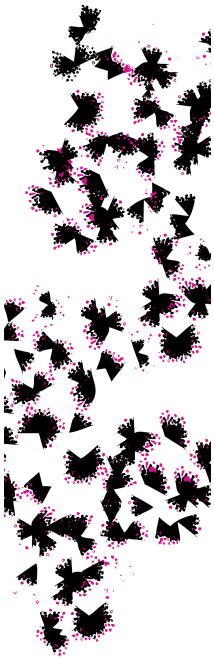


A Front-end Application for Markov Random Field-based Texture Image Segmentation

G.R. (Remco) Huizenga

MSc Report

Committee:
Prof.dr.ir. S. Stramigioli
Dr.ir. F. van der Heijden
Prof.dr.ir. C.H. Slump
Dr.ir. J.N. Driessen
Ir. E. Molenkamp

March 2015

007RAM2015
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Accurately segmenting texture images that are characterized by globally varying patterns that belong to the same class is a challenge, especially when the images are large or highly susceptible to noise. Such textures appear for example in radar images and in H&E stained pathology images. Markov Random Fields segmentation is a powerful technique that is able to take into account information of the surroundings of a pixel to infer the most likely class it belongs to. However, Markov Random Fields work on pixel level which makes it very computationally demanding, especially for large images or when fast processing times are required. Furthermore, working on pixel level means that it is hard to use accurate texture features during segmentation. This document aims to overcome these drawbacks of Markov Random Field segmentation by providing a front-end based on the concept of superpixels, the output of which can be used as input to the Markov Random Field segmentation algorithm.

Acknowledgements

I would like to express the deepest appreciation to everyone in my surroundings who has supported me in any way over the course of this project, a number of which I would like to name here.

I would like to thank Thales Nederland B.V. for allowing me the opportunity to do my internship at their branch in Hengelo and subsequently my graduation project. A lot of the work done for my graduation project has been done here. In particular, I would like to thank my direct supervisor at Thales Nederland B.V., Hans Driessen, for his assistance and guidance with this thesis.

I would like to thank my direct supervisor at the University of Twente, Ferdi van der Heijden, for making it possible for me to carry out my graduation project at Thales Nederland B.V. and for his continued guidance and assistance, especially when it comes to writing MatLab code.

I would also like to thank Oscar Geessink for all his help. He provided me with a different, less mathematical, way of looking at H&E stained pathology images, took the time to have numeral discussions with me and provided me with the ground truth images.

Next I would like thank my parents for believing in me and for all their love and support over the course of not only my graduation project but my entire study duration.

Last, but definitely not least, I would like to thank Sophie Ahlers for being the best girlfriend I could wish for. It has not always been easy on me and therefore on her, but she has always believed in me and supported me in every way possible.

Inhoudsopgave

1 Introduction **1**

2 A brief to Markov Random Field based texture segmentation **2**

2.1 Markov Chains . . . 2

2.2 Markov Random Fields . . . 4

2.2.1 Neighborhood system . . . 5

2.2.2 Cliques . . . 5

2.2.3 Gibbs distribution . . . 7

2.2.4 Hammersley-Clifford theorem . . . 8

2.3 MRF's in image analysis . . . 8

2.3.1 Texture synthesis . . . 8

2.3.2 Sampling . . . 9

2.4 Requirements for pre-processing methods . . . 10

3 Methods for pre-processors for MRF-based texture segmentation **11**

3.1 Pre-processing . . . 11

3.2 Existing superpixel algorithms . . . 12

3.2.1 Mean shift . . . 12

Discontinuity preserving filtering . . . 12

Mean shift clustering . . . 12

Discussion . . . 13

3.2.2 Watershed . . . 14

Top-down . . . 14

Bottom-up . . . 14

Discussion . . . 14

3.2.3 Turbopixels . . . 14

Discussion . . . 16

3.2.4 Normalized graph-cuts . . . 16

Discussion . . . 17

3.2.5 Comparison . . . 17

4 Simple Linear Iterative Clustering **18**

4.1 The algorithm . . . 18

4.1.1 Calculation of the distance . . . 21

4.2 Improvements and extensions . . . 21

4.2.1 Dynamic weighting . . . 21

4.2.2 Sigma filtering . . . 22

4.2.3 Segment merging 23
4.2.4 DBSCAN 24

5 Results **26**
5.1 Measurement algorithm 26
5.1.1 Demo 30

6 Discussion **33**

7 Conclusion and future work **35**

Bibliografie **37**

1 Introduction

In processing the images generated by radar systems, e.g. surveillance radar, extracting detailed information from the surroundings of the radar can be very useful. For objects that are clearly visible in the images, i.e. objects with a large signal to clutter ratio like large or fast objects, this is not a problem, but detection of objects with a small signal to clutter ratio is challenging. In this case, extracting detailed information from the surroundings of the radar can be helpful in the detection of targets. Further complications arise when textures appearing in the radar images are influenced by, for example, currents, wind, water depth or changes in resolution. This results in textures that locally have a fixed orientation but for which this orientation varies globally in an unpredictable way. Then the statistics of the textures can vary per region or, in the case of changing resolution, can be graded. The radar images shown in this document have been provided by Thales Hengelo B.V.

The same type of textures appears in H&E (Hematoxy&Eosin) stained pathology images in so-called stroma. Since research shows that the tumor-stroma ratio in H&E stained sections is a prognostic marker in the diagnoses and survival of a patient, a reliable estimation of the tumor-stroma ratio is desired. This estimation can help a pathologist in better distinguishing between patients with poor or a somewhat better prognosis. The H&E stained pathology images have been provided by LabPON, Laboratory for Pathology in Hengelo.

Many different texture segmentation methods exist. One class of texture segmentation methods is those using probabilistic models and an example of this is segmentation based on Markov Random Fields. Since segments in textured images can have a fixed orientation locally but a globally varying orientation, Markov Random Fields work very well because it gives the ability to use global information to aid in the inference of the local segment class. However, in general, the complexity of algorithms increases with the number of pixels in the image. This is particularly true for methods based on Markov Random Fields which can become computationally intractable if the images are getting larger. Therefore the size of the images is a limiting factor. Especially the pathology images are very large (Millions of pixels) and things are further complicated by the fact that the information in the images should not decrease. This implies that a method is needed to reduce the number of pixels for the Markov Random Field algorithm while not losing any information in the images.

This document provides a front-end application that effectively reduces the number of pixels in an image by a large amount while still maintaining the information present in the image. It is expected that the output of this front-end application can then be used as input for a Markov Random Field based segmentation algorithm in such a way that the algorithm runs much faster and the generated superpixels can be used to extract more meaningful (texture) features from the data as opposed to just using the image pixels as input. To test the front-end and to illustrate the broad applicability of it, both radar images and H&E stained pathology images are used since they both contain textures with the characteristics described above. This does, however, not mean that this front-end application or Markov Random Fields based segmentation is confined to just these areas.

2 will give an introduction to Markov Random Fields and show why a pre-processing step is often required when working with Markov Random Fields and images. 3 explains the choice for SLIC as a pre-processing algorithm and explains the algorithm itself. The results and the validation of the results are then shown in 5, followed up by a discussion in 6 and a conclusion plus future work in 7.

2 A brief to Markov Random Field based texture segmentation

The purpose of this chapter is to give an overview of what Markov Random Fields (MRF's) are and show the reader how they are used in image analysis. It will discuss the mathematical theory behind MRF's, why they are useful in this field and what their limitations are.

2.1 Markov Chains

The discussion will start with Markov Chains (Meyn and Tweedie (1993) is an excellent resource on Markoc Chains, other resources include Cooper (1979), Cooper et al. (1980) and Politis (1994)). Markov Chains are easier to understand. This will then be extended to 2-D Markov Chains, which forms the basis for MRF's.

When a sequence of chance experiments forms an independent trials process (for example repeatedly flipping a coin), the possible outcomes for each experiment are the same and occur with the same probability. In the case of flipping a coin the possible outcomes are heads or tails, both with a probability of 0.5. In other words, the results of past trials have no influence on the outcome of future trials. When the outcome of past trials does influence the outcome of future trials, the chain of processes is not independent anymore. Mathematically, when the outcome of a process depends on the past processes, the probability that X_{n+1} takes a particular value x is given as follows:

$$P(X_{n+1} = x) = P(X_{n+1} = x | \{X_0 = x_0, \cdots, X_n = x_n\}) \tag{2.1}$$

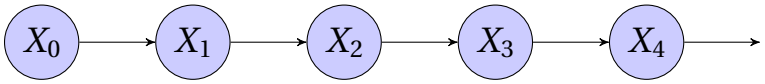
A process is called a Markov Process when its outcome depends only on the outcome of the previous process. A sequence of these processes is called a Markov Chain and can be depicted as in Figure 2.1. Here the X's are random variables. The possible values a particular instance of X can take form a set S called the state space of the chain. In the example of the coin-flip, the state space would be $\{\text{heads}, \text{tails}\}$, for a binary image where a pixel can only take on two distinct values the state space would be $\{0,1\}$ and for an intensity image the state space would be $\{0, \cdots, 255\}$ in the discrete case or $\{0.0, \cdots, 1.0\}$ in the continuous case. Equation 2.1 now becomes:

$$P(X_{n+1} = x) = P(X_{n+1} = x | X_n = x_n) \tag{2.2}$$

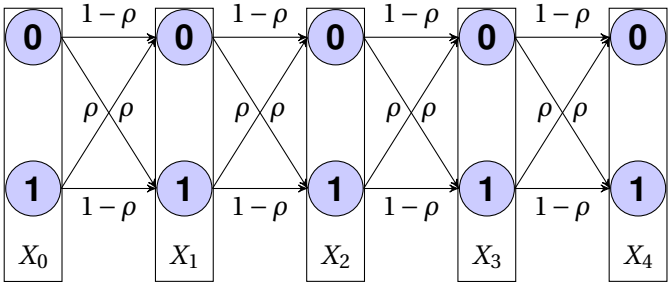
And the probability of a sequence:

$$p(x) = p(x_0) \prod_{n=1}^{N} P(x_n | x_{n-1}) \tag{2.3}$$

A Markov Chain can be viewed as a state machine. The states are the possible values a random variable can take, the state space. The process starts in one of these states and moves to the next state with a certain probability. To clarify this, consider a simple 2-state machine. The state space is $\{0,1\}$ and the probability to transition from state j to state i is given by the transition



Figuur 2.1: A Markov Chain

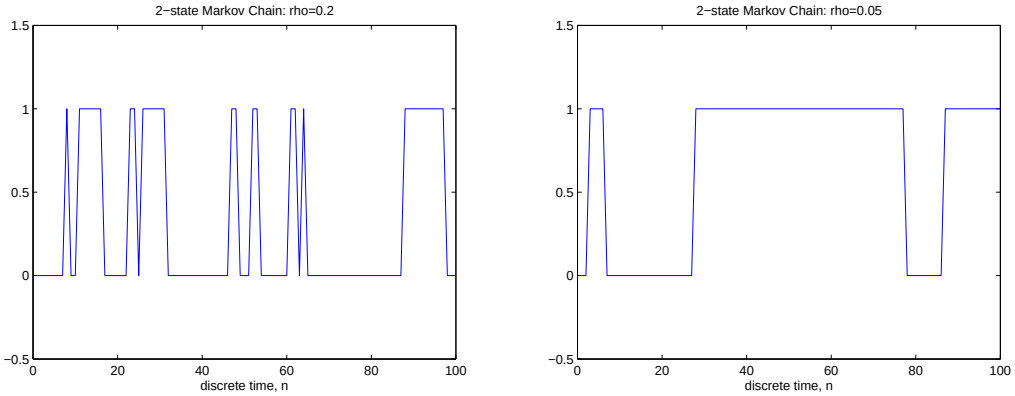


Figuur 2.2: States and transition probabilities for a 2-state Markov Chain where the probability of changing states is ρ

parameters of the Markov Chain denoted as $\theta_{j,i} = p\left(x_n = i | x_{n-1} = j\right)$. Consider a simple 2-state Markov Chain with a transition probability of ρ, then θ is given by the transition matrix below and Figure 2.2 shows dependency on the previous state:

$$\theta = \begin{bmatrix} 1-\rho & \rho \\ \rho & 1-\rho \end{bmatrix} \tag{2.4}$$

Examples of such a Markov Chain for different values of ρ can be seen in Figure 2.3.



$\rho = 0.2$ $\rho = 0.05$

Figuur 2.3: A 2-state Markov Chain with different transition probabilities

Given some observed data, θ is easily estimated using maximum likelihood (ML) estimation:

$$\hat{\theta} = \text{argmax}_\theta \, p\left(x|\theta\right) \tag{2.5}$$

For a Markov Chain the maximum likelihood boils down to counting the observed transitions and dividing by the total number of transitions. Take for example the following Markov Chain:

$$x_n = 0,0,0,1,1,1,0,1,1,1,1,1 \tag{2.6}$$

There are 2 states, 0 and 1, to estimate $\hat{\theta}_{0,1}$ the number of transitions from state 0 to 1 are counted and divided by the total number of transitions originating from state 0. This can be expressed mathematically as follows:

$$\hat{\theta}_{0,1} = \frac{h_{0,1}}{\sum_k h_{0,k}} \tag{2.7}$$

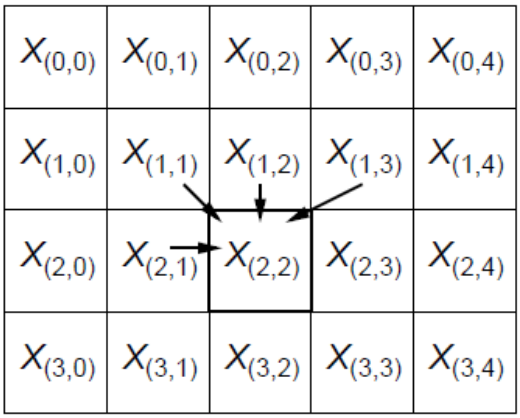
Where $h_{j,i} = \sum_n \delta\left(x_n = i \& x_{n-1} = j\right)$ and k ranges over all possible states reachable from, in this case, 0. Applying this to the Markov Chain in Equation 2.6 results in the following estimate $\hat{\theta}$:

$$h = \begin{bmatrix} h_{0,0} & h_{0,1} \\ h_{1,0} & h_{1,1} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 1 & 6 \end{bmatrix} \tag{2.8}$$

$$\hat{\theta} = \begin{bmatrix} 0.5 & 0.5 \\ \frac{1}{7} & \frac{6}{7} \end{bmatrix} \tag{2.9}$$

2.2 Markov Random Fields

Everything discussed so far can be easily extended to more than 2 states and to 2-D Markov Chains. An example of a 2-D Markov Chain can be seen in Figure 2.4. As indicated by the arrows, the outcome of $X_{(2,2)}$ is dependent on all the processes connected to it that come before it in the ordering. The advantages of Markov Chains are that there are simple expressions for the probabilities as well as for the estimation of the parameters θ. However, 2-D Markov Chains are not readily applied to images because the pixels in an image do not have a natural ordering. Therefore it is impossible to point out which pixels 'come before' another pixel, as was indicated by arrows in Figure 2.4.



