67 pixels have been classified correctly (i.e. class 1) and 26 incorrectly (assigned to classes 3,5,7,12,14 respectively), resulting in an error of omission (i.e. pixels known belonging to a certain class are not assigned to that class) of $(93 - 67)/93 \times 100 = 27.96\%$, see table 7.3. To class 1 however, have been assigned in total $67 + 1 + 50 + 1 + 1 + 1 + 4 = 125$ pixels (originally from ground truth classes 1,2,3,11,12,15,16 respectively). The error of commission, (i.e. pixels assigned incorrectly to this class), is $(125 - 67)/125 \times 100 = 46.40\%$, as shown in table 7.3.

The overall accuracies of commission and omission are given in table 7.6. As can be seen the overall accuracy of the classification is 70.70%, where the average error of omission and commission is about 36% (calculated as averages of percentages without taking specific account of the differences in the numbers of pixels per class).

Classification of the Lisbon image on a per-pixel basis for both the non-segmented and segmented image are shown in images 7.1 and 7.2. The confusion matrix for the classified segmented Lisbon image is shown in table 7.4. The per pixel classification for the segmented Lisbon image was done using the same neural network configuration (i.e. with the same number of layers, number of neurons per layer, weights etc. for the NN) as for the non-segmented Lisbon image, and the same verification, set resulting in the confusion matrix as given in table 7.4. The omission and commission errors are shown for both cases in tables 7.3 and 7.5.

In both tables 7.3 and 7.5, for some classes (e.g. 10, rice) it can be seen that the error of omission (57.81%) is very high compared to the error of commission (8.47%). Other classes (e.g. class 5, freshwater) have a high error of commission (34.92%) and a low error of omission (6.82%). In the first case the classifier has problems recognizing pixels belonging actually to that class but on the contrary the classifier is quite good in distinguishing pixels known not to belong to that class.

The second case indicates that the classifier is not good in distinguishing the different classes. But pixels known belonging to a class are assigned to that class with a low probability of error.

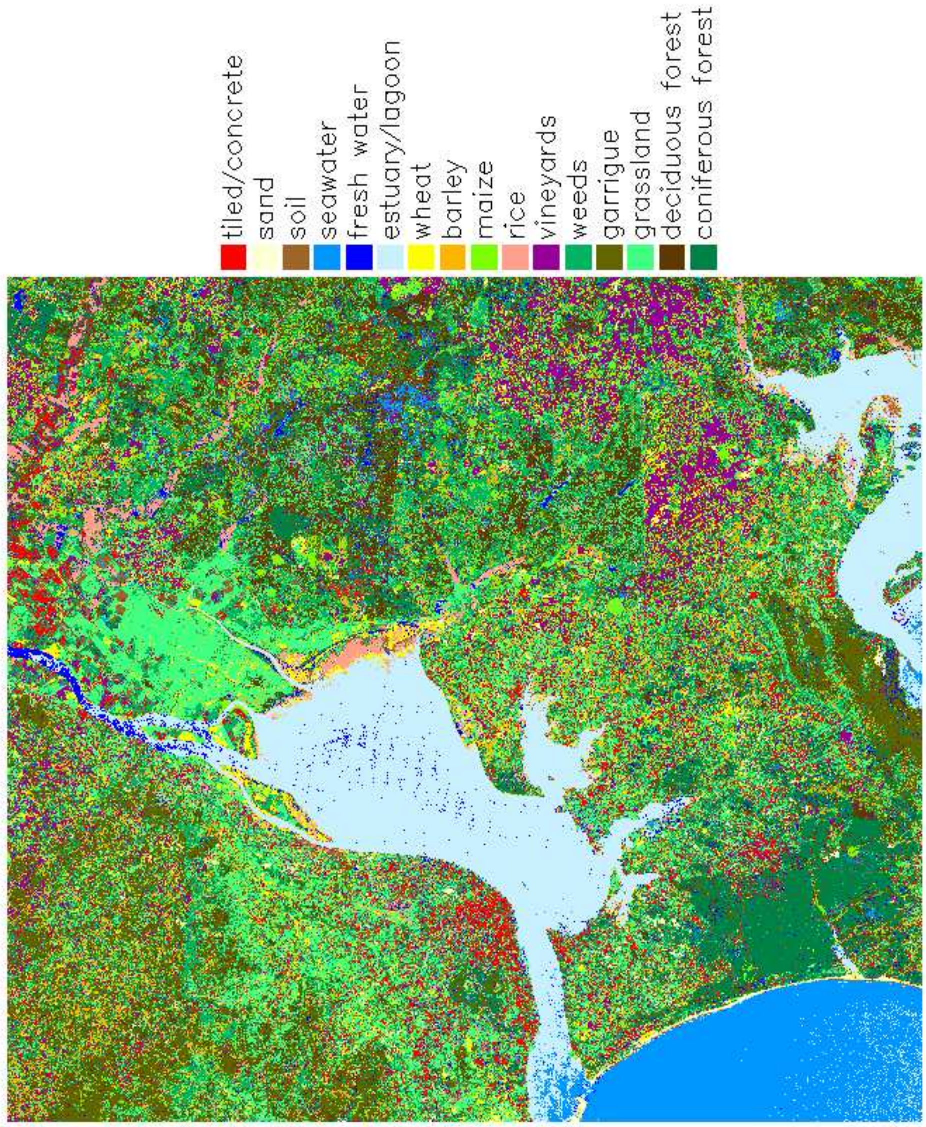



Image 7.1: Lisbon image classified by a neural network on a per-pixel basis.

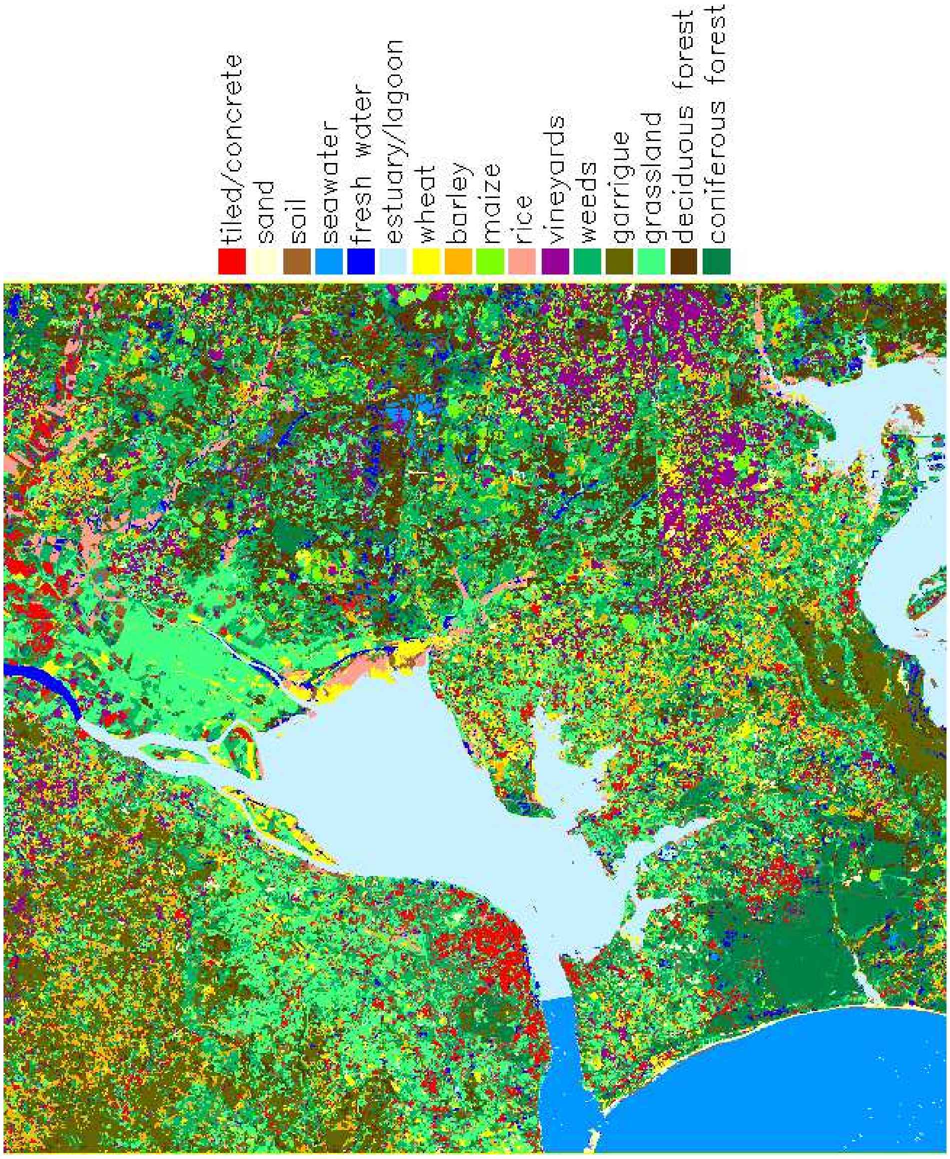



Image 7.2: segmented Lisbon image classified by a neural network.

| | | Ground truth class | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| | 1 | 67 | 1 | 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 4 |
| C | 2 | 0 | 98 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 |
| l | 3 | 1 | 0 | 368 | 0 | 0 | 0 | 37 | 3 | 0 | 5 | 1 | 3 | 2 | 2 | 2 | 4 |
| a | 4 | 0 | 0 | 1 | 125 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s | 5 | 2 | 0 | 8 | 0 | 82 | 0 | 0 | 0 | 0 | 19 | 0 | 1 | 2 | 0 | 6 | 6 |
| s | 6 | 0 | 0 | 0 | 7 | 6 | 251 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 |
| i | 7 | 1 | 0 | 5 | 0 | 0 | 0 | 95 | 2 | 0 | 26 | 8 | 20 | 0 | 7 | 2 | 2 |
| f | 8 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 56 | 0 | 0 | 2 | 20 | 0 | 11 | 1 | 0 |
| i | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 0 | 0 | 8 | 0 | 0 |
| c | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 54 | 0 | 0 | 3 | 0 | 0 | 0 |
| a | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 319 | 0 | 0 | 1 | 1 | 1 |
| t | 12 | 8 | 0 | 28 | 0 | 0 | 0 | 1 | 36 | 0 | 3 | 5 | 216 | 3 | 25 | 75 | 27 |
| i | 13 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 9 | 0 | 1 | 0 | 33 | 178 | 9 | 189 | 46 |
| o | 14 | 14 | 0 | 29 | 0 | 0 | 0 | 20 | 12 | 0 | 4 | 2 | 134 | 3 | 1042 | 13 | 2 |
| n | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 20 | 72 | 3 | 3 | 46 | 0 |
| | 16 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | 15 | 20 | 3 | 34 | 33 |

Table 7.2: confusion matrix for classified non-segmented Lisbon image with a multi-layer perceptron neural network (i.e. per-pixel basis).

| Class no | Row Total | Error of Commission (%) | Column Total | Error of Omission (%) |
|---|---|---|---|---|
| 1 | 125 | 46.40 | 93 | 27.96 |
| 2 | 103 | 4.85 | 99 | 1.01 |
| 3 | 428 | 14.02 | 503 | 26.84 |
| 4 | 126 | 0.79 | 132 | 5.30 |
| 5 | 126 | 34.92 | 88 | 6.82 |
| 6 | 274 | 8.39 | 251 | 0.00 |
| 7 | 168 | 44.45 | 516 | 39.10 |
| 8 | 93 | 39.78 | 121 | 53.72 |
| 9 | 14 | 100.00 | 13 | 100.00 |
| 10 | 59 | 8.47 | 128 | 57.81 |
| 11 | 335 | 4.78 | 366 | 12.84 |
| 12 | 427 | 49.41 | 515 | 58.06 |
| 13 | 474 | 62.45 | 214 | 16.82 |
| 14 | 275 | 18.27 | 1111 | 6.21 |
| 15 | 147 | 68.71 | 370 | 87.57 |
| 16 | 112 | 70.54 | 126 | 73.81 |
| | 4286 | | 4286 | |

Table 7.3: omission and commission errors from table 7.2.

| | | Ground truth class | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| | 1 | 91 | 0 | 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 2 | 0 | 98 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 2 |
| l | 3 | 0 | 0 | 428 | 0 | 0 | 0 | 37 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 4 |
| a | 4 | 0 | 0 | 0 | 123 | 0 | 243 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s | 5 | 0 | 0 | 7 | 0 | 87 | 0 | 0 | 0 | 0 | 26 | 0 | 0 | 0 | 0 | 0 | 0 |
| s | 6 | 0 | 0 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 |
| i | 7 | 0 | 0 | 0 | 9 | 0 | 0 | 103 | 0 | 13 | 29 | 0 | 71 | 0 | 2 | 3 | 2 |
| f | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 61 | 0 | 0 | 1 | 2 | 3 | 4 | 0 | 0 |
| i | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 0 |
| c | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 58 | 0 | 0 | 0 | 0 | 0 | 2 |
| a | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 329 | 0 | 0 | 2 | 4 | 0 |
| t | 12 | 2 | 1 | 21 | 0 | 0 | 0 | 0 | 53 | 0 | 0 | 6 | 292 | 2 | 30 | 80 | 29 |
| i | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 9 | 178 | 10 | 193 | 38 |
| o | 14 | 0 | 0 | 8 | 0 | 0 | 0 | 14 | 2 | 0 | 0 | 0 | 54 | 0 | 1048 | 10 | 5 |
| n | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 21 | 76 | 2 | 6 | 56 | 0 |
| | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 11 | 29 | 0 | 22 | 44 |

Table 7.4: confusion matrix for segmented Lisbon image (threshold 15) classified with a multi-layer perceptron neural network (i.e. per-pixel basis).

