universität wien

MASTERARBEIT

Titel der Masterarbeit

"On Image Segmentation and Applications in Clinical Retinal Analysis"

Verfasser

Anna Breger, BSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2015

Studienkennzahl lt. Studienblatt:	A 066 821
Studienrichtung lt. Studienblatt:	Masterstudium Mathematik
Betreuer:	Ass. - Prof. Dr. Martin Ehler

Contents

Abstract

This thesis deals with mathematical image segmentation, which is the process of dividing a digital image into multiple segments, and its application to clinical images of the human retina. At first we describe some different important image segmentation techniques including direct and variational methods. One significant approach is to receive a segmented image as the minimizer of an appropriate energy functional. We will introduce the *Potts model* as an energy minimization problem on continuous regions and will also consider a discrete domain version. The continuous model is also known as the *piecewise constant Mumford-Shah model* since it is similar to the energy functional introduced by Mumford and Shah. We will work with the discrete Potts model and its minimization, which is a NP - hard nonconvex optimization problem. Therefore we will discuss two different algorithms for a numerical approximation: the *Graph Cuts* method and the *ADMM*. In conclusion we will use an algorithm for minimizing the Potts model via the ADMM to process clinical images of the human retina given by the Vienna Reading Center (VRC) at the Department of Ophthalmology, Medical University of Vienna. Our goal is to segment those images in different areas such that intraretinal cystoid fluid can be located in the retina. Different resulting segmentations of those clinical images are shown at the end of the thesis.

Zusammenfassung

Diese Masterarbeit beschäftigt sich mit mathematischer Bildsegmentierung, der Unterteilung eines digitalen Bildes in verschiedene Segmente, und deren Anwendung auf klinische Bilddaten der menschlichen Netzhaut.
Am Anfang beschreiben wir verschiedene wichtige Techniken der Bildsegmentierung. Ein in der Literatur oft beschriebener Ansatz ist die Segmentierung eines Bildes durch die Minimierung eines passenden Energiefunktionals.
Dann stellen wir das *Potts-Modell* vor als ein Energieminimierungsproblem auf kontinuierlichen Regionen und geben auch eine diskrete Formulierung des Problems. Das kontinuierliche Modell ist auch bekannt als das *stückweise konstante Mumford-Shah-Modell*, da es sehr ähnlich ist zu dem Funktional, das von Mumford und Shah vorgestellt wurde. Wir arbeiten weiter mit dem diskreten Potts-Modell und seiner Minimierung, welches ein nichtkonvexes NP-Problem ist. Hierfür werden zwei verschiedene algorithmische Lösungansätze näher beschrieben und diskutiert: die *Graph Cuts* Methode und der *ADMM*.
Schließlich wird ein Segmentierungsalgorithmus basierend auf dem beschriebenen ADMM verwendet um auf klinische Bilder der menschlichen Netzhaut anzuwenden, welche vom Vienna Reading Center (Fakultät für Ophthalmology, Medizinische Universität Wien) zur Verfügung gestellt wurden. Unser Ziel ist es diese Bilder so zu segmentieren, dass Zysten in der Netzhaut erkannt werden. Einige der numerischen Resultate werden am Schluss der Arbeit präsentiert.

Acknowledgment

I would like to express my gratitude to my supervisor Martin Ehler for the helpful comments, remarks and his supportive guidance through the learning process of this master's thesis. This work was partially supported by the Vienna Science and Technology Fund (WWTF) through project VRG12-009. Furthermore I would like to thank the VRC for providing clinical image data of the human retina (note that the copyright remains with the VRC). I would like to thank all those colleagues who took their time to discuss (mathematical) problems with me during the last years, especially those who became very close friends through that intense time. Without you the last years would not have been such a great and fruitful time. I would also like to thank my loved ones, who have supported me throughout the entire process by keeping me in a good mood, always listening to my concerns and helping wherever they could.

1 Introduction

Mathematical image processing is an important and wide field of Applied Mathematics that is closely related to Computer Science and Engineering. It is essential in many applications including medical image analysis, image compression, movie processing, computer aided quality control, information security, etc ... .

Image segmentation is the process of dividing a digital image into multiple segments (piecewise constant areas) to change the representation of an image. It is a difficult and important problem so that in the literature already several different approaches (that belong to very different theoretical mathematical fields) exist.

First I will state in Chapter 2 some different important approaches of mathematical image segmentation including direct and variational methods. It is important to be aware that some methods are developed for images on a discrete domain and some for images on a continuous domain. I will introduce the Potts model, whose minimizer shall yield a segmented image, in Section 2.2.5 as an energy minimization problem on continuous regions, but a discrete domain version is considered in [SW14]. The continuous model is also known as the *piecewise constant Mumford-Shah model* since it is similar to the energy functional introduced by Mumford and Shah in [MS89]. In the further chapters I will work with the discrete Potts model and its minimization. Since the minimization of the Potts model is a NP - hard nonconvex optimization problem, I am describing in the following chapters two different algorithms for a numerical approximation. In Chapter 3 the combinatorical Graph Cuts method, which is the basis of important algorithms for solving energy minimization problems, is described in general and on the Potts model. The method is widespread, but there are also some disadvantages in those algorithms applied to vector valued images. An alternative are algorithms based on the ADMM (alternating direction method of multipliers) which can sometimes lead better results. In Chapter 4 I am going to describe the alternating direction method of multipliers (ADMM) and two related methods, whose advantages shall be combined in the ADMM. A general convergence result (based on the one by S. Boykov in [BV99]) will be stated and proved in Appendix A. Additionally I will state the application of the ADMM to the Potts model including a different convergence result. In Chapter 5 the algorithm developed in [SW14] for minimizing the Potts model via ADMM is used to process images of the human retina given by the Vienna Reading Center (VRC) at the Department of Ophthalmology, Medical University of Vienna. They also provided those images manually segmented for ground truth information. The VRC has extensive medical expertise and is one of the leading institutions in ophthalmic image analysis. Our goal is to segment those images in different areas such that intraretinal cystoid fluid can be located in the retina. Different resulting segmentations of those clinical images are shown at the end.

2 Mathematical Image Processing

In that master's thesis I am going to work with digital greyscale and vector valued images, which consist of more than one channel. The most common vector valued images are rgb color images, which consist of three channels with color intensity values of red, green and blue. Digital rgb images are multispectral images, which means that each channel represents a certain measured wave length taken by a camera. In general vector-valued images consist of k channels (=features), where every channel can also contain calculated values, e.g. the gradient or the laplacian of the color intensity values, a filtered image, etc... Those values do no longer need to be physically measured length of waves, but also can provide important calculated information of the original image.

2.1 Image representation

Although digital image data is defined on discrete regions $\Omega = \{0,1,\dots,m-1\} \times \{0,1,\dots,n-1\}$ there exists important and helpful theory on continuous regions $\Omega \subset \mathbb{R}^2$ that can be adapted to the discrete case.

2.1.1 Images on continuous regions

Every vector valued image $I \in \Omega \times \mathbb{R}^r$, where $\Omega \subset \mathbb{R}^2$ denotes the image area, can be described by a label function $u : \Omega \to \mathbb{R}^r$. The function u returns a vector $z \in \mathbb{R}^r$ in every point $(x,y) \in \Omega$, that consists of the information given in all r channels of the image, e.g. in a rgb color image $u(x,y)$ returns a three dimensional vector with the color intensities for red, green and blue in the point (x,y).

2.1.2 Images on discrete regions

Digital images on a discrete domain $\Omega = \{0,1,\dots,m-1\} \times \{0,1,\dots,n-1\}$ can be viewed as a matrix $I \in \mathbb{R}^{m \times n \times r}$ consisting of pixels $p_{ij} \in \mathbb{R}^r$, where r denotes the number of channels. The pixels can be labeled by a label function $u : \Omega \to \mathbb{R}^r$, e.g. the intensity function $u(i,j)$ returns the color intensity values in all channels of the pixel p_{ij} on the place $(i,j) \in \Omega$. Usually color intensity values are between 0 and 255 (in the case that the pixel intensity is stored as an 8-bit integer). A greyscale image is given by a matrix $I \in \mathbb{R}^{m \times n}$ and a rgb color image is given by a matrix $I \in \mathbb{R}^{m \times n \times 3}$.

Remark. Since we work in that chapter just with greyscale and rgb images, we assign each pixel of the image matrix $I = (p_{ij})_{i=1\dots m-1, j=1\dots n-1}$ with its color intensity values, i.e. $p_{ij} :=$

$u(i,j) \in \mathbb{R}$ or $\mathbb{R}^3$, where u returns the color intensities. Note that in the discrete case each label function u can be viewed as a $\mathbb{R}^{m\times n\times r}$ matrix.

2.2 Image segmentation

Image Segmentation is a very important and wide field in image analysis and computer vision. The main idea is to partition a given image I into regions with similar properties or to find boundaries of distinct objects in the scene (boundary detection). There do exist several different approaches on solving those problems. In this chapter I will give a short overview of some different ideas, including energy minimization and the Potts model, whose approach will be subject of the further chapters.

Remark. Image denoising is closely related to image segmentation ([CEN04]). The goal in that case is to remove noise or to recover details of a given corrupted digital image.

2.2.1 Edge detection for greyscale images

Edge detection techniques decide wether a pixel is part of an edge (a boundary between regions) or not. With that information one can partition the image in objects and background. There do exist several different approaches in the literature, some surveys can be found in [RKS13], [SR09] and [HB13]. I will give a short summary of popular greyscale edge detection methods on discrete regions. Those methods are also included in the image processing toolbox of matlab. They return a binary image of the same size as the original image with binary pixels, such that $p_{ij} = 1$ (white) on edges, and $p_{ij} = 0$ (black) elsewhere.

There are mainly two different categories of edge detection methods: gradient and Laplacian ones. The gradient methods detect edges by looking for the maximal gradient of the intensity values in the image (the gradients of the label function u). Since edges can be viewed as the set of those pixels with high spatial frequency, finding the pixels with a high first derivative can yield the edges. On the contrary the Laplacian methods look for zero crossings in the second derivative. This makes sense because the second derivative equals 0 when the first derivative is at a maximum (i.e. very high gradient value).

Remark. All the described filter methods yield kernels (convolution masks), that give the wanted result by convolving them with the original image.

For a suitable continuous image $I \in \mathbb{R}^2$ the continuous convolution with a kernel function $w : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ is given by

$$\big(w * I\big)(x,y) = \int_\Omega w(x-u, y-v) I(u,v)\, du\, dv.$$

Similarly the discrete convolution of a discrete image $I \in \mathbb{R}^{m\times n}$ with an $(2k+1)\times(2l+1)$ mask w is given by

$$(w * I)_{(m,n)} = \sum_{i=-k}^{k} \sum_{j=-l}^{l} w(i,j) I(m-i, n-j).$$

Gradient methods

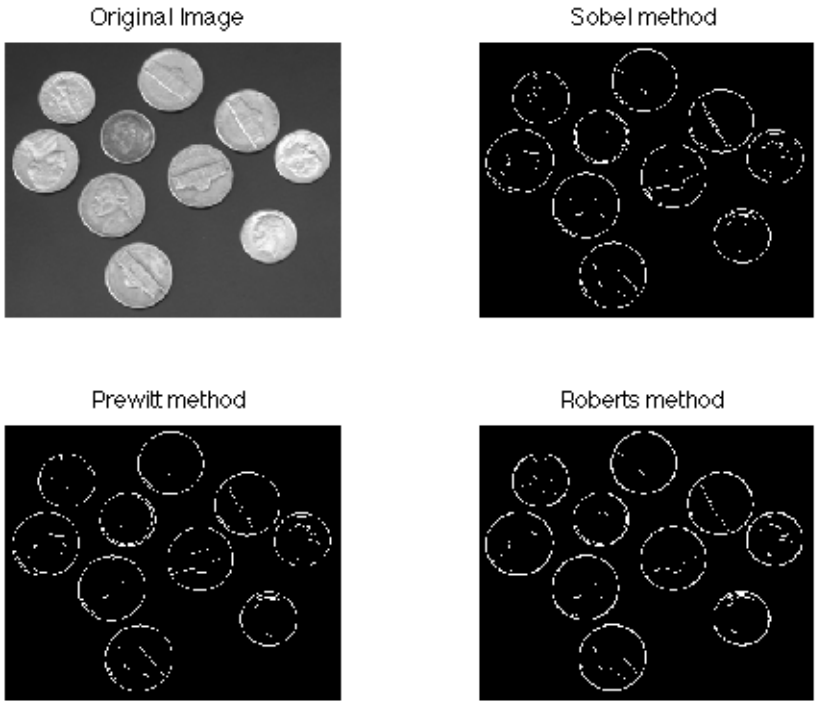


Figure 2.1: Using the matlab Sobel, Prewitt and Roberts methods on a greyscale image.

Sobel method
The suggestion of the Sobel Operator was made in a talk by Irwing Sobel in 1968 [Sob14]. The Sobel operator is a discrete differentiation operator, which computes an approximation of the absolute gradient magnitude at each point of the discrete input image. It returns edges at those points where the estimated gradient exceeds some previously chosen threshold. The standard Sobel Operator uses a 3×3 neighborhood for approximating the gradient in each pixel, e.g. the neighborhood for the pixel e is given by

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}.$$

Note that it cannot be used on the boundary of the image. To get an estimate of the spatial gradient in pixel e, a vector sum of 4 directional derivatives of orthogonal vectors is calculated [Sob14]. Therefore the 8 neighbors of the central pixel group into antipodal pairs: $(a,i),(b,h),(c,g)$ and (f,d). To get the estimated gradient in pixel e, one sums up the discrete directional derivative of all pairs and takes the average. The directional derivative is defined by the difference of the intensities divided by the distance of the pixels. Therefore the vector

sum for the gradient estimate in pixel e is given by

$$G(e) = \frac{1}{R^2}\left((c-g)\cdot[1,1]^T + (a-i)\cdot[-1,1]^T\right) + (b-h)\cdot[0,1]^T + (f-d)\cdot[1,0]^T,$$

where $R = \sqrt{2}$ denotes the pixel distance. Multiplying by 2 gives the vector

$$G(e) = \begin{pmatrix}(c-g-a+i)+2(f-d)\\(c-g+a-i)+2(b-h)\end{pmatrix}.$$

This leads to a pair of 3×3 convolution kernels that give the approximate spatial gradient. Therefore for each point the estimated discrete derivative in the x direction can be calculated by convolving the image with

$$G_x = \begin{pmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{pmatrix}$$

and in the y direction with

$$G_y = \begin{pmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{pmatrix}.$$

The gradient magnitude is then given by

$$|G| = \sqrt{G_x^2 + G_y^2}$$

and the direction of the gradient by

$$\theta = \arctan\left(\frac{G_x}{G_y}\right).$$

A detailed description and an extension to a 5×5 mask can be found in [GM13].

Prewitt method
The Prewitt method is quite similar to the Sobel method. The 3×3 convolution kernels only differ because the directional derivates are now not weighted by the pixel distance. So unlike the Sobel operator, this operator does not emphasize pixels that are closer to the center of the mask. Therefore the two masks are given by

$$G_x = \begin{pmatrix}-1 & 0 & 1\\-1 & 0 & 1\\-1 & 0 & 1\end{pmatrix}$$

and

$$G_y = \begin{pmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{pmatrix}.$$

Again the gradient magnitude is given by

$$|G| = \sqrt{G_x^2 + G_y^2}.$$

Roberts method
The Roberts method is also a gradient-magnitude method. Instead of a 3×3 neighborhood it uses a 2×2 neighborhood to get an approximate discrete gradient measurement. It uses the two masks

$$G_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

and

$$G_y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Since that operator uses a smaller neighborhood to calculate the gradient image, it is even more sensitive to noise than the Sobel and Prewitt method.

Canny method
The classical gradient based operators Roberts, Prewitt and Sobel are weak concerning sharp edges and are highly sensitive to noise in the image ([RKS13]). Also the Laplacian based Marr and Hildreth operator (discussed later) has a high probability of detecting false edges. In 1986 John F. Canny wanted to create an ideal edge detection algorithm for images that are corrupted with noise. He developed one that reduces the probability of detecting false edge and yields sharp edges.

As described in [RKS13], the algorithm of the Canny method seperates in 4 steps:

1. **Smoothing** Blurr the image to remove noise by convolving with a Gaussian Filter.

2. **Finding gradients** The Sobel operator is used to find the approximative gradients of all pixels in the smoothed image. Edges are marked where the gradients of the image are large.

3. **Non - maximum suppression** Only local maxima should be marked as edges to yield sharp edges. The algorithm for each pixel p_{ij} in the image is given by:
   a) round the gradient direction θ of p_{ij} to the nearest 45 degrees;
   b) compare the gradient magnitude $|G(p_{ij})|$ with the magnitude of the pixels in positive and negative gradient direction, e.g. if the rounded gradient direction is 90 degrees compare $|G(p_{ij})|$ with $|G(p_{(i-1)j})|$ and $|G(p_{(i+1)j})|$;
   c) if $G(p_{ij})$ has the largest value, save the value of the gradient and mark the pixel p_{ij} as edge pixel, otherwise suppress it.

4. **Double thresholding** The output of the non-maximum suppression still contains local maxima that are created by noise. To suppress those, the method uses double thresholding with a high and a low bound t_{high} and t_{low}. To decide wether a pixel is part of the

edge or not, we take the gradient magnitude of each pixel p_{ij} that is marked as an edge.

$$If \begin{cases} |G(p_{ij})| < t_{low} & \text{discard } p_{ij}, \\ |G(p_{ij})| > t_{high} & \text{keep } p_{ij}. \end{cases}$$

If $t_{low} < |G(p_{ij})| < t_{high}$ holds, then

$$if \begin{cases} \exists (k,l) \in N_3(p_{ij}) : |G(p_{kl})| > t_{high} & \text{keep } p_{ij}, \\ \exists (k_1,l_1) \in N_3(p_{ij}) : & \\ \quad t_{low} < |G(k_1,l_1)| < t_{high} \wedge \exists (k_2,l_2) \in N_5(p_{ij}) : |G(k_2,l_2)| > t_{high} & \text{keep } p_{ij}, \\ else & \text{discard } p_{ij}, \end{cases}$$

where $N_d(p_{ij})$ denotes the $d \times d$ neighborhood of p_{ij}. After that step the set of edge pixels is finally selected.

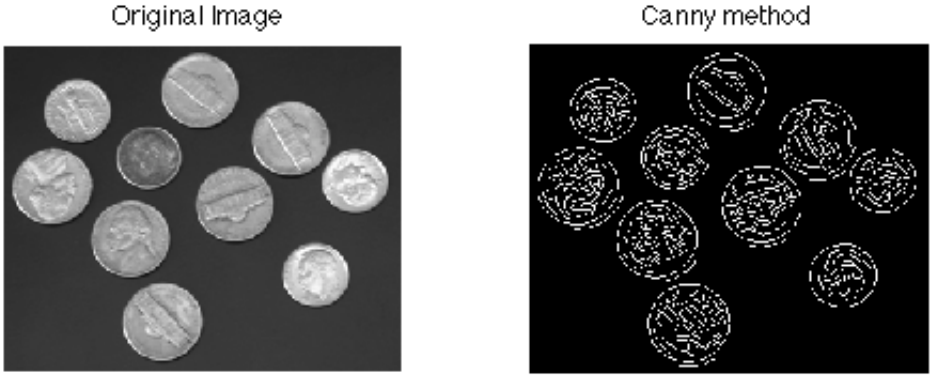


Figure 2.2: The matlab Canny method applied on a greyscale image.