Figuur 2.4: A 2-D Markov Chain

A Markov Random Field (see Li (2009) for a comprehensive resource about Markov Random Fields for image processing), like a Markov Chain, is a statistical model for a collection of random variables that are statistically dependent according to some predefined structure. Formally:

$$p\left(x_s | x_r \text{for } r \neq s\right) = p\left(x_s | \boldsymbol{N}\right) \tag{2.10}$$

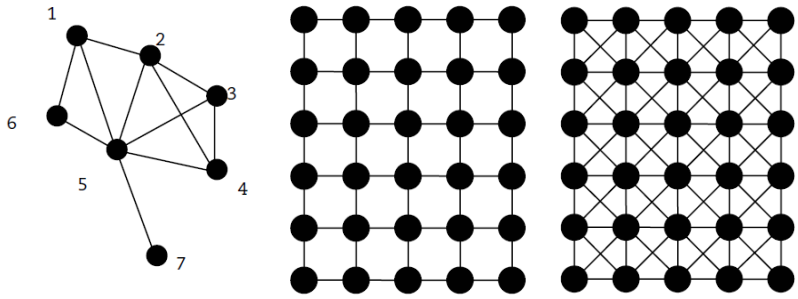
This states that the probability of a site $s \in \boldsymbol{S}$ taking a value x_s, given all the other sites, can be calculated by considering just the neighborhood $\boldsymbol{N}$ of the site. This section will explain the notion of a neighborhood system and the cliques that can be defined on it. It will also show that these concepts apply to regular as well as irregular graphs.

In a Markov Chain the only dependency was the outcome of the previous random variable. In 2-D Markov Chains the dependencies as seen in Figure 2.4 are due to the natural ordering of a 2-D Markov Chain. As was touched upon earlier, image pixels do not have a natural ordering, therefore it is needed to define a structure on the image lattice that enforces this ordering. This

structure is called a graph G and is defined by a set of nodes, $\boldsymbol{V}$, and a set of edges between these nodes, $\boldsymbol{E}$:

$$G = (\boldsymbol{V}, \boldsymbol{E}) \tag{2.11}$$

Examples of graphs can be seen in Figure 2.5. In the case of MRF's, nodes are most commonly called sites, are numbered, and are indicated by $S = \{s_1, s_2, \cdots, s_n\}$. Each site i, i.e. a pixel or superpixel when dealing with images, is associated with a random variable X_i. A realization of this random variable is denoted by x_i.



Figuur 2.5: Examples of graphs. From left to right: an irregular graph, a regular system (orthogonal grid) where each node has 4 neighbors, a regular system where each node has 8 neighbors

2.2.1 Neighborhood system

The sites in $\boldsymbol{S}$ are related to each other via a so-called neighborhood system which is defined as follows where $\boldsymbol{N}_i$ is the set of sites neighboring site i:

$$\boldsymbol{N} = \{\boldsymbol{N}_i | \forall i \in \boldsymbol{S}\} \tag{2.12}$$

A neighborhood system has a few properties:

1. A site is not neighboring to itself: $i \notin \boldsymbol{N}_i$.
2. It is mutual: $i \in \boldsymbol{N}_{i'} \iff i' \in \boldsymbol{N}_i$.

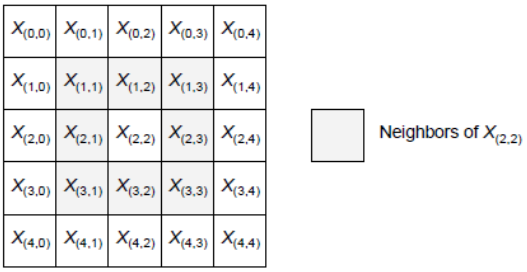
When S is a regular lattice, the neighborhood of site i consists of all the sites within a radius r where the distance between the sites is the Euclidean distance:

$$N_i = \left\{ i' \in S | \, \| i' - i \|^2 \leq r, i' \neq i \right\} \tag{2.13}$$

According to this definition, the 4-connected neighborhood of a site, also said to be a 1$^{\text{st}}$ order neighborhood ($r = 1$), consists of the 4 sites directly left, right, under and above i. Likewise, a 2$^{\text{nd}}$ order neighborhood consists of the eight sites surrounding i (see Figure 2.6). Sites at the edge or a corner of the lattice have fewer neighbors than interior sites. A neighborhood on an irregular lattice, e.g. the lattice resulting after creating super pixels, is defined in the same way as on a regular lattice with the exception that the distance measure has to be defined differently. Examples of distance functions sometimes used are Delaunay triangulation or Voronoi polygons.

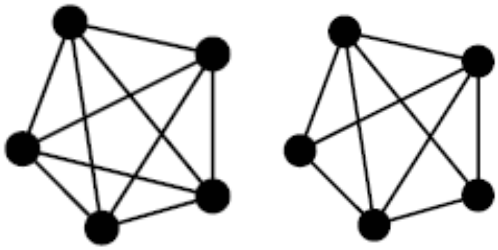
2.2.2 Cliques

Cliques are subsets of sites in the graph $G = (S, E)$ where S is the set of sites and E the set of edges the sites defined by the neighboring relationship. A clique can consist of 1 or more

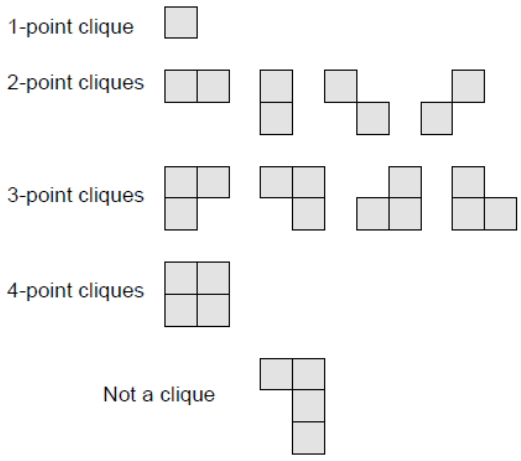


Figuur 2.6: 2nd order, or 8-connected, neighborhood system

sites with the restriction that those sites always have to form a completely connected subset of the graph. An example of a completely connected graph and a graph that is not completely connected is shown in Figure 2.7. A group of single-site cliques is denoted by C_1, a group of pair-site cliques by C_2, etc. Sites in a clique are ordered and the collection of all cliques for G is $C = C_1 \cup C_2 \cup C_3 \ldots$. Assuming a 2nd order neighborhood system, the set of cliques belonging to site $X_{(2,2)}$ in Figure 2.6 is shown in Figure 2.8.

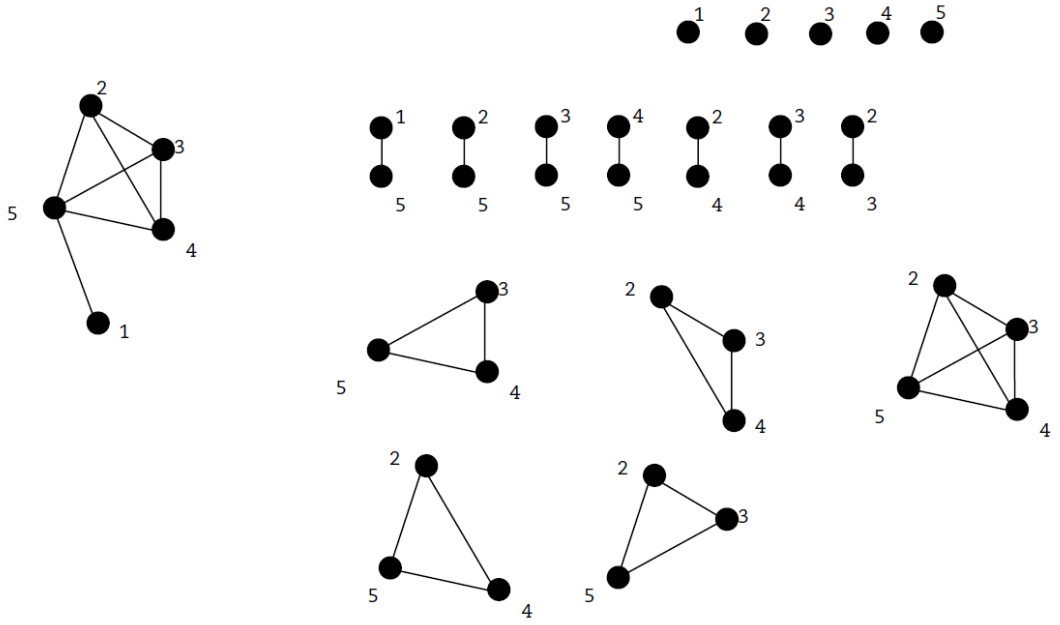


Figuur 2.7: Left: completely connected graph, right: a graph that is not completely connected



Figuur 2.8: Cliques for a 2nd order neighborhood system defined on a regular graph

The notion of a neighborhood system and cliques also applies to irregular graphs, e.g. a super-pixel lattice. Figure 2.9 illustrates this.



Figuur 2.9: An irregular graph with all possible cliques

2.2.3 Gibbs distribution

Now that an ordering has been enforced on the field by means of a graph and the notion of neighborhood systems and cliques has been explained, the next step is to formulate a way to compute the (super)pixel probabilities. This is where the Gibbs distribution comes into play. A density is a Gibbs distribution if the following condition holds:

$$P(x) = \frac{1}{Z} e^{-\frac{1}{T}U(x)} \tag{2.14}$$

T is a constant called the temperature and is in general assumed to be 1. $U(x)$ is the energy function and is defined as follows:

$$U(x) = \sum_{c \in \boldsymbol{C}} V_c(x_c) \tag{2.15}$$

and is a summation of so-called clique potentials $V_c(x_c)$ over all possible cliques $\boldsymbol{C}$ where x_c are instantiations of the random variables in clique $c \in \boldsymbol{C}$. It can also be expressed as a sum of several terms, based on the clique size. For example, considering cliques of up to 3 sites, it can be expressed as:

$$U(x) = \sum_{\{i\} \in C_1} V_1(x_i) + \sum_{\{i,i'\} \in C_2} V_2(x_i, x_{i'}) + \sum_{\{i,i',i''\} \in C_3} V_3(x_i, x_{i'}, x_{i''}) \tag{2.16}$$

A collection of random variables that can be described by a Gibbs distribution is called a Gibbs Random Field (GRF). The easiest GRF's to consider are either isotropic or homogenous. A GRF is said to be homogeneous if $V_c(x)$ is independent of the position of c in $\boldsymbol{S}$ and isotropic if it is independent of the orientation of c in $\boldsymbol{S}$. Basically, $P(\boldsymbol{x})$ is the probability of the occurrence of a certain configuration $\boldsymbol{x}$ of the field. A more probable configuration has lower energy than a less probable one.

2.2.4 Hammersley-Clifford theorem

Specifying the joint probability for an MRF from the conditional probabilities is not possible, but the joint probability of a GRF can be specified by defining clique potential functions $V_c(f)$ and choosing appropriate potentials functions for the system being considered. Also, an MRF is characterized by its local property (the Markovianity) and a GRF by its global property. Therefore, if the equivalence between those two properties can be proved this would provide a way of specifying the joint probability of an MRF in terms of clique potentials defined on a neighborhood system. This is exactly what the Hammersley-Clifford theorem (Hammersley and Clifford (1971)) does. It states that $\boldsymbol{X}$ is an MRF on the set of sites $\boldsymbol{S}$ w.r.t a neighborhood system $\boldsymbol{N}$ defined on this set of sites if and only if $\boldsymbol{X}$ is a GRF on $\boldsymbol{S}$ w.r.t. $\boldsymbol{N}$. For a proof of the theorem see e.g. Besag (1974). The joint probability for an MRF can now be formulated as follows:

$$P(\boldsymbol{x}) = \frac{1}{Z} e^{\frac{-1}{T}U(\boldsymbol{x})} \text{ with } U(\boldsymbol{x}) = \sum_{c \in \boldsymbol{C}} V_c(\boldsymbol{x}) \tag{2.17}$$

Choosing the form and parameters of the potential functions is not a trivial task and requires careful consideration. Next, it is also required to calculate the partition function but because the number of different configurations can get out of hand quickly, this is usually intractable. The issue is further complicated if the energy function $U(\boldsymbol{x})$ contains unknown parameters that require estimating. Therefore the joint probability is often approximated.

2.3 MRF's in image analysis

MRF's were first applied to image analysis by Geman and Geman (1984) and have a wide range of applications in image analysis ranging from texture modeling (Kashyap and Chellappa (1983), Chellappa and Chatterjee (1985) or Cohen et al. (1991)), texture segmentation (Derin et al. (1984), Derin and Cole (1986) or Lakshmanan and Derin (1989)) and image restoration (Geman and Geman (1984), Jeffs and Gunsay (1993), Simchony et al. (1990) or Zhang (1993)) to face recognition (Phillips and Smith (1994)). In order to show how MRF's can be applied in image analysis, this section will discuss an easy example, namely texture synthesis.