A very interesting class is class 6 (estuary/lagoon). After segmentation this class is almost completely absorbed by class 4 (seawater). The reason for this is that the segmentation process has joined two larger regions of spectrally similar classes into one region which is spectrally closer on average to class 4. The eight pixels on the leading diagonal for class 6 are from another separate area, which after segmentation retains the characteristics of class 6. But for purposes such as land cover determination and classification of crops, this is not of interest.

Also it is interesting to note that all of the pixels from class 9 (maize) were erroneously assigned to class 11 (vineyards) by the neural network in the unsegmented image and to class 7 in the segmented image. Class 9 is clearly spectrally similar to these other classes and the segmentation has simply made the maize pixels closer to class 7 rather than class 11 and does not help to reduce in this case the initial classification error.

By examining the confusion matrix belonging to the segmented Lisbon image (table 7.4), it can be concluded that overall this matrix is less noisy than the per-pixel one (table 7.2) with fewer non-zero entries off the leading diagonal.

The overall accuracies of commission and omission for the segmented and per-pixel cases are given in tables 7.6 and 7.7. As can be seen in the segmented case the overall accuracy of the classification is 70.09%, whereas the average error of omission is about 37% and commission is about 39%.

If a comparison is made between tables 7.6 and 7.7 it can be seen that surprisingly the overall accuracy of the classification of the non-segmented image is higher than for the segmented Lisbon image. This behaviour can be explained by the erroneous inclusion of pixels of class 6 into the seawater region (class 4) by the segmentation process. But if the separate classes are examined for the leading diagonal in the confusion matrices it can be seen that the number for all

| class no | Row Total | Error of Commission (%) | Column Total | Error of Omission (%) |
|---|---|---|---|---|
| 1 | 130 | 30.00 | 93 | 2.15 |
| 2 | 109 | 10.09 | 99 | 1.01 |
| 3 | 474 | 9.70 | 503 | 14.89 |
| 4 | 366 | 66.39 | 132 | 6.82 |
| 5 | 120 | 27.50 | 88 | 1.34 |
| 6 | 22 | 63.64 | 251 | 96.81 |
| 7 | 232 | 55.60 | 156 | 33.97 |
| 8 | 71 | 14.08 | 121 | 49.59 |
| 9 | 7 | 100.00 | 13 | 100.00 |
| 10 | 60 | 3.33 | 128 | 54.69 |
| 11 | 335 | 1.79 | 366 | 10.11 |
| 12 | 516 | 43.41 | 515 | 43.30 |
| 13 | 433 | 58.89 | 214 | 16.82 |
| 14 | 1141 | 8.15 | 1111 | 5.67 |
| 15 | 163 | 65.64 | 370 | 84.86 |
| 16 | 107 | 58.88 | 126 | 65.08 |
| | 4286 | | 4286 | |

Table 7.5: omission and commission errors from table 7.4.

| | |
|---|---|
| Average Omit Precision per Class | 64.13% |
| Average Commit Precision per Class | 64.05% |
| Overall Classification Accuracy | 70.70% |

Table 7.6: overall accuracies of classification of non-segmented Lisbon image.

the classes except 4 and 6, are higher for the segmented image. For class 4 the number of pixels is lower because some of the pixels have been assigned to class 7 (wheat).

Thus it can be concluded that (for this image) segmentation improves the classification accuracy for some classes. However, it is also clear that merging regions of spectrally similar classes can have an adverse effect on overall classification accuracy.

Zoomed versions of the non-segmented and segmented classified images are shown in images 7.3 and 7.5 (coincident with image 5.1). Image 7.4 is added for comparison of different threshold settings. In these images it is clearly visible that the segmented images have less class "noise". Without examining the classification accuracy, it can be said that for land cover applications in which homogeneous regions are usually required, this is a more desirable situation.

Images 7.4 and 7.5 look very similar. However, it is expected that or image 7.4 would look similar to image 5.2 and that image 7.5 would look similar to image 5.3. In fact images 5.3 and 7.5 look quite similar, but images 5.2 and 7.4 are very different. This behaviour can be explained by the class coverage of the different

| | |
|---|---|
| Average Omit Precision per Class | 63.32% |
| Average Commit Precision per Class | 61.43% |
| Overall Classification Accuracy | 70.09% |

Table 7.7: overall accuracies of classification of segmented image with threshold 15.

segments found in image 5.2. Probably the different segments all have the same crop, and thus explaining why after classification the images look so different.

Also a comparison can be made between image 5.3 (i.e. the zoomed Lisbon area segmented without polygons threshold 15 image) and the zoomed classified Lisbon image, image 7.5. It can be seen that the classified areas and segmented areas are similar. This is an un-expected result, because image 5.3 is segmented without polygons and subject to over-growing. The behaviour can be explained however, by the fact that both the segmentation and the classification look for pixels with similar spectral values and it is possible that separate regions end up in the same thematic class, even if they were initially separated by the segmentation process.

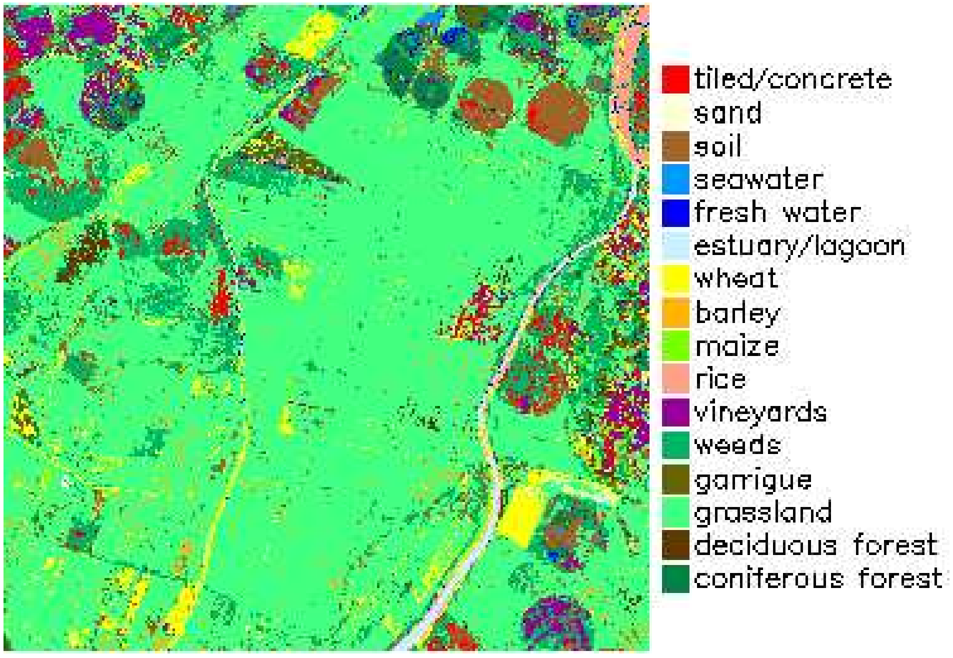



Image 7.3: zoomed part of the classified non-segmented Lisbon image (i.e. image 7.1)

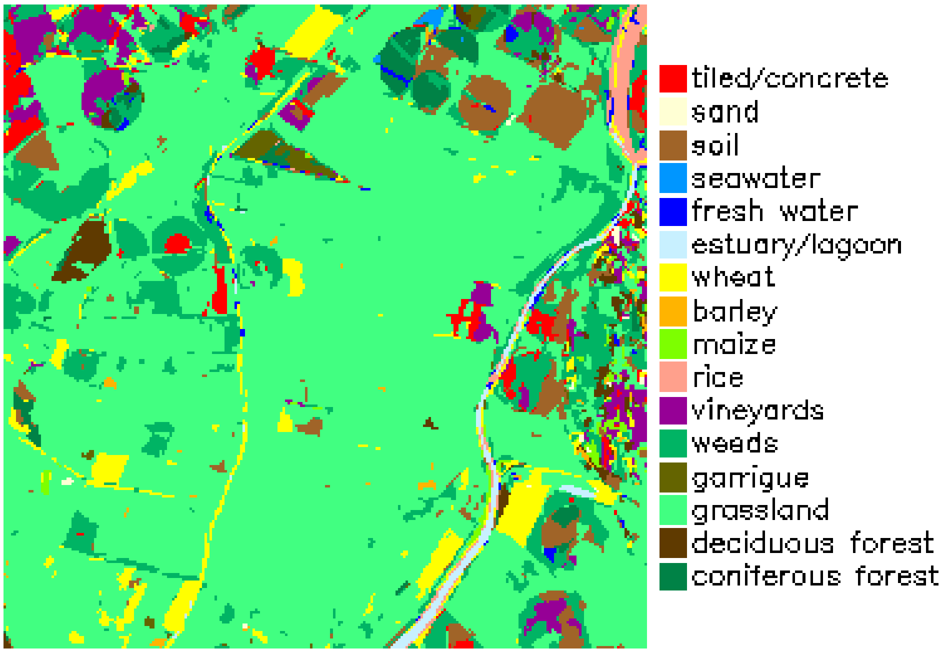



Image 7.4: zoomed part of the classified segmented Lisbon image (i.e. image 7.2 with merging threshold 6)

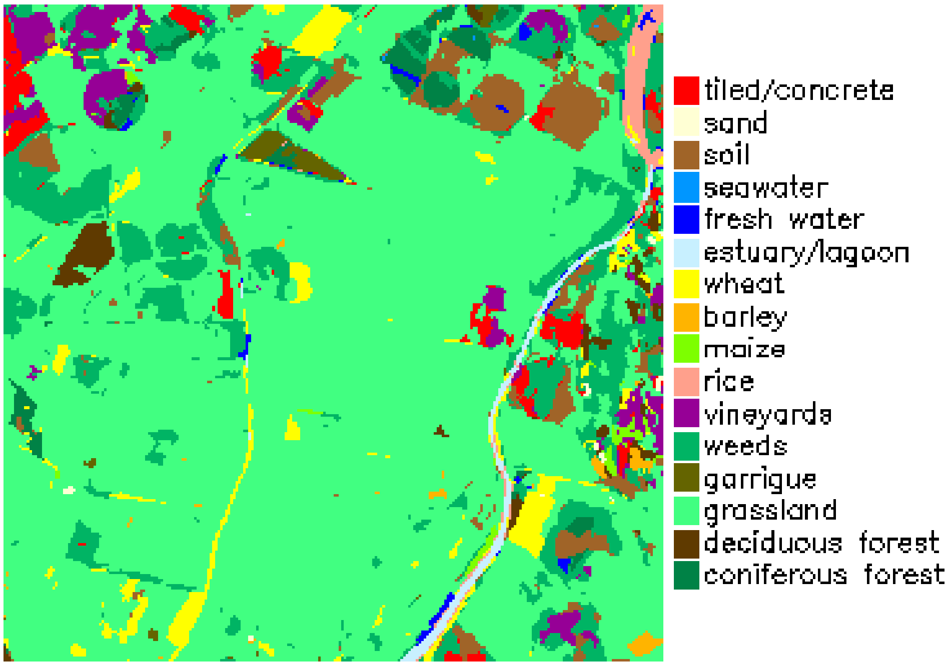



Image 7.5: zoomed part of the classified segmented Lisbon image (i.e. image 7.2 with merging threshold 15)

| | | Ground truth class | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| | 1 | 80 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 0 | 2 |
| C | 2 | 0 | 98 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 1 |
| l | 3 | 3 | 0 | 435 | 0 | 0 | 0 | 39 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 2 | 5 |
| a | 4 | 0 | 0 | 0 | 123 | 0 | 243 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s | 5 | 0 | 0 | 7 | 0 | 87 | 0 | 0 | 0 | 0 | 16 | 0 | 2 | 0 | 0 | 2 | 0 |
| s | 6 | 0 | 0 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |
| i | 7 | 0 | 0 | 1 | 9 | 0 | 0 | 104 | 0 | 13 | 21 | 7 | 59 | 0 | 7 | 0 | 2 |
| f | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 60 | 0 | 0 | 2 | 2 | 1 | 2 | 1 | 0 |
| i | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 6 | 0 | 0 |
| c | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 73 | 0 | 0 | 0 | 0 | 0 | 2 |
| a | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 322 | 0 | 0 | 2 | 5 | 0 |
| t | 12 | 10 | 1 | 26 | 0 | 0 | 0 | 0 | 31 | 0 | 1 | 2 | 253 | 4 | 18 | 75 | 20 |
| i | 13 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 23 | 173 | 12 | 186 | 45 |
| o | 14 | 0 | 0 | 9 | 0 | 0 | 0 | 11 | 12 | 0 | 3 | 0 | 88 | 0 | 1058 | 8 | 2 |
| n | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 7 | 0 | 0 | 26 | 74 | 2 | 0 | 61 | 0 |
| | 16 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 11 | 33 | 0 | 30 | 47 |

Table 7.8: confusion matrix for segmented Lisbon image (threshold 6) classified with a multi-layer perceptron neural network.