Laplacian methods

In those methods it is necessary to find an approximation for the continuous Laplace operator $L(x,y) = \Delta u(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ and to adapt it to a discrete one $L(i,j)$. To get an approximation for the second derivative of the continuous function $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ we take its Taylor series expansions

$$u(x+h) = u(x) + hu'(x) + \frac{h2}{2}u''(x) + \frac{h^3}{3!}u'''(x) + O(h^4)$$

and

$$u(x-h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3!}u'''(x) + O(h^4).$$

If we add those two expansions it leads to the approximation

$$u''(x) \approx \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}.$$

To get an approximation of the discrete second derivative in each pixel of the discrete image, we set $h = 1$ as the step length between the pixels. That leads to an often used numerical approximation of second order derivatives, which is derived from the discrete central difference operator (see e.g. [SK11] or [uTN12]):

$$u_{xx}(i,j) \approx u(i+1,j) - 2u(i,j) + u(i-1,j)$$

and

$$u_{yy}(i,j) \approx u(i,j+1) - 2u(i,j) + u(i,j-1).$$

That yields in x-direction the 1×3 mask

$$u_{xx} = \begin{pmatrix} +1 & -2 & +1 \end{pmatrix}$$

and in y-direction the 3×1 mask

$$u_{yy} = \begin{pmatrix} +1 \\ -2 \\ +1 \end{pmatrix},$$

which will give a matrix containing the approximative second derivatives by convolving with the original image $I(i,j)$. The discretized version of the Laplacian operator (=Laplacian filter) can be found by

$$L(i,j) * I = (u_{xx} + u_{yy}) * I = \left(\begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix} \right) * I = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} * I.$$

The set of edges is given by the zero-crossings. Note that since the Laplacian operator is the sum of second-order derivatives the method is very sensitive to noise.

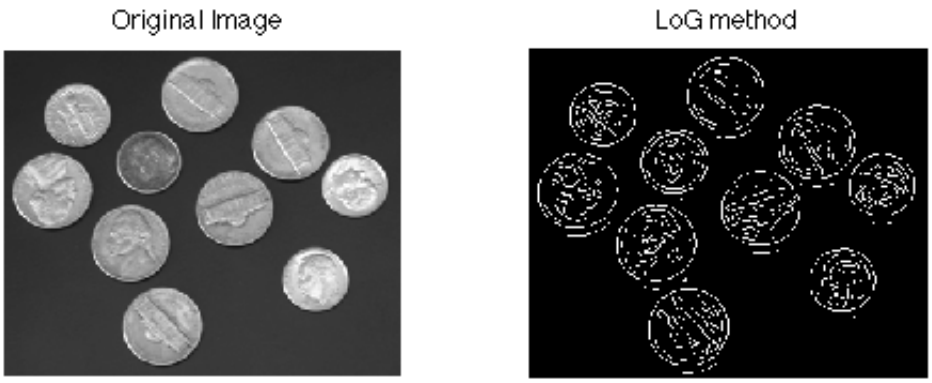


Figure 2.3: Using the matlab LoG method on a greyscale image.

Marr-Hildreth-Operator or Laplacian of Gaussian (LoG)

The LoG Filter first smooths a continuous image with a Gaussian filter $G(x,y)$ and then finds the zero-crossings in the Laplacian of the smoothed image. It uses the kernel

$$\Delta G(x,y) * I(x,y) =: LoG(x,y) * I(x,y).$$

The derivative of the Gaussian filter is independent of the image and therefore can be precomputed. The standard Gaussian smoothing operator is given by the continuous kernel

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}},$$

where x,y denote the image coordinates and σ is a standard deviation of the associated probability distribution. The Laplacian of the Gaussian is given by

$$\Delta G(x,y) = \frac{\delta^2}{\delta x^2}G(x,y) + \frac{\delta^2}{\delta y^2}G(x,y) = (\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})e^{\frac{-(x^2+y^2)}{2\sigma^2}} + (\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2})e^{-\frac{(x^2+y^2)}{2\sigma^2}},$$

which leads to

$$LoG(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-\frac{(x^2+y^2)}{2\sigma^2}}.$$

In [Gun02] one can find ideas how to get, for a given discrete image I, an appropriate discrete convolution mask from the equation above. After convolving the image with the kernel $LoG(x,y)$ the algorithm finds the set of edge pixels by looking for the zero - crossing pixels, i.e. the pixels that cross the zero level.

The advantage of this approach compared to classical edge operators of small size is that a larger area surrounding the current pixel can be taken into account. The influence of more distant points decreases according to the σ of the Gaussian filter, but the σ variation does not affect the location of the zero-crossings. Note that the operator suffers from two main disadvantages: it generates responses that do not correspond to edges and the localization error may be severe at curved edges [RKS13].

More detailed information about Laplacian operators (and especially the LoG operator) can be found in [RKS13], [MS98] and [Wee96].

Comparison of the methods

As stated in [RKS13], in comparison of all these edge detection techniques it is found that the Canny method gives the optimum edge method. Some points of advantages compared to the discussed methods in that section are:

- It is less sensitive to noise as compared to Prewitt, Sobel and Roberts edge detector, because it uses the Gaussian filter before taking the gradient image. The LoG operator is also highly sensitive to noise as it works with the second derivative/Laplacian.
- The classical operators have fixed kernels which do not change for different images. The Canny algorithm depends on the standard deviation σ and two thresholds t_{low} and t_{high} which allow to improve the performance by changing and adapting them.

- Most methods use a single threshold limit, which means if the edge values fluctuate above and below this value, the line will appear broken (streaking). The Canny method is avoiding that by doing double thresholding (described in subsection 2.2.1).

2.2.2 Thresholding

Edge detection methods are based on partitioning an image based on strong changes of intensity. Other methods, including thresholding, partition the image into regions that satisfy some predefined criteria. In the case of thresholding the pixels are partitioned depending on their intensity value. For a discrete greyscale image the result is usually a binary image (i.e. consisting of colorvalues 0 and 1). There are different possibilities of dividing the image:

- **Global thresholding** with an a priori chosen threshold T

$$v(i,j) = \begin{cases} 1 & \text{if} \quad u(i,j) > T, \\ 0 & \text{if} \quad u(i,j) \leq T, \end{cases}$$

  where $v(i,j)$ denotes the label function of the binary result image.

- **Variable thresholding**
  Local or regional thresholding: T depends on a $d \times d$ neighborhood of the pixel p_{ij},
  Adaptive thresholding: T is a function of the coordinates (i,j).

- **Multiple thresholding** with more than one a priori chosen thresholds, e.g.

$$v(i,j) = \begin{cases} a & \text{if} \quad u(i,j) > T_2, \\ b & \text{if} \quad T_1 < u(i,j) \leq T_2, \\ c & \text{if} \quad u(i,j) \leq T_1, \end{cases}$$

which returns the label function of an image with 3 regions.

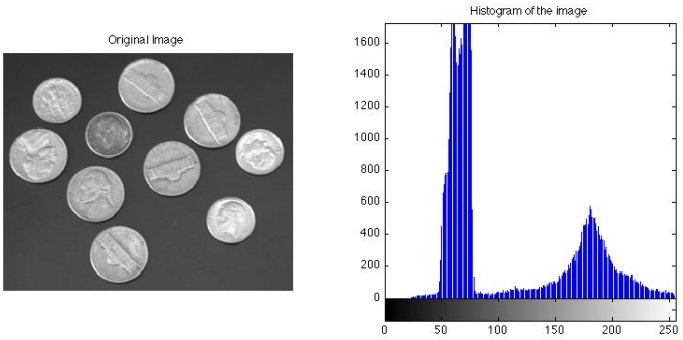


Figure 2.4: Image histogram that shows the amount of the different pixel intensities in the image.

Choosing the threshold in global thresholding

There are different ways of finding an appropriate threshold. A common way is to analyze the image histogram [SaKK10]. An image histogram shows the amount of pixels of each color intensity (usually 0 to 255). The ideal case is given when the histogram presents two dominant modes and a clear valley (bimodal). In that case the threshold T is selected as the valley point between the two modes. In practice histograms are unfortunately often more complex with many peaks and no clear valley. One simple technique to find an appropriate threshold uses the mean value of all pixels. This technique has the main disadvantage that it considers that approximately half of all pixels belong to the objects and the other half to the background. In [SaKK10] and [NS93] other ideas on finding appropriate thresholds can be found.

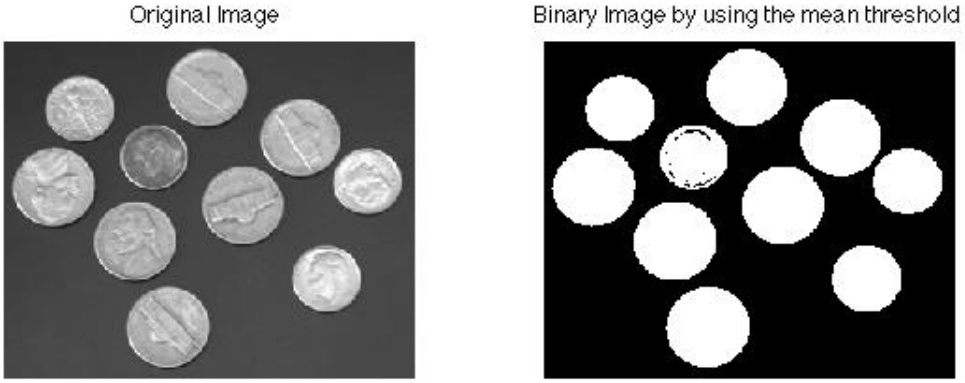


Figure 2.5: Binary image by using the mean as global threshold.

2.2.3 Active Contours

Using active contour models (also called *snakes*) is another edge detection method with a completely different approach as in Subsection 2.2.1. It is evolved on images on continuous regions. Starting with an initial contour around an object in the image, the curve is deformed towards the boundary of that object. The deformation is obtained by minimizing an appropriate energy functional, i.e. its minimum should be the approximate curve at the boundary of the located object.

The classical model introduced by Kass et al. (1988) in [KWT88] wants to find the curve C that minimizes the energy

$$E(C) = \alpha \int_0^1 |C'(q)|^2 dq + \beta \int_0^1 |C''(q)|^2 dq - \lambda \int_0^1 |\nabla z(C(q))| dq, \tag{2.1}$$

where $C(q) : [0,1] \to \mathbb{R}^2$ denotes a parametrized curve, $z : [0,a] \times [0,b] \to \mathbb{R}^+$ the original given continuous image and α, β and λ are real positive constants. The first two addends control the smoothness of the contours (*internal energy*) and the third term attracts the contour

towards the object (*external energy*). But as stated in [CKG97] the energy model (2.1) is not suitable to handle changes in the topology of the evolving contour, i.e. the final curve will be as the initial one, which can be a problem if an unknown number of objects should be detected simultaneously. The solution without some additional procedures will be in most cases a curve which approaches a convex hull type figure of the objects in the image. In [CKG97] a similar energy model is introduced, which shall overcome those problems:

$$E(C) = \alpha \int_0^1 |C'(q)|dq - \lambda \int_0^1 |\nabla z(C(q))|dq. \tag{2.2}$$

Compared to (2.1) it is the same model with chosen parameter $\beta = 0$. By minimizing the energy (2.2) it is wanted to locate the curve at the points of maxima $|\nabla z|$ (edge detection) while keeping smoothness in that curve. Equation (2.2) can be extended by generalizing the edge detection part: Let $g : [0, +\infty[\to \mathbb{R}^+$ be a strictly decreasing function such that $g(r) \to 0$ as $r \to \infty$. Now we can replace $-|\nabla z|$ by $g(|\nabla z|)$ obtaining a more general energy functional given by

$$\begin{aligned} E(C) &= \alpha \int_0^1 |C'(q)|dq + \lambda \int_0^1 g\big(|\nabla z(C(q))|\big)^2 dq \\ &= \int_0^1 E_{int}(C(q)) + E_{ext}(C(q)) \;\; dq. \end{aligned} \tag{2.3}$$

The following results for minimizing (2.2) are shown in [CKG97]:

1. It is first proven that finding a minimum for (2.2) is equivalent to finding a geodesic curve in a Riemannian space with a metric derived from the image content. It describes the connection between energy and curve evolution approaches of active contours. Recall that a geodesic curve in a Riemannian manifold is a (local) minimal distance path between given points.

2. The curve evolution flow that minimizes the weighted length (which is derived from (2.2)) is then given by
$$\frac{\partial C(t)}{\partial t} = g(z)\kappa \vec{N} - (\nabla g \cdot \vec{N})\vec{N}, \tag{2.4}$$
where κ denotes the Euclidean curvature and $\vec{N}$ the unit inward normal.

3. Now this flow is embedded in a level-set formulation in order to complete the model and show its connection with previous curve evolution active contours. This geodesic flow includes the new component *velocity* in the curve that improves the model, because it allows to accurately track boundaries with high variation in their gradient.

4. Furthermore it is shown that the solution of the geodesic flow exists in the viscosity framework - it is unique, stable and consistent.

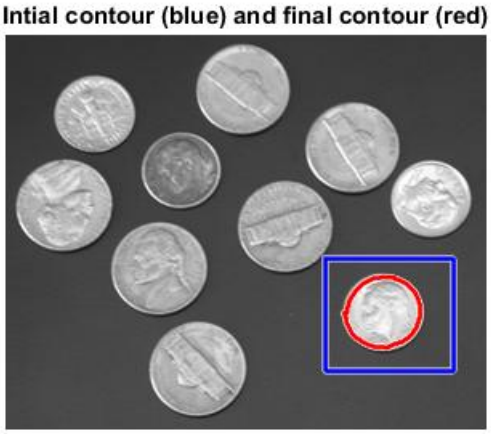


Figure 2.6: Using the activecontour function that is included in the Image Processing Toolbox of Matlab R2015 on a discrete image with an initial rectangle contour.

As stated in [CKG97] the main advantages of the approach are:

- The approach allows simultaneous detection of interior and exterior boundaries in several objects without special contour tracking procedures.
- There exist formal existence, uniqueness, stability and consistency results.
- It does not require special stopping conditions.

Remark. Other ideas for improving the energy model (2.1) can be found in e.g. [MT95] (topologically adaptable snake), [CKG97] (new model for active contous) and [MSV94] (shape modeling and recovery).

Remark. Note that in the previous subsection we derived the model and solutions for continuous images. To use it on digital discrete images either the idea has to be adapted for discrete images or one requires to receive a continuous image by interpolating the stored discrete image samples. Under some circumstances it is possible that a discrete set of samples can accurately represent a continuous image (*sampling theorem*). For more information on that see e.g. [Rus11].

2.2.4 Mumford - Shah functional

David Mumford and Jayant Shah formulated in [MS85] and [MS89] an energy minimization problem which yields an optimal piecewise-smooth or piecewise-constant approximation of a given continuous initial image. The idea is smiliar to *active contours* described in the subsection before. Note that the Mumford - Shah model was already proposed in 1985. Since then

plenty of papers were published in which the model was studied and analyzed to get properties of the minimizer, find numerical approximations and discuss its applications in image segmentation. A lot of different algorithms to find the minimizer of the functional have been developed aswell. E.g. in [BCC+11] some different approaches and formulations, properties and numerical algorithms are discussed.

In the original paper of Mumford and Shah ([MS89]) three variational problems are discussed. We will state now the first one on piecewise smooth functions, which is known in the literature as the *Mumford-Shah functional*. As described by Mumford and Shah it consists in computing a decomposition of the domain Ω of the initial image z such that

$$\Omega = \Omega_1 \cup \cdots \cup \Omega_n$$

and

- the image shall vary smoothly and/or slowly within each Ω_i
- the image shall vary discontinuously and/or rapidly across most of the boundary Γ between different Ω_i.

As stated in [MS89] that is the same as computing optimal approximations of a general initial function $z(x)$ (the initial image) by piecewise smooth functions $u(x)$, i.e. functions u whose restrictions u_i to the subregions Ω_i are differentiable.

Definition of the minimization problem as stated in [MS89]

Let the regions Ω_i be disjoint connected open subsets of a planar image domain Ω, each one with a piecewise smooth boundary and let Γ be the union of the part of the boundaries of the subregions Ω_i inside Ω so that

$$\Omega = \Omega_1 \cup \cdots \cup \Omega_n \cup \Gamma.$$

Let z be a differentiable function on $\cup\Omega_i$ that is allowed to be discontinuous across Γ. Then the functional is given by

$$E(u,\Gamma) = \mu^2 \int_\Omega (u-z)^2 dx + \int_{\Omega-\Gamma} \|\nabla u\|^2 dx \; + \; \nu|\Gamma|, \tag{2.5}$$

where μ and ν are positive parameters and $|\Gamma|$ stands for the total length of the boundaries.

By minimizing the functional above we get that

- the first addend penalizes the distance between the new image u and the initial image z,
- the second addend regulates that u is smooth on each Ω_i and
- the third term controls that the boundaries of the subregions are as short as possible.

Because of those reasons it holds that the smaller the functional (2.5) gets, the better (u,Γ) segments the initial image z. Therefore it makes sense to look for the minimizer of that energy functional to get a segmentation of the original image. Note that at that stage it remains to show that there exists a minimizer and how it would be possible to calculate or approximate it.

As stated in [MS89] the minimizer (u,Γ) would be a cartoon of the actual image z, where u may be taken as a new image in which the edges are drawn sharply and precisely, the objects sourrounded by the edges are drawn smoothly without texture.

Remark. Note that if we restrict $E(u,\Gamma)$ to piecewise constant instead of piecewise smooth functions (i.e. $u = \text{const}_i$ on each open set Ω_i), the second term in (2.5) will vanish since the derivative of constants equals zero. It is also called *Potts model* in the literature (see e.g. [SW14]) and will be discussed later.

First variation approach by Mumford and Shah

Since the functional (2.5) is non-convex it is difficult to solve the minimization problem or to find smooth approximations.

In [MS89] it is shown by analyzing the first variation of the functional that if the set of boundaries Γ **is fixed**, the functional (2.5) is a positive definite quadratic function in u with a **unique minimum**! In that case the minimum will be the function u that solves the elliptic boundary value problem

$$\begin{aligned} \Delta u &= \mu^2(u-z) \\ \left.\frac{\partial u}{\partial n}\right|_{\partial\Omega_i} &= 0 \end{aligned} \tag{2.6}$$

on each subregion Ω_i , where $\partial\Omega_i$ denotes the boundary of Ω_i and $\frac{\partial}{\partial n}$ a unit normal vector to $\partial\Omega_i$. The second condition tells that $\frac{\partial u}{\partial n}$ is zero on both sides of Γ and on the inside of $\partial\Omega$ (the boundary of the whole domain Ω).

Moreover we want to calculate the first variation of $E(u,\Gamma)$ with respect to Γ. If we assume that u_Γ is the solution of (2.6), E reduces to a function of Γ alone. It is shown that if $E(u_\Gamma,\Gamma)$ is minimized at Γ, Γ must satisfy the variational equation

$$e(u^+) - e(u^-) + \nu \cdot curve(\Gamma) = 0,$$

where $e(u;x) = \mu^2(u(x)-z(x))^2 + \|\nabla u(x)\|^2$. In [MS89] more theoretical results of existence and regularity of the minimizers can be found. Moreover in the book [MS95] by J.M. Morel and S. Solimini three detailed chapters about existence and structural properties of the minimal segmentations for the Mumford-Shah model can be found.