2.3.1 Texture synthesis

The easiest models are those which put constraints on just two labels. If only clique potentials of up to two sites are used, the energy function takes the form

$$U(\boldsymbol{x}) = \sum_{i \in \boldsymbol{S}} V_1(x_i) + \sum_{i \in \boldsymbol{S}} \sum_{i' \in \boldsymbol{N}_i} V_2(x_i, x_{i'}) \tag{2.18}$$

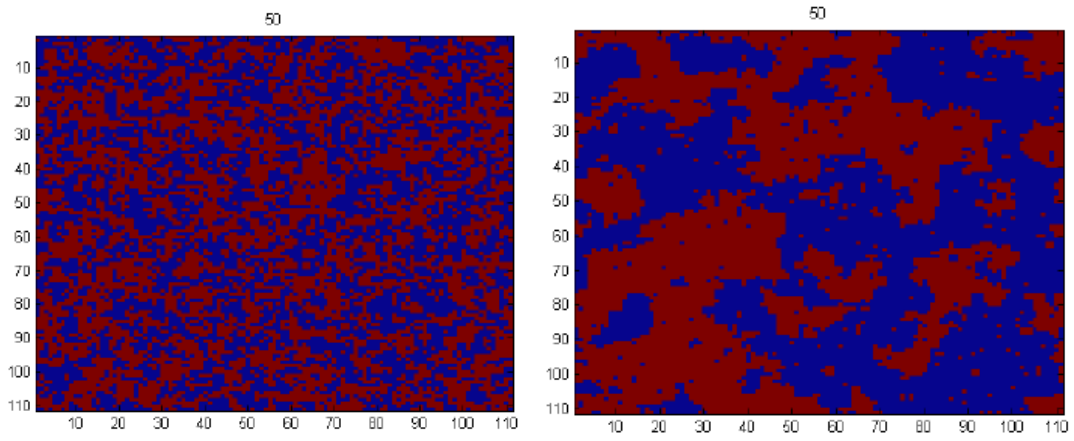
The above energy function is a 2$^{\text{nd}}$ order energy function because it involves up to pair-site cliques. It is most often used because it is the simplest in form but still considers contextual information. Now a specific MRF can be specified by selecting proper functions for V_1 and V_2. One speaks of *auto-models* when $V_1(x_i) = x_i G_i(x_i)$ and $V_2(x_i, x_{i'}) = \beta_{i,i'} x_i x_{i'}$ where $G(\cdot)$ are binary functions and $\beta_{i,i'}$ are constants describing the pair-site interaction between sites in a clique. The energy function then takes the following form:

$$U(\boldsymbol{x}) = \sum_{\{i\} \in C_1} x_i G_i(x_i) + \sum_{\{i,i'\} \in C_2} \beta_{i,i'} x_i x_{i'} \tag{2.19}$$

By constricting the possible labels to just two, e.g. $\{0,1\}$ in the case of binary images, it is called an *auto-logistic* model. The corresponding energy function becomes:

$$U(\boldsymbol{x}) = \sum_{\{i\} \in C_1} \alpha_i x_i + \sum_{\{i,i'\} \in C_2} \beta_{i,i'} x_i x_{i'} \tag{2.20}$$

This model can be further simplified by only considering the 4-connected neighborhood system and label set $\{-1,1\}$. It is then reduced to the *Ising model*. In the case that $\alpha_i = 0$ and $\beta_{i,i'} = 1$ this states that neighboring sites with the same labels are preferred and result in a lower energy. This can be further controlled by the temperature T. When $T \to 0$, the joint probability $P(\boldsymbol{x})$ concentrates on low energy states of $U(\boldsymbol{x})$. In other words this means that T can be used to control the homogeneity of the resulting image when using the *Ising model* to generate a binary texture pattern. A large T implies all configurations being almost equally likely, resulting in an almost uniform $P(\boldsymbol{x})$, whereas a lower temperature results in $P(\boldsymbol{x})$ concentrating on lower energy states and thus higher interaction between sites. This shows in the fact that with lower temperature, larger distinct regions are formed. This is illustrated in Figure 2.10.



Figuur 2.10: Textures generated by using the *Ising model* with $\alpha = 0$ and $\beta = 1$. Left: $T = 1$, right: $T = 0.5$

Many different, and more complicated, MRF models exist (see Li (2009) for more examples and detailed discussions), but what should be taken away from this discussion is that the likelihood of (super)pixel values of a homogeneous texture can be statistically described with an MRF. Furthermore, the labels of (super)pixels and/or regions can also be described by an MRF. See for example Hu and Fahmy (1992) for example where two MRF models are used for segmentation of images, a binomial model for the values of pixels within a segment and a multi-level logistic model for the interaction between different segments.

2.3.2 Sampling

There is one more problem that needs to be tackled. When looking again at Equation 2.17 which states the posterior probability for an MRF, the most likely realization of the MRF (for example a segmentation of an image) is the realization with the lowest energy. However, in order to come to this segmentation, all the possible configurations need to be evaluated to find the one with the lowest energy. For most practical problems this is intractable as even with just two states for a (super)pixel, i.e. there are just two labels, and a small sized image, say 256x256 pixels, this amounts to 2^{65536} possible realizations of the field. A solution to this is to use a Markov Chain Monte Carlo (MCMC) method to sample from the posterior distribution. One can think of Monte Carlo methods as algorithms that help obtain a desired value by performing simulations involving probabilistic choices.

One such sampler is the Gibbs sampler (Geman and Geman (1984)). The Gibbs sampler starts with a random realization of the field and visits every site, i.e. (super)pixel, in a random order. For every possible label it then calculates the probability the current site takes on this label given its surroundings and taking into account the observed data. It then draws a label according to the calculated probabilities and assigns this to the site. After every site has been

visited, the process of visiting every site in a random order is repeated until the field does not change anymore. When the field is stationary, the algorithm has arrived at the realization of the field with the lowest energy and thus the highest probability. Another sampler, also an MCMC methods closely related to the Gibbs sampler, is the Metropolis-Hastings sampler (Metropolis et al. (1953) and Kindermann et al. (1980)). Both of these samplers are a form of energy minimization techniques. There are many more energy minimization techniques, for example the very well-known Iterated Conditional Modes (Besag (1986)) or Simulated Annealing (Geman and Geman (1984)).

The Gibbs sampler is used in the demonstration of how superpixels can be used as input the an MRF-based segmentation algorithm described in chapter 5.

2.4 Requirements for pre-processing methods

This chapter has attempted to explain why MRF's are a good choice for texture segmentation, i.e. they provide a convenient way to combine prior and data likelihood terms in a single graph formulation and make it possible to use global information on a local scale to infer the correct label of a site, even when the texture is varying. While explaining the theory behind MRF's, this chapter has also attempted to identify the difficulties and limitations of MRF's, being the fact that even for moderately large images the computational demand gets very high, when using normal image pixels as input it becomes hard to incorporate texture features and it has to be possible to define a suitable MRF model on the graph structure.

The following chapter will go into pre-processing methods and explains why superpixels as a pre-processing method is a good choice and will help solve these problems. With this, the hypothesis is that the generation of superpixels to be used as input to an MRF-based texture segmentation algorithm will have added value.

3 Methods for pre-processors for MRF-based texture segmentation

This chapter discusses what pre-processing is, why it is applied in the field of image processing and a few different pre-processing algorithms. It also discusses the requirements for the case at hand, motivates the choice for superpixel generation as a pre-processing step and, after comparison of multiple different superpixel algorithms, why the choice for SLIC was made. The next chapter will then explain how SLIC works and discuss some improvements and extensions that were implemented.

3.1 Pre-processing

Pre-processing is the term used for operations on images that results in an improvement of the image data by suppressing undesired distortions or enhancing some image features important for further processing (see Sonka et al. (2007), Chapter 5, for a detailed description of many methods used for pre-processing). It can also mean that the data is represented in a form more suited for the algorithms following the pre-processing. Probably the most well-known example of a pre-processing method is noise removal by filtering the image with a Gaussian kernel (smoothing). Another example of pre-processing is contrast enhancement by using histogram equalization or thresholding to remove pixel with unwanted values. In the case of face recognition, pre-processing can consist of face registration, the translation and alignment (usually by using the eyes) of the faces to create a standard form used for further processing. As a last example of pre-processing, consider image restoration. In image restoration one tries to recover the true values of noisy pixels. Obviously there are many more methods and what defines pre-processing depends on the requirements of the project.

In the previous chapter it was shown that Markov Random Fields, although a very powerful segmentation technique, are also very computationally demanding and this demand increases rapidly with an increase in the number of pixels in the image. This means that in this case the pre-processing step should consist of a way to decrease the effective number of pixels in the image, but without losing the information present in it. Also, an MRF operates on individual pixels which makes it very hard to incorporate accurate texture features since this requires neighborhood operations. In a neighborhood operation a window is placed with its center over the pixel and the texture within this window is analyzed. Here the size of the window is of particular importance. If it is chosen too small, the texture features will not be accurate. On the other hand, the larger the window, the bigger the chance the window overlaps two or more distinct image regions resulting in bad texture features. For these reasons it was decided to generate superpixels. A superpixel is a collection of pixels that were determined to belong together, in other words that are part of the same object in the image, and can therefore be seen as 1 pixel.

The following section will take a closer look at a few superpixel algorithms, namely Watershed, Mean-shift, Turbopixels and Normalized graph-cuts. These were chosen because it showcases the different approaches one can take to generate superpixels. They will be analyzed in terms of control (e.g. the size or number of superpixels), boundary recall (how well the superpixel boundaries adhere to the object boundaries in the image) and computational and memory efficiency. More examples and variations on the shown algorithms can be found in the literature. After comparison of the superpixel algorithms, the choice for SLIC is motivated. A detailed explanation of SLIC and the extension and improvements added to the basic algorithm is deferred to Chapter 4. In this chapter it will also be shown that SLIC requires bookkeeping in the form of an adjacency matrix. For every superpixel, this adjacency matrix keeps track of which superpixels are its direct neighbors. As was explained in the previous chapter, MRF-based seg-

mentation requires the selection of cliques from a neighborhood defined on the graph. In the case of a graph consisting of superpixels, the neighborhood consists of all the direct neighbors of a superpixel and therefore SLIC makes it very easy for MRF-based segmentation to determine the neighborhood of the superpixel by providing the adjacency matrix.

3.2 Existing superpixel algorithms

This section will discuss a few chosen superpixel algorithms and compare them in terms of the properties mentioned in the previous section.

3.2.1 Mean shift

The mean shift superpixel algorithm, Comaniciu and Meer (2002), is a mode-seeking procedure that consists of two main steps: discontinuity preserving filtering and mean shift clustering. The first step basically assigns pixels that point towards the same mode the same value and mean shift clustering groups together all the modes that are similar enough. This will be explained here in a bit more detail.

Discontinuity preserving filtering

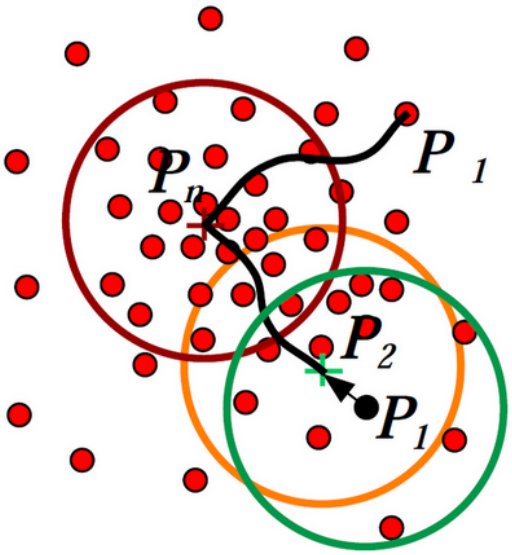
The point of this step is to determine for every pixel in the image the mode it belongs to. To achieve this, the pixels first have to be mapped to feature space. For example, the R, G and B color channels of a pixel can be features and in this case the feature space is thus 3-dimensional. However, more image properties besides color can be used and hence the feature space can be more dimensional. The feature space is more densely populated in locations corresponding to significant image features and the mode (or mean) of these more densely populated locations is what the algorithm is trying to identify. Discontinuity preserving means that the algorithm will group together pixels that belong to the same mode and in doing so will preserve boundaries present in the image. In other words, if there is a clear boundary between, for example, sky and grass in an image, the grass pixels will not be counted as sky and vice versa.

To give an understanding of how discontinuity preserving filtering is done, consider the simple case of 2D feature space as an example, See Figure 3.1. To determine the mode a pixel 'points to', a pixel P_1 is taken and a circle (kernel) is placed over it. This is depicted in Figure 3.1 by the green circle. Now this kernel needs to be shifted so that as many points as possible in feature space are contained within it. To do this, the algorithm iteratively calculates the point of highest density within the kernel and shifts the center of the kernel to this location until convergence, i.e. the kernel does not shift anymore. Referring again to Figure 3.1, the kernel is shifted to the orange location until it converges at the location depicted by the red circle. Now every pixel in the filtered image is given the value of the mode they point to. For this part of the algorithm the only real parameter, besides the type of kernel used, is the size of the kernel (bandwidth). The smaller the kernel, the more modes will most likely be detected so even small changes in, for example color, will be detected.

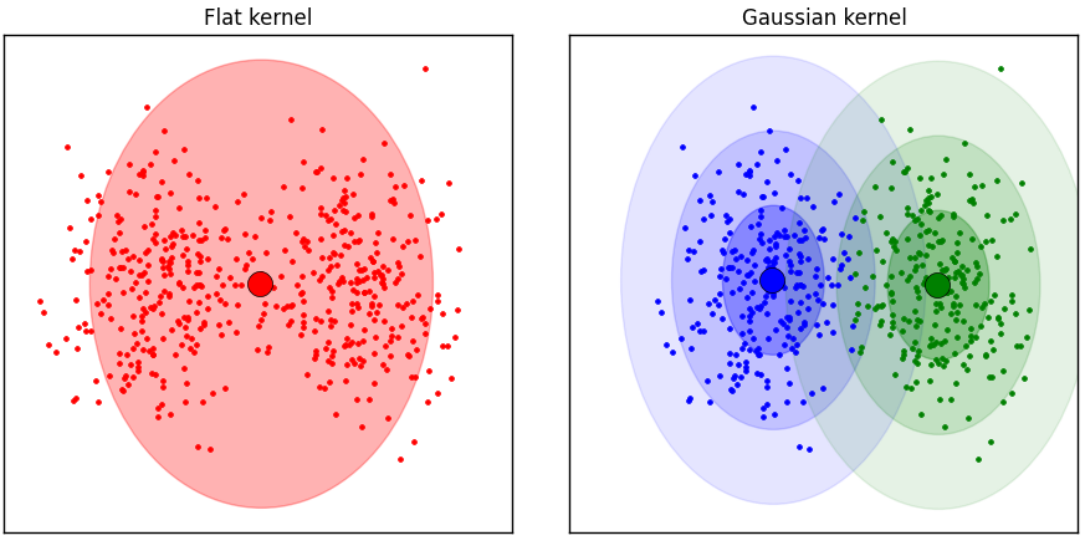
As for the kernel, it is most common to take a Gaussian kernel as opposed to for example just a flat kernel. A Gaussian kernel attaches weights to the points within it where points closer to the center are weighted more heavily than points located closer to the edge of the kernel. Therefore a Gaussian kernel wants as many points as possible near its center. A flat kernel on the other hand just wants to contain as many points as possible. Figure 3.2 illustrates that the Gaussian kernel has the ability to distinguish modes where a flat kernel cannot.

Mean shift clustering

After discontinuity preserving filtering, the result is an image where every pixel has been assigned the value of the mode it points to after convergence. Basically what is left are superpixels, but neighboring superpixels can have a mode that is only slightly different. The point if the



Figuur 3.1: Principle of mean shift (image taken from DeMenthon and Megret (2002))



Figuur 3.2: Flat kernel versus Gaussian kernel

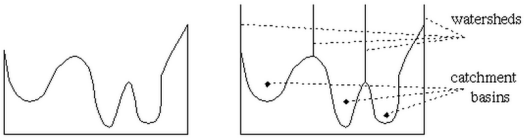
clustering step is to merge neighboring superpixels with a difference in mode that is less than a certain threshold.