### 7.2.1 Influence of segment size

A question arising now is how the segment size influences the classification result. Therefore the segmentation is repeated but with a merging threshold of 6 instead of 15, resulting in tables 7.8-7.10. From these tables it can be seen that the overall accuracy decreases when a lower threshold is used. But a comparison of the diagonal of both confusion matrices, shows that for certain classes (e.g. 3 (soil), 10 (rice), 14 (grassland), 15 (deciduous forest), 16 (coniferous forest)) the accuracy is higher for the segmented image with threshold 6. This can be explained through the inclusion of pixels having a more distinct spectral value for merging with a high threshold. The above indicates that threshold setting is a difficult problem, because increasing the threshold does not indicate that a higher classification accuracy can be achieved for all classes.

The increase in region size is shown in table 7.11. It can be seen that the number of regions with a small size decreases for when the threshold is increased form 6 to 15.

In table 7.12 the average region size and the remaining number of regions is shown for the Lisbon image with a threshold of 6 and 15 for the best merge. If a higher merging threshold is used the number of regions tends to decrease whereas the average region size tends to increase.

## 7.3 ERS-1 filtering

The combination of Landsat TM and ERS-1 SAR data can potentially lead to improved land cover mapping. Initially however SAR data must undergo speckle noise filtering. In Schoenmakers *et al.* [96] the potential use of segmentations

| class no | Row Total | Error of Commission (%) | Column Total | Error of Omission (%) |
|---|---|---|---|---|
| 1 | 95 | 15.79 | 93 | 13.98 |
| 2 | 104 | 5.77 | 99 | 1.01 |
| 3 | 490 | 11.22 | 503 | 13.52 |
| 4 | 366 | 66.39 | 132 | 6.82 |
| 5 | 114 | 23.68 | 88 | 1.14 |
| 6 | 18 | 55.56 | 251 | 96.81 |
| 7 | 223 | 53.36 | 156 | 33.33 |
| 8 | 68 | 11.76 | 121 | 50.41 |
| 9 | 9 | 100.00 | 13 | 100.00 |
| 10 | 75 | 2.67 | 128 | 42.97 |
| 11 | 329 | 2.13 | 366 | 12.12 |
| 12 | 441 | 42.63 | 515 | 50.87 |
| 13 | 460 | 62.39 | 214 | 19.16 |
| 14 | 1191 | 12.27 | 1111 | 4.77 |
| 15 | 172 | 64.53 | 370 | 83.51 |
| 16 | 131 | 64.12 | 126 | 62.70 |
| | 4286 | | 4286 | |

Table 7.9: omission and commission errors from table 7.8.

| | |
|---|---|
| Average Omit Precision per Class | 62.94% |
| Average Commit Precision per Class | 62.93% |
| Overall Classification Accuracy | 69.58% |

Table 7.10: overall accuracies of classification of segmented image with threshold 6.

of Landsat TM data to describe boundaries which can serve as a basis for SAR speckle noise filtering prior to incorporation in integrated multi-sensor classification, is described.

One of the main difficulties in combining Landsat-TM and ERS-1 SAR imagery is the smoothing of the SAR data to remove speckle noise prior to integrated classification. Although several good speckle filters exist (e.g. Lopes *et al.* [64], Nezry *et al.* [75].) they can have the effect of eroding boundaries such as those between adjacent agricultural fields and of 'transporting' noise across segment boundaries. This problem could in principle be solved by using a segmentation of coincident image data from a different source which is not susceptible to speckle noise effects, i.e. from high resolution optical or infra-red satellite imagery.

The full procedure for combining the two types of data involves the main steps given in algorithm 7.1.

The procedure ensures that noise from neighbouring segments is not merged into the average backscatter signal for a particular segment of an ERS-1 SAR image and that this signal is therefore more accurate and more useful in clas-

| | threshold 6 | threshold 15 |
|---|---|---|
| 1 | 1542 | 1504 |
| 2 | 207977 | 114862 |
| 3 | 122965 | 49911 |
| 4 | 62869 | 17922 |
| 5 | 34368 | 8903 |
| 6 | 22895 | 7233 |
| 7 | 22895 | 7125 |
| 8 | 13124 | 7126 |
| 9 | 10299 | 6842 |
| 10 | 8285 | 6641 |
| | 31431 | 31402 |
| | 46092 | 46078 |
| | 195604 | 195560 |
| | 259302 | 259090 |

Table 7.11: number of regions with a size from 1 to 10 for thresholds 6 and 15 respectively (first 10 lines). The second part gives the four largest regions in the image.

| threshold 6 | | threshold 15 | |
|---|---|---|---|
| 555652 | 6.303 | 306132 | 11.441 |

Table 7.12: average region size (columns 2 and 4) and remaining number of regions (columns 1 and 3) in Lisbon image.

sification of landscape features. After the smoothing has been carried out, the per-segment averages of the radar backscatter signal and the optical/infra-red reflectances in the various channels can be used as separate features for a pattern classifier.

The geo-referenced ERS-1/SAR PRI image, for the zoomed Lisbon area image, recorded March 21st, 1992, is shown in image 7.6. The original 16-bit values have been re-scaled into 8-bit values. The speckle filtered ERS-1/SAR image based on the segments derived from the Landsat-TM image is shown in image 7.7.

For comparison, the same image has been filtered using the speckle filter as described in Nezry *et al.* [75]. The result is shown in image 7.8. Discussions with experts in the field of speckle filtering, showed their interest in the optical segmentation method and their preference for this type of speckle removal over speckle filters using the radar data alone.

```
1. acquisition of Landsat TM and ERS-1 data for coincident
   area and similar acquisition dates.
2. geo-referencing of TM and SAR data and geometrical
   correction.
3. segmentation of TM data.
4. overlay of TM segment boundaries on SAR data.
5. smoothing of SAR data within segment boundaries
   by averaging within segment.
6. integrated use of TM spectral channel radiances and
   SAR intensity data in land cover classification.
```

Algorithm 7.1: different steps for speckle filtering a geo-referenced ERS-1/SAR image using segments of an optical image.

## 7.4 Segmentation of NDVI imagery for profile extraction

A first attempt to segment multi-temporal NDVI[1] imagery is done in Genovese *et al.* [35]. A multi-temporal dataset comprising 17 NOAA[2]-AVHRR[3] images of the Andalusian region (Spain) was selected. The data set contained only images having a cloud cover of less than 10 percent. The 17 images were selected over a time interval of one year. For every month one image was selected, with more images in the period when the chlorophyll content of the vegetation was expected to be high.

[1]Normalized Difference Vegetation Index.
[2]National Oceanic Atmospheric Administration.
[3]Advanced Very High Resolution Radiometer.

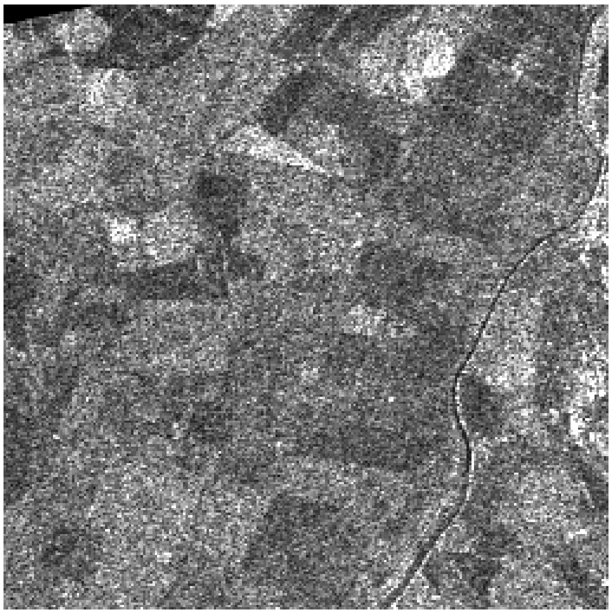


Image 7.6: the ERS1-SAR image, $256 \times 256$ pixels, original satellite data ©ESA.

The data set was pre-processed with the SPACE[4] software package. SPACE performs operations like co-registration calibration, atmospheric correction, and cloud detection and, transforms the original 10 bit per pixel data into 16 bit data of which 6 bits contain information extracted with SPACE and the other 10 bits contain the spectral value per pixel. From the original 5 channel data per date, the NDVI which is defined as

$$\frac{Ch2 - Ch1}{Ch2 + Ch1} \tag{7.1}$$

is computed, where $Ch1$ and $Ch2$ are channel 1 and 2 of the five channel AVHRR data. The NDVI values are scaled in the integer interval $[0, 255]$. Now the lowest values indicate water and clouds, low values indicate bare soil and low vegetation cover, and the highest values indicate areas with a good vegetation cover. The spectral range of the 5 bands is given in table 7.13.

| Band | wavelength ($\mu$m) | nominal spectral location |
|---|---|---|
| 1 | 0.58-0.68 | green/red |
| 2 | 0.72-1.10 | near IR |
| 3 | 3.55-3.93 | mid IR |
| 4 | 10.3-11.3 | thermal IR |
| 5 | 11.5-12.5 | thermal IR |

Table 7.13: the NOAA AVHRR bands.

The 17 NDVI images were integrated into one multi-temporal image, i.e. one

[4]Software for Processing AVHRR data for the Community of Europe, courtesy European Union.

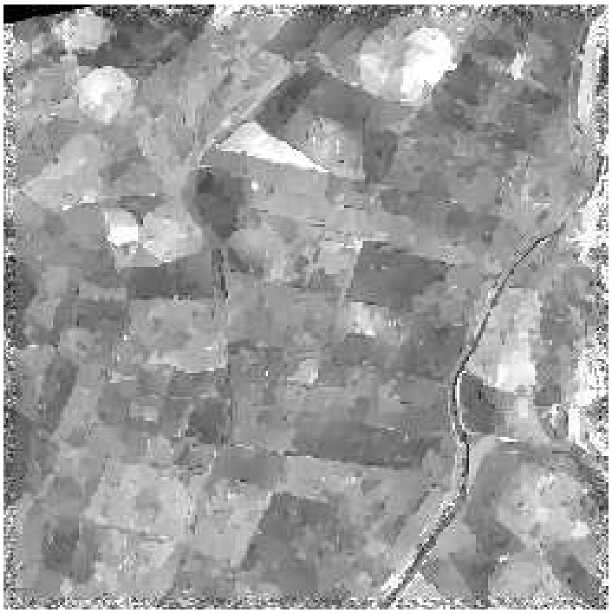


Image 7.7: the region mean filtered ERS1-SAR image based on segmented Landsat-TM image.

17 band image, where each band represented the NDVI value for a pixel for a date selected. Hence every pixel represented the NDVI profile for a ground cover type for 17 discrete points over one year.

Clouds were removed using the cloud mask (as derived by the SPACE software) and a linear interpolation between the previous cloud free date for a pixel and the next cloud free date for that same pixel was used to ensure that a complete 17 component NDVI signature could be built up for each pixel in the image. The 17 band image with deleted clouds was then filtered using the 'butcher' algorithm as described in appendix A. Subsequently the image was segmented using the multi-band, information and was then subject to unsupervised classification using the ISODATA algorithm (ERDAS [29]). [In the future an unsupervised neural network as implemented by Vuurpijl [112, 113, 114] will be used.] Using this approach usefull NDVI profiles were extracted, for some important thematic classes such as forests and rice plantations.

## 7.5 Concluding remarks

In this chapter several applications for the segmentation method have been indicated. These included classification for land cover mapping, radar data speckle filtering and NDVI signature extraction. In the first two cases the ability of the segmentation process to reduce spatial noise, either in an optical classified image or in a raw SAR image, has been demonstrated.

