Realtime Object Separation for Industrial Glass Sorting Systems

Christian Hartbauer

18.12.2012

Abstract

Industrial glass recycling often includes optical sorting machines, which are used to sort a stream of glass cullet according to its color. A crucial problem for the performance of such systems are connected objects which lead to wrong results in the sorting process. The goal of this thesis is to develop a solution to this problem. We formulate this problem as a partitioning problem according to the Ising model, for two partitions, or the Potts model, for k partitions. Furthermore we define approximations, such that both models are defined in a spatially continuous domain and can be solved with variational methods. In order to find global solutions we use a simple convex relaxation approach. An other main aspect of this work is, the requirement, that the algorithms have to be executed in real-time. Thus we provide a fast numerical approach, called the primal dual algorithm, which allows to efficiently find a global solution of the problem. To gain real-time performance we provide a detailed overview in efficient initializations, convergence criteria and show how parts of the computations can be excluded earlier. At last we illustrate the great advantages of our models by comparing them to the well known watershed segmentation algorithm.

Kurzfassung

In der industriellen Glaswiederaufbereitung werden optische Sortiersysteme eingesetzt, welche große Mengen Bruchglas anhand ihrer Farbe sortieren. Eines der größten Probleme solcher Sortiersysteme sind zusammenhängende Objekte, da diese zu falschen Entscheidungen im Sortierprozess führen können. Das Ziel dieser Arbeit ist es, eine Lösung für dieses Problem zu entwickeln. Wir formulieren diese Aufgabe als Partitionsproblem entsprechend des Ising Modells für zwei Partitionen und des Potts Modells für k Partitionen. Des weiteren werden Approximationen zu diesen Modellen formuliert um sie in einer räumlich kontinuierlichen Domäne zu definieren. Um eine globale Lösung zu finden verwenden wir eine einfache konvexe Relaxierung. Anhand dieser Erweiterungen kann eine globale Lösung mit Hilfe einer Variationsmethode gefunden werden. Ein weiterer wichtiger Aspekt dieser Arbeit ist, dass die gefundenen Algorithmen in Echtzeit einsetzbar sein müssen. Daher verwenden wir einen numerischen Ansatz zum Lösen des relaxierten Problems, genannt Primal-Dual Algorithmus, welcher sehr effizient eingesetzt werden kann. Des Weiteren zeigen wir, wie man mit effizienten Initialisierungen, guten Konvergenzkriterien und dem Entfernen von bereits konvergierten Partitionen, das Ziel einer Echtzeitberechnung erreichen kann. Zum Schluss werden die großen Vorteile unserer Modelle gezeigt, indem sie mit dem bereits bekannten und weit verbreiteten Watershed Segmentation Algorithmus verglichen werden.

Acknowledgements

First of all I want to thank my family. Specially my wife, who has covered my back in the time I was working on my masters thesis. And I want to thank my daughter, who has finally accepted that I had to do some work at home. Thanks to Prof. Horst Bischof and Ass. Prof. Thomas Pock for supervising my Master Thesis.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

(Unterschrift)

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date (signature)

Contents

1 Introduction **5**

1.1 Glass Recycling 5

1.1.1 Optical Glas Sorting System - REDWAVE 5

1.2 Problem Description 6

1.2.1 Object separation - A Segmentation Problem 6

1.2.2 Provided Representations of the Objects 7

1.3 Proposed Algorithms 7

1.3.1 Watershed Segmentation 7

1.3.2 Ising/Potts Model 8

2 Related Work **13**

2.1 Image Segmentation 13

2.2 Edge based Image Segmentation 14

2.2.1 Snakes 14

2.2.2 Level Sets 15

2.2.3 Geodesic Active Contours 17

2.2.4 Global Minimization of Geodesic Active Contour . . . 18

2.3 Region Based Image Segmentation 18

2.3.1 Mumford-Shah 19

2.3.2 Chan-Vese 20

2.3.3 Multiphase Segmentation with Level Set functions . . 22

2.3.4 Global Minimization of the Active Contour Model based on Mumford-Shah . . . 24
2.4 Object Separation . . . 25
2.4.1 Minimum Entropy Segmentation . . . 26
2.4.2 Clump splitting via bottleneck detection . . . 26
2.4.3 Segmenting clustered slender-particles . . . 27

3 Proposed Models **32**
3.1 Watershed Segmentation . . . 32
3.1.1 Distance Transform . . . 32
3.1.2 Watershed Transformation . . . 34
3.1.3 Watershed Segmentation - Algorithm . . . 36
3.2 Computing Minimal Partitions . . . 37
3.3 Preliminaries . . . 38
3.3.1 Discrete Settings . . . 41
3.4 Ising Model . . . 43
3.4.1 Convex Relaxation . . . 44
3.4.2 Discrete Ising Model . . . 45
3.4.3 Primal Dual Algorithm . . . 45
3.4.4 Ising Primal Dual . . . 47
3.5 Potts Model . . . 50
3.5.1 Label Cost . . . 50
3.5.2 Discretization . . . 51
3.5.3 Potts Primal Dual . . . 51

4 Implementation Details **56**
4.1 GPU . . . 56
4.1.1 CUDA . . . 56
4.1.2 GPU Architecture . . . 59
4.1.3 Memory Space . . . 59
4.2 Implementation . . . 61

4.2.1 Watershed Segmentation 61
4.2.2 Ising Model 61
4.2.3 Potts Model 62

5 Experimental Results **65**
5.1 Experimental Environment 65
5.1.1 Evaluation Metrics 68
5.1.2 Quality of Input Images 69
5.2 Watershed Segmentation 69
5.2.1 Determining Distance Transformation 70
5.2.2 Enhanced Watershed Algorithm 70
5.3 Potts Model 72
5.3.1 Segmentation Quality 73
5.3.2 Performance Measure 74
5.3.3 Summary 78
5.4 Ising Model 79
5.4.1 Determining λ 79
5.4.2 Convergence Criterion 79

6 Conclusion and Outlook **87**
6.1 Conclusion 87
6.2 Outlook 88

Chapter 1

Introduction

1.1 Glass Recycling

In order to produce new usable products out of waste glass, the recycled glass has to be separated by color. This separation must be done, because different colors of glass are usually chemically incompatible. In most countries the human separation of glass with recycling containers has not the quality to go directly to reproduction. A standard approach in industrial glass recycling plants is to use optical sorting machines to finish the separation process. Another important component for the use of optical sorting machines is the separation of extraneous material out of the glass. There are several materials which disturb the recycling process such as ceramics, stones and porcelain (called CSP) which can be separated with optical sorting machines.

1.1.1 Optical Glas Sorting System - REDWAVE

REDWAVE is a trademark of BT-Wolfgang Binder, which are sensor based sorting machines used to separate incoming stream of matter, based on different characteristics of the material. BT-Anlagenbau is the provider of the software for all REDWAVE systems working with RGB cameras. Figure 1.1 shows the REDWAVE C which is a RGB camera based machine

used in glass recycling plants. This machine will be the reference machine for our object separation approach. As mentioned in Figure 1.1 the camera of such a machine only captures the transmitted light of the objects, so only light through transparent objects, like glass, reaches the camera sensor. Therefore materials like CSP can be separated because of the absence of light.

1.2 Problem Description

The task of this work is, to produce an algorithm which is able to refine the process of labeling. To separate extraneous material or wrong sorts of glass out of a stream of material, the machine has to label the object as a sort of glass or as extraneous material. While the actual software has many good algorithms to find out the label of each object, if the objects are captured separately, there is only one rather simple algorithm which tries to separate connected objects, so that the labeling algorithms can work correctly. Separately captured objects means, that at the time of taking the picture, the object is not connected to any other object.

1.2.1 Object separation - A Segmentation Problem

As mentioned above it is crucial to separate connecting objects. Object separation can be seen as a sub-domain of image segmentation, which is a broadly studied problem in computer vision. A summary of those works will be given in chapter 2. A segmentation problem is the problem of splitting the image in partitions, such that every pixel in the image is assigned to these partitions. In our case each pixel can only be assigned to one partition (mechanically useful). Therefore we can define a continuous setting similar to [11], where $\{E_i\}_{i=0}^{k}$ is the partition of an open set $\Omega \subset \mathbb{R}^2$ into k sets where $E_i \bigcap E_j = 0 \quad \forall i \neq j$ such that $\bigcup_{i=0}^{k} E_i = \Omega$. If the partition is optimal, every set E_i corresponds to one and only one separated object

$\forall i \geq 1$ and E_0 is the background set.

1.2.2 Provided Representations of the Objects

The software provides several representations of the objects which can be used for the segmentation process. The segmentation algorithm will have access to each connected object separately and will get several additional information of the object. Figure 1.2 shows the provided information which are, the original camera picture, the binary object mask which separates the object from the background and the classified image which shows the color class of each pixel of the object. Further more there will be additional information, like which class of color has to be separated mechanically, which will be used in our proposed algorithms.

1.3 Proposed Algorithms

With regard to the requirement of real-time processing and high quality we have chosen to implement three algorithms. In this subsection we show a short survey through the algorithms, which will be explained in more detail in chapter 3.

1.3.1 Watershed Segmentation

The first algorithm concentrates on the shape of a broken glass object. Our assumption is, that one piece of broken glass is nearly convex and connecting objects of glass are often concave. This observation brings us to the first proposed algorithm. The watershed segmentation is a well known algorithm for segmentation of connecting objects proposed in many papers [15, 3, 2]. A distance transform of a binary object mask, can be seen as a topographic representation shown in Figure 1.3. This representation has three notions: minima, catchment basin and watershed line as defined in [15]. Such an topographic representation can be interpreted as a landscape, where rain

is falling. The minima are the lowest parts of the landscape, which are first filled with water. If the rain doesn't stop, the catchment basin of each minima gets filled and only the watershed lines between the basins aren't under water. This watershed lines define the end of the influence of one minima. In our case, each minima represents a object which should be separated and the watershed line represents the border of the object. This algorithms tend to be very fast, but have problems with over-segementation because of the existence of wrong minima's.

1.3.2 Ising/Potts Model

Furthermore most of the pixels of an object are classified correct, such that an algorithm which tries to segment an image based on this information should give good results. This leads us to the minimal partitions problem, which we will explain in Chapter 3. The discrete analogue of this model is called Ising/Potts model, which is the basic principle of the two other proposed object separation algorithms. The Ising model[22] is a mathematical model for ferromagnetism in statistical mechanics, which consists of only two states. The Potts model[31] is the generalization of the Ising model to more than two states. This models can be used to find a two-label image segmentation (Ising), or a multi-label segmentation (Potts). In our case the multi-label segmentation will operate on the provided classified image, where each pixel is assigned to a color class. The two-label segmentation will work on a simplified classified image, where each pixel is assigned to one of two labels. As the Potts Model is known to be NP hard, there have been many papers who have tried to approximately minimize this model. Some try to tackle the discrete problem, like Boykov et. al. [4] who is using binary optimizations via α-expansion. But those works tend to exhibit metrication errors. Other works have tried to work on continuous domains, like Chan and Vese [14], but only found local solutions. We will define both models in a spatially continuous domain, similar to [29], such that the models can

be solved with variational methods. Furthermore we provide a convex relaxation, as defined in Zach et. al. [47], which gives us the opportunity to always find a global solution.

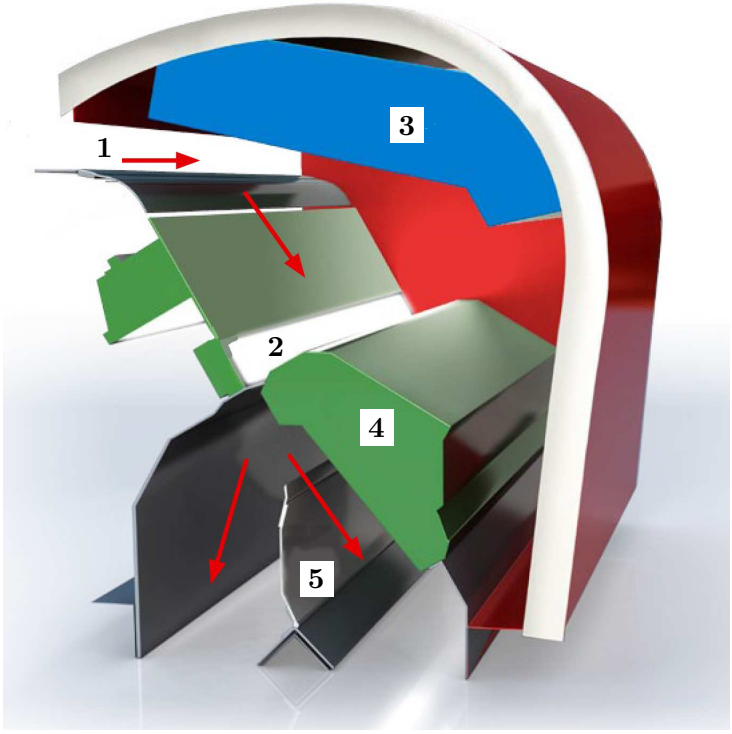
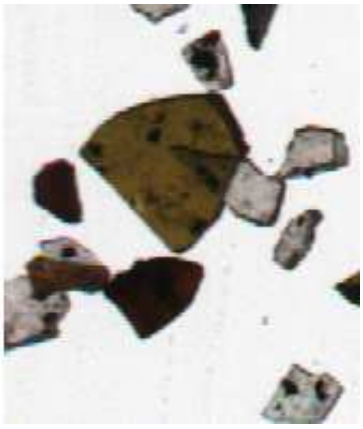
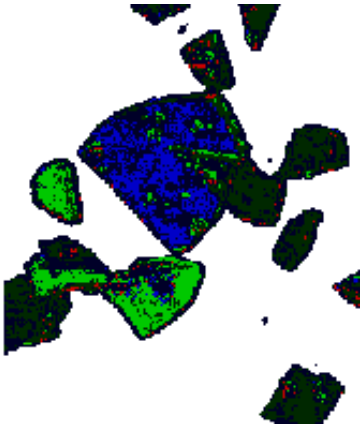


Figure 1.1: Redwave C. A vibration feeder (1) provides a constant flow of infeed material evenly spread over the entire sorting width. The glass slides over a light source (2). Opposite to the light source is the sensor unit (3), containing the RGB camera, which captures the transmitted light of the objects. Short after the objects where captured, an ejection unit (4) separates the extraneous material out of the material stream. The ejection unit contains a tight cluster of compressed air valves, which are activated if a material has to be sorted out. The divider plate (5) is the boundary between the two material streams. Publication of this picture is permitted by BT-Anlagenbau.



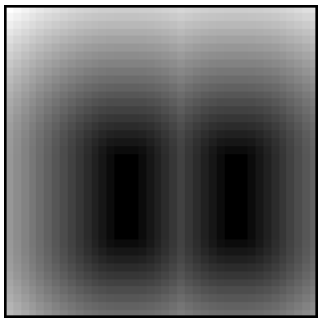
(a) Camera image



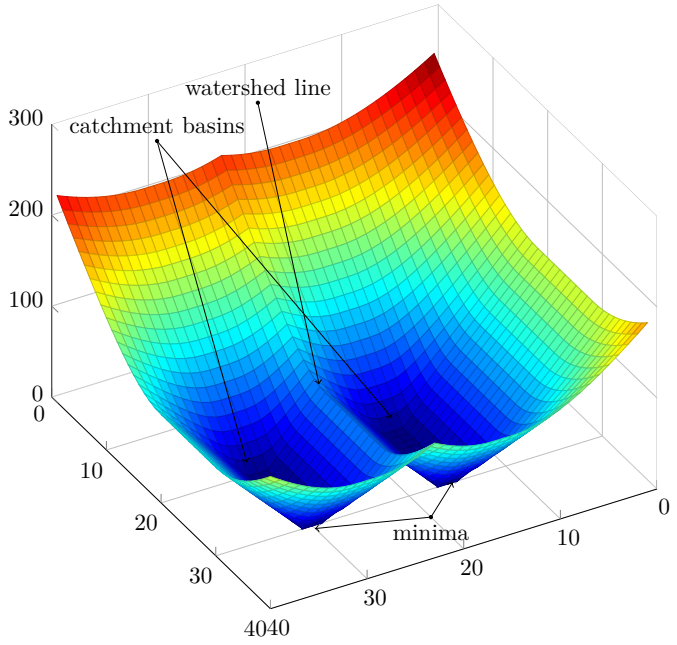
(b) Classified data

(c) Mask image

Figure 1.2: This figure shows the provided representations of a connected object, which can be used for object separation. (a) Is the original picture taken with a linescan-camera. (b) A pseudocolor representation of the classified data. (c) A mask which separates background from the object.



(a) Distance transform



(b) Topographic surface

Figure 1.3: (a) Shows the distance transformation of two objects. (b) Is a three dimensional surface representation of the 2 dimensional distance transformation. Where the z-axis is the value of the distance transformation. The catchment basins define the area of influence of each minima and the watershed line defines the border of the two catchment basins.

Chapter 2

Related Work

This chapter will focus on recent work on image segmentation and object separation. Specially work which could have been used to address our problem of object separation will be presented.

2.1 Image Segmentation

Image Segmentation is one of the main tasks of computer vision. From the early stages of computer vision, there have been methods developed, which try to divide the image in regions, which fulfill some predefined requirements. Most of the methods can be divided into two groups. The first group focuses on the edges of the regions. The theory behind this algorithms is to find the transitions from one region to the other, which should be inhomogeneous. This should result in the contour of the segmented regions. The other group focuses on the regions itself and tries to find homogeneous areas in the image. Other differences lie in finding a global or a local optimum based on the proposed model.

2.2 Edge based Image Segmentation

There are many works that concentrate on finding salient image contours. The first step in finding contours is often the edge detection, followed by linking the edges to a contour. Newer approaches are based on active contour models, such as geodesic active contour models or snakes.