Discussion

The only parameters for mean shift that can be controlled are the size and type of the kernel en the threshold that determines whether neighboring superpixels should be merged. This means that there is no direct control over the amount of superpixels, and hence their size, or the compactness (how nicely they are shaped) of the superpixels resulting in superpixels that are highly irregular in shape and of non-uniform size. With a complexity of $\mathcal{O}\left(N^2\right)$, where N is the number of pixels in the image, it is also quite a slow algorithm.

3.2.2 Watershed

Watersheds are a quite straight-forward concept. An image may be interpreted as a topographic surface where, for example, the gray-levels of the pixels represent altitudes. Thus, region edges correspond to high watersheds and low-altitude region interiors correspond to catchment basins. Figure 3.3 shows a one-dimensional example to clarify this idea. There are two basic approaches to watersheds which are essentially dual to each other and they will be discussed briefly below.



Figuur 3.3: One-dimensional example of watersheds (image taken from REFERENCE)

Top-down

The first approach determines for every pixel in the image a downward path to a local minimum of surface altitude. A catchment basin is then defined as the set of pixels whose downward path all end at the same local minimum. For continuous surfaces the downward paths are easy to determine. For digital surfaces however, there exist no rules to determine the downward paths uniquely. This resulted in inaccurate algorithms with extremely high computational costs.

Bottom-up

The second approach by Vincent and Soille (1991) takes the opposite approach of the first one. Instead of determining downward paths, this approach starts by identifying the local minima and flooding the basins. This can be visualized as follows: imagine there is a hole in each basin and that the surface is submerged into water. The basins that are below the water level will fill with water through their holes as the surface is submerged further and further. The moment two basins are about to merge together, a dam (or watershed) is constructed as high as the highest peak of the surface. The algorithm is based on sorting the pixels in increasing order of their altitudes and a flooding step consisting of a breadth-first scanning of all pixels in the order of their altitudes. Details can be found in Vincent and Soille (1991).

Discussion

With a complexity of $\mathcal{O}\left(N \log N\right)$ the watershed algorithm is relatively fast. However, the resulting superpixels are highly irregular in shape and size and do not adhere to boundaries in the image very well. On top of that, the algorithm does not offer any control over the amount of superpixels or the compactness of the superpixels.

3.2.3 Turbopixels

Turbopixels, Levinshtein et al. (2009), is an algorithm for generating superpixels that is based on geometric flows. It starts with a predefined number of initial seeds, evenly distributed over the image plane, which are dilated so as to adapt to the local image structure. At a basic level, the initial seeds grow outward by means of a curve evolution method with a skeletonization process on the background region to prevent the expanding seeds from merging.

The geometric flows associated with the turbopixel algorithm is implemented using level-set methods. The idea is to devise a flow by which curves evolve to obtain superpixel boundaries. If C denotes the vector of curve coordinates, then each point on the curve moves with speed S in the direction of its outward normal. Let Ψ be the level-set over the image defined as the

signed Euclidean distance of each image pixel to the closest point on the boundary between pixels already assigned to a superpixels and unassigned pixels. The zero level set of Ψ is the only set of interest and is intuitively comprised of the pixels that represent the boundaries of the superpixels. To be able to grow and still maintain an accurate representation of Ψ, pixels in a narrow band around the boundary of the superpixel are counted towards the zero level set.

The speed term S is comprised of two terms: $S = S_I S_B$. S_B is proximity-based velocity term that is used to ensure that growing superpixel boundaries never cross other superpixel boundaries. This term is therefore 0 on the skeleton of the unassigned region and 1 everywhere else. Step three in Figure 3.4 shows the skeleton between the growing superpixels and should clarify this idea.

S_I is an image-based velocity term which is a bit more complicated:

$$S_I(x,y) = [1 - \alpha\kappa(x,y)]\phi(x,y) - \beta[N(x,y)\cdot\nabla\phi(x,y)] \tag{3.1}$$

Here x and y are image coordinates and ϕ is a local affinity function:

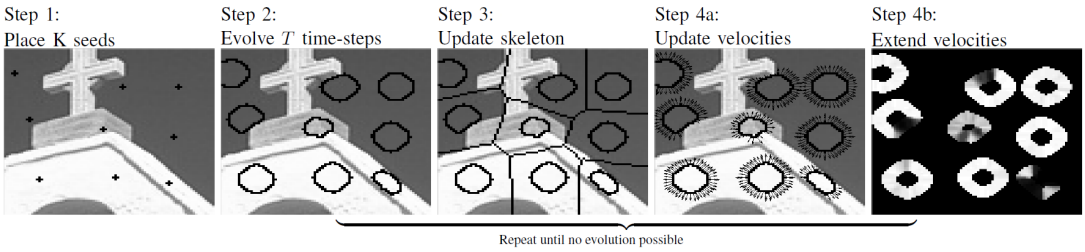
$$\phi(x,y) = exp^{\frac{-E(x,y)}{\nu}}, \quad E(x,y) = \frac{\|\nabla I\|}{G_\sigma * \|\nabla I\| + \gamma} \tag{3.2}$$

What needs to be taken away from this is that S_I slows down the growth of the superpixels when it comes close to boundaries present in the image. Since boundaries in an image are characterized by large gradients, ϕ is function that returns low values near edges and high values in homogeneous regions.

The steps involving the algorithm are depicted in Figure 3.4. First K (the desired number of superpixels) seeds are evenly distributed over the image plane and shifted to a position of low gradient magnitude to ensure they are not accidentally placed on an edge in the image. Step 2 evolves the curves over a number of time-steps by computing the velocity S in a narrow band around the superpixel boundaries, i.e. the zero level set Ψ^0, and evolves the boundaries according to the following equation:

$$\Psi^{n+1} = \Psi^n - S_I S_B \|\nabla\Psi^n\|\Delta t \tag{3.3}$$

Where n denotes the n-th iteration and Δt is a time step. This might look complicated but basically this computes the next iteration by evolving Ψ^n with the narrow band, extending Ψ with those pixels that have a positive value. When there are little or no pixels being assigned to superpixels anymore, the skeleton is updated in step 3. Next the velocities are calculated and extended again and the process repeats from step 2.



Figuur 3.4: Steps of the superpixel algorithm (image taken from the original paper Levinshtein et al. (2009))

After the algorithm has ended, generally not all pixels have been assigned to a superpixel. Therefore the remaining pixels are assigned to their closest superpixel.

Discussion

Even though Levinshtein et al. (2009) claim their algorithm has a complexity of $\mathcal{O}(N)$, Achanta et al. (2012) have found in their comparison that it is amongst the slowest superpixel algorithms. It does however give the user control over the amount of superpixels created by selecting the number of initial seeds. The created superpixels also have uniform size and compactness and adhere well to image boundaries.

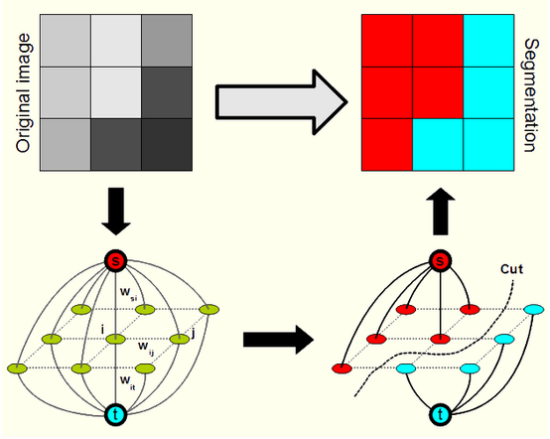
3.2.4 Normalized graph-cuts

Normalized graph-cuts (Shi and Malik (2000)) is a graph based approach where an image is viewed as a weighted graph $G = (V, E)$ with vertices V represented by the pixels and edges E. Edges between two pixels i and j have a weight $w(i, j)$ which is a measure of the similarity between the pixels. The goal is to partition the set of vertices, i.e. pixels, in disjoint sets $V_1, V_2 \cdots V_n$ where the similarity of pixels within the set V_i is high and across sets V_i and V_j it is low.

The graph can be partitioned into two disjoint sets V_1 and V_2 by removing edges where $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$. A measure for the dissimilarity is the total sum of the removed weights. This is called the cut:

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w(i, j) \tag{3.4}$$

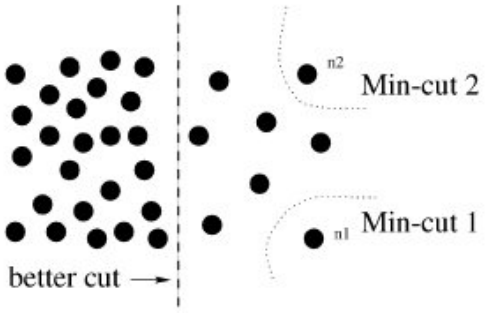
The optimal bi-partitioning of the graph is the one where the cut is minimized. See Figure 3.5 for a visualization.



Figuur 3.5: Minimum cut in an image.

The problem with this definition of a cut is that is favors small sets of isolated points in the feature space, i.e. nodes in the graph or pixels in the image. This happens because the value of the cut increases with the number of edges removed. In terms of an image, this means that pixels with a much higher or lower value that its surroundings, e.g. due to speckle noise, that should be seen as part of a segment are singled out. See Figure 3.6.

To overcome this problem, Shi and Malik (2000) propose a new criterion to partition the graph. Instead of just looking at the total value of the removed edges, they compute the cut value as a fraction of the total connections to all the nodes in the graph and call this criterion the normalized cut. Because this is just a bi-partition of the graph, the algorithm is called recursively to partition each of the two found segments as long as the normalized cut value is below a certain threshold. A too high a value for the normalized cut means one is trying to force a partition that does not provide a meaningful segmentation anymore.



Figuur 3.6: Minimum cut giving a bad partition of the graph.

Discussion

The normalized cuts algorithm produces very regularly shaped superpixels but the superpixels don't adhere to image boundaries very well and the algorithm gives no direct control over the amount of superpixels created other than variation of the threshold discussed above. With a complexity of $\mathcal{O}\left(N^{\frac{3}{2}}\right)$ it is among the slowest algorithms.

3.2.5 Comparison

After having discussed a few superpixel algorithms, Table 3.1 shows their performance based on the properties used in the discussion above. The columns for adherence to boundaries and computational and memory efficiency are rated with – (very bad), - (bad), + (good) or ++ (very good). A more comprehensive comparison can be found in Achanta et al. (2012). Because SLIC has very good adherence to boundaries, is faster and more memory efficient than other superpixel algorithms and gives the user control over both the amount of superpixels generated and their compactness, it was chosen to be used as a pre-processing method for MRF-based image segmentation. The following chapter will discuss the SLIC algorithm in detail, including the extensions and improvements that were implemented.

	Adherence to boundaries	Computational and memory efficiency	Control over amount of superpixels	Control over compactness
Mean-shift	+	-	no	no
Watershed	-	++	no	no
Turbopixels	+	-	yes	no
N-cuts	-	- -	yes	no
SLIC	++	++	yes	yes

Tabel 3.1: Performance of different superpixel algorithms.

4 Simple Linear Iterative Clustering

Chapter 2 explained the theory behind MRF's and showed their strengths and weaknesses when used for texture image segmentation. It showed the need for a pre-processing algorithm and the properties it should have, i.e. reduce the number of effective pixels in the image while maintaining the information present in the image, significantly speed up the MRF-based algorithm following it and enable the MRF-based algorithm to accurately take into account texture features. It was concluded that a superpixel algorithm possesses these properties. Then the previous chapter compared superpixel algorithms based on how well the resulting superpixel adhere to image boundaries, the computational and memory efficiency of the algorithm and whether it enables the user to control the number of superpixels and the compactness of the superpixels. At the end of the chapter the Simple Linear Iterative Clustering (SLIC, Achanta et al. (2010)) algorithm was chosen as a superpixel pre-processing method for MRF-based texture image segmentation. Also, SLIC requires bookkeeping in the form of an adjacency matrix that can be used later to easily retrieve the neighborhood of a superpixel. This chapter contains a detailed explanation of SLIC.

SLIC is based on the well-known k-means clustering algorithm (the term k-means was first used by Macqueen (1967) while the idea goes back to Steinhaus (1956)), which aims to partition a of n d-dimensional observations $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ into k clusters $\boldsymbol{S} = \{S_1, S_2, \ldots, S_k\}$. A cluster is represented by its mean and for every observation the distance to all the clusters is calculated and it is assigned to the cluster that it is closest to. Mathematically, this can be expressed as follows:

$$\underset{\boldsymbol{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \boldsymbol{C}_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$

Where $\boldsymbol{\mu_i}$ is the mean of the points forming cluster C_i.

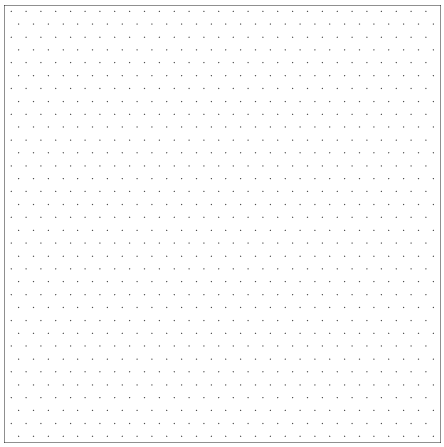
The big difference with k-means clustering is that SLIC restricts its search to a small area around the cluster center whereas k-means clustering compares every observation with all the cluster centers. The area the algorithm searches is determined by the number of roughly desired superpixels K. If an image contains N pixels then the size of a superpixel is roughly $\frac{N}{K}$ pixels. This means that if the superpixels are of roughly equal size, there would be a superpixel center every $S = \sqrt{\frac{N}{K}}$ pixels and the size of a superpixel is approximately S^2. For this reason it can be assumed that pixels associated with this superpixel are within a $2Sx2S$ area around the superpixel center. Therefore SLIC searches in a $2Sx2S$ area around the cluster center. This results in a significant speedup over k-means clustering.

4.1 The algorithm

This section will attempt to give a brief overview of the algorithm in order to establish a general understanding of how it works. The details will be addressed in later sections.

Figure 4.2 shows a flowchart of the algorithm. As can be seen, the first step is to initialize the grid, i.e. distribute the initial cluster centers evenly over the image plane. This can be visualized as in Figure 4.1. The authors of Achanta et al. (2010) propose to calculate the gradients in a $3x3$ neighborhood around each cluster center and move the cluster center to the point with the smallest gradient inside this neighborhood. The reason to do this is to make sure that the cluster center is not initially placed on a region boundary or noisy pixel, thereby giving a biased representation of the cluster. The influence of this bias can be quite large in mainly homogenous images, but since the radar images as well as the H&E stained pathology images are not

very homogenous, i.e. there are large variations within a cluster, and this step was omitted. Also, the effect of possible bias is negated by the fact that the final clusters are formed over a number of iterations where after each iteration the cluster center is recalculated. Experiments confirmed that this step can be safely omitted in this case.