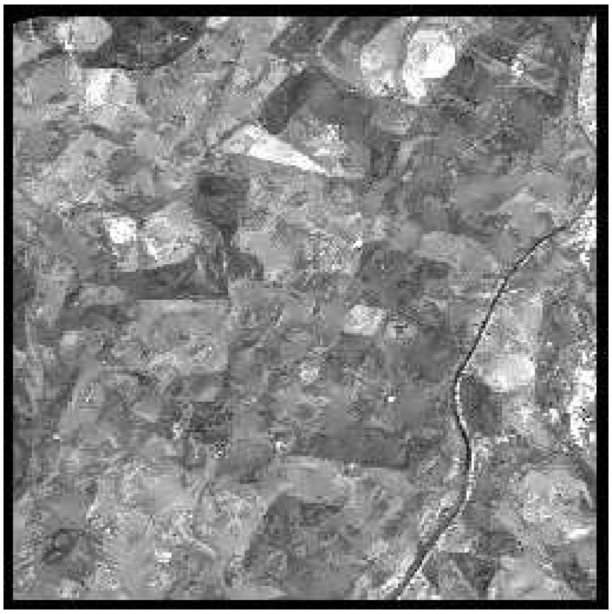


Image 7.8: the speckle filtered ERS1-SAR image using the method of Nezry *et al.* [75].

# Chapter 8

## Conclusions

In this dissertation a segmentation method has been developed which integrates edge detection, edge linking, and region growing into a hybrid method.

In chapter 1 it was shown that segmentation methods based on edge detection and region growing separately have their shortcomings, especially in regions where many mixed and isolated pixels are found. For edge detection methods on the one hand, the main problem is that few closed object boundaries are found. For the region based methods on the other hand, the main problem is when to stop growing. If the growing is stopped at an early stage many small regions or isolated pixels remain. If the growing is stopped at a later stage, too large regions might result, covering different fields on the ground. Another problem inherent to region growing is its order dependence.

In chapter 2 different existing methods for both edge detection and region growing were discussed. For the edge detection part, several options existed. The Frei & Chen [33] edge detector, adapted by Lacroix [60], was chosen. It has been shown by Frei & Chen that their edge detector gives improved results over the edge detectors developed by Sobel [26], and Kirsch [55]. Of course also other edge detectors like the ones developed by Canny [13], Haralick [38] and, Marr & Hildreth [65], can be used. Implementation of these and comparison with the Frei & Chen edge detector has to show which ones gives the best results on remotely sensed imagery. But results obtained with the improved Frei & Chen operator are satisfactory. For the region growing on the other hand, the best merge algorithm (Tilton [106, 107, 108, 109] and Beaulieu & Goldberg [7]) was chosen because it is order independent.

In chapter 3, the Frei & Chen [33] edge detector has been used for boundary detection. It has been shown in that chapter that the edge magnitude as computed by the Frei & Chen edge detector is dependent on the absolute intensity (grey level) values. Higher intensity values give a lower edge magnitude, but lower spectral values give a higher one. This undesirable behaviour has been overcome by subtracting the spectral value of the pixel having the minimum intensity value from all the pixels within the window $W$. It has been shown that this radically improves the sensitivity of the Frei-Chen edge detector.

Another problem to be resolved was that the Frei & Chen edge detector was developed for use on single band imagery. Applying the edge detector on multi-band imagery causes a problem. For every band of every pixel the edge magnitude and edge direction are computed. Thus for a pixel several magnitude and direction values are derived. Thereafter the direction and magnitude values are taken for the band with the maximum edge magnitude although other possibilities for selecting edge magnitude and direction should be tried.

In chapter 4, a new method has been developed which tries to connect edge fragments resulting from the edge detection module, by adding missing edge pixels. This way gaps are closed and closed boundaries are defined. Redundant information, i.e. edge fragments not belonging to such closed polygons are deleted. The set of closed polygons divides the image into a set of areas having no assumed physical relationship.

In chapter 5, the order dependence properties of the best merge algorithm were examined. The best merge is the only region growing method known to be order independent, and therefore this method was chosen. In the original version as proposed by Tilton [106, 107, 108, 109] and Beaulieu & Goldberg [7], at every merging iteration the pair with the lowest merging cost is merged into a new region. It was shown that when several pairs simultaneously have the lowest merging costs, this method is order dependent, because a pair for merging is randomly selected. In order to overcome this problem a merging scheme has been proposed in which pairs having the same lowest merging costs are merged simultaneously. It has been indicated that when several connected pairs have to be merged simultaneously, the final segmentation result need not be the 'best', although the order dependence, and hence the repeatability, of the method is a distinct advantage.

To treat isolated and mixed pixels, a 'post-merging' stage has been added in which the neighbourhood of isolated pixels is examined and rules have been developed to add them to existing regions. The addition is done without updating the segment mean, in order to keep the segment mean 'pure'.

In chapter 6, the integration of the edge detection and the region growing was discussed. The implemented integration technique does the following: Within the image edges are detected and the separate edge fragments are inter-connected by the linking method. Doing so divides the image into a set of areas which can be processed (region grown) independently. It is assumed for this case that the areas covered by different polygons belong to different objects, having no physical relationship.

Already on small images (size $500 \times 500$ by 3 channels) the integrated use of edge detection, edge linking and region growing resulted in much faster processing on a sequential single processor computer, than the region growing by itself.

Due to the limited amount of RAM available on current UNIX-based workstations, it has been impossible until now to store full scene imagery in RAM. Moreover it is expected that the amount of data is going to increase due to the increased ground resolution of new satellite platforms to be launched in the future. For these reasons, the method as presented for segmenting a complete image, (which cannot be stored in RAM), needed to be adapted.

For the filtering and the edge detection, the slicing technique was introduced. For the edge linking part the slicing technique was expected to cause problems because one edge fragment could extend all over the image. This would cause too many I/O operations while following the edge fragment. Another problem was that the amount of memory needed to store the trees, could easily exceed available RAM. Therefore a method was developed which divides the output edge fragment image from the edge detection module, into square sub-images. On the square sub-images the linking and pruning are carried out independently. After the sub-images have been processed the sub-images are re-constructed into an edge image containing only closed polygons.

For the best merge algorithm, a different method was developed. Since the closed polygons divide the image into (small) areas, which can be best merged independently of each other, the best merge method has been adapted so that only pixels within a polygon are considered in the merging process. Hence edge pixels and pixels outside the polygon have been excluded.

One problem still remains: if a polygon exists having more pixels than can be stored in RAM, the program as it is currently implemented, collapses. To solve this, a simple solution has been proposed. Large polygons (size given by the user) are excluded from merging. Those polygons are left as one region and take the average of the intensity values of the pixels within the polygon as the region mean.

From the way the method has been developed to handle full scene imagery, an implementation for a parallel computer architecture is straightforward. As the filtering and the edge detection use the slicing technique, the image can be divided into sub-images which can then be given to separate processors. The implementation of the edge linking for whole images is already done so that the image can be divided into sub-images. These sub-images which are now started as separate processes, could be started as several processes on different CPUs. The region growing is also clear: supply the area covered by one polygon to one processor.

In chapter 7 it was shown that classification accuracy (for some classes) can be increased by performing segmentation. However, the segmentation showed a side effect, that pixels known to belong to different classes, are merged into one region, because they are spectrally similar. For those classes the classification result was worse than classification on the non-segmented image.

In the same chapter it was shown that segmentation can be useful for speckle filtering ERS-1/SAR imagery. A method has been described which uses the segmentation result of a Landsat-TM image to smooth a geo-referenced ERS-1 image of the same location. The segments of the Landsat image are projected on the ERS-1 image. Within the projected segment the average is taken.

Also its use in vegetation profile extraction in NOAA/AVHRR NDVI imagery was indicated. This has been done by creating a multi-temporal NDVI dataset. Using available ground truth, vegetation index profiles for specific crops were extracted.

Currently the method is operational in that it is able to segment large satellite images in a reasonable time (a few hours) on a single CPU-computer (SUN

SPARC10). But the method needs further development. Especially the edge linking needs further improvement in making explicit use of the edge direction. In the 'parallel' version of the edge linking, the image is divided into square sub-images, which are then independently linked. The borders of these images need separate processing in order to remove all un-connected edge fragments. Further improvement in I/O etc. could be made.

Also the process for addition of edge and isolated pixels to regions needs further development. Currently only the spectral distance between an edge/isolated pixel and a neighbouring region is examined. It would be better to examine the whole neighbourhood more closely.

Overall by looking at the segmentation results on different imagery and different kinds of data, I can conclude that the results are promising. But it has to be kept in mind that segmentation is subjective to human interpretation and therefore it is difficult to objectively judge the "quality", as it depends on the context in which it has been carried out.

# Appendix A

## A Pre-Segmentation Noise Filtering Procedure

### A.1 The method developed

Filters are defined with different window sizes. Within the window pixels are selected for smoothing using either a fixed selection (e.g. Nagao-Matsuyama [73]) or a fixed weighting function of all the pixels (e.g. adapted Gaussian weighted filtering [102]). Bovik *et al.* [11] state that the selection of pixels within a window for filter function computation is important for two reasons:

1. The degree of overlap between adjacent or neighbouring filter windows determines the statistical dependence of the output and
2. The spatial position of the pixels selected within the window influences the performance of the filter near edges aligned along various orientations.

The first reason indicates that if the window size is too large then the smoothing of two adjacent pixels $p$ and $q$, depends on the selection, of pixels for smoothing $p$ and $q$. In case $p$ is the central pixel of the moving window, then pixels within its window are selected for filtering. However, when $q$ is the central pixel of the moving window, because of the overlap of the windows, pixels being selected for $q$, might have been selected for $p$ and thus the output for $p$ as well as $q$ can be strongly related. If the window is small, then this relationship is expected to be less.

The second reason indicates that it is important how pixels are selected for smoothing. If the current pixel to be filtered is situated along an edge, then it is important to select the right candidates for smoothing. If pixels are selected from both sides of the edge, the output for that pixel might 'weaken' the edge.

To avoid the two above mentioned problems a new filter category has been developed for remotely sensed imagery with the property of smoothing regions but preserving roads, brooks etc., mostly appearing as the so called 'mixed' pixels.

The method presented has many degrees of freedom in implementation. It uses a window $W(i,j)$ as defined in equation (1.3) with $k = 1$. As the ground resolution of most high resolution satellite images currently in use for land cover mapping is 10-30 meters, ditches, roads, brooks etc. can be described within a single pixel (or several pixels), and thus within the window $W(i,j)$. Larger window sizes are not necessary to describe primitive image features like an edge, a line or an isolated pixel. The pixels selected for smoothing the central pixel $p_{i,j}$ of $W(i,j)$, depend on the spectral values and the local gradient.

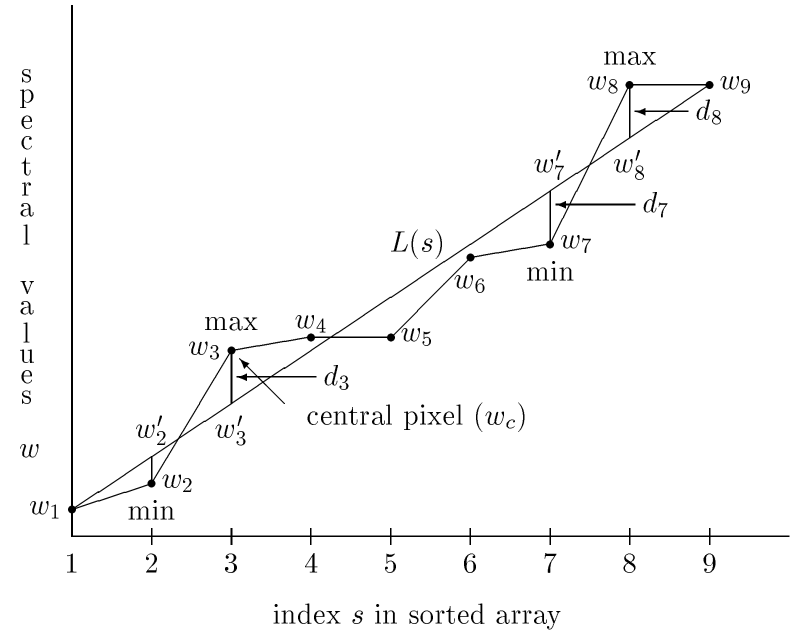



Figure A.1: $L(s)$ as a function of the spectral values.

The method is based on the morphological analysis of the joined-up line created by linking the sorted spectral values of a $3\times 3$ window as shown in figure A.1. Within a $3 \times 3$ window the adjacency of every pixel to the central position (pixel $p_{i,j}$ in equation (1.3)), is assured. The pixels within the window are sorted, and are plotted in the $s, w$-plane with the index in the sorted array on the $s$-axis and the spectral values of the pixels $w_i$ on the $w$-axis. In analyzing the gradient of this joined-up line around the central pixel $p_{i,j}$ ($w_c$ in figure A.1), it is attempted to divide the total window $W(i,j)$ into parts with similar reflectance values using morphological criteria, i.e. the examination of the local gradient of two adjacent pixels within the plot. The aim is to divide the window into parts belonging to a slope and a segment by applying simple, logical rules.

## A.2 The new algorithm

Roughly the method can be described as given in algorithm A.1. The first 6 steps are the same for all of the different kinds of applications. Later steps (algorithm A.2) can differ depending on the application for which it will be used.

