Efficient probability distribution function estimation for energy based image segmentation methods

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades
einer Doktorin der Ingenieurwissenschaften
genehmigte Dissertation

vorgelegt von

Diplom-Ingenieurin,
Ines Dedović
aus Sarajevo, Bosnien und Herzegowina

Berichter:
Univ.-Prof. Dr.-Ing. Jens-Rainer Ohm
Univ.-Prof. Dr.-Ing. Dorit Merhof
Tag der mündlichen Prüfung: 17.03.2016

Contents

Acknowledgment

Here I want to acknowledge the work of people who helped me to do my PhD work and write this thesis. Firstly, I would like to thank the Forschungszentrum Jülich and the RWTH Aachen University for giving me the opportunity and providing me with material and equipment to perform my research.

I want to thank my supervisors within the IBG-2 institute, Dr. Hanno Scharr, Kai Krajsek and Dr. Gregor Huber for offering me a research position. I also want to thank Prof. Dr.-Ing. Til Aach for accepting me as a PhD student of the RWTH University. Prof. Aach, unfortunately, left to another world the following year. Prof. Dr.-Ing. Jens-Rainer Ohm and Prof. Dr.-Ing. Dorit Merhof allowed me to continue my PhD work and agreed to supervise it. For this, I want to thank them as well.

Next, I wish to thank all people involved in the Garnics project and all my colleagues within the IBG-2 for exchanging ideas with me regarding the research topic. Their feedback has helped me in organizing my research and its presentation.

I would like to thank Kai Krajsek for discussions and help in development of our stochastic approach to solving the Ambrosio and Tortorelli functional, presented in Chapter7. The block Gibbs sampler, first introduced to Computer vision in the paper by Krajsek *et al.* [53] was implemented within an edge-detection algorithm. My contribution to this method was derivation of equations and implementation in Matlab. This research has led to two publications by Krajsek, Dedovic, and Scharr [51, 52].

To my family and dear friends I owe time and attention which I was not able to devote them during my PhD. I do, however, hope that my PhD work will somehow be of use to them in the future and this is what gave me an inspiration to start it in the first place and finish it.

Chapter 1

Introduction to the thesis work

1.1 Preface

This work has been devoted to development of image segmentation methods. Image segmentation is a process of partitioning digital images into meaningful regions. Images can be partitioned into two regions, foreground and background, or into a number of regions corresponding to different objects. Examples of image segmentation applications include:

1. Personal cameras and mobile phones have face and expression recognition. Face recognition is also used on personal notebooks on a log-in screen as a replacement for a password. We use mobile phones to read e.g. QR-codes. Text recognition is a usual step after scanning text-documents. All these applications require an image segmentation step.

2. Scene and object categorization divide images based on some keyword: seaside, mountains, etc. They require a segmentation step as well to create some knowledge about an image. Image editing software packages, like Photoshop, GIMP, Corel PhotoPaint *etc.* contain tools for background removal, object separation and repairing damaged images using knowledge gained from undamaged areas (inpainting). These tools are implemented *e.g.* within Microsoft Office as well.

3. The game industry tends to combine, as much as possible, the virtual world with the real one acquiring knowledge of an environment from a camera.

4. In industry, image segmentation is implemented within algorithms for measuring objects, detecting and tracking them, as well as, examining for malfunctions of products. Hand gestures are used to control processes instead of some other software interface tools.

The research was performed for needs of the Garnics project, which is explained in the next section.

1.2 The Garnics project

This work was funded by the Garnics (GARdeNIng with a Cognitive System) project [1]. The Garnics project was coordinated by The Institute of Bio- and Geo-sciences (IBG-2) at the Forschungszentrum Jülich, the institute devoted to Plant Sciences. Its research area, Plant Enabling Technologies, targets accurate measuring of structural and functional plant traits under close-to-natural environmental conditions. The project was aimed at developing methods for sensing plants' structure and growth. Its goal was to develop a system for monitoring and maintenance of sets of plants using a robot-gardener.

The main idea of the Garnics project is to use the channel representation of data within different segmentation methods. Segmentation methods are modified to use channel values as inputs. The research within this thesis, as well as papers by Wallenberg *et al.* [92, 93, 46] are devoted to the task above described.

The image segmentation task tackled with the Garnics project was to locate plants in images, detect leaves and track them over time. Solving this task would enable measuring leaf size and monitoring leaves' and plants' growth. This could enable *e.g.* phenotyping.

Therefore, large amount of research was devoted to development of image segmentation, leaf localization and leaf tracking methods when only 2D color images are given. The goal is to develop fast methods that achieve highly accurate and reliable results when only little input data exists. Robust recognition with little input data can improve results when additional prior knowledge about plant features or background environment is introduced.

This work is devoted to research on plant recognition and leaf localization using only knowledge on their color distribution.

1.3 Novelties presented in the thesis

The main idea of this research is to examine ways to introduce efficient estimation of probability distribution functions to image processing methods. The work follows two directions:

1. it investigates the possibility and advantages of using the channel framework over the kernel density estimation approach within the Chan-Vese model and the Potts model.

2. it proposes the block Gibbs sampler to estimate the probability distribution function of the smooth image and the edge-map of the Ambrosio and Tortorelli functional.

Energy functionals are modified so that the channel framework can be used within data-terms. Traditionally, kernel density estimators (KDE) are used to describe the probability distribution functions in segmentation methods. Theoretically, channel density estimates (CDE), are equivalent to sampled kernel density estimates (see Section 3.4).

[1]Garnics (GARdeNIng with a Cognitive System); FP 7 ICT project no. 247947; website: `www.garnics.eu`

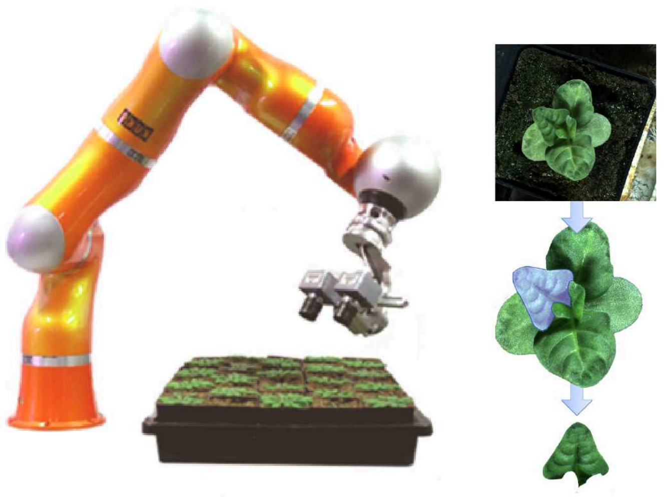


Figure 1.1: The image segmentation task in the Garnics project was plant recognition and leaf detection. The left image presents the robot used within the project. On the right, we have an example of an ideal leaf extraction from a tobacco plant image.

The main difference of the CDE and KDE is how they approach the segmentation problem:

- KDEs are calculated on pixel values on the image and the speed of segmentation depends a lot on the image content.
- For the channel framework look-up tables are precalculated and loaded (see Section 3.6).

As channels are not calculated on values contained in image, basis functions are not positioned exactly on data-points and this can cause slightly worse segmentation results than with KDEs. Therefore, when using the CDE approach, an appropriate look-up table needs to be chosen.

Another novelty presented in this thesis is the stochastic approach to smooth-image and edge-map reconstruction (see Chapter 7). This approach is similar to the one by Geman and Geman [34] with a difference that in this thesis the block-Gibbs sampler is used instead of the pixel-wise one. My contribution within this method is derivation of equations and implementation of the algorithm within MATLAB which was used to obtain and demonstrate the advantages of the idea.

Combination of the results from the region based segmentation and edge-map retrieved from the AT method could allow better differentiation of objects, or further partitioning of regions.

1.4 Organization of the thesis

The thesis is structured in the following way:

- Chapter 2 introduces the problem of image segmentation, gives examples of its application and an overview of state-of-the-art image methods related to the methods presented within the thesis.
- The channel framework is in the core of all image segmentation methods presented within this thesis. Therefore its basics are introduced in Chapter 3.
- Chapter 4 presents the unsupervised 2-region segmentation method that partitions a gray-value image into foreground and background using the Chan-Vese level-set approach. This method can divide an image in two separate regions, but does not allow a user input in defining regions. Difference between this and other state-of-the-art level-set methods is that the channel framework is used to describe the data term.
- Chapter 5 extends the previous method to vector/color images and adds the use of prior knowledge for detecting foreground. Its novelty is the use of the weight coefficient to adjust the trade-off between unsupervised and supervised segmentation.
- Chapter 6 deals with multi-label image segmentation using user inputs for constructing prior knowledge on different objects. In this contest the channel framework is introduced to the Potts model.
- Chapter 7 gives a novel approach for smooth reconstruction of noisy images and to construct their edge-map. It introduces the block-Gibbs sampler to edge-detection methods.
- Chapter 8 summarizes the work presented within the thesis and presents possible steps for further improvements.

The following information is contained in appendices:

- Appendix A describes notation used throughout the thesis.
- Appendix B describes data-sets used for evaluating segmentation algorithms. Besides other, we have created a specific data-set which contains top-view images of tobacco plants. Within this chapter several segmentation evaluation measures are suggested as well.

Chapter 2

Image segmentation

2.1 Introduction

Segmentation is a process of partitioning an image into a number of connected, non-overlapping regions. These regions represent different objects or parts of objects. As a segmentation result pixels of an image are assigned a unique label that indicates which region they belong to.

The segmentation result is expected to be a set of meaningful regions, *e.g.* different objects. Unfortunately, segmentation is an ill-posed problem, which means (according to Hadamard [41, 40]) that the problem may not have a solution, or the solution is not unique, or it does not continuously depend on data. Therefore, it is hard to evaluate how successful segmentation is. The correct segmentation result differs from the task image segmentation is aimed at. Given an image depicting some scene or containing several objects, results can differ depending on criteria used in the algorithm. Several different areas can be segmented containing similar features as foreground objects. An algorithm *e.g.* may choose one area based on its color intensity. A different algorithm can choose a different object, or a part of an object, based on its distinctive texture rather than color.

In supervised image-segmentation methods a user gives hints to what a good result would be. A user *e.g.* selects parts of regions and/or adjusts desired attributes of each region. This improves the segmentation process significantly. However, the development of new technologies aims at automating processes and therefore, introducing as much as possible, human way of thinking in algorithms. This way, less input from a user is needed, most parameters can be left on their default values and processes are sped up.

2.2 Unsupervised image segmentation

Unsupervised image segmentation is segmentation where no prior knowledge on objects or background is available. An image is partitioned in a number of regions that differ from each-other based on some criterion. In Chapter 4 a method is developed that extracts foreground by separating an image in two regions that have different color distributions.

Results of unsupervised segmentation may sometimes seem meaningless to a human

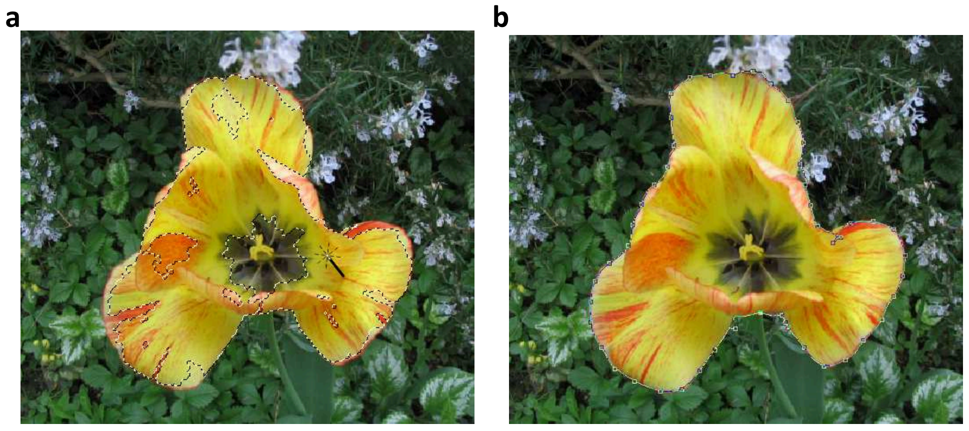


Figure 2.1: Some of selection tools implemented in *Adobe Photoshop CS3* (**a**) Magic Wand; (**b**) Magnetic Lasso

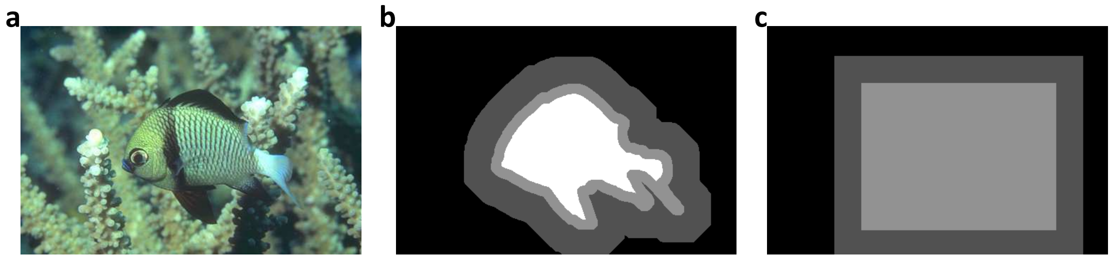


Figure 2.2: Examples of trimaps from GrabCut data-set [76, 9] **a** an image for segmentation; **b** trimaps imitating the use of lasso-tool: black –background pixels, dark gray –background pixels used for training, light gray –unknown pixels, white –foreground pixels; **c** trimaps imitating the use of bounding box for prior knowledge calculation. Gray-scale coding is the same as for lasso trimaps, except the set of foreground pixels is empty.

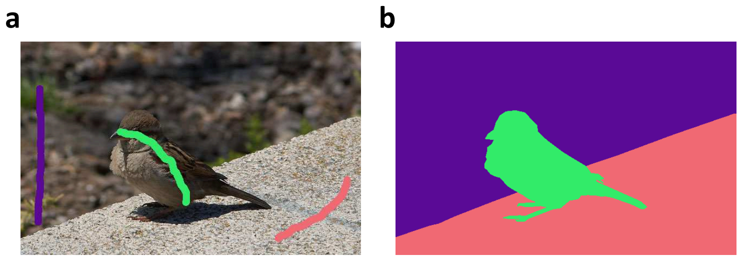


Figure 2.3: An example of using brush strokes for defining prior knowledge on multi-label segmentation. **a** an image from Graz data-set [81] with brush strokes indicating different regions; **b** segmentation result using the approach described in Chapter 6

eye. Having in mind that no hints are used for determining what the foreground object is, results may be quite different from human segmented ones. In the paper by Alpert *et al.* [4] they try to develop a data-set that contains ground-truth segmentation that can be used for evaluating unsupervised segmentation methods. Ground-truth segmentation data are ”correct” segmentation results that are provided by humans or some algorithm that others should refer to.

2.3 Supervised image segmentation

A user can provide hints to segmentation in many ways and some of those are described in following text. Some of these examples are presented also in [76]. *Magic Wand/Magic mask* and *Magnetic Lasso/Intelligent Scissors* are common selection tools in photo editing software. These tools appear in *e.g. Adobe Photoshop, Corel Photo-Paint* and *GIMP*. They divide an image into the active selection and the rest of the image. After selection, the active part of an image is editable, while the rest of the image is preserved. Here, we will present only an idea of these selection tools, and how user interaction is designed.

Magic Wand/Magic mask [1] A user gives only one point as a seed to compute the active selection area. The active area contains then pixels connected to the seed point and with similar color up to some tolerance. Figure 2.1 (**a**) shows the result using *Magic Wand* in *Adobe Photoshop CS3*.

Magnetic Lasso/Intelligent Scissors Here, a user is asked to trace the outline of the desired area. While drawing the outline, the path is refined so that the selection area would match better selected object at borders. As described in [76, 64] this is done by calculating *the minimum cost contour*. Figure 2.1 (**b**) shows the use of *Magnetic Lasso/Intelligent Scissors* in *Adobe Photoshop CS3*.

Trimaps partition an image domain into three regions $\Omega = \{\Omega_f, \Omega_b, \Omega_u\}$. Region Ω_f contains definitive foreground pixels, Ω_b contains background pixels and Ω_u contains pixels that need yet to be decided if they are foreground or background. They are introduced in the paper by Boykov and Jolli [14]: Ω_f contain scribbles that mark foreground, Ω_b contain scribbles that mark background and the rest of the image is an unknown region Ω_u.

Rother *et al.* developed a data-set[2] that contains also trimaps as training data for foreground and background segmentation for needs of their work [76, 9]. The data-set contains trimaps that imitate the bounding boxes selection of objects or the lasso selection of an object. Figure 2.2 gives an example of trimaps from the GrabCut data-set.

The bounding box selection contains a full foreground object and some parts of background. Edges of the bounding box should be as close as possible to borders of the targeted object. A bounding box trimap is made of three regions: $\Omega = \{\Omega_b, \Omega_{bt}, \Omega_u\}$. Ω_b contains pixels that definitely belong to background, Ω_{bt} pixels that are used for training an algorithm what background should be and Ω_u contains pixel that an algorithm has to

[1]https://helpx.adobe.com/photoshop/topics.html

[2]GrabCut data-set: `http://tinyurl.com/grabcut`

classify as foreground or background. The bounding box selection is worth to examine as a selection method because it requires an easy user interaction, just two clicks to define a rectangle and with that a prior knowledge of the object. This makes it practical for fast selection. Figure 2.2 gives an example of trimaps for bounding box selection from the GrabCut data-set.

The lasso selection requires a user to roughly draw a boundary of the targeted object. Lasso trimaps consists of 4 sets of pixels $\Omega = \{\Omega_b, \Omega_{bt}, \Omega_u, \Omega_f\}$. Ω_b, Ω_{bt} and Ω_u are the same as for bounding box trimaps, while Ω_f contains pixels that are definitely foreground and should be used for forming a prior knowledge. Lasso trimaps are used in following works [9, 76]. They are used also in the work by Nieuwenhuis and Cremers [69].

Brush strokes/scribbles are present in papers by Santner *et al.* [81] and Nieuwenhuis and Cremers[69] for providing prior knowledge. Unlike previous approaches presented in this chapter, these methods are not limited to only 2 regions. Instead, labels that denote different regions are assigned to scribbles. It is possible to use a solid brush or airbrush of different sizes. A solid brush requires only its radius to be defined. An airbrush is defined with two parameters: its radius/size and its opacity. Opacity defines the percentage of pixels within the defined radius that are used for creating a brush stroke.

While in Santner *et al.* color and texture information from scribbles is used, Nieuwenhuis and Cremers consider location of scribbles as well and estimate a joint probability distribution of color and space from scribble data. Figure 2.3 shows an example of scribble-driven multi-label segmentation.

Prior knowledge can be calculated also from an another image or a set of images. In Chapter 5, Experiment 4 shows segmentation where the method is first trained on a set of images of leaves and then segmentation is performed over a set of images to recognize plants.

2.4 Energy functionals for image segmentation

2.4.1 The Mumford-Shah functional

In their paper [65], Mumford and Shah present an image as a function of two variables, $g(x, y)$, defined on an image domain Ω with $(x, y) \in \Omega \subset \mathbb{R}^2$. If $g(x, y)$ is a gray-value image, then $g(x, y) : \Omega \to \mathbb{R}$. Generalizations of $g(x, y)$ include additional image features (*e.g.* local orientation or curvature in [20]), color or vector-valued images [22], *i.e.* $g : \Omega \to \mathbb{R}^h$, $h \in \mathbb{N}$.

The image segmentation goal is to appropriately decompose the image domain Ω:

$$\Omega = \Omega_1 \cup \cdots \cup \Omega_n \tag{2.1}$$

so that:

- the image g or, in our case, some property of it is smooth or varies slowly inside each region Ω_i
- the image/some property of image g is discontinuous or varies rapidly across most of the boundary C between the regions Ω_i.

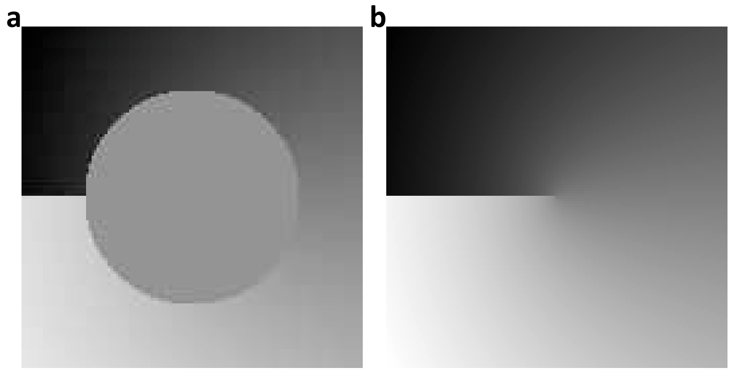


Figure 2.4: The crack tip problem **a** data inside the circle is missing; **b** the global minimum of Mumford-Shah functional [74]

Mumford and Shah formulated the functional to measure similarity between $g(x, y)$ and its smooth reconstruction $u(x, y)$.

$$E(u, C) = \int_{\Omega \setminus C} (u - g)^2 + \lambda |\nabla u|^2 dxdy + \nu |C| \tag{2.2}$$

where C is the edge set and $\lambda, \nu \in \mathbb{R}_+$ denote some regularization parameters.

The first term of the functional (2.2) influences u to be similar to the original image g. The second term influences u to be smooth on the whole image except its borders. The third term is the border length and influences contours to be as short as possible.

In their original paper, Mumford and Shah examine minimization of the functional in special cases, but do not offer a general solution on how to calculate the edge term. In addition, the MS functional is not formulating a convex problem, and therefore does not always have a global minimum.

This gives an opportunity to introduce different approximations of the functional in order to calculate the edge-set. Different modifications lead to different segmentation methods, *e.g.* the Ambrosio and Tortorelli [5], or the Chan-Vese [20] approach. The Potts model [75], although not originated in the MS functional, takes its form in a special case.

The local behavior of the MS functional and its edge set is studied in the original paper [65] and also in studies by Bonnet and David [10, 11] who prove the global minimum of the crack-tip problem for MS functional. In the paper [74], authors design a synthetic image that demonstrates that their algorithm reaches the global minimum in the crack-tip problem. The synthetic image (Fig. 2.4 **a**) is given by $I(x, y) = \sqrt{r(x, y)}\,\text{sin}(\theta(x, y)/2)$ where $r(x, y)$ is the Euclidean distance of a point (x, y) to the image center and $\theta(x, y)$ is the angle of the point (x, y) to the horizontal line. The center part of the image, denoted by the circle, is missing. The global minimum of the MS functional recovers the missing part and provides smooth reconstruction of the image as shown in Fig. 2.4 **b**.

2.4.2 The Chan-Vese level-set approach

The Chan and Vese [20] level-set approach serves for dividing an image in two regions. Both regions can be composed of unconnected parts of the image.

it is a modification of the Mumford-Shah functional so that the level-set approach can be used. The idea of the level-set approach is to introduce the level-set function ϕ which describes an image in the following way:

- Its positive values correspond to the foreground of the image.
- Its negative values correspond to the background of the image.
- The zero-level of ϕ is the border between foreground and background.

The function ϕ is evolved during the minimization process of the Chan-Vese functional until a minimum is reached.

Figure 2.5 shows an example of a segmentation result using the method described in Chapter 5. This method segments an image in the following way:

1. A user selects a rectangle that contains only pixels belonging to the foreground.
2. From this rectangle, a probability distribution function of the foreground is calculated.
3. The level-set function ϕ is then initialized using this PDF.
4. ϕ evolves while the functional in Equation (2.9) is minimized.
5. Finally, values where $\phi > 0$ are considered the foreground, $\phi < 0$ are considered as background and $\phi = 0$ constructs the boundary.

Our contribution is the integration of *the channel framework* in the data-term of the Chan-Vese functional.

Starting from Equation (2.2) and letting $\lambda \to \infty$ leads to *the cartoon limit* or *the minimal partition problem*, where the functional (2.2) reduces to the piecewise constant Mumford-Shah functional (PCMS).

$$E(u, C) = \sum_{i}^{N} \int_{\Omega_i} (u_i - g)^2 dx + \nu |C| \tag{2.3}$$

where Ω_i are N different segmented regions for which u_i takes constant values

$$u_i = \frac{1}{|\Omega_i|} \int_{\Omega_i} (g) dx \tag{2.4}$$

Zhu and Yuille [101] suggest using probability distribution functions $p(\boldsymbol{g}|\Omega_i)$ as segmentation descriptors. Using log-likelihood $\log p(\boldsymbol{g}|\Omega_i)$ instead of the data term $(u_i - g)^2$ in (2.3) yields:

$$E(p, C) = \sum_{i}^{N} \int_{\Omega_i} - \log p(\boldsymbol{g}|\Omega_i) dx + \nu |C| \tag{2.5}$$

For a Gaussian model with constant variance σ^2, *i.e.*

$$-\log p(\boldsymbol{g}|\Omega_i) = \frac{(u_i - g)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2) \ , \tag{2.6}$$

the functional (2.5) reduces to (2.3).

The foreground boundary is described as:

$$C = \{(x, y) \in \Omega \subset \mathbb{R}^2 : \phi(x, y) = 0\}, \tag{2.7}$$

being the zero-level of a level set function $\phi : \Omega \to \mathbb{R}$. The image foreground is defined by the part of the image domain for which $\phi(x)$ takes positive values:

$$\Omega_f = \{(x, y) \in \Omega \subset \mathbb{R}^2 : \phi(x, y) > 0\} \tag{2.8}$$

and the image background is defined by the set Ω_b where the level set function takes negative values.

The general piecewise constant Mumford-Shah functional (PCMS), Equation (2.5), has the following form:

$$E_{CV}(\phi) = \int_\Omega (H(\phi) - 1)\log(p(\boldsymbol{g}|\Omega_b)) - H(\phi)\log(p(\boldsymbol{g}|\Omega_f)) + \nu|\nabla H(\phi)|dx \tag{2.9}$$

The Zhu and Yuille extension of the original Chan-Vese approach allows better segmentation of highly textured images. Experiment 1 in Chapter 4 demonstrates the inability of the original Chan-Vese approach to segment images in which region distributions do not resemble an uni-modal Gaussian distribution.

Minimization of the functional in Equation (2.9) requires posterior probability distribution functions $p(\boldsymbol{g}|\Omega_f)$ and $p(\boldsymbol{g}|\Omega_b)$ to be calculated. *The channel framework* is used to approximate these PDFs. The channel framework is introduced in Chapter 3. The NRI minimization method of the functional is given in Chapters 4 and 5.

Figure 2.5 shows an example of a segmentation result using the method described in Chapter 5. This method segments an image in the following way:

1. A user selects a rectangle that contains only pixels belonging to the foreground.
2. From this rectangle, a probability distribution function of the foreground is calculated.
3. The level-set function ϕ is then initialized using this PDF.
4. ϕ evolves while the functional in Equation (2.9) is minimized.
5. Finally, values where $\phi > 0$ are considered the foreground, $\phi < 0$ are considered as background and $\phi = 0$ constructs the boundary.

The Chan-Vese level-set approach is limited to image segmentation in two regions only. To segment an image into multiple regions depicting different objects and background usually *the Potts model* [75] is used. This model is introduced in the following subsection.

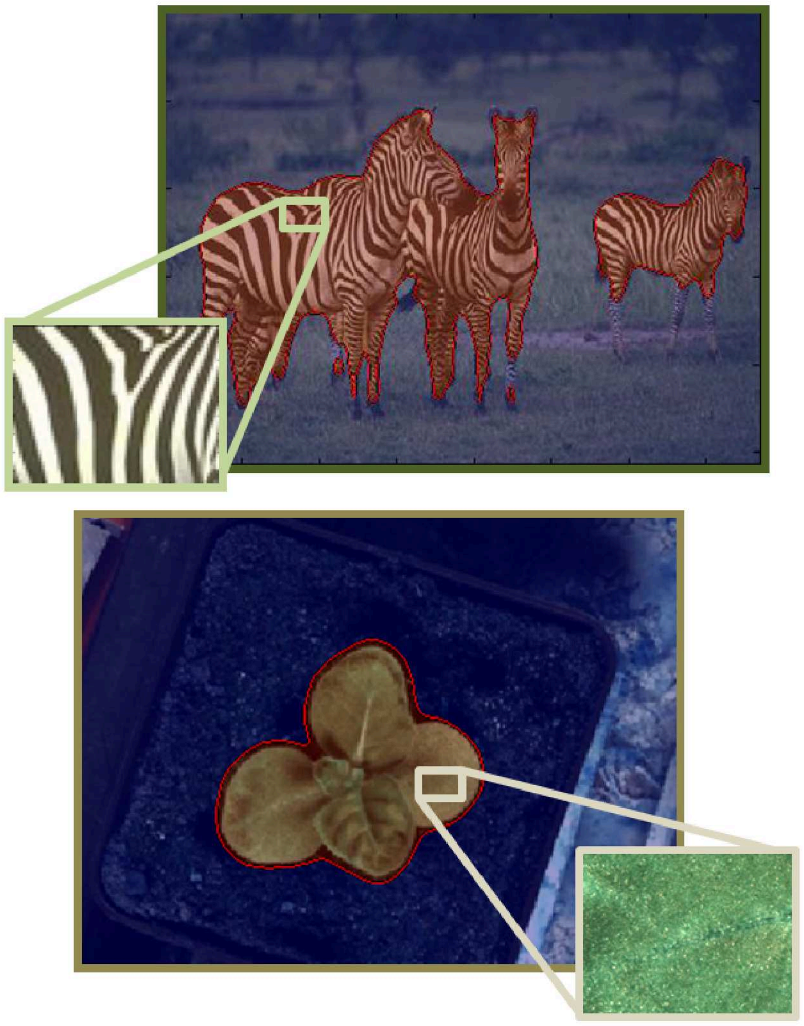


Figure 2.5: Segmentation results using a modification of Chan-Vese level-set approach with a prior knowledge described in Chapter 5: A user selects the rectangle area from which the prior knowledge on foreground distribution is calculated. Images are divided into foreground (the red area) and background (the blue area).

2.4.3 The Potts model

The Potts model [75] is originally suggested in its discrete setting. The disadvantage of the discrete setting is that the discrete grid causes artifacts in results. Artifacts consist in solo pixels within regions or on a region edges that belong to random regions. Methods in [50, 16, 49] suggest regularization to reduce artifacts, but they require complex calculations as well. Nieuwenhuis *et al.* [68] provide reviews of minimization methods that solve the Potts model in the discrete or the continuous setting.

The Potts model can be solved in the spatially continuous setting with a bounded image domain. Methods described in papers [19, 70, 73, 97, 82, 80, 69] suggest solving the convex relaxation of the Potts model. While there is no mathematical proof that the minimum they reach is the global minimum of the original non-convex Potts model, these methods give better results to segmentation than methods using the discrete setting of the Potts model. This is shown in many experiments in related papers and within this thesis.

Within the thesis, the toolbox provided by Yuan *et al.* [99, 97, 98] for solving the Potts model is used. They solve the convex relaxation of the continuous Potts model using the continuous max-flow approach.

We have implemented the channel representation of image features in the data term of the functional. The setup of minimization of the functional and the data-term are described in following text. This method is used in Chapter 6.

The Convex Relaxed Potts model [99, 97, 98] has the following form:

$$\min_{\rho\in\mathcal{S}} \sum_{i=1}^{N} \int_{\Omega} \rho_i(x) C_i(x) dx + \sum_{i=1}^{N} \int_{\Omega} \omega(x) |\nabla \rho_i| dx \tag{2.10}$$

where $\mathcal{S}$ is the constrained set of $\rho(x) := \big(\rho_1(x), \dots, \rho_N(x)\big)$.

$$\mathcal{S} = \left\{ \rho(x) \,\middle|\, \sum_{i=1}^{N} \rho_i(x) = 1; \quad \rho_i(x) \in [0,1], \quad i = 1,\dots,N; \quad \forall x \in \Omega \right\}, \tag{2.11}$$

Using this model, the continuous image domain $\Omega \in \mathbb{R}^2$ is partitioned into a set of disjoint regions Ω_i:

$$\bigcup_{i=1}^{N} \Omega_i = \Omega; \quad \Omega_k \cap \Omega_l = \emptyset, \quad \forall k \neq l \tag{2.12}$$

Functions $\rho_i \in [0,1]$ indicate the belonging of the value x to a region Ω_i.