2.2.1 Snakes

Kass et. al. [23] defined a snake as an energy minimizing spline, which is guided through external constraint forces and influenced by image forces, which pull it towards lines and edges. Therefore snakes are defined parametrically by a curve function $C(s) = (x(s), y(s))$ where $s \in [0, 1]$ can be any arbitrary parametrization. The energy function of this contour can be written as [23]

$$E_{snake} = \int_0^1 E_{int}C(s) + E_{image}C(s) + E_{con}C(s)\,\mathrm{d}s \tag{2.1}$$

Internal Energy

Where E_{int} is the internal contour energy and is defined as

$$E_{int} = \frac{\alpha(s)|C(s)'| + \beta(s)|C(s)''|}{2} \tag{2.2}$$

$\alpha(s)$ and $\beta(s)$ define the weighting of the smoothness constraints, which are the first and second order derivatives of the curve.

Image Energy

The image Energy E_{image} is defined through three terms.

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{2.3}$$

Where E_{line} is just the gray value of the image, such that the snake can be forced to be pushed to dark or light lines, depending on the sign of w_{line}. As

edge functional Kass et. al. [23] proposes $E_{edge} = -|\nabla I(x,y)|$ which forces the contour to regions with large image gradients. The third term, E_{term}, uses the curvature of the slightly smoothed image to find terminations of line segments and corners. Each of the terms can be weighted with their correspondign weights $w_{line}, w_{edge}, w_{term}$.

Constraint Energy

The last term, E_{con}, of the snake energy functional (2.1) is used to model user constraints into the contour.

Assets and Drawbacks

If the region of interest, which has to be separated by the snake, is approximately known, snakes can be a powerful tool to find contours. Also Kass et. al. [23] introduced an edge energy functional, which can be used to address edges from a longer distance, by using the theory of scale space. Therefore the functional was changed into $E_{edge} = -(G_\sigma * \nabla^2 I(x,y))^2$ where G is a Gaussian of standard deviation σ. But the problem with this functional is, that there has to be made a tradeoff between localisation of edges and the distance of which the snake is attracted to an edge. An other major drawback of snakes is, that they are defined parametrically. With such a definition the curve cannot change in topology. That means during the curve evolution process it is impossible to perform splitting or merging of the curve.

2.2.2 Level Sets

Osher and Sethian [27] introduced the level set method, a short overview of the method can be found in Osher et. al [26]. The method defines a

boundary Γ of an open region Ω by a smooth function $\varphi(x)$.

$$\varphi(x) \begin{cases} = 0 & \partial\Omega = \Gamma \\ < 0 & \forall x \notin \Omega \\ > 0 & \forall x \in \Omega \end{cases} \tag{2.4}$$

The function $\varphi(x)$ can be seen as a helper function to define a boundary of a region without a parametrization, as shown in Figure 2.1. The behavior of the boundary under a velocity field $\vec{v}$ can be computed by the PDE (Partial Differential Equation)

$$\frac{\partial \varphi}{\partial t} + \langle \vec{v}, \nabla \varphi \rangle = 0 \tag{2.5}$$

where $\langle . \rangle$ defines the scalar product. As there is only the normal component of $\vec{v}$ needed, we can define a force $F = \langle \vec{v}, \frac{\nabla \varphi}{|\nabla \varphi|} \rangle$ which is the normal component of $\vec{v}$. Now equation (2.5) becomes

$$\frac{\partial \varphi}{\partial t} + F|\nabla \varphi| = 0 \tag{2.6}$$

With such a level set formulation topological changes will be realized naturally by the level set function, which represents a great advantage to parametric models such as snakes.

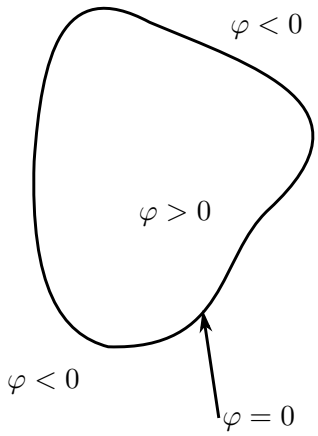


Figure 2.1: Level set function where φ defines the boundary of the region

2.2.3 Geodesic Active Contours

Caselles et. al. [8] introduced an energy minimizing function, which transformed the minimizing function of a snake into a problem of geodesic computation in a Riemannian space.

$$\min \int_0^1 g(|\nabla I(C(q))|)|C'(q)|\,\mathrm{d}q \tag{2.7}$$

Where g is a general edge detector with the property:

$$\lim_{z\to\infty} g(z) = 0$$

For example [8] chooses g as:

$$g(I) = \frac{1}{1+|\nabla(G_\sigma * I)|}$$

Where G_σ is a Gaussian with standard deviation σ.
It is remarkable that [8] only used the first order term of the internal energy of the snake, shown in Equation 2.2. Because they showed that a smooth curve can be achieved without a higher order term. In order to obtain a parameter free representation of the curve, a level set formulation of Equation 2.7 was represented. Therefore the contour $C(q)$ will be represented through the level set function $\varphi(x)$ as defined in section (2.2.2). Resulting in a PDE:

$$\frac{\partial\varphi}{\partial t} = |\nabla\varphi|\,\mathrm{div}\left(g(I)\frac{\nabla\varphi}{|\nabla\varphi|}\right) + cg(I)|\nabla\varphi| \tag{2.8}$$

Caselles et. al. [8] states, that the second term can be seen as an area constraint which adds a velocity according to the area enclosed by the contour. They show in their experimental results that $c = 0$ can be choosen and the model still converges, but with a slower motion.

Assets and Drawbacks

A problem with geodesic active contours is, that their energy functional is not convex, which results in finding local-optima depending on the initialization of the curve. Furthermore the globally optimal solution of the geodesic active contour is the empty set.

2.2.4 Global Minimization of Geodesic Active Contour

Many works concentrate on the circumstance that active contour models only find local minimizers. Such as Bresson et. al. [6], who proposed an global minimizer based on the famous ROF model [34].

$$E_{ROF}(u,\lambda) = \int_\Omega |\nabla u| \,\mathrm{d}x + \lambda \int_\Omega (u-f)^2 \,\mathrm{d}x \tag{2.9}$$

Where the first term denotes the Total Variation as shown in Section 3.3 and the second term is the quadratic data term. Bresson et. al. [6] changed the ROF model in order to get an global minimizer for the geodesic active contour model which is not the empty set.

$$E(u,\lambda) = \int_\Omega g(x)|\nabla u| \,\mathrm{d}x + \lambda \int_\Omega |u-f| \,\mathrm{d}x \tag{2.10}$$

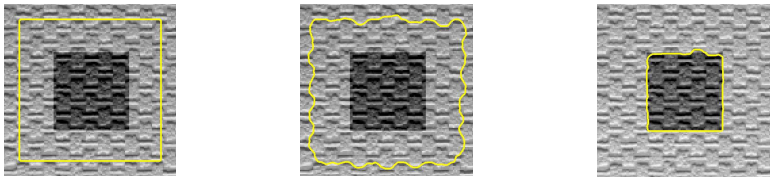
The first term denotes the weighted Total Variation, where the weighting function $g \in [0,1]$ gives the link between the proposed model and the geodesic active contour model. If g is an edge indicator function and $u = \mathbf{1}_{\Omega C}$ is a characteristic function of a closed set $\Omega_C \subset \Omega$ where C are the boundaries of Ω_C. A link between the weighted Total Variation and the geodesic active contour model(GAC) can be shown.

$$TV_g(u = \mathbf{1}_{\Omega C}) = \int_\Omega g(x)|\nabla \mathbf{1}_{\Omega C}| \,\mathrm{d}x = \int_C g(s) \,\mathrm{d}s = E_{GAC}(C) \tag{2.11}$$

Secondly the $L^2 - norm$ of the ROF model is changed to a $L^1 - norm$, which better preserves the contrast and the order, according to the feature size, in which the features disappear. Since this model provides a global minimum, standard calculus of variations can be used to solve it, see [6]. The advantage of a global model as it is presented in this section, is shown in Figure 2.2.

2.3 Region Based Image Segmentation

Early methods for region based image segmentation often include thresholding an image such that different intensities lead to different regions. For



(a) Local initialization (b) Local GAC (c) Global GAC

Figure 2.2: Global vs. local geodesic active contour models, reprinted from [6].

example see White and Rohrer [45] who used an adaptive threshold for optical character recognition, or Sahoo et. al [35] for an overview on thresholding techniques. More recent methods try to find an appropriate model for the segmented image and solve this model based on level set or variational methods.

2.3.1 Mumford-Shah

In their famous paper, Mumford-Shah [25] tried to find an approximation to the problem of image segmentation. Their approximation concentrates on two properties of a great number of images.

1. the image f varies smoothly within a region E_i
2. the image f varies discontinuously across most of the boundary Γ between different regions E_i, E_j

Where E_i are disjoint connected open subsets of a domain Ω and Γ are the boundaries of E_i inside Ω.

$$\Omega = E_1 \cup E_2 \cup \ldots \cup E_n \cup \Gamma \tag{2.12}$$

They defined an energy functional, which measures the degree of match between an image $f(x)$ and a segmentation, where the energy is smaller if the degree of match is higher:

$$E(u, \Gamma) = \mu^2 \int_\Omega (u - f)^2 \, \mathrm{d}x + \int_{\Omega \setminus \Gamma} |\nabla g|^2 \, \mathrm{d}x + \nu |\Gamma| \tag{2.13}$$

Where u is a differentiable function on $\bigcup_i^n E_i$. The three terms are measuring (left to right):

1. the similarity between u and f.
2. the smoothness of each region E_i
3. the length of the boundaries of the segmentation.

As stated in their paper [25], there are many drawbacks which lead to wrong segmentation, like textured objects, partially transparent objects or noisy images.

2.3.2 Chan-Vese

An other concept of active contours was represented by Chan and Vese [13], who changed the stopping function of an active contour, which is normally an edge function (see Section 2.2.3) into a stopping term based on the Mumford-Shah functional [25], see Equation (2.13). Which results in following engergy minimization problem:

$$\begin{aligned}\inf_{c_1,c_2,C} F(c_1,c_2,C) =& \mu Length(C) + \nu Area(C) \\ &+ \lambda_1 \int_{inside(C)} |f(x) - c_1|^2 \,\mathrm{d}x \\ &+ \lambda_2 \int_{outside(C)} |f(x) - c_2|^2 \,\mathrm{d}x\end{aligned}$$

Where the first two terms are the regularization terms, which penalize the length or the area of the contour C. The second two terms, which are the stopping terms, are penalizing the difference of the two image regions called $inside(C)$ and $outside(C)$ to their respective mean intensities. Again Chan and Vese [13], described their method through a level set forumulation where the contour C was replaced by the level set function $\varphi(x)$ according to Section 2.2.2. The function $inside(C)$ is changed by $\varphi(x) > 0$ and $outside(C)$

is represented by $\varphi(x) < 0$. A Heaviside function H and a one dimensional dirac measure δ is used to address the two regions and the contour:

$$H(z) = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ else} \end{cases}$$

$$\delta_0(z) = \frac{\mathrm{d}}{\mathrm{d}z} H(z)$$

With this functions the energy functional can be rewritten as:

$$\begin{aligned} \inf_{c_1,c_2,\varphi} F(c_1, c_2, \varphi) =& \mu \int_\Omega \delta(\varphi(x)) |\nabla \varphi(x)| \,\mathrm{d}x \\ &+ \nu \int_\Omega H(\varphi(x)) \,\mathrm{d}x \\ &+ \lambda_1 \int_\Omega |f(x) - c_1|^2 H(\varphi(x)) \,\mathrm{d}x \\ &+ \lambda_2 \int_\Omega |f(x) - c_2|^2 (1 - H(\varphi(x))) \,\mathrm{d}x \end{aligned} \tag{2.14}$$

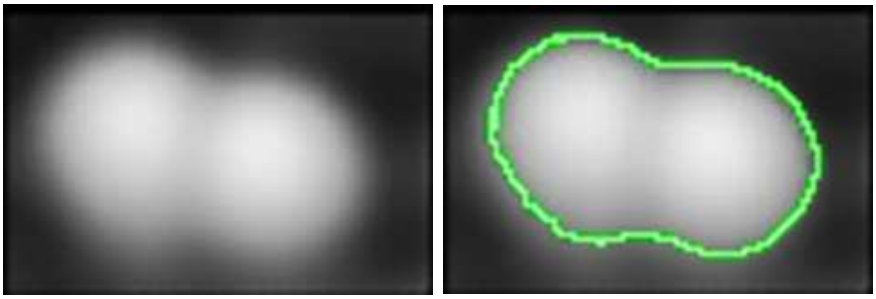
If the mean intensities of the two regions aren't known a priori, they can be easy calculated at each step of the curve evolution process by the following equations:

$$c_1 = \frac{\int_\Omega f(x) H(\varphi(x)) \,\mathrm{d}x}{\int_\Omega H(\varphi(x)) \,\mathrm{d}x}$$

$$c_2 = \frac{\int_\Omega f(x)(1 - H(\varphi(x))) \,\mathrm{d}x}{\int_\Omega (1 - H(\varphi(x))) \,\mathrm{d}x}$$

Assets and Drawbacks

The active contour model proposed by Chan and Vese has an advantage to the geodesic model shown in Section 2.2.3, because it can find objects with smooth boundaries as shown in Figure 2.3. But again a problem of this formulation is, that the formulation results in a non-convex minimization problem.



(a) Input image. (b) Result of Chan Vese algorithm.

Figure 2.3: Segmentation result of the Chan Vese algorithm, reprinted from [36]. It shows that the algorithm can find objects with smooth boundaries.

2.3.3 Multiphase Segmentation with Level Set functions

The algorithms proposed in Section 2.3.2 and Section 2.2.3 only separate two regions from each other, similar to our Ising model proposed in Section 3.4. Many papers concentrated on separating more regions from each other using level set functions to achieve results similar to the Potts model proposed in Section 3.5.

Multiphase Chan-Vese

Chan-Vese proposed an extension of their two region algorithm as shown in Section 2.3.2. They changed the energy function from Equation (2.14) to a multiphase version [41]:

$$\begin{aligned} F_n(c,\phi) = &\sum_{i=1}^{n} \int_\Omega (f-c_i)^2 \chi_i \,\mathrm{d}x \\ &+ \sum_{i=1}^{m} \nu \int_\Omega |\nabla H(\varphi_i)| \end{aligned} \tag{2.15}$$

Where $\phi = (\varphi_1, \ldots, \varphi_m)$ is a vector level set function with $m = \log n$ level set functions (see Section 2.2.2). And $H(\phi) = (H(\varphi_1), \ldots, H(\varphi_m))$ is a vector Heaviside function whose components are 0 or 1. The function χ_i is the characteristic function for the class i. With this formulation the authors

were able to define a n class segmentation with only $m = \log n$ level set functions, as shown in Figure 2.4. The Heaviside function can be seen as binary codification of the classes.

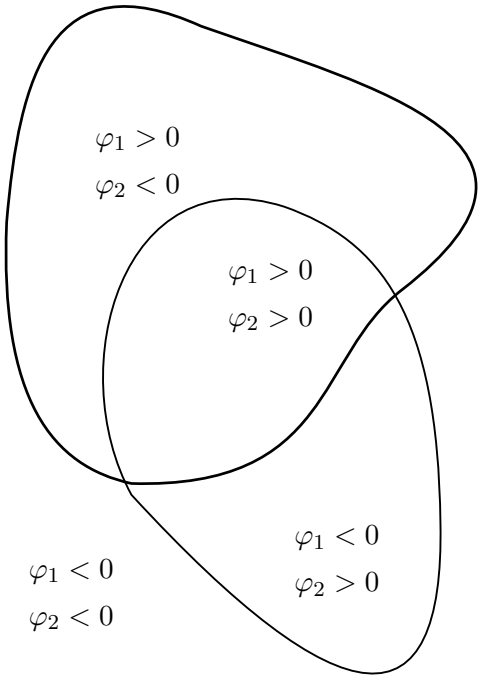


Figure 2.4: Example of a multiphase Chan Vese, with 2 level set functions, which result in 4 separated regions.

Splitting Active Contour Model

Li et. al. [24] proposed a model that splits the image in two regions with the Chan Vese model in Section 2.3.2. The new regions are splitted again until the result is an image with constant mean value. The advantage of this method is, that it uses only one level set function and is easy to implement.

Multi-Layer Image Segmentation

Wang et. al. [44] proposed a quite similar algorithm, which splits the image with one level set function according to the Chan Vese model in Section 2.3.2. After the splitting, the found region insight the contour is replaced by the average gray value outside the contour and the algorithm will start splitting again, until the difference of pixel intensities doesn't reach a predefined threshold anymore.

Conclusion

The multiphase Chan Vese model [41] heavily depends on the initialization of the level set functions. Whereas the other two papers [44, 24] try to extend the two phase model [13] with additional algorithmic overhead.

2.3.4 Global Minimization of the Active Contour Model based on Mumford-Shah

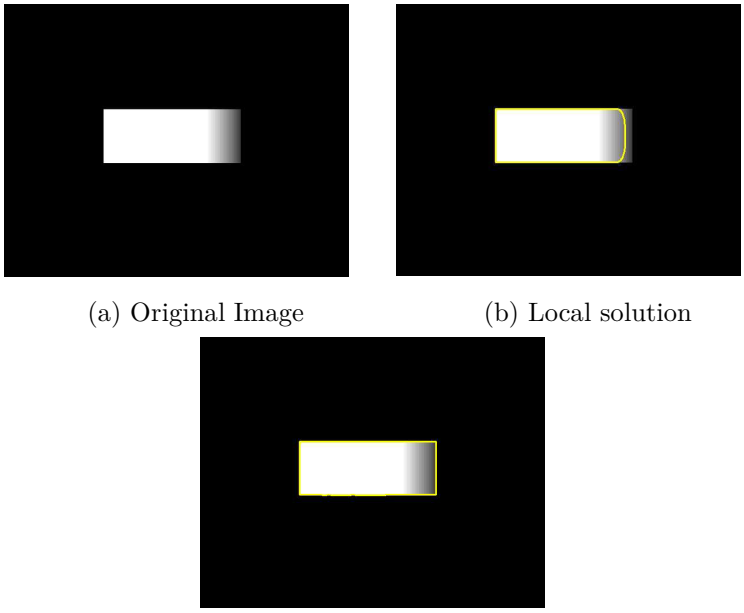
Bresson et. al. [6] enhances the standard Chan Vese model in order to determine a global minimum. Therefore they changed the energy functional proposed by Chan Vese [13],which was presented in Section 2.3.2.

$$E(u, c_1, c_2, \lambda) = \int_\Omega g(x)|\nabla u| \, \mathrm{d}x + \lambda \int_\Omega u((c_1 - f(x))^2 - (c_2 - f(x))^2) \, \mathrm{d}x \tag{2.16}$$

Where the first term is the weighted Total Variation. Bresson et. al. [6] states, that the Energy functional is homogeneous of degree 1 in u. Therefore it has only a stationary solution, if the minimization of u is restricted to $0 \leq u(x) \leq 1$. This leads to a minimization:

$$\min_{0 \leq u(x) \leq 1} \{E(u, c_1, c_2, \lambda)\} \tag{2.17}$$

With this changes of the Chan Vese model the proposed energy functional provides a global minimum. The advantages of this model to the original Chan Vese model, which only find a local minima, is shown in Figure 2.6.