Before giving the algorithm, first some terms used in the algorithm are explained. The variable $d_i$ is defined as the length of the vector in $w$-direction from $w_i$ to $L(s)$, i.e. $d_i$ is defined as the length of the vector from $w_i$ to $w_i'$, where $w_i \in w_s$, and $w_i'$ is the point on $L(s)$.

For figure A.1, $w_3 - w'_3 > 0$ and $w_8 - w'_8 > 0$, where $w_7 - w'_7 < 0$. Now a local minimum, a local maximum, a slope and, a segment can be defined as:

LOCAL MINIMUM Given $w_j, \ldots, w_i, \ldots, w_k$, $j \le i \le k$ and for all $w_i$, $w_i \le w'_i$, then $w_i$ is declared to be a local minimum, if $\forall_{d_{i1}} \mid d_i < d_{i1}$ with $j \le i1 \le k$.

LOCAL MAXIMUM Given $w_j, \ldots, w_i, \ldots, w_k$, $j \le i \le k$ and for all $w_i$, $w_i \ge w'_i$, then $w_i$ is declared to be a local maximum, if $\forall_{d_{i1}} \mid d_i > d_{i1}$ with $j \le i1 \le k$.

SLOPE Given $w_{min}$ and $w_{max}$ in $\{w_2, \ldots, w_8\}$, where $w_{min}$ is a local minimum and $w_{max}$ a local maximum. Then a slope is defined as:

- ▷ $w_1, \ldots, w_{max}$, if $w_{max}$ exists, or
- ▷ $w_{min}, \ldots, w_9$, if $w_{min}$ exists, or
- ▷ $w_{min}, \ldots, w_{max}$, if $w_{min}$ and $w_{max}$ exist.

SEGMENT Given $w_{min}$ and $w_{max}$ in $\{w_2, \ldots, w_8\}$, where $w_{min}$ is a local minimum and $w_{max}$ a local maximum. Then a segment is defined as:

- ▷ $w_1, \ldots, w_{min}$, if $w_{min}$ exists, or
- ▷ $w_{max}, \ldots, w_9$, if $w_{max}$ exists, or
- ▷ $w_{max}, \ldots, w_{min}$, if $w_{min}$ and $w_{max}$ exists.

Examining these definitions it is thus possible that a given pixel $w_i$ is situated on a segment and on a slope! (e.g. $w_3$ in figure A.1). If this is the case then $w_c$ is a local minimum or a local maximum. The interval $w_j, \ldots, w_i$ (or $w_i, \ldots, w_k$) for which $w_i$ is on a segment will be chosen.

1. sort $W(i, j)$ in increasing order giving $w_s$ = $(w_1, \ldots, w_9)$.
2. plot $w_1, \ldots, w_9$. On the horizontal axis $1, \ldots, 9$. On the vertical axis the spectral values of $w_1, \ldots, w_9$.
3. plot a linear function $L(s)$ from $w_1$ to $w_9$.
4. compute for $w_i$, i = $2, \ldots, 8$, the distance $d_i$.
5. determine the local minima and local maxima.
6. determine $w_j, \ldots, w_c, \ldots, w_k$ being a slope or a segment, $j \le c \le k$.

Algorithm A.1: slope/segment determination.

Different implementations, algorithm A.2, are possible for smoothing $w_c$. In algorithm A.2 $SD(x, y)$ indicates the spectral distance from $x$ to $y$.

The first implementation will be further referred to as 'the one iteration smoother'. The second implementation will be further referred to as 'the butcher'. The third implementation will be further referred to as 'the iterative one'. Herein a minimum segment/slope is a segment/slope which contains at least a minimum (user defined) number of pixels. This means that the iterative process is re-applied

```
1. { 'one iteration smoother' }
   smooth over w_j,...,w_k by taking the average.
2. { 'the butcher' }
   IF w_c on segment
   THEN smooth over w_j,...,w_k.
   ELSE { w_c is on a slope }
        IF SD(w_j,w_c) < SD(w_c,w_k) THEN w_c:=w_j ELSE w_c:=w_k.
3. { 'the iterative one' }
   apply steps 3-6 of algorithm A.1 iteratively around w_c on
   both slopes and segments until a minimum segment/slope
   w_j',...,w_k' is found.
   IF w_c on segment
   THEN smooth over w_j',...,w_k'.
   ELSE { w_c on slope }
        expand w_j',...,w_k' until the closest segment
        w_j'',...,w_j', or w_k',...,w_k'' is included.
        smooth over w_j'',...,w_j',...,w_c, or w_c,...,w_k',...,w_k''.
```

Algorithm A.2: slope/segment selection for three different implementations.

until a segment/slope is found which has less than the user defined number of pixels, and the former found slope/segment will be used.

Within the different implementations of algorithm A.2, diverse scenarios can be implemented. If the central pixel $w_c$ is a drop-out or a pulse within the sorted array $w_s$ (i.e. $w_c = w_1$ or $w_c = w_9$) then it can be seen as a noise pixel and replaced by $w_2$ or $w_8$ respectively. Of course no smoothing and gradient analysis will take place.

Regarding figure A.1 the local minima are $w_2$ and $w_7$ and local maxima are $w_3$ and $w_8$. Hence the segments are found between pixels $w_1$ and $w_2$, between pixels $w_3$ and $w_7$, and pixels $w_8$, $w_9$. The slopes are found between pixels $w_2$ and $w_3$, and between pixels $w_7$ and $w_8$.

The effect of the above algorithm is that by finding the correct adjacent values in the window, not restricted to a fixed template or number of neighbours, every thinkable combination of pixels is possible. If there is any ambiguity in neighbour selection the central pixel will keep its original spectral value.

Regarding figure A.1 the central pixel is set to $w_3$, and a maximum is found for $w_3$ and $w_8$ and a minimum for $w_2$ and $w_7$.

1. 'the one iteration smoother'
   As $w_c$ is $w_3$, $w_c$ is a part of $w_2, w_3$ and $w_3, \ldots, w_7$. As defined above, the pixels forming a segment $w_3, \ldots, w_7$ are selected.

2. 'the butcher'
   The same pixels are selected as in 1.

3. 'the iterative one'
   The first segment selected is $w_3, \ldots, w_7$. Now to find a minimum segment,

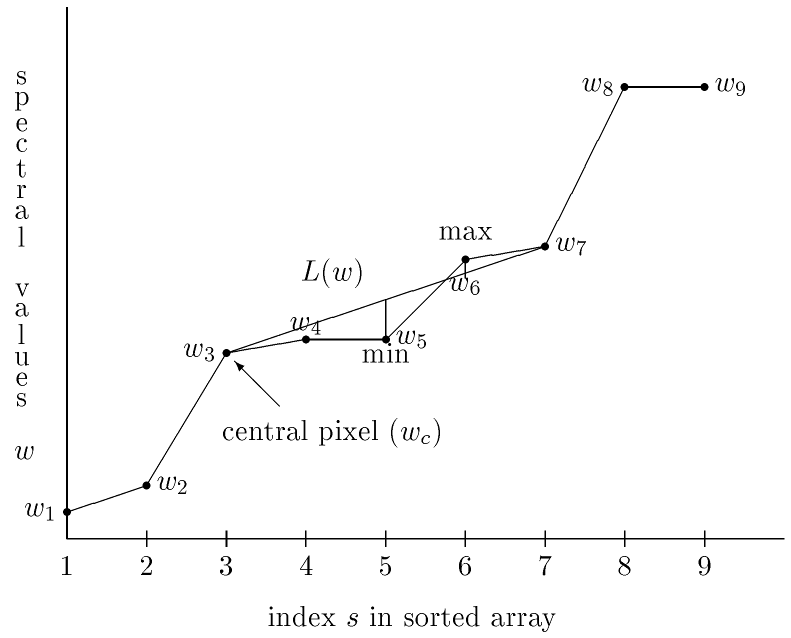



Figure A.2: the new L(s) as a function of the spectral values.

$L(s)$ is re-drawn from $w_3$ to $w_7$ as shown in figure A.2. A new maximum is found for $w_6$, a minimum for $w_5$. Because $w_3, w_5$ forms as defined a minimum segment, the pixels used here for smoothing $w_c$ are $w_3$, $w_4$ and $w_5$.

## A.3 Spectral dependency in multi-channel imagery

The functionality of the new category (and the existing filters) has been shown for one-channel imagery. What should be done with multi-channel imagery? Spectral correlation between the spectral bands of the pixels can be maintained in the following way: for every channel of a pixel the filter function is performed (i.e. the minimum/maximum and slope/segment calculation is carried out). This means that for every channel different pixel combinations for smoothing can be found. From these combinations a 'best' has to be selected. By computing some measure over the combinations per band, (e.g. the minimum variance found), a pixel combination is selected. This combination is then used to smooth all the other bands (for that pixel). Experiments should be carried out to see whether or not this results in an improvement in filtering.

The method described here can be applied of course also on the existing filters. For example the Nagao-Matsuyama filter which already computes the variance for the nine templates per channel. The template resulting in the lowest variance can be used to smooth the current pixel for all the channels.

# Appendix B

## Thresholding the edge magnitudes

Kapur *et al.* [53] define $Z$ as being the set of integer values [0,255]. The image function can then be defined as a mapping function

$$f : N \times M \to Z. \tag{B.1}$$

where $N, M$ as in equation (1.2). The brightness of a pixel is denoted as $f(i,j) = p_{i,j}$. where $p_{i,j}$ as in equation (1.2) and the number of channels is 1.
Let $t \in Z$ be a threshold and $T = \{t_0, t_1\}$ be a pair of binary gray levels and $b_0, b_1 \in Z$. The result of thresholding an image function $f(i,j)$ at gray-level $t$ is a binary image function $f_t : N \times M \to T$ such that

$$f_t(i,j) = \begin{cases} t_0 & \text{if } f(i,j) < t. \\ t_1 & \text{if } f(i,j) \geq t. \end{cases} \tag{B.2}$$

In general, a thresholding method is one that determines the value $t^*$ of $t$ based on a certain criterion.

Let the numbers of pixels with gray level $i$ be $n_i$, then the total number of pixels is

$$N \times M = n = \sum_{i=0}^{N \times M} n_i \,. \tag{B.3}$$

The probability of occurrence of gray level $i$ is defined as

$$p_i = \frac{n_i}{n} \tag{B.4}$$

where $0 \leq i \leq 255 = l$.

Two probability distributions (edge/non-edge) are derived as follows

$$\frac{p_0}{P_t}, \frac{p_1}{P_t}, \ldots, \frac{p_t}{P_t} \tag{B.5}$$

and

$$\frac{p_{t+1}}{1-P_t}, \frac{p_{t+2}}{1-P_t}, \ldots, \frac{p_l}{1-P_t} \tag{B.6}$$

where $t$ is the value of the threshold and

$$P_t = \sum_{i=0}^{t} p_i \,. \tag{B.7}$$

Define the entropy as:

$$\begin{aligned} H_b(t) &= -\sum_{i=0}^{t} \frac{p_i}{P_t} \log_e \left( \frac{p_i}{P_t} \right) \\ H_w(t) &= -\sum_{i=t+1}^{l} \frac{p_i}{1-P_t} \log_e \left( \frac{p_i}{1-P_t} \right) \end{aligned} \tag{B.8}$$

then the optimal threshold $t^*$ is defined as the gray level which maximizes $H_b(t)+H_w(t)$, that is

$$t^* = \mathrm{ArgMax}_{t \in Z} \left\{ H_b(t) + H_w(t) \right\} \ . \tag{B.9}$$

# Appendix C

## The principle of the multi-layer perceptron

Artificial Neural Networks (NN) consist of separate processing units which are inter-connected in some way. Each processing unit is often called a ‘neuron’.

The neural network used in chapter 7, is a Multi-Layer Perceptron (MLP) with back-propagation learning. The network consists of three layers of neurons. Between the separate layers the neurons are fully connected, within a layer however, the neurons are not connected, as shown in figure C.1. In figure C.1

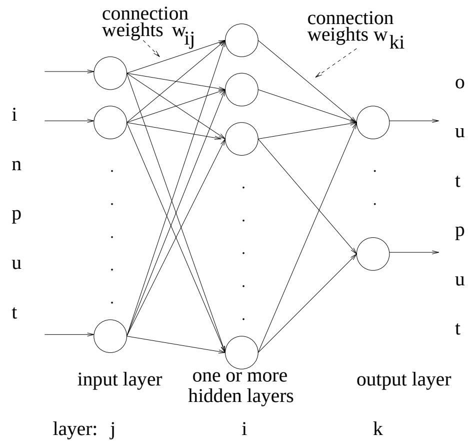



Figure C.1: the network architecture.

the three layers can be differentiated as an input layer (layer ‘$j$’), a hidden layer (layer ‘$i$’), and an output layer (layer ‘$k$’).