The Potts model becomes the Mumford-Shah functional, Equation (2.3), by using following cost functions $C_i(x)$ in Equation (2.10):

$$C_i(x) = (\boldsymbol{u}_i(x) - \boldsymbol{g}(x))^2 \tag{2.13}$$

where u_i takes constant values within each region, see Equation (2.4). Within this thesis, probability density estimates are used as measures of x values to belong to regions $\Omega_i, i = 1,\dots,N$.

$$C_i(x) = -\log p(\boldsymbol{g}(x)|\Omega_i) \tag{2.14}$$

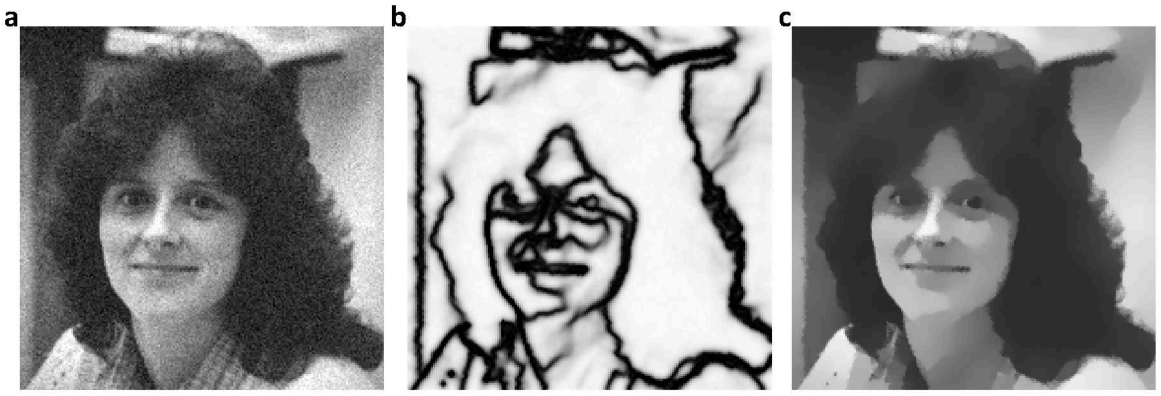


Figure 2.6: The goal of the Ambrosio-Tortorelli segmentation is to estimate the edge-map on the image and give its cartoon/smooth reconstruction. **a** original images degraded with a Gaussian noise; **b** estimated edge maps; **c** smoothed images.

The first term in Equation (2.10) is the data term, while the second term is the total variation term that measures the perimeter of the responding region $|\delta\Omega_i|$. The function $\omega(x)$ is a penalty function for the total variation term. Without using image-edge weights, $\omega(x)$ is constant everywhere. Within this thesis the function:

$$\omega(x) = \lambda \cdot e^{-\gamma|\nabla g(x)|}; \quad \lambda, \gamma \in \mathbb{R}_+ \tag{2.15}$$

is used to add priority to values close to image edges.

2.4.4 The Ambrosio and Tortorelli approximation of the MS functional

Solving the AT functional recovers a smooth reconstruction of an image and its edge-map. An example of using the AT functional is given in Figure 2.6.

Ambrosio and Tortorelli suggest the following approximation of the MS functional in their paper [5]:

$$\begin{aligned} E_{AT} = \iint\limits_R & (\beta(u(x,y) - g(x,y))^2 + (\alpha(1-v(x,y))^2|\nabla u(x,y)|^2) \\ & + (\frac{\rho}{2}|\nabla v(x,y)|^2 + hv(x,y)^2))\mathrm{d}x\mathrm{d}y \end{aligned} \tag{2.16}$$

where $v(x,y) : \mathbb{R}^2 \to [0,1]$ is the smooth edge indicator function with $v(x,y) \approx 1$ on the edges, and $v(x,y) \approx 0$ on smooth regions. $\alpha, \beta, \rho, h \in \mathbb{R}^+$ are regularization parameters. For $\rho \to 0$ the energy E_{AT} converges to the original value of the Mumford-Shah functional. Here, the idea behind this claim will be explained, while for the full proof it is suggested to consult the original paper [5].

1. The first term influences $\boldsymbol{u}$ to be similar to $\boldsymbol{g}$.
2. The second term influences $\boldsymbol{u}$ to be smooth on the whole domain except on edges, and $\boldsymbol{v}$ to have non-zero values on edges.

3. The third term penalizes the edge set. The term $\frac{\rho}{2}|\nabla v|^2$ influences that the edge is continuous and $\boldsymbol{v}^2$ effects that $\boldsymbol{v}$ has only a small amount of non-zero elements. Letting $\rho \to 0$ makes the gradient of $\boldsymbol{v}$ take high values $\frac{1}{\sqrt{\rho}}$ where it is non-zero and therefore leads to edges becoming as thin as possible and as sharp as possible.

To reduce the number of parameters, in further development of equations, h is chosen as $h = \frac{1}{2\rho}$

The parameter β can be a scalar value $\beta \in \mathbb{R}^+$ or it can also be a function of the form $\beta(x, y)$. This way a priority is given to certain regions for smoothing. The AT functional can be used for inpainting as well. Inpainting is the process of reconstructing the missing or corrupted parts of an image. This is achieved by choosing $\beta(x, y) = 0$ for unknown or damaged areas.

Our stochastic approach to solve the functional is presented in Chapter 7.

2.5 Summary

This chapter gave a short introduction to image segmentation and energy functionals that are used within our methods. These include the Mumford-Shah functional, the Chan-Vese functional, the Potts model and the Ambrosio-Tortorelli functional.

Functionals for image segmentation are modified so that the channel framework and the block-Gibbs sampler can be used. The related image processing methods are explained in the following chapters.

Chapter 3

The channel framework

3.1 Introduction

Originally, the channel framework was used to describe the information representation in a human brain. *E.g.* in [33], authors show that different neurons are activated for different orientations in image structures. They present different orientations in channels and observe positions of peak values.

The channel representation is also known as *population codes* [100] as it is used to describe the joint activity of a certain group. Population codes are *e.g.* used to characterize cells, animal behavior in a group, *etc.*

The channel framework has already been introduced in computer vision problems *e.g.* [30, 38, 39, 45, 92, 93]. In [30] channel smoothing is used to smooth a noisy image but at the same time keep edges sharp. Therein, the channel framework is compared to kernel density estimates. Authors show that channel density estimation is equivalent to sampled kernel density estimation. This is demonstrated within this thesis chapter as well.

The channel framework is also used in the work by Wallenberg *et al.* [93] where they use channels to fuse information about color distribution features and depth of an image. The channel representation, presented in this paper, aligns all features one after another in a channel vector. This enables them to combine different features and use simple calculations to achieve segmentation. However, using this representation, the correlation between color components is lost.

Our approach and the work by Jonsson [46] form channel tensors by tensor-multiplying vectors responding to different features. This is explained in detail in Section 3.3. This representation is equivalent to sampled kernel density estimators. It also enables a better approximation of PDFs compared to histograms. Still, creating channel tensors requires a large amount of memory, and therefore, only up to three features are combined within this thesis.

In the work by Jonsson [46], the channel framework is studied in detail and as a final application, object recognition is chosen. in his work, channel-coded feature maps are described as a generalization of the SIFT descriptor [57] with options of including more features and replacing the linear interpolation between bins by a more general basis function.

In this thesis, we suggest the channel framework to be implemented within the Chan-Vese level-set approach. This method is described in Chapters 4 and 5. Also, it is implemented, but in a different way than in [93] for representing the data term in the Potts model for multi-label segmentation. The segmentation method presented in Chapter 6 is an interactive multi-label segmentation method based on a color distribution.

Channels are mainly used for approximating probability distribution functions (PDF). Traditionally PDFs are approximated using either histograms or kernel density estimators. Histograms are used in the work of Weiler and Eggert, [95]. Kernel density estimates are also called Parzen density estimates and are used for estimating PDFs in the papers [77], [48], [88] and [69].

3.2 Channel representation of a scalar

A channel vector is constructed from a scalar signal value s using the following transformation:

$$\boldsymbol{c}(s) = [b(s - \tilde{s}_1), \ldots, b(s - \tilde{s}_{N_c})]. \tag{3.1}$$

$b(s)$ is a symmetric non-negative basis function with compact support. Basis functions of channel representation are formed by shifting the function $b(s)$ to channel centers $\tilde{s}_i, i \in [1, N_c]$. Channel centers are usually regularly spaced on the domain of the signal value s. Figure 3.2 shows the encoding of a single value $s \in [s_{min}, s_{max}]$ with $N_c = 6$ channels formed using cos^2 basis functions.

As basis functions within this thesis, truncated cos^2 functions (functions that gives only one cos^2 period and otherwise give 0) and truncated Gaussian functions are used. Truncated cos^2 functions are used also in papers [38, 39, 45, 92, 93], while Gaussian functions are implemented in [31]. In [46] B-splines are a primary choice as basis functions. A special case is if basis functions are non-overlapping box functions regularly spaced on the signal domain. In this case, the channel representation is equivalent to histograms.

In following text implementation of our basis functions is given.

A truncated cos^2 basis function

$$\begin{aligned} b(s) &\triangleq \begin{cases} \cos^2(\frac{\pi s}{3d}) & |s| < \frac{3d}{2} \\ 0 & |s| \geq \frac{3d}{2} \end{cases} \\ c_n(s) &= b(s - s_{min} + 3\frac{d}{2} - nd) \end{aligned} \tag{3.2}$$

Basis functions are positioned at fixed points on the interval I_s with a shift $d \in \mathbb{R}_+$ between functions:

$$d = \frac{s_{max} - s_{min}}{N_c - 2}. \tag{3.3}$$

This mapping is shown in Figure 3.1.

A truncated Gaussian kernel as a basis function

$$
\begin{aligned}
\boldsymbol{c} &= (c_1, c_2, \ldots, c_{N_c}); \\
\alpha &= \frac{\Delta_n^2}{2\sigma^2}\left(\frac{s - s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right)^2; \\
c_n(s) &\triangleq \begin{cases} e^{-\alpha}, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| < \Psi \\ \\ 0, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| \geq \Psi \end{cases}; \\
\Delta_s &= s_{max} - s_{min}; \\
\Delta_n &= N_c - 1
\end{aligned}
\tag{3.4}
$$

Here, $N_c \in \mathcal{N}$ is the number of basis functions/ channels while $\sigma, \Psi \in \mathcal{R}_+$ are regularization parameters.

A gray-value image $g(\boldsymbol{x}) : \Omega \to I_s$ is defined on an image domain $\Omega \in \mathbb{R}^2$ and takes scalar values from an interval $I_s = [s_{min}, s_{max}] \subset \mathbb{R}$. To encode a gray-value image, it is necessary to encode each pixel value. The pixel value s at a position $\boldsymbol{x}_s$ is a scalar from the interval I_s. This produces the data-set of a size $|\Omega| \times N_c \in \mathbb{R}^3$, where N_c is a number of channels. This can be seen as a stack of N_c parallel images of a size Ω, Figure (3.3).

Mapping the signal s to N_c basis functions gives the vector of channel coefficients:

$$
\begin{aligned}
\boldsymbol{c}(s) = [c_1, c_2, \ldots, c_{N_c}] \\
c(s, i) = c_i; \\
i = 1, \ldots, N_c
\end{aligned}
\tag{3.5}
$$

Channel parameters, *e.g.* the width of basis functions, are then chosen such that the following equation holds:

$$
\forall s \in I_s, \quad \sum_{k=1}^{N_c} c_k(s) \approx C_1; \quad C_1 \in \mathbb{R} \tag{3.6}
$$

This means that the sum of channel coefficients across channels should be constant, being exactly fulfilled for cos^2 basis functions and suitably chosen spline basis functions as suggested in [30]. For the truncated Gaussians (3.4) is only approximately fulfilled. Therefore, one should not use truncated Gaussians in algorithms where (3.4) needs to hold exactly. At the same time, the following equation should hold as well:

$$
\int_{-\infty}^{+\infty} c_k(x)dx = \int_{-\infty}^{+\infty} b(x)dx = C_2; \quad k = 1, \ldots N_c; \quad C_2 \in \mathbb{R} \tag{3.7}
$$

Equations (3.6) and (3.7) ensure accurate approximations of probability distribution functions later. If these equations are not met, a bias between regions will appear and some regions might be preferred to other regions.

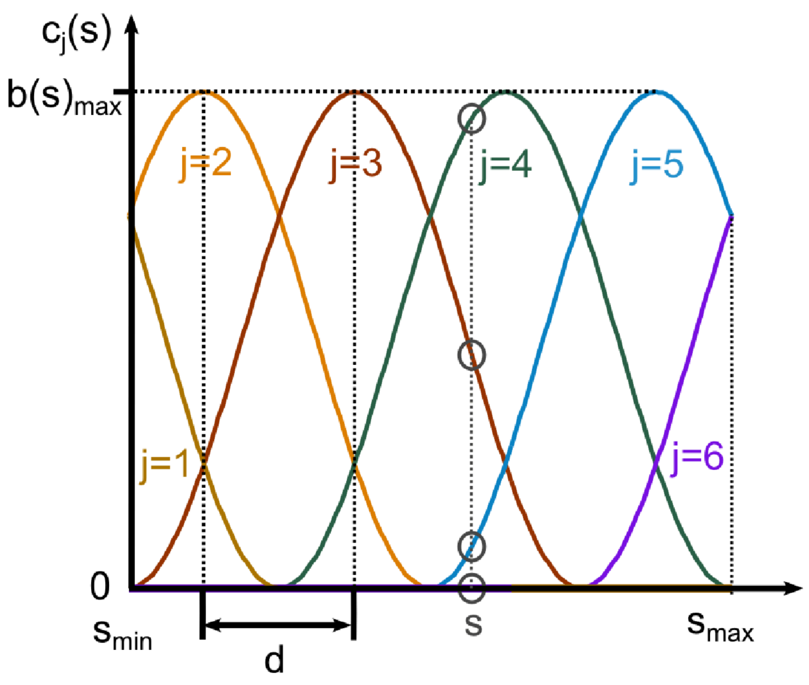


Figure 3.1: Mapping a signal value s to $N_c = 6$ channels with cos^2 basis functions, based on Equation 3.2.

Usually, several features are considered while segmenting an image and not just a gray-value intensity. There is a tendency to combine these features together in a channel vector and process them all at the same time. Therefore, encoding just a scalar value in channels is not enough. Also, pixel values of color images are vector quantities. *E.g.* RGB images contain three signals to describe a pixel value: red value, green value and blue value. To encode color images in channels we need to encode vector values instead of scalar values. This encoding is presented in the next section.

3.3 Channel representation of a vector

To encode vector quantities [46], it is necessary that basis functions take vector quantities as inputs. If a signal $\boldsymbol{s}$ is a vector of a size $h \in \mathbb{N}$, it is encoded in *a channel tensor*. The size of a channel tensor is then ${N_c}^h$.

$$\boldsymbol{c}(\boldsymbol{s}) = [b(\boldsymbol{s} - \tilde{\boldsymbol{s}}_1), \ldots, b(\boldsymbol{s} - \tilde{\boldsymbol{s}}_{N_c})] \tag{3.8}$$

If basis functions are separable, *e.g.* $\boldsymbol{s} = [u, v, w]$, $b(\boldsymbol{s}) = b_u(u)b_v(v)b_w(w)$ it is possible to encode all elements of $\boldsymbol{s}$ separately in channel vectors and then combine them into a channel tensor by taking the tensor product between them.

$$\begin{aligned} \boldsymbol{c}(u) &= [b_u(u - \tilde{u}_1), \ldots, b_u(u - \tilde{u}_{N_c})] \\ \boldsymbol{c}(v) &= [b_v(v - \tilde{v}_1), \ldots, b_v(v - \tilde{v}_{N_c})] \\ \boldsymbol{c}(w) &= [b_w(w - \tilde{w}_1), \ldots, b_w(w - \tilde{w}_{N_c})] \\ \hat{\boldsymbol{c}}(\boldsymbol{s}) &= \boldsymbol{c}(u) \otimes \boldsymbol{c}(v) \otimes \boldsymbol{c}(w) \end{aligned} \tag{3.9}$$

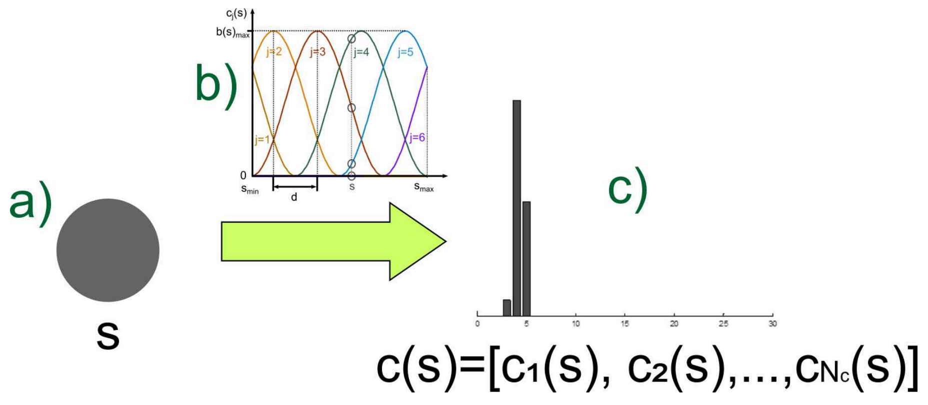


Figure 3.2: Encoding a scalar value in channels. **a)** a scalar value s, this is just one pixel value *e.g.* $s = 128$; **b)** basis functions for encoding in channels; **c)** a resulting channel vector

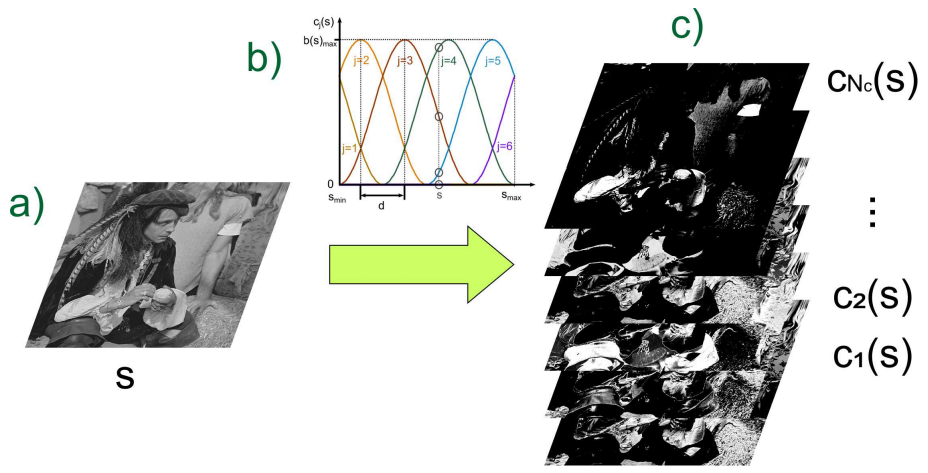


Figure 3.3: Encoding a gray-scale image in channels. **a)** a gray-scale image; **b)** basis functions for encoding in channels; **c)** an encoded gray-scale image viewed as a stack of N_c gray-scale images

If $\boldsymbol{s}$ is a vector with three coordinates, encoding it in channels maps the vector value to a tensor. Tensor is a cube with a size $N_c \times N_c \times N_c$. This mapping is given in Figure 3.4. Within the thesis, only separable basis functions are used.

A pixel value of RGB images $g(\boldsymbol{x}) : \Omega \to \mathbb{R}^3$ consists of three color signals: red, green and blue. The pixel value $\boldsymbol{s} = [s_R, s_G, s_B]$ is then encoded into a cube with a size N_c^3 where N_c is the number of channels:

$$\begin{aligned} \hat{\boldsymbol{c}}(\boldsymbol{s}) &= \boldsymbol{c}(s_R) \otimes \boldsymbol{c}(s_G) \otimes \boldsymbol{c}(s_B); \\ \hat{c}(\boldsymbol{s}, k, i, j) &= c(s_R, k) \cdot c(s_G, i) \cdot c(s_B, j); \\ & i, j, k = 1, \dots, N_c. \end{aligned} \tag{3.10}$$

An image can contain h different signal components, responding to *e.g.* color values or different image features. A pixel value $\boldsymbol{s} = [s_1, s_2, \dots, s_h]$ is then encoded into a hypercube with a size N_c^h:

$$\hat{\boldsymbol{c}}(\boldsymbol{s}) = \boldsymbol{c}(s_1) \otimes \boldsymbol{c}(s_2) \otimes, \dots, \otimes \boldsymbol{c}(s_h) \tag{3.11}$$

As another example, different features are combined in the channel framework in the work by Wallenberg *et al.* [93]. Therein, color signals and the depth information are separately encoded in channels. However, they do not form the channel density tensor, but they arrange channel vectors one after another, forming a new vector of channel coefficients. Using this CDE representation, the correlation between color signals is lost. The final channel vector is formed by putting channel vectors for different features one after another in a line. Considering that for h features a channel tensor of a value has a size N_c^h, it is not recommended to choose many features in a representation if the number of channels is required to be high as well. Otherwise, using channel representation becomes memory costly.

3.4 Calculating channel density estimates (CDE)

If several samples of a signal value s exist, it is possible to build a soft histogram of the signal s. Let $s^{\{1\}}, s^{\{2\}}, \dots s^{\{K\}}$ be K samples of the signal s. All samples can be encoded in channels:

$$\boldsymbol{c}(s^{\{1\}}), \boldsymbol{c}(s^{\{2\}}), \dots, \boldsymbol{c}(s^{\{K\}}). \tag{3.12}$$

Summing up, or averaging channel vectors for each channel position gives *a soft histogram $\boldsymbol{q}(s)$* of a signal s. This soft histogram is also called *a channel density estimate* (CDE) of s. The CDE of s at a channel position $n \in [1, N_c]$ has the following value:

$$q(s, n) = \frac{1}{K} \sum_{i=1}^{K} b(s_n^{\{i\}} - \tilde{s}_n) \tag{3.13}$$

The vector $\boldsymbol{q}$ is called a soft histogram because basis functions can be referred to as bin functions. If basis functions are non-overlapping box functions, the channel representation is equivalent to histogram representation. Unlike regular histograms, soft histograms have bin functions that are smooth and overlap. Samples in histograms are put

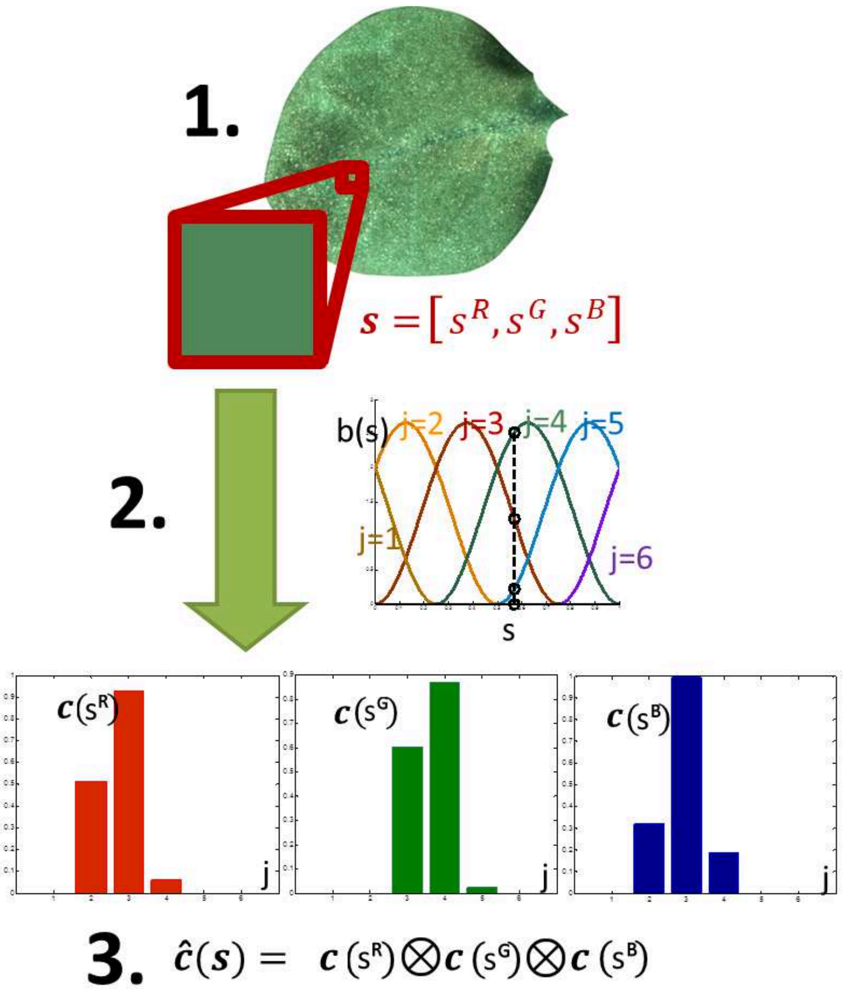


Figure 3.4: Encoding a vector value in channels. **1)** a color pixel value $\boldsymbol{s} = [s^R, s^G, s^B]$; **2)** basis functions for encoding in channels; **3)** a resulting channel tensor is a tensor product of separately encoded signal values $\boldsymbol{c}(\boldsymbol{s}) = [c(s^R) \otimes c(s^G) \otimes c(s^B)]$

in the closest bin, therefore the accuracy of locating peaks in a histogram is limited by bin spacing. In soft histograms bin functions overlap and weight a sample with the distance to closest bin centers. This allows locating peaks in a soft histogram with a sub-bin accuracy. Therefore, soft histograms require more complex bin functions, but also less bins than histograms to achieve similar performance in peak detection.

Channel density estimates are also similar to kernel density estimates. While KDEs are calculated at data dependent positions, CDEs are calculated on fixed positions of the signal domain. They can be seen as a regularly sampled kernel density estimate if a kernel used for KDE is the same as a basis functions used for calculating CDE. This is proven in the paper [30]. Here, we provide a sketch of a proof. In Figure (3.6), a value s is encoded in channels. Channels contain kernels at fixed points in a signal space. Therefore, the encoded signal value produces the value $c_t(s)$ at the position t. Kernel density estimators are calculated on data points. Let the kernel $k(s)$ be chosen the same as in the channel representation, and centered at the signal value s. If it is then sampled at the same positions as basis functions are placed in the case of CDE representation, at the position t, KDE will produce the same value as a channel vector value at position $k(t, s) = c_t(s)$. The function used for forming the kernel $b(s)$ needs to be symmetric, since otherwise, kernels should be mirrored in the KDE representation and then compared to the CDE representation. While KDEs are used to accurately estimate a continuous density function, CDEs usually in practice give satisfying results with less computations.

To approximate the foreground probability distribution function, firstly, the set Ω_f is formed. This set is formed of samples of pixel values or regions that represent the foreground area. Channel values are averaged over Ω_f such that a soft histogram or a channel density estimate of a foreground area is formed. Then, we calculate a probability that a pixel value would be an outcome, given the region Ω_i.

In Chapter 4 the Chan-Vese level-set approach is used. The functional is given in Equation (2.9). Therein, the foreground is considered the area where the level-set function $\phi > 0$ is positive. Therefore, in each iteration, the channel density estimate of the foreground is calculated on the area where $\phi > 0$. Accordingly, the channel density of the background is calculated on the area where $\phi < 0$. For gray-value images, the following equations are used to calculate channels density estimates (CDEs) of foreground and background:

$$\begin{aligned} \boldsymbol{q}_f &= \frac{\int_\Omega H(\phi(\boldsymbol{x}))\boldsymbol{c}(s(\boldsymbol{x}))d\boldsymbol{x}}{\int_\Omega H(\phi(\boldsymbol{x}))d\boldsymbol{x}} \\ \boldsymbol{q}_b &= \frac{\int_\Omega (1 - H(\phi(\boldsymbol{x})))\boldsymbol{c}(s(\boldsymbol{x}))d\boldsymbol{x}}{\int_\Omega (1 - H(\phi(\boldsymbol{x})))d\boldsymbol{x}} \end{aligned} \tag{3.14}$$

where $H(x)$ is the Heaviside function:

$$H(x) = \begin{cases} 1, & x > 0; \\ 0, & x \leq 0 \end{cases} \tag{3.15}$$

and $\boldsymbol{c}$ is a channel vector calculated according to Equation (3.1).

Vector images, *e.g.* RGB images, or images with several features, have tensors assigned to pixels after encoding. CDEs are calculated in the same way as if scalar values are used:

$$
\begin{aligned}
\hat{\boldsymbol{q}}_f &= \frac{\int_\Omega H(\phi(\boldsymbol{x}))\hat{\boldsymbol{c}}(\boldsymbol{s}(\boldsymbol{x}))d\boldsymbol{x}}{\int_\Omega H(\phi(\boldsymbol{x}))d\boldsymbol{x}} \\
\hat{\boldsymbol{q}}_b &= \frac{\int_\Omega (1 - H(\phi(\boldsymbol{x})))\hat{\boldsymbol{c}}(\boldsymbol{s}(\boldsymbol{x}))d\boldsymbol{x}}{\int_\Omega (1 - H(\phi(\boldsymbol{x})))d\boldsymbol{x}}
\end{aligned}
\tag{3.16}
$$

where $\hat{\boldsymbol{c}}$ is a tensor of channel coefficients calculated according to Equation (3.11).

Chapter 5 includes user-provided prior knowledge for foreground. CDEs of foreground and background are formed as a linear combination of the following terms:

$$
\begin{aligned}
\hat{\boldsymbol{q}}_f &= \omega_f \hat{\boldsymbol{q}}_{\phi>0} + (1-\omega_f)\hat{\boldsymbol{q}}_{fu}; \quad 0 < \omega_f < 1 \\
\hat{\boldsymbol{q}}_b &= \omega_b \hat{\boldsymbol{q}}_{\phi<0} + (1-\omega_b)\hat{\boldsymbol{q}}_{bu}; \quad 0 < \omega_b < 1
\end{aligned}
\tag{3.17}
$$

Terms $\hat{\boldsymbol{q}}_{\phi>0}$ and $\hat{\boldsymbol{q}}_{\phi<0}$ are estimated on the region where the level set function is $\phi > 0$ and $\phi < 0$, respectively. $\hat{\boldsymbol{q}}_{fu}$ and $\hat{\boldsymbol{q}}_{bu}$ are estimated from a user defined prior knowledge of the targeted object (drawing scribbles, marking a rectangular area, other images…).

ω_f and ω_b are weight coefficients. Putting $\omega_f = 0$ gives a user defined foreground, $\omega_f = 1$ gives us an automatic segmentation of the foreground. Values in between make a trade-off between a complete unsupervised segmentation and a supervised segmentation.

Chapter 6 describes an interactive multi-label segmentation method. Since this is an interactive approach, channel density estimates for each region/ label Ω_i are calculated from scribbles ${\Omega_i}^{us}$ marked by a user. The channel density estimate for each label $i = 1, \ldots, \mathcal{N}$ is calculated as:

$$
\hat{\boldsymbol{q}}_i = \frac{1}{|\Omega_i|}\int_{{\Omega_i}^{us}} \hat{\boldsymbol{c}}(\boldsymbol{s}(\boldsymbol{x}))d\boldsymbol{x}. \tag{3.18}
$$

A channel density estimate of *e.g.* foreground describes the foreground area. CDEs are then compared to encoded pixel values to estimate if a pixel belongs to foreground or not. The probability of a pixel to belong to a certain region is investigated in the next section.