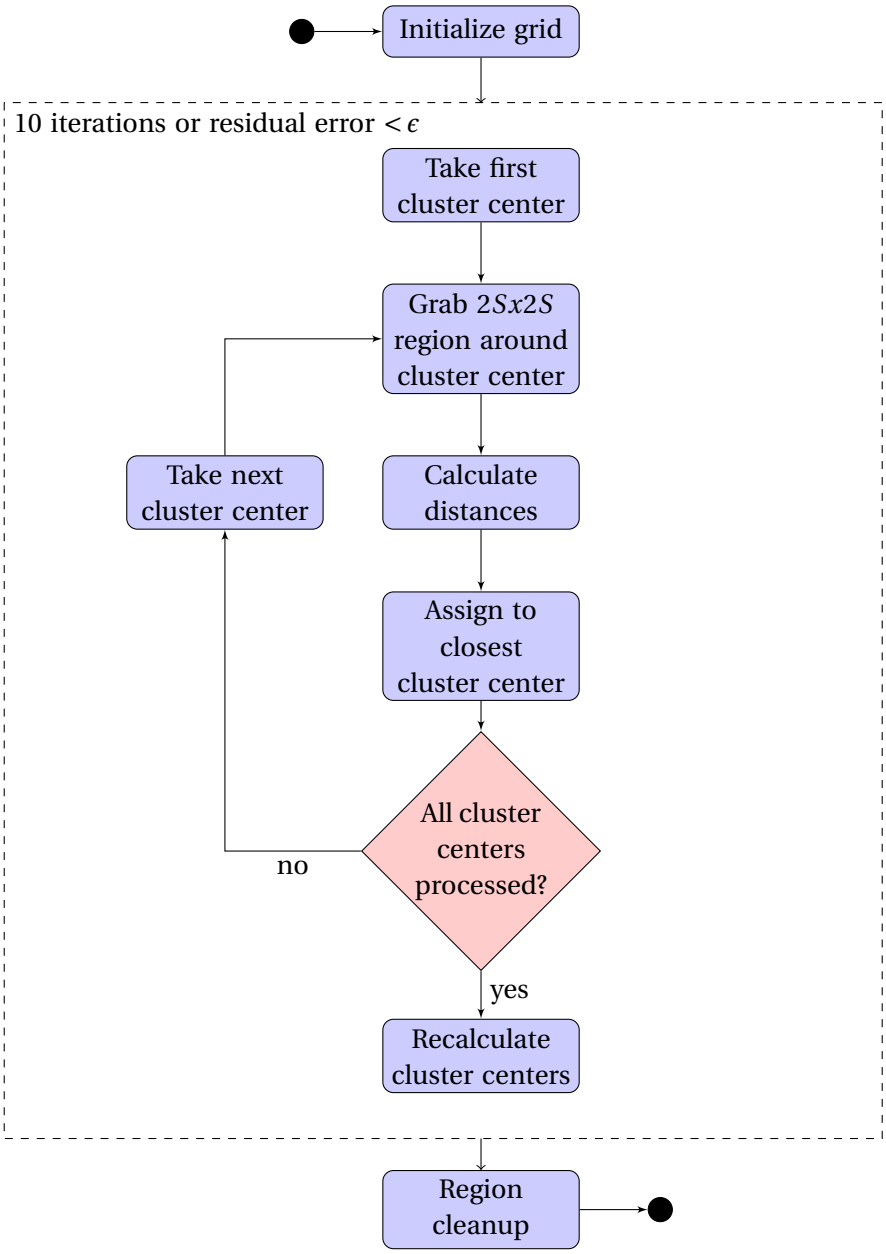
Figuur 4.1: Initial distribution of the cluster centers over the image plane

Next, the first cluster center is processed. A $2Sx2S$ image region around it is taken and for every pixel in this region the distance to the cluster center is calculated according to the distance measure described in the next section. Some bookkeeping is done to know whether the calculate distance for a pixel to the cluster center currently being processed is smaller than the distance to the cluster center it is currently assigned to. The pixels for which the calculated distance is smaller than the distance to the cluster center they are currently assigned to get assigned to the cluster center currently being processed.

When all the cluster centers have been processed, the cluster centers are recalculated with the pixels that are currently assigned to it. This process is repeated a number of times. In practice this process converges in most cases in just a couple of iterations. Here 10 iterations were chosen to be on the safe side, but, depending on the type of images, as few as 2 or 3 can be enough. Another way to determine when to terminate the algorithm is by calculating the residual error after processing all the cluster centers. The residual error is defined as the L1 distance between previous cluster centers and the recomputed cluster centers. It algorithm is terminated when the residual error is smaller than a set threshold ϵ. However, the bookkeeping and extra calculations involved make the algorithm slower and more complex.

The last step is technically not part of the SLIC superpixel algorithm but is a necessary post-processing step. Because SLIC does not enforce connectivity of the superpixels, after the previously described process stray labels may remain. In other words, by continuously reassigning pixels to cluster centers, some clusters may become very small or splintered. The region cleanup is the last step in Figure 4.2 and assigns clusters that are too small to the largest neighboring cluster. What defines a segment as being *too small* is application dependent but in this case it was chosen to merge segments that are less than 25% of the desired superpixel size.

This concludes the description of the basic SLIC algorithm. In the following sections improvements and extensions to the basic algorithm will be discussed.



Figuur 4.2: Flowchart depicting the SLIC algorithm

4.1.1 Calculation of the distance

In order to determine to which cluster center a pixel belongs, a measure for the distance between a pixel and a cluster center is needed. In the case of SLIC, this measure is based on their spatial proximity in the image and color distance. This means that a cluster center represented by a 5-dimensional vector consisting of its x- and y-location and its color vector lab in CIELAB color space:

$$\boldsymbol{C}_k = [l_k, a_k, b_k, x_k, y_k]^T$$

The reason the CIELAB color space was chosen, as opposed to for example the well-known RGB color space, is that in other color spaces the colorimetric distances between individual colors do not correspond to perceived color differences. For example, the distance between green and greenish-yellow is perceived as being quite small but the colorimetric distance is relatively large, whereas the colorimetric distance between red and blue is quite small while it perceived as being very large. CIELAB overcomes this problem and therefore the Euclidean distance, which gives the most accurate distance, can be used for both the spatial and the color distance and this results in the following equations:

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \tag{4.1}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{4.2}$$

However, Euclidean distances in CIELAB color space are only meaningful for small distances. If the spatial distance between pixels exceeds this distance, they begin to outweigh the color similarities. The result of this is that the superpixels don't adhere well to region boundaries anymore and become very irregular and splinter easily. This motivates the introduction of a variable to control the compactness of the superpixels. To accomplish this, the spatial distance is normalized to the grid size S and a variable m is introduced in the total distance measure:

$$D_{total} = d_{lab} + \frac{m}{S} d_{xy} \tag{4.3}$$

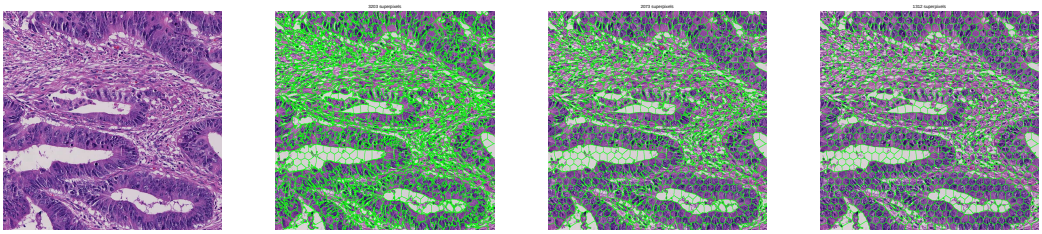
In the above equation, m controls the compactness of the superpixels. A higher value for m results in more emphasis on the spatial distance term and hence more compact, i.e. more regularly shaped, superpixels. However, a too high value will result in bad adherence to region boundaries. Therefore, an application specific balance between the 2 terms has to be found. Achanta et al. (2010) states that meaningful values for m can be in the range $[1,20]$ and propose to choose $m = 10$ as a default value that provides a good balance between the 2 terms. However, as can be seen in Figure 4.3, this is also application dependent. In this case a value of $m = 10$ does not give a good balance between the color and proximity terms.

4.2 Improvements and extensions

Although the algorithm described above performs very well as is, there are a couple improvements that can be made and extensions that can be added to further facilitate the subsequent processing. This is the topic of this section and the following subsections will discuss them in turn.

4.2.1 Dynamic weighting

in the section above the distance measure was explained. The spatial distance is normalized to the grid size S, or the average expected superpixel size. Then a value for m is chosen that



Figuur 4.3: The original image and the original image overlaid with the generated superpixels without dynamic weighting for $m = 10, 30, 50$ respectively

finds a good balance between the color term and the spatial term in equation 4.3. This value for m however, is used over the entire image and not adjusted on a per superpixel basis. This means that the balancing between the two terms might not be as good as one would like it to be in every part of the image. Better would be, if in the balancing of the terms, the local color distribution was taken into account, i.e. dynamic weighting of the color distance term. This can be done by normalizing the color distance term by the maximum color distance of a pixel within the cluster to the cluster center. 4.3 then changes to the following:

$$D_s = \frac{d_{lab}}{c} + \frac{d_{xy}}{S^2} \tag{4.4}$$

Where c is the maximum color distance within the cluster.

Implementation of dynamic weighting results in superpixels that are more compact and adhere better to region boundaries within the image. Also, because of the application of dynamic weighting, less splintering of the superpixels occurs which results in a much faster region cleanup. A downside to dynamic weighting might be that the parameter m is no longer used and therefore there is no control over the compactness of the superpixels anymore. In general, however, the results obtained with dynamic weighting are better than without.

4.2.2 Sigma filtering

Sigma filtering (Jong-Sen and Lee (1983)) can be combined with the SLIC superpixel algorithm to counteract the misclassification of pixels. Misclassification of pixels can occur e.g. when noisy pixels are present in the image, when small protrusions are classified to belong to the wrong superpixel due to spatial proximity having more weight than color or, in the case of radar images, when a point target is present in the image when one is just interested in characterizing the environment.

What all of these examples have in common is that the misclassified pixels greatly differ from the mean pixel in the superpixel. Having one or more of those misclassified pixel in the superpixel will introduce an error when the cluster center is recalculated using the pixels that are determined to belong to said superpixel in the current iteration of the algorithm. Since SLIC is an iterative algorithm, the error will be propagated and probably grow larger after more iterations. This motivates the need for a refinement of the SLIC algorithm that makes sure that pixels that are not a good representation of the superpixel, because they are misclassified or simply outliers or noisy pixels, are not included in the calculation of the cluster center. This to make sure that the cluster center is an as good as possible representation of the superpixel. This is where the sigma filter comes into play.

Kwang-Shik et al. (2013) shows a way to filter out the pixels that are most likely not a good representation of the superpixel they belong to by looking at the L* values and not taking into account pixels with an L* value more than α standard deviations away from the mean. If the

center of a cluster $\boldsymbol{C}_k$ is denoted by $\boldsymbol{\psi}_k$ and $\boldsymbol{c}_k$ is the 5-dimensional vector of a pixel in that cluster, then mathematically this can be expressed as follows:

$$\boldsymbol{\psi}_k = \frac{1}{N} \sum_{l \in \boldsymbol{C}_k} \boldsymbol{c}_l \tag{4.5}$$

In which N is the number of pixels in the superpixel. If however, one wants to include only pixels inside the cluster with an L* value not more than α standard deviations from the mean L* of the cluster, than equation 4.5 changes into the following:

$$\boldsymbol{\psi}_k = \frac{1}{N} \sum_{l \in \boldsymbol{\Omega}_k} \boldsymbol{c}_l \tag{4.6}$$

Where Ω_k is the set of all pixels with an L* value not more than α standard deviations from the mean L* of the cluster:

$$\boldsymbol{\Omega}_k = (\|L_k - L_l\| < \alpha \cdot \sigma_k) \cap \boldsymbol{C}_k \tag{4.7}$$

4.2.3 Segment merging

As mentioned earlier in this chapter, the SLIC algorithm can generate a few stray labels or very small segments because of the iterative nature of the algorithm or the splintering of superpixels. In the cleanup step, see Figure 4.2, these stray labels and small segments are merged with other superpixels. In the original SLIC algorithm the authors use a connected components algorithm. If $\boldsymbol{\Theta}$ is the set of neighboring superpixels of the small segment $\boldsymbol{C}_k$, then it simply merges the small segment with the first superpixel in $\boldsymbol{\Theta}$ that is a neighbor of the first pixel in $\boldsymbol{C}_k$.

While this is a fast method, it is certainly not the most accurate method. It does not take any information about the small segment or its neighboring superpixels into account and therefore canâĂŹt make an informed decision whether merging with the particular neighboring superpixel actually improves the final result. Kwang-Shik et al. (2013) describes a different way to determine to which neighboring superpixel, if at all, the small segment should be merged. This is discussed here.

The authors of Kwang-Shik et al. (2013), just as was the case with the sigma filter, assume the luminance is the most defining characteristic of a superpixel and use a simple method based on just the distance in luminance between the small segment and its neighboring superpixels to determine with which superpixel the small segment has to be merged.

Let μ denote the mean luminance of the small segment that is to be merged, then the neighboring superpixel to merge with is determined by the following criterion:

$$\underset{\boldsymbol{C}_k \in \boldsymbol{\Theta}}{\operatorname{argmin}} (\mu - \mu_k)^2 \tag{4.8}$$

In other words, the small segment is simply to be merged with the neighboring superpixel that it is closest to it in terms of the distance in luminance.

This is already an improvement over the way the original SLIC algorithm handled stray labels and small segments. It now merges the small segment with the most similar neighboring superpixel in terms of the luminance distance instead of with the largest segment which might not be similar at all. However, the fact that a particular superpixel is closest to the small segment does not mean that it is also similar to the segment. It can occur that in terms of the luminance distance all neighboring superpixels are very different. In this case the small segment is different enough from all its neighboring superpixels that it should be its own superpixel. Therefore, Kwang-Shik et al. (2013) set a threshold for the luminance distance to determine if they

should be merged at all. Let d_L be the distance in luminance between the small segment and the closest neighboring superpixel. Then they should only be merged if $d_L < T$, where T is a predetermined threshold. In their paper they propose to use a default value of $T = 300$, but this parameter can be set lower or higher depending on the application's needs.