In figure C.2 a neuron from the hidden layer, e.g. neuron ‘$i1$’, is shown in more detail. On the left (i.e. neurons from the input layer, neuron ‘$j1$’, ‘$j2$’ and ‘$j3$’) are the inputs to the processing unit (neuron ‘$i1$’).

Each inter-connection in figure C.2 has an associated connection strength, called the weight, given as $w_{i1,j1}$, $w_{i1,j2}$ and $w_{i1,j3}$. The input to unit ‘$i1$’ is the weighted sum of the outputs of the previous layer units, i.e.

$$S_{i1} = \sum_{l} o_{jl} \cdot w_{i1,jl}. \tag{C.1}$$

Then neuron ‘$i$’ uses a nonlinear threshold function, $f_{i1}$, called activation function, to compute its output. The computed output is then sent to the neurons in a

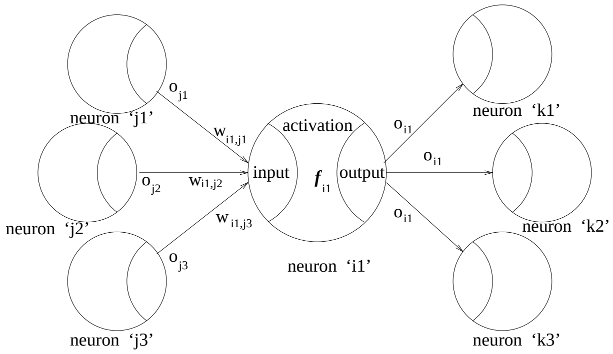



Figure C.2: simplified neuron model.

layer above (neuron '$k1$', '$k2$', '$k3$'). If more than one hidden layer is present, then the same procedure is repeated until the output layer is reached.

The activation function $f_{i1}$ is often a sigmoidal function:

$$f_{i1}(x) = \frac{1}{1 + e^{-(x)}} = o_{i1} \tag{C.2}$$

The variable $x$ is in our simplified model equal to $S_j$. The sigmoidal function is shown in figure C.3.

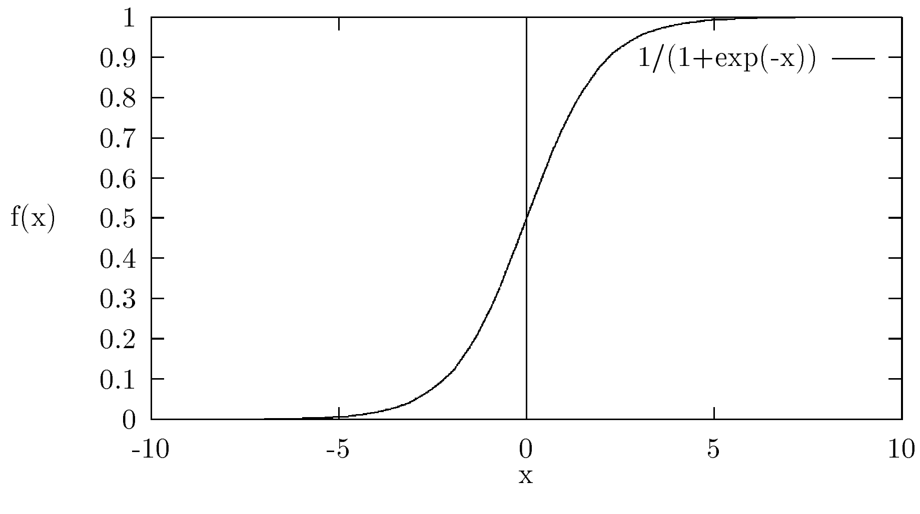



Figure C.3: the sigmoidal function.

In figure C.1 the first layer is the input layer. Normally there are as many neurons in the input layer as there are spectral bands in the input image. Normally for the input layer the weights are constant and the input values to the neurons are the spectral values of the pixels. The second layer is the 'hidden' layer. The selection of the number of neurons in the hidden layer is still a difficult problem and is mostly done by trial and error. The third layer is the output layer. Normally this layer has as many neurons as there are output classes.

Learning of the network is done via the back-propagation principle. This is a supervised learning algorithm. The back-propagation learning algorithm is an iterative procedure involving computation of output errors in a forward step followed a backward adjustment of the weights. The forward step begins with the presentation of an input pattern to the input layer, and continues as activation level calculations propagate forward through the hidden layers. In each successive layer, every processing unit sums its inputs and then applies the activation function to compute its output. The output layer then produces the output.

The backward propagation step begins with the comparison of the computed output to the target output, when the difference or the 'error' $\delta$ is computed. The backward propagation step then calculates error values for hidden units and changes their incoming weights, starting with the output layer and moving backward through the successive hidden layers.

The error value $\delta$, associated with each unit reflects the amount of error associated with that unit. This parameter is used during the weight correction procedure, while learning is taking place. A larger value for $\delta$ indicates that a larger correction should be made to the incoming weights, and its sign reflects the direction in which the weights should be changed.

Within most MLP networks with back-propagation learning, the generalized delta rule (GDR) is used. A first description of the GDR is given independently by Werbos [117] and Parker [78]. Rumelhart, Hinton and Williams [89] presented a clear and concise description of the method.

A problem of the backpropagation algorithm is its excessive learning time. No general methodology exists for determining the network configuration, i.e. the number of layers and the number of neurons per layer, and the different parameter settings, like the learning rate (a parameter determining how fast the network learns - a too high value may result in the network not converging, a too low value may lead to long learning times).

The network is tested by using a test data set which needs to be different from the set used for learning. The test set can be used to create confusion matrices which give information about the performance of the network for the different classes. If the performance is judged as 'not good enough' parameters can be adjusted and the network can be re-trained until the classification results for the test set are satisfying.

After learning and testing of the network has taken place the actual classification takes place. A pixel is presented to the network. The propagation of the pattern over the network towards the output layer results in an output value for every neuron of the output layer. The neuron having the highest output value is taken as the output class for that pixel.

# Bibliography


[1] M. Ait Belaid, G. Edwards, A. Jaton, K.P.B. Thomson, and J-M. Beaulieu. Post-Segmentation Classification of Images Containing Small Agricultural Fields. *Geocarto International*, 3:53–60, 1992.

[2] A. Asano, K. Itoh, and Y. Ichioka. The Nearest Neighbour Median Filter: Some Deterministic Properties and Implementations. *Pattern Recognition*, 23(10):1059–1066, 1990.

[3] G. Asrar, editor. *Theory and Applications of Optical Remote Sensing*. John Wiley & Sons Inc., 1989.

[4] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Inc, Englewood Cliffs, N.J., 1982.

[5] A. Baraldi and F. Parmiggiani. Segmentation Driven by an Iterative Pairwise Mutually Best Merge Criterion. Paper submitted to: IEEE Transactions on Pattern Analysis and Machine Intelligence, IMGA-CNR via Emilia Est 770, 41100 Modena, Italy, 1994.

[6] J-M. Beaulieu. Versatile and Efficient Hierachical Clustering for Picture Segmentation. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS90)*, page 1663, 1990.

[7] 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.

[8] A.S. Belward and C.V. Valenzuela, editors. *Remote Sensing and Geographical Information Systems for Resource Management in Developing Countries*. Kluwer Academic Publishers, Dordrecht-Boston-London, 1991.

[9] C.B. Benie and K.P.B Thomson. Hierarchical Image Segmentation using Local and Adaptive Similarity Rules. *International Journal of Remote Sensing*, 13(8):1559–1570, 1992.

[10] M. Berman. Aautomated Smoothing of Image and Other Regularly Spaced Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):460–468, 1994.

[11] A.C. Bovik, T.S. Huang, and D.C. Munson JR. The Effect of Median Filtering on Edge Estimation and Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):181–194, 1987.

[12] C. Brice and C. Fennema. Scene Analysis Using Regions. *Artificial Intelligence*, 1:205–226, 1970.

[13] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[14] C-I. Chang, K. Chen, J. Wang, and L.G. Althouse. A Relative Entropy-Based Approach to Image Thresholding. *Pattern Recognition*, 27(9):1275–1289, 1994.

[15] P.C. Chen and T. Pavlidis. Segmentation by Texture using a Co-Occurence Matrix and a Split-and-Merge Algorithm. *Computer, Vision, Graphics, and Image Processing*, 10:172–182, 1979.

[16] C.K. Chow and T. Kaneko. Automatic Boundary Detection of the Left Ventricle from Cineangiograms. *Computers and Biomedical Research*, 5(4):388–410, 1972.

[17] D.G. Corr, A.M. Tailor, A. Cross, D.C. Hogg, D.H. Lawrence, D.C. Mason, and M. Petrou. Progress in Automatic Analysis of Multi-Temporal Remotely-Sensed Data. *International Journal of Remote Sensing*, 10(7):1175–1195, 1989.

[18] A.M. Cross, D.C. Mason, and S.J. Dury. Segmentation of Remotely-Sensed Images by a Split-and-Merge Process. *International Journal of Remote Sensing*, 9(8):1329–1345, 1988.

[19] E.R. Davies. Constraints on the Design of Template Masks for Edge Detection. *Pattern Recognition Letters*, 4:111–120, 1986.

[20] L.S. Davis and A. Rosenfeld. Noise Cleaning by Iterated Local Averaging. *IEEE Transactions on Systems, Man, and Cybernetics*, 8:705–710, 1978.

[21] F. Deravi and S.K. Pal. Grey Level Thresholding Using Second Order Statistics. *Pattern Recognition Letters*, 1:417–422, 1983.

[22] R. Deriche. Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector. *International Journal of Computer Vision*, 1:167–187, 1987.

[23] P.A. Devijver and J. Kittler. *Pattern Recognition.* Prentice-Hall, Inc, Englewood Cliffs, N.J., 1982.

[24] N.A. Drake and J.J Settle. Linear Mixture Modelling of Thematic Mapper Data of the Peruvian Andes. In *Proceedings of the Earsel Conference*, pages 490–495, Espoo, Finland, 1989.

[25] J.M.H. Du Buf and T.G. Campbell. A Quantitive Comparision of Edge-Preserving Smoothing Techniques. *Signal Processing*, 21:289–301, 1990.

[26] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons Inc., 1971.

[27] C. Elachi. *Introduction to the Physics and Techniques of Remote Sensing.* John Wiley & Sons Inc., 1987.

[28] A. Elkardoudi and P. Janssen. Parallelisation of the JRC-filters on a Unix Based Computer Network. Internal Report TTP project, Informatica, Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, November 1993.

[29] ERDAS. ERDAS Manual, Versions 7.2, 7.4 and 7.5. ERDAS, Inc. Atlanta, GA, USA, 1991.

[30] Eurostat. Statistical Yearbook. Published by Eurostat, Luxemburg, 1988.

[31] M.M. Fleck. Some Defects in Finite-Difference Edge Finders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):337–345, 1992.

[32] M.R.B. Forshaw. Speeding up the Marr-Hilldreth Operator. *Computer, Vision, Graphics, and Image Processing*, 41:172–185, 1988.

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

[34] K.S. Fu and J.K. Mui. A Survey on Image Segmentation. *Pattern Recognition*, 13:3–16, 1981.

[35] G. Genovese, M. Millot, and R.P.H.M. Schoenmakers. Segmentation and Interpretation of NOAA-AVHRR Multi-Temporal NDVI-Based Data Sets in Temperate Regions. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS94)*, pages 2525–2527, Pasadena, CA, USA, August 8-12 1994.

[36] R.C. Gonzalez and P. Wintz. *Digital Image Processing.* Addison-Wesley, Reading, Mass, 1977.

[37] G.E.L. Grimson and E.C. Hildreth. Comments on 'Digital Step Edges from Zero Crossings of Second Directional Derivatives'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):121–126, 1985.

[38] R.M. Haralick. Digital Step Edges from Zero Crossing of Second Directional Derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):58–68, 1984.

[39] R.M. Haralick and L.G. Shapiro. Survey: Image Segmentation Techniques. *Computer, Vision, Graphics, and Image Processing*, 29:100–132, 1985.

[40] A.R. Harrison, S.J. Hurcom, and M. Taberner. Examining Spectral Mixture Modelling using in situ Spectroradiometric Measurements. In *Proceedings of the Conference Spatial Data 2000*, pages 242–249, Christ Church, Oxford University, England, Sep 17-20 1991.

[41] C. Hendriks and R. Schreurs. Parallelisation of the JRC-filters on the Nijmegen Super Cluster. Internal Report TTP project, Informatica, Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, September 1992.

[42] M.H. Hueckel. An Operator which Locates Edges in Digitized Pictures. *Journal of the Association of Computing Machinery*, 18(1):113–125, 1971.

[43] M.H. Hueckel. A Local Visual Operator Which Recognizes Edges and Lines. *Journal of the Association of Computing Machinery*, 20(4):634–647, 1973.

[44] J.R. Irons, B.L. Markham, R.F. Nelson, D.L. Toll, and D.L. Williams. The Effects of Spatial Resolution on the Classification of Thematic Mapper Data. *International Journal of Remote Sensing*, 6(8):1385–1403, 1985.