3.5 Approximation of probability distribution functions

The goal of introducing the channel framework is to retrieve a pixel's probability to belong to some region Ω_i. Therefore, an approximation of the probability distribution function of Ω_i is needed. Within this thesis, this approximation is done by a simple linear interpolation of an image encoded in channels with its foreground CDE. To calculate a probability distribution function of a region Ω_i, for each pixel the following procedure is applied: For gray-value images, the scalar product of a pixel channel vector $\boldsymbol{c}(s)$ and the CDE $\boldsymbol{q}_f$ is taken. This scalar product gives large values when peaks of a channel vector and a CDE

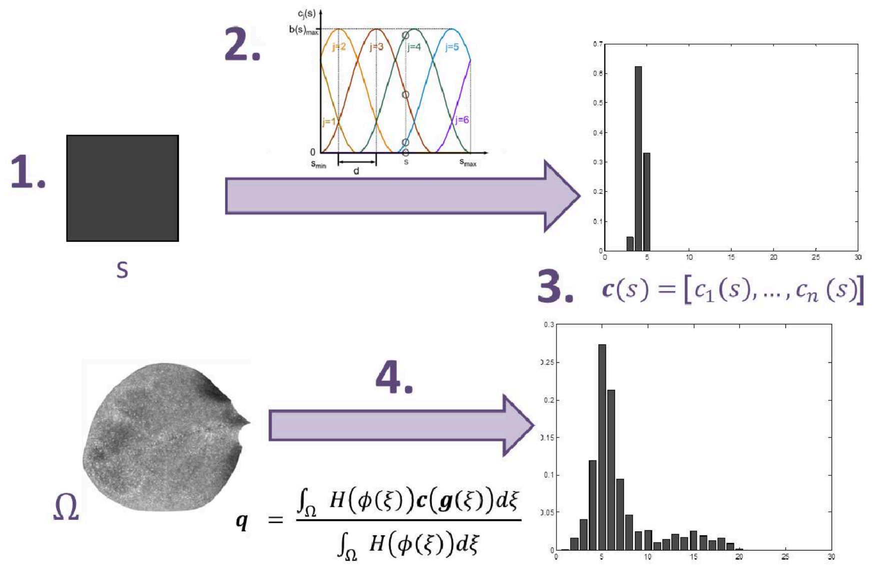

2.
1.
s
3. $c(s) = [c_1(s), \dots, c_n(s)]$
4.
Ω
$q = \frac{\int_\Omega H(\phi(\xi))c(g(\xi))d\xi}{\int_\Omega H(\phi(\xi))d\xi}$

Figure 3.5: (**1.**) a pixel gray-scale intensity s is encoded with *e.g.* (**2.**) cos^2 basis functions in channels to form *the vector of channel coefficients* (**3.**). Averaging vectors of channel coefficients over a region $\Omega \in \mathbb{R}^2$ gives *the channel density estimates* (CDE) of it (**4.**).

are at same channel positions. These pixel values are likely to be chosen given the CDE of Ω_i. If peaks of a channel vector and a CDE do not coincide, the scalar product will give a small value. These pixel values have low probability to be chosen, given the CDE of Ω_i.

$$p(x(\boldsymbol{y})|\Omega_i) \propto \sum_{k=1}^{N_c} c_k(x(\boldsymbol{y}))q_i(k) \tag{3.19}$$

The procedure is the same for vector-valued images. There, each pixel contains a channel tensor $\hat{\boldsymbol{c}}(x)$. According to this, CDEs of regions on a vector valued images are tensors as well, $\hat{\boldsymbol{q}}_i$. Probability distributions are then calculated as:

$$p(\boldsymbol{x}(\boldsymbol{y})|\Omega_i) \propto \sum_{k=1}^{N_c}\sum_{l=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{x}(\boldsymbol{y}), k, l, j)\hat{\boldsymbol{q}}_i(k, l, j) \tag{3.20}$$

In Chapter 5, in Experiment 1 trimaps for lasso selection are used. While calculating PDFs, locations of pixels that are definitely part of a region Ω_i are taken into account. Also, in Chapter 6 positions of scribbles on an image domain Ω is used in an addition to the color distribution to form the PDF of a region Ω_i. Pixels whose coordinates on an image are close to areas that are definitively part of Ω_i are given a higher probability, while those that are far away have low probability. This probability is then fused with the probability calculated from other image features.

This dependence of pixel location on an image is introduced in a PDF $p(\boldsymbol{x}|\Omega_i)$ in the following way:

$$\begin{gathered} p(\boldsymbol{s}(\boldsymbol{x})|\Omega_i) \propto (\hat{\boldsymbol{c}}(\boldsymbol{s})\hat{\boldsymbol{q}}_i) \cdot \sigma_{\Omega_i}(\boldsymbol{x}); \\ \boldsymbol{x} \in \Omega \end{gathered} \tag{3.21}$$

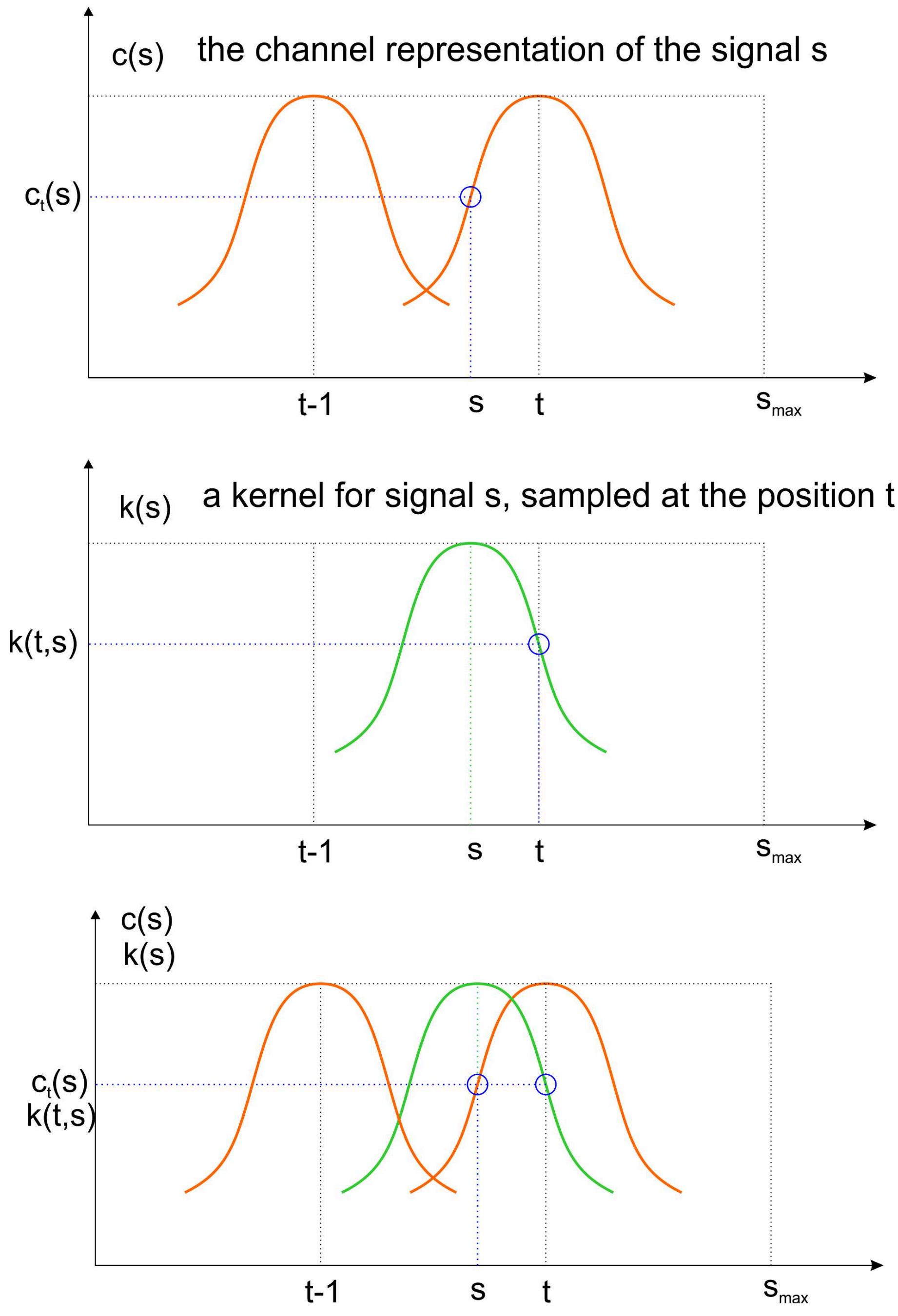

c(s)
the channel representation of the signal s
$c_t(s)$
t-1
s
t
s_{max}
k(s)
a kernel for signal s, sampled at the position t
k(t,s)
t-1
s
t
s_{max}
c(s)
k(s)
$c_t(s)$
k(t,s)
t-1
s
t
s_{max}

Figure 3.6: Comparison of the channel representation to sampling kernels centered at the signal position

where σ_{Ω_i} is a function that describes the spatial dependence for a region Ω_i. The set defined by user interaction to belong to region Ω_i has the notation Ω_i^{us}. Pixels contained in $\boldsymbol{x} \in \Omega_i^{us}$ are described to definitely belong to region Ω_i, either with trimaps or scribbles. The function $\sigma_{\Omega_i}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \in \Omega_i^{us}$. The function has values close to $\sigma_{\Omega_i}(\boldsymbol{x}) \sim 1$ for pixels that are spatially close to pixels in $\boldsymbol{x} \in \Omega_i^{us}$. It has values close to $\sigma_{\Omega_i}(\boldsymbol{x}) \sim 0$ for pixels that are spatially far from pixels in $\boldsymbol{x} \in \Omega_i^{us}$.

σ_{Ω_i} is implemented by applying a transform to a given trimap or a position of scribbles. In our implementation we use the Matlab function `bwdist` to calculate the distance transform. Using this function each pixel is assigned a Euclidean distance to the closest pixel that is part of the region Ω_i^{us}.

$$\sigma_{\Omega_i}(x, y) = e^{-\min\limits_{(x_u, y_v) \in \Omega_i^{us}} \gamma \sqrt{(x - x_u)^2 + (y - y_v)^2}} \tag{3.22}$$

where $\gamma \in \mathbb{R}^+$.

3.6 Building a look-up table

To save time for calculating channel vector coefficients, a look-up table is used in the folowing way:

1. Reducing the number of different colors reduces memory requirements for the look-up table as well as the time needed for loading table values. An image is quantized to η possible different pixel values, such that pixel values take integer values in the range $[0, \eta - 1]$. $\eta \in \mathbb{N}$ is called *quantization number*.

2. Channel coefficients are calculated for all possible pixel values and stored in the table. The look-up table contains axes for the number of channels and number of possible pixel values.

3. When the segmentation algorithm runs, it loads a look-up table and reads needed channel coefficients.

4. Segmentation is performed calculating PDFs as explained in Section 3.5.

Pixels of color images are composed of three signal values. Therefore, encoding color images requires a table of a size $N_c^3 \times \eta^3$. In this thesis, only the uniform quantization is presented.

Look-up tables require a large amount of memory. However, according to Equations (3.5) and (3.10) a look-up table is *a sparse band matrix*. This means that most of table entries are zero and its entries are concentrated around the main diagonal. Therefore, special functions and methods are used to store and access table entries, *e.g.* [35, 36, 26].

To calculate a probability distribution function $p(\boldsymbol{g}|\Omega_i)$, the following procedure is applied:

1. The CDE of a region Ω_i is calculated. Let it be calculated from an area α. This area can be a user selected region as the a-priori knowledge on a region Ω_i. In the Chan-Vese level set method, α can also be a region where a level-set function $\phi > 0$ takes positive values if Ω_i is a foreground, or where $\phi < 0$ if Ω_i is a background of an image.
$$q_\alpha(j) = \frac{1}{|\alpha|} \sum_{\forall \boldsymbol{x} | \boldsymbol{x} \in \alpha} c_j(\boldsymbol{x}) \tag{3.23}$$

2. The number of table entries can be reduced to only those where $q_\alpha > 0$. Approximation of a PDF has the following form:
$$p(\boldsymbol{g}|\Omega_i) = \sum_{\forall j | q_\alpha(j) > 0} c_j(\boldsymbol{x}) \cdot q_\alpha(j) \tag{3.24}$$

3.7 Performance of the channel framework

To test advantages of the channel framework over the KDE approach in a segmentation method, we perform a segmentation experiment, where we either apply KDE or CDE, keeping the rest of the settings and source code identical. In detail, we do the following::

1. 11 images are chosen from the Graz data-set [80]. These are RGB images of a size 391×625. This data-set is aimed at supervised segmentation where the prior knowledge is obtained from scribbles.

2. Initialization as well as the minimization part of codes are exactly the same.

3. The prior knowledge is given by scribbles.

4. Kernels used in the KDE method and basis functions for the channel framework are functions of the same shape. The following kernel is used for kernel density estimates:
$$\begin{aligned} k(s, w) &= e^{-\frac{1}{2\sigma^2}\left(s-w\right)^2}; \\ s, w &\in [0, 1]; \\ \sigma &= 0.05; \end{aligned} \tag{3.25}$$
where, s is a signal value, and w is the kernel center.

5. Only color distribution is used for describing regions. The spatial dependance of scribbles is not introduced, so that results would not be influenced by the distance function presented in Equation (3.21), but only by the kernel density estimation and the channel representation of features.

6. Dice-score, Section B.3.3, is measured and reported.

Results show the following findings:

1. With an increase of number of channels and quantization levels, the dice-score increases and CDE comes close to KDE estimation. While the KDE performance is not reached using these images, results are similar enough according to Table 3.2. The KDE approach leads to $dice - score = 0.9352$, while the CDE approach using 50 quantization levels and 10 channels lead to $dice - score = 0.8896$.

2. Increasing the number of channels and quantization levels does not guarantee the increase of the dice-score. That is because the used performance measure is a dice-score and not the energy value. Segmentation results are evaluated to human segmentation and can be quite subjective. Having this in mind, it can happen that CDE estimation, by accident, fits better than KDEs user-defined results, although the PDF approximation of color distribution is usually more accurate by using KDEs.

3. Larger number of quantization levels can sometimes cause that distributions of different regions become similar and cause false recognitions. Therefore it happens that a large number of quantization levels produces worse results than a small number.

4. A smaller number of channels can sometimes act for smoothing of an image and can cause better segmentation.

Results of the test are given in Figure 3.7.

Next we test the influence of noise introduced to images. Uniform noise is added to each image with a different intensity k:

$$g_n(x) = g(x) + k \cdot \omega(x) \tag{3.26}$$

where $\omega \in [0, 1]$ is a uniform noise. Results are presented in Table 3.3 for three images from the Graz data-set and they show following: Segmentation gets poorer in most cases as the noise level increases. For the small number of channels, with an increase of the noise level, results change dramatically, while for larger number of quantization levels and larger numbers of channels, similar results are obtained even with addition of noise.

Examples of image segmentation are given in Figures 3.8, 3.9 and 3.10.

3.8 Memory requirements

The channel framework requires loading look-up tables. In our calculations we use single data-type representation. This means that 32 bits are required for presenting one number. For a choice of $\eta = 32, N_c = 8$ each data-point has 5 relevant channel coefficients per signal and the others are close-to or equal 0. The look-up tables per signal are band matrices. Therefore for one look-up table for only one signal the table has $5 \cdot \eta = 160$ single data-type non-zero entries, and takes $160 \cdot 4 = 640 byte$. For a color image with 3 signals this results in $160^3 = 4096000$ non-zero entries and requires $16.384 MB$.

If kernels are positioned on all possible positions of data-points and the same kernel is used for forming all channels, it is necessary only to store points of the kernel and translate its position in channel space for different values of pixels. This means that if

kernel density estimate	**the channel framework**
• kernels are positioned at all data-points that belong to corresponding scribbles	• channels are located at fixed positions in signal space • channel vectors/tensors are read from a look-up table for all scribble data-points and then averaged. • this gives us channel density estimates (CDEs) of regions.
• kernels for all data-points are necessary for further calculations	• the channel density estimate consists of N_c values and can be stored and loaded also for other segmentation where the same prior is used. • there is no need for individual data-point values of scribbles anymore
• estimates are calculated for each pixel value using all above kernels	• channel vectors/tensors for each pixel-value are read from a look-up table and then interpolated with values from the channel density estimate

Table 3.1: Main differences between the kernel density estimation and the channel framework

N_c/qu	2	5	10	30	50
2	0.7664	0.7078	0.7160	0.7156	0.7156
5	–	0.8378	0.8156	0.8179	0.8170
10	–	–	0.8926	0.8884	0.8896
20	–	–	–	0.9150	0.9149

Table 3.2: Performance of the channel framework in terms of dice-score. The average dice-score over chosen images using KDE approach is: $dice - score = 0.9352$. The graphical interpretation is presented in Figure 3.7.

	$k = 0$			$k = 0.1$			$k = 0.5$			$k = 0.8$		
image	1	2	3	1	2	3	1	2	3	1	2	3
$n_c = 2; qu = 5$	0.94	0.3	0.11	0.92	0.09	0.11	0.42	0.09	0.12	0.42	0.09	0.47
$n_c = 5; qu = 5$	0.97	0.62	0.98	0.97	0.65	0.98	0.97	0.52	0.98	0.96	0.52	0.98
$n_c = 10; qu = 50$	0.94	0.57	0.98	0.94	0.57	0.98	0.93	0.52	0.96	0.86	0.51	0.94
$n_c = 10; qu = 100$	0.94	0.57	0.98	0.94	0.57	0.98	0.94	0.52	0.98	0.86	0.51	0.94
$n_c = 20; qu = 50$	0.92	0.58	0.98	0.92	0.57	0.98	0.91	0.52	0.97	0.87	0.51	0.96
$n_c = 20; qu = 100$	0.92	0.58	0.98	0.92	0.57	0.98	0.91	0.52	0.97	0.86	0.51	0.95

Table 3.3: Performance of the channel framework on noisy images. In the table, the dice-score is presented for all cases. According to experiments, additive noise causes worse results. Noise is added to three images from the Graz data-set; k presents the level of noise as described in Equation (3.26)

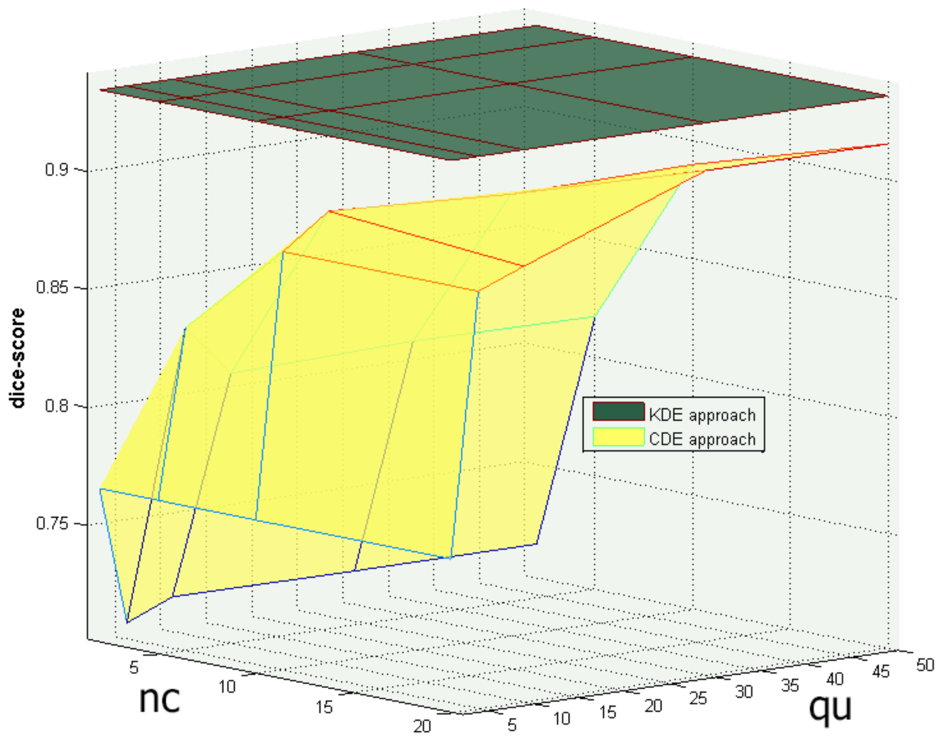


Figure 3.7: Quality performance of the channel framework approach and the kernel density estimation approach in terms of dice-score. Results are presented in Table 3.2.

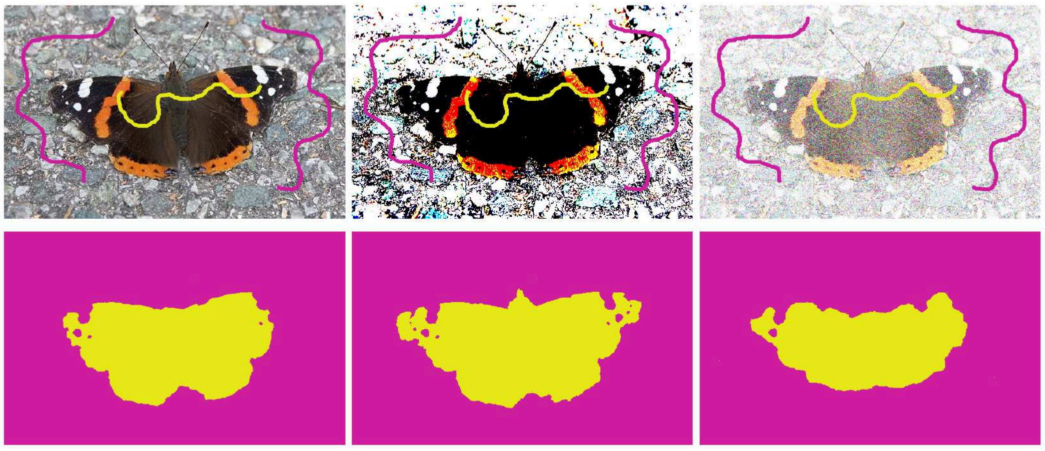


Figure 3.8: Segmentation of Image 1 in Table 3.3: original image, image with 2 quantization levels, noisy image with $k = 0.8$.

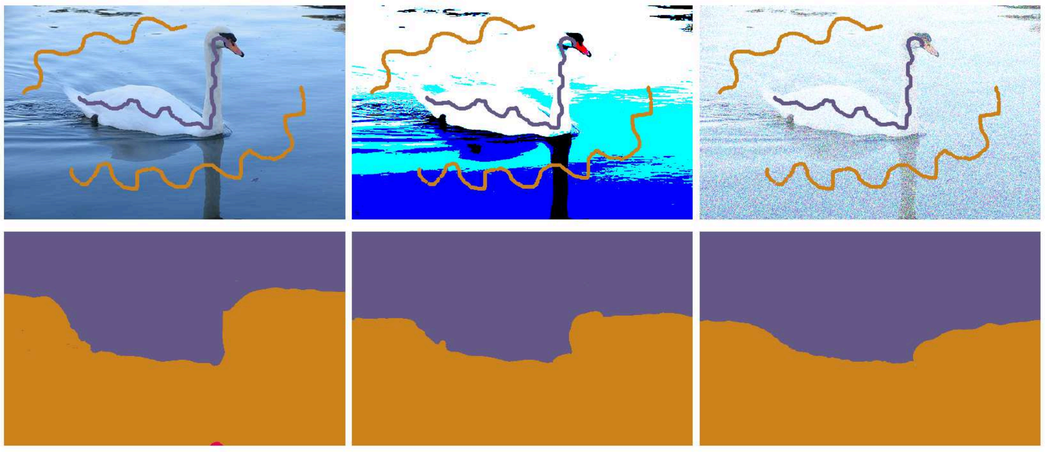


Figure 3.9: Segmentation of Image 2 in Table 3.3: original image, image with 2 quantization levels, noisy image with $k = 0.8$.

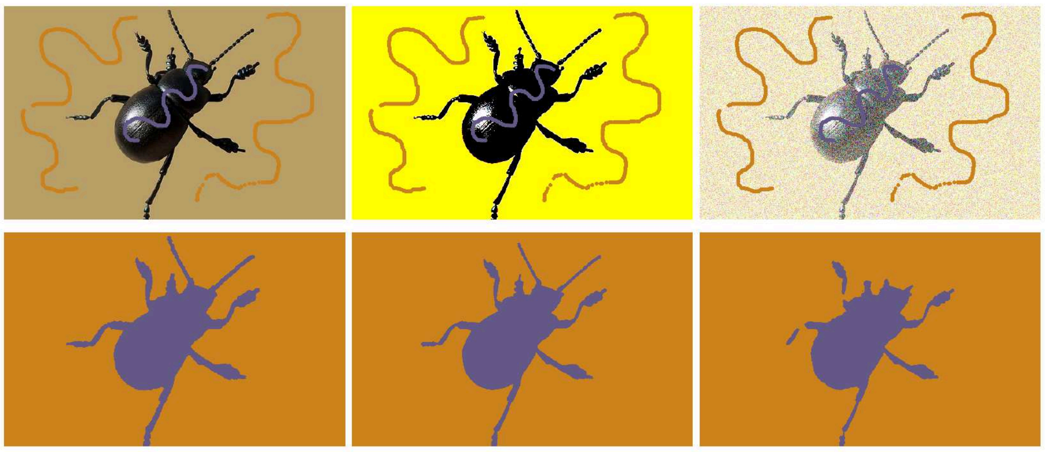


Figure 3.10: Segmentation of Image 3 in Table 3.3: original image, image with 2 quantization levels, noisy image with $k = 0.8$.

the kernel has 5 valuable channel coefficients per signal, for one signal table only these 5 entries are necessary and a position of the kernel, resulting in $5 \cdot 4 = 20byte$ are needed for table entries and $1byte$ for a position of the kernel in the channel space in a case that there are 256 different possible positions in channel space. In a case that an image is composed of 3 signals with integer values within the range $[0, 255]$, $5 \cdot 5 \cdot 5 \cdot 4 = 500bytes$ are required for entries of the table, plus $3 \cdot 1 = 3bytes$ for saving positions of the kernel.

After reading the look-up table values, CDEs are calculated. A CDE is a vector that contains for one signal images maximum N_c non-zero values of a type single, requiring $N_c \cdot 4bytes$, or for images composed of three signals it contains a maximum of N_c^3 non-zero values, requiring $N_c^3 \cdot 4bytes$.

The channel framework has an advantage over the kernel density approach that it offers simple calculations to achieve similar results. Let us assume that N_c channels are chosen and η quantization levels for a color image with 3 signals. To calculate a PDF of related to the foreground of an image of size $m \times n$ and using $k \in \mathbb{N}$ data-points for calculating the channel density estimate, the following amount of operations (summations and multiplications) is required:

- for calculating the CDE: $(k - 1) \cdot N_c^3$;
- for calculating the PDF: $(N_c^3 + (N_c - 1)^3) \cdot m \cdot n$.

3.9 Summary

In this chapter, basics of the channel representation were given together with its implementation in approximating PDFs. The channel framework has already been used for solving computer vision problems [30, 38, 39, 45, 92, 93]. In this thesis it is used to approximate posterior probability distribution functions within image segmentation algorithms.

The channel framework is related to histograms and kernel density estimates. Histograms are calculated using N number of bins. Each value is assigned to a bin whose center is closest in the signal space. Averaging bin values over an area gives us a histogram, the density estimate of that area. Channels are also calculated at N positions, but in the channel framework basis functions that overlap are used. Overlapping of basis functions allows a sub-bin accuracy when calculating channel density estimates. This means that the channel framework gives a richer information on probability distribution functions than histograms for the same number of bins.

To calculate kernel density estimate of an area that contains M different sample values, kernels are positioned on all M data-points. This number is usually large, $M \gg N$ and calculating probability distribution functions requires many operations. This way, the channel framework combines advantages of both histograms and kernel density estimators.

Channels give more accurate approximation of smooth PDFs than histograms for the same number of bins and channels require less computations than kernel density estimators for $M > N$.

It is possible to form look-up tables for special type of images where different and non-uniformly distributed basis functions will be used. This way it is possible, if the appropriate look-up table is chosen, to reduce the number of channels while achieving outstanding segmentation performance. Using look-up tables is useful especially if instead of one image, a whole set of images needs to be segmented. If the same look-up table is used for all images in the set, a look-up table is loaded in memory only once before the segmentation process starts. During segmentation of all images table entries are read.

Chapter 4

Unsupervised 2-region segmentation of gray-value images

4.1 Introduction

The method in this chapter uses gray-value distribution as a segmentation criterion and searches for two most distinguishing distributions to divide an image into foreground and background. It is an unsupervised method, therefore no hints are given on what a correct segmentation should be.

For image segmentation the Chan-Vese [20] approach is used. The functional is modified such that the channel framework can be used for describing and combining different features in the data term of the functional. The channel framework is explained in Chapter 3. It is presented as an alternative to histograms and KDEs offering combination of advantages of both approaches.

The modified Chan-Vese functional is then solved using *the Non-linear Richardson fixed-point Iteration (NRI) approach* which is an alternative to traditional gradient descent minimization. This combination is another contribution of the thesis.

The example given in Figure 4.1 shows difficulties in developing an unsupervised method.

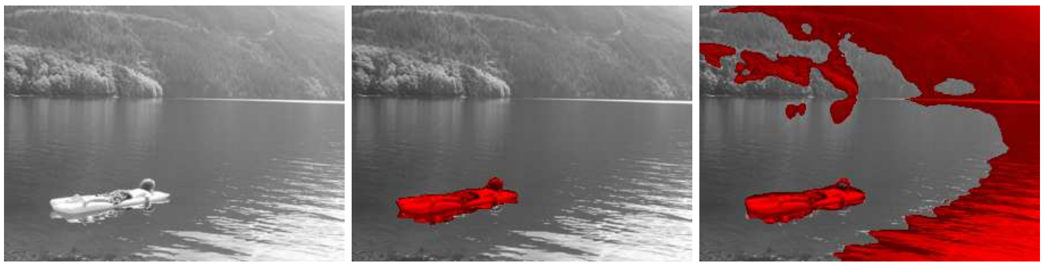


Figure 4.1: Example of an unsupervised segmentation. Having in mind that no prior knowledge is given and the only feature the algorithm deals with is a gray-value distribution, segmentation results can sometimes seem meaningless to a human observer, while still correct based on the presented criterion. Images respond to the original image, the ground-truth segmentation and a result of our method, respectively.

All human observers have selected a person as a foreground on the image. However, the image could have been as well divided into a mountain area and a lake area. The segmentation algorithm that uses only gray-value distribution as a segmentation criterion divides the image into an area that contains lighter parts of mountains and their reflection in the water, together with the person.

4.2 Related methods

4.2.1 Approximating probability distribution functions

In experiments, the method within this chapter is compared to the following unsupervised 2-region segmentation methods: Kim *et al.* [48] and Brox and Cremers [88] use kernel density estimates, also called Parzen density estimates, while Weiler and Eggert [95] use histograms. Brown *et al.* [17] use the Gaussian model with constant variance. They reduce the global Chan-Vese problem to a sequence of minimization problems.

Kim *et al.* [48] designed an experiment showing superiority of kernel density estimators over Gaussian assumption models.

Ni *et al.* [67] create local histograms to describe intensity distributions which are compared using the Wasserstein distance (L. N. Vaserstein [91]). In the paper by Ni *et al.* , it is used for measuring distances between histograms. Their approach does not need a smoothness term, but requires a high number of bins. This method is so far limited to gray-value images, while the extension to color images is foreseen. Also, as a part of future steps, they suggest addition of other image features described in local histograms as well.

The original Chan-Vese formulation [20] has a Gaussian assumption of gray-value intensity distribution for foreground and background. This can be seen in Equation (2.6). The method by Brown *et al.* [17] is a convex relaxation of the original Chan-Vese model such that global minimizers can be applied. It also contains a Gaussian assumption on regions.

In addition to the above mentioned methods, our method is compared to a method that uses just texture descriptors. The idea is to show that using the channel framework it is possible to segment images with a gray-value distribution that would otherwise need texture descriptors to be segmented. The Houhou-Thiran-Bresson model [43] is a segmentation method based on texture features. They derive a texture descriptor based on semi-local pixel information and use the Kullback-Leibler distance for comparing distances between distributions.

4.2.2 Minimizing the functional

Brown *et al.* [17] derived a convex formulation of the Chan-Vese energy functional. Still, their experiments show that their method is independent on initialization, unlike the gradient descent applied in the original Chan-Vese approach [20]. Brown *et al.* observe also that their approach offers better results than the ones obtained by the gradient descent applied to the original Chan-Vese model [20] and the method presented in the paper by

Chan *et al.* [21] and is, therefore, closer to the possible global minimum of the original Chan-Vese problem.