Further published approaches

Since the publishing of the Mumford - Shah functional in 1985 several approaches of approximations and reformulations have been proposed: e.g. approximation by elliptic functionals

defined on Sobolev spaces by L. Ambrosio and V.M. Tortorelli [AT90]; finite-differences discretization of the Mumford-Shah functional by A. Chambolle [Cha99]; convex approximation by X. Cai, R. Chan and T.Zeng in [CCZ]; level set approximations by T. Chan and L. Vese [VC02]; ...

Level set formulation proposed by L.Vese and T.Chan

In 1999 L. Vese and T. Chan formulated in their paper *An Active Contour Model without Edges* [CV99] a new model for active contours to detect objects in a given image. It uses techniques of curve evolution, the Mumford-Shah functional restricted to piecewise constant functions and level set formulation. In [CV99] the model is restricted to greyscale images and two-phase segmentation (i.e. the calculated image consists only of two constant colors - inside and outside the object), whereas in [CSV99] it is generalized to vector-valued images and in [CV00] Chan and Vese propose a multiphase level set model that divides the image in n piecewise constant regions. In [CV01] they generalize their previous work to piecewise smooth functions and describe how the piecewise-smooth Mumford-Shah functional can be solved by a level set algorithm based on the level set method of Osher and Sethian ([OS88]). I am going to describe their basic ideas of two-phase and multiphase segmentation formulated in the papers [CV99] and [CV00].

Two - phase Level Set Model
The active contour model in [CV99] is given by

$$\begin{aligned} E(\Gamma,c_1,c_2) = {} & \mu\cdot length(\Gamma)+\nu\cdot area(inside\Gamma) \\ & +\lambda_1\int_{inside(\Gamma)}|z-c_1|^2dx+\lambda_2\int_{outside(\Gamma)}|z-c_2|^2dx, \end{aligned}$$

where z is the initial continuous image, Γ the unknown set of edges, c_1 and c_2 are constant unknowns (the two colors of the resulting image) and $\mu>0, \nu\geq 0$, $\lambda_1,\lambda_2>0$ are fixed parameters. As stated in [CV99], in almost all computations it lasts to choose the parameters as $\nu=0$ and $\lambda_1=\lambda_2$. Therefore we will work with the functional

$$E(\Gamma,c_1,c_2)=\mu\cdot length(\Gamma)+\lambda\left(\int_{inside(\Gamma)}|z-c_1|^2dx+\int_{outside(\Gamma)}|z-c_2|^2dx\right). \qquad (2.7)$$

That model is a special case of the Mumford-Shah model (2.5) restricted to a two - phase piecewise constant function u, taking the two unknown values $c_1=average(z)$ inside Γ and $c_2=average(z)$ outside Γ.

Based on the level set formulation in [OS88], an evolving curve Γ can be represented by the zero level set of an appropriate Lipschitz continuous function $\phi:\Omega\to\mathbb{R}$, i.e. $\Gamma=\{x\in\Omega : \phi(x)=0\}$. We assume here that the unknown set of edges Γ in the image can be represented by such a zero level set and we set $\phi(x)>0$ for x inside Γ and $\phi(x)<0$ for x outside Γ. Therefore the functional (2.7) becomes

$$E(\phi,c_1,c_2)=\mu\cdot length\{\phi=0\}+\lambda\left(\int_{\phi\geq 0}|z-c_1|^2dx+\int_{\phi<0}|z-c_2|^2dx\right).$$

Using the one-dimensional Heaviside function H defined by

$$H(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0, \end{cases}$$

we can express (as stated in [CV99]) the perimeter surface area (length) by

$$length\{x \in \Omega : \phi(x) = 0\} = \int_\Omega |\nabla H(\phi)| dx$$

and the area of the region $\{x \in \Omega : \phi(x) \geq 0\}$ by

$$area\{x \in \Omega : \phi(x) \geq 0\} = \int_\Omega |H(\phi(x))| dx.$$

Therefore it holds that

$$\begin{aligned} \int_{\phi \geq 0} |z - c_1|^2 dxdy &= \int_\Omega |z - c_1|^2 H(\phi) dx, \\ \int_{\phi < 0} |z - c_2|^2 dxdy &= \int_\Omega |z - c_2|^2 (1 - H(\phi)) dx. \end{aligned} \tag{2.8}$$

The functional (2.7) can now be written as its level set formulation

$$E(\phi, c_1, c_2) = \mu \int_\Omega |\nabla H(\phi(x))| + \lambda \int_\Omega |z - c_1|^2 H(\phi(x)) + |z - c_2|^2 (1 - H(\phi(x))) \, dx.$$

Keeping ϕ fixed and minimizing the energy $E(\phi, c_1, c_2)$ with respect to the constants c_1 and c_2, one can express (as stated in [CV99]) the constant functions by

$$\begin{aligned} c_1(\phi) &= \frac{\int_\Omega z(x) \, H(\phi(x)) \, dx}{\int_\Omega H(\phi(x)) \, dx} \qquad \text{(the average of } z(x) \text{ in } \{\phi \geq 0\}), \\ c_2(\phi) &= \frac{\int_\Omega z(x) \, (1 - H(\phi(x))) \, dx}{\int_\Omega 1 - H(\phi(x)) \, dx} \quad \text{(the average of } z(x) \text{ in } \{\phi < 0\}). \end{aligned} \tag{2.9}$$

Now keeping c_1 and c_2 fixed, by minimizing the energy $E(\phi, c_1, c_2)$ with respect to ϕ, it is possible to calculate the Euler - Lagrange equation for ϕ, which is given by

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left[\mu \, \text{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda \left((z(x) - c_1)^2 + (z(x) - c_2)^2 \right) \right] \quad \text{in } \Omega$$

and

$$\frac{\delta(\phi)}{|\nabla \phi|} \frac{\partial \phi}{\partial n} = 0 \quad \text{on } \delta\Omega,$$

where $\delta(x)$ denotes the one-dimensional Dirac measure concentrated at 0 $\left(\delta(x) := \frac{d}{dx} H(x)\right)$. As stated in [CV99] for the calculations we have to consider a slightly regularized version of the functions H and δ. Further information on that can be found in [CV99] and [BCC+11].

Multiphase Level Set Model

In [CV00] they generalize the idea in the sense that the minimization problem is now restricted to piecewise constant functions u taking $(n+1)$ unknown constant values $c_0,\dots,c_n$. They present the idea and solution for greyscale images (i.e. $z:\Omega\subset\mathbb{R}^2\to\mathbb{R}$), again based on level set formulation. Using n disctinct levels $\{l_1<l_2<\cdots<l_n\}$, with $l_i\in\mathbb{R}$, the level set function $\phi:\Omega\to\mathbb{R}$ partitions the domain Ω into $n+1$ disjoint open regions

$$\begin{aligned}\Omega_0&=\{x\in\Omega:-\infty<\phi(x)<l_1\},\\ \Omega_j&=\{x\in\Omega:l_j<\phi(x)<l_{j+1}\},\quad 1\le j\le n-1\\ \Omega_n&=\{x\in\Omega:l_n<\phi(x)<+\infty\}.\end{aligned}$$

As stated in [BCC$^+$11], it holds for a level parameter $l\in\mathbb{R}$ that

$$length\{x\in\Omega:\phi(x)\ge l\}=\int_\Omega|\nabla H(\phi(x)-l)|dx$$

and

$$area\{x\in\Omega:\phi(x)\ge l\}=\int_\Omega H(\phi(x)-l)dx.$$

Therefore the energy is now given by

$$\begin{aligned}E(c_0,c_1,\dots,c_n,\phi)=\mu\sum_{j=1}^{n}\int_\Omega|\nabla H(\phi(x)-l_j)|+\int_\Omega|g(x)-c_0|^2H(l_1-\phi(x))dx\\ +\sum_{j=1}^{n-1}\int_\Omega|g(x)-c_j|^2H(\phi(x)-l_j)H(l_{j+1}-\phi(x))dx+\int_\Omega|g(x)-c_n|^2H(\phi(x)-l_n)dx.\end{aligned}$$

Again it is possible to calculate the Euler - Lagrange equations associated with the minimization of $E(c_0,c_1,\dots,c_n,\phi)$. They can be found in [BCC$^+$11], where also a numerical algorithm (using gradient descent) is presented for solving those equations.

Visualization

Using a matlab code by Yue Wu (http://www.mathworks.com/matlabcentral/fileexchange/23445-chan-vese-active-contours-without-edges) who implemented the papers [CV99], [CSV99] and [CV00] of Chan and Vese yields figure 2.7, 2.8 and 2.9.

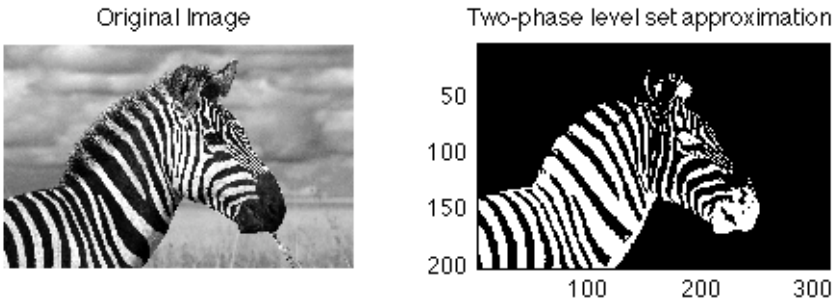


Figure 2.7: Using the matlab code by Yue Wu, which is based on the papers of Chan - Vese.

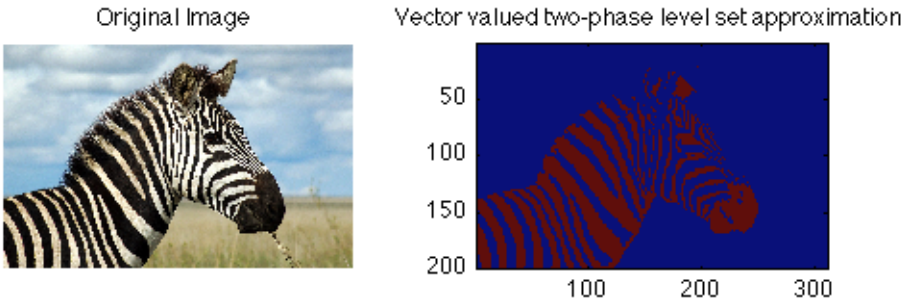


Figure 2.8: Using the code for two - phase level set approximation for vector - valued images.

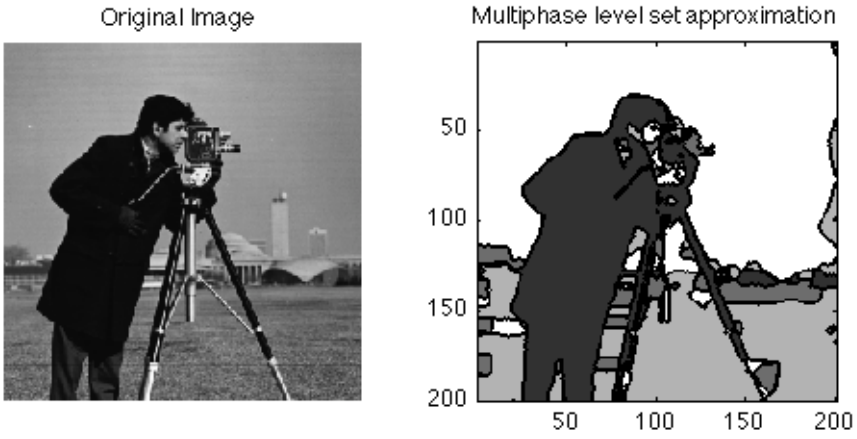


Figure 2.9: Using the matlab code for multiphase level set approximation by Yue Wu.

2.2.5 Potts model

For some initial image $z : \Omega \to \mathbb{R}^r$ on a continuous region $\Omega \subset \mathbb{R}^2$ and $\gamma > 0$ the minimization problem of the *Potts model* is formulated in [SW14] as

$$u^* = \operatorname*{argmin}_u E(u) \; = \operatorname*{argmin}_u \gamma \cdot \|\nabla u\|_0 + \int_\Omega (u-z)^2 dx, \tag{2.10}$$

where $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^r$ denotes a piecewise constant function and $\|\nabla u\|_0$ the total boundary length of its partitioning. The energy in (2.10) is basically the piecewise constant energy model proposed by Mumford and Shah in [MS89] (compare it with the functional stated in (2.5)). It is named after R. Potts, who introduced in 1952 (see [Pot52]) the jump penalty in a fully discrete setting in the context of his work on statistical mechanics generalizing the Ising model.

The second term in (2.10) provides approximation to the original data and the first term takes care of the regularity of the partitioning. The parameter γ controls the balance between the two penalties.

In [SW14] a fast strategy for solving the Potts problem for vector-valued images is presented. For a given discrete $n \times m$ image z with values in $\mathbb{R}^s$ they consider a discrete domain version of (2.10) by

$$u^* = \operatorname*{argmin}_{u \in \mathbb{R}^{m \times n \times s}} \gamma \cdot \sum_{i,j} \sum_{(a,b) \in N} \omega_{ab} \cdot [u_{i,j,:} \neq u_{i+a,j+b,:}] \; + \sum_{i,j,k} |u_{ijk} - z_{ijk}|^2, \tag{2.11}$$

where $u_{i,j,:}$ denotes the label vector of u at the pixel (i, j), N denotes a neighborhood system with nonnegative weights ω and the Iverson bracket $[\cdot]$ equals one if the expression in the bracket is true and zero otherwise.

In a simple case one may use the coordinate unit vectors as neighborhood and set ω uniformly equal to 1. That corresponds to the jump penalty $\|\nabla_1 u\|_0 + \|\nabla_2 u\|_0$ which counts the nonzero entries of the directional difference operators. Applied to a scalar image z in $\mathbb{R}^{m \times n}$ that yields the problem

$$u^* = \operatorname*{argmin}_{u \in \mathbb{R}^{m \times n}} \gamma \cdot \sum_{i,j} [u_{i,j} \neq u_{i+1,j}] + [u_{i,j} \neq u_{i,j+1}] + \sum_{i.j} |u_{ij} - z_{ij}|^2. \tag{2.12}$$

Minimization strategies

The continuous minimization problem (2.10) is nonconvex and as shown in [BVZ14] the discrete problem (2.11) is NP-hard, which means that finding a global minimizer is at the moment a computationally intractable task.

Since nevertheless the problem is important in image processing lots of efficient approximative strategies have been developed. We already discussed in Subsection 2.2.4 the level set method of Chan and Vese. Another approach is the minimal graph cut algorithm (which will be described in Chapter 3) developed by Boykov, Veksler and Zabih (see [BVZ14]). P. Felzenszwaln and D. Huttenlocher combine in their paper *Efficient Belief Propagation for Early Vision* the ideas and advantages of graph cuts and belief propagation to a faster algorithm. In

[HS08] H. Hirschmueller proposes a noniterative strategy called cost aggregation. The advantage of that algorithm is the lower computational cost but it comes with lower quality results compared to graphcuts. There do exist already several more approaches for efficient and fast approximation algorithms, a short overview can be found in [SW14].

In Chapter 4 and 5 I am going to describe and use the algorithm proposed in [SW14]. Their idea is to reformulate the problem (2.11) to a suitable constrained optimization problem so that it is possible to apply the ADMM (described in Chapter 4) to obtain computationally tractable subproblems (called univariate Potts problems). Then they propose an accelerated dynamic programming strategy so that those subproblems can be solved quickly and exactly.

They claim in [SW14] that the main feature of that algorithm is its efficiency with respect to runtime and memory consumption. Compared with e.g. the graph cuts method in higher-dimensional codomains (i.e. vector - valued images) the computational cost is already significantly lower for rgb images. The cost grows linearly and not exponentially with the number of channels (i.e. the dimension of the codomain). Because of that it is possible to process multispectral images with even more than 30 channels in a reasonable time. Moreover the algorithm needs on their experiments only a few iterations to converge and due the efficient dynamic program the most time consuming parts of the algorithm can be solved pretty fast. Moreover it is shown that the proposed algorithm converges, although due the NP - hardness of the problem it is not possible to show that the limit point is in general a minimizer of the functional (2.11).

3 Energy Minimization via Graph Cuts

A common combinatorical way for minimizing energy functionals in image processing (e.g. functionals as stated in the Sections 2.2.3, 2.2.4 and 2.2.5) are algorithms that use the GRAPH CUTS method. I will now give an overview of the ideas behind that widespread method.

3.1 Appropriate energy functionals

In [BVZ14] we can find a detailed description of two different algorithms (the optimal swap move and the optimal expansion move) using Graph Cuts to minimize an energy function of the form

$$E(u) = E_{\text{smooth}}(u) + E_{\text{data}}(u), \tag{3.1}$$

where $u : \Omega \to \mathbb{R}^s$ denotes a label function of a digital image, that is given by pixels p on the discrete image area $\Omega = \{0,1,\dots,m-1\} \times \{0,1,\dots,n-1\}$. The function u assigns each pixel p with a label $u_p := u(i,j)$, where (i,j) denotes the location of the pixel p in Ω. Furthermore let P denote the set of all pixels p of the discrete domain Ω. Typically $E_{data}(u)$ is given by

$$E_{data}(u) = \sum_{p \in P} D_p(u_p), \tag{3.2}$$

where D_p measures the disagreement between the new labeling u_p and the observed data z_p (original labeling of the pixel p). The addend $E_{\text{smooth}}(u)$ is supposed to measure somehow the smoothness of u and and should have the form

$$E_{smooth}(u) = \sum_{p,q \in N} V_{p,q}(u_p, u_q), \tag{3.3}$$

where N denotes the set of interacting pairs of pixels and $V_{p,q}$ the interacting penalty between the pixels p and q.

In general finding the optimal point of such an energy function is a NP-hard optimization problem. For the cases that $V_{p,q}$ is given by a metric or a semi-metric, two efficient algorithms that approximately minimize (3.1) are developed in [BVZ14]. In practice these methods run in near-linear time.

3.1.1 The Potts model

There exist several appropriate energy functions in the form of (3.1), one of them is the (described in 2.2.5) two dimensional discretized Potts model

$$u^* = \underset{u\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \gamma \cdot \sum_{i,j} [u_{i,j} \neq u_{i+1,j}] + [u_{i,j} \neq u_{i,j+1}] + \sum_{i.j} |u_{ij} - z_{ij}|^2.$$

In the notation of (3.2) and (3.3) the terms V_p and D_p are now given by the metrices

$$D_p(u_p) = |u_p - z_p|^2,$$

and

$$V_p(u_p) = \gamma \cdot \sum_{q\in N_p} [u_p \neq u_q],$$

where N_p denotes the pixel neighbors north and east of p.

In order to understand the ideas behind the Graph Cuts method I will state in 3.2 some basic definitions and theorems of graph theory.