[45] L.L.F. Janssen. *Methodology for Updating Terrain Object Data from Remote Sensing Data*. PhD thesis, Landbouw Universiteit Wageningen, The Netherlands, January 19 1994.

[46] L.L.F. Janssen and J.D. van Amsterdam. An Object Based Approach to the Classification of Remotely Sensed Images. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS91)*, pages 2191–2195, Espoo, Finland, June 3-6 1992.

[47] L.L.F. Janssen, R.P.H.M. Schoenmakers, and R.G. Verwaal. Integrated Segmentation and Classification of High Resolution Satellite Images. In *Workshop on Multisource Data Integration in Remote Sensing for Land Inventory Applications IAPR TC-7*, pages 65–83, Delft, The Netherlands, september 7-9 1992.

[48] R.H. Jeurissen. Syllabus Lectures on Graph Algorithms. Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, 1983.

[49] G. Johanssen and J. Bille. A Threshold Selection Method Using Information Measures. In *Proceedings 6-th International Conference Pattern Recognition*, pages 140–142, Munich, Germany, 1982.

[50] I. Kanellopoulos, A. Varfis, G.G. Wilkinson, and J. Megier. Land Cover Discrimination in SPOT HRV Imagery Using an Artificial Neural Network - a 20 Class Experiment. *International Journal of Remote Sensing*, 13(5):917–924, 1992.

[51] I. Kanellopoulos and G.G. Wilkinson. Experiments with Backpropagation Neural Networks for Image Classification. Technical Report I.90.64, Joint Research Centre, Laboratory for Image Processing, I-21020, Ispra (Varese), Italy, 1990.

[52] I. Kanellopoulos, G.G. Wilkinson, and J. Megier. Segmentation of Remotely-Sensed Images Using Best-Merge Region Growing and Spectral-Spatial Clustering with Spatial Context Rules. Technical Report I.91.119, Joint Research Centre, Laboratory for Image Processing, I-21020, Ispra (Varese), Italy, 1991.

[53] N. Kapur, P.K. Sahoo, and A.K.C. Wong. A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram. *Computer, Vision, Graphics, and Image Processing*, 29:273–285, 1985.

[54] R.L. Kettig and D.A. Landgrebe. Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects. *IEEE Transactions on Geoscience Electronics*, 14(1):19–26, 1976.

[55] R.A. Kirsch. Computer Determination of the Constituent Structure of Biomedical Images. *Computers and Biomedical Research*, 4:315–328, 1971.

[56] J. Kittler and J. Illingworth. On Threshold Selection Using Clustering Criteria. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(5):652–655, 1985.

[57] M.S. klein Gebbinck. Decomposition of Mixed Pixels. Private Communication, Informatica, Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, September 1994.

[58] A. Klinger. Datastructures and Pattern Recognition. In *Proceedings of the First International Joint Conference Pattern Recognition*, pages 479–498, Washington D.C., October 1973.

[59] M. Kunt. Source Coding of X-Ray Pictures. *IEEE Transactions on Biomedical Engineering*, 25(2):121–138, 1978.

[60] V. Lacroix. A Three Module Strategy for Edge-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):803–810, 1988.

[61] J. Le Moigne and J.C. Tilton. Refining Image Segmentation by Integration of Edge and Region Data. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS92)*, pages 1406–1408, Houston, Texas, USA, May 26-29 1992.

[62] J.S. Lee. Digital Image Enhancement and Noise Filtering by Use of Local Statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:165–168, 1980.

[63] T.M. Lillesand and R.W. Kiefer. *Remote Sensing and Image Interpretation.* John Wiley & Sons Inc., 1987.

[64] A. Lopez, E. Nezry, R. Touzi, and H. Laur. Structure Detection and Adaptive Speckle Filtering in SAR Images. *International Journal of Remote Sensing*, 14(9):1735–1758, 1993.

[65] D. Marr and E. Hilldreth. Theory of Edge Detection. In *proceed. R. Soc. Lond. B 207*, pages 187–217, 1980.

[66] G.A. Mastin. Adaptive Filters for Digital Noise Smoothing: An Evaluation. *Computer, Vision, Graphics, and Image Processing*, 31:103–121, 1985.

[67] P.M. Mather. *Computer Processing of Remotely-Sensed Images: An Introduction.* John Wiley & Sons Inc., 1987.

[68] P. Meer, R-H. Park, and K. Cho. Multiresolution Adaptive Image Smoothing. *cvgip: Graphical Models and Image Processing*, 56(2):140–148, 1994.

[69] P. Meer, S. Wang, and Wechsler H. Edge Detection by Associative Mapping. *Pattern Recognition*, 22(5):491–503, 1989.

[70] L. Mero and Z. Vassy. A Simplified and Fast Version of the Hueckel Operator for Finding Optimal Edges in Pictures. In *Proceedings 4th International Conference on Artificial Intelligence*, pages 650–655, Tblisi, Georgia USSR, 1975.

[71] J.M. Meyer-Roux. The Ten Year Research and Development Plan for the Application of Remote Sensing in Agriculture Statistics. Technical Report SP.1.87.39/EN, Commission of the European Communities, Joint Research Centre, Institute for Remote Sensing Applications, I-21020, Ispra (Varese), Italy, 1987.

[72] J. Muerle and D. Allen. Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene. In G. Cheng et al., editor, *Pictorial Pattern Recognition*, pages 3–13. Thompson, Washington, D.C., 1968.

[73] N. Nagao and M. Matsuyama. Edge Preserved Smoothing. *Computer Graphics and Image Processing*, 9:394–407, 1979.

[74] A.M. Nazif and M.D. Levine. Low Level Image Segmentation -An Expert System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):555–577, 1984.

[75] E. Nezry, H-G. Kohl, and H. de Groof. Restoration of Textural Properties in SAR Images using Second Order Statistics. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS94)*, pages 2165–2167, Pasadena, California, USA, August 8-12 1994.

[76] R-H Park. A Fourier Interpretation of the Frei-Chen Edge Masks. *Pattern Recognition Letters*, 11(9):631–636, 1990.

[77] R-H Park and W Choi. Comments on a Three Module Strategy for Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):223–224, 1990.

[78] D.B. Parker. Optimal Algorithms for Adaptive Networks: Second Order Backpropagation, Second Order Direct Propagation, and Second Order Hebbian Learning. In *Proceedings of IEEE the first international Conference on Neural Networks*, San Diego, CA, 1987.

[79] T. Pavlidis. Segmentation of Pictures and Maps Through Functional Approximation. *Computer Graphics and Image Processing*, 1:360–372, 1972.

[80] T. Pavlidis. A Critical Survey of Image Analysis Methods. In *Proceedings of the Eight International Conference on Pattern Recognition (IAPR-8)*, pages 502–511, 1986.

[81] T. Pavlidis and Y-T. Liow. Integrating Region Growing and Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, 1990.

[82] W.A. Perkins. Area Segmentation Images using Edge Points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):8–15, 1980.

[83] I. Pitas and A.N. Venetsanopoulos. Order Statistics in Digital Image Processing. *Proceedings of the IEEE*, 80(12):1893–1921, 1992.

[84] J.M.S. Prewitt. Object Enhancement and Extraction. In B.S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 75–149. New York: Academic Press, 1970.

[85] L.G. Roberts. Machine Perception of Three Dimensional Solids. *Optical and Electro-Optical Image Processing*, pages 159–197, 1970.

[86] T.V. Robertson. Extraction and Classification of Objects in Multispectral Images. In *Machine Processing of Remotely Sensed Data*, pages 3b–27–3b–34, IEEE 73 CHO 837-2GE, Purdue University, West Lafayette, Indiana, October 16-18 1973.

[87] G.S. Robinson. Edge Detection by Compass Gradient Masks. *Computer Graphics and Image Processing*, 6:492–501, 1977.

[88] A. Rosenfeld and I. De La Torre. Histogram Concavity Aanalysis as an Aid in Threshold Selection. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):231–235, 1983.

[89] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representations by Error Propagation. In Rumelhart D.E. and J.L. McClelland, editors, *Parallel Distributed Processing*, pages 318–362. The MIT press, Cambridge, Massachusetts, 1986.

[90] D.E. Rumelhart and J.L. McClelland, editors. *Parallel Distributed Processing*. The MIT press, Cambridge Massachusetts, 1986.

[91] P. Sahoo, S. Soltani, and A. Wong. A Survey of Thresholding Techniques. *Computer, Vision, Graphics, and Image Processing*, 41:223–260, 1988.

[92] B.J. Schachter, L.S. Davis, and A. Rosenfeld. Some Experiments in Image Segmentation by Clustering of Local Feature Values. *Pattern Recognition*, 11(1):19–28, 1979.

[93] R.P.H.M. Schoenmakers. Automatic Field Boundary Extraction in Optical Remotely Sensed Imagery by a Hybrid Segmentation Method. In *Workshop on Multisource Data Integration in Remote Sensing for Land Inventory Applications IAPR TC-7*, pages 171–181, Delft, The Netherlands, September 7-9 1992.

[94] R.P.H.M. Schoenmakers and J.H.T. Stakenborg. Adaptive Filtering Using Gradient Morphology. In *Proceedings of the Eleventh International Conference on Pattern Recognition (IAPR-11)*, pages C240–C243, The Hague, The Netherlands, August 30-september 3 1992.

[95] R.P.H.M. Schoenmakers, G.G. Wilkinson, and Th.E. Schouten. Multi-Temporal Image Segmentation Using a Distributed-Memory Parallel Computer. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS92)*, pages 1114–1116, Houston, Texas, May 26-29 1992.

[96] R.P.H.M. Schoenmakers, G.G. Wilkinson, and Th.E. Schouten. Use of Landsat-TM Image Segmentation for Smoothing ERS-1 SAR Imagery in Combined Multi-Sensor Landscape Classification. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS93)*, pages 1228–1230, Tokyo, Japan, August 18-21 1993.

[97] Th.E. Schouten. Adding Edges to Regions, Private Communication. Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, 1993.

[98] Th.E. Schouten, M. klein Gebbinck, R.P.H.M. Schoenmakers, and G.G. Wilkinson. Finding Thresholds for Image Segmentation. In *Proceedings of The European Symposium on Satellite Remote Sensing*, Roma, Italy, September 26-30 1994.

[99] R.A. Schowengerdt. *Techniques for Image Processing and Classification in Remote Sensing.* Academic Press, 1983.

[100] R. Schreurs. Afstudeer Scriptie no 297: Correctness and Efficiency of the Best Merges Region Growing. Master's thesis, Katholieke Universiteit Nijmegen, Faculty of Computer Science, Realtime Systems, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, november 1993.

[101] A.R. Smith. Tint Fill. In *Proceedings 6th Annual Conference on Computer Graphics and Interactive Techniques-Papers- Association for Computing Machinery*, pages 276–283, Chicago, Il, August 1979.

[102] M. Spann and A. Nieminen. Adaptive Gaussian Weighted Filtering for Image Segmentation. *Pattern Recognition Letters*, 8:251–255, 1988.

[103] J.H.T. Stakenborg. Exploring the Spatial Domain. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS87)*, pages 125–130, The University of Michigan, Ann Arbor, Michigan, May 18-21 1985.

[104] H.S. Swain and Davis S.M. *Remote Sensing: The Quantative Approach.* Mc Graw-Hill, 1978.

[105] H.D. Tagare and R.J.P. deFigueiredo. On the Localization Performance Measure and Optimal Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1186–1190, 1990.

[106] J.C. Tilton. Multiresolution Spatially Constrained Clustering of Remotely Sensed Data on the Massively Parallel Processor. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS84)*, pages 661–666, Strasbourg, France, 1984.

[107] J.C. Tilton. Image Segmentation by Iterative Parallel Region Growing and Splitting. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS89)*, pages 2235–2238, Vancouver, Canada, 1989.

[108] J.C. Tilton. Experiences Using TAE Plus Command Language for an Image Segmentation Algorithm Interface. In *Proceedings of TAE Ninth User's Conference*, pages 295–312, New Carrolton, MD, USA, Nov. 5-7 1991.

[109] J.C. Tilton and S.C. Cox. Segmentation of Remotely Sensed Data Using Parallel Region Growing. In *Ninth International Symposium on Machine Processing of Remotely Sensed Data*, pages 130–137, Purdue University, West Lafayette, Indiana, USA, 1983.

[110] J. Ton, J. Sticklen, and A.K. Jain. Knowledge-Based Segmentation of Landsat Images. *IEEE Transactions on Geoscience and Remote Sensing*, 29(2):222–232, 1991.

[111] F.A.G. Verstraaten. Afstudeer Scriptie no 294: Segmentation of Large Satellite Images. Master's thesis, Katholieke Universiteit Nijmegen, Faculty of Computer Science, Realtime Systems, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, september 14 1993.

[112] L.G. Vuurpijl. Experiments Carried Out with the Satellite Project. Technical report, Katholieke Universiteit Nijmegen, Faculty of Mathematics and Informatics, Mathematics and Informatics, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, 1993.