We design a similar experiment as the one presented in their paper to test the performance of our NRI method. We test the sensitivity of the minimization method for getting stuck in local minima. As the Experiment 2 shows, our method is fast, but is not immune to local minima.

4.3 Solving the Chan-Vese functional

The Chan-Vese level set approach is introduced in Section 2.4.2. The functional to be solved is:

$$\begin{aligned} E(\phi) &= \sum_{i\in\{f,b\}} \int_{\Omega_i} -\log p(\boldsymbol{g}|\Omega_i)dx + \nu|C| \\ C &= \{(x,y)\in\Omega\subset\mathbb{R}^2 : \phi(x,y)=0\} \\ \Omega_f &= \{(x,y)\in\Omega\subset\mathbb{R}^2 : \phi(x,y)>0\} \\ \Omega_b &= \{(x,y)\in\Omega\subset\mathbb{R}^2 : \phi(x,y)<0\} \end{aligned} \tag{4.1}$$

For calculating probabilities in the functional, the channel framework is used. Channel density estimates $\boldsymbol{q}_f$ and $\boldsymbol{q}_b$ for foreground Ω_f and background Ω_b regions are calculated using:

$$\begin{aligned} \boldsymbol{q}_f &= \frac{\int_\Omega H(\phi(x,y))\boldsymbol{c}(g(x,y))dxdy}{\int_\Omega H(\phi(x,y))dxdy}, \\ \boldsymbol{q}_b &= \frac{\int_\Omega (1-H(\phi(x,y)))\boldsymbol{c}(g(x,y))dxdy}{\int_\Omega (1-H(\phi(x,y)))dxdy}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{c}$ are channel vectors as given in Section 3.3 and $H(\phi)$ is the Heaviside function:

$$H(\phi) = \begin{cases} 1, & \phi > 0; \\ 0, & \phi \leq 0. \end{cases} \tag{4.3}$$

Then, approximations of PDFs are calculated as:

$$\begin{aligned} p(g(x,y)|\Omega_f) &= \sum_{k=1}^{N_c} c_k(g(x,y))q_f(k); \\ p(g(x,y)|\Omega_b) &= \sum_{k=1}^{N_c} c_k(g(x,y))q_b(k). \end{aligned} \tag{4.4}$$

For solving the functional (4.1), the non-linear Richardson fixed point iteration (NRI) is used which is explained in the following section.

4.4 Minimizing the functional using the Non-linear Richardson fixed-point Iteration (NRI) method

The non-linear Richardson fixed-point iteration (NRI) method was introduced in Computer Vision in the paper by Papenberg *et al.* [71] and the paper by Krajsek and Scharr [54]. In these publications authors claim that the NRI method has preferable properties compared to traditional gradient descent. They show that the NRI method requires less iterations than the gradient descent to converge to the solution.

We modify the Chan-Vese functional and calculate Euler-Lagrange equations. To be able to derive the Euler-Lagrange equation, instead of the original Heaviside and Dirac function, regularized Heaviside and Dirac functions are used, $H_\epsilon(\phi)$ and $\delta_\epsilon(\phi)$. We use the same regularization as in the paper by Chan and Vese [20]. There, following regularization functions are used:

$$\begin{aligned} H_\epsilon(\phi) &= \begin{cases} 1 & \phi > \epsilon, \\ 0.5\left(1 + \frac{1}{\epsilon}\phi + \frac{1}{\pi} sin\left(\frac{\pi\phi}{\epsilon}\right)\right), & |\phi| \leq \epsilon, \\ 0 & \phi < -\epsilon \end{cases} ; \\ \delta_\epsilon(\phi) = H'_\epsilon(\phi) &= \begin{cases} \frac{1}{2\epsilon}\left(1 + cos\left(\frac{\pi\phi}{\epsilon}\right)\right), & |\phi| \leq \epsilon, \\ 0 & |\phi| > \epsilon \end{cases} ; \\ \int_{-\epsilon}^{\epsilon} \delta'_\epsilon(\phi) d\phi &= 0 \end{aligned} \tag{4.5}$$

In the code, we describe the foreground boundary in Equation (4.1), using the following approximation:

$$|C| = \delta_\epsilon(\phi)|\nabla\phi|^2 \tag{4.6}$$

Assuming that ϕ makes sharp transitions between positive and negative values, and that ϵ is sufficiently small, the width of edges in the edge-map obtained by calculating the gradient $|\nabla\phi|$ may be small and approximately the same for all edges. In this case, the integral of $\delta^2_\epsilon(\phi)|\nabla\phi|^2$ is proportional to the edge length.

To obtain Euler-Lagrange equations, we variate the Chan-Vese energy given in Equation (4.1) with respect to ϕ using the test function $\omega : \Omega \to \mathbb{R}$, $\eta \in \mathbb{R}$:

$$\begin{aligned} \left.\frac{dE(\phi + \eta\omega)}{d\eta}\right|_{\eta=0} = \frac{d}{d\eta}\Bigg(\int_\Omega \big(&- H_\epsilon(\phi + \eta\omega) \log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_f) \\ &- (1 - H_\epsilon(\phi + \eta\omega)) \log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_b) \\ &+ \nu\delta_\epsilon(\phi + \eta\omega)|\nabla(\phi + \eta\omega)|^2 \big) d\boldsymbol{x} \Bigg)\Bigg|_{\eta=0} \end{aligned} \tag{4.7}$$

We separately calculate parts of the integral:

$$\begin{aligned}\frac{dE(\phi+\eta\omega)}{d\eta}\bigg|_{\eta=0} =& \underbrace{\omega\int\limits_{\Omega}\Big(-\delta_\epsilon(\phi)\log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_f)+\delta_\epsilon(\phi)\log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_b)\Big)d\boldsymbol{x}}_{\mathcal{A}} \\ &+\underbrace{\int\limits_{\Omega}\Big(-H_\epsilon(\phi)\frac{d}{d\eta}\Big(\log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_f)\Big)\Big|_{\eta=0}\Big)d\boldsymbol{x}}_{\mathcal{B}1} \\ &+\underbrace{\int\limits_{\Omega}\Big(-(1-H_\epsilon(\phi))\frac{d}{d\eta}\Big(\log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_b)\Big)\Big|_{\eta=0}\Big)d\boldsymbol{x}}_{\mathcal{B}2} \\ &+\underbrace{\int\limits_{\Omega}\Big(\nu\frac{d}{d\eta}\Big(\delta_\epsilon(\phi+\eta\omega)|\nabla(\phi+\eta\omega)|^2\Big)\Big|_{\eta=0}\Big)d\boldsymbol{x}}_{\mathcal{C}}\end{aligned} \tag{4.8}$$

Using following entities:

$$\begin{aligned}|\Omega_f| &= \int_\Omega H_\epsilon(\phi(\boldsymbol{x}))d\boldsymbol{x} \\ |\Omega_b| &= \int_\Omega (1-H_\epsilon(\phi(\boldsymbol{x})))d\boldsymbol{x}\end{aligned} \tag{4.9}$$

and considering that probability distributions functions have the following form:

$$p(\boldsymbol{g}(\boldsymbol{x})|\Omega_f) = \frac{\int\limits_\Omega H_\epsilon(\phi(\boldsymbol{y}))\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x})c_k(\boldsymbol{y})d\boldsymbol{y}}{\int\limits_\Omega H_\epsilon(\phi(\boldsymbol{y}))d\boldsymbol{y}}, \tag{4.10}$$

integral $\mathcal{B}1$ becomes:

$$\begin{aligned}\mathcal{B}1 = -\omega\int\limits_\Omega \frac{H_\epsilon(\phi(\boldsymbol{x}))}{p(g(\boldsymbol{x}|\Omega_f))}\left(\frac{\int\limits_\Omega \delta_\epsilon(\phi(\boldsymbol{z}))\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x})c_k(\boldsymbol{z})d\boldsymbol{z}}{\int\limits_\Omega H_\epsilon(\phi(\boldsymbol{y}))d\boldsymbol{y}}\right. \\ \left. -\frac{\int\limits_\Omega H_\epsilon(\phi(\boldsymbol{y}))\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x})c_k(\boldsymbol{y})d\boldsymbol{y}}{\left(\int\limits_\Omega H_\epsilon(\phi(\boldsymbol{y}))d\boldsymbol{y}\right)^2}\int\limits_\Omega \delta_\epsilon(\phi(\boldsymbol{z}))d\boldsymbol{z}\right)d\boldsymbol{x}.\end{aligned} \tag{4.11}$$

We change the order of coordinates and obtain:

$$\mathcal{B}1 = -\omega \int\limits_{\Omega} \delta_\epsilon(\phi(\boldsymbol{z})) \left(\int\limits_{\Omega} \frac{H_\epsilon(\phi(\boldsymbol{x})) \sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x}) c_k(\boldsymbol{z})}{p(g(\boldsymbol{x})|\Omega_f)|\Omega_f|} d\boldsymbol{x} - \int\limits_{\Omega} \frac{H_\epsilon(\phi(\boldsymbol{x}))}{|\Omega_f|} d\boldsymbol{x} \right) d\boldsymbol{z} \tag{4.12}$$

The integral $\mathcal{B}1$ finally becomes:

$$\mathcal{B}1 = -\omega \int\limits_{\Omega} \delta_\epsilon(\phi(\boldsymbol{x})) \left(\int\limits_{\Omega} \frac{H_\epsilon(\phi(\boldsymbol{y})) \left(\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x}) c_k(\boldsymbol{y}) - p(g(\boldsymbol{y})|\Omega_f) \right)}{p(g(\boldsymbol{y})|\Omega_f)|\Omega_f|} d\boldsymbol{y} \right) d\boldsymbol{x} \tag{4.13}$$

Using the same procedure, we calculate the $\mathcal{B}2$ integral. The integral $\mathcal{B} = \mathcal{B}1 + \mathcal{B}2$ then has the following form:

$$\begin{aligned} \mathcal{B} = \mathcal{B}1 + \mathcal{B}2 = -\omega \int\limits_{\Omega} \delta_\epsilon(\phi(\boldsymbol{x})) \Bigg(& \int\limits_{\Omega} \frac{H_\epsilon(\phi(\boldsymbol{y})) \left(\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x}) c_k(\boldsymbol{y}) - p(g(\boldsymbol{y})|\Omega_f) \right)}{p(g(\boldsymbol{y})|\Omega_f)|\Omega_f|} d\boldsymbol{y} \\ & - \int\limits_{\Omega} \frac{(1 - H_\epsilon(\phi(\boldsymbol{y}))) \left(\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x}) c_k(\boldsymbol{y}) - p(g(\boldsymbol{y})|\Omega_b) \right)}{p(g(\boldsymbol{y})|\Omega_b)|\Omega_b|} d\boldsymbol{y} \Bigg) d\boldsymbol{x} \end{aligned} \tag{4.14}$$

And can be also written as:

$$\begin{aligned} \mathcal{B} = & \int\limits_{\Omega} \Big(- H_\epsilon(\phi) \frac{d}{d\eta} \Big(\log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_f) \Big) \Big|_{\eta=0} \Big) d\boldsymbol{x} \\ & + \int\limits_{\Omega} \Big(- (1 - H_\epsilon(\phi)) \frac{d}{d\eta} \Big(\log p(\boldsymbol{g}(\boldsymbol{x})|\Omega_b) \Big) \Big|_{\eta=0} \Big) d\boldsymbol{x} \end{aligned} \tag{4.15}$$

If $p(\boldsymbol{g}|\Omega_f)$ and $p(\boldsymbol{g}|\Omega_b)$ are constant, integral $\mathcal{B} = 0$. The integral $\mathcal{B}$ consists of simple mathematical operations. However, a large number of calculation needs to be performed and it is requires a significant amount of time for calculation. Therefore, to speed up the algorithm, PDFs $p(\boldsymbol{g}|\Omega_f)$ and $p(\boldsymbol{g}|\Omega_b)$ are held fixed during the minimization process. Their values are updated in iterations of the segmentation process instead.

In the paper by Kim *et al.* [48], authors investigate influence of the integral $\mathcal{B}$. In their experiments they show that $\mathcal{B}$ is much smaller than other terms in Equation (4.8) for most images and in many cases neglecting it does not influence the segmentation result. However, they observe that neglecting this term may in special cases result in a loss of accuracy at corners of objects. This is also discussed in the paper by Jehan-Besson *et al.* [44].

If segmentation is done only in one step, PDFs are just estimated once and then minimization is done, if $\mathcal{B}$ is not calculated during the minimization process, the influence of the change of PDFs with the change of the level-set function ϕ is ignored.

Next, we calculate the integral $\mathcal{C}$ in Equation (4.8). Using $\int\limits_{-\epsilon}^{\epsilon} \delta'(\phi)d\phi = 0$, the integral $\mathcal{C}$ becomes:

$$\begin{aligned}
\mathcal{C} &= \int\limits_{\Omega} \left(\nu \frac{d}{d\eta} \Big(\delta_\epsilon(\phi(\boldsymbol{x}) + \eta\omega(\boldsymbol{x})) |\nabla(\phi + \eta\omega(\boldsymbol{x}))|^2 \Big) \Big|_{\eta=0} \right) d\boldsymbol{x} \\
&= \int\limits_{\Omega} \Big(2\nu\delta_\epsilon(\phi(\boldsymbol{x})) \langle \nabla\phi(\boldsymbol{x}), \nabla\omega(\boldsymbol{x}) \rangle \Big) d\boldsymbol{x} \\
&= \int\limits_{\Omega} \Big(-2\omega(\boldsymbol{x})\nu\delta_\epsilon(\phi(\boldsymbol{x}))\mathsf{div}(\nabla\phi(\boldsymbol{x})) + 2\nu\delta_\epsilon(\phi(\boldsymbol{x}))\mathsf{div}(\omega(\boldsymbol{x})\nabla\phi(\boldsymbol{x})) \Big) d\boldsymbol{x} \\
&= -\,\omega \int\limits_{\Omega} \Big(2\nu\delta_\epsilon(\phi(\boldsymbol{x}))\mathsf{div}(\nabla\phi(\boldsymbol{x})) \Big) d\boldsymbol{x} + \omega \oint\limits_{\partial\Omega} \Big(\nu \langle \delta^2(\phi(\boldsymbol{x}))\nabla\phi(\boldsymbol{x}), \vec{n} \rangle \Big) ds
\end{aligned} \tag{4.16}$$

where $\vec{n}$ is the exterior normal to the boundary of Ω. Inserting integrals $\mathcal{B}$ and $\mathcal{C}$ (Equations (4.14) and (4.16)) in Equation (4.8) and equalizing the Gâteaux differential with zero, $\langle \delta E, \omega \rangle = 0$, we obtain Euler-Lagrange equations:

$$\begin{cases}
0 = -\delta_\epsilon(\phi) \log \dfrac{p(\boldsymbol{g}|\Omega_f)}{p(\boldsymbol{g}|\Omega_b)} - 2\nu\delta_\epsilon(\phi)\mathsf{div}(\nabla\phi) - \delta_\epsilon(\phi)\delta\mathcal{B}, \quad \text{inside } \Omega \\
0 = \delta_\epsilon^2(\phi)\langle \nabla\phi, \vec{n} \rangle, \quad \text{on } \partial\Omega
\end{cases}$$

$$\begin{aligned}
\delta\mathcal{B} = \delta_\epsilon(\phi(\boldsymbol{x})) \Bigg(&\int\limits_{\Omega} \frac{H_\epsilon(\phi(\boldsymbol{y})) \left(\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x})c_k(\boldsymbol{y}) - p(g(\boldsymbol{y})|\Omega_f) \right)}{p(g(\boldsymbol{y})|\Omega_f)|\Omega_f|} d\boldsymbol{y} \\
&- \int\limits_{\Omega} \frac{(1 - H_\epsilon(\phi(\boldsymbol{y}))) \left(\sum\limits_{k=1}^{N_c} c_k(\boldsymbol{x})c_k(\boldsymbol{y}) - p(g(\boldsymbol{y})|\Omega_b) \right)}{p(g(\boldsymbol{y})|\Omega_b)|\Omega_b|} d\boldsymbol{y} \Bigg)
\end{aligned} \tag{4.17}$$

$\delta_\epsilon(\phi)$ has non-zero values for $|\phi| < \epsilon$ and gradient changes $div|\nabla\phi|$ are highest on the boundary between foreground and background, $\phi \sim 0$ or $\phi \leq \epsilon$. Considering this, $\delta_\epsilon(\phi)$

function can be neglected as values $\phi \leq \epsilon$ have the highest influence in the equation outcome. The Euler-Lagrange equations then become:

$$\begin{cases} 0 & = -\log \dfrac{p(\boldsymbol{g}|\Omega_f)}{p(\boldsymbol{g}|\Omega_b)} - 2\nu \text{div}(\nabla\phi) - \delta\mathcal{B}, \quad \text{inside } \Omega \\ 0 & = \langle \nabla\phi, \vec{n} \rangle, \quad \text{on } \partial\Omega \end{cases} \tag{4.18}$$

These Euler-Lagrange equations (4.17) are numerically solved by defining a regular discrete grid $\mathcal{G}_\kappa$ with the space of a size $\kappa = 1$ between N grid points on the image domain Ω. Discrete image values evaluated on the grid points are represented as column vectors, *e.g.* $\boldsymbol{\phi} \in \mathbb{R}^N$. We obtain Euler-Lagrange equations in matrix/vector notation:

$$\boldsymbol{0} = -\boldsymbol{r}(\boldsymbol{\phi}) + \nu \boldsymbol{A}(\boldsymbol{\phi}) \tag{4.19}$$

where

$$\begin{aligned} \boldsymbol{r}(\boldsymbol{\phi}) &= \log\left(\frac{p(\boldsymbol{g}|\Omega_b)}{p(\boldsymbol{g}|\Omega_f)}\right) + \delta\boldsymbol{B}(\boldsymbol{\phi}) \\ \boldsymbol{A}(\boldsymbol{\phi}, i, j) &= 2\nu\left(\frac{\partial^2\phi(x_i, y_j)}{\partial x^2} + \frac{\partial^2\phi(x_i, y_j)}{\partial y^2}\right) \\ &= 2\nu\left(\boldsymbol{\phi}(x_{i+1}, y_j) + \boldsymbol{\phi}(x_{i-1}, y_j) + \boldsymbol{\phi}(x_i, y_{j+1}) + \boldsymbol{\phi}(x_i, y_{j-1})\right) - 8\nu\boldsymbol{\phi}(x_i, y_j) \\ \boldsymbol{A}(\boldsymbol{\phi}) &= 2\nu\boldsymbol{G}(\boldsymbol{\phi}) - 8\nu\boldsymbol{\phi} \end{aligned} \tag{4.20}$$

where $i = 1, \ldots, N_i$.

In the NRI method, all nonlinear terms are initialized with some fixed value ϕ_m transforming the nonlinear equation into a linear one. After solving the linear equation for ϕ_m, the nonlinear terms are updated using $\phi_m := \phi_{m+1}$ and the new linear equation is solved again. Thus, we obtain the fixed-point iteration for the m-th iteration.

$$\boldsymbol{\phi}_{m+1} = \frac{1}{4}\boldsymbol{G}(\boldsymbol{\phi}_m) - \frac{1}{8\nu}\boldsymbol{r}(\boldsymbol{\phi}_m) \tag{4.21}$$

There is no proof of convergence of this minimization scheme.

4.5 Implementation details

The method is implemented in the following way:

1. choose an image $\boldsymbol{g} \in \mathbb{N}^{m\times n}$ to segment
2. initialize ϕ
3. calculate $p(\boldsymbol{g}|\Omega_f)$ and $p(\boldsymbol{g}|\Omega_b)$
4. minimize the energy functional $E(\phi)$, (4.1) using (4.21) and keeping $\mathcal{B}$ fixed

5. repeat 1-4 until convergence

ϕ is initialized as a random vector with the same size as the image $\boldsymbol{g}$, $m \times n$. Channel vectors are calculated using Gaussian functions as basis functions. For a gray-value $g(x, y) = s$ at a position (x, y), the channel vector has the following form:

$$\begin{aligned}
\boldsymbol{c} &= (c_1, c_2, \ldots, c_{N_c}); \\
\alpha &= \frac{\Delta_n^2}{2\sigma^2}\left(\frac{s - s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right)^2; \\
c_n(s) &= \begin{cases} e^{-\alpha}, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| < \Psi \\ \\ 0, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| \geq \Psi \end{cases}; \\
\Delta_s &= s_{max} - s_{min}; \\
\Delta_n &= N_c - 1
\end{aligned} \tag{4.22}$$

In the experiments, $8-bit$ images are used, and in the implementation the full gray-value range is considered $[s_{min}, s_{max}] = [0, 255]$.

Channel density estimates, $\boldsymbol{q}_f$ and $\boldsymbol{q}_b$, are calculated from a region where the level set function takes values $\phi > 0$ for foreground and $\phi < 0$ for background according to:

$$\begin{aligned}
q_f(k) &= \frac{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} H(\phi(x,y))c_k(g(x,y))}{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} H(\phi(x,y))} \\
q_b(k) &= \frac{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n}(1 - H(\phi(x,y)))c_k(g(x,y))}{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n}(1 - H(\phi(x,y)))} \\
k &= 1, \ldots, N_c
\end{aligned} \tag{4.23}$$

Finally, approximations of PDFs are given by Equations (4.4).

Instead of calculating channel coefficients in Equation (4.22) during the segmentation process, a look-up table is used. Since images segmented within this chapter are gray-value images that take integer values from interval $[0, 255]$, no quantization of an image is needed. Look-up tables are calculated for different sets of parameters. Parameters that are varied in experiments are: the number of channels N_c, regularization parameter ω of the basis function in (4.22) and regularization parameters ν and N_i for NRI minimization.

4.6 Experiments

Within the experiments, look-up tables with $\eta = 256$ quantization levels and $N_c = 200$ channels are used. The table contains 116 non-zero elements per channel of a type single. This means that the table uses $20640 \cdot 4 = 82.560KB$.

Each CDE contains a maximum of 200 non-zero single elements which means $200 \cdot 4 = 800 bytes$ per CDE. We calculate CDE for foreground and background as well, which means $1600 bytes$ for representing CDEs.

4.6.1 Experiment 1: Quality of the PDF approximation using channels

This experiment is designed to resemble experiments performed in Kim *et al.* [48]. Therein, authors show the superiority of KDE estimators over other density estimators, mainly histograms and Gaussian mixture models. Histograms depend on their bin size and are only efficient if large number of bins are used. Still, even then, KDEs approximate PDFs better. Gaussian assumptions approximate PDFs with Gaussian functions, and usually require mean and variance to be specified. This is, however, a considerable limitation, and Gaussian mixture models are shown to be unsuitable in a case where true distributions of foreground or background are different, but feature the same mean and variance.

We show that, using channel density estimators, it is possible to give reliable approximations of PDFs. Several synthetic images are constructed where foreground objects and background objects are samples from different Gaussian distributions.

For the experiment, three synthetic images are constructed:

1. Foreground and background are constructed out of samples from Gaussian distributions with different means and the same variance, Figure 4.2 **(a)**.

2. Foreground and background are constructed out of samples from Gaussian distributions with the same mean and different variances, Figure 4.2 **(b)**.

3. Foreground objects are filled with samples from a uni-modal Gaussian distribution and background is filled with samples from a bimodal Gaussian distribution with the same mean and variance, Figure 4.3.

Figure 4.2 shows results of our method performed on setups 1 and 2. Our method has no problem finding objects. Figure 4.3 shows that the method has no issues separating a bimodal Gaussian distribution with the same mean and variance from a uni-modal one. Table 4.1 gives statistics of segmentation results. The quality of segmentation is measured using the dice-score measure.

The experiment is also performed using methods with available codes[1] for the following models: Original Chan-Vese implementation (smooth/non-texture images) [20], Brown *et al.* model [17] and Houhou-Thiran-Bresson model [43].

Tests performed using the original Chan-Vese implementation (smooth/non-texture images) [20] and the Brown *et al.* model are performed using 40 different parameter setups for each method and the same parameters are used on all three experiment setups. Values of parameters are hand-tuned and chosen to be close to their default values. The average result is reported. Results show that both methods are able to segment images from setups 1 and 2. All methods show to be highly sensitive to the choice of parameters for the active contour function and the same parameter choice does not necessarily give

[1]website:`https://googledrive.com/host/0B3BTLeCYLunCc1o4YzV1Ui1SeVE/codes.html`

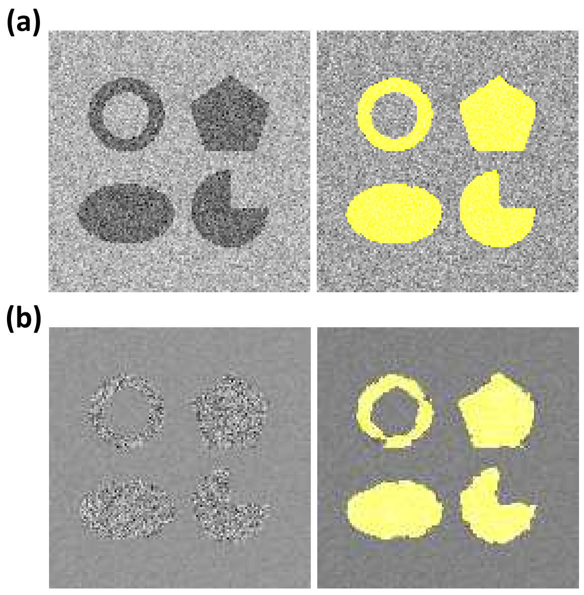


Figure 4.2: Experiment 1: The first image is the original image and the second one is the segmentation result of our method. In **(a)** the setup 1 of the experiment foreground and background are constructed out of samples from Gaussian distributions with different means and the same variance. In **(b)** the setup 2 of the experiment foreground and background are constructed out of samples from Gaussian distributions with the same mean and different variances.

correct segmentation results for all setups. Also, both methods fail at segmenting the setup 3. This is due to the fact that the Gaussian assumption can not properly approximate bimodal distributions. This observation is also documented in the work by Kim et al, where authors discuss that the approach of Yezzi *et al.* requires an appropriate a priori selection statistics for the first and second image, while it completely fails for the third image.

Results are given in Figures 4.4 -4.6 respectively, statistics are presented in Table 4.2.

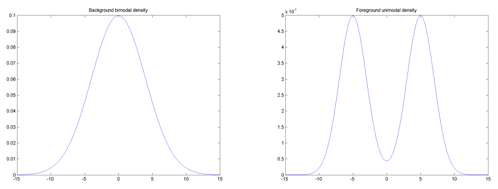
The setup 3 is successfully segmented using texture descriptors in the Houhou-Thiran-Bresson model [43]. On the other hand, this method is unable to recognize objects, when the foreground is made from samples of a Gaussian distribution and is different by mean, but not by variance from the background, Figure 4.6, **(c)**.

Our method manages to successfully locate objects in all three images with the same parameter setup. Therefore, using the channel framework, it is possible to obtain a satisfying approximation of PDFs. Our method manages to outperform methods with Gaussian assumption that use either gray-value distribution or even texture descriptors as features.

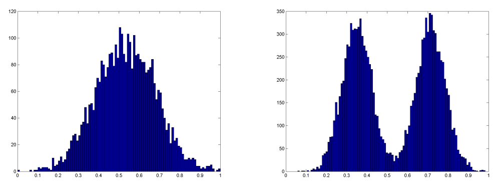
4.6.2 Experiment 2: Dependence on the initialization

Traditionally, gradient descent is used for energy functional minimization. It is implemented in works by Chan *et al.* [20], Brox and Cremers [88]. Kim *et al.* [48] and Weiler *et*

(a)



(b)



(c)

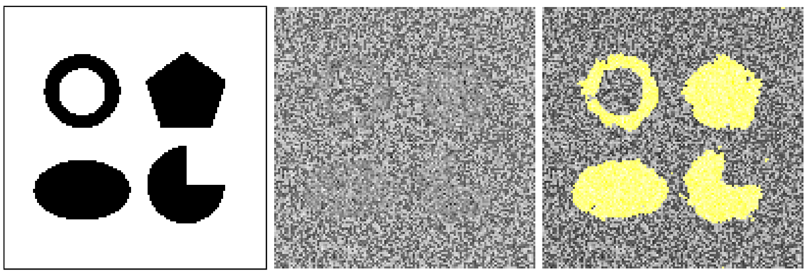


Figure 4.3: Setup 3 of Experiment 1: **(a)** foreground objects are filled with samples from a uni-modal Gaussian distribution and background is filled with samples from a bimodal Gaussian distribution with the same mean and variance; **(b)** histograms of the foreground objects and background objects; **(c)** objects that need to be found, the original image and the segmentation result of our method.

	Fig. 4.2 **(a)** –setup 1	Fig. 4.2 **(b)** –setup 2	Fig. 4.3 –setup 3
dice score	0.9904	0.9853	0.9955
num. of alg. iter.	12	12	9
est. time (s)	0.2674	0.1427	0.2747
NRI run-time (s)	0.0250	0.0144	0.0166
num. of NRI iter	74	40	44

Table 4.1: Performance of our method on synthetic images given in Experiment 1

.

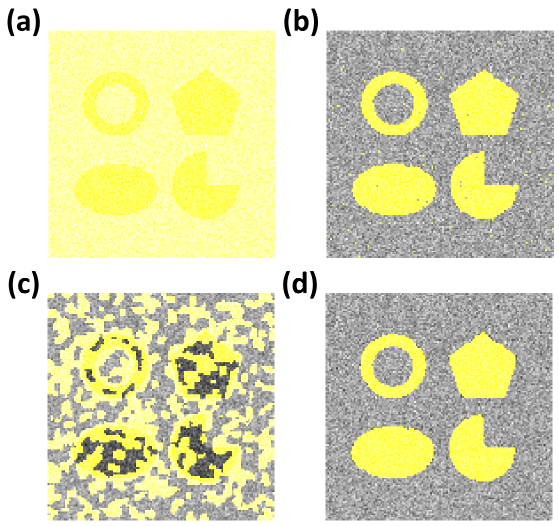


Figure 4.4: Performance of related methods on setup 1 in Experiment 1: **(a)** the Gaussian assumption model as described in Brown *et al.* [17]; **(b)** the Chan-Vese model (smooth/non-texture images) [20]; **(c)** the Houhou-Thiran-Bresson model [43]; **(d)** our method.

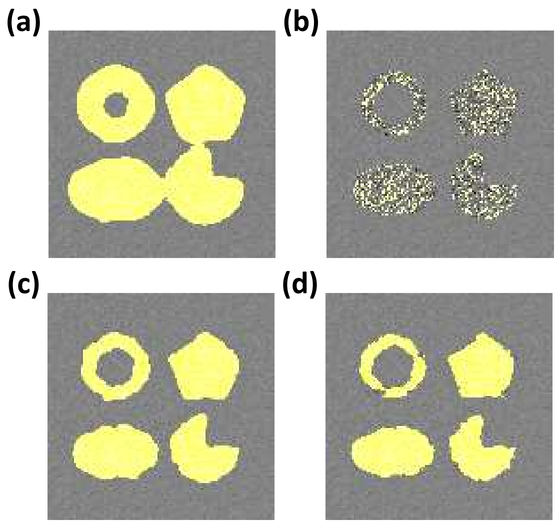


Figure 4.5: Performance of related methods on setup 2 in Experiment 1: **(a)** the Gaussian assumption model as described in Brown *et al.* [17]; **(b)** the Chan-Vese model (smooth/non-texture images) [20]; **(c)** the Houhou-Thiran-Bresson model [43] **(d)** our method.

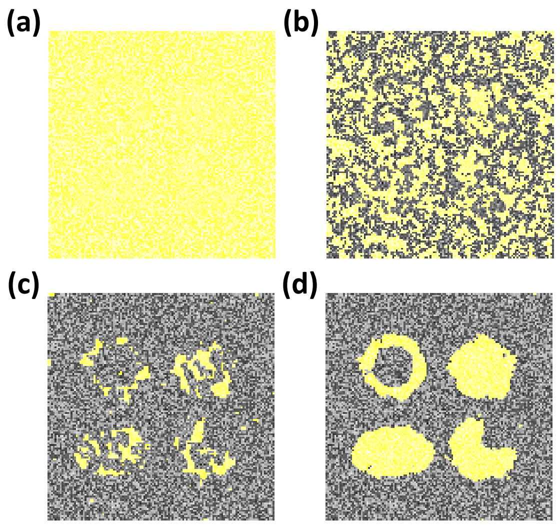


Figure 4.6: Performance of related methods on setup 3 in Experiment 1: **(a)** the Gaussian assumption model as described in Brown *et al.* [17]; **(b)** the Chan-Vese model (smooth/non-texture images) [20]; **(c)** the Houhou-Thiran-Bresson model [43]; **(d)** our method.