3.2 Some basic concepts of graph theory

Definition 1 (Directed Graph)**.** A directed graph (or digraph) is a set of nodes connected by edges, where the edges have a direction associated with them. In formal terms, a digraph is a pair $G = (V,E)$ of

1. a set V whose elements are called vertices or nodes,
2. a set E of ordered pairs of vertices called arcs, (directed) edges or arrows.

It differs from an undirected graph, where the set E is defined in terms of unordered pairs of vertices, which are usually called edges.

Definition 2 (Flow network)**.** Let $G = (V,E)$ be a finite directed graph in which every edge $(u,v) \in E$ has a non-negative, real-valued capacity $c(u,v)$. It represents the maximum amount of flow that can pass through an edge. If $(u,v) \notin E$, we assume that $c(u,v) = 0$. We distinguish two vertices: a source s and a sink t.

A flow in a flow network is a real function $f : V \times V \to \mathbb{R}$ with the following three properties for all nodes u and v:

CAPACITY CONSTRAINTS

$$f(u,v) \leq c(u,v),$$

i.e. the flow along an edge cannot exceed its capacity.

SKEW SYMMETRY

$$f(u,v) = -f(v,u),$$

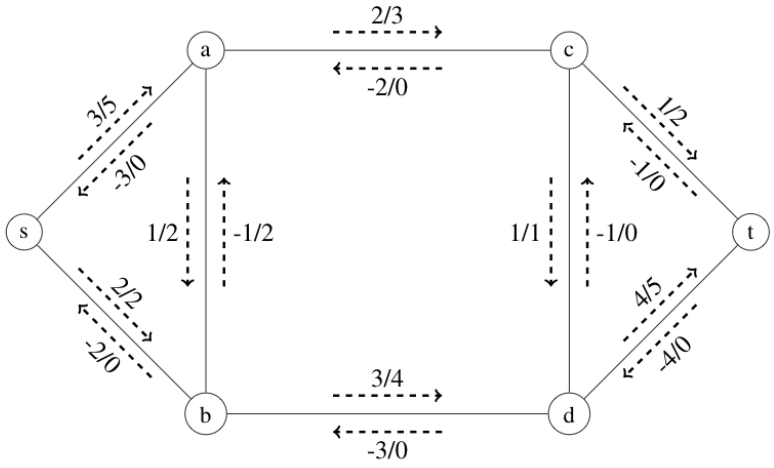


Figure 3.1: A flow network with flow (left) and capacity (right)

i.e. the net flow from u to v must be the opposite of the net flow from v to u.

FLOW CONSERVATION

$$\sum_{w \in V} f(u,w) = 0 \text{ unless } u = s \text{ or } u = t,$$

i.e. the net flow to a node is zero, except for the source, which produces flow, and the sink, which consumes flow. Therefore the flow conservation implies

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(v,z) \in E} f(v,z) \text{ for each vertex } v \in V \setminus \{s,t\}.$$

Figure 3.1 is an example of a flow network showing flow (left) and the capacity (right) of each edge.

3.2.1 Max-Flow Min-Cut Theorem

Let $N = (V,E)$ be a flow network (directed graph) with s and t being the source and the sink of N.

Definition 3 (Value of flow)**.** The value of flow is defined by

$$|f| = \sum_{v \in V} f(s,v),$$

where s is the source of N. It represents the amount of flow passing from the source to the sink.

The MAXIMUM FLOW PROBLEM is to maximize $|f|$, i.e. to route as much flow as possible from the source s to the sink t.

Definition 4 (S-T Cut)**.** An S-T cut $C = (S,T)$ is a partition of the set V of vertices such that $s \in S$ and $t \in T$, i.e. the sink and the source, get separated. The cut-set of C is the set

$$\{(u,v) \in E \ : \ u \in S, v \in T\},$$

those edges that get removed and separate the graph. Note that if the edges in the cut-set of C are removed, the value of flow equals zero ($|f| = 0$).

Definition 5 (Capacity of an S-T cut)**.** The capacity of an S-T cut is defined by

$$c(S,T) = \sum_{(u,v) \in S \times T} c(u,v).$$

The MINIMUM CUT PROBLEM is the problem of minimizing $c(S,T)$ over all S and T, i.e. to determine S and T such that the capacity of the S-T cut is minimal.

Theorem 3.2.1 (Max-Flow Min-Cut Theorem)**.** *The value of the maximum flow is equal to the capacity of the minimum cut.*

Remark. The theorem was proven by different people independently in 1956. One proof is by L.R. Ford and D.R.Fulkerson and can be found in [FD56] .

3.2.2 Application to the optimization problem

Recall that the optimization problem we want to solve is given by

$$u^* = \operatorname{argmin}_u E(u) = \operatorname{argmin}_u E_{\text{smooth}}(u) + E_{\text{data}}(u).$$

The main idea is to construct an appropriate directed graph such that the minimizer can be found by finding the minimum cut over that graph. As stated in the duality Theorem 3.2.1 finding the minimum cut is equivalent to finding the maximum flow. Therefore the max-flow problem and the min-cut problem can be formulated as two primal-dual linear programs. There are several fast and efficient algorithms for solving the max-flow problem. A comparison of some of them can be found in [BK04].

Now it just remains to find an appropriate graph for our energy functional (3.1). As mentioned before we can find in [BVZ14] two appropriate graphs for solving the optimization problem. In the following I will describe those two without the detailed proofs, which can be found in [BVZ14].

3.3 Two different approaches

3.3.1 The optimal α - β swap move

Let P be the set of all pixels of an image, $P_l = \{p \in P \mid u_p = l\}$ be the subset of the pixels which are assigned with the same label l (e.g. all pixels with the same color intensity $l \in \{0,\dots,255\}$)

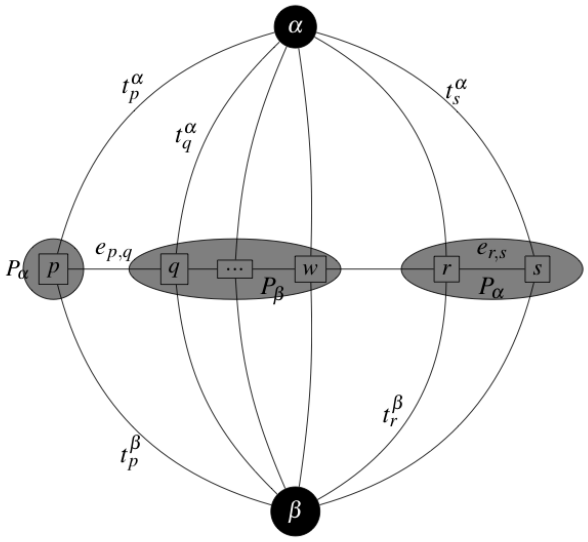


Figure 3.2: Example of the graph $G_{\alpha\beta}$ for a 1D image

and L the set of all different labels l. Given a pair of labels α and β, a move from a labeling u to a new labeling u' is called an $\alpha-\beta$ swap if $P_l = P'_l$ for any label $l \neq \alpha, \beta$. Therefore the only difference between u and u' is that some pixels that were labeled α in u are now labeled β in u' and the other way round.

Given an input labeling u and a pair of labels α and β we want to find a labeling u^* that minimizes the energy function $E(u)$ over all labelings within one α - β swap of u.

The structure of the appropriate graph $G_{\alpha\beta} = (V_{\alpha\beta}, E_{\alpha\beta})$ will be as follows. The set of vertices includes two terminals (i.e. the source and the sink) α, β and image pixels p which are in the sets $\mathscr{P}_\alpha$ and $\mathscr{P}_\beta$. The set of vertices $V_{\alpha\beta}$ therefore consists of α, β and $\mathscr{P}_{\alpha\beta} = \mathscr{P}_\alpha \cup \mathscr{P}_\beta$. Each pixel $p \in \mathscr{P}_{\alpha\beta}$ is connected to the terminals α and β by edges t_p^α and t_p^β, called t-links. Each pair of pixels $p, q \subset \mathscr{P}_{\alpha\beta}$ that are neighbors is connected by an edge $e_{p,q}$ called n-link (neighbor link). Therefore the set of edges $E_{\alpha\beta}$ consists of the t-links $\bigcup_{p\in\mathscr{P}_{\alpha\beta}}(t_p^\alpha, t_p^\beta)$ and the n-links $\bigcup_{\substack{p,q\in\mathscr{P}_{\alpha\beta} \\ p,q \text{ neighbors}}}(e_{p,q})$.

Figure 3.2 shows an example of such a graph for an 1D image.

The weights assigned to the edges are given by
$$\begin{cases} D_p(\alpha) + \sum\limits_{q\notin\mathscr{P}_{\alpha\beta}\wedge \text{ neighbor of} p} V_{p,q}(\alpha, u_q) & \text{for } t_p^\alpha, \\ D_p(\beta) + \sum\limits_{q\notin\mathscr{P}_{\alpha\beta}\wedge \text{ neighbor of} p} V_{p,q}(\beta, u_q) & \text{for } t_p^\beta, \\ V_{p,q}(\alpha,\beta) & \text{for } e_{p,q}, \end{cases}$$

where $D_p(l)$ measures the distance of the actual label of the pixel p to the labeling l as defined

in (3.2)(e.g, $D_p(l) = 0$ for $p \in \mathscr{P}_l$) and $V_{p,q}(\alpha,\beta)$ is given by the definition (3.3).

We can now define a natural labeling u^C corresponding to a cut C on $G_{\alpha\beta}$:

$$u_p^C = \begin{cases} \alpha & \text{if } t_p^\alpha \in C \text{ for } p \in \mathscr{P}_{\alpha\beta}, \\ \beta & \text{if } t_p^\beta \in C \text{ for } p \in \mathscr{P}_{\alpha\beta}, \\ u_p & \text{for } p \in P, p \notin \mathscr{P}_{\alpha\beta}. \end{cases}$$

The pixel $p \in \mathscr{P}_{\alpha\beta}$ is assigned with the label α if the cut C seperates p from the terminal α, similarly with the β terminal. If p is not in $\mathscr{P}_{\alpha\beta}$ it keeps its initial u_p.

Remark. It now can be shown that the following results hold, which implies that we found an appropriate graph that returns the minimizer for our optimization problem by calculating the minimum cut of that graph.

Theorem 3.3.1. *There is an one to one correspondence between cuts C on $G_{\alpha\beta}$ and labelings that are one α - β swap from u. Moreover the cost of a cut C on $G_{\alpha\beta}$ is $|C| = E(u^C)$ plus a constant.*

Corollary 1. *The optimal α - β swap from u is $u^* = u^C$ where C is the minimum cut on $G_{\alpha\beta}$.*

The α - β swap algorithm

The algorithm for the optimization via the α - β swap is then given by

1. Start with an arbitrary labeling u
2. Set success := 0
3. For each pair of labels $\alpha, \beta \subset L$

   Find $u' = \text{argmin } E(u')$ among u' within one α - β swap of u as described above.

   If $E(u') < E(u)$ set $u := u'$ and success := 1
4. If success = 1 go to step 2
5. Return u

Remark. Each cycle in the algorithm consists of an iteration for every pair of labels (in a fixed or random order). The algorithm stops after the first unsuccessful cycle, i.e. if no further improvement is possible. One cycle in the swap move algorithm takes $|L|^2$ iterations. The algorithm has the major advantage that it is guaranteed to terminate in a finite number of cycles. As stated in [BVZ14] one can prove termination in $\mathscr{O}(|P|)$ cycles. On the other hand the major limitation of the method lies in the discrete label space in vector valued images. Since the number of discrete labels scales exponentially with the dimension s of the codomain, the computational costs grow exponentially in s. For example using a vector valued and just binary image with $s = 30$ channels, we have to deal with 2^{30} different label possibilities. Therefore just one cycle in the algorithm takes $|2^{30}|^2$ iterations! The resulting immense computational cost is a big disadvantage of using Graph Cuts on vector valued images.

3.3.2 The optimal α expansion move

Given a label α, a move from a labeling u to a new labeling u' is called an $\alpha-$ expansion if $P_\alpha \subset P'_\alpha$ and $P_l \subset P'_l$ for any label $l \neq \alpha$, i.e. an $\alpha-$expansion move allows any set of image pixels to change their labels to α.

The idea is quite smiliar to the α - β swap and can be found in [BVZ14], therefore I will not go into details this time. Alike the α - β swap we want to find a labeling u' that minimizes the energy function $E(u)$ over all labelings within one $\alpha-$expansion of u. It is based again on computing a labeling corresponding to a minimum cut on a graph $G_\alpha = (V_\alpha, E_\alpha)$. The structure of the graph G_α is similar as before and it can be shown that the elementary cuts C on G_α correspond in a natural way to labelings u^C which are within one α expansion move of u. That leads to the main results:

Theorem 3.3.2. *Let G_α be constructed as in Section 4 of [BVZ14].Then there is a one to one correspondence between elementary cuts on G_α and labelings within one α-expansion of u. Moreover the cost of an elementary cut C is $|C| = E(u^C)$.*

Corollary 2. *The optimal α- expansion from u is $u^* = u^C$, where C is the minimum cut on G_α.*

The α-expansion algorithm

The algorithm is the same as in section 3.3.1, expect for step 3.1., where we look for the minimum within one α-expansion of u.

Remark. Of course there also exist other approaches for finding appropriate graphs. E.g. in [BT09] a detailed description of using Graph Cuts on the popular Mumford-Shah Model can be found.

3.3.3 Optimality properties

As already mentioned it is a NP-hard problem to compute the exact minimum of general energy functionals of the form (3.1). But as stated in [BVZ14] it is shown that any local minimum generated by the $\alpha - expansion$ algorithm is within a known factor of the global optimum depending on V. The factor is given by

$$2c = 2 \max_{p,q \in P} \frac{max_{\alpha \neq \beta} V(\alpha, \beta)}{min_{\alpha \neq \beta} V(\alpha, \beta)}.$$

Note that this algorithm just works if V is a metric, otherwise the $\alpha - \beta$ swap algorithm will be used, which can be applied to a wider class of semimetrices V but does not have any guaranteed optimality properties. In that case a local minimum can be arbitrarily far from the global minimum. But it is also shown that a provable good solution can be obtained even for a semimetric V by approximating it by a simple Potts metric.

3.4 Graph Cuts and the Potts model

A stated in Section 3.1.1 the interaction penalty V in the Potts model energy is a metric. Therefore the $\alpha - expansion$ algorithm gives a solution within a factor of two of the global minimum, see [BVZ14]. The Potts model energy minimization problem is closely related to the multiway cut problem, which is a generalization of the standard two-terminal graph cut problem described in that chapter. In [BVZ14] it is shown that the Potts model energy minimization problem can be reduced to the multiway cut problem. The global minimum of the Potts model energy E_p can be computed by finding the minimum cost multiway cut on an (again) appropriately constructed graph. Moreover it is proven that if it would be possible to compute the global minimum of E_p it would also be possible to solve a certain class of multiway cut problems that are known to be NP-hard. That implies that minimizing the Potts model is always NP-hard and also the more general energy functions of the form (3.1).

4 Energy Minimization via the Alternating Direction Method of Multipliers

4.1 Motivation

The alternating direction method of multipliers (ADMM) is an iterative algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle. It has recently found wide application in a number of areas. Some can be found in [Ess09], [DR15] and [WBAW]. The ADMM was already developed in the 1970s, with roots in the 1950s. In [BPC+11] one can find a detailed description and background of the algorithm and an argumentation why the method is well suited to distributed convex optimization and in particular to large-scale problems arising in statistics, machine learning and other related areas.

The method is closely related to several optimization algorithms. Two important of them are the DUAL ASCENT METHOD with dual decomposition and the METHOD OF MULTIPLIERS. The ADMM combines the advantages of both algorithms. Therefore I will give a brief overview of those two algorithms and their background. Then I will go into detail of the ADMM and the possible application to the discretized Potts model including convergence results.

4.2 The dual ascent method

At first I will provide some theoretical basics in convex optimization.

4.2.1 Dual problem

Given an equality-constrained convex optimization problem

$$\min_{\{x\in\mathbb{R}^n : Ax=b\}} f(x), \tag{4.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, we assume its domain $D = \{x \in \mathbb{R}^n : Ax = b\}$ is nonempty and denote the optimal value by x^*. The LAGRANGIAN $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ of (4.1) is defined by

$$L(x,y) = f(x) + y^T(Ax - b).$$

The variable $y \in \mathbb{R}^m$ is called LAGRANGE MULTIPLIER or dual variable. The LAGRANGE DUAL FUNCTION $g : \mathbb{R}^m \to \mathbb{R}$ for the problem (4.1) is given by

$$g(y) = \inf_x L(x,y).$$

Remark. For fixed x, the Lagrangian L is affine in y. Since the dual function g is the pointwise infimum of an affine function, it is always a concave function (even if the original problem (4.1) is not convex!).

Remark. Note that the dual function $g(y)$ may take the value $-\infty$.

Theorem 4.2.1. *It holds that*

$$g(y) \leq x^*,$$

for any $y \in \mathbb{R}^m$. A simple proof can be found in [BV99].

The LAGRANGE DUAL PROBLEM is defined by

$$\max_y g(y) \tag{4.2}$$

and y^* denotes the optimal dual point. Since g is always concave, (4.2) is a convex problem and therefore always has an optimal point.

We know from Theorem 4.2.1 that $y^* \leq x^*$ always holds, which is called WEAK DUALITY. The difference between both optimal points $\Delta^* := x^* - y^* > 0$ is called DUALITY GAP. In general $\Delta^* \neq 0$, but under some additional properties on the objective function f (e.g. Slater's condition) STRONG DUALITY holds, i.e. $\Delta^* = 0$. Further details about duality can be found in [BV99].

We now assume that strong duality holds, therefore we can recover a primal optimal value x^* from a dual optimal point as

$$x^* = \operatorname*{argmin}_x L(x,y^*),$$

provided that there is only one minimizer of $L(x,y^*)$, e.g. if f is strictly convex.

4.2.2 Dual ascent method

The dual ascent method provides x^* by solving the dual problem (4.2) iteratively. The iteration is given by

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L(x,y^k) \\ y^{k+1} &:= y^k + \alpha^k (Ax^{k+1} - b). \end{aligned} \tag{4.3}$$

In each iteration step we first calculate $x^{k+1} = \operatorname{argmin} L(x^k,y)$, then we update the dual variable via line search. The LINE SEARCH strategy (see e.g. [WS06]) is a numerical method for finding some local optimum iteratively. In our case we want to maximize the dual function (because of (4.2)), i.e. we want some y^* such that $\nabla g(y^*) = 0$ holds.

Remark. The dual ascent method can be used in some cases when the dual function g is not differentiable [BPC$^+$11], it is then called DUAL SUBGRADIENT METHOD. In that case the residual $Ax^{k+1} - b$ is not the gradient of g, but the negative of a subgradient (see Definition 6) of $-g$. Those cases require a different choice of the step length α and it appears that the convergence is not monotone.

Each line search method computes a search direction s^k and then tells with a step length α^k how far to move along that direction. One iteration step is given by

$$y^{k+1} = y^k + \alpha^k s^k.$$

The success of a line search method depends on effective choices of the direction and the step length. Since we want to find the maximum for the dual problem we require that in each iteration step $g(y^{k+1}) > g(y^k)$ holds (which can be satisfied with some appropriate choice of α^k). In the dual ascent method we have to assume that g is differentiable. The search direction s^{k+1} is defined as $Ax^{k+1} - b$, which is the gradient of $g(y^{k+1})$. That can be calculated since we first compute $x^{k+1} = \operatorname*{argmin}_x L(x^k, y)$ in every step. Thus

$$\begin{aligned} \nabla g(y^{k+1}) &= \nabla_y \inf_x L(x, y^{k+1}) \\ &= \nabla_y L(x^{k+1}, y^{k+1}) \\ &= \nabla_y \left(f(x^{k+1}) + y^{(k+1)^T} (Ax^{k+1} - b) \right) \\ &= Ax^{k+1} - b \end{aligned} \tag{4.4}$$

holds.