[113] L.G. Vuurpijl and Th.E. Schouten. PREENS, a Parallel Research Execution Environment for Neural Systems. In *High Performance Computing and Networking '94*, München, April 1994.

[114] Louis Vuurpijl, Theo Schouten, and Jan Vytopil. CONVIS: Action Oriented Control and Visualization of Neural Networks. Technical report, Katholieke Universiteit Nijmegen, Faculty of Mathematics and Informatics, Informatics for Technical Applications, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, November 1994.

[115] D. Wang, A. Vagnucci, and C.C. Li. Gradient Inverse Weighted Smoothing Scheme and the Evaluation of its Performance. *Computer Graphics and Image Processing*, 15:167–181, 1981.

[116] D. Wang and O. Wang. A Weighted Averaging Method for Image Smoothing. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 981–983, Miami Beach, Florida, USA, 22-26 June 1986.

[117] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[118] G.G. Wilkinson and I. Kanellopoulos. The Application of Neural Network Algorithms in Image Processing for Remote Sensing, Theoretical Basis. Technical Report I.90.27, Joint Research Centre, Laboratory for Image Processing, I-21020, Ispra (Varese), Italy, 1990.

[119] M. Willebeek-LeMair and A.P. Reeves. Region Growing on a Highly Parallel Mesh-Connected SIMD Computer. In *Proceedings of the Second Symposium on the Frontiers of Massively Parallel Computation*, Fairfax, VA, USA, Oct 10-12 1988.

[120] C. Woodcock and V.J. Harward. Nested-Hierarchical Scene Models and Image Segmentation. *International Journal of Remote Sensing*, 13(16):3167–3187, 1992.

[121] R.C. Wrigley, D.H. Card, C.A. Hlavka, J.R. Hall, F.C. Mertz, C. Archwamety, and R.A. Schowengerdt. Thematic Mapper Image Quality: Registration, Noise, and Resolution. *IEEE Transactions on Geoscience and Remote Sensing*, 22(3):263–270, 1984.

[122] A.Y. Wu, T-H. Hong, and A. Rosenfeld. Threshold Selection Using Quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(1):90–94, 1982.

[123] S.W. Zucker. Region Growing: Childhood and Adolescence. *Computer Graphics and Image Processing*, 5(3):382–399, 1976.

# Samenvatting

Segmentatie is een process dat pixels van ieder type digitaal beeld groepeert conform gebruiker gedefinieerde karakteristieken door gebruik te maken van de spectrale waardes van deze pixels en hun buurschap. In dit proefschrift, een segmentatie methode voor optische telegedetecteerde beelden is gepresenteerd welke nieuw ontwikkelde en verbeterde bestaande methodes, i.e. rand detectie, rand verbinding, regio groeiing, integreert in een hybride methode met het doel klassifikatie nauwkeurigheid in land gebruik toepassingen te verhogen en vooral voor landbouw. Bestaande segmentatie methodes, de aangepaste en de ontwikkelde worden besproken en vergeleken. Voor twee segmentatie methoden, rand detectie en regio-groeiing, worden zowel de voor- als de nadelen besproken. Voor regio-groeiing is aangetoond dat volgorde afhankelijkheid en te groot groeien van een regio de hoofdproblemen zijn waar voor rand detectie het is aangetoond dat niet altijd de volledige randen worden gedetecteerd. In de literatuur is aangegeven dat een combinatie van beide methodes een toename in segmentatie nauwkeurigheid kan betekenen. Daarom is een hybride methode ontwikkeld die in het kort als volgt beschreven kan worden: randen worden gedetecteerd door gebruik te maken van een gradient gebaseerde rand detector. De verscheidene rand fragmenten worden dan verbonden door missende rand pixels toe te voegen. Op deze manier worden gesloten polygonen gecreëerd. Rand fragmenten die niet tot een gesloten polygon behoren worden verwijderd. Binnenin zo'n polygon wordt regio-groeiing toegepast. In een post segmentatie stadium, worden gevonden randen aan de regios gedetecteerd door de regio-groeiing, toegevoegd. In een pre-segmentatie stadium wordt ruis verwijderd door gebruik te maken van een nieuw ontwikkeld filter gebaseerd op de morfologie van de gesorteerde grijs waardes in een beeld venster.

De gebruikte "Lacroix" rand detectie methode had twee nadelen waarvoor een oplossing gevonden moest worden. De eerste was dat hij ontwikkeld was voor èèn kanaal beelden. De tweede was zijn afnemende gevoeligheid voor hoge intensiteit waardes van de pixels.

Door het feit dat de rand detectie niet altijd gesloten randen vindt, moest een rand verbindings methode worden ontwikkeld welke rand pixels toevoegd daar waar tussen rand fragmenten een rand pixel miste zodanig dat gesloten randen gevormd worden. Overblijvende onverbonden rand fragmenten worden verwijderd. Het verbindings process maakt gebruik van graph data structuren om rand pixels en hun buurrelaties te representeren.

De best merge (Tilton and Beaulieu & Goldberg) regio-groeiing algorithme voorkomt het volgorde afhankelijkheidsprobleem door alleen die buurparen te

groeperen in iedere iteratie zodanig dat een groeperings criterium vervuld wordt terwijl de groeperings kosten geminimaliseerd worden. In dit proefschrift echter wordt aangetoond dat in het geval dat verschillende paren dezelfde groeperings kosten hebben, ook dit algorithme volgorde afhankelijk is. Een methode is ontwikkeld om dit op te lossen. Echter het is aangetoond dat deze groeperings procedure een ongewenst zij effect kan hebben waarin pixels die op een beeld hellingen liggen onjuist gegroepeerd kunnen worden. Echter voor teledetectie beelden schijnt dit geen serieus nadeel te zijn.

Het voorgestelde algorithme kan volledige landschap beelden die tussen de 38 Mb (SPOT-P) en ongeveer 250 Mb (Landsat-TM) kan beslaan, segmenteren. Om deze aantallen data te verwerken, gebruikt de methode een beeld opdeel techniek op zo'n manier dat uitwisselen van data van schijf naar hoofd geheugen en vice versa onafhankelijk gebeurt van het besturings systeem.

Rand detectie en rand verbinding om het beeld onder te verdelen in onafhankelijke gesloten polygons wordt gebruikt om de benodigde computer tijd te verminderen. Dit maakt het voor de regio groeiings methode mogelijk om in onafhankelijke processen onderverdeeld te worden die gebaseerd worden op iedere onafhankelijke polygoon.

Dit maakt een eenvoudige implementatie op een parallelle computer mogelijk.

Resultaten van de hybride segmentatie methode zijn gepresenteerd voor Landsat-TM en SPOT-panchromatische beelden. Het gebruik van de segmentatie methode is geillustreerd in drie teledectie contexten. 1) De combinatie van segmentatie van Landsat-TM en een meerlagig perceptron neural netwerk for land gebruik inventarisatie. 2) Het gebruik van segmentatie van Landsat-TM voor speckle filteren van een geo-gerefereerd ERS-1/SAR beeld. 3) Het gebruik van segmentatie van NOAA/AVHRR NDVI beeld voor het extraheren van vegetatie profielen.

# Sommario

La segmentazione e un processo nel quale gruppi di pixel di un imagine di qualsiasi tipo conforme alle caratteristiche definite dall utente usando i valori spettrali di questi pixel e la loro vicinanza. In questa tesi di dottorato e presentato un metodo di segmentazione di immagini ottiche ottenute tramite telerilevamento il quale integra metodi nuovi e gia esistenti, come edge detection, edge linking e region growing, in un metodo ibrido con lo scopo di aumentare l'accuratezza della classificazione nelle applicazioni riguardanti il territorio in paricolar modo l'agricolura. I metodi di segmentazione esistenti, sia quelli gia esistenti che quelli svilupati sono discussi e confrontati. Sono descritti i vantaggi e gli svantaggi dei due metodi di segmentazione, edge detection e region growing. Per region growing e mostrato che order dependence e region overgrowing sono i problemi principali, e per edge detection e mostrato che i completi di confini dei campi non sono sempre rilevati. E stato indicato nei testi scritti che la combinazione dei due metodi puo aumentare l' accuratezza della segmentatzione. Per questo motivo un metodo ibrido e stato svillupato il quale puo essere brevemente discritto come segue: edges sono rilevati usando un gradient based edge detector. I frammenti separati di edge sono collegati aggiungendoglii edge pixel mancanti. In questo modo poligoni chiusi sono creati. I frammenti di edge non appartenenti ad un poligono chiuso sono cancellati. Dentro i poligoni region growing e realizzato. In uno stadio di post-segmentazione (rilevamento) edges sono agguinti alle regioni trovate dal region growing. In uno stadio di pre-segmentazione il disturbo e eliminato usando un nuovo filtro basato sulla morfologia dei livelli di grigio ordinate in una finestra.

Il metodo di edge detection "Lacroix" usato aveva due svantaggi per i quali bisognava trovare delle soluzioni. Il primo svantaggio era lo sviluppo solo per una singola banda di immagine, il secondo era la diminuzione di sensibilita per valori dei pixels molto intensi.

A causa del fatto che edge detection non sempre trova confini chiusi, un metodo di collegamento di edge e stato svilupato. Questo metodo aggiunge edge-pixel dove manca un singolo edge pixel per creare confini chiusi. I frammenti non collegati sono rimossi. Il processo di collegamento degli edge usa strutture di dati grafiche per rappresentare gli edge pixels e le loro relazioni.

L'algoritmo di region growing best merge oltrepassa il problema del order dependence unendo quelle coppie di regioni.....

Ma in questa tesi di dottorato e mostrato che quando alcune coppie hanno gli stessi bassi costi di unione anche questo algoritmo e order dependent. Un metodo e stato svillupato per rissolvere questo. Communque e mostrato che questa procedure di uniione puo avere un effetto laterale non voluto in cui rampe di immagini

sono unite in maniera inappropriata. Communque per le immagini ottiche ottenute tramite telerilevamento questo no sembra essere un serio svantaggio.

L'algoritmo proposto a capace di segmentare full scene immagine che possano occupare da 38 Mb (SPOT-P) fino a circa 250 MB (Landsat-TM). Per manipolare questa quantita di dati il metodo svilupato usa un slicing technique utillizato dal algoritmo independentemente dal sistema operativo che permette uno swapping dal disco alla RAM e vice versa.

Per aumentare la velocita di esecuzione l'uso del edge detection e del edge linking per dividere l'immagine in poligoni independenti permette al region growing di essere diviso in separati processi per ogni singolo poligono.

Questo crea un implementazione diretta su un machina multi-processore.

I risultati del metodo di segmentatzione ibrido sono presentati per un Landsat-TM e un SPOT-P... L'uso del metodo di segmentazione e mostrato in 3 contesti di telerilevamento. 1) La combinazione di un immagine Landsat-TM e un multi-layer perceptron rete neurale per linventorio dei territori. 2) L'uso della segmentazione di un immagine Landsat-TM per speckle filtering di un georeferenced ERS-1/SAR immagine. 3) L'uso della segmentazione di un immagine NOAA/AVHRR NDVI per estrazione dei profili vegetali.

# CURRICULUM VITAE

Ron Schoenmakers was born on October 23, 1961 at Brunssum, The Netherlands. In 1974 he started his “Atheneum-B” education at the “Rombouts college” Brunssum, which he finished in 1981. In that same year he went to Nijmegen (NL) to start his studies in “computer science” at the “Katholieke Universiteit”. In 1984 he received his “kandidaats degree”. He concluded his studies in 1987, at the “real-time systems” department under supervision of professor Jan Vytopil. The title of his master’s thesis was “Automatic processing of high speed films” which was undertaken in collaboration with the section of “Biomechanics” of the department of “Orthopediatrics” of “Radboud hospital” at Nijmegen. Then he worked for 6 weeks at the real-time systems department of the faculty Mathematics and Informatics as a software-hardware maintainer. In May 1987 he joined the Royal Dutch Navy, his education allowing him the rank of officer (Luitenant ter zee III, speciale diensten), stationed at the Royal Navy base Den Helder (NL). After this he was detached to the Navy’s airbase at Valkenburg (NL) where he was in charge of developing software for sonar frequency distinction and the computation of the area covered by a pattern of buoys dropped on the sea surface. During his military service he applied for a job at “Getronics” Amsterdam (NL), where he was employed as a programmer/technical developer on the companies administrative database management system. In October, 1989, he started a 4 year Ph.D grant at the Joint Research Centre of the European Union at Ispra (VA) Italy. In August 1993, he received an “IEEE travel award” for work in the field of Remote Sensing. Before starting to work as an consultant for Telespazio Roma (I), March 1994, he worked for SIS te Gendt on the development of segmentation software. At telespazio he worked on the application of neural networks in remote sensing. Since March 1995, he works as a managing director of Cortese s.a.s. Ispra (VA). Most of the work carried out during the last 5 and a half years are given in this thesis.