(a) Original Image (b) Local solution

(c) Global solution

Figure 2.5: Global vs. local Chan Vese models. The local solution is generated with the original Chan-Vese model, whereas the global solution works with the model of [6].Reprinted from [6].

2.4 Object Separation

Other works concentrate on segmentation approaches used for object separation, similar to our work. In most papers a connected object is given and has to be separated. This section shows some of those approaches. All of those works can be split in two groups. One group concentrates on the shape of the connected object, like the watershed algorithm which we represent later. The other group tries to partition the connected object in homogeneous regions, similar to our Ising and Potts model which will also represented in the next chapter.

2.4.1 Minimum Entropy Segmentation

A work that uses a quite different approach to find homogeneous regions is the paper of Schwartzkopf et. al. [37]. They used a minimum entropy segmentation technique on classified multi-spectral chromosome images, to separate connecting components of chromosomes. Therefore they implemented an algorithm that uses the entropy definition of Shannon [38],

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{2.18}$$

Where p_i is the probability of the occurrence of a class C_i in the connected component. And C_i is one of the n classes with which a pixel can be labeled. They try to find a cut line in the connected object where the sum of the entropy of the two new objects is lesser than the entropy of the original object. If this cut line reduces the entropy more than any other cut line, the object will be separated. This is done recursively until no more cut lines can be found. This leads to a set of objects which tend to be over-segmented. Therefore they combine those separated objects if the combination reduces the entropy once more.

2.4.2 Clump splitting via bottleneck detection

Wang et. al. [43] proposed an algorithm which separates connected components, called clump, with two steps.
First they find a pair of points for splitting. They assume that the objects are convex in shape and connected objects tend to be concave. Further more they assume, that two objects are most likely connecting on the bottleneck of the clump contour. Therefore they apply following function, where A, B are all points on the boundary of the clump:

$$(A^*, B^*) = \arg\min_{A,B} \frac{dist(A,B)}{\min(length(A,B))} \tag{2.19}$$

Resulting in the points A^*, B^* which define the bottleneck of the clump, as shown in Figure 2.7. Where $dist(.,.)$ is a distance function on the Cartesian

Grid and $length(.,.)$ is the length of the boundary arc on the boundary of the clump. As the boundary of a clump is connected, there are always two results of the function $length(.,.)$, where the smaller one is taken by the function $\min(.,.)$.

The second step is, to find the best cut between the two found points. This is done by finding the cut with the minimal geodesic distance. A cut $c = (c_1, c_2 \dots c_L)$ between the two bottleneck points A^*, B^* is defined as a path between those points on the image I. Where $c_i = (x, y)$ is a point on the cut at position i and L is the total length of the cut. Therefore the best cut is defined as:

$$c^* = \arg\min_c \sum_{i=1}^{L} e(I(c_i))$$

where $e(I(c_i))$ is the L_1 norm of the discrete gradient at position c_i, as described in Section 3.3.1.

2.4.3 Segmenting clustered slender-particles

Zhong et. al. [48] proposed an algorithm, which extends the standard watershed segmentation as shown in Section 3.1, by reducing the over-segmentation.

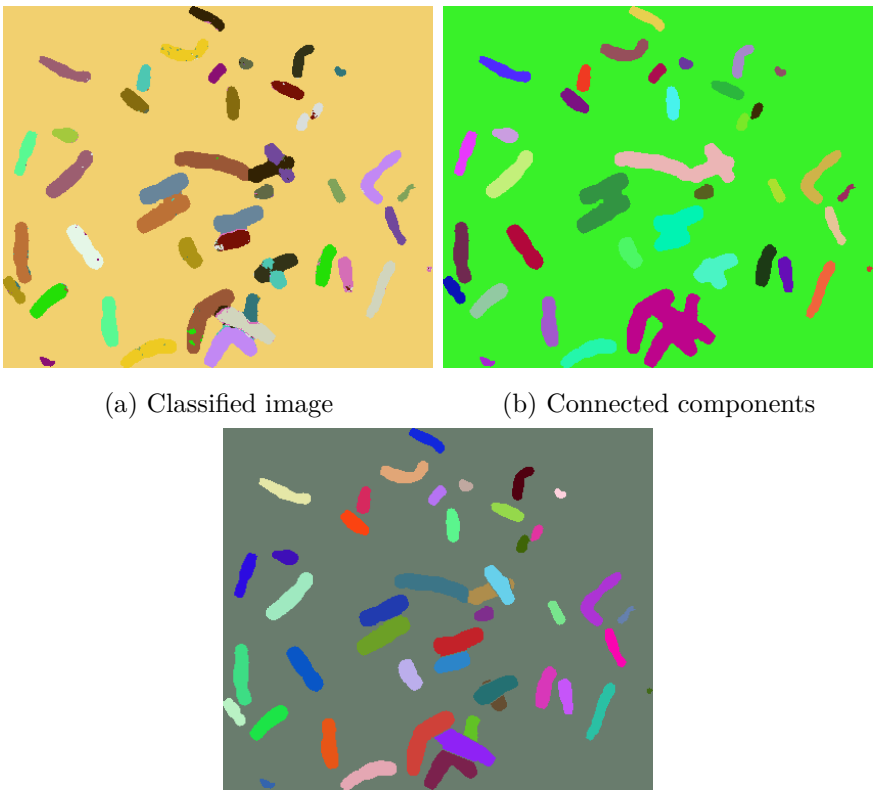
To reduce over-segmentation they only allowed watershed lines, which end-points are both concave, to be cut lines. They proposed an easy way to determine the concavity of a point x on the boundary of the object:

$$concavity(x) = \frac{A_x}{L} \tag{2.20}$$

Therefore they define a circular mask, whose center is x. L defines the perimeter of the circular mask and A_x is the arc length of the circular mask insight the object. Additionally they define an orientation of a concave point x, which is defined through the point x itself and the center point of the arc of the circular mask outside the object. See Figure 2.8 for an additional illustration.

Further more they proposed an algorithm, which allows to find additional cut lines, which are not found by the watershed segmentation. The algorithm starts on a concave point and tries to find a cut line to an other concave point according to six rules, which are best illustrated by Figure 2.9.

1. A very small path between the endpoints
2. Start and end have opposite orientations.
3. Splitting path orientation.
4. No crosses between paths.
5. Shortest length of all possible splitting paths are taken.
6. If there is only a single concavity in the object, the object is splitted the opposite way, to the points orientation.



(a) Classified image (b) Connected components

(c) Result of segmentation

Figure 2.6: Minimum entropy segmentation on a multi-spectral chromosome image. (a) Is the classified data used for the object separation. (b) shows the connected components which should be separated. (c) Separation result of minimum entropy segmentation as proposed in Section 2.4.1. Reprint from [37].

Figure 2.7: Bottleneck detection. Bottleneck pixels (red) of a clump, found by the clump splitting algorithm as defined in Equation 2.19. Reprinted from[43].

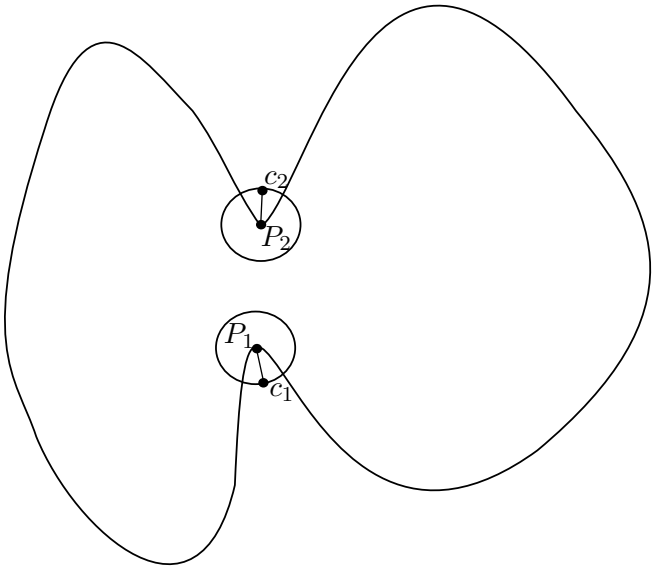


Figure 2.8: Determining concavity of a point. P_1 and P_2 are concave points where $\frac{A_x}{L} > 0.5$. Their orientation is defined through P_1, c_1 and P_2, c_2.

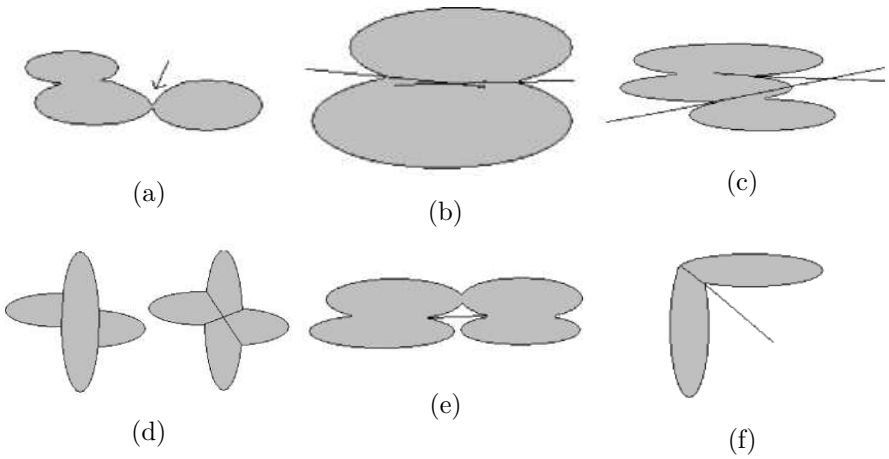


Figure 2.9: Determining correct splitting paths for clump splitting, according to the proposed rules in Section 2.4.3. Reprinted from [48].

Chapter 3

Proposed Models

This chapter shows the proposed algorithms to fulfill an adequate object separation. First we will concentrate on a standard algorithm for object separation, whereas the second two algorithms are state of the art variational models to solve such a problem.

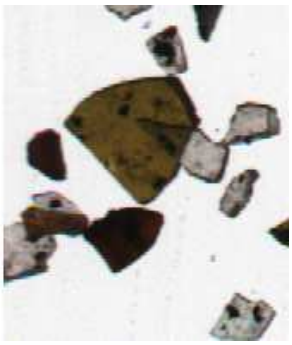
3.1 Watershed Segmentation

The first algorithm, we will use to separate objects, is the watershed segmentation for binary images. This algorithm will only depend on the binary representation of the objects. It depends on our observation, that the contour of one object of broken glass is almost convex and many connecting objects have often a concave contour, as shown in Figure 3.1. To separate such concave objects we will first perform a distance transform. After that, the watershed transformation will provide the borders (called watershed lines) of the separated objects.

3.1.1 Distance Transform

In order to use the contour information of a binary image, a distance transform has to be accomplished. Given a binary image $f : \Omega^h \to \{0, 1\}$ where

(a) Convex objects



(b) Connected concave objects

Figure 3.1: Convex objects vs. concave objects. (a) shows nearly convex objects which are properly separated. (b) shows that connected objects tend to be concave.

f contains two sets.

$$\Lambda_0 = \{f \in \Omega^h : f(x) = 0\}$$
$$\Lambda_1 = \{f \in \Omega^h : f(x) = 1\}$$

The distance transform $d : \Omega^h \to \mathbb{R}$ is the calculation of the distance of each pixel of the set Λ_0 to the nearest pixel of the set Λ_1.

$$d(x) = \begin{cases} 0 & \text{if } x \in \Lambda_1, \\ dist(x, \Lambda_1) & \text{else.} \end{cases} \tag{3.1}$$

The function $dist(x, A)$, where $x \in \Omega^h$ and $A \subset \Omega^h$, is defined as:

$$dist(x, A) = inf\{m(x, y) | y \in A\} \tag{3.2}$$

Where $m(x, y)$ can be any metric, for example the euclidean metric. An example of one watershed transformation, with different metrics, is shown in Figure 3.2. It is remarkable, that the chosen metric, can change the result of the watershed transformation significantly. Therefore different metrics will be reviewed in chapter 5.

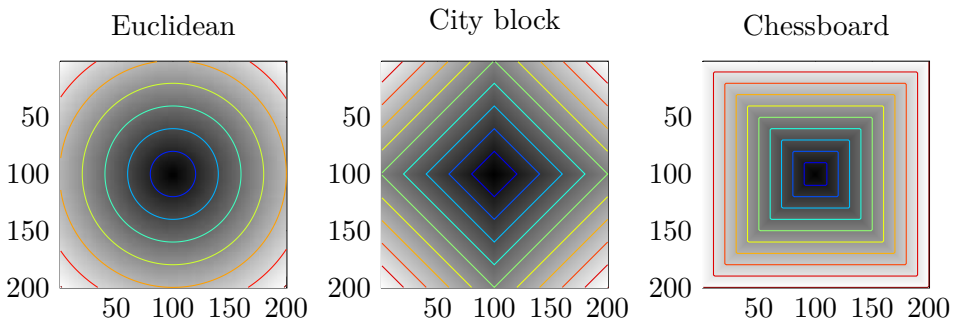


Figure 3.2: This reprint from [15] shows 3 distance transformations from a black image with a white dot in the middle. Each transformation used an other metric function. The used metric functions can be found in Equation (5.4)(5.5)(5.6)

3.1.2 Watershed Transformation

The task of the watershed transformation is to find the catchment basin and watershed lines of predefined minima. First of all we define the notions minima, catchment basin and watershed line in a formal way.

Definition The set of neighbours of a subset $X \subset \Omega^h$ of a Cartesian grid is defined trough their distance.

$$N_X = \{y \in \Omega^h : dist(X, y) \leq 1 \text{ and } y \notin X\} \tag{3.3}$$

Definition A connected set is a subset $X \subset \Omega^h$ of a Cartesian grid, where each element $x \in X$ has at least one neighbour, which is also in the subset.

Definition A minima of an image is a connected subset $X \subset \Omega^h$ of a Cartesian grid, where $f(x) < f(y)$ for all $x \in X$, $y \in N_X \setminus X$. This means, that the neighbours of the minima have all higher function values, than the minima itself.

Definition The catchment basin defines the influence of the minima. If you have an image with two minima X, Y, the catchment basin of the minima

X called $C(X)$ is defined as

$$\max_{|C(X)|} \{C(X) \supseteq X \text{ and } C(X) \cap C(Y) = \emptyset\} \tag{3.4}$$

Definition A watershed line is the set of pixels, which are in no catchment basin. That means this set defines the borders of all minima of the image.

Watershed Transformation - Algorithm

There are many algorithm which full-fill the watershed transformation. We have chosen an algorithm, which is called immersion and proposed in [42, 33]. This algorithm works by the idea of flooding the minima until they touch each other. A short pseudo code of the algorithm is presented in Algorithm 1.

Algorithm 1 Watershed Transformation

Require: $f : \Omega^h \to \mathbb{R}$

Require: Sets of all minima $S_1 \dots S_n$

```
Initialize watershed set W = {∅}
for k = 1 to MAXVALUE do
  for l = 1 to n do
    for all y ∈ N_{S_l} \ W do
      if y ∈ {S_1 ∪ ... ∪ S_n} then
        W = {W ∪ y}
      else
        if f(y) = k then
          S_l = {S_l ∪ y}
        end if
      end if
    end for
  end for
end for
```

At the end, each set $S_l \setminus W$ contains the pixel of one catchment basin and the set W contains all watershed pixel.

3.1.3 Watershed Segmentation - Algorithm

As proposed in [15] and [3] the watershed segmentation can separate connected, or overlapping objects. Therefore a distance transform has to be accomplished out of the binary representation of the object, in order to get a topographic surface representation. The local minimum of the inverse distance function also represents the minima for the watershed transformation. The produced watershed lines, represent the border of the separated objects. At last we will propose an additionally step similar to [48], which approximately calculates the concavity of the watershed line endpoints and the length of the watershed line. Only if both endpoints are concave or

the watershed line is small enough, the line will be used to separate found objects.

Algorithm 2 Watershed Segmentation

- Given binary image $f : \Omega^h \rightarrow \{0, 1\}$
- Invert the binary image
- Generate distance transform
- Generate watershed transformation out of inverted distance transform
- If $avg^l(w_i(first)) > c_1 \wedge avg^l(w_i(last)) > c_1 \vee length(w_i) < c_2$ then separate objects.

Where w_i is a list of all pixels of the i'th watershed line. $avg^l(w_i(x))$ calculates the average of the endpoints of w_i on the binary image f, by using a circular mask of size l.And $length(w_i)$ calculates the length of the watershed line.