4.2.4 DBSCAN

Going from pixels to superpixels is already an incredible reduction in terms of effective pixels, but inspecting the result of the SLIC algorithm one can immediately see that certain superpixels are very similar. To reduce the effective number of pixels even more, a simple clustering algorithm, DBSCAN (Ester et al. (1996)), is used to group very similar superpixels together. Since information about neighboring superpixels has already been needed in earlier steps in the algorithm, bookkeeping is already being done in the form of an adjacency graph. This makes DBSCAN an efficient choice. It also has the benefit that which superpixels are being merged can be controlled by a parameter, a threshold. This is needed to make sure that superpixels that shouldn't be grouped definitely will not be grouped. This can be accomplished by setting the threshold high enough.

DBSCAN is effectively a region growing technique that can best be explained by referring to the pseudo code below:

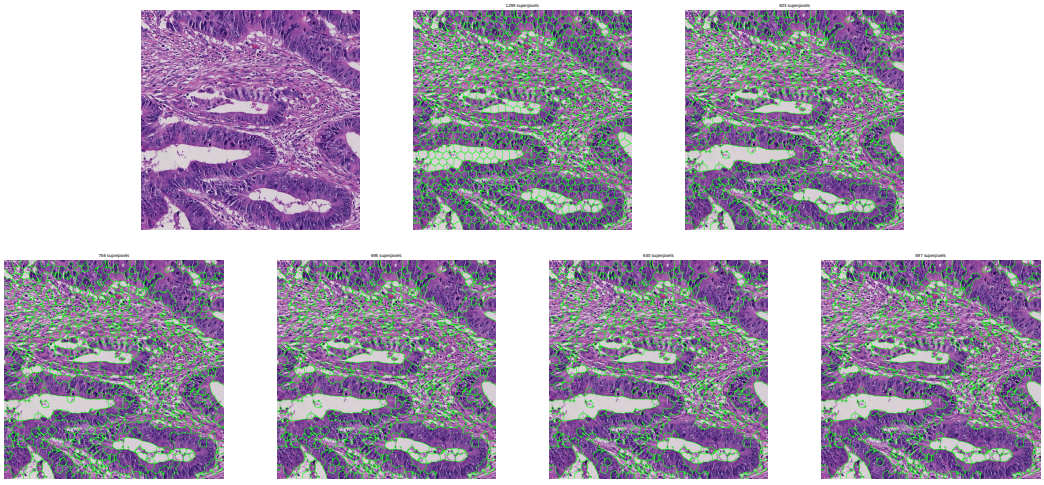
```
1. Repeat until every superpixel has been visited
        1.1. Take superpixel, S, and start a new cluster
        1.2. Mark S as visited
        1.3. Find neighboring superpixels closer than
                threshold
        1.4. While the set of neighboring superpixels is not
                empty
                1.4.1. Take the first in the list of
                        neighboring superpixels, T
                1.4.2. Mark T as visited
                1.4.3. Expand the current cluster with T
                1.4.4. Find neighboring superpixels of T and
                        add to the list
2. Create new labels
```

Basically the algorithm starts by taking a superpixel, marking it as visited and starting a cluster with it. Next it finds all the neighboring superpixels with a distance smaller than the threshold. It visits this list of neighboring superpixels, adds them to the cluster and marks them as visited, then expands the list of neighbors with the neighboring superpixels of those, again, within a distance smaller than the threshold. So the list of neighbors is being expanded whenever there are connected superpixels being found within a distance smaller than the threshold. This process continues until the list of neighboring superpixels is exhausted. At this point a new cluster is started with a superpixel that has not been visited by the algorithm yet.

Because in this process, the label the newly made cluster is assigned is simply the label of the superpixel the cluster has been started with, the labels will not be sequential anymore. Therefore, the last step is a renumbering of all the clusters so their labels are again a sequential list $[1,\ldots,N]$ where N is the total number of clusters.

Figure 4.4 shows the results of the SLIC algorithm with and without the use of the DBSCAN algorithm to illustrate the difference. As can be seen, using the DBSCAN algorithm, even with a low threshold to make sure errors are definitely not being introduced, results again in a substantial reduction of the effective number of resulting superpixels. Note however, that in this particular case a threshold over 5 should not be used as that does introduce new errors. For

example at the top of the round tumor area on the bottom, stroma is taken with the tumor in the last two images.



Figuur 4.4: Left to right, top to bottom: The original image, the original overlaid with the superpixels and 5 images after using the DBSCAN algorithm for different thresholds (4, 4.5, 5, 5.5 and 6 respectively)

5 Results

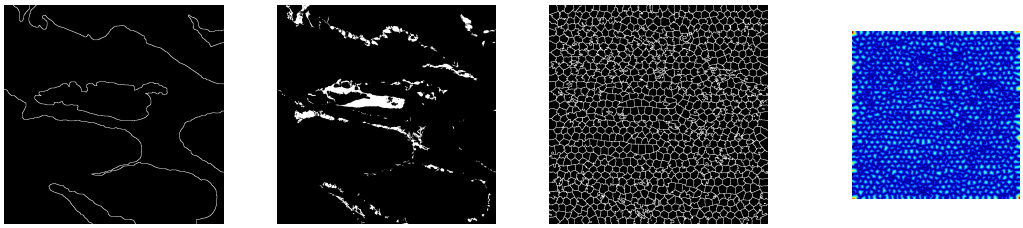
In the previous chapter the SLIC algorithm was explained in detail, a few intermediate results were shown as were the workings of the DBSCAN algorithm. But to be able to say something about how well the algorithm as a whole performs, a couple things are needed.

First a ground truth is needed. In the case of the H&E stained pathology images the ground truth was provided by someone with expertise in the area. An example of an H&E stained pathology image with its ground truth can be seen in columns (a) and (b) of Figure 5.2. The ground truth shows a partition of the original image in a black and a gray section. The black section represents tumor and the gray section stroma.

The second thing that is needed is a clear formulation of the requirements the superpixels have to meet. In other words, properties by which to indicate how good a set of superpixels is. Since the goal of the pre-processing method is to reduce the effective number of pixels without losing the information present in the image, the first property is readily stated: size. The superpixels have to be as large as possible. The larger the superpixels, the easier for the MRF-based segmentation algorithm. Of course, there is no size for the superpixels to refer to. This depends for example on the image size, but also on the features present in the image. The first row in Figure 5.2 shows an H&E stained pathology image with a fairly clear distinction between tumor and stroma so the superpixels can be larger in this case than when tumor and stroma are more intertwined (see for example the last two rows in Figure 5.2). However, if two sets of superpixels for the same image are compared and the accuracy is comparable, then the set with the largest superpixels is preferable. This leads to the second property: accuracy. The accuracy of a set of superpixels is a measure of how well the boundaries of the superpixels agree with the boundaries of the ground truth. This is also called boundary recall. Since even segmenting H&E stained pathology images by hand by an expert pathologist is prone to error, two limits were chosen. The lower limit was set to three and the upper limit to six. The lower limit says that if a superpixel boundary falls within three pixels of the ground truth boundary it is still counted as correct. If it is further away but falls within six pixels from the ground truth boundary it is still acceptable. Superpixel boundaries further away are counted as wrong. There is one exception to this. Even if a superpixel boundary pixel doesnâĂŹt fall within three pixels from the ground truth boundary, it can still be counted as correct if that pixel falls within a white area. As can be seen on the original H&E stained images (column (a) in Figure 5.2), there can be quite a lot of white in the image. This white matter is neither tumor nor stroma so if a superpixel boundary cuts through this white matter it does not influence the accuracy of the set of superpixels. Because measuring the accuracy this way is not exact and to still be able to distinguish between sets of superpixels with comparable accuracy, the average distance (in number of pixels) the superpixel boundaries are away from the ground truth boundaries is also calculated for the boundary pixels classified as acceptable.

5.1 Measurement algorithm

To determine which and how many boundary pixels from the ground truth can be classified as correct, acceptable and wrong, two edge maps are created: one from the ground truth and one from the superpixels generated by the SLIC algorithm. Next a distance transform is applied to the superpixel edge map. The distance transform determines for every pixel that is not a boundary pixel how far away from the closest boundary it is located. Lastly, as explained earlier, pixels are also classified as correct if they are further away from the ground truth boundaries than *LB* but fall within the white matter. Therefore, an image indicating the white matter connected to the ground truth boundaries is also required. These four images are shown in Figure 5.1 below. The white matter image is acquired by a simple relative thresholding operation



Figuur 5.1: Left to right: The ground truth edge map, white matter connected to the ground truth boundaries, superpixel edge map and the distance transform from the superpixel edge map

on the original H&E stained image. It is assumed that pixels with a luminance value larger than 0.7 times the maximum luminance value found in the image are white matter. The distance transform of the superpixel edge map is shown in color, where a darker color indicates a closer proximity to a border, for clarity.

Let A be the set of pixels belonging to the boundaries of the ground truth, B the set of pixels belonging to the boundaries of the superpixels and $d_B(A) \leq n$ a function that returns the ground truth boundary pixels within a distance n from a superpixel boundary. In the case that pixels within a distance $> LB$ and $\leq UB$ pixels from the ground truth border are classified as acceptable, this can now be expressed mathematically as follows:

$$A_{acceptable} = \{A | d_B(A) > LB \wedge d_B(A) \leq UB\} \tag{5.1}$$

Where LB and UB are the lower and upper bounds for accepted distances respectively. The percentage $P_{acceptable}$ of correctly classified pixels can be expressed as:

$$P_{acceptable} = \frac{|A_{acceptable}|}{|A|} \tag{5.2}$$

Similar equations hold for the pixels classified as wrong. The equation for correctly classified pixels has an extra condition that even when pixels are further away from the ground trust boundaries than LB pixels, they can still be classified as correct if they are in the white matter connected to the boundary. If C is the set of pixels consisting of white matter connected to the ground truth boundary then the equation becomes:

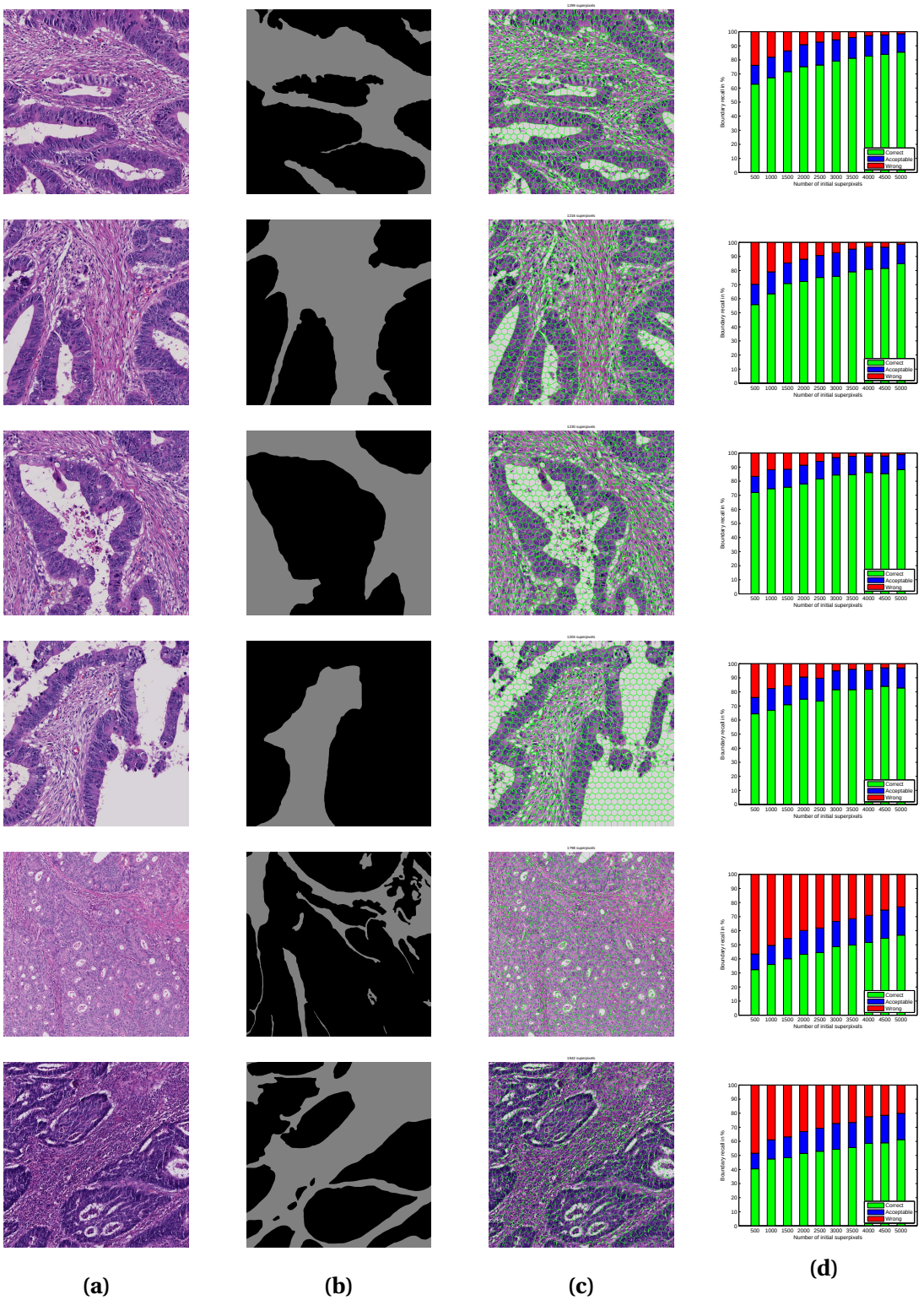
$$A_{correct} = \{A | d_B(A) \leq LB \vee C(A)\} \tag{5.3}$$

When interested in the average distance $d_{average}$ that pixels classified as acceptable are from the ground truth boundaries, this can be calculated as follows:

$$d_{average} = \sum_{n=LB+1}^{UB} \frac{|A_{average}(n)|}{|A|} n \tag{5.4}$$

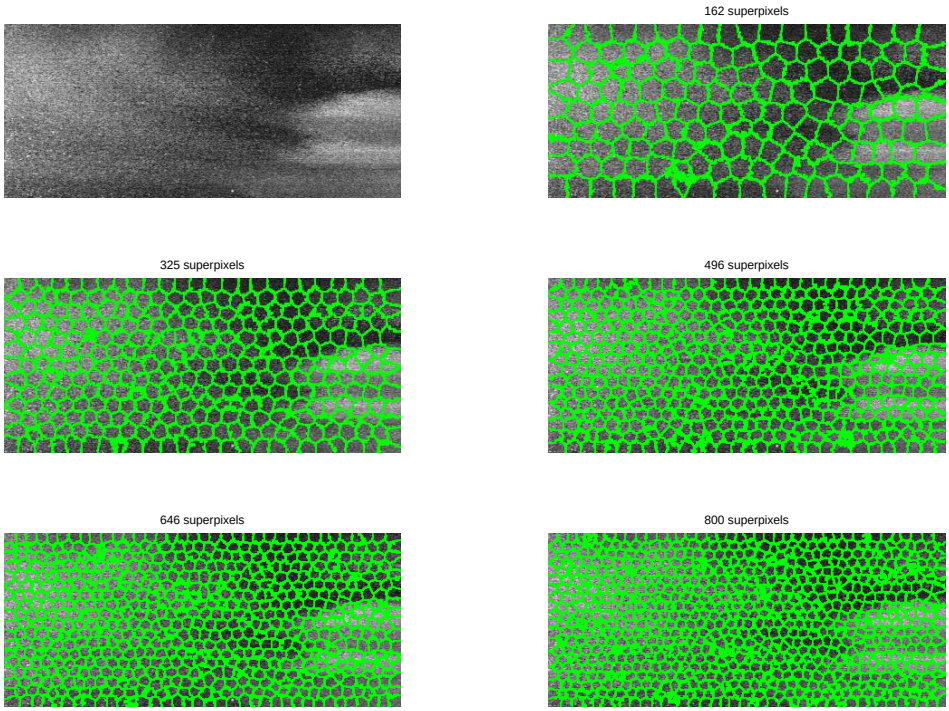
$A_{average}(n)$ is a function that returns the set of pixels classified as average at distance n from the ground truth boundaries. The results in column (d) in Figure 5.2 where obtained in this way and show a bar chart indicating which portion of the ground truth boundaries was classified as correct, acceptable and wrong for an increasing number of initial superpixels.