Remark. More about LINE SEARCH methods on dual problems can be found in [ZL96].

As stated in [BPC$^+$11] it needs several assumptions on α^k and f such that x^k and y^k converge to an optimal (dual) point. Unfortunately the assumptions often do not hold in applications, therefore the method often is not appropriate.

4.2.3 Dual decomposition

The major benefit of the dual ascent method is that it can lead to a decentralized algorithm. For example if we can split the objective function f such that

$$f(x) = \sum_{i=1}^{N} f_i(x_i)$$

holds (f is seperable), where $x = (x_1, ..., x_N)$ and the variables $x_i \in R^{n_i}$ are subvectors of x. We now can take the suitable partition of the Matrix $A = [A_1, ..., A_N]$, such that $Ax = \sum_{i=1}^{N} A_i x_i$. Therefore the Lagrangian can be rewritten as

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} \left(f_i(x_i) + y^T A_i x_i - \frac{1}{N} y^T b \right).$$

The x-minimization step in (4.3) splits now into N seperate problems that can be solved in parallel. Explicitly the algorithm can now be written as

$$\begin{aligned} x_i^{k+1} &:= \operatorname*{argmin}_{x_i} L_i(x_i, y^k) \\ y^{k+1} &:= y^k + \alpha^k (Ax^{k+1} - b). \end{aligned} \tag{4.5}$$

For each $i = 1, ..., N$ the x-minimization step is processed independently. In the dual update step the $A_i x_i^{k+1}$ are collected to compute the residual $Ax^{k+1} - b$. After the dual variable y^{k+1} is computed it will be distributed again to carry out the N individual x_i minimization steps. Those two steps are called broadcast and gather operations (see [BPC+11]). The main benefit is that we now have split the problem into lower dimensional problems, that can be solved with less computational cost.

Remark. That method is now called dual decomposition instead of dual ascent method. Dual decomposition is based on ideas of the early 1960s and there are several papers that discuss different approaches. Some references can be found in [BPC+11].

4.3 Method of multipliers

4.3.1 Augmented Lagrangian

Augmented Lagrangian methods are developed to bring robustness to the dual ascent method and to yield convergence without that strict assumptions on f. The AUGMENTED LAGRANGIAN for the original problem (4.1) is defined by

$$L_\rho(x, y) = f(x) + y^T (Ax - b) + (\rho/2)\|Ax - b\|_2^2,$$

where $\rho > 0$ denotes the PENALTY PARAMETER. It is the same as calculating the Lagrangian of the problem

$$\min_{s.t. Ax = b} f(x) + (\rho/2)\|Ax - b\|_2^2,$$

what is clearly equivalent to the original problem. The associated dual function is now given by

$$g_\rho(y) = \inf_x L_\rho(x, y).$$

The major benefit of including the penalty term is that g_ρ can be shown to be differentiable under much milder conditions on the original problem as before.

4.3.2 Algorithm

The algorithm is very similar to the standard dual ascent method, but it uses the augmented Lagrangian instead of the usual Lagrangian and ρ is used as the step length in the line search. Therefore the algorithm is given by

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} - b) \end{aligned} \tag{4.6}$$

and is called METHOD OF MULTIPLIERS for (4.1).

Remark. It is easy to motivate why the penalty parameter ρ is an appropriate choice for the step length in the line search. For simplicity we assume here that f is differentiable (which is not required for the algorithm to work). In the first step in the algorithm x^{k+1} minimizes $L_\rho(x, y^k)$, therefore it holds that

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T(y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1}. \end{aligned} \tag{4.7}$$

So by using ρ as the step length it yields that the iterate (x^{k+1}, y^{k+1}) is always dual feasible (i.e. y^{k+1} is a solution of the dual problem $max_y\, g_\rho(y)$ since $\nabla g_\rho(y^{k+1}) = \nabla f(x^{k+1}) + A^T y^{k+1} = 0$).

4.3.3 Advantages and disadvantages

As already proposed the method converges under rather mild conditions [BPC+11], but that comes at a cost. When f is separable, the augmented Lagrangian L_ρ is not separable, so the x-minimization step cannot be carried out separately in parallel for each x_i as in (4.5). Therefore the basic method of multipliers cannot be used with decomposition. We are looking for an algorithm that combines the convergence properties of the method of multipliers with the decomposability of dual ascent. That leads to the ADMM.

4.4 ADMM (alternating direction method of multipliers)

As described in [BPC+11] we require in the ADMM that the optimization problem has the form

$$\min_{Ax+Bz=c} f(x) + g(z), \tag{4.8}$$

with primal variables $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$ and $A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}$ and $c \in \mathbb{R}^p$. Assumptions on f and g will be discussed in Appendix A. Compared to the general problem (4.1) it is the only difference that the primal variable x has been split into two parts. The optimal value of the new problem will be denoted by

$$p^* = \inf\{f(x) + g(z) \mid Ax + Bz = c\}.$$

As in Section 4.3.1 we form the augmented Lagrangian

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2. \tag{4.9}$$

4.4.1 Algorithm

The iterations of the ADMM are given by

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \operatorname*{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c), \end{aligned} \tag{4.10}$$

where $\rho > 0$ again is the penalty parameter. The algorithm consists of an x and a z-minimization step and an update of the dual variable y. As in the method of multipliers the line search for the dual variable update uses a step length equal to the augmented Lagrangian penalty parameter ρ. The main difference to the method of multipliers applied to the new problem (4.8) is that the primal variable is split into two primal variables x and z, which are not treated jointly, i.e. they are updated in an alterning/sequential fashion (which gives the method its name). The algorithm state in ADMM consists of z^k and y^k, i.e. (z^{k+1}, y^{k+1}) is a function of (z^k, y^k), on the contrary x^k is not part of the state - it is an intermediate result computed from the previous state (z^{k-1}, y^{k-1}). Therefore the roles of x and z are almost symmetric, but not quite.

Scaled Form

The algorithm can be rewritten in a scaled form, which is often more convenient to use. The idea is to combine the linear and quadratic terms in the augmented Lagrangian and to scale the dual variable. Let $d = \frac{1}{\rho}\, y$ be the SCALED DUAL VARIABLE and $r := Ax + Bz - c$ for simplicity. Then the second part of the augmented Lagrangian (4.9) can easily be transformed by the equality

$$\begin{aligned} y^T r + \frac{\rho}{2}\|r\|_2^2 &= \frac{\rho}{2}\|r + \frac{1}{\rho}y\|_2^2 + \frac{1}{2\rho}\|y\|_2^2 \\ &= \frac{\rho}{2}\|r + d\|_2^2 - \frac{\rho}{2}\|d\|_2^2. \end{aligned} \tag{4.11}$$

Hence we can reformulate the ADMM algorithm (4.10) by

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x \left(f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + d^k\|_2^2 \right) \\ z^{k+1} &:= \operatorname*{argmin}_z \left(g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + d^k\|_2^2 \right) \\ d^{k+1} &:= d^k + Ax^{k+1} + Bz^{k+1} - c. \end{aligned} \tag{4.12}$$

Defining the residual at iteration step k as $r^k = Ax^k + Bz^k - c$ it holds that

$$d^k = d^0 + \sum_{j=1}^{k} r^j,$$

i.e. the scaled dual variable can be calculated via the running sum of the residuals.

The algorithm in (4.10) is called UNSCALED FORM and the one in (4.12) the SCALED FORM. The two algorithms are equivalent, but the notation of the scaled one is shorter and more convenient to use, therefore we will work with that one. The unscaled form is used when it is important to emphasize the role of the dual variable or to give an interpration that is based on the unscaled dual variable.

Remark. There do exist many extensions and variations on the classic ADMM algorithm in the literature. A standard extension is to use different penalty parameters ρ^k (instead of the same for each iteration) to improve the convergence in practice or to get less dependent on the initial choice of the penalty parameter. As stated in [BPC+11] it can be difficult to prove the convergence of ADMM in those cases, but the fixed-ρ theory still holds if one assumes that ρ becomes fixed after a finite number of iterations.

Another variation is to change the order of updating the variables, to do it in multiple times or to add an d-update between the x and y-update with half the step length. E.g. one can divide the primal variable into k blocks/primal variables and update each of them in turn before performing each dual variable update. In [HL13] a proof for global linear convergence of the ADMM can be found for k-splitted convex and separable functions.

A detailed proof of convergence of the ADMM with assumptions on the objective functions can be found in Appendix A (based on the one in [BPC+11]).

4.5 The ADMM applied to the Potts model

We now apply the unscaled form (4.12) of the ADMM to the Potts problem

$$u^* = \operatorname*{argmin}_u \gamma \cdot \|\nabla u\|_0 + \|u - z\|_2^2 \,. \tag{4.13}$$

We will deduce it the same way as described in [SW14], where they consider a four-connected neighborhood for each pixel and first start with a given greyscale image $z \in \mathbb{R}^{m\times n}$. We now set the neighborhood weights ω in the discretized Potts model (2.11) uniformly equal to 1. Since we chose a four-connected neighborhood the jump penalty can be expressed as

$$\|\nabla u\|_0 = \|\nabla_1 u\|_0 + \|\nabla_2 u\|_0 := \sum_{i,j} [u_{i,j} \neq u_{i+1,j}] + \sum_{i,j} [u_{i,j} \neq u_{i,j+1}].$$

To get the Potts problem in the needed form (4.8) for the ADMM, we rewrite it as the bivariate constrained optimization problem

$$\min_{\{u,v\in\mathbb{R}^{m\times n}: u-v=0\}} \quad \gamma \|\nabla_1 u\|_0 + \gamma \|\nabla_2 v\|_0 + \frac{1}{2}\|u - z\|_2^2 + \frac{1}{2}\|v - z\|_2^2, \tag{4.14}$$

with $u, v \in \mathbb{R}^{m\times n}$.

Remark. Note that we calculate with the ADMM the objective minimum instead of the minimal argument u^*wanted in (4.13). But it can be shown (see Theorem 4.5.1) that the primal variables u and v (i.e. the arguments) converge in the algorithm.

The functional is now splitted in two parts with two primal variables u and v. For simplicity we define

$$f(u) := \gamma \|\nabla_1 u\|_0 + \frac{1}{2}\|u-z\|_2^2$$

and

$$g(v) := \gamma \|\nabla_2 v\|_0 + \frac{1}{2}\|v-z\|_2^2.$$

As stated in (4.9) the augmented Lagrangian is now given by

$$L_\rho(u,v,y) = f(u) + g(v) + y^T(u-v) + \frac{\rho}{2}\|u-v\|_2^2. \tag{4.15}$$

Again $\rho > 0$ regulates how strongly the difference between u and v is penalized and the dual variable y is an $(m \times n)$-dimensional matrix of Lagrange multipliers. Since

$$y^T(u-v) + \frac{\rho}{2}\|u-v\|_2^2 = \frac{\rho}{2}\|u-v+\frac{y}{\rho}\|_2^2 - \frac{\rho}{2}\|\frac{y}{\rho}\|_2^2$$

holds, the augmented Lagrangian (4.15) can be rewritten as

$$L_\rho(u,v,y) = f(u) + g(v) + \frac{\rho}{2}\|u-v+\frac{y}{\rho}\|_2^2 - \frac{\rho}{2}\|\frac{y}{\rho}\|_2^2,$$

which is the form we need for the scaled ADMM.

Remark. Note that the variables u, v and y in (4.15) are now matrices in $\mathbb{R}^{m\times n}$ instead of vectors as described in (4.9). But since the functions $f(u)$ and $g(v)$ just consist of the discrete matrix norms $\|\cdot\|_2$ and $\|\cdot\|_0$ which can be easily adapted to vector norms by reordering the entries of the matrix, it is equivalent to the form of the augmented Lagrangian for the ADMM in (4.9).

4.5.1 Algorithm for the Potts problem

Thus the ADMM for the Potts problem reads

$$\begin{aligned} u^{k+1} &:= \operatorname*{argmin}_u \left(f(u) + \frac{\rho}{2}\|u - (v^k - \frac{y^k}{\rho})\|_2^2 \right), \\ v^{k+1} &:= \operatorname*{argmin}_v \left(g(v) + \frac{\rho}{2}\|v - (u^{k+1} + \frac{y^k}{\rho})\|_2^2 \right), \\ y^{k+1} &:= y^k + \rho \cdot (u^{k+1} - v^{k+1}). \end{aligned} \tag{4.16}$$

We now want to simplify the expression for u^{k+1} and v^{k+1}. To do that we first note that both iteration steps are of the same form. I will do the transformations for u^{k+1}, which are equivalent to the ones on v and can be adapted easily. We use the general notation

$$u^{k+1} = \gamma \|\nabla u_1\|_0 + \frac{1}{2}\left(\|u-z\|_2^2 + \rho\|u-(v+\frac{y}{\rho})\|_2^2\right)$$

and will do some transformations on the second addend $\|u-z\|_2^2+\rho\|u-(v+\frac{y}{\rho})\|_2^2$. For simplicity we set $c := v+\frac{y}{\rho}$. We can easily calculate that

$$\begin{aligned}(u-z)^2+\rho(u-c)^2 &= (1+\rho)u^2-2u(z+\rho c)+z^2+\rho c^2\\ &= (1+\rho)\Big(u-\frac{z+\rho c}{1+\rho}\Big)^2-\frac{(z+\rho c)^2}{1+\rho}+z^2+\rho c^2\end{aligned}$$

holds. Since the argument of the minimum only depends on the first addend, we just use that. Combining that with the equivalent transformation of v and scaled primal variables yields the iteration steps

$$\begin{aligned}u^{k+1} &:= \operatorname*{argmin}_u \frac{2\gamma}{1+\rho}\|\nabla_1 u\|_0+\|u-(1+\rho)^{-1}(z+\rho v^k-y^k)\|_2^2,\\ v^{k+1} &:= \operatorname*{argmin}_v \frac{2\gamma}{1+\rho}\|\nabla_2 v\|_0+\|v-(1+\rho)^{-1}(z+\rho u^k+y^k)\|_2^2,\\ y^{k+1} &:= y^k+\rho\cdot(u^{k+1}-v^{k+1}).\end{aligned} \tag{4.17}$$

We observe that the expressions for u^{k+1} and v^{k+1} are separable into n subproblems of the form

$$\begin{aligned}u_{:,j}^{k+1} &:= \operatorname*{argmin}_{h\in\mathbb{R}^m} \frac{2\gamma}{1+\rho}\|\nabla h\|_0+\|h-(1+\rho)^{-1}(z_{:,j}+\rho v_{:,j}^k-y_{:,j}^k)\|_2^2,\\ v_{i,:}^{k+1} &:= \operatorname*{argmin}_{h\in\mathbb{R}^n} \frac{2\gamma}{1+\rho}\|\nabla h\|_0+\|h-(1+\rho)^{-1}(z_{i,:}+\rho u_{i,:}^k+y_{i,:}^k)\|_2^2,\end{aligned} \tag{4.18}$$

for $j=1,\dots,n$ and $i=1,\dots,m$. The fundamental strategy is that these subproblems are univariate Potts problems which can be solved exactly and efficiently using dynamic programming. A detailed description about that can be found in [SW14].

4.5.2 Choosing the parameters ρ and γ

In the algorithm described in [SW14] the penalty parameter ρ changes with each iteration (which is a common variation in the ADMM as stated in Remark 4.4.1). The parameter ρ_0 is initialized small and positive and increases during the iteration by a factor $\tau>1$. Hence, ρ is given by the geometric progression

$$\rho=\rho_k=\tau^k\rho_0.$$

That strategy ensures that u and v can evolve quite independently at the beginning and will be close to each other at the end of the iteration. That variation of the standard ADMM is used because the geometric progression yields satisfactory results while being very fast. When the difference of u and v falls below some fixed tolerance, the iterations will be stopped.

The other important decision is the choice of the regularization parameter γ. Intuitively γ can be interpreted as a scaling parameter. In connection with one-dimensional Potts functionals different approaches for an appropriate choice of γ are reported in the literature. E.g. some ideas can be found in [DK01] and [WWLK]. In [SW14] they choose the parameter γ in their algorithm empirically.

Algorithm as stated in [SW14]

INPUT Image $z \in \mathbb{R}^{m\times n}$, parameter γ, initial value $\rho_0 > 0$, step size $\tau > 1$.
OUTPUT Computed result $u \in \mathbb{R}^{m\times n}$ to the Potts problem.
begin
$v = z$; $\rho = \rho_0$; $y = 0$;
repeat
  for j = 1:n **do**
    minimize subproblem (4.18a) to get $u_{:,j}$
  end for
  for i = 1:m **do**
    minimize subproblem (4.18b) to get $v_{i,:}$
  end for
  $y = y + \rho(u - v)$;
  $\rho = \tau \cdot \rho$;
until reached stopping criterion
end

Algorithm 1: ADMM strategy for the Potts problem

Theorem 4.5.1. *The algorithm 1 converges in the sense that there exists an u^* such that $u^k \to u^*$ and $v^k \to u^*$.*

A detailed but very technical proof for the primal variable convergence of the algorithm (4.17), i.e. of Theorem 4.5.1, can be found in [SW14].

Remark. Note that since the problem (4.13) is a non convex optimization problem and the functions $f(u)$ and $g(v)$ in (4.9) do not fulfill the needed assumptions to show convergence of the algorithm (see Appendix A) objective and residual convergence must not always appear. Nevertheless Theorem 4.5.1 guarantees primal variable convergence which is from main interest since we are looking for the minimal argument.

4.5.3 Vector-valued images

The proposed method can be extended to vector-valued images $z \in \mathbb{R}^{m\times n\times s}$ with $s \geq 1$, e.g. color or multispectral images. Taking the general Potts model from (2.11) the data and jump terms in the ADMM are given by

$$\|u - z\|_2^2 = \sum_{i,j,k} |u_{ijk} - z_{ijk}|^2$$

and

$$\|\nabla u\|_0 = \sum_{(a,b)\in N} \omega_{a,b} \cdot [u_{i,j,:} \neq u_{i+a,j+b,:}].$$

The jump penalty cannot be evaluated componentwise, it is declared if two neighboring vectors $u_{i,j,:}$ and $u_{i+a,j+b,:}$ are not equal. In the ADMM algorithm the intermediate solutions u,v and the multipliers y_i in (4.17) are now in $\mathbb{R}^{m\times n\times s}$, but the Lagrangian multipliers can be carried out componentwise. As stated in [SW14] the same dynamic program as in the two dimensional case is used.

5 Application of the Potts Problem on Clinical Retinal Images

In that chapter I am going to apply the algorithm described in 4.5, which is proposed in [SW14], to clinical retinal images. I am using the Pottslab matlab code Version 0.42 from the website *http://pottslab.de/pottslab-version-0-42/*.