3.2 Computing Minimal Partitions

As mentioned in section 1.3.2, we want a continuous formulation of the Potts, or for the case of two states, the Ising model, which is, as proposed in [29], the partitioning problem.

$$\min_{E_i}\{\frac{1}{2}\sum_{i=0}^{k}\mathrm{Per}(E_i,\Omega)+\sum_{i=0}^{k}\int_{Ei}f_i(x)\,\mathrm{d}x\}, \tag{3.5}$$

such that

$$\bigcup_{i=0}^{k}E_i=\Omega \qquad and \qquad E_i\bigcap E_j=0 \quad \forall i\neq j$$

Where k defines the number of partitions (with $k = 1$ in the Ising case). The first sum with the function $\mathrm{Per}(E_i, \Omega)$ defines the perimeter of the partition

E_i on the whole image domain $\Omega \subset \mathbb{R}^d$ and is the regularization term of the variational model. Therefore this term tends to minimize the perimeter of the partitions, which results in smooth segmentation boundaries. While each perimeter is counted two times, because each boundary is part of two segments, the coefficient $\frac{1}{2}$ is added. The second term is the data term, which measures how accurate the data is represented with the actual segmentation. The non-negative function $f_i : \Omega \to \mathbb{R}^+$ measures how well a point x is represented by the partition E_i. For example take Equation (3.6), here the function f_i is a binary function, which is 0 if the point x has the corresponding label of the partition E_i and 1 else. But this function can also have different representations.

$$f_i(x) = \begin{cases} 0 & \text{if } x = i, \\ 1 & \text{if } x \neq i. \end{cases} \tag{3.6}$$

3.3 Preliminaries

This subsection gives some definitions, which we later use to formulate our approach.

Definition Gradient and divergence. The gradient in the continuous setting is defined via the ∇ operator.

$$\nabla f = \left(\frac{\delta f}{\delta x_1}, \ldots, \frac{\delta f}{\delta x_n} \right)$$

The divergence operator div is the adjoint of ∇.

$$-\operatorname{div} = \nabla^*$$

Definition Total Variation.
The Total Variation of a smooth function θ is is defined as:

$$J(\theta) = \int_\Omega |\nabla \theta| \, \mathrm{d}x \tag{3.7}$$

For a L^1 integrable function the Total Variation can also be formulated by means of a dual formulation [7, 10, 9, 29].

$$J(\theta) = \sup_{\xi:|\xi(x)|\leq 1} \{-\int_\Omega \theta \operatorname{div} \xi \, \mathrm{d}x\} \tag{3.8}$$

Where $\xi : \Omega \to \mathbb{R}^d$ is the dual variable.

Definition Bounded Variations.
A function is said to have bounded variations (BV) if the Total Variation of the function $J(u) < +\infty$.

A often used theorem in connection with the Total Variation is the co-area formula from Federer and Fleming [21]. This formula shows an geometrical property of the Total Variation, because it can be decomposed by means of the level sets of the function. So the Total Variation can be seen as the accumulated surface of all of its level sets of a function.

Theorem 3.3.1. *Co-area Formula. For a function $u \in BV(\Omega)$ and for a.e. $s \in \mathbb{R}$, the set $\{u > s\}$ is a set with finite perimeter in Ω and following equation holds [9]:*

$$J(u) = \int_\Omega |\nabla \theta| \, \mathrm{d}x = \int_{-\infty}^{\infty} Per(\{u > s\}; \Omega) \, \mathrm{d}s \tag{3.9}$$

Please refer to [20] for a proof of this formula.

Definition The Subdifferential.
Is a generalisation of the gradient to non differentiable functions and is defined as:

$$\partial F(X) = \{y \in \mathbb{R}^n : \langle y, x' - x \rangle \leq f(x') - f(x), \forall x' \in \mathbb{R}^n\} \tag{3.10}$$

A member of the subdifferential is called subgradient. In Figure 3.3 subgradients of a non differentiable function are illustrated.

Definition Legendre-Fenchel transform.
Is a transformation of a function $f : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$, which has to be continuous but not necessarily differentiable, into:

$$F^*(y) = \sup_y \{\langle x, y \rangle - F(x)\} \tag{3.11}$$

It describes the relation between the slope of the function $F(x)$ to the intersection of the tangent on a point $x \in F(x)$ with the y-axis. As stated in Handa et. al. [1], the transformation turns points of the function F into slopes of F^*, and slopes of F into points of F^*. An example of the transformation can be seen in Figure 3.3.

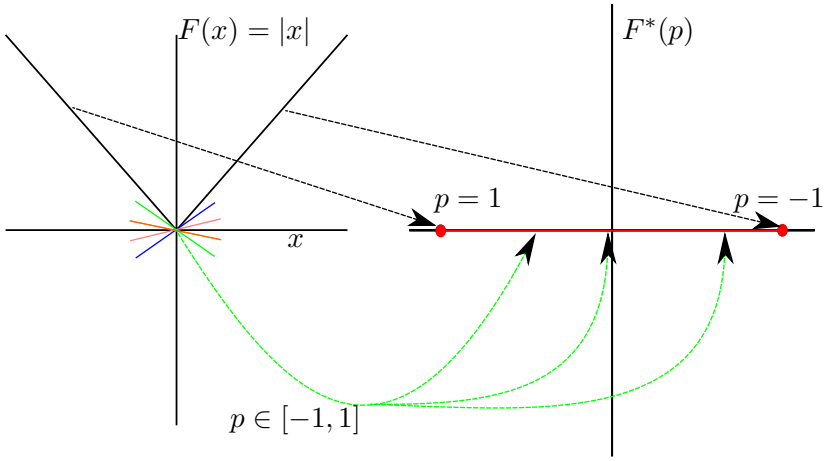


Figure 3.3: This reprint from [1] shows the Legendre-Fenchel transformation of the function $F(x) = |x|$. It shows the connection between points on the function F to slopes on the function F^*. It also shows some subgradients on the position $F(x) = 0$.

Theorem 3.3.2. *The Legendre-Fenchel transform is always convex. No matter if F is convex or not, F^* is always convex. Therefore the Legendre-Fenchel transform is also called the convex conjugate. For a proof please refer to [1].*

Definition Duality

Duality is the principle of looking at a function or problem from two different perspectives, which are called the primal and the dual form [1]. In the case of the Legendre-Fenchel transform, we use the duality of tangents and points to represent a function. That means a function can be represented by its points, or by its tangents. Usually the function is represented by its points, such that $(x, F(x))$ define the curve of the function. The other tangential representation is parameterized by the slope and the intercept it cuts on the negative y-axis. Therefore $(x, F(x)) \Leftrightarrow (p, F^*(p))$ are the dual from each other. Figure 3.4 shows the duality between the point-wise representation of a function and the Legendre-Fenchel transform. For further readings about primal, dual and conjugate of functions we refer to [32].

3.3.1 Discrete Settings

For discretization we use the same notation as [29, 9, 10]. Therefore we use a standard Cartesian grid Ω^h with size $M \times N : \{(ih, jh) : 1 \leq i \leq M, 1 \leq j \leq N\}$. Where (i, j) are indizes of the grid and h is the width of the spatial discretization, called discretization step. For example h could be $h = 1/N$. u is now a discretized image function $u : \Omega^h \to \mathbb{R}$. To simplify the notation we also use x for the location on the grid, and use function arguments to index the images.

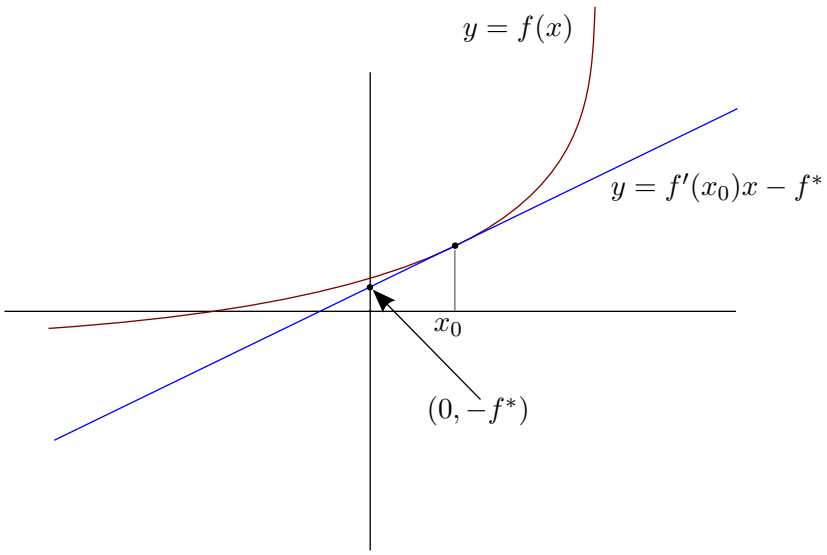


Figure 3.4: Dual representation of a function. This figure shows the Legendre-Fenchel transformation of a function. Where a point x_0 of a function f is represented by its tangent. It can be seen that f^* is the intercept, the tangent of $f(x_o)$ cuts on the negative y-axis.

Definition Discrete gradient.

$$(\nabla u)_{i,j} = \begin{pmatrix} (\nabla u)^1_{i,j} \\ (\nabla u)^2_{i,j} \end{pmatrix}$$

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & i < M \\ 0 & i = M \end{cases}$$

$$(\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & j < N \\ 0 & j = N \end{cases}$$

Now $(\nabla u)_{i,j}$ is a function $X \to Y$ where $Y = X \times X$.

Definition The discrete divergence operator $\operatorname{div} p : Y \to X$ is defined as

$$(\operatorname{div} p)_{i,j} = \begin{cases} p^1_{i,j} - p^1_{i-1,j} & 1 < i < M \\ p^1_{i,j} & i = 1 \\ -p^1_{i-1,j} & i = M \end{cases} \tag{3.12}$$

$$+ \begin{cases} p^2_{i,j} - p^2_{i,j-1} & 1 < j < N \\ p^2_{i,j} & j = 1 \\ -p^2_{i,j-1} & j = N \end{cases} \tag{3.13}$$

where $p = (p^1, p^2)$

Definition Discrete Version of the Total Variation. Let h be the discretization step, then a simple discretization of the Total Variation is:

$$TV_h(u) = h^2 ||\nabla u||_{2,1} \tag{3.14}$$

Where

$$||p||_{2,1} \quad = \quad \sum_{i,j=1}^{N} \sqrt{(p^1_{i,j})^2 + (p^2_{i,j})^2},$$

i, j are the indizes of the discrete grid.

It can be shown that the discrete version of the Total Variation $\Gamma-converges$ to the Total Variation as $h \to 0$. So only by minimizing the discretization step, the discrete Total Variation converges to the continuous formulation. A proof of this theorem can be found in [11].

3.4 Ising Model

As stated in [14, 29] the minimal partition problem in Section 3.5 for the two label case with $k = 1$ can be rewritten in term of the variational model

$$\min_{\theta} \{ \int_{\Omega} |\nabla \theta| \, \mathrm{d}x + \lambda \int_{\Omega} (1 - \theta(x)) f_0(x) + \theta(x) f_1(x) \, \mathrm{d}x \} \tag{3.15}$$

where θ denotes an binary function $\theta : \Omega \to \{0, 1\}$, which is used to define the partition of the image domain. We use a slightly different formulation of the variational model of Equation (3.15), which has a more compact mathematical representation.

$$\min_u \{ \int_\Omega |\nabla \theta| \, \mathrm{d}x + \lambda \int_\Omega (f_1 - f_0) u \, \mathrm{d}x \} \tag{3.16}$$

Where $u : \Omega \to \{0, 1\}$ is a binary function, same as θ from (3.15). It can be shown that the two representations of (3.15) and (3.16) are the same up to a constant f_0, which does not change the minimization problem up to a change in λ.

$$\int_\Omega (1 - \theta) f_0 + \theta f_1 \, \mathrm{d}x = \int_\Omega f_0 + \theta f_1 - \theta f_0 \, \mathrm{d}x = \int_\Omega f_0 + u f_1 - u f_0 \, \mathrm{d}x$$
$$\Rightarrow \int_\Omega (1 - \theta(x)) f_0(x) + \theta(x) f_1(x) \, \mathrm{d}x = \int_\Omega f_0 + (f_1 - f_0) u \, \mathrm{d}x$$

The regularization term $\int_\Omega |\nabla \theta| \, \mathrm{d}x$ is the Total Variation of the binary function u, which is equal to the total interface area.

3.4.1 Convex Relaxation

Our minimization problem (3.16) has to be convex in order to find a global minimum. Therefore the easiest way to achieve a convex relaxation is to allow u to vary smoothly $u : \Omega \to [0, 1]$. As it can be shown in Proposition 3.4.1, the solution of the relaxed problem can be transformed to a solution of the binary problem by simply thresholding it.

Proposition 3.4.1. *The minimization problem in Equation* (3.5) *can be relaxed as follows:*

$$\min_{u \in BV(\Omega;[0,1])} \{ J(u) + \lambda \int_\Omega u(x) f(x) \, \mathrm{d}x \} \tag{3.17}$$

given any solution of u with any value $s \in [0, 1)$, the set $\{u > s\}$ is a solution of (3.5).

Proof. This proposition is a direct consequence of the Co-area formula of theorem 3.3.1. For a detailed proof see [9]. □

3.4.2 Discrete Ising Model

With the convex relaxation of u we now obtain the convex relaxed Ising model:

$$\min_{u \in BV(\Omega;[0,1])} \{J(u) + \lambda \int_{\Omega} (f_1 - f_0) u \, \mathrm{d}x\} \tag{3.18}$$

In order to use the Ising model on images, we have to discretize the continuous definition of (3.18). Therefore f_0, f_1, u are now discrete functions on a discrete grid as defined in the Preliminaries 3.3.1 of this section. We use the discrete version of the Total Variation $TV_h(u)$ defined in Equation (3.14) with $h = 1$. So the discrete Ising model is defined as:

$$\min_{u \in [0,1]} ||\nabla u||_{2,1} + \lambda (f_1 - f_0) u \tag{3.19}$$

3.4.3 Primal Dual Algorithm

To find a solution for our discrete Ising model we use the primal dual algorithm form [30, 12]. We have chosen this algorithm, because it can be easy accelerated with the GPU(Graphic Processing Unit).

Primal Formulation

The primal dual algorithm, as defined in [12], works for a general class of minimization problems of this form:

$$\min_{x \in X} F(Kx) + G(x) \tag{3.20}$$

Where $G, F : X \to [0, +\infty)$ are proper, convex, lower-semicontinuous functions. X, Y are finite-dimensional real normed vector spaces, where $N = dim(X)$ and $M = dim(Y)$. $K : X \to Y$ is a continuous linear operator with operator norm

$$||K|| = \sup\{||Kx|| : x \in X \text{ with } ||x|| \leq 1\} \tag{3.21}$$

Primal-Dual Formulation

With the convex conjugate, we can turn the primal formulation into a primal dual formulation, used for the proposed algorithm.

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \tag{3.22}$$

Where $F^* : Y \to [0, +\infty)$ is the convex conjugate of F which is defined in Definition 3.3. Further more we suppose that F, G are simple in the way, that their proximal operator has a closed form representation.

$$x = (I + \tau \partial F)^{-1} = \min_x \left\{ \frac{||x - y||^2}{2\tau} + F(x) \right\} \tag{3.23}$$

With this information the primal dual algorithm from [12] can be established.

Algorithm 3 Standard Primal Dual Algorithm

Initialization: $\tau, \sigma > 0$,$\Theta \in [0, 1]$,$x^0, y^0 \in X \times Y$

For all iterations ($n \geq 0$):

$$\begin{aligned} x^{n+1} &= (I + \tau \partial G)^{-1}(x^n - \tau K^* y^n) \\ y^{n+1} &= (I + \sigma \partial F^*)^{-1}(y^n + \sigma K(x^{n+1} + \theta(x^{n+1} - x^n))) \end{aligned}$$

This algorithm performs alternating projected gradient descent steps in x and gradient ascend steps in y. After each iteration the algorithm re-projects the primal and dual variable into their respective sets.

Setting up the step size

As stated in [12], the step sizes τ, σ must be chosen with respect to following inequation.

$$\tau \sigma L^2 < 1 \tag{3.24}$$

Where L is called the Lipschitz constant. Chambolle et. al. [12] showed, that the Lipschitz constant is the operator norm of the linear operator K:

$$L = ||K|| \tag{3.25}$$

If the structure of K is complicated, for example K has different entries in each row such that the closed from solution of the operator norm of K is hard to compute Pock et. al. [28] proposed an other way to distinguish τ and σ out of K.

$$\tau_j = \frac{1}{\sum_{i=1}^{M} |K_{i,j}|^{2-\alpha}} \tag{3.26}$$

$$\sigma_i = \frac{1}{\sum_{j=1}^{N} |K_{i,j}|^{\alpha}} \tag{3.27}$$

Where $\alpha \in [0, 2]$. As stated in [28] this leads to dimension dependent time steps τ_j, σ_i, which do not change the computational complexity of the algorithm. Furthermore they showed, that this time steps are faster for complicated K, than those computed by the operator norm.

3.4.4 Ising Primal Dual

In order to implement the discrete Ising model of Equation (3.19) with the primal dual algorithm stated in 3.4.3, we have to turn the primal formulation from (3.19) into a primal dual formulation. Therefore we have to turn the discrete Total Variation in its dual formulation, according to (3.8).

$$\min_{u \in X} \max_{p \in Y} \langle \nabla u, p \rangle + \lambda \langle f_1 - f_0, u \rangle - \delta_P(p) + \delta_U(u) \tag{3.28}$$

Where $X \in [0, 1]$, $Y = X \times X$ and δ_p is a indicator function on a convex set P.

$$P = \{p \in Y : ||p||_\infty \leq 1\}$$

$$\delta_P(p) = \begin{cases} 0 & \text{if } p \in P \\ \infty & \text{else} \end{cases} \tag{3.29}$$

$||p||_\infty$ is called the discrete maximum norm which is defined as,

$$||p||_\infty = \max \sqrt{(p^1_{i,j})^2 + (p^2_{i,j})^2} \tag{3.30}$$

So p has to lie in the unit circle of the L^2 norm to achieve a maximum, other than $-\infty$ in equation (3.28). To fulfil the additional requirement of u being between zero and one the indicator function δ_u is added, which works on the convex set U.

$$U = \{u \in X : 0 \leq u \leq 1\}$$

$$\delta_U(u) = \begin{cases} 0 & \text{if } u \in U \\ \infty & \text{else} \end{cases} \tag{3.31}$$

To implement the algorithm in 5 we have to find the proximal operators of $\delta_P(p)$ which is $F^*(p)$ from (3.22) and $\lambda\langle f_1 - f_0, u\rangle + \delta_U(u)$, which is $G(u)$ from (3.22). As stated in [12] the proximal of $\delta_P(p)$ reduces to pointwise euclidean projectors on L^2 balls.

$$(I + \sigma\partial(\delta_P(p)))^{-1} = \frac{p_{i,j}}{\max(|p_{i,j}|, 1)} \tag{3.32}$$

The proximal operator of $G(u)$ can be calculated by the definition of the proximal operator (3.23) and has following result.

$$(I + \tau\partial(G(u)))^{-1} = \max(\min(1, (f_0 - f_1)\tau\lambda + u), 0) \tag{3.33}$$

As shown in Figure 3.5 by Equation (3.33), the proximal operator finds the optimum within $x \in [0, 1]$. Now the primal dual algorithm of the Ising model can be implemented. A detailed description of the single algorithm steps are found in Algorithm 4.