Unfortunately there is no ground truth available for the radar images which means that the superpixel results cannot be measured and expressed as they has been done for the H&E stained pathology images. However, visual inspection of the image and the result do show that the algorithm seems to work as intended. For the results see Figure 5.3 below.



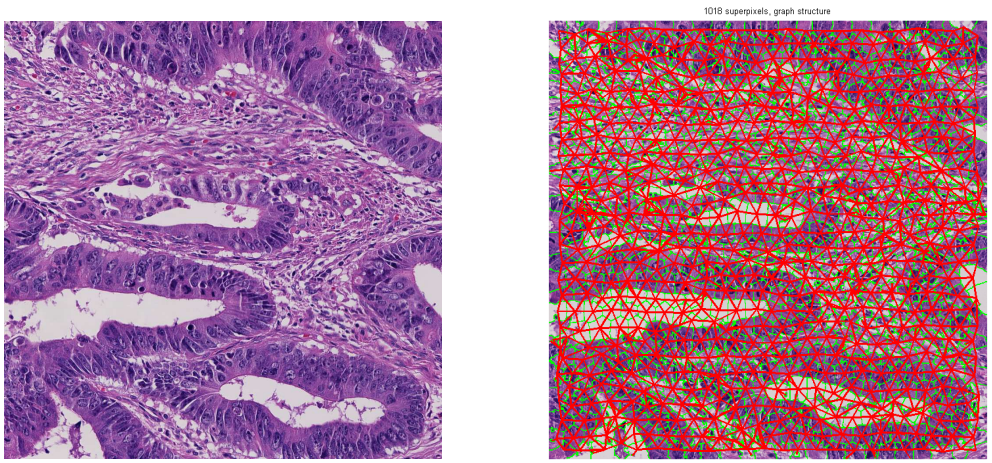
Figuur 5.2: (a) Original H&E stained image (b) The ground truth (c) Superpixels overlaid on the original image (d) Bar chart of boundary recall

The last thing left to do is show that the results can be used for an MRF-based segmentation algorithm. As explained earlier in this document, in order to be able to use an MRF-based segmentation algorithm, a neighborhood structure with cliques needs to be defined on the graph. It is therefore useful to visualize the graph after generating the superpixels. This is shown in



Figuur 5.3: The original radar image and the original image with the superpixels overlaid for 250, 500, 750, 1000 and 1250 inital superpixels respectively

red in Figure 5.4. There is an edge between every pair of directly neighboring superpixels. This means that as a local neighborhood of superpixel i, the superpixels connected to it with an edge can be taken. Then the single cliques are the individual superpixels in the neighborhood and pair-site cliques are superpixel i with one of its direct neighbors. Below is a small demonstration that will show how this can be used in practice.



Figuur 5.4: The original image and the original image overlaid with the superpixel boundaries in green and the graph structure in red.

5.1.1 Demo

To illustrate the ideas presented in this document and to prove that the superpixels can be used as input for MRF-based texture image segmentation, a small demonstration was implemented and will be discussed below. The aim of the demonstration was not to provide an optimal segmentation, but was kept simple on purpose to prove the ideas can be used in practice.

Recall that the Hammersley-Clifford theorem stated that the joint probability of a Markov Random Field can be expressed as the joint probability of a Gibbs Random Field:

$$P(\boldsymbol{x}) = \frac{1}{Z} e^{\frac{-1}{T} U(\boldsymbol{x})} \text{ with } U(\boldsymbol{x}) = \sum_{c \in \boldsymbol{C}} V_c(\boldsymbol{x}) \tag{5.5}$$

$U(\boldsymbol{x})$ is the energy function which is a summation over the clique potential functions. To define the clique potential functions, a choice of model needs to be made. In the case of the H&E stained pathology images, there are just two labels that need to be considered: tumor and stroma. To further simplify matters, only clique potentials up to two sites are considered resulting in the following energy function for superpixel X_i:

$$U(x_i) = V_1(x_i) + \sum_{i' \in N_i} V_2(x_i, x_{i'}) \tag{5.6}$$

Since there are just two labels to be considered, one speaks of an *auto-logistic* model and the potential functions can be expressed as $V_1(x_i) = \alpha_i x_i$ and $V_2(x_i, x_{i'}) = \beta_{i,i_{i'}} x_i x_{i'}$. By chosing the possibles a superpixel can take to be -1 or 1, i.e. tumor is represented by -1 and stroma by 1, the model is further reduces to the *Ising* model. In the *Ising* model α is commonly chosen as 0 and β as 1. This results in the following energy function:

$$U(x_i) = \sum_{i' \in N_i} x_i x_{i'} \tag{5.7}$$

and the conditional probability for superpixel x_i becomes:

$$P(x_i | N_i) = \frac{1}{Z} e^{\frac{-1}{T} x_i \left(\sum_{i' \in N_i} x_{i'} \right)} \tag{5.8}$$

One thing to clarify here is the choice of neighborhood. As was seen in Figure 5.4, the resulting superpixel graph is an irregular graph. The neighborhood of superpixel x_i in the above described model consists of all its direct neighbors. On a regular graph, the direct neighbors of a pixel are the four directly next to and above and under the pixel in a 4-connected neighborhood, or all eight surrounding pixels in an 8-connected neighborhood. In an irregular superpixel graph there is no telling on forehand how many direct neighbors a superpixel has and therefore this was written as a sum in Equation 5.8.

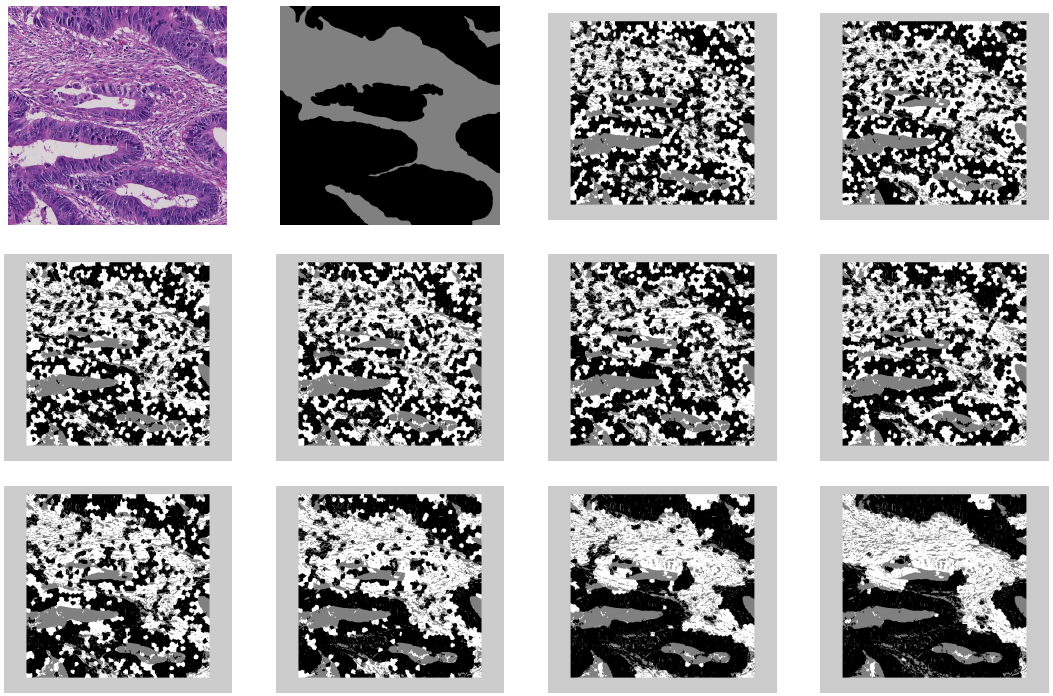
Now that the model has been established, the joint probability of the field needs to be computed and evaluated. That realization of the field with the highest probability, or in other words the lowest energy, has to be found. Looking again at Equation 5.5 it is clear that even when only considering two labels and n superpixels, to find the configuration of the field with the lowest energy amounts to evaluating 2^n configurations of the field. Even with a low amount of superpixels this very quickly becomes intractable. Therefore, a Gibbs sampler was used. The Gibbs sampler is a way to sample from the underlying probability distribution and by doing this enough times, get as close as possible to the global minimum. One starts with an initialization of the field and visits the superpixels in random order. For the visited superpixel probabilities are computed for every possible label that the superpixel takes on this label given its neighborhood by using the energy function derived above. In this case that results in a probability

that the superpixel should be classified as tumor and a probability that it should be classified as stroma, given how its direct neighbors are labeled. These probabilities are then multiplied by the probability density function of the observed data, i.e. the features extracted from the superpixel describe the likelihood of the superpixel taking a certain label based only on the observed data. According to the probabilities calculated, it is determined whether the assigned label of the superpixel should be changed or not. This process is repeated a number of times until no change in labels happens anymore. At this point the realization with the lowest energy has been found.

There is one thing that was not talked about here yet and that is the temperature parameter. As explained earlier, low values for the temperature enforce a higher interaction between superpixels and thus larger distinct regions are formed whereas a higher temperature drives the field to a more uniform distribution of the labels. See Figure 2.10 for an illustration of this idea. In classic Gibbs sampling the temperature is being kept constant at a value of 1 or slightly lower, e.g. 0.9. This means that in order to arrive a realization of the field with the lowest energy, the starting point, the initial random distribution of the labels, has to be a very accurate guess as to not end up in a local minimum.

To counteract this behavior and to not have to rely on a very good starting point, a scheme called Simulated Annealing (SA) was implemented. In SA it is possible to start with a random realization of the field and update the labels as described above starting with a high temperature, and at every iteration lowering the temperature slightly. This results in a more accurate sampling and it is less likely to settle at a local minimum instead of the global minimum. As a more intuitive explanation, a high temperature motivates a more uniform distribution of labels so an iteration with a high temperature can be thought of as violently shaking the field. This results in a new realization that is quite different of the previous one. When lowering the temperature gradually, the shaking becomes less and less violent until a state is reached where the field does not change anymore. This is visualized in figure 5.5 where the temperature is gradually lowered over ten iterations. It is visible that the early iterations shake up the field pretty hard whereas the later iterations change the field less and less drastically.

As can be seen in Figure 5.5, after a number of iterations the realization of the fields comes closer and closer the ground truth. In Figure 5.5 tumor is represented by black, stroma by white and white matter by gray. The demonstration does not result in a perfect segmentation but as explained earlier serves the purpose of showing that superpixels can indeed be effectively used as input to an MRF-based segmentation algorithm. The result can be significantly improved by e.g. choosing a more accurate model, considering a larger neighborhood system and extracting more and better features from the superpixels. In this case, the only features used were the L*a*b* colors of the superpixel.



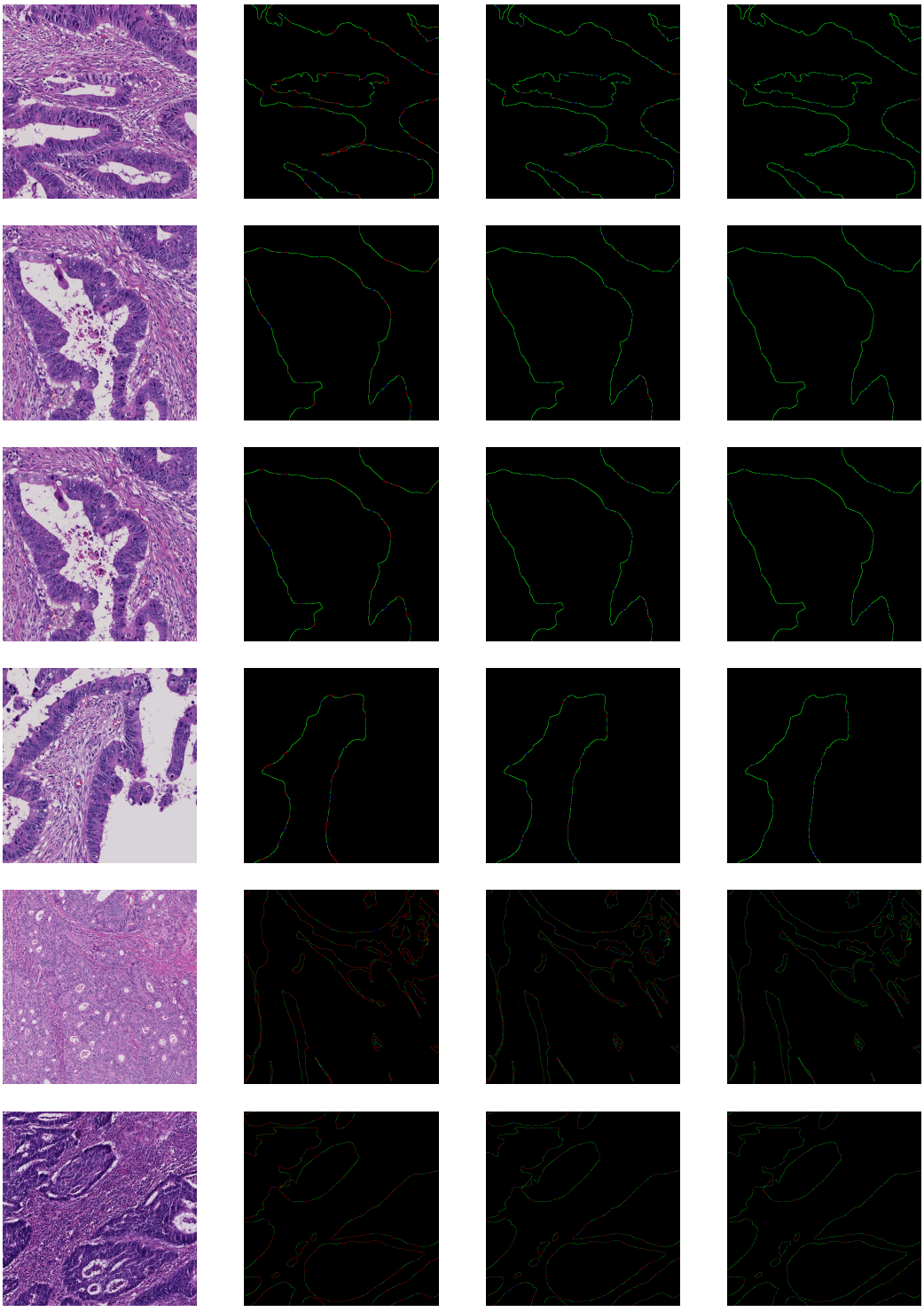
Figuur 5.5: The original H&E stained pathology image with its ground truth followed by ten iterations of the MRF-based segmentation algorithm