	(a)	**(b)**	**(c)**	**(d)**
setup 1	0.362	0.9911	0.5077	0.9904
setup 2	0.779	0.1447	0.9690	0.9853
setup 3	0.362	0.1370	0.4674	0.9955

Table 4.2: Performance in terms of dice-score of methods on setups 1, 2 and 3 in Experiment 1: **(a)** the Gaussian assumption model as described in Brown *et al.* [17]; **(b)** the Chan-Vese model (smooth/non-texture images) [20]; **(c)** the Houhou-Thiran-Bresson model [43]; **(d)** our method. The experiment shows that the CDE method is able to recognize objects in images where either foreground or background have bimodal gaussian distributions. On the other hand, Gaussian mixture models fail in the attempt as well as the Houhou-Thiran-Bresson texture model.

al. [95] as well. They all observe to get stuck in local minima.

Brown *et al.* [17] derived a convex formulation of the Chan-Vese energy functional whose solution is shown to be close to the global minimum.

Within this chapter, the NRI minimization method is used. NRI has not been used before for region segmentation, although it has already been successfully applied in image reconstruction [54] and optical flow [71]. The major drawback of the NRI scheme currently is that the convergence of the scheme is not theoretically proven.

To test if the algorithm is sensitive to the local minima, or if it reaches the global minimum of the energy functional, the dependency of segmentation to its initialization is tested. The setup of Experiment 2 is as follows:

1. The 11 images which are also chosen in the paper of Brown *et al.* [17] for demonstration of their algorithm are considered.

2. the NRI approach is used for minimization

3. the segmentation is performed 50 times for each image

As Brown *et al.* 's method is made not to take assumptions on the foreground object position, there is no reason to prefer some region during its initialization. Therefore, different initialization schemes are not investigated and only samples of uniform noise are used for initialization.

In Table 4.3, we give for each image energies which the method converged to, runtimes and numbers of iterations till convergence of the NRI. In Figures 4.7 and 4.8 segmentation results and sensitivity to the initialization are given. The left image in all pairs is an example of image segmentation result. The right image in all pairs is constructed as the mean of 50 segmentation results with different initializations. It means that regions that have highest values, the lightest regions of images, are chosen as foreground in most cases. Darker regions correspond to pixels that are chosen as foreground in only few from all 50 cases. If the image only contains white and black regions, the method converged to the same solution in all 50 cases.

The conclusions are as follows:

- To examine the sensitivity of becoming stuck in local minima, we investigate energy variances. A high variance means that the method converges to many different solutions for the given image. A low variance is related to images where the method converged to the same solution in all or most cases.

- Images of a boat, plane, landscape and cameraman converge to different solutions. It shows that NRI is sensitive to local minima. This means that further improvements to the minimization method are required or other minimization method should be considered.

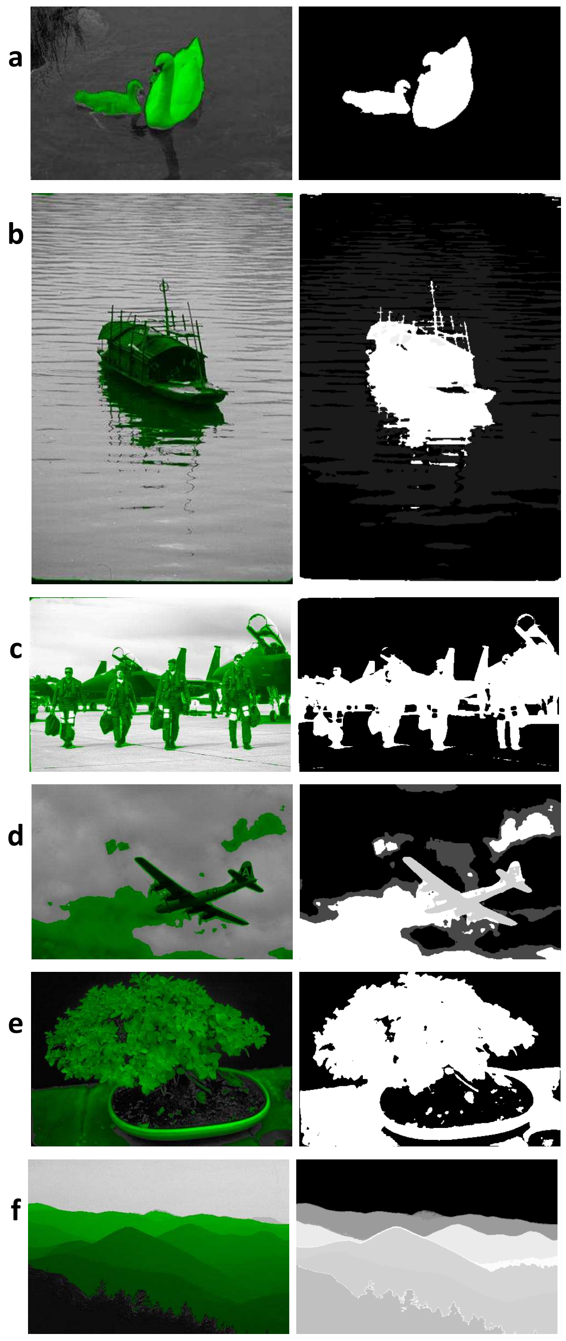


Figure 4.7: Experiment 2 **(part 1)**: Segmentation of images used in paper by Brown *et al.* [17]. The left image depicts a randomly chosen segmentation result, while the right one is the mean of 50 segmentation results with different initializations.

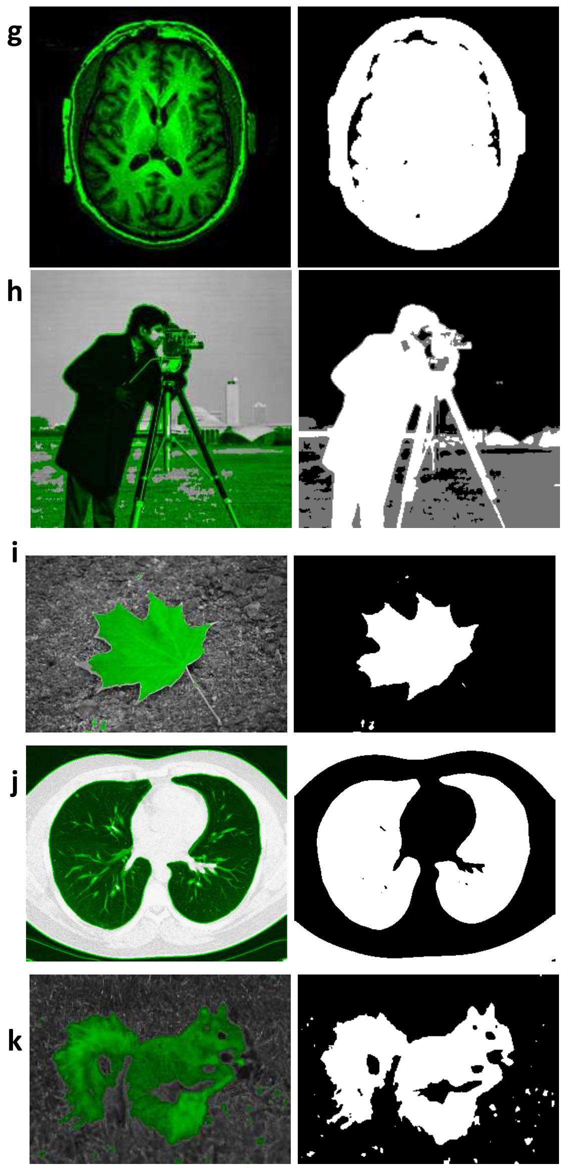


Figure 4.8: Experiment 2 **(part 2)**: Segmentation of images used in the paper by Brown *et al.* [17]. The left image depicts a randomly chosen segmentation result, while the right one is the mean of 50 segmentation results with different initializations.

	energy val.	average min. time (*s*)	num. of iter. for min.
a swans	21.2 ± 0	0.0222	31
b boat	23.3 ± 0.6	0.0502	28
c planes	24.3 ± 0	0.0557	31
d plane	21.5 ± 0.55	0.0435	24
e bonsai	24.7 ± 0	0.0474	26
f landscape	28.4 ± 3.5	0.0499	28
g brain	25 ± 0	0.0176	30
h cameraman	22.9 ± 0.9	0.0156	20
i leaf	27.6 ± 0	0.0258	37
j lungs	31.9 ± 0	0.0512	32
k squirrel	21.3 ± 0	0.0191	28

Table 4.3: Performance of our segmentation method on images from the paper by Brown *et al.* [17]
.

4.6.3 Experiment 3: Performance of the segmentation method on the Weizmann data-set

The performance of our method is tested on the data-set given in the work by Alpert *et al.* [4]. We use the data-set of 100 gray-value images each containing one object that differs from its surroundings with respect to intensity, texture or low cues. For defining the ground-truth, each image is segmented by three different human subjects. A pixel is marked as foreground if it is marked like that by two of three human subjects.

For evaluation, the dice score is used as explained in Appendix B.3. The method is trained on the first 20 images of the data set, using 50 different parameter setups. There is no rich prior to be learned, only several parameters are tuned. These parameters are the number of quantization levels, the number of channels and the variance of channel basis functions.

Segmentation is then run over all images in the data set. Minimization convergence is considered when $|\Delta\phi|$ drops below 10^{-8}. Results of performance are presented in Table (4.4) together with the results derived by Alpert *et al.* for several other segmentation algorithms, namely:

- Image Segmentation by Probabilistic Bottom-Up Aggregation and Cue Integration [4]
- Segmentation by weighted aggregation (SWA) [86] with all features described in [32] (SWA V1)
- SWA with intensity contrast and filter responses as features (SWA V2) [86]
- Normalized cuts segmentation including intervening Contours [58]
- Mean-Shift [24] with intensity cues as features

- Globalized probability of a boundary (gPb) algorithm [6]

The method by Alpert *et al.* [4] has the best dice-score, 0.86 ± 0.01. Here, only automatic segmentation methods are considered, while Bagon *et al.* [8] developed an interactive method and reported even higher dice-score, 0.87 ± 0.01. Our method comes on the fifth place of presented algorithms.

Performance of our method can be seen in Table 4.5. In Figures 4.9, 4.10 and 4.11, we see examples of segmentation results. Segmentation statistics of mentioned images are given in Table 4.6.

The following observations can be made:

- Figure 4.9 gives examples of images where the method performed best. As seen on the figure, all images have a large object as foreground and foreground and background areas are similar in size, $|\Omega_b| \sim |\Omega_f|$.

- Figure 4.10 gives us some insights what are problematic areas to segment for the method. These images contain an object distinguishing from background, but images contains many variates in texture. Therefore, sometimes parts of background are fused with foreground or some foreground parts are considered background. These results are a consequence of the segmentation method getting stuck in a local minimum.

- Our method is unable to adapt to the posed problem in images that performed the worst, Figure 4.11. This is because some of these images have more than one distinguishable object, but human observers selected the only human made object as foreground. *E.g.* in Figure 4.11 **(b)** there is a pen, a box and a wall. All users, however, have chosen the pen as a foreground object. The segmentation algorithm distinguishes between the wooden box as one region, and the wall together with the pen as the other region.

Our method does not outperform all other state-of-the-art methods, as it only considers a gray-value distribution as the criterion, no additional cues. Also, no additional image pre-processing is performed on images.

4.7 Summary

The method presented within the chapter is aimed at unsupervised segmentation of gray-value images in two regions. An image is divided in two regions that have most distinguishing gray-value distributions.

Two novelties to traditionally used methods are introduced:

1. the use of *the channel-framework* to approximate probability densities.

2. implementation of *the NRI minimization method* for solving the Chan-Vese level set approach.

Algorithm	dice-score
method by Alpert *et al.* [4]	0.86 ± 0.01
SWA V1	0.83 ± 0.02
SWA V2	0.76 ± 0.02
N-Cuts	0.72 ± 0.02
our approach	0.61 ± 0.005
MeanShift	0.57 ± 0.02
Gpb	0.54 ± 0.01

Table 4.4: The single-segment coverage test results for the single-object data-set table from the paper by Alpert *et al.* [4]. While our method uses just the intensity distribution in distinguishing two regions, other methods consider texture or low cues as well while segmenting an image.

dice score	0.61 ± 0.005
algorithm iterations	22 ± 13
estimation time	$0.61 \pm 0.26s$
run-time of NRI	$0.62 \pm 0.27s$
number of iterations for NRI	23 ± 4

Table 4.5: Performance of our method on the data-set given in [4]

	Fig. 4.9			Fig. 4.10			Fig. 4.11	
	a	b	c	a	b	c	a	b
dice score	0.9656	0.9691	0.9735	0.6539	0.6610	0.6836	0.1732	0.1420
num. of alg. iter.	24	20	12	31	18	12	22	14
est. time (*ms*)	0.2674	0.1427	0.2747	0.1520	0.1528	0.1412	0.2718	0.1516
NRI run-time (*ms*)	0.2073	0.0941	0.1687	0.0700	0.1108	0.1209	0.0845	0.0859

Table 4.6: Performance of our method on the data-set given in [4]

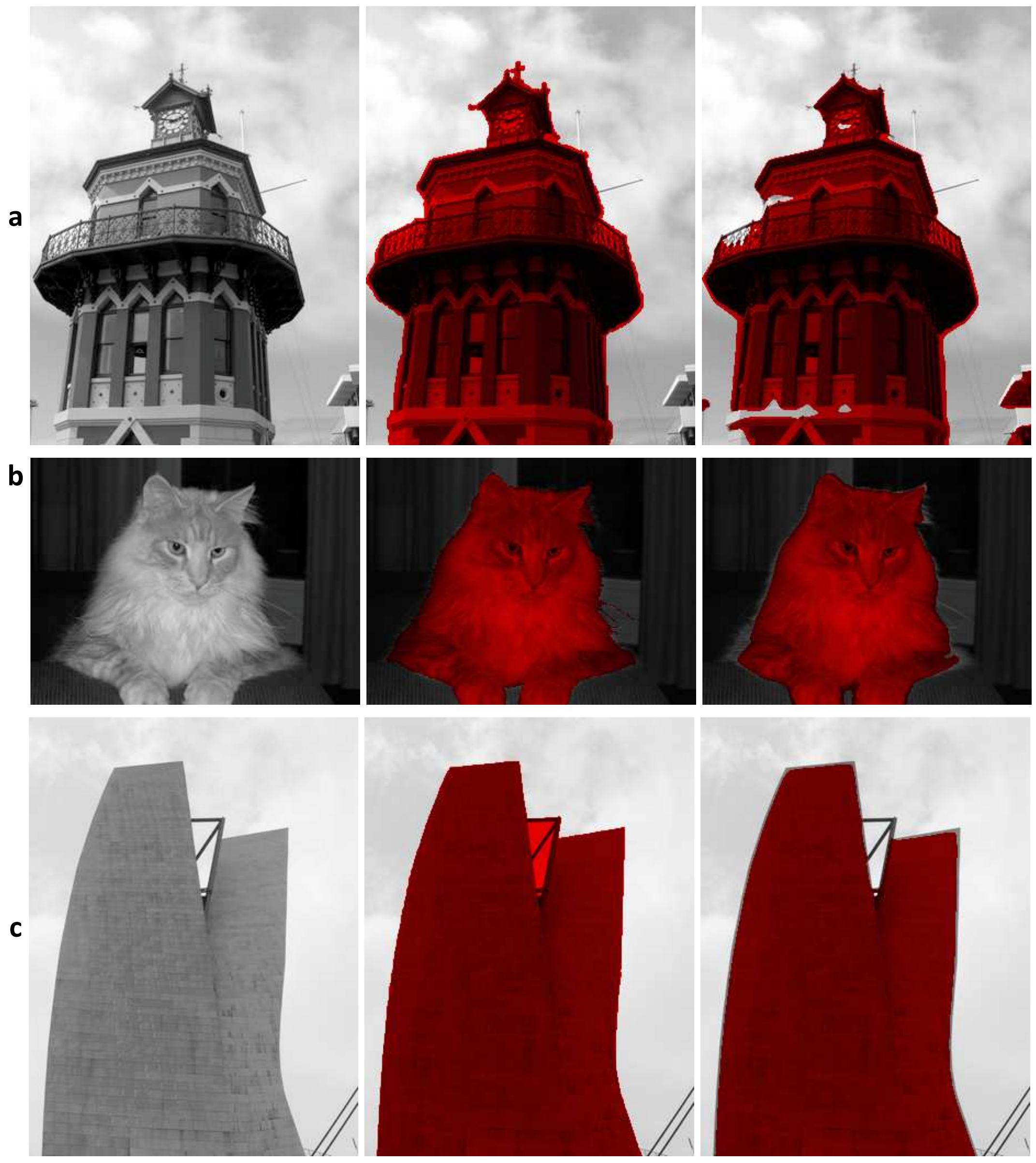


Figure 4.9: Images where our method gave the best dice-score result. Columns respond to the original image, the ground-truth segmentation and the result of our method respectively.These images have two clearly distinguishing objects that contrast in intensities of foreground and background (while one has mostly higher intensities, the other has lower intensities), and quite different dynamics in intensity (mean gradient vallue) and therefore quite different distributions.

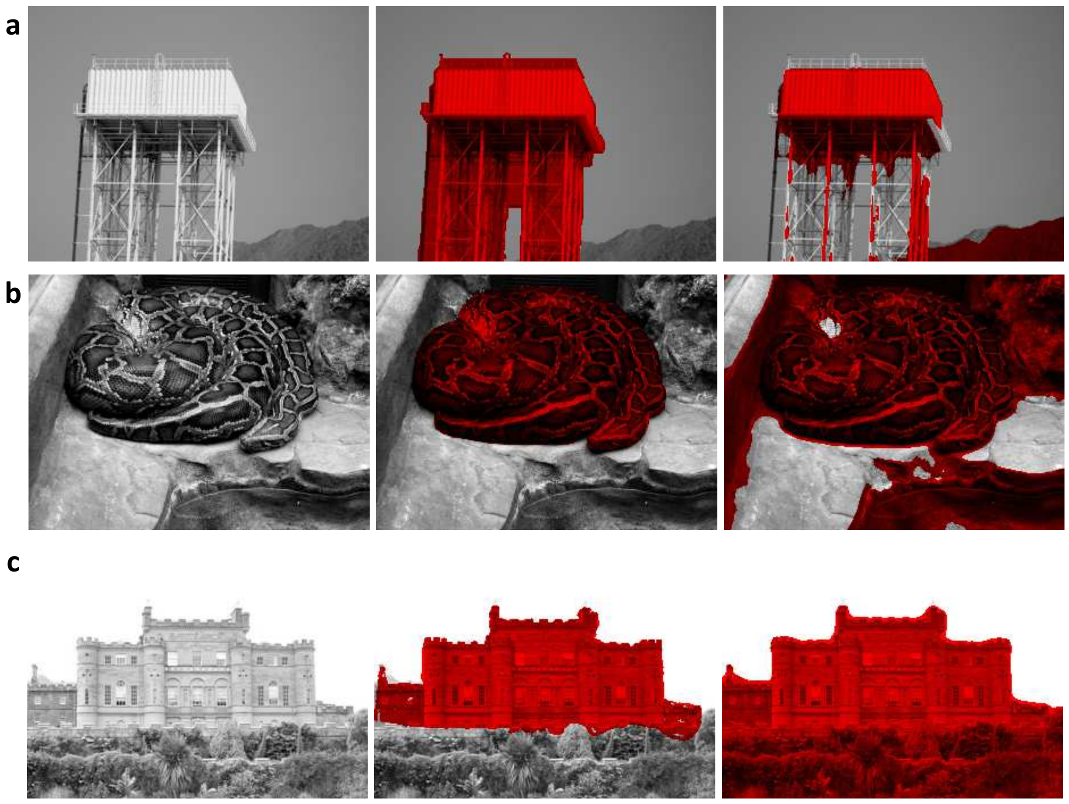


Figure 4.10: Images where our algorithm gave an average dice score result. Columns respond to the original image, the ground-truth segmentation and the result of our method respectively.

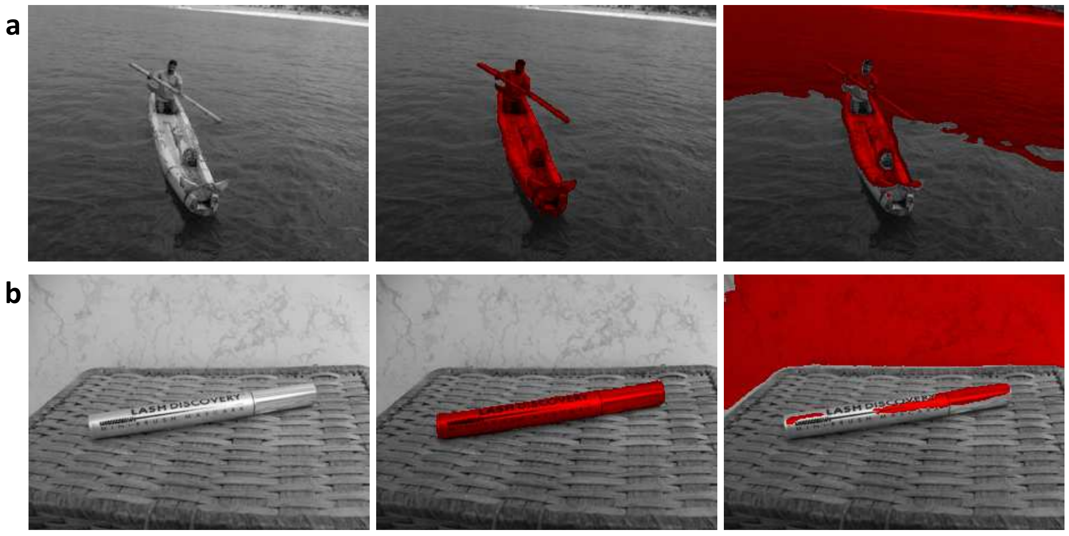


Figure 4.11: Images where our algorithm gave the poorest dice-score result. Columns respond to the original image, the ground-truth segmentation and the result of our method respectively.

Experiments separately demonstrate performance of the segmentation method regarding the choice of the data term and performance of the NRI minimization method.

- Experiment 1 shows that our channel approximation of probability distribution functions contains rich information. It is shown that the method outperforms the Gaussian assumption model. Methods that use KDE estimation, *e.g.* Kim *et al.* , or histograms, *e.g.* Ni *et al.* [67] are also able to segment images presented in the experiment, as shown in respective papers. As discussed in Chapter 3, the channel framework requires less bins than histograms for achieving similar results. It is also faster to compute PDFs than using kernel density estimators.

- Experiment 2 shows the sensitivity of the NRI method to its initialization. We see that even in MATLAB implementation, NRI converges fast to a minimum, Table 4.3. Still, the experiment shows that the NRI method is not immune against getting stuck in local minima.

- Experiment 3 shows performance of our method on the Weizmann data-set. While our method does not outperform other state-of-the-art ones, it gives comparable results although only gray intensity distribution is used as a segmentation criterion.

In conclusion, the CDE approach to solving Chan-Vese functional shows to be promising, but requires additional features to be considered in the segmentation process in order to compete with the state-of-the-art methods. Also, the NRI approach needs additional investigation in order to avoid local minima.

Chapter 5

2-region segmentation using color distribution prior knowledge

5.1 Introduction

In this chapter we present an extension to the method described in Chapter 4. Prior knowledge on color distributions of image regions is introduced. The targeted object is predefined and it is known how its distribution looks like. This is then called supervised segmentation.

The novelty presented in this chapter is the ability to adjust the weight coefficient for the prior knowledge and therefore adjusting the trade off between a completely supervised and completely unsupervised segmentation. The method also aims at color images, or more general vector images.

Having both the supervised and the unsupervised segmentation is useful in cases where the prior is formed from data which does not fully describe the foreground object and may also contain false information on its color distribution. In such a case, the unsupervised part can improve the foreground prior.

The combination of the supervised and unsupervised method therefore, comes useful in tools for rough selections of an object. We examine the following:

- A user can roughly select some parts of an object on an image to mark its position and areas from which prior knowledge should be calculated. A user specifies hard constraints that some areas are definitely foreground or background, while a segmentation method needs to decide on unspecified pixels. Examples of different user hints to segment an image are given in Chapter 2.

- Prior knowledge can be calculated on a different image or a set of images. Images that are used as an example of foreground or background are called the training data. In Section 5.5, a method is trained on the leaf collection data-set to recognize tobacco plants on images in the tobacco plant data-set, see Chapter B.

- Prior knowledge can be given by a user designed channel density estimate of foreground or background.

Another addition to the method in the previous chapter is the use of a distance map for introducing the spatial dependency of foreground and background. A distance map is formed based on location of scribbles or lasso trimaps.

5.2 Related methods

Boykov and Jolly present their interactive graph cuts based image segmentation method in [14]. They define an image energy cost as a linear combination of two cost functions that describe region and boundary properties, respectively. Regularization terms give a trade-off between region and boundary based segmentation. The region cost function is calculated from gray-value intensities that are marked by a user as foreground or background. Using histograms, they approximate intensity distributions.

For energy minimization they use the max-flow algorithm as described in [15]. Regularization terms are usually fixed, while other parameters are optimized during the minimization process. However, they limit their research on images that contain scalar values, *i.e.* gray-value images. They do not handle color images as they require a suitable histogram representation of color distribution for vector valued images.

The GrabCut data-set [1] for foreground and background segmentation was developed for papers by Rother *et al.* [76] and Blake *et al.* [9]. The data-set contains a set of trimaps that simulates lasso selection and a set that simulates bounding-box selection. The data-set is evaluated using the average misclassification error E_{miss}.

In the paper by Blake *et al.* [9] they suggest the Gaussian Mixture Markov Random Field model to represent even vector valued intensities like color images in a compact form. They test the performance of their algorithm on a lasso selection data-set.

Nieuwenhuis and Cremers [69] use kernel density estimates for approximating color distributions. They also combine a color information with a spatial information of prior data. Their approach is mainly aimed at multi-label segmentation, but is also evaluated on the GrabCut data-set. They evaluate their approach only on lasso-selection trimaps, while they dismiss the bounding-box as an unsuitable selection for their method. This is because bounding-boxes contain both pixels that belong to foreground and pixels that belong to background.

In the paper by Rother et al, authors test their method on the GrabCut bounding box selection data-set. From a thin outer line of the bounding box, prior knowledge on background is calculated. On the other hand, there is no prior knowledge on foreground. They observe that, while in some cases their algorithm gives good segmentation results, in other cases selection does not contain enough information on what the targeted object is and additional user interaction is needed. In those cases, additional brush strokes are added to define regions.

Lempitsky *et al.* [55] test the bounding-box selection. They also use Gaussian mixture models for representing data-term. They introduce new bounding-boxes to the Grab-cut data-set and their *tightness*. Tightness is defined using margins for the bounding boxes marked by a user to restrict segmentation attention to the interior.

[1]GrabCut dataset: `http://tinyurl.com/grabcut`

Our method is tested on the GrabCut data-set and evaluated using the E_{miss} measure. In addition to color distribution, where possible, the spatial information on foreground and background is considered.

5.3 Solving the Chan-Vese functional

The Chan and Vese level set approach of (4.1), described in detail in Chapter 2, Section 2.4.2, solves the energy functional introducing the level-set function ϕ. Evolving the level-set function ϕ, the functional is minimized and segmentation results are obtained. Foreground and background regions are regions where ϕ has positive and negative values, respectively. The zero-level of the level-set function ϕ corresponds to the boundary. Probability distribution functions of foreground and background are approximated using the channel framework. The functional is then minimized using the NRI method as described in Chapter 4.

The difference *wrt.* Chapter 4 is the interpretation of probabilities $p(\boldsymbol{g}|\Omega_i)$. We recognize several different cases. The method can be:

1. fully unsupervised, where no prior knowledge is given, *i.e.* the case of Chapter 4;
2. fully supervised, where channel density estimates of regions are completely predefined based on the prior knowledge.
3. semi-supervised, where regularization terms that serve to adjust the influence of prior knowledge are introduced;

In the unsupervised method, channel density estimates are calculated from regions where the level set function is $\phi > 0$ for foreground and $\phi < 0$ for background according to:

$$\begin{aligned}\hat{\boldsymbol{q}}_{\phi>0} &= \frac{\int_\Omega H(\phi(\boldsymbol{x}))\hat{\boldsymbol{c}}(g(\boldsymbol{x}))d\boldsymbol{x}}{\int_\Omega H(\phi(\boldsymbol{x}))d\boldsymbol{x}};\\ \hat{\boldsymbol{q}}_{\phi<0} &= \frac{\int_\Omega (1-H(\phi(\boldsymbol{x})))\hat{\boldsymbol{c}}(g(\boldsymbol{x}))d\boldsymbol{x}}{\int_\Omega (1-H(\phi(\boldsymbol{x})))d\boldsymbol{x}}.\end{aligned} \tag{5.1}$$

In the supervised method channel density estimates are calculated from a-priori given regions Ω_{fu} and Ω_{bu} that represent foreground and background:

$$\begin{aligned}\hat{\boldsymbol{q}}_{fu} &= \frac{\int_{\Omega_{fu}} \hat{\boldsymbol{c}}(g(\boldsymbol{x}))d\boldsymbol{x}}{|\Omega_{fu}|};\\ \hat{\boldsymbol{q}}_{bu} &= \frac{\int_{\Omega_{bu}} \hat{\boldsymbol{c}}(g(\boldsymbol{x}))d\boldsymbol{x}}{|\Omega_{bu}|}.\end{aligned} \tag{5.2}$$

In the semi-supervised method, channel density estimates of previous two cases are combined:

$$\begin{aligned}\hat{\boldsymbol{q}}_f &= \omega_f\hat{\boldsymbol{q}}_{\phi>0} + (1-\omega_f)\hat{\boldsymbol{q}}_{fu}; \quad 0 < \omega_f < 1\\ \hat{\boldsymbol{q}}_b &= \omega_b\hat{\boldsymbol{q}}_{\phi<0} + (1-\omega_b)\hat{\boldsymbol{q}}_{bu}; \quad 0 < \omega_b < 1\end{aligned} \tag{5.3}$$

ω_f and ω_b are regularization parameters that adjust the trade-off between the supervised and the unsupervised approach. Choosing $\omega_f = 0$ and $\omega_b = 0$ we get a fully supervised segmentation. Values $\omega_f = 1$ and $\omega_b = 1$ lead to fully unsupervised segmentation.

Unsupervised and semi-supervised methods can be:

1. a one-step segmentation method

2. an iterative segmentation method

In one-step segmentation, channel density estimates of regions are calculated only once. In the iterative approach, prior knowledge is updated with segmentation results of previous iterations and CDEs are recalculated.

The main difference between the iterative and non-iterative approach is explained in Chapter 4. It is related to the minimization procedure. To recap, we tried to neglect the $\delta\boldsymbol{B}$ in Equation (4.20) because it requires a large number of calculations and is, therefore, computationally costly. If the one-step segmentation approach is used, $\delta\boldsymbol{B}$ should be calculated within the minimization process. According to [44, 48], ignoring $\delta\boldsymbol{B}$, in most cases, does not change results, but it can in special cases cause the loss of accuracy at corners of the foreground object.

In the iterative approach, PDFs are held fixed during the minimization process and updated over iterations. This allows to neglect $\delta\boldsymbol{B}$ (see Section 4.4).