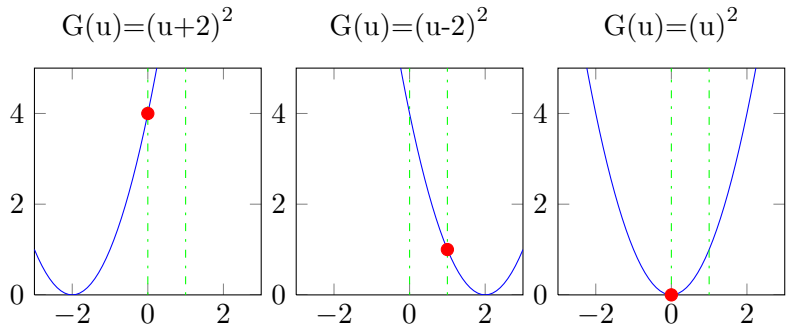


Figure 3.5: Proximal Operator of G(u) with $u \in [0,1]$. This figure shows how the proximal operator of a function $G(u)$, finds the minimum of $G(u)$ with respect to $u \in [0,1]$. Where the red dot marks the found minimum of u.

Algorithm 4 Primal Dual Ising Model

Initialization: $\tau, \sigma > 0$,$\Theta \in [0,1]$,$u^0 \in X, p^0 \in Y$

For all iterations ($n \geq 0$):

$$
\begin{aligned}
u^{n+\frac{1}{2}} &= (u^n - \tau \nabla^* p^n) \\
u^{n+1} &= \max(\min(1, (f_0 - f_1)\tau\lambda + u^{n+\frac{1}{2}}), 0) \\
p^{n+\frac{1}{2}} &= (p^n + \sigma \nabla (u^{n+1} + \theta(u^{n+1} - u^n))) \\
p^{n+1} &= \frac{p_{i,j}^{n+\frac{1}{2}}}{\max(|p_{i,j}^{n+\frac{1}{2}}|, 1)}
\end{aligned}
$$

The step sizes of the two constants τ, σ have to fulfil the requirements of Equation (3.24). Therefore the Lipschitz constant can be calculated according to Equation (3.25). In the case of the primal dual Ising model the operator K is simple $K = \nabla$. Therefore the operator norm of ∇ can be estimated by using the Matlab function *normest*, which results in: $L = ||\nabla|| = \sqrt{8}$ and

$$\tau\sigma \leq \frac{1}{8} \tag{3.34}$$

3.5 Potts Model

After we found an algorithm to solve the Ising model we want to find a variational model for the case of k labels, the Potts model, to get a more precise algorithm for our object separation task. Therefore we have to rewrite Equation (3.17) in order to gain more than two partitions. We use the relaxation approach from Zach et al. [47]

$$\min_u = \frac{1}{2}\sum_{l=1}^{K}\int_\Omega |\nabla u|\,\mathrm{d}x + \lambda\sum_{l=1}^{K}\int_\Omega u_l(x) f_l(x)\,\mathrm{d}x \tag{3.35}$$

with the additional conditions $u_l(x) \in [0,1]$ and $\sum_{l=1}^{k} u_l(x) = 1$. So the function u has changed to a vector function $u : \Omega \to [0,1]^k$. If we change u to a binary function $u(x) \in \{0,1\}^k$ this formulation is equal to the minimal partitions problem defined in (3.5). The first term in (3.35) is simple the sum of the Total Variation of u for each label. As shown in [29] this relaxation is too small and there exists a more precise formulation, but this formulation results in a faster algorithm, which brought us to the decision to use this relaxation.

3.5.1 Label Cost

A major problem with this model is to define the correct number of partitions, which should be found. In our case an upper bound of k can easily be defined by the number of classes of the image's pixels. This upper bound is in many cases too high, so there will be resulting partitions which are empty. But this empty partitions increment the computation time of the algorithm, because they have to be computed in every iteration. Our approach is, to include a label cost term in the model, which gives us information of the priority of the label, so the unused labels can be excluded form the calculation. An other positive effect is, that the label prior helps to minimize the amount of used labels for the partitioning problem. We expand our model with the label cost term proposed in [46]. The label cost term for the case

that $u(x) \in \{0,1\}^k$ can be written as

$$\gamma M \text{ where } M = \#\{1 \leq l \leq k | u_l \neq 0\} \tag{3.36}$$

Where γ is used to weight the term in our model. If we include this term in the convex relaxed model from (3.35) we obtain,

$$\min_{u \in [0,1]} = \frac{1}{2} \sum_{l=1}^{k} \int_\Omega |\nabla u_l| \, \mathrm{d}x + \lambda \sum_{l=1}^{k} \int_\Omega u_l(x) f_l(x) \, \mathrm{d}x + \gamma \sum_{l=1}^{k} \max_{x \in \Omega} u_l(x) \tag{3.37}$$

where the third term is the infinity norm of $u_l(x)$, which is the convex analogue of the label cost term defined in (3.36).

3.5.2 Discretization

Now $f, u :\in \mathbb{R}^k$ are discrete vector functions on a discrete grid as defined in Preliminaries 3.3.1. According to [46] the infinity norm of the label prior can be replaced by an additional scalar $y \in \mathbb{R}^k$, for each label, with some additional constraints. Furthermore the Total Variation in the regularization term is replaced by its discrete version defined in (3.14). With this modifications we obtain a discrete model.

$$\min_{u,y} \sum_{l=1}^{k} ||\nabla u_l||_{2,1} + \lambda \sum_{l=1}^{k} \langle u_l, f_l \rangle + \gamma \sum_{l=1}^{k} y_l \tag{3.38}$$

with additional constraints

$$u_l(x) \in [0,1] \tag{3.39}$$

$$\sum_{l=1}^{k} u_l(x) = 1 \tag{3.40}$$

$$0 \leq (u_l)_{i,j} \leq y_l \; \forall i, j \tag{3.41}$$

3.5.3 Potts Primal Dual

To obtain an algorithm which is fast and well suited for GPU processing, we have chosen to take the primal dual algorithm 3. We use a very similar

primal dual definition of the potts model as in [40]. In order to obtain a primal dual definition of the discrete model from (3.38), we have to change the Total Variation definition into its dual definition, as defined in (3.8), similar to (3.28). Furthermore we define Lagrange multipliers to take the additional constraints (3.39)(3.40)(3.41) into account. First we define the Lagrange multiplier $q :\in \mathbb{R}^K$ to tackle constraint (3.41). We use a additional matrix P, as defined in [40], to write the constraint in a single condition. Where $P = (I, -1^t)$ with an identity matrix $I \in X \times X$ and a row vector 1 with all entries equal to one. So an additional term $\langle q, P \begin{pmatrix} u \\ y \end{pmatrix} \rangle$ and an indicator function $\delta(q)$, which forces q to be positive, can be added to tackle the label-cost constraint (3.41). The other constraints (3.39) (3.40) can be combined by defining an Lagrange multiplier $s :\in \mathbb{R}$, for the term $\langle s, \sum_{l=1}^{k} u_l - 1 \rangle$, which forces that $\sum_{l=1}^{k} u_l(x) = 1$ at each position x on the discrete grid. To avoid that $u_l(x) < 0$ an indicator function $\delta(u)$ is added.

$$\begin{aligned} \min_{u,y} \max_{p,q,s} \sum_{l=1}^{k} \Bigg(& \langle \nabla u_l, p_l \rangle + \lambda \langle u_l, f_l \rangle + \gamma y_l + \\ & \langle q_l, P \begin{pmatrix} u_l \\ y_l \end{pmatrix} \rangle - \delta_Q(p_l) - \delta_\Lambda(q_l) + \delta_\Lambda(u_l) \Bigg) \\ & + \langle s, \sum_{l=1}^{k} u_l - 1 \rangle \end{aligned} \tag{3.42}$$

where the indicator functions are defined by the sets

$$\begin{aligned} Q = & \quad \{p \in \Omega^h \times \Omega^h : ||p||_\infty \leq 1\} \\ \Lambda = & \quad \{z \in \Omega^h : z(x) \geq 0\} \end{aligned}$$

and the indicator function for a set C

$$\delta_C(p) = \begin{cases} 0 & \text{if } p \in C \\ \infty & \text{else} \end{cases}$$

Algorithm

In order to obtain the primal dual algorithm for the Potts model out of Equation (3.42), we can transfer it to a problem of the form defined in Equation (3.22). Therefore we concatenate the two primal variables and three dual variables to one primal and one dual variable:

$$x = \begin{pmatrix} u_1 \\ y_1 \\ \vdots \\ u_k \\ y_k \\ 1 \end{pmatrix}, \; z = \begin{pmatrix} p_1 \\ q_1 \\ \vdots \\ p_k \\ q_k \\ s \end{pmatrix}, \; O = \begin{pmatrix} \tilde{\nabla} & | & \dots & | & 0 \\ & & \vdots & & \\ 0 & | & \dots & | & \tilde{\nabla} \\ P & | & \dots & | & 0 \\ & & \vdots & & \\ 0 & | & \dots & | & P \\ & & Z & & \end{pmatrix} \tag{3.43}$$

Such that x is the concatenated primal variable, z the concatenated dual variable and O is the concatenated linear operator. Where $\tilde{\nabla} = (\nabla, 0^t)$ and $Z = (I, 0^t, \dots I, 0^t, -1^t)$ with an identity matrix $I \in X \times X$. Furthermore we define $G(x) = \sum_{l=1}^{k} \lambda \langle u_l, f_l \rangle + \gamma y_l + \delta_\Lambda(u_l)$ and $F^*(z) = \sum_{l=1}^{k} \delta_Q(p_l) + \delta_\Lambda(q_l)$. With this definition of the Potts model, we can implement the primal dual algorithm as defined in Section 3.4.3, which is similar to the algorithm proposed in [40], without the map uniqueness constraint.

Algorithm 5 Potts Primal Dual with Label Prior

Initialization: choose s^1, q^1, u^1, p^1, y^1

for all $n \geq 0$ **do**

for $l = 1$ to K **do**

$s^{n+1} = s^n + \sigma(u_l^n - \frac{1}{k})$

$p_l^{n+\frac{1}{2}} = p_l^n + \sigma \nabla u^n$

$p_l^{n+1} = \frac{p_l^{n+\frac{1}{2}}}{\max(1, ||p_l^{n+\frac{1}{2}}||)}$

$q_l^{n+1} = \max\left(0, q_l^n + \sigma\Big(P \begin{pmatrix} u_l^n \\ y_l^n \end{pmatrix}\Big)\right)$

end for

for $l = 1$ to K **do**

$y_l^{n+\frac{1}{2}} = y_l^n - \tau(\gamma - \sum_{i,j}(q_l)_{i,j})$

$y_l^{n+1} = 2y_l^{n+\frac{1}{2}} - y_l^n$

$u_l^{n+\frac{1}{2}} = max(0, u_l^n - \tau(-\operatorname{div} p_l^{n+1} + q_l^{n+1} + s^{n+1} + \lambda f_l)$

$u_l^{n+1} = 2u_l^{n+\frac{1}{2}} - u_l^n$

end for

end for

Setting up the Step Size

As mentioned in Section 3.4.3, there are two ways to calculate the correct step size. The linear operator O of the Potts model is complex. Therefore we calculate τ and σ with the method proposed by Pock et. al. [28] which is shown in Equation (3.26) and Equation (3.27). There are three different steps $\sigma_p, \sigma_q, \sigma_s$ according to the dual variables p, q, s, which will be calculated with Equation (3.27). Where $\sigma_p = \frac{1}{2}$ because the absolute sum of one row of the ∇ operator is 2. Same for $\sigma_q = \frac{1}{2}$, because the absolute sum of one row of P is 2, too. Each row of the Z operator has $k+1$ entries, where k is the number of labels used by the Potts algorithm. Therefore $\sigma_s = \frac{1}{k+1}$.
The two primal steps are computed with Equation (3.26), where the absolute

sum of the columns of O have to be calculated. This leads to $\tau_u = \frac{1}{6}$, because the absolute sum of the first columns of O according to ∇, P, Z are $4, 1, 1$. And $\tau_y = \frac{1}{MN}$, because the absolute sum of the last columns of O according to ∇, P, Z are $0, MN, 0$.

Chapter 4

Implementation Details

As mentioned in the previous chapter, the Ising and Potts model in combination with the primal dual algorithm is suitable to be run on a Graphic Processing Unit, called GPU. This will provide a speedup, which allows us to run the algorithms in realtime environments.

4.1 GPU

We have chosen to implement two of our algorithms on a Nvidia GPU, because of its general purpose parallel computing architecture CUDA. A good description of the purpose and use of CUDA can be found in [19, 17, 18]

4.1.1 CUDA

It includes the CUDA instruction set architecture, and the parallel computing engine in the GPU. The major benefit of CUDA is, that it provides a small extension to the standard programming language C, which enables a straightforward implementation of parallel algorithms. As the whole REDWAVE software is coded in C and CUDA supports heterogeneous computation, where the serial tasks are run on the CPU and parallel tasks are loaded to the GPU, it can be easy adapted to the existing software product. Fur-

thermore parallel computing on CUDA can be applied step wise by changing sequential CPU code to parallel CUDA code. An other benefit of CUDA, stated in [19] is, that the CPU and GPU are treated as separate devices with separate memory spaces, which allows simultaneous computation on both devices at the same time, without contention of the memory.

Programming Details

CUDA C extends the programming language by defining C functions, which are executed N times in parallel by N different CUDA threads.
This functions are called kernels and are defined by the use of the keyword **___global___** in the declaration specifier. Furthermore by calling the kernel function, the programmer has to define the number of threads per block and the number of blocks per grid. Which defines how often the kernel function will be executed parallel. An example of a two dimensional grid with a two dimensional kernel can be seen in Figure 4.1.

Threads per Block All threads of a block reside on the same processor core, and have to share the same limited amount of memory. The advantage of threads in one block is that they can access a shared memory which allows them to communicate faster.

Blocks per Grid But there can only be a limited number of threads in one block. So several equal sized blocks have to be defined, which are organized into a grid of thread blocks. Unfortunately the memory access between blocks is much slower.

CUDA C has predefined variables which exist in kernel functions, in order to identify the thread which is executed. The **blockIdx** variable identifies the block within the grid and the **threadIdx** variable identifies the thread within a block. Both variables can be of dimension one to dimension three, depending on the definition in the program. In order to compute an unique

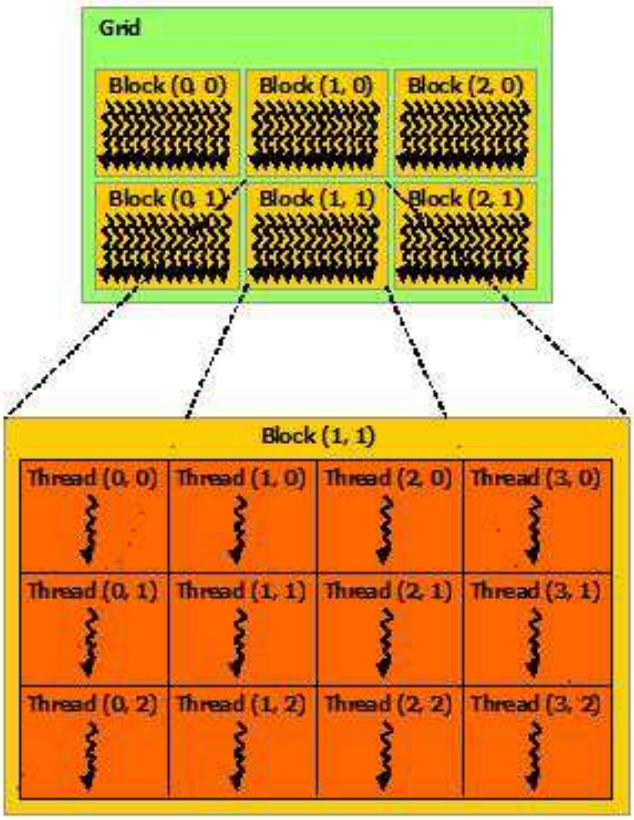


Figure 4.1: A reprint from [18] which shows a two dimensional grid of two dimensional blocks, where each block contains 4×3 threads. Which are used to define how often a kernel function will be executed parallel.

identifier a third variable, **blockDim**, can be accessed, which provides the dimensions of each block.

The ideal use of those threads is to compute numerous data elements simultaneously. Where each thread computes at least one data element, which is connected to the thread by its index, parallel to the other threads.

4.1.2 GPU Architecture

We have chosen to test our implementations with an Nvidia Geforce GTX 580, which is based on the Fermi architecture, which will be shortly discussed in this section.

Fermi

The Geforce GTX 580 is a Fermi based GPU. As specified in [16] it features 512 CUDA cores, which are separated into 16 Streaming Multiprocessors (SM) with 32 cores. Each CUDA core has a fully pipelined integer arithmetic logic unit and a floating point unit. Furthermore each SM has 16 load/store units, which allows to calculate source and destination addresses for 16 threads per clock. An other feature of the fermi architecture is, that its L1 chache and its shared memory of the SM are on-chip and the programmer is able to change the size between the two memories. An architecture overview of one SM is shown in Figure 4.2.