The data I am using is provided by the VRC. The VRC stores and analyses millions of images, mainly from optical coherence tomography (OCT) that are recorded in multi-center clinical trials in many hundred study sites all over the world. For this master thesis the VRC provided retinal OCT images and annotations of intraretinal fluid that were performed by medical expert graders. The images are stored in a dcm file as a three dimensional $(1024 \times 512 \times 128)$ matrix, i.e. the volume data consists of 128 cross-sectional images of one retina each of the size (1024×512). Our goal is to identify intraretinal cystoid fluid, further referred to as "cysts" in the human retina. I will do presegmentation of the separated $2D$ greyscale images, which can be a useful sub-step of a more sophisticated and comprehensive image processing pipeline which is developed at the VRC in collaboration with the Vienna Research Group for Young Investigators on computational harmonic analysis of high-dimensional biomedical data.

Remark. To work with smaller data I scaled the images to the size (512×512) since the surroundings do not include necessary information for cyst identification.

5.1 Results

Taking an image containing cysts (e.g. image no. 75 of the volume data) I first processed it directly with the $2D$ potts algorithm. The result with chosen parameters $\gamma = 0.01$, Isotropic = false and Verbose = true yields the image shown in figure 5.3. For bigger γ the vertical boundaries of the cysts vanish and for smaller γ the smoothness gets even worse, see figure 5.4. The processing time was approximately 30 seconds. After trying the potts algorithm on lots of different features of the image, the best result was achieved by using the multispectral image consisting in the first channel of a Gabor filter (see e.g. [FS98]) applied to the original image and in the second channel of a smoothed version of the original image (see figure 5.5). Using the 2D potts algorithm on that multispectral image yields a sufficient result - the cysts are located mainly correct, boundaries are kept well and the whole cross section is segmented in reasonable regions. Furthermore separated cysts remain separated in the segmentation (boundaries are preserved). The time for processing the multispectral (512 x 512 x 2) image with the potts algorithm was approximately 60 seconds.

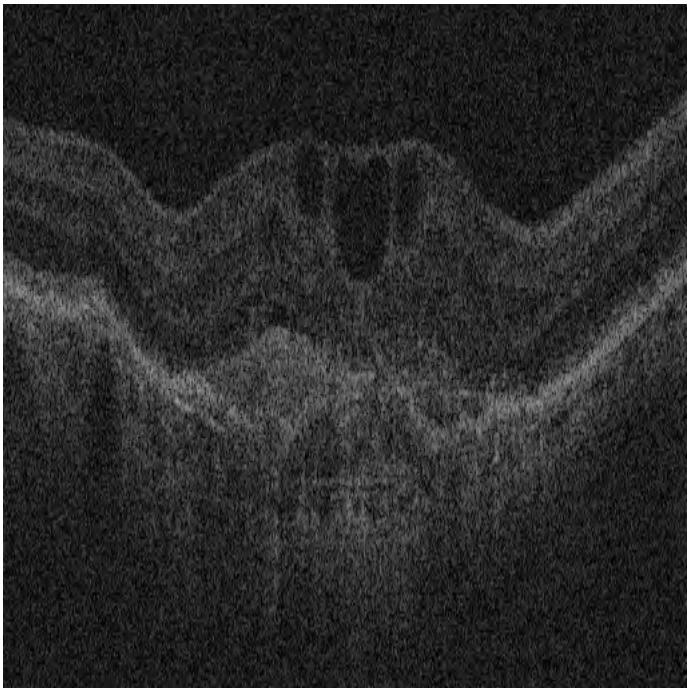


Figure 5.1: Original image no. 75 of size 512x512.

Figure 5.2: Image no. 75 where the regions we want to identify are manually marked (red) via the annotations of intraretinal fluid provided by the VRC.

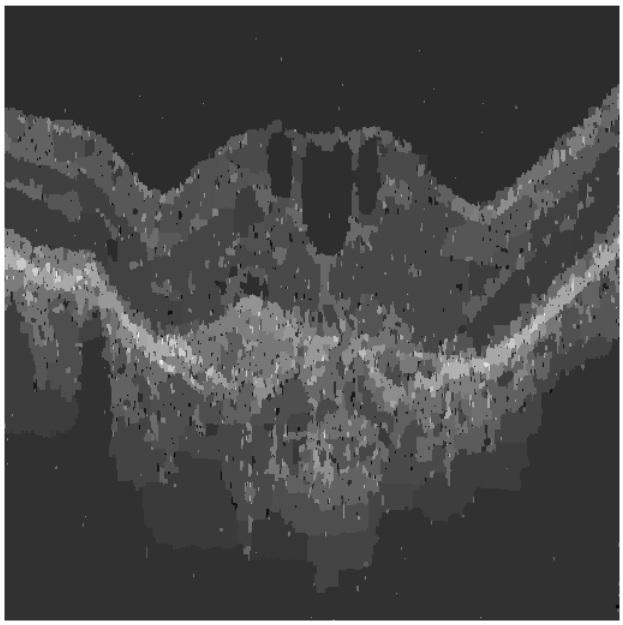
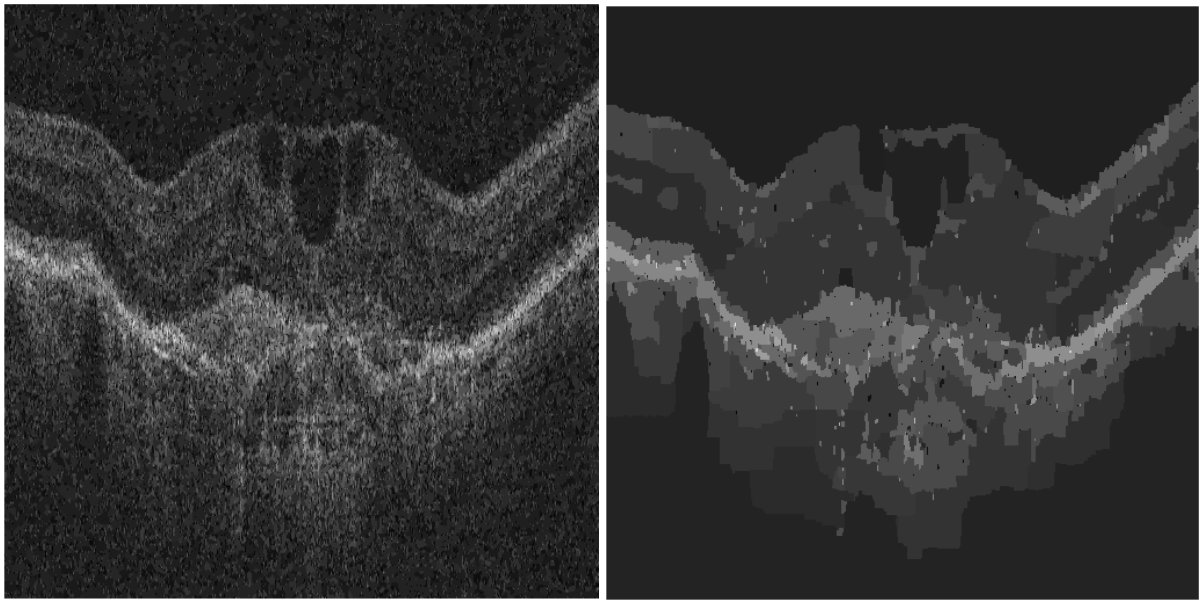
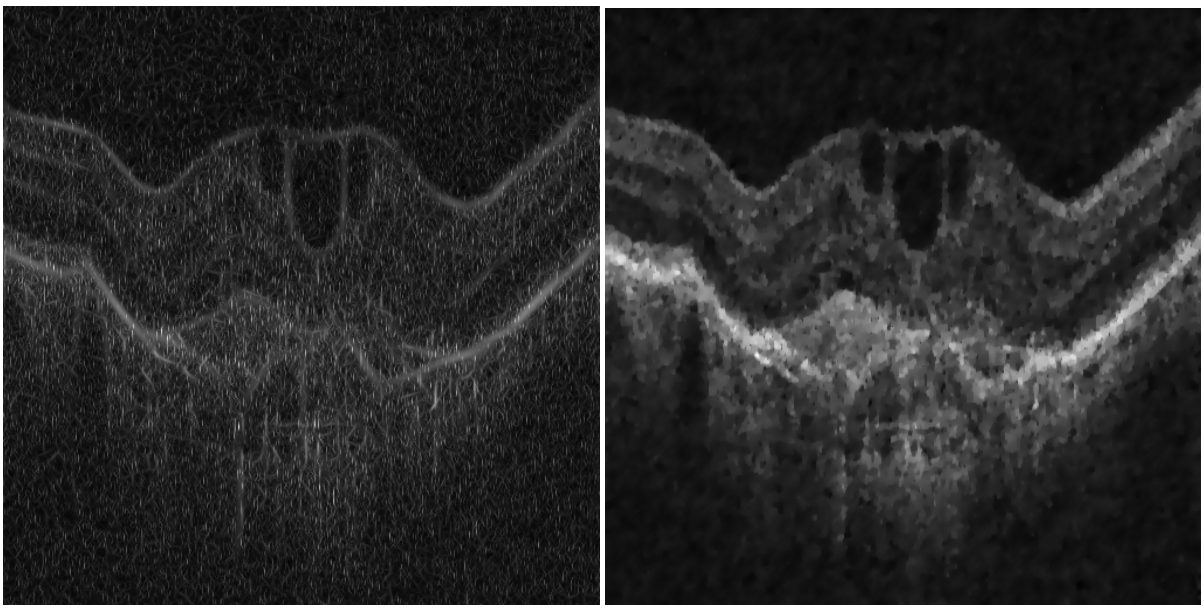


Figure 5.3: Image no. 75 processed with 2D potts and parameter $\gamma = 0.01$.



(a) $\gamma = 0.005$ (b) $\gamma = 0.015$

Figure 5.4: Image no. 75 processed with 2D potts and different γ parameters.



(a) Gaborfilter on image 5.1 (b) Smoothingfilter on image 5.1

Figure 5.5: Features used for the multispectral image of size (512x512x2).

The first channel of the multispectral image consists of a Gabor filter applied to the original image. Gabor filters, named after D. Gabor, are bandpass filters that are used in image processing for feature extraction and texture analysis. A band-pass filter chooses frequencies within a certain range and rejects those outside that range. The impulse response of the $2D$ Gabor filters is created by multiplying an Gaussian kernel function with a sinusoidal plane wave. It is possible to extract different features of an image by choosing different frequencies and orientations in the Gabor filters. They are directly related to Gabor wavelets. More about Gabor analysis and its applications can be found in the book of H.G. Feichtinger and T.Strohmer ([FS98]). A Gabor analysis toolbox can be found on the website *http://ltfat.sourceforge.net/*.

The smoothed image used for the second feature in the multispectral image is reached again by minimizing an energy function. In that case the minimizer of the functional

$$E(u) = \frac{1}{2}\|u-z\|_{L^2}^2+\gamma\|Du\|_{L^1}$$

yields the resulting image. More information about that and the developed matlab algorithm can be found in my Bachelor thesis [Bre13].

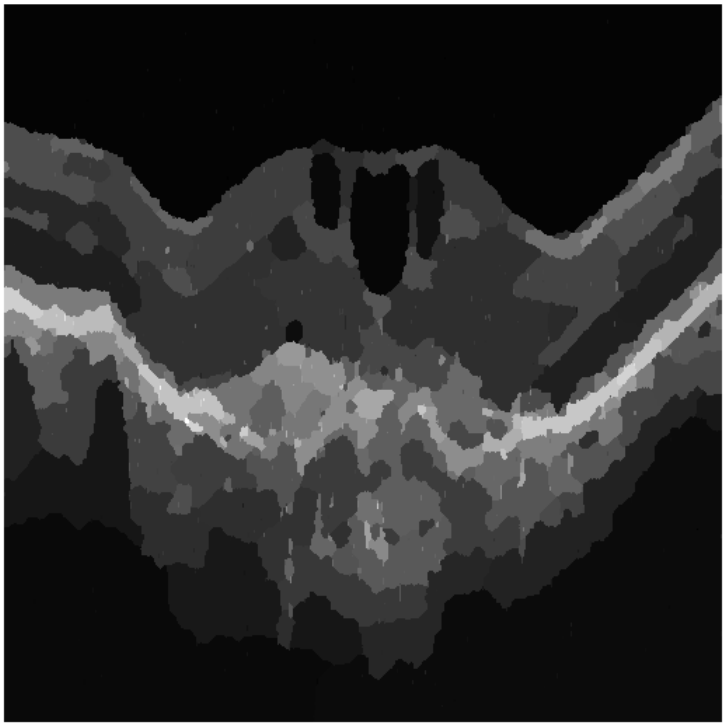


Figure 5.6: Using 2D potts with parameter $\gamma = 0.1$ on the multispectral image of size $512 \times 512 \times 2$ consisting of the 2 channels shown in figure 5.5 .

Because of the previous good result by using the multispectral image (consisting of the first channel with a Gabor filter applied to the original image and the second channel of a smoothed version of the original image) I use the same idea on other images of the data. Each of the multispectral images is then processed with the 2D potts algorithm. It led good results on all 128 images of the volume data and I will show here the most important ones. Next to the original images I will show the image where the regions we want to identify are manually marked (red) via the annotations of intraretinal fluid provided by the VRC.

Image no. 68 (figure 5.7) is the last image of the data before cysts first appear in the cross section, the processed image correctly does not localize cysts here. Image no. 69 (figure 5.9) is the first image where cysts appear - the processed image (figure 5.10) already segments cysts here too. In image no. 77 (figure 5.11) a big amount of cysts appear and again the processed image (figure 5.12) localizes it mainly correct and the boundaries are sharp. Image no.91 is the last image of the cross-sectional data where cysts appear - again it is localized in the processed image. From image no. 92 (figure 5.15) no more cysts appear in the image data, the processed image does not show cysts as well.

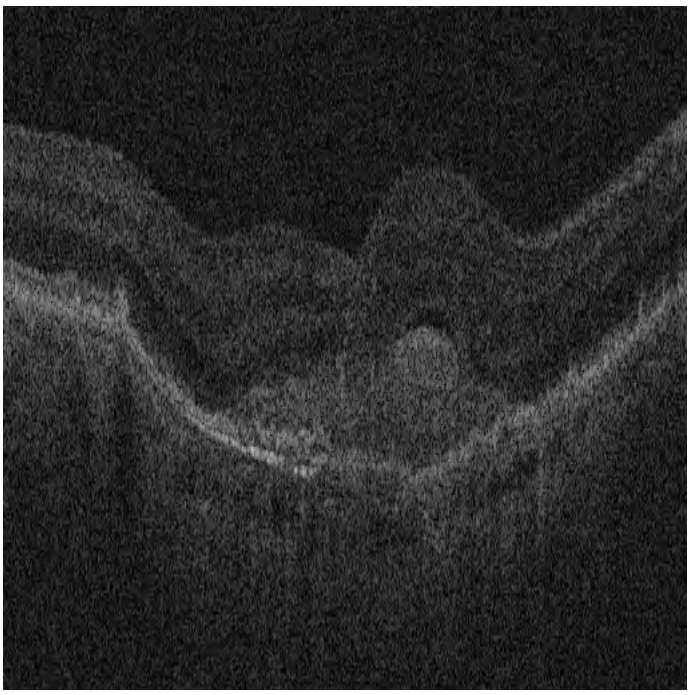


Figure 5.7: Bild no. 68

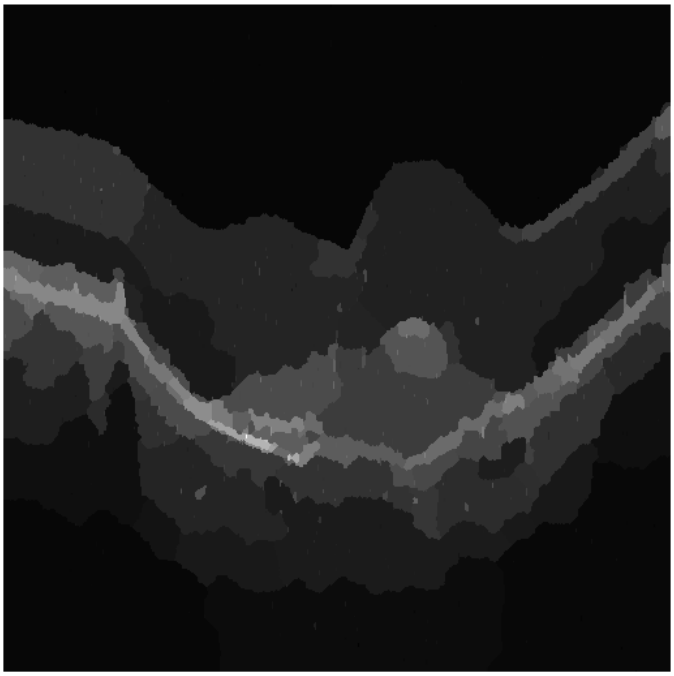
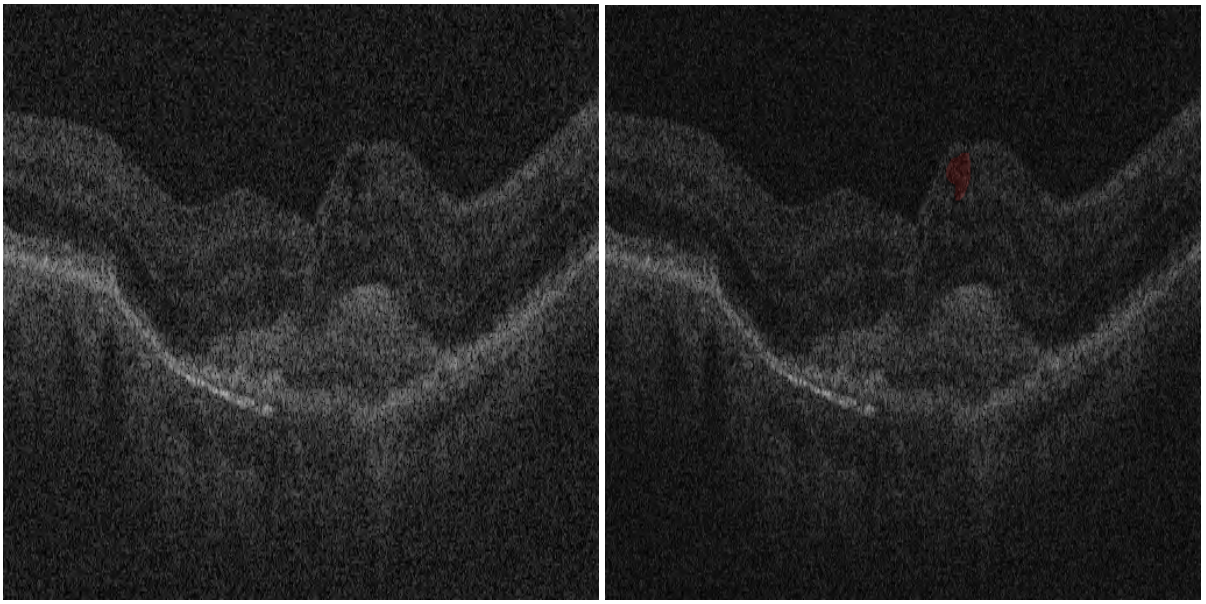