Having CDEs of foreground and background regions, it is possible to calculate probability distribution functions using the following equations:

$$\begin{aligned} p(\boldsymbol{g}|\Omega_f) &\propto \sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g},k,i,j)\hat{q}_f(k,i,j); \\ p(\boldsymbol{g}|\Omega_b) &\propto \sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g},k,i,j)\hat{q}_b(k,i,j). \end{aligned} \tag{5.4}$$

Compare Section A.1 for notation. Scribbles and lasso selection allow spatial dependence to be introduced in calculating PDFs. This dependance is introduced by a distance function σ.

$$\begin{aligned} p(\boldsymbol{g}(\boldsymbol{x})|\Omega_f) &\propto \Big(\sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g}(\boldsymbol{x}),k,i,j)\hat{q}_f(k,i,j)\Big)\cdot\sigma_f(\boldsymbol{x}); \\ p(\boldsymbol{g}(\boldsymbol{x})|\Omega_b) &\propto \Big(\sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g}(\boldsymbol{x}),k,i,j)\hat{q}_b(k,i,j)\Big)\cdot\sigma_b(\boldsymbol{x}). \end{aligned} \tag{5.5}$$

The distance function should give high probability of outcome for pixels that are close to scribbles. For distant pixels, the probability should be lower. The distance functions σ_f and σ_b are calculated as described in (3.22):

$$\begin{aligned}\sigma_f(x,y) &= e^{-\min\limits_{(x_u,y_v)\in\Omega_f^{us}} \gamma\sqrt{(x-x_u)^2+(y-y_v)^2}} \\ \sigma_b(x,y) &= e^{-\min\limits_{(x_u,y_v)\in\Omega_b^{us}} \gamma\sqrt{(x-x_u)^2+(y-y_v)^2}}\end{aligned}, \tag{5.6}$$

where $\gamma \in \mathbb{R}+$.

5.4 Implementation details

The one-step segmentation method is organized in the following way:

1. choose an image $\boldsymbol{g} \in \mathbb{N}^{m\times n}$ to segment
2. acquire data to calculate prior knowledge on foreground and/or background
3. initialize ϕ
4. calculate $\hat{\boldsymbol{q}}_{\phi>0}$ and $\hat{\boldsymbol{q}}_{\phi<0}$ according to Equations (5.1). Their numerical implementation is given in Equation (5.9).
5. calculate $\hat{\boldsymbol{q}}_{fu}$ and $\hat{\boldsymbol{q}}_{bu}$ according to Equation (5.2). Their numerical implementations are given in (5.10)
6. calculate CDEs $\hat{\boldsymbol{q}}_f$ and $\hat{\boldsymbol{q}}_b$ according to Equation (5.11)
7. calculate $p(\boldsymbol{g}|\Omega_f)$ and $p(\boldsymbol{g}|\Omega_b)$
8. minimize the energy functional $E(\phi)$, Equation (4.1)

The iterative approach does the same but in addition updates prior knowledge in each iteration:

9. repeat steps 4-8 with the updated ϕ.

ϕ is initialized as a random vector with the same size as the image $\boldsymbol{g}$, $m \times n$. Here, we segment color images. RGB images are made of three signals $\boldsymbol{g}(x,y) = \boldsymbol{s} = [s^R, s^G, s^B]$. Channel vectors are calculated separately for each signal $s^i \in \{s^R, s^G, s^B\}$ using Gaussian functions as basis functions.

$$\begin{aligned}\boldsymbol{c} &= (c_1, c_2, \ldots, c_{N_c}); \\ \alpha &= \frac{\Delta_n^2}{2\sigma^2}\left(\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right)^2; \\ c_n(s) &= \begin{cases} e^{-\alpha}, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| < \Psi \\ 0, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| \geq \Psi \end{cases}; \\ \Delta_s &= s_{max} - s_{min}; \\ \Delta_n &= N_c - 1\end{aligned} \tag{5.7}$$

For RGB images, signals usually take integer values from interval $[s_{min}, s_{max}] = [0, 255]$. Channel tensors are formed as:

$$\hat{\boldsymbol{c}}(s^R, s^G, s^B) = [c(s^R) \otimes c(s^G) \otimes c(s^B)] \tag{5.8}$$

Channel densities for unsupervised and semi-supervised methods are calculated from areas where the level-set function is positive $(\phi > 0)$ for the foreground and where the level-set function is negative $(\phi < 0)$ for the background.

$$\begin{aligned}
\hat{q}_{\phi>0}(k,i,j) &= \frac{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} H(\phi(x,y))\hat{c}(g(x,y),k,i,j)}{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} H(\phi(x,y)))} \\
\hat{q}_{\phi<0}(k,i,j) &= \frac{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} (1-H(\phi(x,y)))\hat{c}(g(x,y),k,i,j)}{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} (1-H(\phi(x,y)))} \\
k,i,j &= 1,\dots,N_c
\end{aligned} \tag{5.9}$$

Channel density estimates for the semi-supervised and the supervised method are calculated from a-priori given regions Ω_f^{us} and Ω_b^{us} that represent foreground and background:

$$\begin{aligned}
h_f(x,y) &= \begin{cases} 1, & (x,y) \in \Omega_f^{us}; \\ 0, & \text{otherwise} \end{cases} \\
\hat{q}_f^{us}(k,i,j) &= \frac{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} h_f(x,y)\hat{c}(g(x,y),k,i,j)}{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} h_f(x,y)} \\
h_b(x,y) &= \begin{cases} 1, & (x,y) \in \Omega_b^{us}; \\ 0, & \text{otherwise} \end{cases} \\
\hat{q}_b^{us}(k,i,j) &= \frac{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} h_b(x,y)\hat{c}(g(x,y),k,i,j)}{\sum\limits_{x=1}^{m}\sum\limits_{y=1}^{n} h_b(x,y)} \\
k,i,j &= 1,\dots,N_c
\end{aligned} \tag{5.10}$$

For the semi-supervised methods, combined CDEs are formed as a combination of terms presented in Equations (5.9) and (5.10), according to:

$$\begin{aligned}
\hat{\boldsymbol{q}}_f &= \omega_f \hat{\boldsymbol{q}}_{\phi>0} + (1-\omega_f)\hat{\boldsymbol{q}}_{fu}; \quad 0 < \omega_f < 1 \\
\hat{\boldsymbol{q}}_b &= \omega_b \hat{\boldsymbol{q}}_{\phi<0} + (1-\omega_b)\hat{\boldsymbol{q}}_{bu}; \quad 0 < \omega_b < 1
\end{aligned} \tag{5.11}$$

Approximations of PDFs are given by the following equations:

$$\begin{aligned} p(\boldsymbol{g}|\Omega_f) &= \sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g}(x,y),k,i,j)\hat{q}_f(k,i,j) \\ p(\boldsymbol{g}|\Omega_b) &= \sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g}(x,y),k,i,j)\hat{q}_b(k,i,j) \end{aligned} \tag{5.12}$$

In the case where a distance function is introduced, the PDFs have following form:

$$\begin{aligned} p(\boldsymbol{g}|\Omega_f) &= \sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g}(x,y),k,i,j)\hat{q}_f(k,i,j)\cdot\sigma_f, \\ \sigma_f(x,y) &= e^{-\min\limits_{(x_u,y_v)\in\Omega_f^{us}} \gamma\sqrt{(x-x_u)^2+(y-y_v)^2}} \\ p(\boldsymbol{g}|\Omega_b) &= \sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{g}(x,y),k,i,j)\hat{q}_b(k,i,j)\cdot\sigma_b; \\ \sigma_b(x,y) &= e^{-\min\limits_{(x_u,y_v)\in\Omega_b^{us}} \gamma\sqrt{(x-x_u)^2+(y-y_v)^2}} \end{aligned} \tag{5.13}$$

where $\gamma \in \mathbb{R}+$.

A look-up table is used to encode the channels. The methods are tested with around 40 different parameter setups and the best case is reported. For each method, parameters are generated around their default values.

5.5 Experiments

In experiments 1 and 2, the procedure using this method with a possible application is presented. It is required to locate a plant, by selecting a part of it, or providing another set of images for its description. There should also be a parameter to allow us to adjust the level of supervision when locating objects. To locate objects, especially plants, it is advised to use more different features to describe them and not just the color distribution. Therefore, the training and test data-set are chosen just large enough to show the performance of the method, while in real application, these should be much larger, the test data-set of around 100 images and around 20% of images for the training set.

In experiments $\eta = 80$ quantization values and $N_c = 20$ channels are used. The look-up table has $5 \times 5 \times 5 = 125$ non-zero channel values for each pixel value. Within this chapter $double$ precision type is used for presenting the channel values resulting in needed $8 \times 125 \times 80 = 80KB$. CDEs are tensors of 20 vectors containing 3 different values. Therefore, one CDE requires maximum $20 \times 20 \times 20 \times 8 = 64KB$.

5.5.1 Experiment 1: Influence of the trade-off regularization parameter between supervised and unsupervised segmentation

Since the segmentation method uses only the color distribution as a criterion for segmenting foreground, the experiment performed on the tobacco-plant data-set is limited to recognizing a plant on an image. From the data-set 30 images are chosen. The foreground prior is defined by a rectangle selection marked by user that contains only parts of foreground. Based on this selection, the prior CDE $\boldsymbol{q}_f^{us}$ is calculated and combined with the CDE $\boldsymbol{q}_{\phi>0}$ calculated from values where the level-set function is positive, $\phi > 0$: There is no prior knowledge on background, so, CDE of the background region is calculated on region where the level-set function is negative, $\phi < 0$. Different values of a regularization parameter ω_f are used. This parameter adjusts the trade-off between a fully automatic and a fully supervised segmentation.

The experiment is performed as follows:

1. 80 quantization values and 20 channels are used.

2. After each iteration, $\boldsymbol{q}_{\phi>0}$ and $\boldsymbol{q}_{\phi<0}$ are recalculated. Posterior distribution functions of foreground and background are calculated using only color distribution and no spatial information.

3. The result is considered final after 5 iterations.

Examples of segmentation results are given in Table 5.1 and Figures 5.1 and 5.2.
We observe the following:

1. Choosing $\omega_f = 0$ in Equation (5.11) results in a fully supervised case. Therein, only parts of leaves are chosen as foreground region whose densities are similar to the CDE $\boldsymbol{q}_f^{us}$ built up from user's selection. The foreground CDE is built only from values contained in a user specified rectangle.

2. Small values of $\omega_f > 0$ result in adding tolerance to other pixel values on the image, while still preserving a similar foreground density estimate as when only prior knowledge is used. This increases the dice-score in most cases.

3. For values $\omega_f > 0.5$, $\boldsymbol{q}_{\phi>0}$ has more influence than $\boldsymbol{q}_f^{us}$ in $\boldsymbol{q}_f$ and the posterior distribution function $p(\boldsymbol{x}|\Omega_f)$ is more similar to distribution built in the automatic segmentation case. In Figures 5.1 and 5.2, we notice that parts of the floor around plants are being added to foreground as well, *e.g.* the case **a)** for values $\omega_f \geq 0.5$.

4. For $\omega_f = 1$ we obtain the automatic segmentation (unsupervised segmentation) case. This case gives as a result of segmentation two regions that have different color distributions, while none of them has to be similar to the plant/leaf distribution. Therefore, large parts of the floor around plants are chosen as foreground.

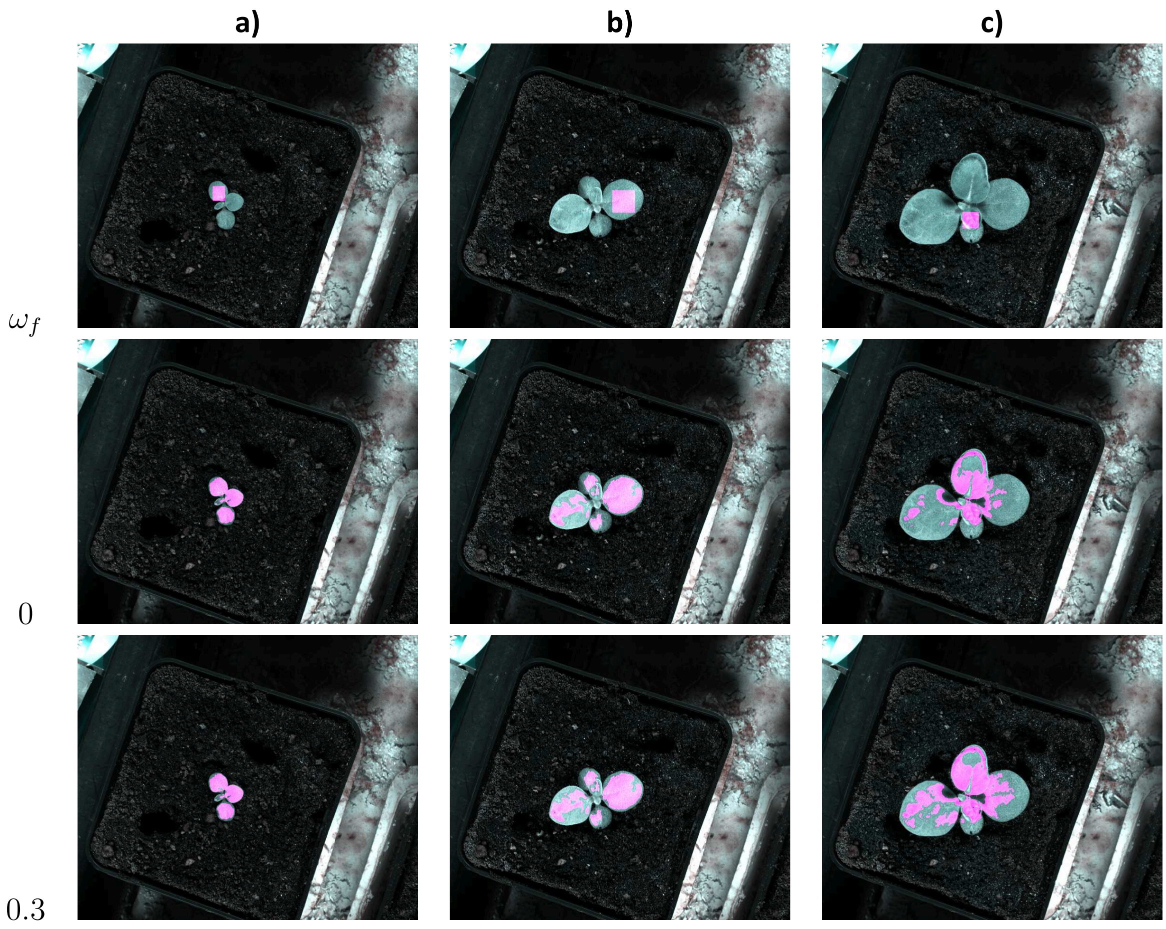


Figure 5.1: Experiment 1: The influence of regularization parameter ω_f (part 1). The first row contains images of a tobacco plant with a rectangle area selected by a user to form prior knowledge. Each next row contains segmentation results with a different choice of ω_f. $\omega_f = 0$ corresponds to fully supervised segmentation, while $\omega_f = 1$ corresponds to fully unsupervised segmentation. Dice-score results are given in Table 5.1

dice-score	$\omega_f = 0$	$\omega_f = 0.3$	$\omega_f = 0.5$	$\omega_f = 0.7$	$\omega_f = 1$
a)	0.9304	0.9236	0.6774	0.5615	0.4856
b)	0.8268	0.8303	0.8316	0.8212	0.5732
c)	0.7087	0.8098	0.8720	0.8982	0.6412

Table 5.1: Experiment 1: The influence of regularization parameter ω_f. The table shows the dice-score performance of our method on images given in Figures 5.1 and 5.2.

5.5.2 Experiment 2: Performance of our segmentation method when prior knowledge training and segmentation are performed on different images

This experiment tests if it is possible to use different images as training data such that it is not necessary that a user marks interesting regions on any image to be processed later.

The data-set from Appendix B.2 with $\omega_f = 0$ is used for the test. We train the color distribution prior from 10 leaves from the leaves collection of a tobacco plant. Then, we run the segmentation method on 50 images of a tobacco plant.

Examples of training data and result images are given in Figure 5.3. The average dice-score performance is 0.9271.

This means that it is not obligatory that objects are selected on every image if a set of images has properties that are similar enough. Instead, a different data-set describing the object - object template, can be used to obtain prior knowledge. The segmentation method will locate targeted object on given set of images. This is a quite common approach in training classifiers.

5.5.3 Experiment 3: Evaluating our method on the GrabCut benchmark using lasso prior

This experiment examines performance of our method on the Grab-cut lasso selection data-set. The data-set contains 50 different natural images showing one object distinct from background.

Lasso selection requires a user to roughly select edges of a foreground object. Lasso trimaps imitate a lasso selection of a foreground. Trimaps partition an image in three regions: Pixels that definitely belong to background, pixels that definitely belong to foreground and unknown pixels. The segmentation task is to classify unknown pixels. For the evaluation of methods *the average misclassification error* is used (See Appendix B.3.4)

We used 10 channels for 50 quantization levels. In each iteration, the foreground prior is updated with values where $\phi > 0$. While calculating probabilities of foreground and background, location is also taken into consideration.

Performance of our algorithm is compared to results presented in the paper by Nieuwenhuis and Cremers [69]. Nieuwenhuis and Cremers test their interactive multi-label method on the GrabCut benchmark. They combine color information with spatial information of prior data. Their approach is compared to the GrabCut [76, 9] and to Random Walker presented in [37]. Results are given in Table 5.2.

We obtain better results than ones tested in paper by Nieuwenhuis and Cremers [69], except for the Random Walker presented in [37].

In most cases, parts of objects have the same color distribution as background, see *e.g.* Figure 5.5. In Figure **d)** legs of the kangaroo have the same color distribution as the ground next to it, while they contain different colors from the rest of its body. Therefore, they are assigned to background.

To get an idea of how our segmentation method works, we discuss segmentation of an image in Figure 5.6. Colors of trimap have following meaning:

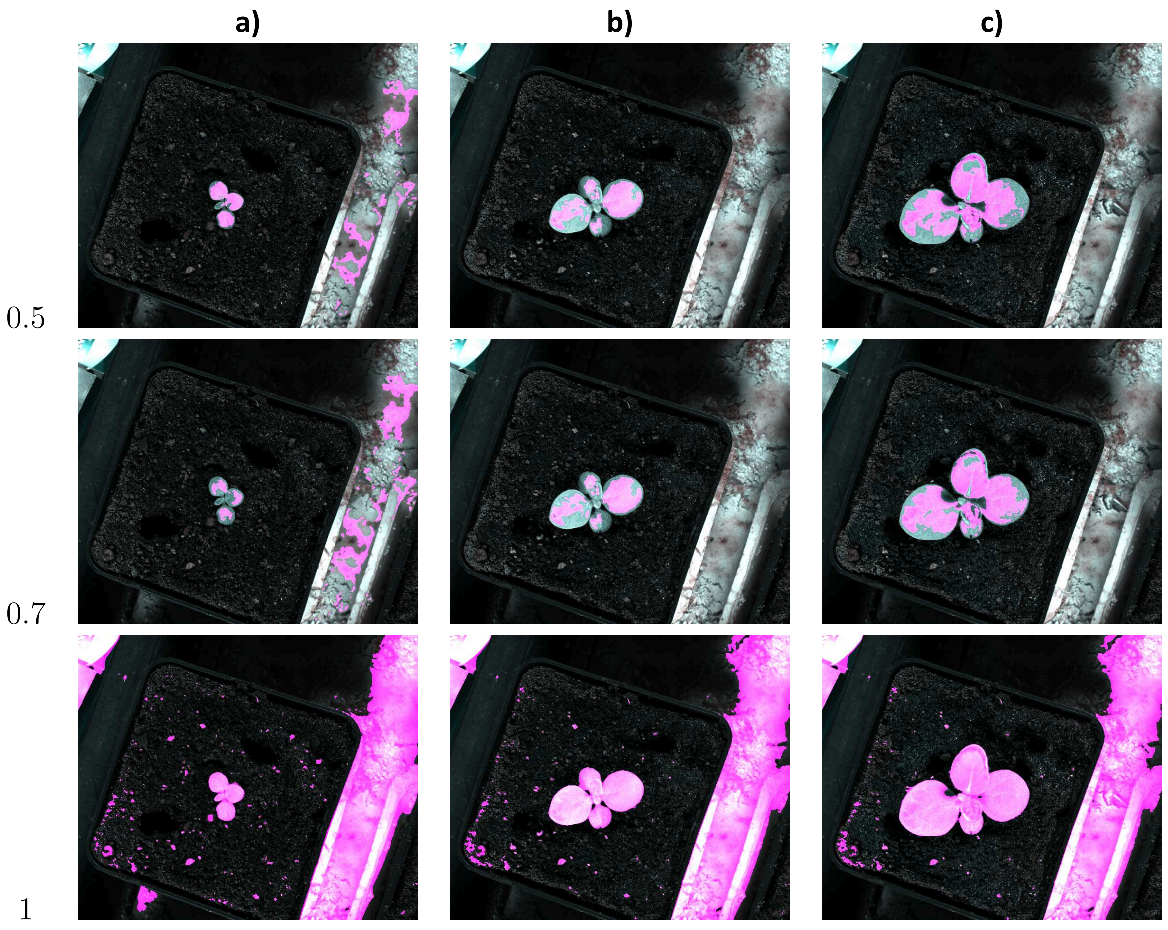


Figure 5.2: Experiment 2: The influence of regularization parameter ω_f (part 2). The first row contains images of a tobacco plant with a rectangle area selected by a user to form prior knowledge. Each next row contains segmentation results with a different choice of ω_f. $\omega_f = 0$ corresponds to fully supervised segmentation, while $\omega_f = 1$ corresponds to fully unsupervised segmentation. Dice-score results are given in Table 5.1

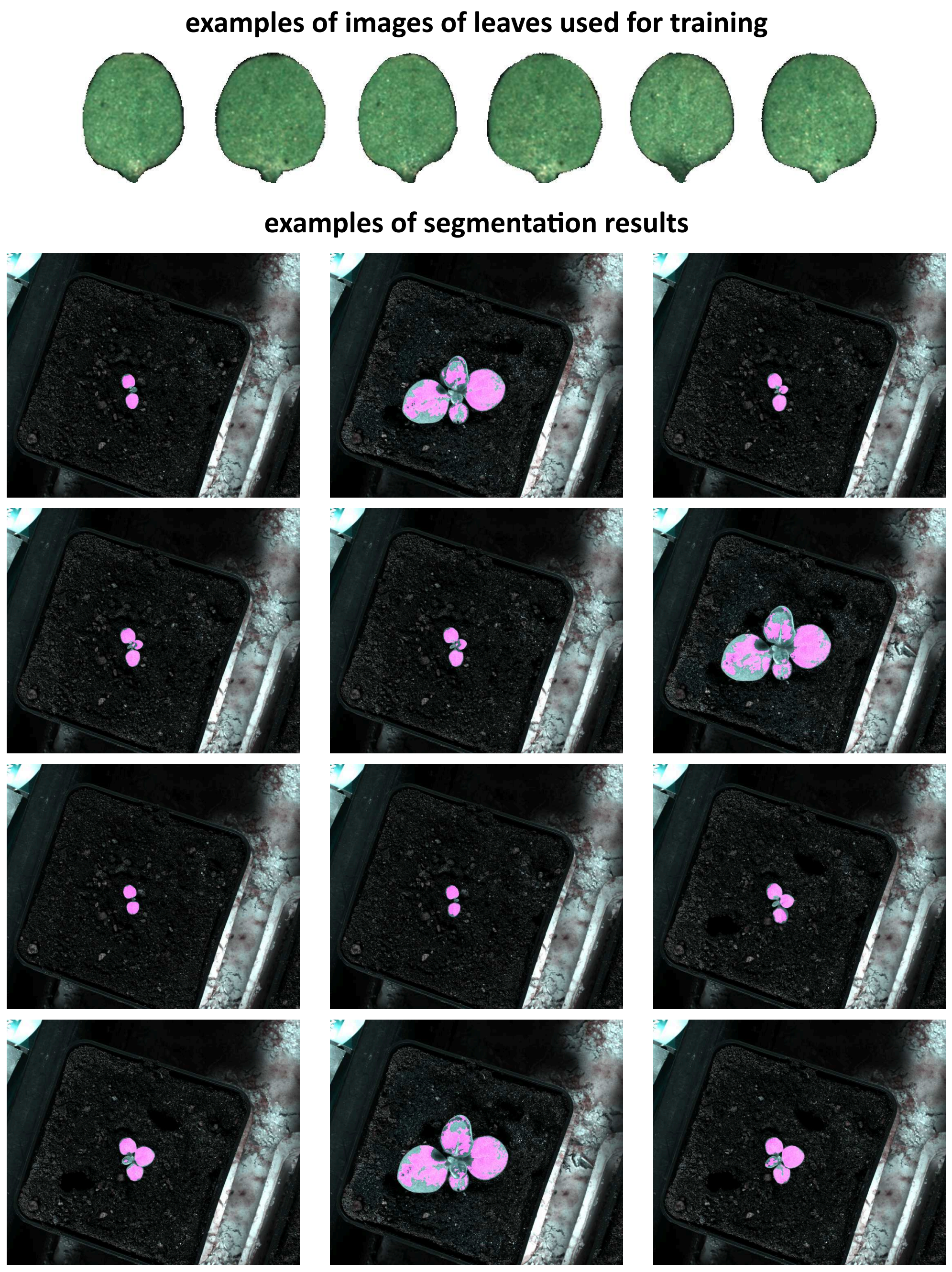

examples of images of leaves used for training
examples of segmentation results

Figure 5.3: Experiment 2: An example of a segmentation setup. The method is trained from images of leaves to locate a plant in images. In the experiment 100 quantization levels are used with 20 channels and $\omega_f = 0.3$ in Equation (5.11).

Method	E_{miss} in %
GrabCut - Simple mixture model [9]	16.3
GrabCut - learned GMMRF [9]	7.9
Nieuwenhuis and Cremers [69]	6.5
our method	5.49
Random Walker [37]	1.1

Table 5.2: Experiment 1: Results presented in the paper by Nieuwenhuis and Cremers.

1. black - definite background
2. blue - training pixels for background
3. green - unknown region
4. white - definitive foreground

As a result the hat on the image is segmented perfectly with perfect fit to the border. The method misses some parts of the ears, so here an additional brush stroke might be applied to improve segmentation. While the method manages to follow the border of the neck part and to successfully distinguish it from the collar, the method includes a part of the shirt in the middle of the image. Examining the associated image and the trimap, we notice that the shirt contains similar colors as the neck. Also, the shirt color distribution is not within the set of training pixels from background. Therefore, the algorithm chooses it to be a foreground region.

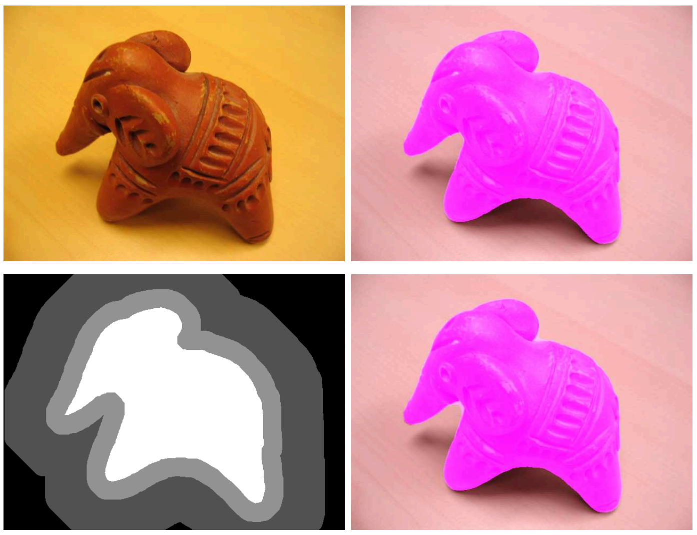
5.5.4 Experiment 4: Evaluating our method on the GrabCut benchmark using bounding-boxes as training data

The advantage of using a bounding box as a prior is that only two coordinates are needed for specifying the rectangular area. We test our method on the GrabCut data-set but using bounding boxes developed for [55]. In their paper, they specify bounding boxes and define margins. In their experiments, the thickness of all margins is $d = 0.06$ of the largest bounding box dimension. Prior knowledge on background is given from the bounding-box margins, while prior knowledge on foreground does not exist. The inside of the bounding box is then segmented.

Segmentation is performed on the whole data-set and the average misclassification error is presented. The average performance is $E_{miss}(\%) = 15.14$. Compared to results obtained in the paper by Lempitsky *et al.* [55], we observe that the segmentation performance is much worse than the one reported for other methods in the paper, Table 5.3. In the table performance of methods that use one-step segmentation is presented. Figures 5.7-5.10 show performance of the method on some images from the GrabCut data-set.