4.1.3 Memory Space

As one can mention from the previous sections, a major optimization concern in parallel programming with CUDA are the memory optimizations. There are several types of memories, which can be accessed by the GPU, which have all different advantages and disadvantages. An overview of the different memories and their position on the device is given in Figure 4.3.

Host Memory

The GPU cannot access host memory direct, so the memory has to be copied from host to the device and vice versa. This data transfer between host and device should be minimized because it is very slow.

Global Memory

Is the device memory which is located in the DRAM and can be accessed by all SM and all threads. It is the slowest type of memory and read/write operations should be avoided. Fermi architecture has L1 cache which enables the SM to buffer the global memory variables.

Local Memory

If a thread has not enough registers to save all local variables into, it uses the DRAM as local memory. This should be avoided by using not too much local variables.

Registers

If there are enough registers the local variables are loaded into it automatically. This is the fastest memory.

Shared Memory

Located in the SM and very fast, but it only can be accessed block wise. Therefore each block of threads has to load its data from global memory to shared memory before usage. Should be used if the same location in global memory is read by more than one thread. Be aware shared memory uses the same memory than the L1 cache. Therefore the programmer has to make a tradeoff between the usage of shared memory and L1 cache.

Constant and Texture Memory

Can be accessed by all threads. The advantage of both is that they are cached and therefore there is only a DRAM access on a cache miss. For further readings please refer to [17].

4.2 Implementation

4.2.1 Watershed Segmentation

We have chosen to implement the watershed segmentation algorithm in Matlab, which gives us the advantage of a pre-defined distance transformation and watershed segmentation. As there are many libraries that provide the watershed algorithm like the OpenCV Library [5], there are many ways to include this algorithm with little programming effort in the existing REDWAVE software.

4.2.2 Ising Model

As mentioned above the Ising model is implemented in CUDA C, where the primal and dual steps are implemented as CUDA kernel functions. Algorithm 4 shows the two steps, which can be computed parallel per pixel, because the calculation doesn't need much information of their neighbours.

Enhanced use of Shared Memory

The only ∇ and div operators need information from pixel neighbours. Therefore we installed an static shared memory, where each thread copies its repeatedly used data into. Although this should minimize the data access of global memory, the benefit of this change with respect to the consumed time was minor. One reason for the lack of benefit is, that the shared memory can only be used per block in CUDA. This has the disadvantage, that the neighbouring pixels, which aren't in the block, have to be loaded into the shared memory. Which leads to an overlapping grid of blocks, where the border pixels of each block are used by two blocks. An other reason is the use of L1 cache for global memory by cards with Fermi architecture, which reduces the consecutive load of the same global memory.

Binary Object Mask

As we only have to segment the object, represented by the binary object mask shown in Figure 1.2, there are many pixels in the picture, which don't have to be calculated by our algorithm. In other words, the border of the segmented area is not the border of the picture, but the border of the object. Therefore the primal and dual update CUDA kernels need extra conditional branches to dedicate if a pixel is within an object or not. Those branches are very inefficient in context of computation time on CUDA kernels. We gained as much performance as possible by minimizing the branches in the two CUDA kernels.

4.2.3 Potts Model

Like the Ising model, also the Potts model is implemented in CUDA C where each update step is implemented in a CUDA kernel. We used an existing implementation from [39, 11] of a Potts model. But we extended the implementation by a label prior as defined in Equation (3.42). An other major difference to the implementation of [39, 11], is the use of different convergence criteria, which will be discussed accurately in the next chapter.

Enhanced use of L1 Cache

Further more we have chosen to not implement the shared memory enhancement as for the Ising model, but to leaf the additional memory for the L1 cache. At last the texture memory, which is used by the original implementation has gained no performance increase, quite the contrary was true, because L1 cache works faster than texture memory. So the texture memory usage was canceled by our implementation. The extra conditional branches to include the binary object mask where included with respect to keep as much performance as possible.

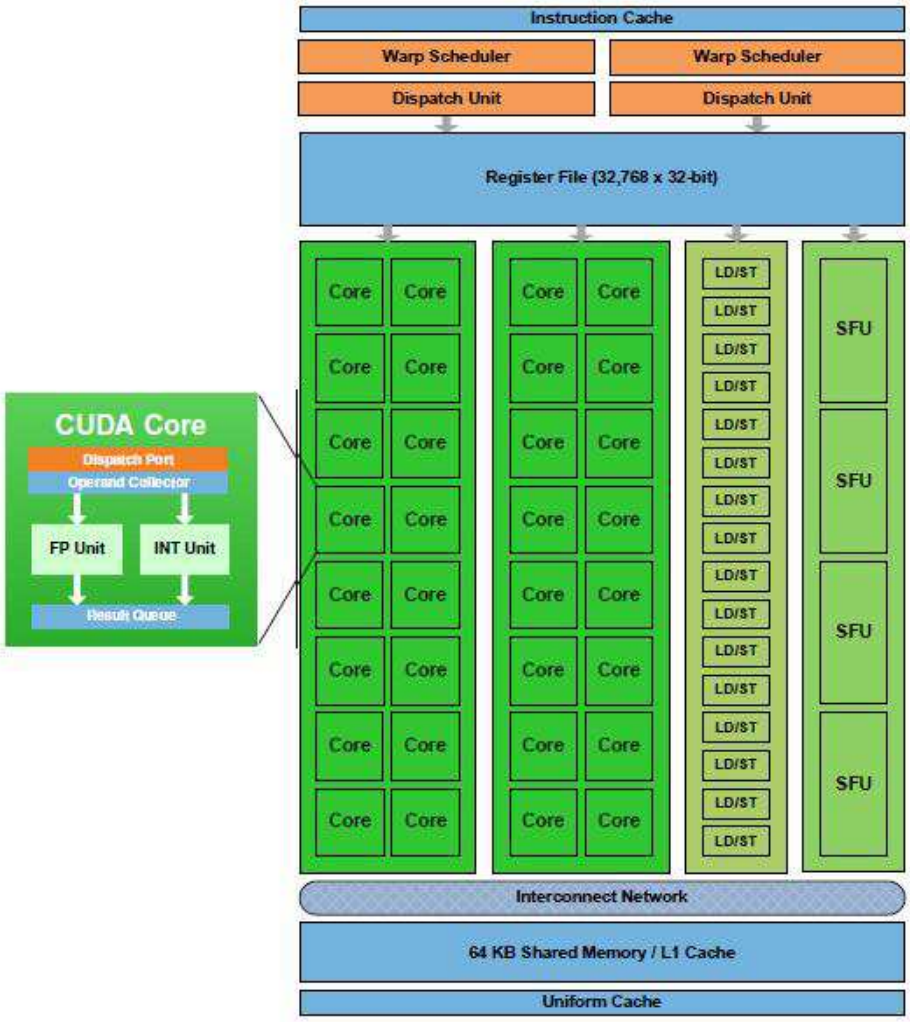


Figure 4.2: A reprint from [16] which shows a streaming multiprocessor with its 32 cores and 16 load/store units. As well as other features like the on-chip L1/shared memory.

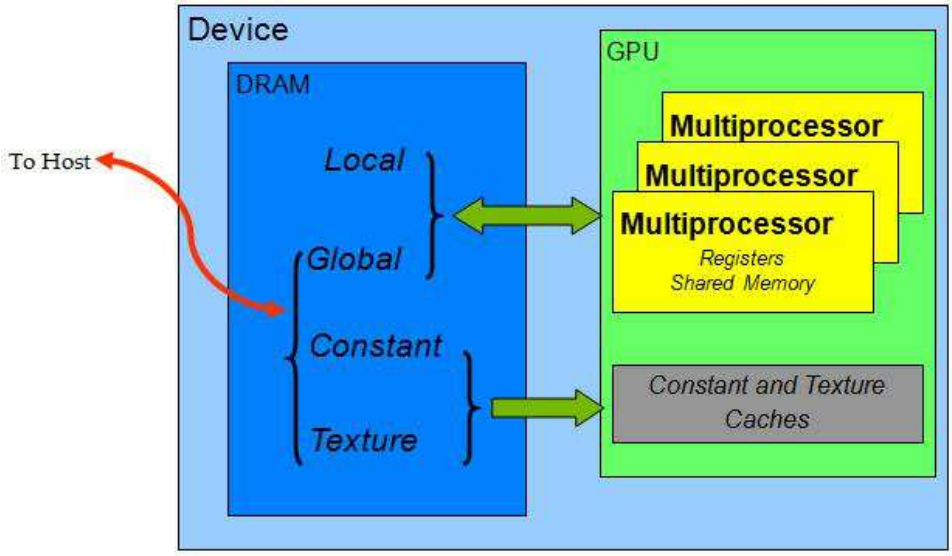


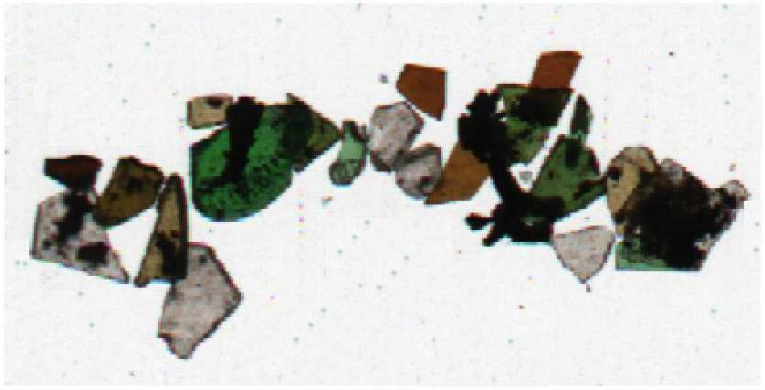
Figure 4.3: A reprint from [17] which shows the location of all sorts of memory available by the kernel functions.

Chapter 5

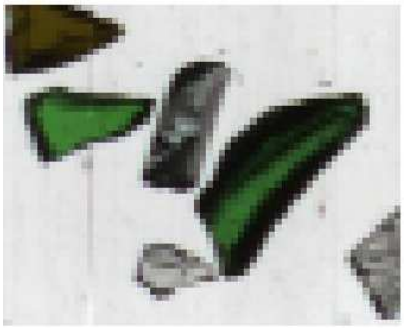
Experimental Results

5.1 Experimental Environment

We have set up 2 different experimental environments, which are shown in Figure 5.3. The first picture, it is taken from a machine working in a real glass sorting environment, will be used to investigate the performance of the Potts and watershed algorithms. The second picture, taken from our test-machine, will be used to investigate the performance of the Ising model. As the Ising model only uses 2 labels, it is better suited for applications where a small amount of extraneous material have to be sorted out, which is given by the second picture. To analyze the maximum time that an object takes to be computed, we show the computation time of the largest objects in each environment separately. These objects are shown in Figure 5.1. Both pictures have a ground truth which was manually implemented. Opposite to other works our ground truth doesn't define the segmented objects, but the band in which the cut of the connecting objects should go trough. Figure 5.2 shows the different types of cut bands, which can be used to evaluate the proposed algorithms. As the separation of objects who correspond to the same type of glass doesn't have to be done, this separation is optional and isn't counted as error if it isn't separated.



(a) Largest object in picture 1



(b) Largest object in picture 2

Figure 5.1: Largest objects of the two proposed evaluation environments. (a) shows the largest object in picture 1, which contains over 20 objects. (b) shows the largest object in picture 2, which contains 4 objects.

(a) Yellow band

(b) Purple band

Figure 5.2: The yellow band defines the area, in which two objects of different class have to be separated. The purple band defines the area, in which two objects of the same class can be separated.

(a) Picture 1

(b) Picture 2

Figure 5.3: Picture 1 (1100×5000) shows a two second snapshot from a real working machine. Picture 2 (1100×1500) shows a 0.6 seconds snapshot taken from our test machine.

5.1.1 Evaluation Metrics

We used two metrics to measure the relevance of our segmentation, called precision (Pre) and recall (Rec).

$$Rec = \frac{Th(\sum_{i=0}^{\#SepObj} \frac{CPO(SepObj,GTObj)}{CPO(GTObj,GTObj)})}{\#SepObj} 100 \tag{5.1}$$

$$Pre = \frac{Th(\sum_{i=0}^{\#GTObj} \frac{CPO(SepObj,GTObj)}{CPO(SepObj,SepObj)})}{\#GTObj} 100 \tag{5.2}$$

Where the function $CPO(mask1, mask2)$ counts the number of pixels that occur in both masks. $SepObj$ is the mask of the separated object and $GTObj$ is the mask of the ground truth object. The constants $\#SepObj$ and $\#GTObj$ define the number of separated, or ground truth objects, in a picture. We have set up an threshold function $Th()$, which defines which object is acceptable. In all following evaluations we have set the threshold to 0.9, such that 9 out of 10 pixels of an object have to be assigned correct. Recall is a metric for over-segmentation, which means that an object is separated incorrectly. It should be noted that the recall is calculated based on the separated objects. In other words for each separated object the recall of this object is calculated, by searching for the ground truth object that covers the biggest part of the separated object. This has the advantage that every over-segmented object is counted separately. On the contrary precision is based on the ground truth objects. Precision measures the under-segmentation of an object, which means that an object is connected after the segmentation process. Therefore for each ground truth object the precision is calculated, by searching for the separated object that covers the biggest part of the ground truth object. Here every under-segmented object is counted separately which results in an accurate measurement.
To provide a summary of the two metrics we use the F-Measure function.

$$FMeasure = \frac{2RecPre}{Rec + Pre} \tag{5.3}$$

5.1.2 Quality of Input Images

At first we want to investigate the quality of the input images in order to get an overview of how much objects are connected, and how good our proposed models work in contrast to no separation. Table 5.1 shows the results of picture 1 and picture 2 without any object separation. It can be seen that about 36% of the objects are connected objects and even in optimal environments like in picture 2 about 20% of the objects are connected objects, which belong to different output streams. As there is no object separation there is no over-segmented object, which results in a 100% recall. This shows how important object separation is in industrial glass sorting systems. It should be noted that all evaluations with picture 1 have been established with a preliminary erosion with a 3×3 mask, because objects at this size will also be eroded in the REDWAVE software.

Table 5.1: Precision and recall of picture 1 and picture 2 without proposed algorithms

	Picture 1	Picture 2
Precision	63.71	82.02
Recall	100	100

5.2 Watershed Segmentation

This section determines, how efficient the watershed segmentation algorithm 2 works. As this algorithm is standard and there are many existing implementations, we concentrate on the performance of the algorithm without concentrating on the computation time.

5.2.1 Determining Distance Transformation

One big influence to the results of the watershed segmentation is the use of the metric for the distance transformation, defined in Equation 3.1. We have investigated the performance of some standard metrics, described below.

$$dist_1(x,y) = |x_1 - y_1| + |x_2 - y_2| \quad (5.4)$$

$$dist_2(x,y) = \max(|x_1 - y_1|, |x_2 - y_2|) \quad (5.5)$$

$$dist_3(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad (5.6)$$

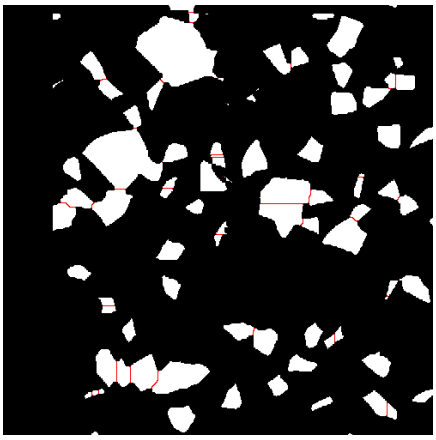
Where $dist_1(x,y)$ is called cityblock distance, $dist_2(x,y)$ is called chessboard distance and $dist_3(x,y)$ is called euclidean distance. An example that shows the difference between the different distance transformations according to the object separation results can be found in Figure 5.4. Table 5.2 shows the precision and recall results of the watershed segmentation with different distance transformations. It can be seen that $dist_1(x,y)$ achieves the highest result.

Table 5.2: Precision and recall of watershed segmentation with different distance transformations

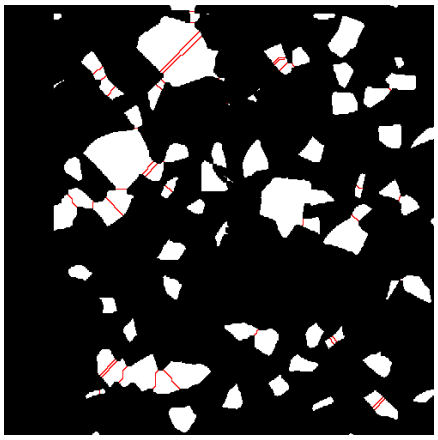
	$dist_1(x,y)$	$dist_2(x,y)$	$dist_3(x,y)$
Precision	92.59	91.19	95.50
Recall	73.48	67.51	43.88

5.2.2 Enhanced Watershed Algorithm

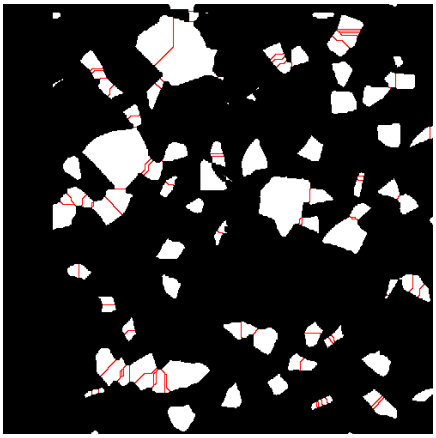
As described in Algorithm 2 we have proposed an additional step which excludes a part of the watershed lines from being separation lines. This section analyzes the influence of this additional parameters to the quality of the segmentation. First we calculate the concavity approximation by using a 13×13 circular mask and vary the threshold c_1 defined in Algorithm 2.



(a) Cityblock Distance Transform



(b) Chessboard Distance Transform



(c) Euclidean Distance Transform

Figure 5.4: Influence of different distance transformations to the watershed object separation. (a) Performance of cityblock distance transformation, which provides the best results. (b) Performance of chessboard distance transformation. (c) Performance of euclidean distance transformation.