Figure 5.8: Processed image



(a) Original image (b) Intraretinal fluid manually marked

Figure 5.9: Image no. 69

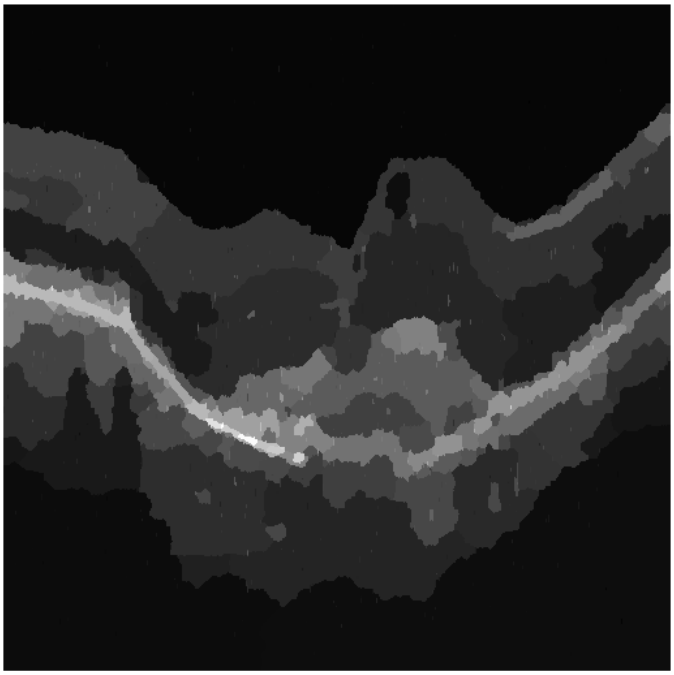
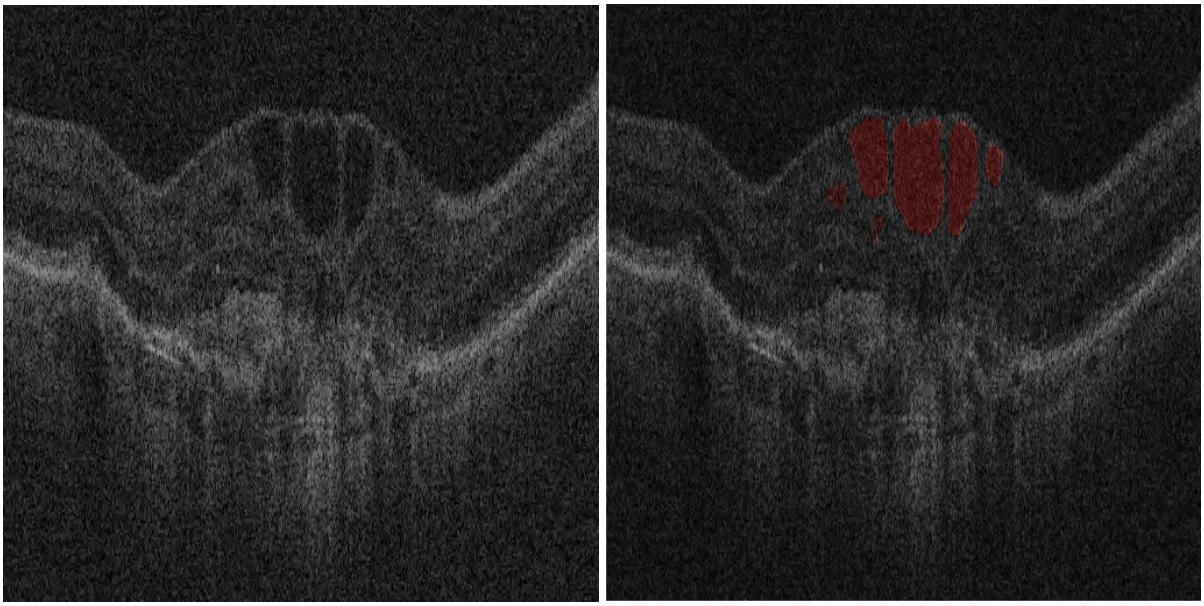


Figure 5.10: Processed image



(a) Original image (b) Intraretinal fluid manually marked

Figure 5.11: Image no. 77

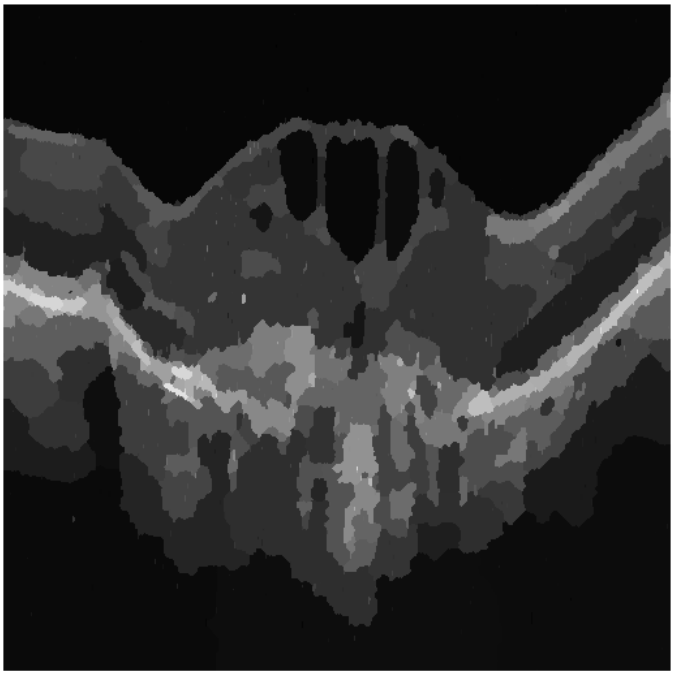
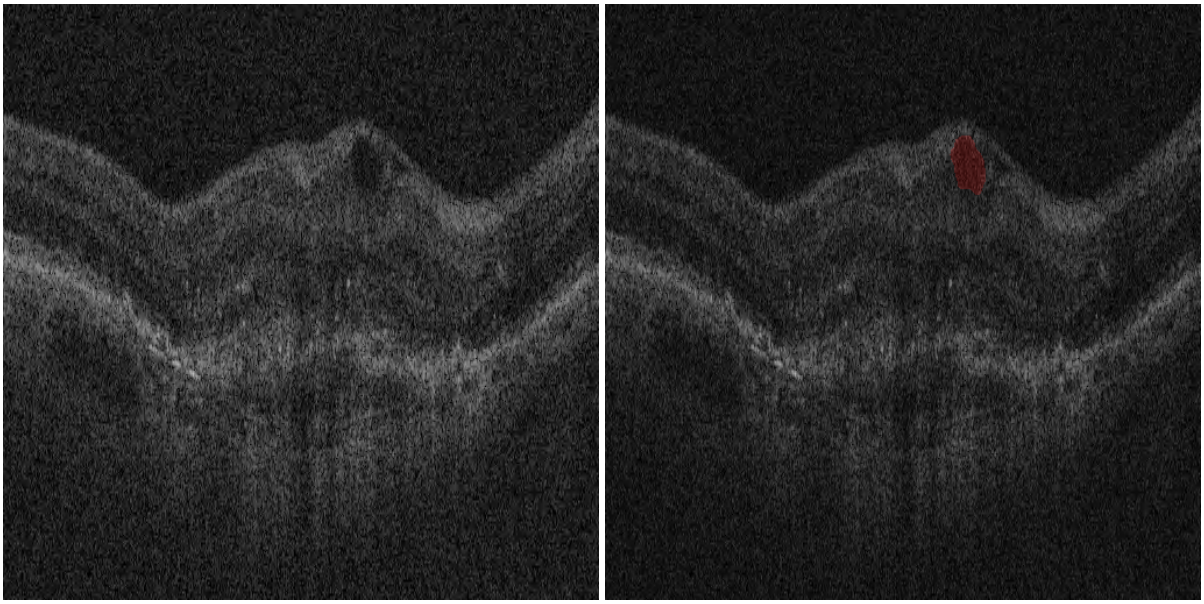


Figure 5.12: Processed image



(a) Original image (b) Intraretinal fluid manually marked

Figure 5.13: Image no. 91

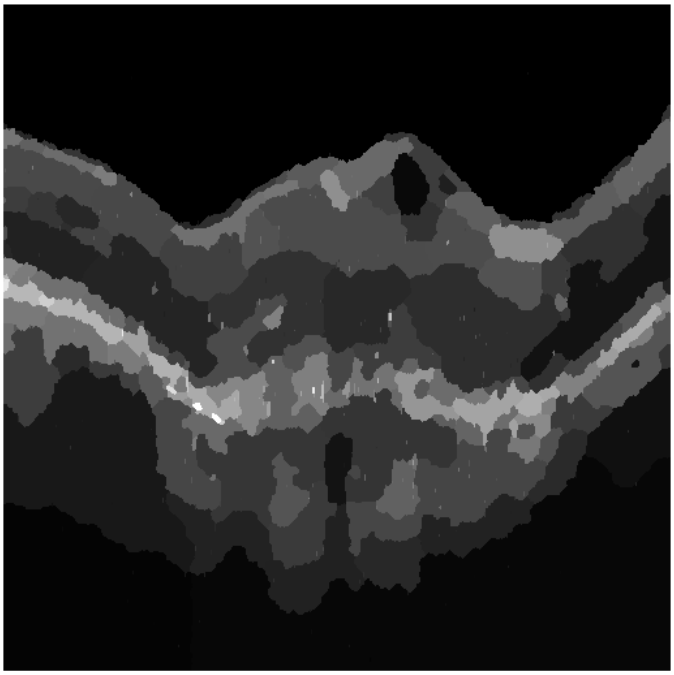


Figure 5.14: Processed image

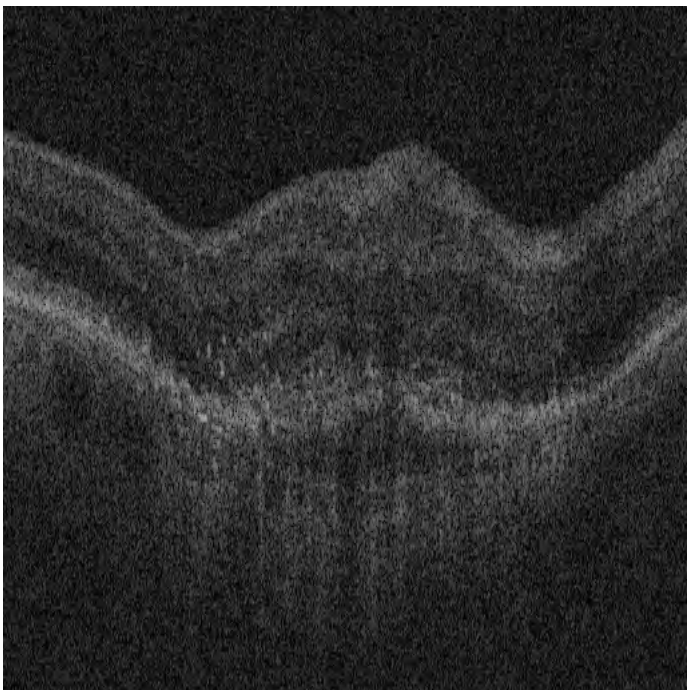


Figure 5.15: Bild no. 92

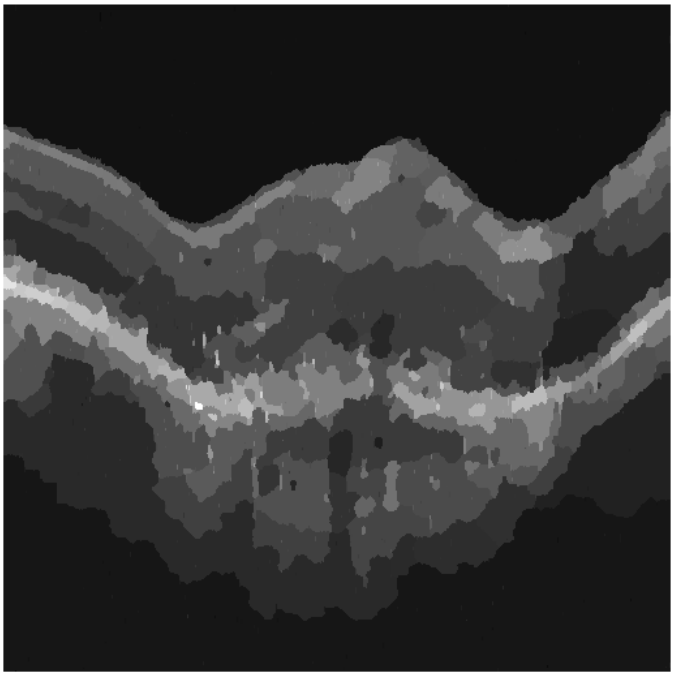


Figure 5.16: Processed image

A Convergence Proof of the ADMM

There do exist many convergence results of the ADMM that are already discussed in the literature. Here, a basic and general convergence proof based on the one in [BPC+11] will be stated. In that proof we need some assumptions on the splitted objective functions f, g and the unaugmented Lagrangion L_0.

Assumption 1. The functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper and convex.

That is equivalent to claiming that the epigraphs of f and g are closed nonempty convex sets. It is important to note that the assumption allows f and g to be nondifferentiable and to assume the value $+\infty$.

Assumption 2. The unaugmented Lagrangian L_0 has a saddle point, i.e. there exist $(x^*, z^*, y^*) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ (not necessarily unique) such that

$$L_0(x^*, z^*, y) \leq L_0(x^*, z^*, y^*) \leq L_0(x, z, y^*)$$

holds for all $x \in \mathbb{R}^n, z \in \mathbb{R}^m, y \in \mathbb{R}^p$.

Remark. With those two assumptions we can already follow some important facts. With assumption 1 we get that $L_0(x^*, z^*, y^*)$ is finite for any saddle point. Combined with assumption 2 that implies that (x^*, z^*) is primal optimal and y^* is dual optimal. Moreover strong duality holds, i.e. the primal and dual problems are equal.

Note that we do not need any explicit assumptions on A, B or c of the equality constraint. The matrices A and B are not required to have full rank.

Theorem A.0.1. *Under the assumptions 1 and 2 the iterates of the ADMM stated in* (4.16) *satisfy*

1. ***Residual convergence***
   $r^k := (Ax^k + Bz^k - c) \to 0$ *as* $k \to \infty$*, i.e. the iterates approach primal feasibility.*

2. ***Objective convergence***
   $p^k := f(x^k) + g(z^k) \to p^*$ *as* $k \to \infty$*, where* $p^* := inf\{f(x) + g(z) | Ax + Bz = c\}$*. I.e. the objective function of the iterates approaches the optimal value.*

Remark. Note that under that conditions the primal variables x^k and z^k in (4.12) do not necessarily converge to (primal) optimal values. On the other hand in [SW14] it is shown that the algorithm applied to the Potts model yields primal variable convergence (see Theorem 4.5.1), but the functions do not fulfill the assumptions 1 and 2 and therefore neither residual nor objective convergence must appear.

A.1 Proof of Theorem A.0.1

Let (x^*, z^*, y^*) be a saddle point for the unaugmented Lagrangian $L_0 = f(x) + g(y) + y^T(Ax + Bz - c)$ and define

$$V^k = \frac{1}{\rho}\|y^k - y^*\|_2^2 + \rho\|B(z^k - z^*)\|_2^2.$$

The proof is based on three key inequalities, which will be stated first and proved afterwards using basic results from convex analysis.

Lemma 1. *It holds that*

$$V^{k+1} \leq V^k - \rho\|r^{k+1}\|_2^2 - \rho\|B(z^{k+1} - z^k)\|_2^2.$$

This states that V^k decreases in each iteration by an amount that depends on the norm of the residual and the change in z. Therefore it holds that $V^k \leq V^0$ for every k and because of that y^k and Bz^k are bounded. By iterating the inequality we get that

$$\rho \sum_{k=0}^{\infty} \|r^{k+1}\|_2^2 + \|B(z^{k+1} - z^k)\|_2^2 \leq V^0,$$

which implies our wanted result that $r^k \to 0$ and $B(z^{k+1} - z^k) \to 0$ as $k \to \infty$. Therefore the **residual convergence** stated in Theorem A.0.1 holds.

Lemma 2. *It holds that*

$$p^{k+1} - p^* \leq -(y^{k+1})^T r^{k+1} - \rho(B(z^{k+1} - z^k))^T(-r^{k+1} + B(z^{k+1} - z^*)).$$

The righthand side converges to zero as $k \to \infty$, because $B(z^{k+1} - z^*)$ is bounded and we already saw that $r^{k+1} \to 0$ and $B(z^{k+1} - z^k) \to 0$ for $k \to \infty$.

Lemma 3. *It holds that*

$$p^* - p^{k+1} \leq y^{*T} r^{k+1}.$$

Here again the righthand side goes to zero since $r^{k+1} \to 0$ for $k \to \infty$. Thus it follows from Lemma 2 and Lemma 3 that $\lim_{k\to\infty} p^k = p^*$ holds, i.e. the **objective convergence** (stated in Theorem A.0.1) is shown.

Proof of Lemma 3. As assumed in assumption 2 (x^*, z^*, y^*) is a saddle point for L_0. So it holds that

$$L_0(x^*, z^*, y^*) \leq L_0(x^{k+1}, z^{k+1}, y^*).$$

Using the primal feasibility $Ax^* + Bz^* = c$ we get the identity

$$L_0(x^*, z^*, y^*) = f(x^*) + g(z^*) + y^T(Ax^* + Bz^* - c) = f(x^*) + g(z^*) = p^*.$$

It follows immediatly that

$$p^* \leq p^{k+1} + y^{*T} r^{k+1}$$

holds, where $p^{k+1} := f(x^{k+1}) + g(z^{k+1})$. □

Proof of Lemma 2. In that proof we will need some basic theory about subdifferentials. Therefore I will provide first some important facts (without proofs). The proofs and more information about subdifferentials can be found in [Roc66], [BV08] and [Bal10].

Definition 6. Subdifferential
Let $f : \mathbb{R}^n \to (-\infty, +\infty]$ be a convex function. The essential domain of the function f is given by $\mathrm{dom} f =: \{x \in \mathbb{R}^n : f(x) < +\infty\}$. A vector $g \in \mathbb{R}^n$ is called a subgradient of f at the point x_0 if

$$f(x) \geq f(x_0) + g^T(x - x_0) \quad \forall x \in \mathbb{R}^n.$$

The SUBDIFFERENTIAL $\partial f(x_0)$ is then given by the set of all subgradients of f in x_0. A function f is called subdifferentiable at x if there exists at least one subgradient at x. The function is called subdifferentiable if it is subdifferentiable at all $x \in \mathrm{dom} f$. Note that if $x_0 \in \mathbb{R}^n \; dom f$, then $f(x_0) = +\infty$, so $\partial f(x_0) = \emptyset$.

Remark. The subdifferential $\partial f(x)$ is always a closed convex set, even if f is not convex. This follows from the fact that it is the intersection of an infinite set of halfspaces

$$\partial f(x) = \bigcap_{x \in dom f} \{g | f(x) \geq f(x_0) + g^T(x - x_0)\}.$$

Theorem A.1.1. *If f is convex and $x \in$ int dom$f$ then $\partial f(x)$ is nonempty and bounded.*

Theorem A.1.2. *A convex function f is differentiable at $x_0 \in$ int dom$f$ if and only if the subdifferential $\partial f(x_0)$ consists of only one point, which is the derivative at x_0 ($i.e. \partial f(x_0) = \{\nabla f(x_0)\}$).*

Remark. Therefore if f is convex and differentiable, its gradient at x_0 is a subgradient. But a subgradient can exist even if f is not differentiable at x_0.