6 Discussion

The previous chapter explained the method by which the accuracy of the superpixel algorithm was measured and showed some of the results in Figure 5.2. As expected, column (d) shows that with a higher number of initial superpixels the accuracy of the algorithm becomes better. The images shown are all 1000x1000 pixels (one megapixel). Results for the larger images are not shown here because the images are so large that when displayed here it is hard to distinguish anything in them. However, as expected, the results do not change with images of different size. If two images are comparable in nature (contrast, the way the tumor and stroma are intertwined) but the second image is four times larger, then the results are approximately the same for the same size superpixels. I.e. the larger image in this case needs to start with four times more initial superpixels. Also, a correctly chosen threshold for the DBSCAN algorithm will not change the accuracy of the superpixel algorithm. As long as no new errors are introduced, i.e. a too high threshold resulting in superpixels covering stroma being merged with superpixels covering tumor or vice versa, the accuracy of the superpixels algorithm with or without the use of DBSCAN will be the same.

To get an idea of what areas in an image are problematic, the results can also be shown as in Figure 6.1. Here the parts of the ground truth boundaries are colored according to how the pixels are classified.

When looking at the places in the original image where the ground truth boundaries are colored red, or, equivalently, classified as wrong, and the superpixels boundaries thus do not adhere well to the tumor/stroma boundaries in the image, it can be seen that this is the case in places where a distinction between stroma and tumor is very hard to make. It also occurs at locations in the image where the objects (tumor or stroma) are smaller than the superpixel size. This can be seen in the image in Figure 6.1 by looking at the very narrow piece of stroma between the larger tumor segments. Because the algorithm enforces a certain level of regularity for the superpixels, it can be that when the superpixel size is chosen too large, the trade-off between color and proximity importance is made in favor of proximity. This results in 'prettier' superpixels in the sense that they are shaped more regularly, but has as a side-effect that they do not adhere to object boundaries very well. This effect can be countered by choosing the superpixels smaller. As a rule of thumb the size of the superpixels should be chosen no larger than the smallest object that one wants to be able to identify in the image. This is, however, a trade-off itself since generating more superpixels takes longer. As explained in the previous section, the size of the superpixels is also a property by which to state how well the algorithm performs, but using the DBSCAN algorithm will eventually reduce the final number of superpixels again. As mentioned earlier, the DBSCAN algorithm is very fast so that the output of the superpixel algorithm can be used to try different threshold values.



Figuur 6.1: (a) Original H&E stained image (b-d) ground truth boundaries colored according to how their pixels are classified for an initial number of superpixels of 500, 2500 and 5000 respectively

7 Conclusion and future work

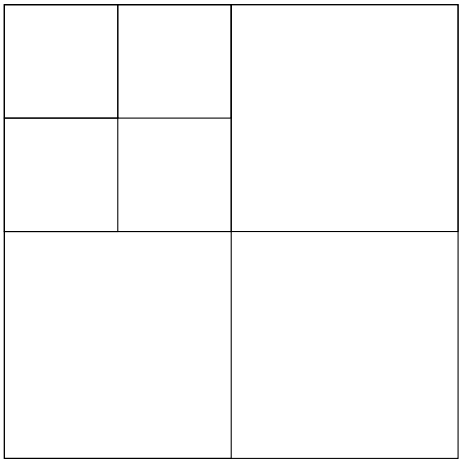
In Chapter 2 this document tried to explain why Markov Random Fields are a good choice when dealing with images that contain globally varying textures in an unpredictable way, are very large or are subject to heavy noise causing outliers. This last property can occur in the H&E stained pathology images for example when more of the staining material attaches to certain parts of the stroma or tumor resulting in very bright or dark spots that should still be classified as belonging to the rest of the stroma or tumor. MRF's can infer the true class of these spots given its surroundings. In the case of radar images the same argument can be made. These images are sensitive to speckle noise or point targets appearing in the environment of the radar. In this case MRF's can also infer the true class of the superpixels giving its surroundings.

Chapter 3 explained that most of the time a pre-processing step is run on the raw data in order to prepare it for the main algorithm. This means that the data is transformed into a form more suited for the algorithm following it. Face recognition was used as an example where the pre-processing consists of face registration and translation resulting in a standard form for the faces to then be processed further. In this case, a pre-processing step is required to bring down the effective number of pixels in the image without losing the information present in it. The choice for SLIC over other superpixel algorithm was motived and the algorithm was explained in detail including improvements and extensions. Especially the DBSCAN algorithm is an important step. The results showed that more accurate results were obtained when the number of superpixels is larger, but the objective was to bring the number of effective pixels down as far as possible. The DBSCAN algorithm performs a quick clustering on the superpixels, merging superpixels together that definitely belong together without introducing new errors as long as the threshold is chosen low enough.

Chapter 5 then showed the results of applying the superpixel algorithm and, in the case of the H&E stained pathology images, verified these results. It was observed that with more initial superpixels the accuracy of the algorithm increased. Chapter 5 also identified problematic areas and images and showed that, although superpixels are performing very well on a lot of images, there is still room for improvement. Therefore, the first step in improving the algorithm is to find a way to make the superpixel algorithm more effective on images with low contrast between stroma and tumor or when stroma and tumor are highly intertwined. This can be done by extending the distance measure, taking more features into account than just color and distance, e.g. the dominant direction of the texture, or by trying to enhance the contrast between tumor and stroma in the images. It would also be very helpful to have more reference data for the radar images so the accuracy of the superpixel algorithm can also be verified on these images.

Since the focus here was not on optimizing the algorithm for speeds but rather on correctness and accuracy, this is also future work. At this time the algorithm is especially slow on larger images with a high number of superpixels. This can most likely be improved by a significant amount by splitting the original image into sub images, e.g. like shown in Figure 7.1, running the algorithm on the small sub images with a correspondingly smaller number of superpixels and later merging the results.

With an algorithm optimized for speed the initial number of superpixels can be set quite high to achieve a higher accuracy, but DBSCAN still requires some manual inspection of the image and results to determine the optimal threshold value resulting in the smallest number of superpixels while not losing in accuracy. To make the algorithm completely automatic it is also desirable to find a way to determine the optimal DBSCAN threshold value for a particular image. This would eliminate completely the need for any human supervision.



Figuur 7.1: Division of the original image in smaller sub images

The next step would be to actually implement an MRF-based segmentation algorithm, fed with features extracted from the superpixels. An example of a feature for the H&E stained pathology images could, again, be color since tumor tends to be significantly darker than stroma. Also, direction of the texture in the superpixel compared with that of its surrounding superpixels can be used as a feature since the general direction of tumor and stroma tend to be perpendicular to each other. Another discriminating feature is the size of the cells and nuclei contained within tumor and stroma. An algorithm like SIFT (Scale Invariant Feature Transform, Lowe (1999)) can be used to function as a 'blob' detector. The size and the blobs and the amount of them can be a good indicator to determine whether the particular superpixel should be classified as tumor or as stroma. The dominant direction of the texture within a superpixel can most likely also be used as a feature for superpixels created for radar images, but, unfortunately, an algorithm like SIFT will most likely not work for radar images. In this case it is probably better to look at second-order statistics, obtained for example by using co-occurrence matrices (Gotlieb and Kreyszig (1990)). A co-occurrence matrix keeps track of the frequency of co-occurrence values at a given offset within the superpixel. From these co-occurrence matrices, statistics like the entropy, contrast (also known as variance), energy, correlation or homogeneity can be computed. These values are a representation of the superpixel and can be used for classification of the superpixels.

Bibliografie

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua and S. SuÌĹsstrunk (2010), SLIC Superpixels, Technical Report EPFL 149300.

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua and S. SuÌĹsstrunk (2012), SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, **vol. 34**, no.11, pp. 2274–2282, ISSN 0162-8828, doi:10.1109/tpami.2012.120.

Besag, J. (1974), Spatial interaction and the statistical analysis of lattice systems, *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 192–236.

Besag, J. (1986), On the statistical analysis of dirty pictures, *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 259–302.

Chellappa, R. and S. Chatterjee (1985), Classification of textures using Gaussian Markov random fields, **vol. 33**, no.4, pp. 959–963.

Cohen, F. S., Z. Fan and M. A. Patel (1991), Classification of rotated and scaled textured images using Gaussian Markov random field models, **vol. 13**, no.2, pp. 192–202.

Comaniciu, D. and P. Meer (2002), Mean shift: A robust approach toward feature space analysis, **vol. 24**, no.5, pp. 603–619.

Cooper, D. (1979), Markov-Process Blob Boundaries in Noisy Images, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **vol. PAMI-1**, pp. 172–384.

Cooper, D. B., H. Elliott, F. Cohen, L. Reiss and P. Symosek (1980), Stochastic boundary estimation and object recognition, **vol. 12**, no.4, pp. 326–356.

DeMenthon, D. and R. Megret (2002), *Spatio-temporal segmentation of video by hierarchical mean shift analysis*, Computer Vision Laboratory, Center for Automation Research, University of Maryland.

Derin, H. and W. S. Cole (1986), Segmentation of textured images using Gibbs random fields, **vol. 35**, no.1, pp. 72–98.

Derin, H., H. Elliott, R. Cristi and D. Geman (1984), Bayes smoothing algorithms for segmentation of binary images modeled by Markov random fields, , no.6, pp. 707–720.

Ester, M., H. P. Kriegel, J. Sander and X. Xu (1996), A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in *Second International Conference on Knowledge Discovery and Data Mining*, Eds. E. Simoudis, J. Han and U. Fayyad, AAAI Press, Portland, Oregon, pp. 226–231.

Geman, S. and D. Geman (1984), Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, , no.6, pp. 721–741.

Gotlieb, C. C. and H. E. Kreyszig (1990), Texture descriptors based on co-occurrence matrices, **vol. 51**, no.1, pp. 70–86.

Hammersley, J. and P. Clifford (1971), Markov field on finite graphs and lattices, *Unpusblished*.

Hu, R. and M. M. Fahmy (1992), Texture segmentation based on a hierarchical Markov random field model, **vol. 26**, no.3, pp. 285–305.

Jeffs, B. D. and M. Gunsay (1993), Restoration of blurred star field images by maximally sparse optimization, **vol. 2**, no.2, pp. 202–211.

Jong-Sen and Lee (1983), Digital image smoothing and the sigma filter, **vol. 24**, no.2, pp. 255–269, doi:10.1016/0734-189X(83)90047-6.

Kashyap, R. L. and R. Chellappa (1983), Estimation and choice of neighbors in spatial-interaction models of images, **vol. 29**, no.1, pp. 60–72.

Kindermann, R., J. L. Snell et al. (1980), *Markov random fields and their applications*, volume 1, American Mathematical Society Providence, RI.

Kwang-Shik, K., Z. Dongni, K. Mun-Cheon and K. Sung-jea (2013), Improved simple linear iterative clustering superpixels, in *Consumer Electronics (ISCE), 2013 IEEE 17th International Symposium on*, IEEE, pp. 259–260.

Lakshmanan, S. and H. Derin (1989), Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing, **vol. 11**, no.8, pp. 799–813.

Levinshtein, A., A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson and K. Siddiqi (2009), TurboPixels: Fast Superpixels Using Geometric Flows, **vol. 31**, no.12, pp. 2290–2297, ISSN 0162-8828, doi:10.1109/tpami.2009.96.

Li, S. Z. (2009), *Markov random field modeling in image analysis*, Springer Science & Business Media.

Lowe, D. G. (1999), Object recognition from local scale-invariant features, in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, Ieee, pp. 1150–1157.

Macqueen, J. B. (1967), Some Methods for classification and analysis of multivariate observations, in *Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, volume 1, University of California Press, pp. 281–297.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller (1953), Equation of state calculations by fast computing machines, **vol. 21**, no.6, pp. 1087–1092.

Meyn, S. P. and R. L. Tweedie (1993), *Markov chains and stochastic stability*, Springer–Verlag.

Phillips, D. and A. Smith (1994), Bayesian faces via hierarchical template modeling, **vol. 89**, no.428, pp. 1151–1163.

Politis, D. N. (1994), Markov chains in many dimensions, *Advances in Applied Probability*, pp. 756–774.

Shi, J. and J. Malik (2000), Normalized cuts and image segmentation, **vol. 22**, no.8, pp. 888–905, ISSN 0162-8828, doi:10.1109/34.868688.

Simchony, T., R. Chellappa and Z. Lichtenstein (1990), Relaxation algorithms for MAP estimation of gray-level images with multiplicative noise, **vol. 36**, no.3, pp. 608–613.

Sonka, M., V. Hlavac and R. Boyle (2007), *Image Processing, Analysis, and Machine Vision*, Thomson-Engineering, ISBN 049508252X.

Steinhaus, H. (1956), Sur la division des corp materiels en parties, *Bull. Acad. Polon. Sci*, **vol. 1**, pp. 801–804.

Vincent, L. and P. Soille (1991), Watersheds om Digital Spaces: An Efficient Algorithm Based on Immersion Simulations, **vol. 13**, no.6.

Zhang, J. (1993), The mean field theory in EM procedures for blind Markov random field image restoration, **vol. 2**, no.1, pp. 27–40.