Table 5.3 shows, that a much higher recall can be achieved by a little performance lost on precision. The second Table 5.4 analyzes the threshold c_2 which enables very small watershed lines to be separation lines, regardless to the concavity of their endpoints. Figure 5.5 shows the result of the enhanced watershed segmentation, which can be compared to the results of the standard watershed segmentation shown in Figure 5.4. The varying quality of the different watershed parametrization are shown in the precision/recall plot of Figure 5.6.

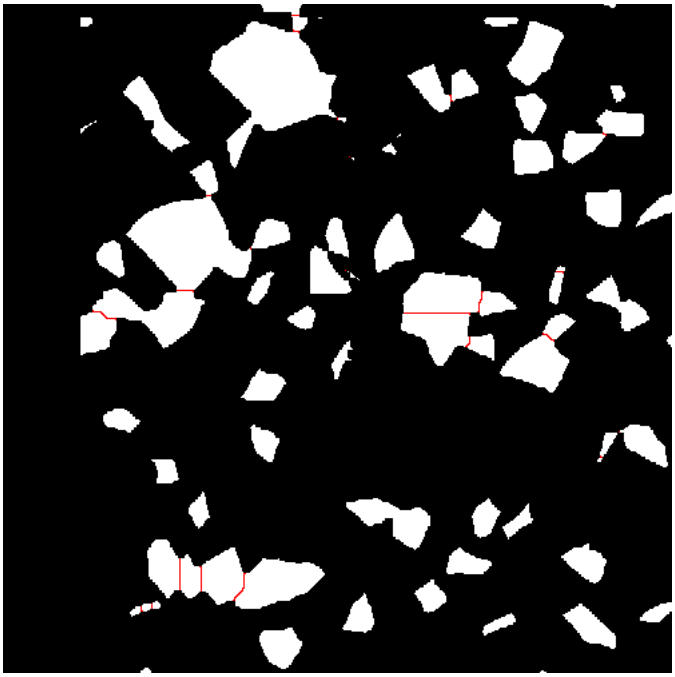


Figure 5.5: This picture shows the performance of the enhanced watershed segmentation, with cityblock distance transformation.

5.3 Potts Model

To obtain an overview of the quality of our Potts model, we have performed a set of tests which will be provided in the next section. First of all we focus

Table 5.3: Precision and recall of watershed segmentation with additional concavity constraint

	$c_1 = 0.5$	$c_1 = 0.55$	$c_1 = 0.6$
Precision	84.44	**79.38**	75.44
Recall	86.14	**94.50**	98.07

Table 5.4: Precision and recall of watershed segmentation with additional length constraint

	$c_2 = 5$	$c_2 = 7$	$c_2 = 10$
Precision	85.84	87.07	89.22
Recall	93.30	92.06	89.81

on the quality, regardless to the computation time. The second section will discuss the time issue and provide an overview of how to boost it. At last we will give an advise of how to find a tradeoff between the two diverging requirements. As the Potts model returns a continuous segmentation result u, we use a simple argmax to obtain a discrete segmentation result.

$$\dot{u}(x) = \text{argmax}(u_1(x), u_2(x), \ldots, u_K(x)) \tag{5.7}$$

Furthermore we initialize $f_l(x) = 0$, if the picture contains the class l at position x and $f_l(x) = 1$ else, as defined in Equation 3.6.

5.3.1 Segmentation Quality

Determining λ

First of all we varied λ, which defines a tradeoff between a smooth segmentation boundary and the similarity of the segmentation result to the input image. A lower λ defines a smoother segmentation. As it is shown in Table 5.5 and Figure 5.7 a precision of 90% and a recall of 96% can be achieved by $\lambda = 0.15$, which results in the highest F-Measure. An other result of

this evaluation is, that lambda regulates the tradeoff between over- and under-segmentation. At a lower lambda there is nearly no over-segmentation whereas a higher lambda means lesser under-segmentation.

Table 5.5: F-Measure of varying λ

λ	0.01	0.05	0.1	**0.15**	0.2	0.4
F-Measure	80.78	88.37	92.26	**93.41**	92.14	91.22

Table 5.6: F-Measure of varying γ

γ	1	**5**	10	15	25	35	50
F-Measure	92.19	**93.41**	92.07	91.61	91.44	90.95	89.94

Determining γ

The second test deals with the influence of γ, which defines the weight of the label cost term. Table 5.6 shows us that the influence of gamma to the segmentation result is very low.

5.3.2 Performance Measure

Label Convergence with Label Cost Term

As it is shown in Equation (3.42), we added an additional label cost term to our Potts model, in order to determine the importance of a label. As a consequence of the term an unimportant label can be excluded from calculation, which should result in a faster performance of our algorithm. Because in the later iterations, only the data of the important labels have to be updated. The label cost constraint from Equation (3.41) shows that the variable y_l is greater than any $u_l(x)$. Therefore we have added a simple constraint, that stops the calculation of p_l, u_l, q_l, y_l for every l where $y_l < c$. If we choose that $c = \frac{1}{2k}$, the quality of our segmentation is exactly the same as in Table 5.6,

but the computation time of our segmentation is nearly 2 times faster, for details see Table 5.7.

Table 5.7: Performance of label convergence with label cost term

c	no criterion	$c = \frac{1}{2k}$
Picture in sec.	62.53	**34.26**
Largest Obj. in sec.	2.83	**2.18**

Label Convergence without Label Cost Term

The major drawback of the label convergence is the necessity of calculating the label cost term and with it the Lagrange multiplier q_l, which costs a lot of computation time. As there is only little benefit in segmentation quality, according to Table 5.6, we tried to exclude the label cost term from our model. With this modification the computation time was much faster and the segmentation quality stayed the same at 93.41 (F-Measure).

On the other side we have developed a work around for the label convergence, which works without the label cost term. Therefore we have developed a similar convergence criterion.

$$\tilde{y}_l^n = \frac{\sum_x |u(x)_l^n - u(x)_l^{n-t}|}{\#xt} \tag{5.8}$$

Where t is a predefined step of iterations and $\#x$ is the number of pixels of the object. If $\tilde{y}_l^n < c$ the calculation of p_l, u_l, y_l stops. As the computation of $\tilde{y}_l^n$ is only for convergence checking, it can be done at regular intervals, which decreases the computation time. If we choose that $c = 2e - 07$ and $t = 50$, with checking intervals of 50, the quality of our segmentation is exactly the same at 93.41 (F-Measure) and the computation time is 2 times faster, see Table 5.10.

Table 5.8: Performance of Label Convergence without Label Cost Term

c	no criterion	$c = 2e - 07$
Picture in sec.	4.61	**2.38**
Largest Obj. in sec.	0.122	**0.050**

Convergence Criteria

Convergence Criterion 1 All previous sections have used the same convergence criterion which is defined as:

$$\chi = \frac{\sum_{l=1}^{k} \sum_{x} |u(x)_l^n - u(x)_l^{n-t}|}{\#xtk} \tag{5.9}$$

This is the summation over all labels of the label convergence criterion defined in Equation 5.8. The computation of our Potts model stops, if $\chi < c$. Table 5.9 shows the results with different c.

Table 5.9: Performance of Convergence Criterion 1

c	$c = 1e - 10$	$c = 1e - 06$	$c = 1e - 05$	$c = 1e - 04$	$c = 1e - 03$	$c = 1e - 02$
F-Measure	93.41	93.41	**93.41**	91.72	89.55	86.73
Picture in sec.	2.39	2.31	**2.15**	1.65	1.00	0.84
Largest Obj. in sec.	0.05	0.05	**0.05**	0.032	0.009	0.005

Convergence Criterion 2 As the result of our segmentation is simply the argmax of u, as mentioned in Equation 5.7, we don't have to continue the computation until u doesn't change any more. Therefore we have installed a second convergence criteria which allows us to stop the computation earlier, which is shown in Table 5.10.

$$\xi = \sum_{x} \phi_{u^n, u^{n-t}}(x) \tag{5.10}$$

$$\phi_{u^n, u^{n-t}}(x) = \begin{cases} 1 & \text{if } \operatorname{argmax}(u_1^n \dots u_K^n) = \operatorname{argmax}(u_1^{n-t} \dots u_K^{n-t}) \\ 0 & \text{else} \end{cases} \tag{5.11}$$

With this criteria we stop the computation, if $\xi = 0$, so the only parameter is t, which has to be large enough to not stop too early.

Table 5.10: Performance of Convergence Criterion 2

t	200	**100**	50	25
F-Measure	93.41	**93.41**	91.91	91.85
Picture in sec.	2.58	**1.73**	1.28	1.09
Largest Obj. in sec.	0.042	**0.039**	0.034	0.041

Initialization

Because the Potts model, proposed in this paper, provides a global solution the initialization of u is unimportant related to the quality of the segmentation results. But it is not unimportant related to the computation time. Therefore we have tried different initializations of u which results in different computation times, that are shown in Table 5.11. Where

$$\begin{aligned} \hat{u} &= 1 - f \\ \breve{u}_l &= \begin{cases} 1 & \text{if l is most frequent used class of object} \\ 0 & \text{else} \end{cases} \\ \bar{u}_l &= \frac{1}{K} \end{aligned}$$

are the different initializations of u. The results show quite different computation times for the different initializations. The major benefit in time consumption is achieved by $\breve{u}$, because it is already the result for objects that don't have to be separated. So those objects have a computation time of less than 1ms with the drawback of higher computation time for larger objects that have to be separated.

Table 5.11: Initialization of u

u	$\hat{u}$	$\breve{u}$	$\bar{u}$
Picture in sec.	2.57	1.73	2.50
Largest Obj. in sec.	0.027	0.039	0.029

Resize the Object

The next step was to reduce the object size by simply resizing the input matrizes, in order to gain an extra computation time boost. With $\lambda = 0.3$ and a resize factor of 0.5, we have gained an F-Measure of 90.02 with a recall of 90.18 and a precision of 89.68. Which shows that the results of computing the Potts model with lesser information are respectable. The computation time of the largest object in the picture was 5 times faster and lasted only 0.007 seconds and even the computation time of the whole picture has been reduced to 1.33 seconds.

5.3.3 Summary

The results of our evaluation show that there are many ways to achieve a good segmentation with the Potts model, which fulfill the diverging requirements of quality and time. As the computation time of a picture taken in two seconds, is less than two seconds with the right convergence criterion and initialization, the computation of this model is in real-time and achieves very good results as shown in Figure 5.8. The REDWAVE System records the object with a linescan camera, the system has to wait until the whole connecting object has been recorded, which can be larger than the distance between the capturing line and the valves. The current software cuts the object in order to obtain information about the object early enough. This hard cut could be modified by our model, such that all finished objects can be calculated and the rest will be computed later.

5.4 Ising Model

As mentioned in the previous chapters the Ising model only works on 2 labels, and therefore can not be used with the classified representation as proposed in Figure 1.2. But a standard REDWAVE machine can only split the input material stream in two output material streams. Therefore each color class wherewith a pixel is labeled has a parameter, which defines what output stream the class is part of. So if we label the pixels with the output stream their class is part of, we get a binary image of the objects, which can be used for the Ising model.

5.4.1 Determining λ

First we have analyzed the influence of λ to the quality of the object separation. As one can see in Table 5.12, the F-Measure is highest at $\lambda = 0.1$, with a recall of 91.50 and a precision of 94.93. Figure 5.10 shows the precision/recall plot, which implies that a higher λ yields a higher precision with loss of recall. That conforms with our model, because a higher λ implies, that the resulting segmentation has to be more similar to the highly fragmented input.

Table 5.12: F-Measure, with varying λ, of the Ising Model

λ	0.01	0.05	**0.1**	0.15	0.2	0.4
F-Measure	89.61	92.26	**93.18**	92.47	91.02	60.29

5.4.2 Convergence Criterion

As discussed in the Potts model evaluation, we use the label changed convergence criterion, as described in Equation 5.10 for the Ising model, too. Therefore we have to analyze the quality and performance of the time step, which is shown in Table 5.13. The best performance with nearly constant quality is achieved with $t = 25$ and the time, used for the calculation is

Table 5.13: Performance of Convergence Criterion of the Ising Model

t	200	100	50	**25**	15
F-Measure	93.18	93.18	92.98	**92.99**	93.13
Picture in sec.	1.76	1.03	0.68	**0.56**	0.57
Largest Obj. in sec.	0.008	0.005	0.006	**0.0078**	0.0077

lesser than the time, which is used to take the picture. So the Ising model is fast enough to perform in real-time. As shown in the Potts model evaluation, we have also used the initialization of $\breve{u}$ shown in Equation 5.12. The performance result of the Ising model on our test picture 2 is shown in Figure 5.11.

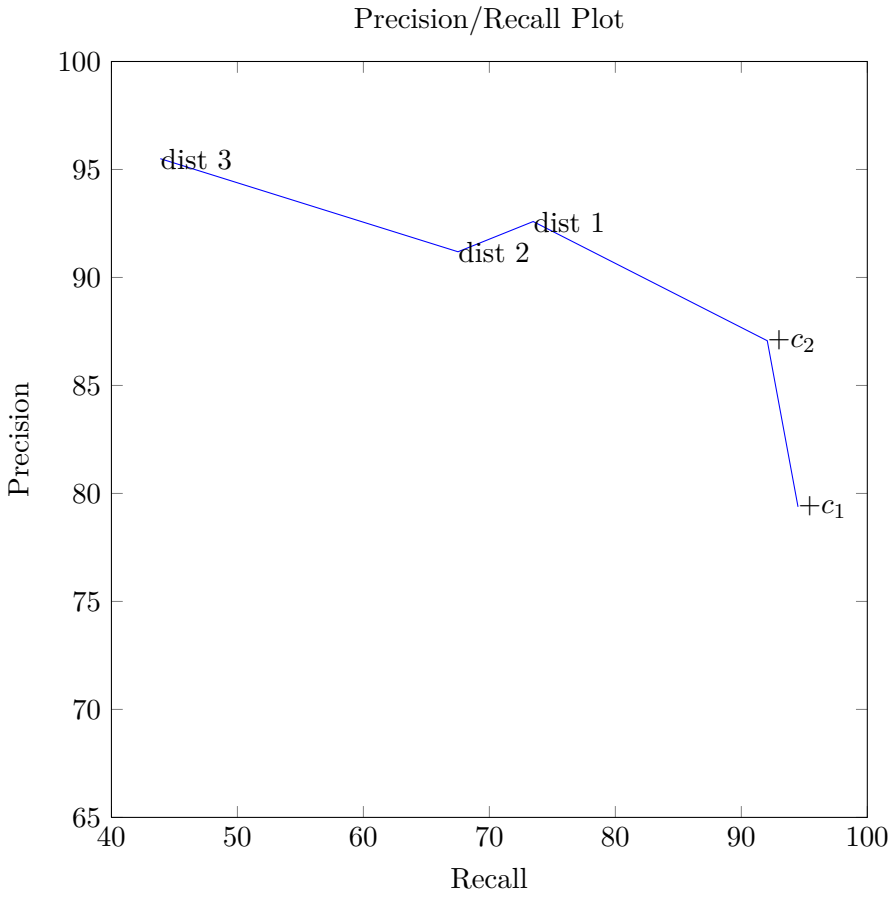


Figure 5.6: Precision/Recall plot of wathershed segmentation. Where dist 1 to dist 3 describes the precision/recall which can be achieved with different distance transformations. And $+c_1$ describes the result of cityblock distance transformation with additional concavity constraint, where $c_1 = 0.55$. At last $+c_2$ has the same parameter as $+c_1$ with additional length constraint, where $c_2 = 7$.

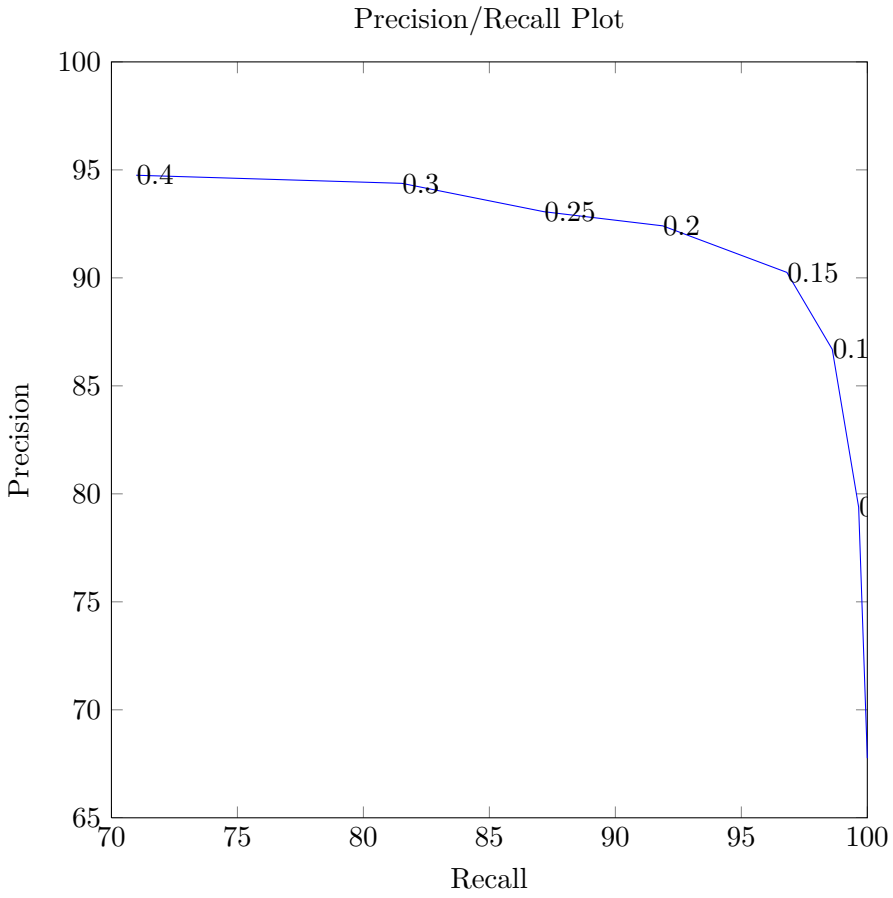
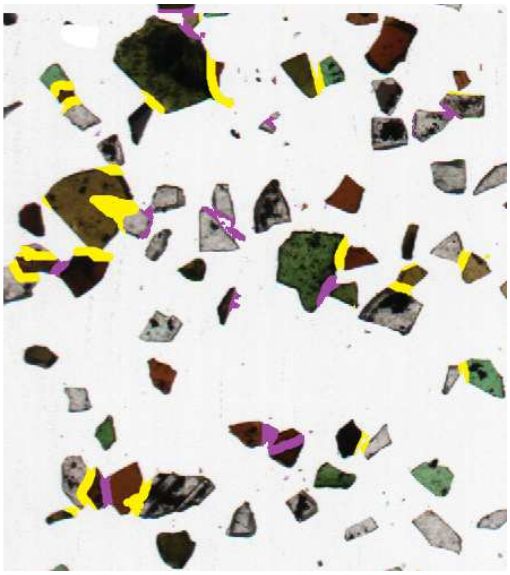


Figure 5.7: Precision/Recall plot of the Potts model with varying λ. At a lower lambda there is nearly no over-segmentation, whereas a higher lambda means lesser under-segmentation.