Example. Let $f(x) = |x|$. For $x_0 \neq 0$ the subgradient is unique: $\partial f(x_0) = \{+1\}$ for $x_0 > 0$ and $\partial f(x_0) = \{-1\}$ for $x_0 < 0$. Therefore it is differentiable in those points. At $x_0 = 0$ the subdifferential is defined by the inequality $|x| \geq g^T x$ for all x, which is satisfied if and only if $g \in [-1, 1]$. Therefore we get $\partial f(0) = [-1, 1]$ and so f is not differentiable in 0.

Theorem A.1.3. *A point x^* is a minimizer of a convex function f if and only if f is subdifferentiable at $x^*$and $0 \in \partial f(x^*)$, i.e. $g = 0$ is a subgradient of f at x^*. This follows directly because in that case $f(x) \geq f(x^*)$ for all $x \in$ dom$f$. If f is differentiable at x^* the condition reduces to $\nabla f(x^*) = 0$.*

Corollary 3. *It holds that*

1. $\partial(\alpha f)(x) = \alpha(\partial f(x))$ *for* $\alpha \geq 0$*;*
2. $\partial f(x) = \partial f_1(x) + \cdots + \partial f_m(x)$ *if* $f = f_1 + \ldots f_m$*, where* $f_1, \ldots, f_m$ *are convex functions;*
3. $\partial h(x) = A^T \partial f(Ax + b)$ *if* $h(x) = f(Ax + b)$ *and* f *is convex.*

Theorem A.1.4. ***Moreau-Rockafeller***
Let $f, g : \mathbb{R}^n \to (-\infty, +\infty]$ be convex functions. Then

$$\partial f(x_0) + \partial g(x_0) \subset \partial(f+g)(x_0)$$

holds for every $x_0 \in \mathbb{R}^n$. If int dom$f \bigcup$dom $g \neq \emptyset$ *it also holds that*

$$\partial(f+g)(x_0) \subset \partial f(x_0) + \partial g(x_0)$$

for every $x_0 \in \mathbb{R}^n$.

Remark. As stated in [BPC$^+$11] f and g are subdifferentiable, because we know from assumption 1 that they are closed, proper and convex. The augmented Lagrangian L_ρ is the sum of subdifferentiable and differentiable functions, therefore it is subdifferentiable too.

By definition x^{k+1} minimizes $L_\rho(x, z^k, y^k)$. Combining the optimality condition in Theorem A.1.3 with the calculus rules in Corollary 3 and the result about differentiability in Theorem A.1.2, we get

$$0 \in \delta L_\rho(x^{k+1}, z^k, y^k) = \delta f(x^{k+1}) + A^T y^k + \rho A^T (Ax^{k+1} + Bz^k - c).$$

Since $y^k := y^{k+1} - \rho r^{k+1}$ the equation can be rearranged as

$$0 \in \delta f(x^{k+1}) + A^T (y^{k+1} - \rho B(z^{k+1} - z^k)).$$

Using again Theorem A.1.3, this implies that x^{k+1} minimizes

$$f(x^{k+1}) + (y^{k+1} - \rho B(z^{k+1} - z^k))Ax.$$

One can show with a similar argument that z^{k+1} minimizes $g(z) + y^{(k+1)T} Bz$. It follows immediatly that

$$f(x^{k+1}) + (y^{k+1} - \rho B(z^{k+1} - z^k))^T Ax^{k+1} \leq f(x^*) + (y^{k+1} - \rho B(z^{k+1} - z^k))^T Ax^*$$

and

$$g(z^{k+1}) + y^{(k+1)T} Bz^{k+1} \leq g(z^*) + y^{(k+1)T} Bz^*.$$

Adding those two inequalities we receive

$$p^{k+1} - p^* \leq y^{(k+1)T} A(x^* - x^{k+1}) + y^{(k+1)T} B(z^* - z^{k+1}) - \rho B(z^{k+1} - z^k)^T A(x^* - x^{k+1}).$$

Using $Ax^* + Bz^* = c$, it can be rewritten to

$$p^{k+1} - p^* \leq -y^{(k+1)T} r^{k+1} - \rho(B(z^{k+1} - z^k))^T(-r^{k+1} + B(z^{k+1} - z^*)).$$

□

Proof of Lemma 1. Adding the inequalities of Lemma 2 and 3 and multiplying by 2 we get that

$$2(y^{k+1}-y^*)^T r^{k+1} - 2\rho(B(z^{k+1}-z^k))^T r^{k+1} + 2\rho(B(z^{k+1}-z^k))^T(B(z^{k+1}-z^*)) \leq 0. \quad \text{(A.1)}$$

To prove the wanted inequality we have to do some technical transformations on (A.1). First we just rewrite the first term $2(y^{k+1}-y^*)^T r^{k+1}$. Since $y^{k+1} = y^k + \rho r^{k+1}$ we receive

$$2(y^k-y^*)^T r^{k+1} + 2\rho\|r^{k+1}\|_2^2,$$

substituting $r^{k+1} = \frac{1}{\rho}(y^{k+1}-y^k)$ and splitting the second term in two addends, it yields

$$\frac{2}{\rho}(y^k-y^{*T})(y^{k+1}-y^k) + \frac{1}{\rho}\|y^{k+1}-y^k\|_2^2 + \rho\|r^{k+1}\|_2^2.$$

Treating just the first two terms and using the identity $y^{k+1}-y^k = (y^{k+1}-y^*)-(y^k-y^*)$ in the second addend we get

$$\begin{aligned}
&2(y^k-y^{*T})(y^{k+1}-y^k) + \|y^{k+1}-y^k\|_2^2 \\
=&2(y^k-y^*)^T(y^{k+1}-y^k) + [(y^{k+1}-y^*)-(y^k-y^*)]^T(y^{k+1}-y^k) \\
=&(y^{k+1}-y^k)[(y^k-y^*)^T + (y^{k+1}-y^*)^T] \\
=&[(y^k-y^*)+(y^{k+1}-y^*)][(y^k-y^*)+(y^{k+1}-y^*)]^T \\
=&\|y^{k+1}-y^*\|_2^2 - \|y^k-y^*\|_2^2.
\end{aligned}$$

Therefore the inequality (A.1) can be rewritten as

$$\frac{1}{\rho}\left(\|y^{k+1}-y^*\|_2^2 - \|y^k-y^*\|_2^2\right) + \rho\|r^{k+1}\|_2^2 - 2\rho(B(z^{k+1}-z^k))^T r^{k+1} + 2\rho(B(z^{k+1}-z^k))^T(B(z^{k+1}-z^*)) \leq 0.$$

We now rewrite the last three terms

$$\rho\|r^{k+1}\|_2^2 - 2\rho(B(z^{k+1}-z^k))^T r^{k+1} + 2\rho(B(z^{k+1}-z^k))^T\left(B(z^{k+1}-z^*)\right). \quad \text{(A.2)}$$

Completing the square yields to

$$\rho\|r^{k+1} - B(z^{k+1}-z^k)\|_2^2 - \rho\|B(z^{k+1}-z^k)\|_2^2 + 2\rho(B(z^{k+1}-z^k))^T\left(B(z^{k+1}-z^*)\right).$$

Using now the identity $(z^{k+1}-z^*) = (z^{k+1}-z^k)+(z^k-z^*)$ leads to

$$\rho\|r^{k+1} - B(z^{k+1}-z^k)\|_2^2 + \rho\|B(z^{k+1}-z^k)\|_2^2 + 2\rho(B(z^{k+1}-z^k))^T\left(B(z^k-z^*)\right).$$

Splitting the last term and using the identity $(z^{k+1}-z^k) = (z^{k+1}-z^*)-(z^k-z^*)$ we finally receive for the last three terms (A.2)

$$\rho\|r^{k+1} - B(z^{k+1}-z^k)\|_2^2 + \rho\left(\|B(z^{k+1}-z^*)\|_2^2 - \|B(z^k-z^*)\|_2^2\right).$$

Therefore the inequality (A.1) can be rewritten as

$$\frac{1}{\rho}\left(\|y^{k+1}-y^*\|_2^2-\|y^k-y^*\|_2^2\right)+\rho\|r^{k+1}-B(z^{k+1}-z^k)\|_2^2+\rho\left(\|B(z^{k+1}-z^*)\|_2^2-\|B(z^k-z^*)\|_2^2\right)\leq 0,$$

which is equivalent to

$$V^k-V^{k+1}\geq\rho\|r^{k+1}-B(z^{k+1}-z^k)\|_2^2. \tag{A.3}$$

To show the wanted inequality

$$V^{k+1}\leq V^k-\rho\|r^{k+1}\|_2^2-\rho\|Bz^{k+1}-z^k\|_2^2$$

we have to check if the middle term of the square

$$\rho\|r^{k+1}-B(z^{k+1}-z^k)\|_2^2=\rho\|r^{k+1}\|_2^2-2\rho r^{(k+1)T}(B(z^{k+1}-z^k))+\rho\|B(z^{k+1}-z^k)\|_2^2$$

is negative. Recall from the proof of Lemma 2 that z^{k+1} minimizes $g(z)+y^{(k+1)T}Bz$ and similarly z^k minimizes $g(z)+y^{kT}Bz$. Therefore we can sum up the two inequalities

$$g(z^{k+1})+y^{(k+1)T}Bz^{k+1}\leq g(z^k)+y^{(k+1)T}Bz^k$$

and

$$g(z^k)+y^{kT}Bz^k\leq g(z^{k+1})+y^{kT}Bz^{k+1}$$

to get that

$$(y^{k+1}-y^k)^T(B(z^{k+1}-z^k))\leq 0.$$

Using that $y^{k+1}-y^k=\rho r^{k+1}$ yields the desired result

$$\rho r^{(k+1)T}(B(z^{k+1}-z^k))\leq 0.$$

Therefore we can follow from (A.3) that

$$V^{k+1}\leq V^k-\rho\|r^{k+1}\|_2^2-\rho\|Bz^{k+1}-z^k\|_2^2$$

holds. □

Bibliography

[AT90] L. Ambrosio and V.M. Tortorelli. Approximation of functionals depending on jumps by elliptic functionals via γ-convergence. *Communications on Pure and Applied Mathematics*, 43:999 – 1036, 1990.

[Bal10] Erik. J Balder. On subdifferential calculus. LNMB Ph.D. course "Convex Analysis for Optimization", 2010.

[BCC+11] L. Bar, T. Chan, G. Chung, M. Jung, N. Kiryati, R. Mohieddine, N. Sochen, and L. Vese. Mumford and shah model and its applications to image segmentation and image restoration. *Handbook of Mathematical Methods in Imaging*, pages 1095 – 1157, 2011.

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on pattern analysis and machine intelligence*, 26, 2004.

[BPC+11] Stephen Boyd, Neal Parkih, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundation and Trends in Machine Learning*, 3, 2011.

[Bre13] Anna Breger. Mathematische bildverarbeitung mit funktionen beschraenkter variation. Bachelor thesis, 2013.

[BT09] E. Bae and X.-C. Tai. Graph cut optimization for the piecewise constant level set method applied to multiphase image segmentation. *Springer - Verlag Berlin Heidelberg*, pages 1–13, 2009.

[BV99] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 1999.

[BV08] S. Boyd and L. Vandenberghe. Subgradients. Notes for EE364b, Stanford University, 2008.

[BVZ14] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Cornell University*, 2014.

[CCZ] X. Cai, R. Chan, and T. Zeng. Image segmentation by convex approximation of the mumford - shah model.

[CEN04] T. Chan, S. Esedoglu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. 2004.

[Cha99] A. Chambolle. Finite-differences discretizations of the mumford-shah functional. *M2AN*, 33(2), 1999.

[CKG97] V. Caselles, R. Kimmel, and G.Sapiro. Geodesic active contours. *International Journal of Computer Vision*, pages 61–79, 1997.

[CSV99] T. Chan, B. Sandberg, and L. Vese. Active contours without edges for vector-valued images. *Journal of Visual Communication and Image Representation*, 1999.

[CV99] T. Chan and L. Vese. An active contour model without edges. *Springer - Verlag Berlin Heidelberg*, 1999.

[CV00] T. Chan and L. Vese. An efficient variational multiphase motion for the mumford - shah segmentation model. In *Conference Record of the 34th Asilomar conference on signals, systems and computers*, 2000.

[CV01] T. Chan and L. Vese. A level set algorithm for minimizing the mumford - shah functional in image processing. *IEEE Proceedings on Variational and Level Set Methods in Computer Vision*, 2001.

[DK01] P.L. Davies and A. Kovac. Local extremes, runs, strings and multiresolution. *The Annals of Statistics*, 29(1):1–65, 2001.

[DR15] R.E.H. Tappenden D.P. Robinson. A flexible admm algorithm for big data applications. 2015.

[Ess09] Ernie Esser. Applications of lagrangian-based alternating direction methods and connections to split bregman. 2009.

[FD56] L.R. Ford and D.R.Fulkerson. Maximal flow through a network. 1956.

[FS98] H.G. Feichtinger and T. Strohmer. *Gabor Analysis and Algorithms - Theory and Applications*. Springer Science + Business Media New York, 1998.

[GM13] S. Gupta and S.G. Mazumdar. Sobel edge detection algorithm. *International Journal of Computer Science and Management Research*, 2(2), February 2013.

[Gun02] S. R. Gunn. Edge detection error in the discrete laplacian of gaussian. *Image, Speech and Intelligent Systems Group*, 2002.

[HB13] A. Kumar H.S. Bhadauria, A. Singh. Comparison between various edge detection methods on satellite image. *International Journal of Emerging Technology and Advanced Engineering*, 3(6), 2013.

[HL13] M. Hong and Z. Luo. On the linear convergence of alternatin direction method of multipliers. *National Science Foundation*, 2013.

[HS08] H. Hirschmüller and D. Scharstein. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2), 2008.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321 – 331, 1988.

[MS85] D. Mumford and J. Shah. Boundary detection by minimizing functionals. *IEEE Proceedings on Computer Vision and Pattern Recognition, San Francisco, California*, 1985.

[MS89] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, XLII:577 – 685, 1989.

[MS95] J.M. Morel and S. Solimini. *Variational Methods in Image Segmentation*. Birkhäuser, 1995.

[MS98] R. Boyle M. Sonka, V. Hlavac. *Image Processing, Analysis and Machine Vision*. Cengage Learning Emea, 2 edition, 1998.

[MSV94] R. Malladi, J.A. Sethia, and B. C. Vermuri. Evolutionary fronts for topology-independet shape modeling and recovery. *ECCV '94 Proceedings of the third European confernce on Computer vision*, 1:3–13, 1994.

[MT95] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. *Fifth International Conferenc on Computer Vision (ICCV), Cambridge*, pages 840 – 845, 1995.

[NS93] N.R.Pal and S.K.Pal. A review on image segmentation. *Pattern Recoqnition*, 26(9), 1993.

[OS88] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulation. *Journal of Computational Physics*, 79, 1988.

[Pot52] R. Potts. Some generalized order-disorder transofrmations. *Proceedings of the Cembridge Philosophical Society*, 48:106, 1952.

[RKS13] Rashmi, M. Kumar, and R. Saxena. Algorithm and technique on various edge detection: a survey. *Signal and Image Processing: An International Journal(SIPIJ)*, 4(3), 2013.

[Roc66] R.T. Rockafellar. *Characterization of the Subdifferentials of Convex Functions*, volume 17. Pacific Journal of Mathematics, 1966.

[Rus11] J. C. Russ. *The Image Processing Handbook*. CRC Press, 6 edition, 2011.

[SaKK10] S.S.Al-amri, N.V. Kalyankar, and S.D. Khamitkar. Image segmentation by using threshold techniques. *Journal of computing*, 2, 2010.

[SK11] H. Schwarz and N. Köckler. *Numerische Mathematik*, volume 8. Springer Fachmedien Wiesbaden GmbH, 2011.

[Sob14] I. Sobel. *History and Definition of the Sobel Operator*, 2014.

[SR09] N. Senthilkumaran and R. Rajesh. Edge detection techniques for image segmentation - a survey of soft computing approaches. *International Journal of Recent Trends in Engineering*, 1(2), 2009.

[SW14] M. Storath and A. Weinmann. Fast partitioning of vector-valued image. *SIAM J. Imaging sciences*, 7:1826–1852, 2014.

[uTN12] U. Krause und T. Nesemann. *Differenzengleichungen und diskrete dynamische Systeme*. De Gruyter Studium, 2012.

[VC02] L. Vese and T. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[WBAW] B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang. An admm algorithm for a class of total variation regularized estimation problems.

[Wee96] A.R. Weeks. *Fundamentals of Electronic Image Processing*. SPIE/IEEE Series on Imaging Science and Engineering, 1996.

[WS06] Y.-X. Yuan W. Sun. *Optimization Theory and Methods*. Springer Science + Business Media, LLC, 2006.

[WWLK] G. Winkler, O. Wittich, V. Liebscher, and A. Kempe. *Don't Shed Tears over Breaks*. GSF - National Research Center for Environment and Health.

[ZL96] G. Zhao and Z. Liu. A line search method in lagrangian relaxation algorithms. *National University of Singapore*, 1996.

Curriculum Vitae of Anna Breger, BSc

E-mail: anna.breger@univie.ac.at
Address: Berggasse 11/6, A-1090 Vienna, Austria
Telephone: (+43) 699 1111 8323
Born: October 26, 1990
Citizenship: Austria (EU)

University education and degrees

Since 09/2013 — University of Vienna, Austria (Department of Mathematics)
Master of Science Program for Applied Mathematics
Main focus on image processing, numerical mathematics and different programming languages.
Master's Thesis
On image segmentation and its application in clinical retinal analysis

09/2009 - 09/2013 — University of Vienna, Austria (Department of Mathematics)
Bachelor of Science (awarded with distinction) (BSc)
Bachelor thesis 2
Mathematical Image Processing on functions with bounded variation
Bachelor thesis 1
The Jacobi Method
(Numerical method to solve Eigenvalue problems)

Additional Qualifications

06/2015 — Graduation in Violin Pedagogy (teaching certification)
Joseph Haydn Conservatory, Eisenstadt (Austria)

Semesters abroad

Summer, 2012 — University of Texas at Austin, Dept. of Mathematics
01/2013 - 06/2013 — University of Leeds, Leeds, United Kingdom

Professional positions

09/ 2013 - 09/2014	Tutor, University of Vienna, Dept. of Mathematics Teaching how to use the software LaTex and Mathematica.
07/2014	Internship at ProFactor GmbH, Vienna Development of a Matlab toolbox for creating various feature - vectors to classify errors in a given image.

Awards

2012	Merit Scholarship for bachelor studies, University of Vienna
2014	Merit Scholarship for master studies, University of Vienna
2015	Nominated for the high potential programme "Naturtalente" of the University of Vienna Successful completion of expertise workshops for management, leadership and personality skills.

Language skills

German	native
English	fluent
Spanish	basic

Additional computer skills

Microsoft Office (Word, Excel, PowerPoint, Access)

LaTex, Mathematica, Amira (image processing)

JAVA, MATLAB and C++ programming