The second setup examines the performance of our iterative segmentation method, using 80 quantization levels and 20 channels. Segmentation is performed with 5 iterations. Prior knowledge is recalculated in each iteration using pixels that are beforehand assigned as foreground pixels. The foreground prior is formed using pixels where $\phi > 0$

a



b

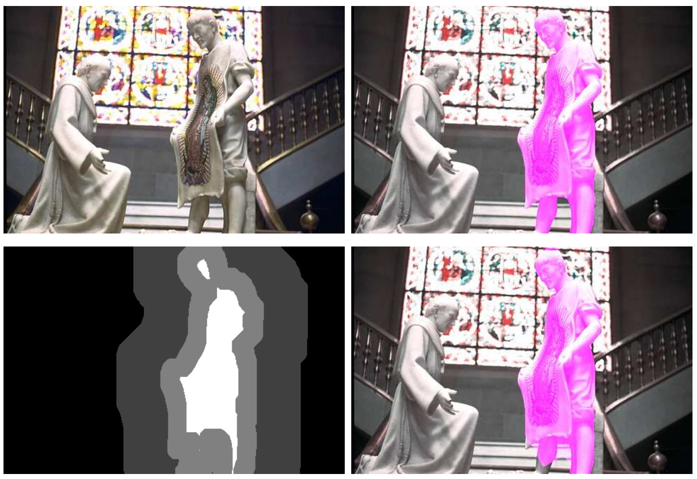
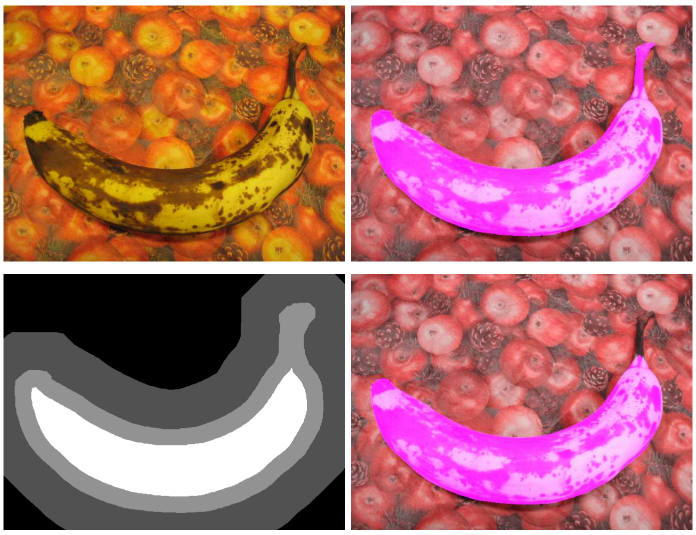


Figure 5.4: Experiment 3: Performance of our method. The first image in a set depicts the original image to be selected, the second image is the ground-truth segmentation, the third image is the trimap that is used for segmentation and the fourth is the segmentation result of our method. **a)** $E_{miss} = 0.9$% **b)** $E_{miss} = 6.02$%

c



d

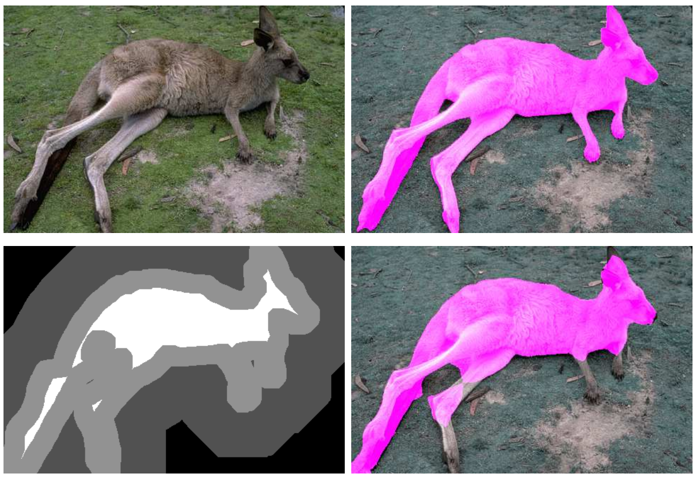


Figure 5.5: Experiment 3: Performance of our method. The first image in a set depicts the original image to be selected, the second image is the ground-truth segmentation, the third image is the trimap that is used for segmentation and the fourth is the segmentation result of our method. **c)** $E_{miss} = 3.54$% **d)** $E_{miss} = 10.28$%

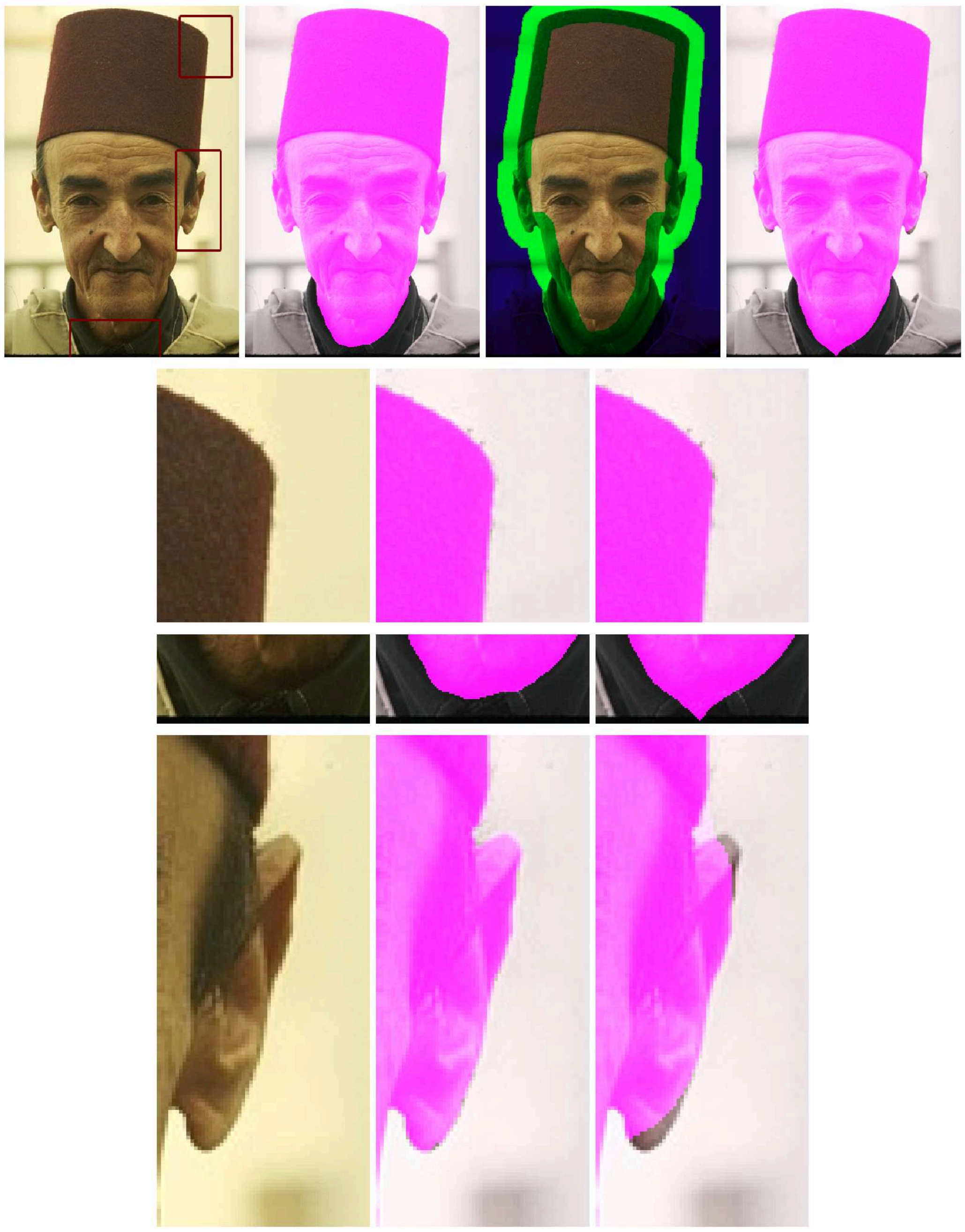


Figure 5.6: Experiment 3: Performance of our method. The first image in the first row depicts the original image to be selected, the second image is the ground-truth segmentation, the third image is the trimap that is used for segmentation and the fourth is the segmentation result of our method. $E_{miss} = 3.44$. The second and the third row depict details of segmentation.

Method	E_{miss} **in %**
our method	15.14
GrabCut [9]	6.7
LP-Threshold [55]	5.4
MinMarginal-Pinpoint [55]	5.4
Unary-Threshold [55]	5.2
LP-Pinpoint [55]	5.0

Table 5.3: Experiment 4: Results of methods that estimate prior knowledge only once, presented in the paper by Lempitsky *et al.* [55]

Method	E_{miss} **in %**
our method	16.09
GrabCut-original [9]	5.9
GrabCut-Pinpoint [55]	3.7

Table 5.4: Experiment 4: Results of methods that use iterative approach presented in the paper by Lempitsky *et al.* [55]

from the previous iteration. For calculating probability distribution functions, no spatial dependence is introduced and PDFs are calculated using Equation (5.12). The rectangle area is cropped and segmentation is performed just on that area using information from training pixels.

The outcome of the experiments are presented in Table 5.4 and Figures 5.11-5.14. The average performance is $E_{miss}(\%) = 16.09$. This table contains results of other methods that re-estimate foreground prior knowledge over iterations. it is compared to other iterative approaches presented in the paper by Lempitsky *et al.* [55].

Following conclusions can be made by investigating segmentation results. For images where margins overlap with a large amount of a foreground region, the background prior contains information about the foreground distribution as well, and results can be undesirable. This is shown in Figure 5.13. Large part of the ground is contained in the margin and is therefore assigned to background.

In case that the region within margins contains parts with a similar color distribution as the foreground object, those parts of the foreground object will be assigned to background. *E.g.* , in Figure 5.14 there is a person whose hair has a similar color distribution as the ground contained in the margins, while these colors are not contained in the clothes. Therefore, the hair is assigned to background.

Our method gave worse results than state-of-the-art methods for the presented problem. Also, there is no improvement when the iterative approach is implemented. We conclude that when relying solely on color distribution as prior, the indication of this prior by the training data needs to be more precise. Indicating bounding boxes alone, is not enough.

a

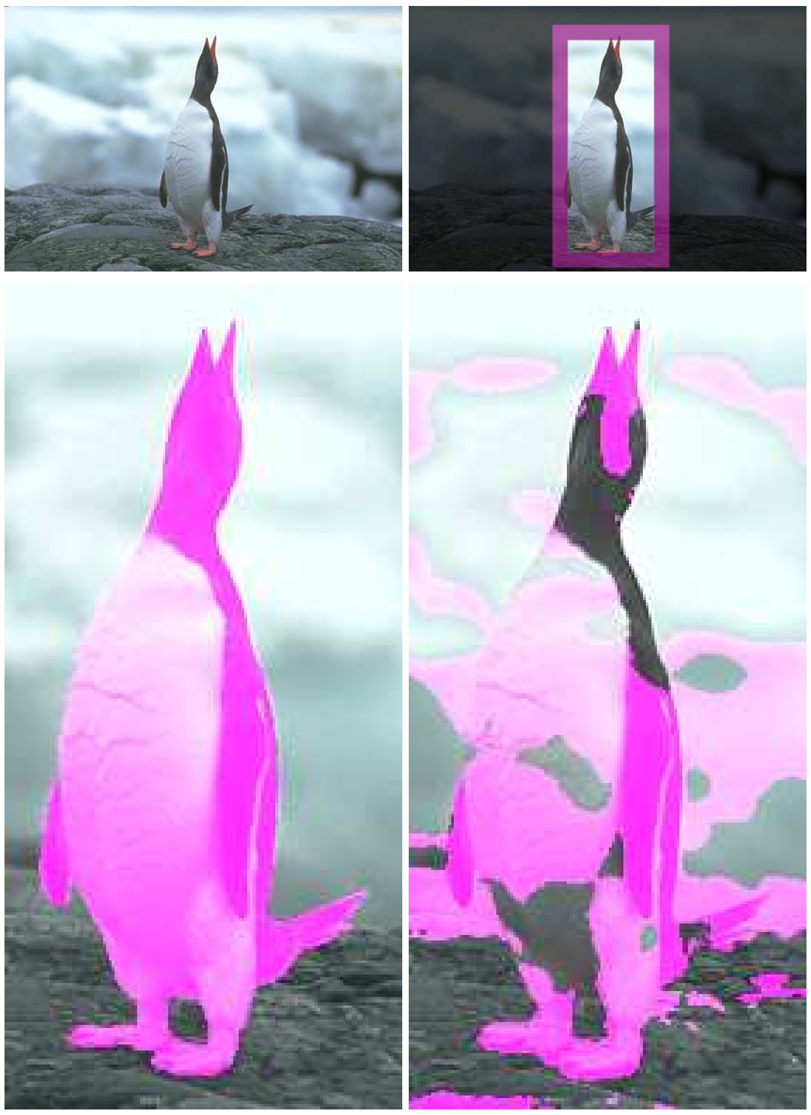


Figure 5.7: Experiment 4: Performance of our method; the one-step segmentation; $E_{miss} = 19.22$%

b

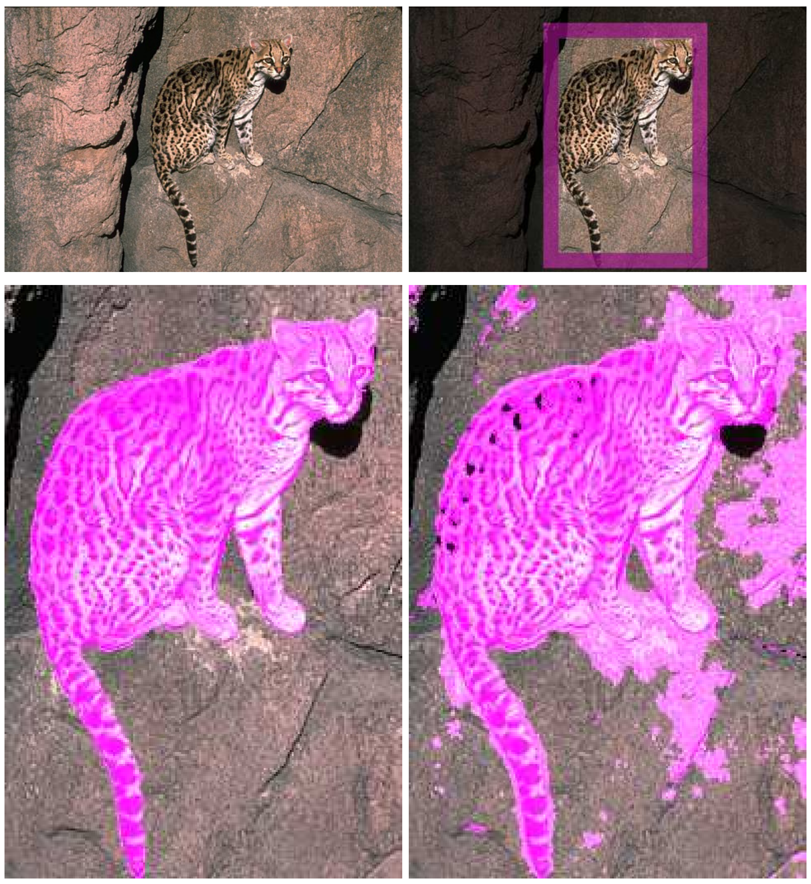


Figure 5.8: Experiment 4: Performance of our method; the one-step segmentation; $E_{miss} = 11.69$%

c

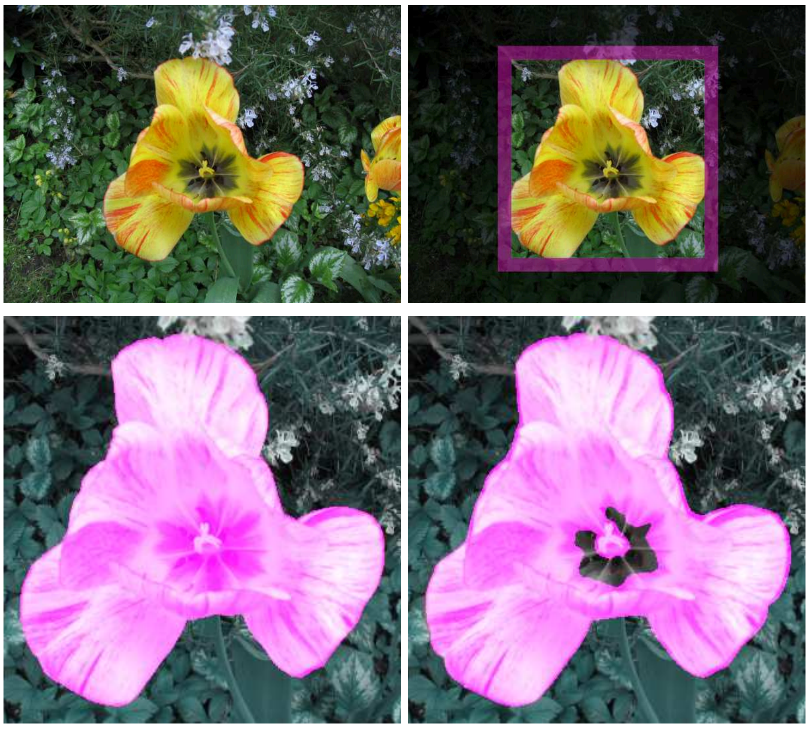


Figure 5.9: Experiment 4: Performance of our method; the one-step segmentation; $E_{miss} = 3.61$%

d

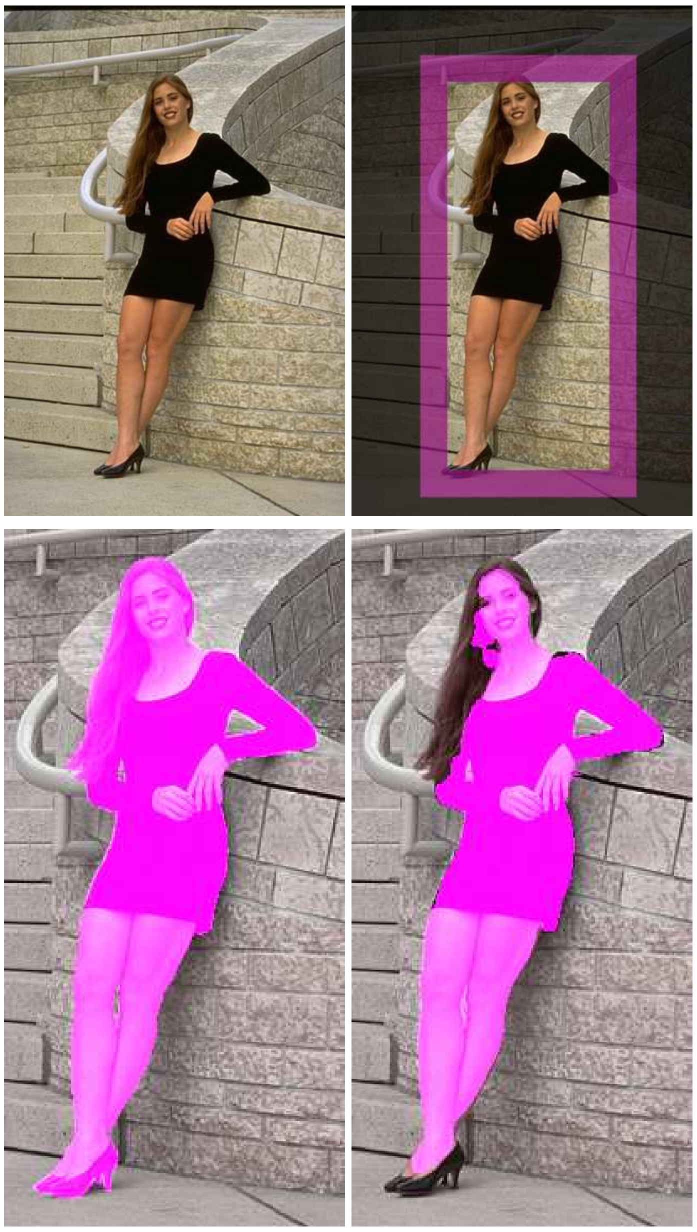


Figure 5.10: Experiment 4: Performance of our method; the one-step segmentation; $E_{miss} = 5.38$%

a

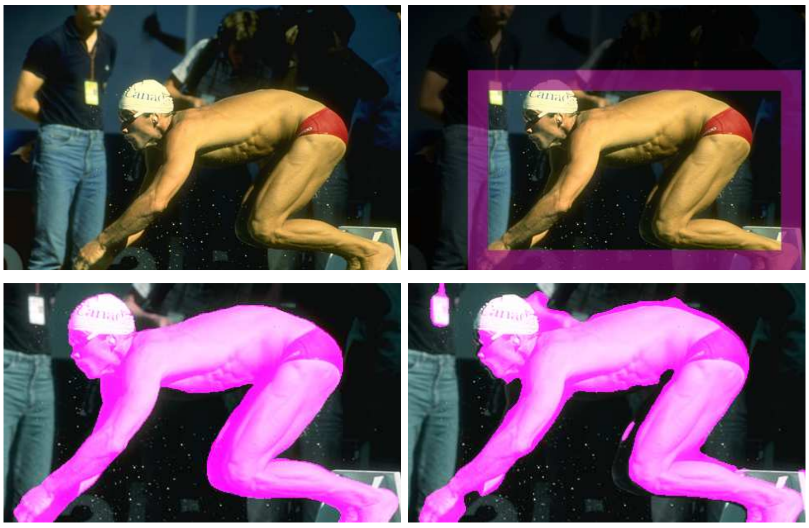


Figure 5.11: Experiment 4: Performance of our method; the iterative approach; $E_{miss} = 9.40\%$

b

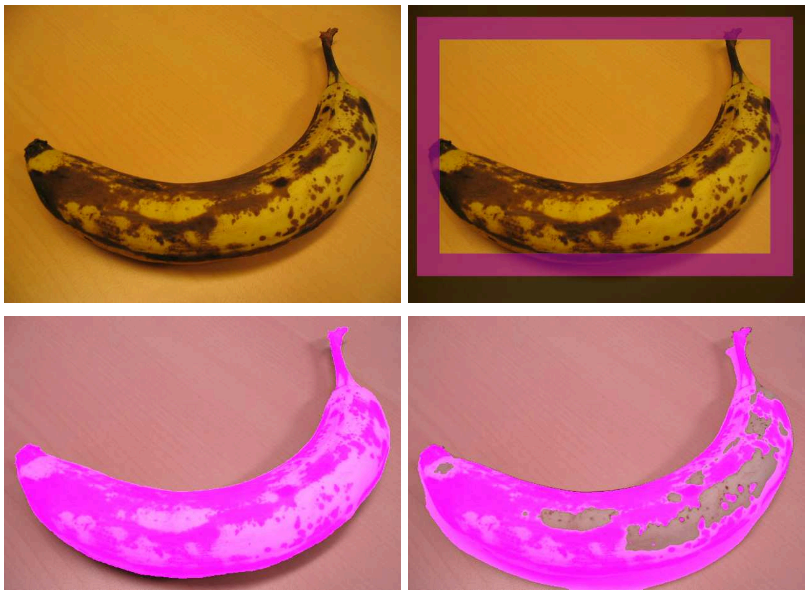


Figure 5.12: Experiment 4: Performance of our method; the iterative approach; $E_{miss} = 11.05\%$

c

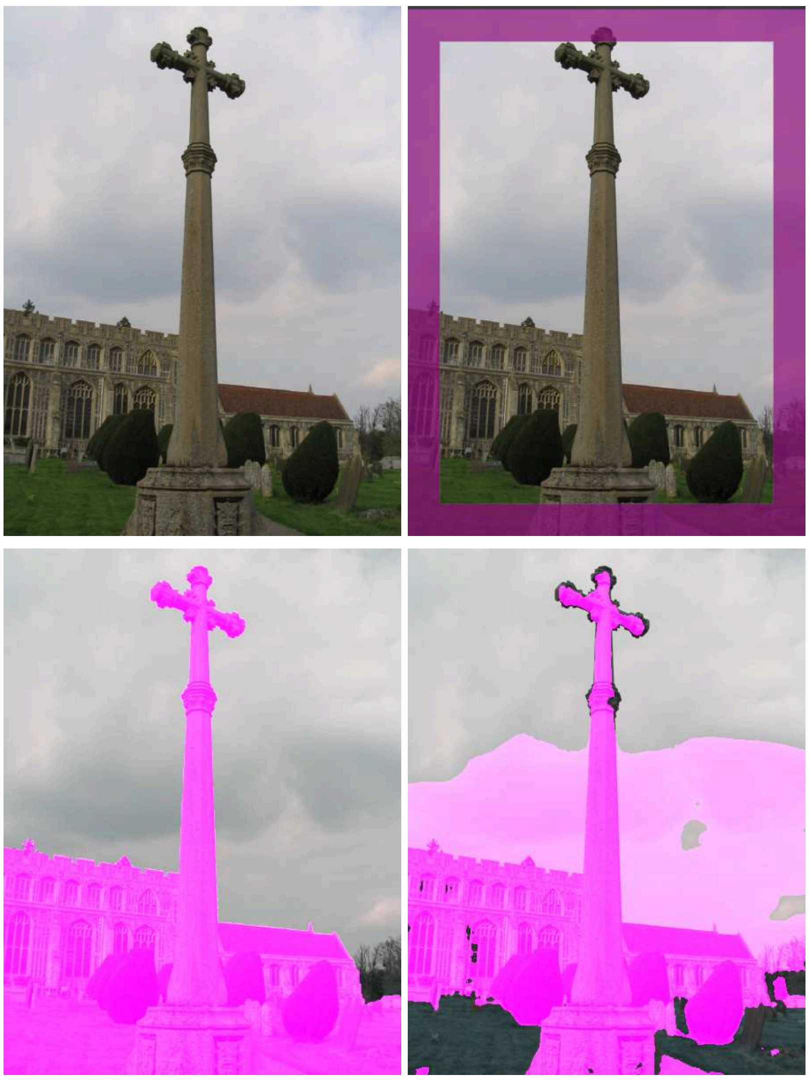


Figure 5.13: Experiment 4: Performance of our method; the iterative approach; $E_{miss} = 38.4\%$

d

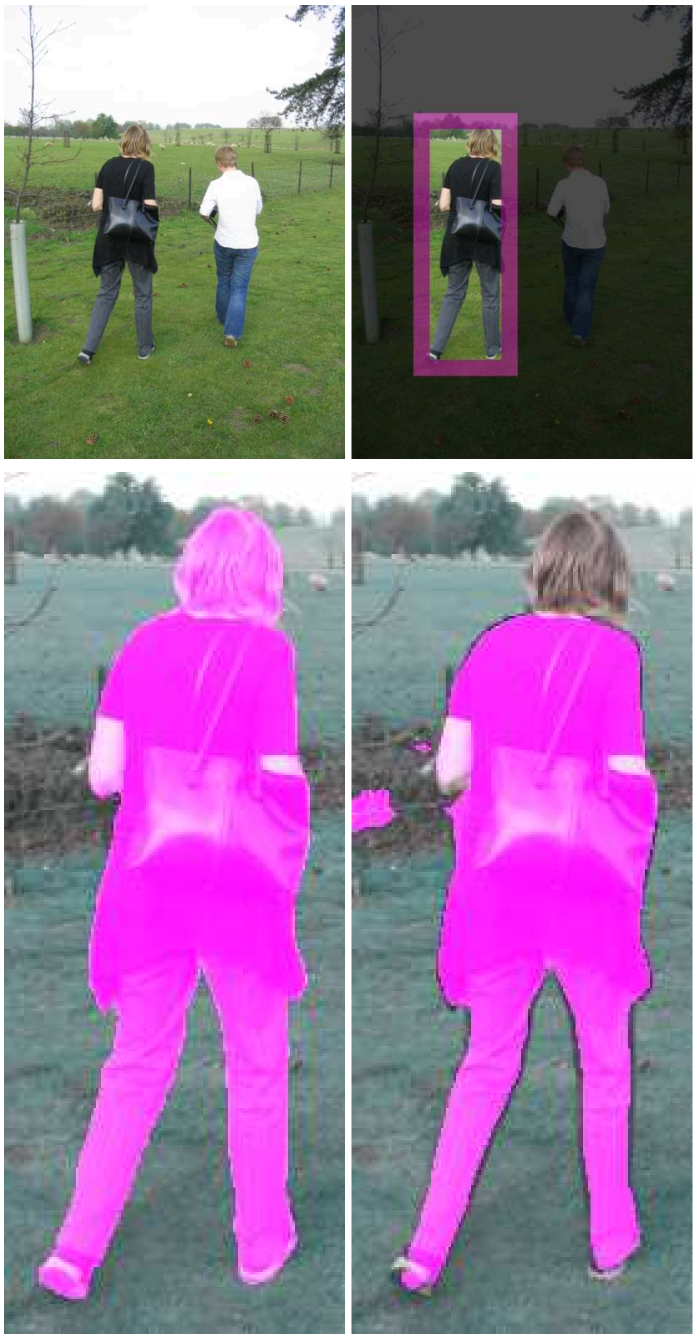


Figure 5.14: Experiment 4: Performance of our method; the iterative approach; $E_{miss} = 8.24\%$

5.6 Summary

In this chapter a supervised segmentation method in two regions was presented. It is also possible to adjust the trade-off between a fully supervised and fully automatic segmentation. For minimization of the functional, the NRI method is used again.

We examine the performance on the tobacco plant data-set presented in Appendix B for foreground/ background segmentation. The task for our method is to extract a plant from background based on prior knowledge gained from a user specified rectangle area. We examine the influence of the regularization parameter ω_f in Equation (5.11) that gives the trade-off between a fully automatic and fully supervised segmentation. As only color distribution is used for forming prior knowledge, leaves cannot be separated on an image. The dice-score is used as a measure of success.

The second experiment demonstrates that it is possible to calculate prior knowledge on different images than those that shall be segmented. We train the method on a leaf collection and perform segmentation on the tobacco plant data-set to separate a plant from background.

We also evaluate our approach on the lasso selection GrabCut data-set. Results are compared to the ones presented in the paper by Nieuwenhuis and Cremers [69] (two-region case). Our method outperforms all tested methods except the Random Walker [37].

The method is also tested against the bounding box selection method presented in Lempitsky *et al.* [55] where it obtains poor results. Therefore, it is concluded that the bounding-box selection does not specify the prior knowledge on regions well enough for our method.

Chapter 6

Interactive multi-region segmentation

6.1 Introduction

The methods described in previous chapters were limited to segmentation of two regions. Images usually contain more than only two objects and therefore, removing background does not provide the desired solution. Further improvement would then go in a direction to increase the number of searched objects. Still, it is difficult to automatically decide on how many different objects are present in images, and to define what should be considered as a whole object, weather small details should be considered as a separate item and stand-alone objects, rather that just parts of a bigger object. Therefore, it is often desired that a user gives a hint on the number.

In this chapter, we investigate multi-label segmentation approaches that contain prior knowledge on targeted regions. The purpose of this chapter is to demonstrate the implementation of the channel framework within a multi-region method.

For multi-label segmentation, instead of the Chan-Vese functional, the Potts model [75] (see Section 2.4.3) is used. Our main contribution is implementing the channel framework to approximate probability distribution functions of regions. PDFs are contained within the cost function of the Potts model. For the minimization of the Potts model, we use the implementation by Yuan *et al.* [99, 97, 98] of *the continuous max-flow algorithm for solving the 2D continuous-cut problem with multiple labels*.

The method is evaluated on the Graz data-set contributed by Santner *et al.* [82]. This data-set is also used for evaluation of the segmentation method presented within the work by Nieuwenhuis and Cremers [69]. Within this data-set, scribbles are provided for calculating prior knowledge on regions. For evaluation, the dice-score is used.

6.2 Related methods

The Potts model has been used in several approaches for multi-label image segmentation, see *e.g.* [93, 80, 99, 69]. Different methods have been proposed for solving it. Nieuwenhuis *et al.* [68] give overviews of minimization methods for the Potts model in discrete and continuous settings.

The Graph-cut algorithm is not preferred for multi-label segmentation problems as it requires solving a higher dimensional graph. Instead, a popular decision for solving the Potts model is minimizing its convex relaxation in the continuous setting. These are described in papers *e.g.* [19, 70, 73, 97, 82, 80, 69]. Within our method, the minimization by Yuan *et al.* [99, 97, 98] is used.

The channel framework is used for describing the data-term within the Potts model. Locations of scribbles are considered as well, PDFs are multiplied with the distance function that gives high probabilities to values close to scribbles, while it gives low probabilities to values farther away.

This is not the first time that channels are introduced to multi-label segmentation. Channels are used together with the Potts model also in works by Wallenberg *et al.* [93]. They fuse information on color distributions with depth map intensity distribution of regions. As this leads to > 3 different variables (R, G, B and the depth information - the depth data and its gradients along axes), the tensor channel representation becomes calculation costly, so instead, they encode all colors in channels and concatenate them to form the vector of channel coefficients. In other words, the RGB vector is interpreted as a channel vector of the spectral density with length $N = 3$ and the color matching function as basis functions. The resolution of the channel vector is then increased applying spatial averaging[29].

This way, they do not exploit correlation between different features which our tensor representation enables. Figures 6.1, 6.2 and 6.3 demonstrate disadvantages of this approach.

Let us consider a synthetic image on Fig. 6.1 **a)**. The image contains the following regions (Fig. 6.1 **b)**):

- Square 1 contains all combinations of colors whose red and blue channel take values in the range $R, B \in [0, 128)$.
- Square 2 contains values where $\{R, B\} = \{15, 230\}$
- Square 3 contains values where $\{R, B\} = \{230, 15\}$
- Square 4 contains all combinations from the range $R, B \in [128, 255]$

The green channel is always 0 and will not be encoded in channels.

If the first and the fourth squares are used for forming the channel density estimate of the foreground, the CDEs depicted on Figure 6.2 are obtained. Fig. 6.2 **a)** shows the representation used by Wallenberg *et al.* [93] and Fig. 6.2 **b)** the tensor channel representation.

If the probability distribution is calculated using Equation (5.4), the vector channel representation will lead to the conclusion that the squares 2 and 3:

- **a) the concatenated channel vector representation:** have the same probability of being a foreground as other two squares;
- **b) the tensor channel representation:** are not the part of the foreground.

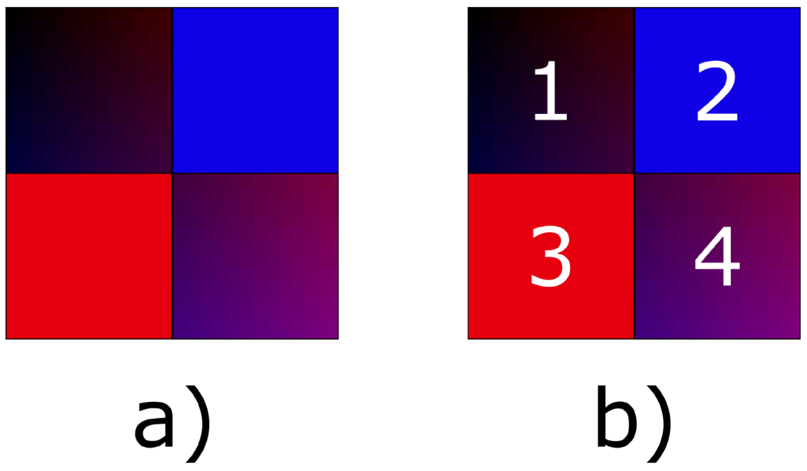


Figure 6.1: **a)** The synthetic image used for comparing the concatenated channel vector representation to channel tensor representation. **b)** description of used values - refer to the text.