(a) Detail of Picture 1

(b) Pseudo Color Result

Figure 5.8: Result of the Potts model. The resulting object separation performed with the Potts model in pseudo color. Where every object has its own color coding ($\lambda = 0.15$).

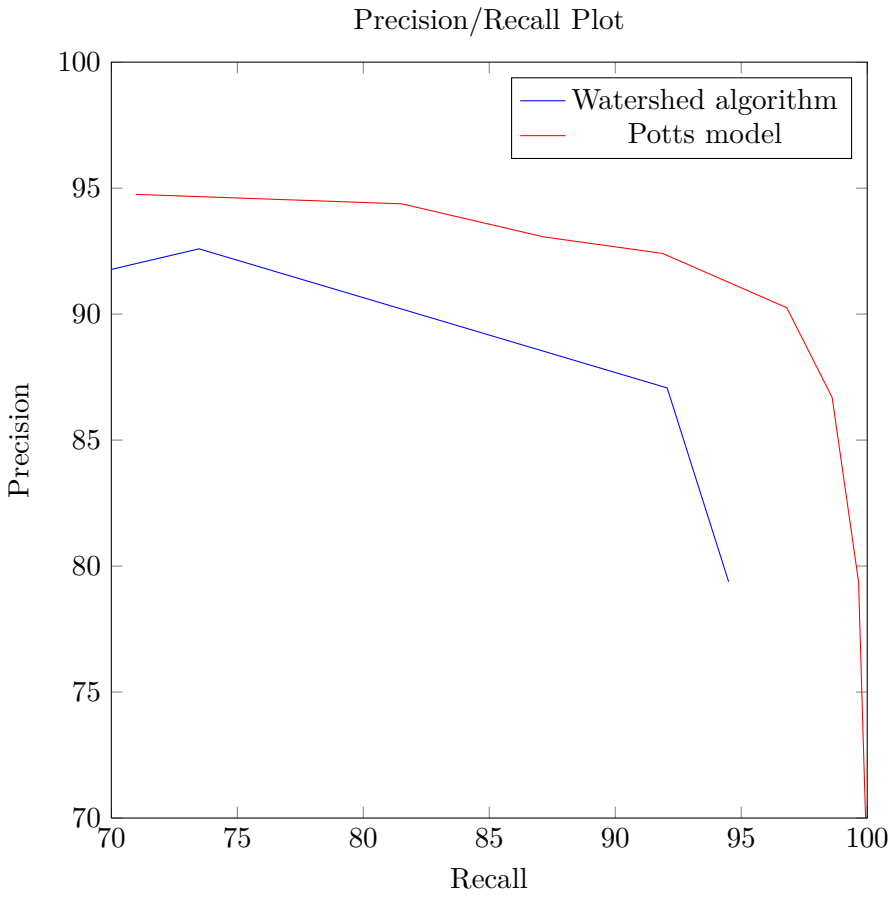


Figure 5.9: This plot shows the performance gain achieved with the Potts model, in contrast to the watershed algorithm.

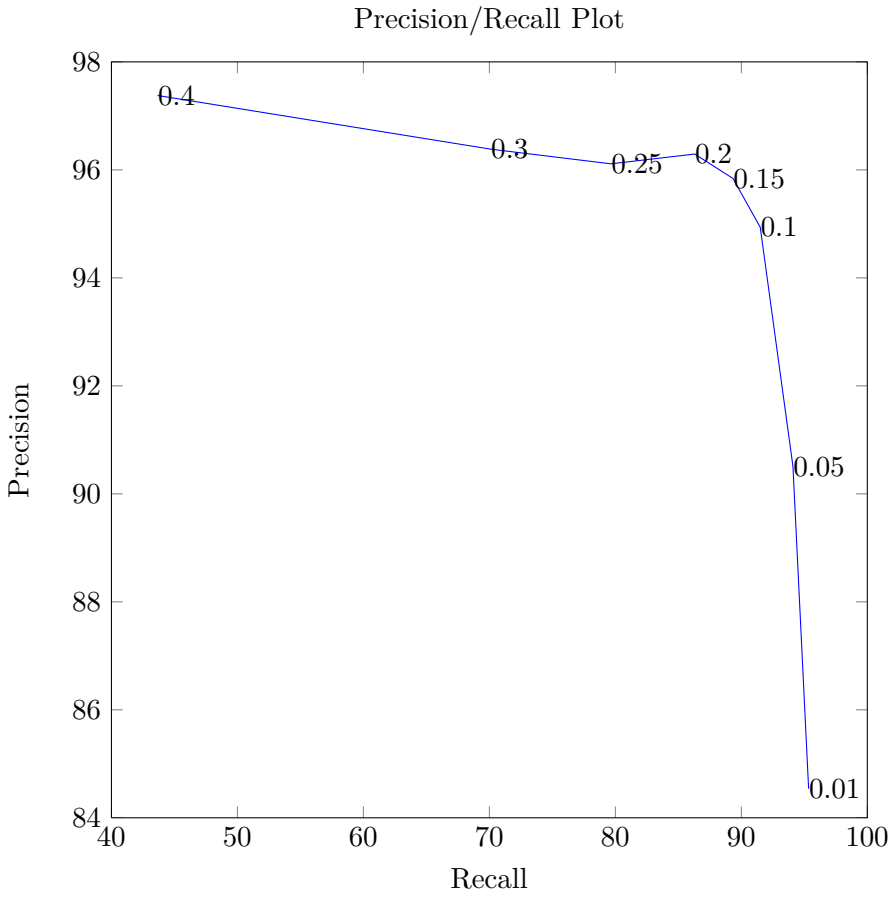
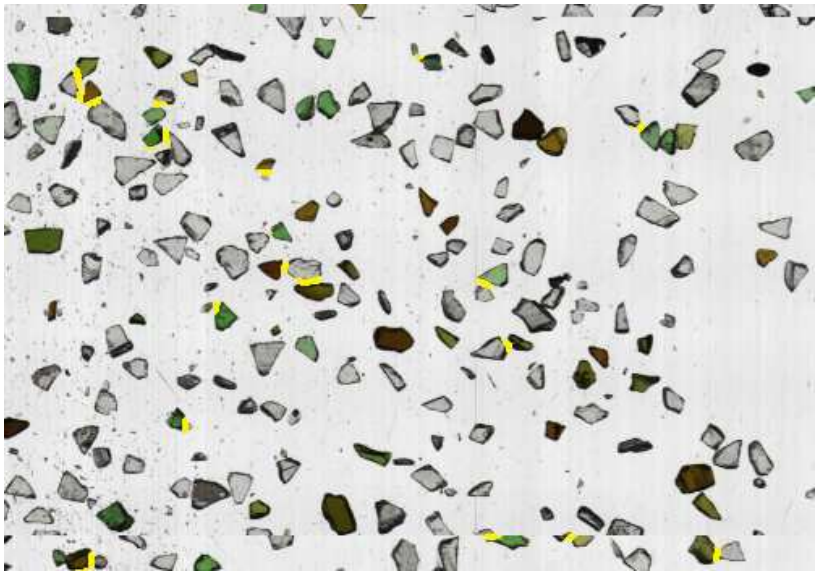
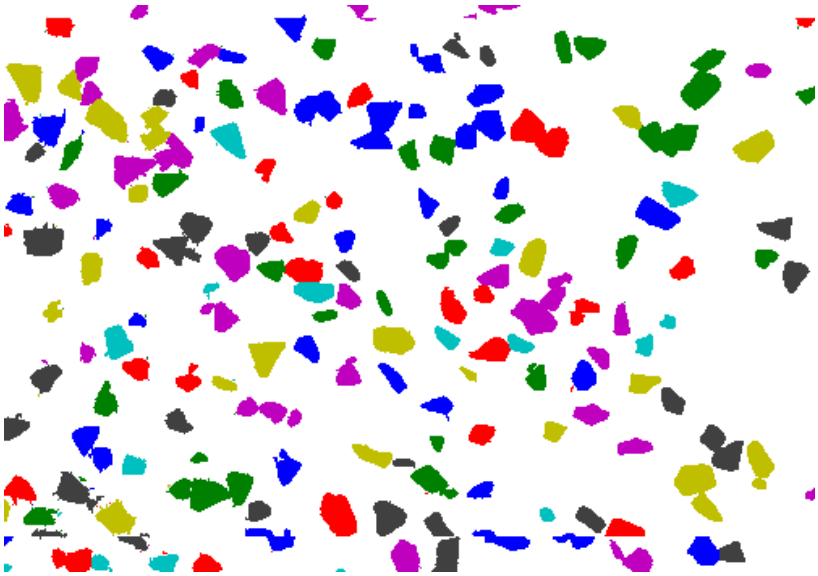


Figure 5.10: Precision/Recall plot of the Ising model with varying λ. At a lower lambda there is nearly no over-segmentation, whereas a higher lambda means lesser under-segmentation.



(a) Detail of Picture 2



(b) Pseudo Color Result

Figure 5.11: The resulting object separation performed with the Ising model in pseudo color. Where every object has its own color coding. ($\lambda = 0.10$)

Chapter 6

Conclusion and Outlook

6.1 Conclusion

We have shown, how object separation can be established in order to perform in real-time. Chapter 1 has shown the construction of a REDWAVE, which is the industrial machine where our proposed algorithms should run on. Further more we have provided a short overview of our algorithms. In the second chapter we have shown works that are similar to our proposed algorithms, or could have been used to achieve our goals. We have illustrated the ideas behind this works and what their drawbacks are. Chapter 3 has given a detailed description of our three proposed algorithms. The first algorithm we have proposed, is the well known watershed transformation segmentation, which we have used to separated the objects by applying it to a binary representation of the objects. The algorithm works quite well, because of the observation that connected objects have often concave contours, whereas a single object is nearly convex. To minimize the number of watershed lines we only allow watershed lines to separate objects if they have concave endpoints. The other two algorithms which we have proposed, are very similar and differ only in the number of different labels they can work with. Where the Potts model can work with k labels, the Ising model

only works with two labels. We have defined both models in a spatially continuous domain, such that the models can be solved with variational methods. In order to provide global solutions to our models, we have used simple convex relaxations. After all we have proposed the use of the primal dual algorithm to obtain an algorithm, that can be easily executed parallel and also converges with a small amount of iterations. Chapter 4 illustrates the CUDA framework and the structure of an Nvidia GPU with Fermi architecture. Further more we have described how we have implemented the algorithms in concern of using the GPU with high performance. In the last Chapter of our work we have evaluated the different algorithms and have shown what initializations, convergence criteria and performance boosts can be installed in order to achieve a real-time performance without loosing segmentation quality. Further more we have shown the impressive performance of the proposed Potts model and the fast and also respectable performance of the Ising model.

6.2 Outlook

We recommend to use the proposed Potts model to achieve a higher performance with the REDWAVE system. The next steps should be to implement the algorithm in the system and perform some tests in real industrial environments which will quantify the performance boost. In order to prohibit that very big objects take too much computation time, different solutions can be implemented. We recommend to apply the algorithm on parts of the connected object and cut away all finished parts, as described in the evaluation summary in Section 5.3.3. An other possibility is to measure the time at predefined iteration steps and exit the computation, if it lasts too long. Also the Ising model performs very good and should be used if it is possible.

Bibliography

[1] Adrien Angeli Ankur Handa, Richard A. Newcombe and Andrew J. Davison. Applications of legendre-fenchel transformation to computer vision problems. Technical Report DTR11-7, Imperial College - Department of Computing, September 2011.

[2] S. Beucher and C. Lantuejoul. Use of Watersheds in Contour Detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France.*, September 1979.

[3] S. Beucher and Centre De Morphologie Mathmatique. The watershed transformation applied to image segmentation. In *Scanning Microscopy International*, pages 299–314, 1991.

[4] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Xavier Bresson, Selim Esedoglu, Pierre Vandergheynst, Jean-Philippe Thiran, and Stanley Osher. Fast global minimization of the active contour/snake model. *J. Math. Imaging Vis.*, 28(2):151–167, June 2007.

[7] J.L. Carter. *Dual Methods for Total Variation-based Image Restoration.* University of California, Los Angeles, 2001.

[8] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, February 1997.

[9] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. In *Theoretical Foundations and Numerical Methods for Sparse Recovery*. De Gruyter, 2010.

[10] Antonin Chambolle. An algorithm for total variation minimization and applications, 2004.

[11] Antonin Chambolle, Daniel Cremers, and Thomas Pock. A convex approach to minimal partitions. *preprint*.

[12] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *preprint*.

[13] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 10(2):266–277, February 2001.

[14] Tony F. Chan, Selim Esedo Glu, and Mila Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. Technical report, SIAM Journal on Applied Mathematics, 2004.

[15] Qing Chen, Xiaoli Yang, and E.M. Petriu. Watershed segmentation for binary images with different distance transforms. In *Haptic, Audio and Visual Environments and Their Applications, 2004. HAVE 2004. Proceedings. The 3rd IEEE International Workshop on*, pages 111 – 116, oct. 2004.

[16] NVIDIA Corp. Whitepaper nvidia's next generation cuda compute architecture: Fermi. Technical report, NVIDIA Corp., 2009.

[17] NVIDIA Corp. Cuda c best practices guide. Technical Report DG-05603-001-v4.1, NVIDIA Corp., January 2012.

[18] NVIDIA Corp. Nvidia cuda c programming guide. Technical Report V4.2, NVIDIA Corp., Mai 2012.

[19] NVIDIA Corp. Nvidia cuda getting started guide for microsoft windows. Technical Report DU-05349-001-v04, NVIDIA Corp., April 2012.

[20] L. C. Evans and R. F. Gariepy. *Measure Theory and Fine Properties of Functions*. CRC-Press, 2000.

[21] WendellH. Fleming and Raymond Rishel. An integral formula for total gradient variation. *Archiv der Mathematik*, 11:218–222, 1960.

[22] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, February 1925.

[23] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[24] Can-Fei LI, Yao-Nan WANG, and Guo-Cai LIU. A new splitting active contour framework based on chan-vese piecewise smooth model. *Acta Automatica Sinica*, 34(6):659 – 664, 2008.

[25] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[26] Stanley Osher and Ronald P. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys*, 169:463–502, 2001.

[27] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *JOURNAL OF COMPUTATIONAL PHYSICS*, 79(1):12–49, 1988.

[28] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *International Conference on Computer Vision (ICCV 2011)*, 2011. To Appear.

[29] Thomas Pock, Antonin Chambolle, Daniel Cremers, and Horst Bischof. A convex relaxation approach for computing minimal partitions. In *CVPR*, pages 810–817. IEEE, 2009.

[30] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. An algorithm for minimizing the mumford-shah functional. In *IEEE International Conference on Computer Vision (ICCV)*, 2009. to appear.

[31] R. B. Potts. Some generalized order-disorder transformations. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48(01):106–109, 1952.

[32] R. T. Rockafellar. *Convex Analysis (Princeton Mathematical Series)*. Princeton Univ Pr, 1970.

[33] Jos B. T. M. Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies, 2000.

[34] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, November 1992.

[35] P. K. Sahoo, S. Soltani, A. K.C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Comput. Vision Graph. Image Process.*, 41(2):233–260, February 1988.

[36] Michalis A. Savelonas, Eleftheria A. Mylona, and Dimitris Maroulis. Unsupervised 2d gel electrophoresis image segmentation based on active contours. *Pattern Recogn.*, 45(2):720–731, February 2012.

[37] Wade Schwartzkopf, Brian L. Evans, and Alan C. Bovik. Minimum entropy segmentation applied to multi-spectral chromosome images. In *ICIP (2)*, pages 865–868, 2001.

[38] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[39] Markus Unger. *Convex Optimization for Image Segmentation*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria, October 2012.

[40] Markus Unger, Manuel Werlberger, Thomas Pock, and Horst Bischof. Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[41] Luminita A. Vese and Tony F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50:271–293, 2002.

[42] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, June 1991.

[43] Hui Wang, Hong Zhang, and Nilanjan Ray. Clump splitting via bottleneck detection. In Benoît Macq and Peter Schelkens, editors, *ICIP*, pages 61–64. IEEE, 2011.

[44] Xiao Feng Wang and De-Shuang Huang. A novel multi-layer level set method for image segmentation. *J. UCS*, 14(14):2427–2452, 2008.

[45] J. M. White and G. D. Rohrer. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM J. Res. Dev*, 27:400–411, 1983.

[46] Jing Yuan and Yuri Boykov. Tv-based multi-label image segmentation with label cost prior. In Frédéric Labrosse, Reyer Zwiggelaar, Yonghuai Liu, and Bernie Tiddeman, editors, *BMVC*, pages 1–12. British Machine Vision Association, 2010.

[47] Christopher Zach, David Gallup, Jan michael Frahm, and Marc Niethammer. Fast global labeling for real-time stereo using multiple plane sweeps, 2008.

[48] Qufa Zhong, Ping Zhou, Qingxing Yao, and Kejun Mao. A novel segmentation algorithm for clustered slender-particles. *Computers and Electronics in Agriculture*, 69(2):118 – 127, 2009.