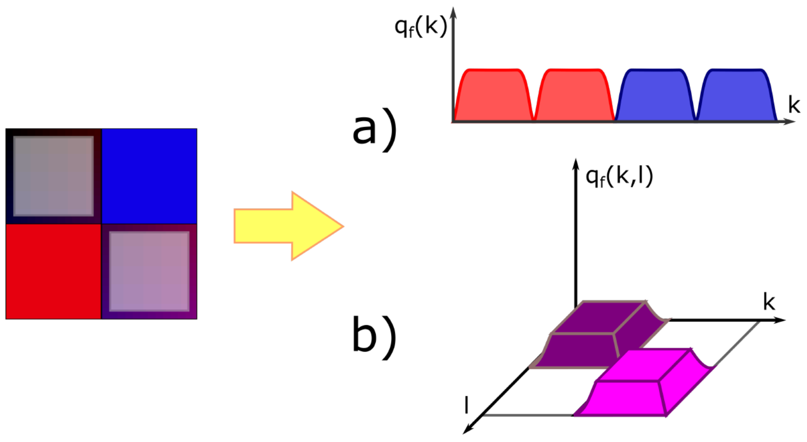


Figure 6.2: Channel density estimates: **a)** the concatenated channel vector representation; **b)** the tensor channel representation.

This is shown on Figure 6.2.

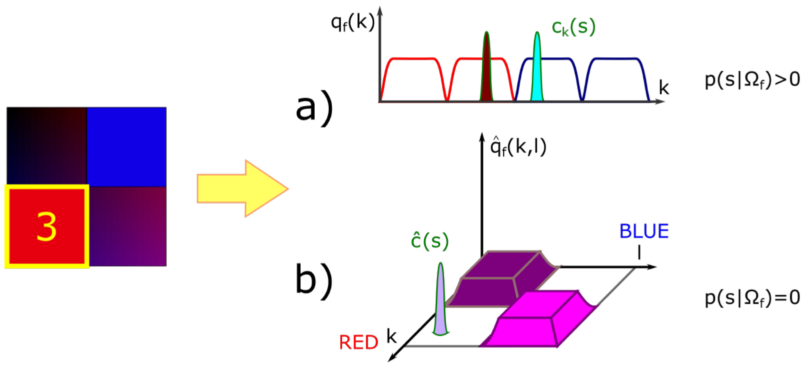


Figure 6.3: Probability of the square 3 to be a foreground part: **a)** the concatenated channel vector representation shows it is foreground; **b)** the tensor channel representation shows it is not foreground.

Our method is compared to the following ones:

- Santner *et al.* test several color models (grayscale, RGB, HSV and CIELAB), as well as, texture descriptors and implement algorithms on GPU. They approximate sampling probability distributions using kernel density estimators. For minimizing the Potts model, they use the minimization method presented in [73]. For segmentation evaluation they use the dice-score measure.
- Nieuwenhuis and Cremers [69] take into consideration locations of scribbles together with color values of marked pixels. They implement PDF estimation and minimization of the convex relaxation of the Potts model on GPU. Gaussian functions are used as kernels for calculating KDEs. They test their method on the Graz benchmark and show high quality results.

6.3 Solving the Potts model

The method minimizes the convex relaxation of the continuous setting of the Potts model, as described in papers by Yuan *et al.* [99, 97, 98]:

$$\min_{\rho \in \mathcal{S}} \sum_{i=1}^{N} \int_{\Omega} \rho_i(x) C_i(x) dx + \sum_{i=1}^{N} \int_{\Omega} \omega(x) |\nabla \rho_i| dx \tag{6.1}$$

where $\rho_i \in [0, 1]$ indicate the membership of the value at position x to a region Ω_i. The first term is the data term. The second term in the equation is the total variation term which measures the perimeter of the corresponding region $|\delta\Omega_i|$.

With this model, the continuous image domain $\Omega \in \mathbb{R}^2$ is partitioned into a set of disjoint regions Ω_i:

$$\bigcup_{i=1}^{N} \Omega_i = \Omega; \quad \Omega_k \cap \Omega_l = \emptyset, \quad \forall k \neq l \tag{6.2}$$

In their original implementation, Yuan *et al.* [99, 97, 98] solve the convex relaxation of the Potts model while labels are defined by different color values. Within the thesis, the following cost function is used:

$$C_i(x) = -\log p(\boldsymbol{g}(x)|\Omega_i) \tag{6.3}$$

Sampling distribution functions $p(\boldsymbol{g}(x)|\Omega_i)$ are approximated using the channel framework. Channel density estimates are calculated using the color distribution obtained from the scribbles.

$$\hat{\boldsymbol{q}}_i = \frac{\int_{\Omega_i^{us}} \hat{\boldsymbol{c}}(g(x,y))d\boldsymbol{x}}{\int_{\Omega_i^{us}} 1 d\boldsymbol{x}} \tag{6.4}$$

Finally, probabilities that pixels belong to a certain region are calculated using channel density estimates. The location of scribbles is also taken into account using the distance function σ_{Ω_i}, which has high values near scribbles belonging to region Ω_i and low values for distant regions.

$$p(\boldsymbol{x}|\Omega_i) = \Big(\sum_{k=1}^{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} \hat{c}(\boldsymbol{x},k,i,j)\hat{q}_i(k,i,j)\Big) \cdot \sigma_{\Omega_i}(\boldsymbol{x}); \quad \boldsymbol{x} \in \Omega \tag{6.5}$$

where $\sigma_{\Omega_i}(\boldsymbol{x})$ is a distance function that gives the dependency to sampling probability distributions of scribble locations:

$$\sigma_{\Omega_i}(x,y) = e^{-\min\limits_{(x_u,y_v)\in\Omega_i^{us}} \gamma\sqrt{(x-x_u)^2+(y-y_v)^2}} \tag{6.6}$$

6.4 Implementation details

Segmentation is performed in the following way:

1. calculate $\hat{\boldsymbol{q}}_i$ from scribbles for each region Ω_i using (5.3)
2. calculate $p(\boldsymbol{g}|\Omega_i)$ for each region Ω_i using (5.4)
3. minimize the energy functional $E(\phi)$, (6.1)

For encoding pixel values into channels, Gaussian basis functions are used:

$$\begin{aligned}
\boldsymbol{c} &= (c_1, c_2, \ldots, c_{N_c});\\
\alpha &= \frac{\Delta_n^2}{2\sigma^2}\left(\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right)^2;\\
c_n(s) &= \begin{cases} e^{-\alpha}, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| < \Psi \\ 0, & \left|\frac{s-s_{min}}{\Delta_s} - \frac{n-1}{\Delta_n}\right| \geq \Psi \end{cases};\\
\Delta_s &= s_{max} - s_{min};\\
\Delta_n &= N_c - 1
\end{aligned} \tag{6.7}$$

N_c	η	**num. of non-zero channel values per pixel value**	**table memory requirements**	**max. size of CDE**
10	50	$5 \times 5 \times 5 = 125$	$125 \times 50 \times 4 = 25KB$	$10 \times 10 \times 10 \times 4 = 4KB$
8	32	$5 \times 5 \times 5 = 125$	$125 \times 32 \times 4 = 16KB$	$8 \times 8 \times 8 \times 4 = 2.04KB$
16	64	$7 \times 7 \times 7 = 343$	$343 \times 64 \times 4 = 87.8KB$	$16 \times 16 \times 16 \times 4 = 16.38KB$

Table 6.1: Memory requirements for the implementation of the channel framework in experiment setups

Channel tensors are formed as:

$$\hat{\boldsymbol{c}}(s^R, s^G, s^B) = [c(s^R) \otimes c(s^G) \otimes c(s^B)] \tag{6.8}$$

The functional (6.1) is finally minimized using software by Yuan *et al.* [99], [97] and [98].

6.5 Experiments

In the experiments, different parameter selections have been used. Table 6.1 shows memory requirements for each parameter choice. For representing channel values, $single$ data-precision is used.

6.5.1 Experiment 1: Evaluating the algorithm on tobacco plant images

In this experiment, we demonstrate that by an addition of scribbles it is possible to separate leaves on an image from the tobacco plants data-set presented in Appendix B.

This can be assumed from the high dice-score values (> 0.8) of performance results in Table 6.2; and visually verified in Figure 6.4, where we present results on 3 images.

Leaves are separated from background without a problem even with strokes of a small radius. Stone areas that are often assigned to foreground in automatic segmentation (*cmp.* Figure 5.2, $\omega_f = 1$) are now correctly marked as background. However, in some cases where leaves overlap, it is possible that a part of one leaf is fused with another leaf. This can be observed in Figure 6.5. The reason behind is that color distribution of the fused part is very similar to the one proposed by the scribble. As no constraint *e.g.* shape prior is introduced, these parts are assigned to the targeted leaf as well.

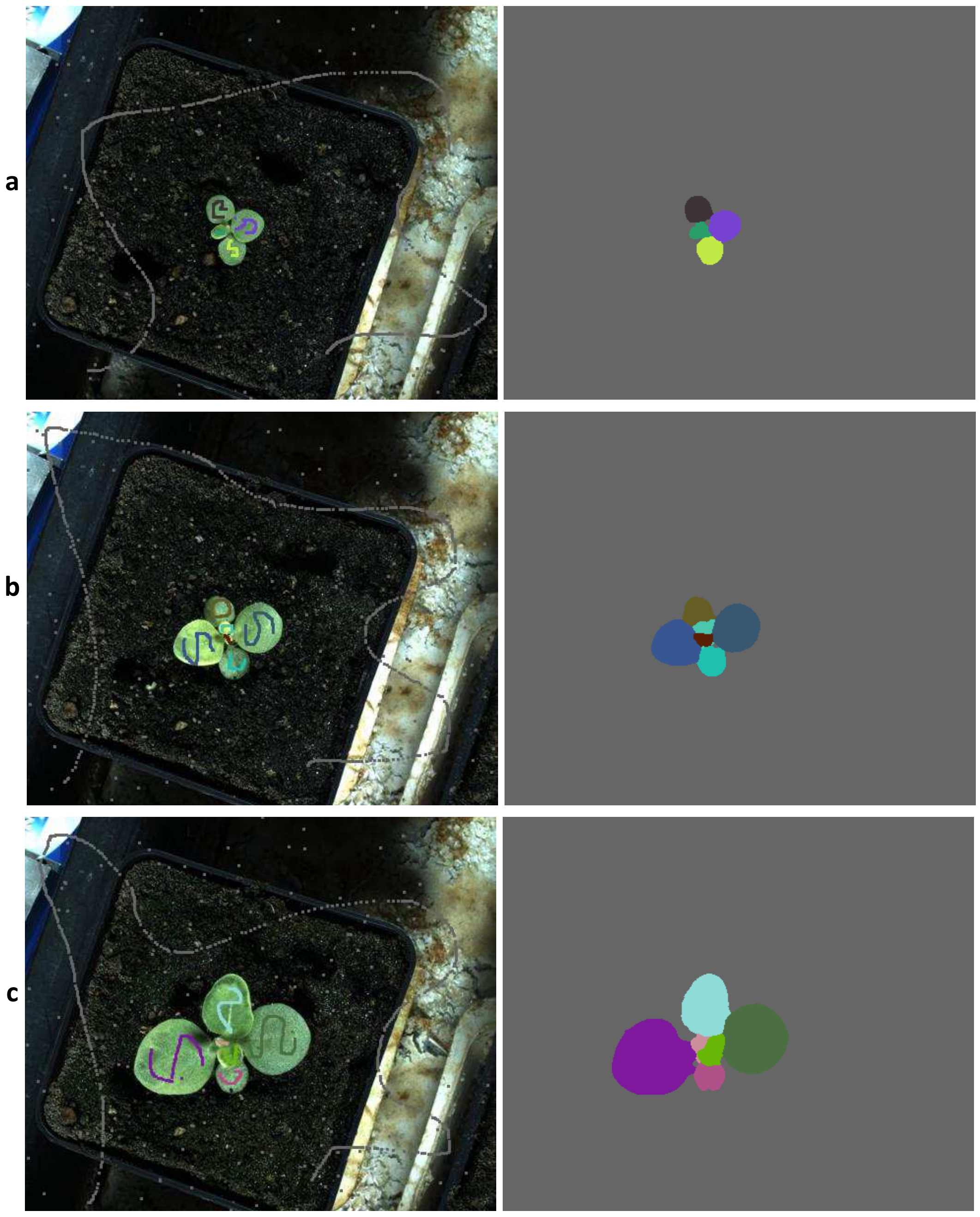


Figure 6.4: Experiment 1: Scribbles are added to images of a tobacco plant. We see that the method is able to distinguish leaves on the plant.

	a	**b**	**c**	**d**
dice-score	0.8933	0.8453	0.8467	0.8463

Table 6.2: Experiment 1: Dice-score performance of our method applied to images in Figures 6.4 and 6.5

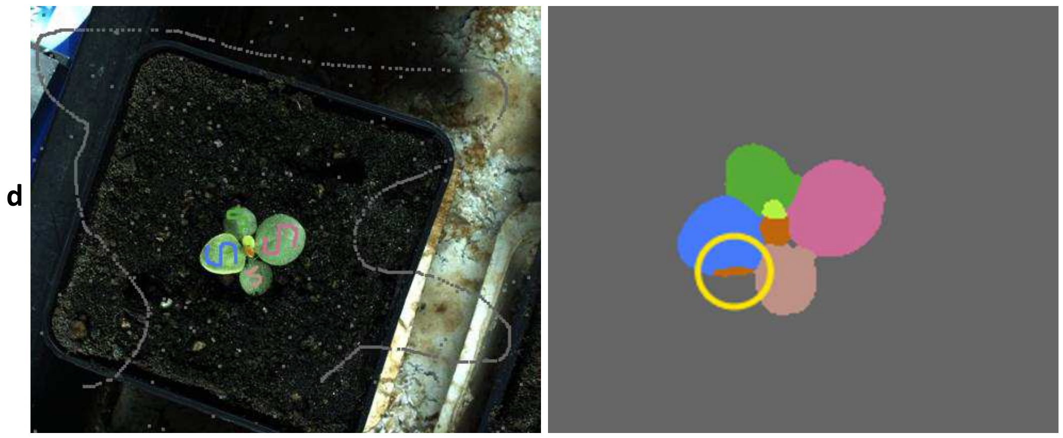


Figure 6.5: Experiment 1: A part of a leaf is assigned to a wrong leaf. This is marked with a yellow circle.

6.5.2 Experiment 2: Evaluating the algorithm on the Graz database

In this experiment, we perform segmentation on the Graz data-set[82] and compare results to those presented in papers by Santner *et al.* [82, 81] and [69]. The data-set contains seeds in addition to the labels. The benchmark consists of 262 seed-ground-truth pairs for 158 images.

Comparison to other segmentation methods can be found in Tables 6.3 and 6.4 and Figures 6.6, 6.7 and 6.8.

Results are compared only to those of methods which use KDEs for describing the data-term. The method is not compared to methods which use additional features, *e.g.* texture information in CIELab + LBP [81].

Our RGB method gives better results than Santner's basic KDE approach which does not introduce the spatial dependency of scribble locations. Our method shows to keep almost same performance even when the size of scribbles is increased from 3 to 13. Our method outperforms the spatially constant method described in the paper by Nieuwenhuis and Cremers [69] if the HSV color space is used, and has similar, but lower dice-score than their spatially variant method.

RGB representation is the color space where each color is defined by how much of the basic color component (in this case red, green and blue) it contains. An important feature of the object, however, is whichever of the basic components in the color prevails. Therefore, it is necessary to find another measure that would describe the order of the R, G, B values and which is actually the color tone of the object. And this is exactly what hue measure does. Hue describes the position on the circular wheel that contains the pure colors. The hue rotates through red, yellow, green, cyan, blue, magenta and back to red. The values between these are mixtures of neighboring pure colors.

Another important feature of an object is its brightness. The maximum value of the RGB triplet gives us the brightness of the color, or the value component in the HSV representation.

Saturation in the HSV triplet is the difference between the highest and the lowest value of the RGB triplet. Saturation is the purity of the color, or how much of the gray component the color has. Gray values can be formed as mixtures of equal amount of all of the pure colors. The zero saturation corresponds to gray values, while the maximum saturation corresponds to pure colors - no white component

Eq. (6.9) shows the transformation between the RGB and HSV representation. For a further description of color spaces consult *e.g.* [1].

$$\begin{aligned} &R, G, B \in [0,1];\\ &\quad V = \mathit{max}\{R,G,B\};\\ &\quad S = V - \mathit{min}\{R,G,B\};\\ &\quad H = \begin{cases} 0, & R = G = B = 0;\\ 60^\circ \cdot \left(\frac{(G-B)}{S}\right), & \mathit{max}\{R,G,B\} = R;\\ 60^\circ \cdot \left(2 + \frac{(B-R)}{S}\right), & \mathit{max}\{R,G,B\} = G;\\ 60^\circ \cdot \left(4 + \frac{(R-G)}{S}\right), & \mathit{max}\{R,G,B\} = B; \end{cases};\\ &\quad H \in [0,360);\\ &\quad S, V \in [0,1] \end{aligned} \tag{6.9}$$

In our method, only linear transformations are used and therefore it is not possible to achieve the same quality of results with the RGB as well as HSV representation. Thus, it is needed beforehand to convert the RGB representation to the HSV representation. The HSV space shows in experiments as well to lead to better segmentation results.

Next, we examine images to see why some images show poor segmentation results. We observe marked parts in Figure 6.8 **a**. All objects on the image have similar color distribution as background which is constructed of small stones, ground, grass and bushes.

In this image the method has assigned only one part of the branch as a foreground, while the other part, which seemed separate and was not marked with scribbles is recognized as a background. Therefore, in this case, the scribble definition is not sufficient and additional strokes on the other part would have helped the method to approach the ground-truth segmentation.

In Figure 6.8 **d**, both the foreground and the background are constructed of gray values. The scribble that describes the background contains only the bright gray values and therefore the dark gray values are missing in the training data of the background PDF. Also, for the spacial dependency, the background scribble just defines that the middle of the picture. On the other hand, foreground is defined with both bright and dark values.

Method	**Brush**	**Benchmark Score**
Random walker [37]	13	0.855
Santner *et al.* [81], RGB	–	0.877
our approach - RGB, $N_c = 10$,$\eta = 50$	3	0.8860
our approach - RGB, $N_c = 10$,$\eta = 50$	13	0.8865
our approach - CIELab, $N_c = 16$,$\eta = 64$	13	0.8879
Nieuwenhuis and Cremers [69], spatially constant	3	0.889
our approach - HSV, $N_c = 8$,$\eta = 32$	5	0.9004
our approach - HSV, $N_c = 16$,$\eta = 64$	13	0.9071
[69], spatially variant	5	0.922
[69], spatially variant	13	0.931

Table 6.3: Experiment 2: Comparison of our method to methods tested in [69]

Figure	**Image**	**dice-score**
Fig. 6.6	**a**	0.9911
	b	0.9906
	c	0.9938
	d	0.9802
Fig. 6.7	**a**	0.9729
	b	0.9669
	c	0.9650
	d	0.9434
Fig. 6.8	**a**	0.8331
	b	0.8478
	c	0.7239
	d	0.7449

Table 6.4: Experiment 2: Performance of our method on the Graz data-set [69]

As a result, dark parts close to borders of the image are recognized as foreground. To improve the segmentation, additional background scribbles should be put closer to image borders.

In the method, the size of the brush stroke is not unlimited. One should pay attention that the brush stroke stays within the object. Larger brushes cause more data to be added to prior knowledge. While scribbles are concentrated within the targeted object, segmentation results might not change as prior knowledge might not change a lot if the object parts have the same color distribution. With an increased radius of the stroke brush, we observe that scribbles can go over the borderline and catch parts of another object. As the prior knowledge in this case contains also parts of another object, segmentation performance decays. An example of this case is given in Figure 6.9.

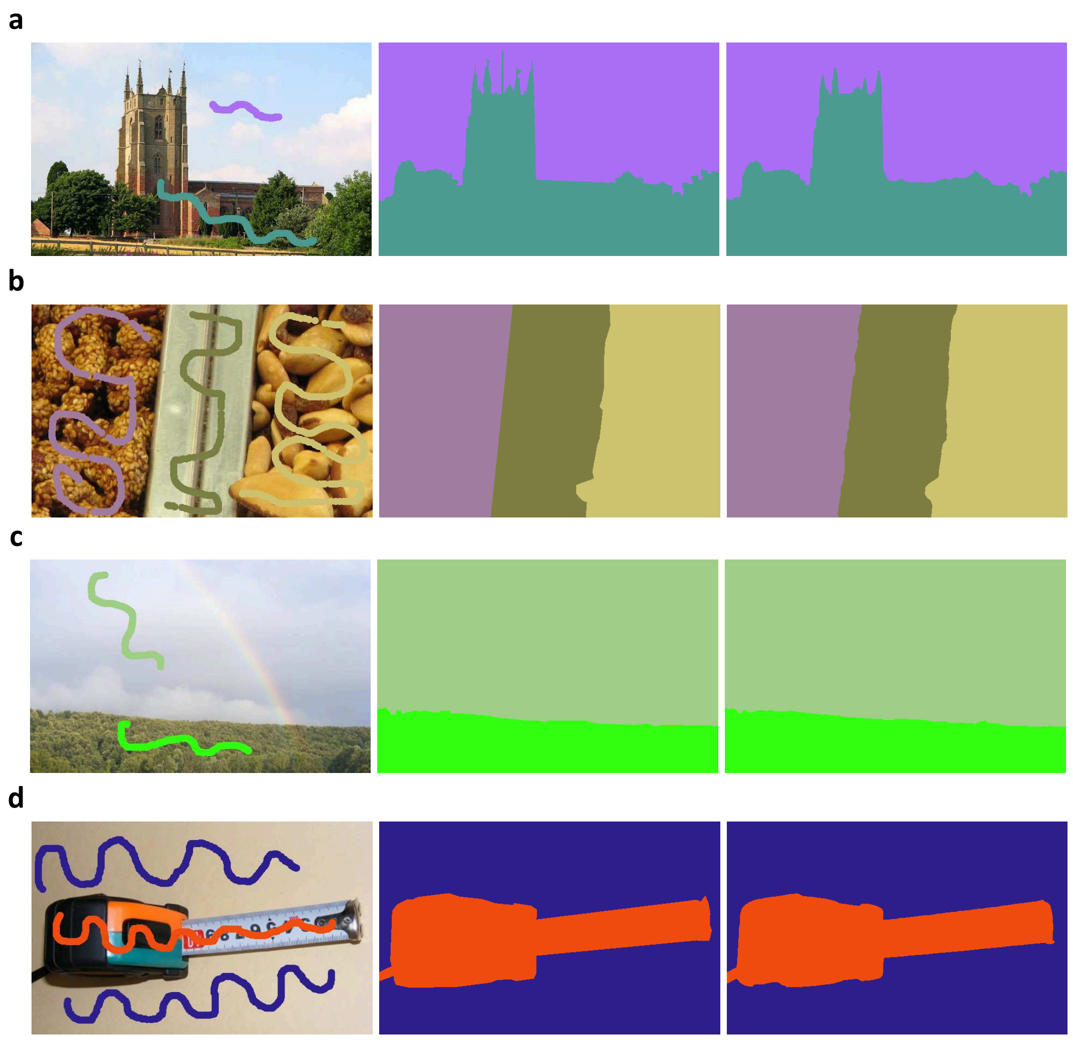


Figure 6.6: Experiment 2: Images where our method performed the best in terms of dice score measure: the image with scribbles, the ground-truth segmentation, results obtained using our method. These images show that using our method on simple images with only two to three distinguishable regions, it is possible to approach really close to the segmentation provided by human.

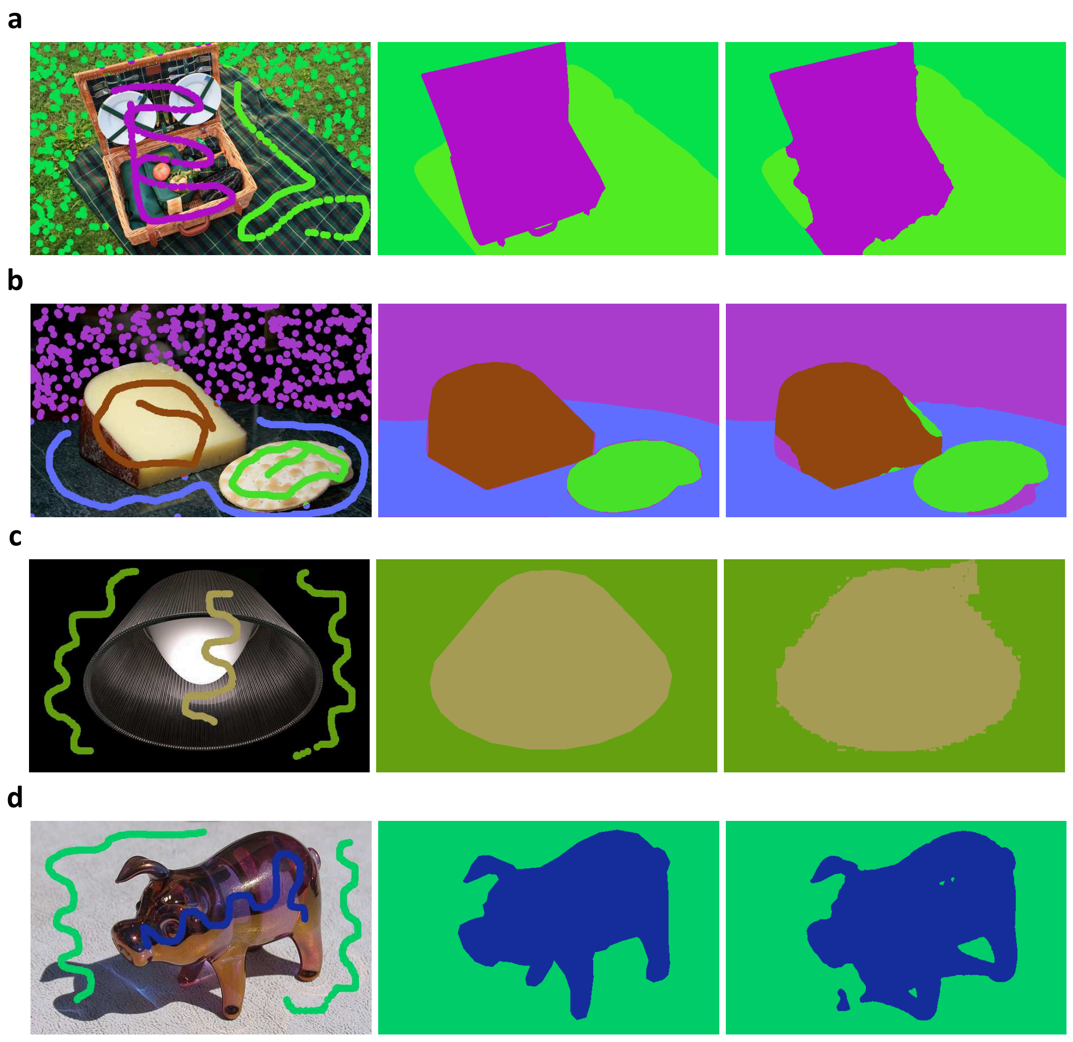


Figure 6.7: Experiment 2: Images where our algorithm performed good: the image with scribbles, the ground-truth segmentation, results obtained using our method. As on previous images, our method tends to give similar results to those provided by human, but some parts of objects, mostly those in shadows are assigned to wrong objects.

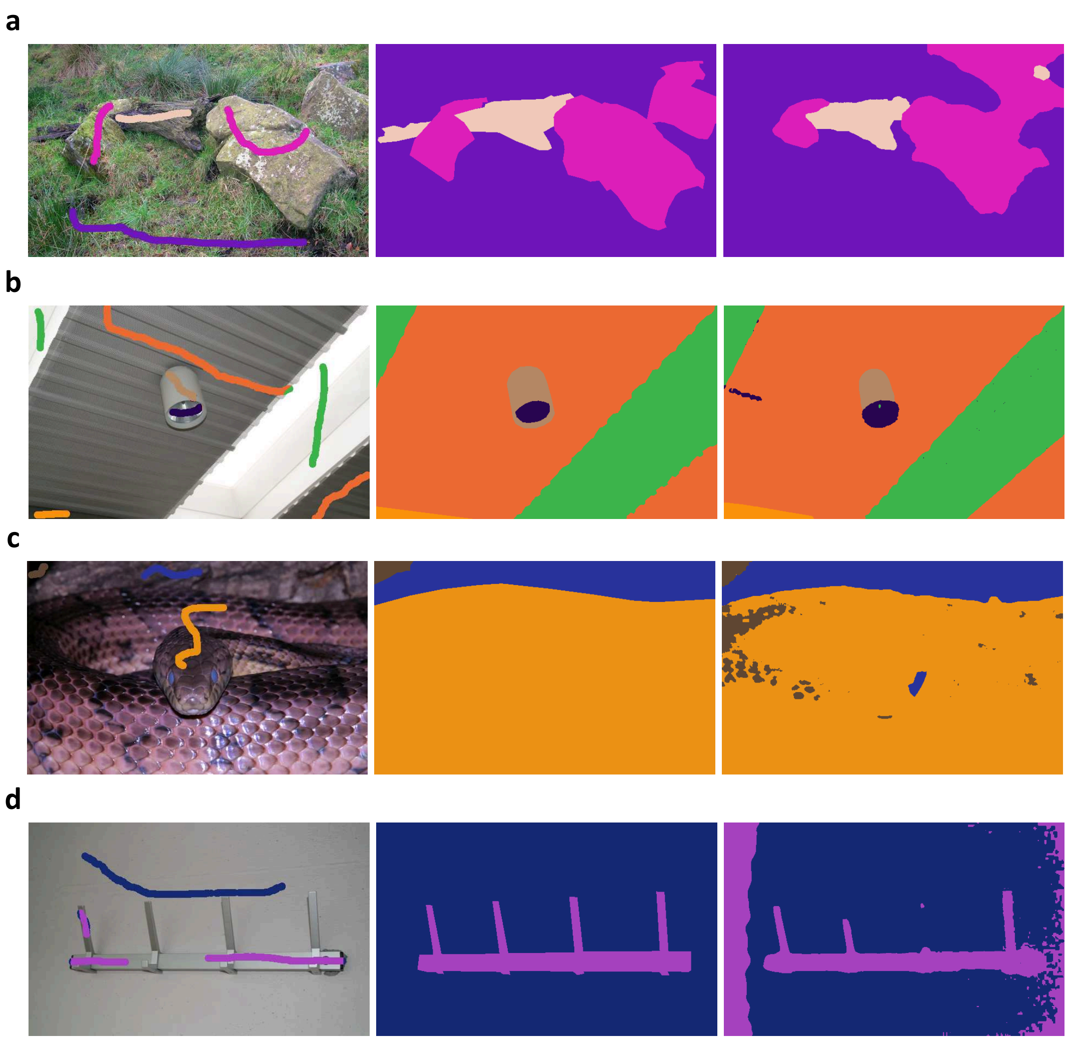


Figure 6.8: Experiment 2: Images where our algorithm gave the poorest dice score measure: the image with scribbles, the ground-truth segmentation, results obtained using our method. On these images, color distribution is not sufficient to separate the targeted objects and additional features should be included.

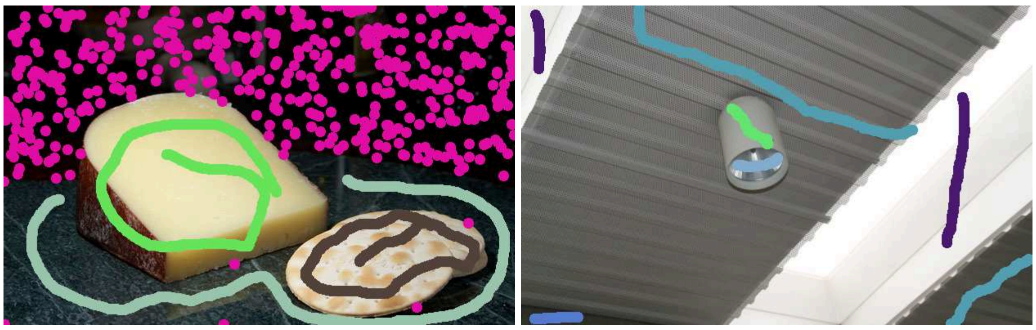


Figure 6.9: Experiment 2: Increasing scribble size may result in scribbles spreading to other objects. This means that parts of other objects are used for calculating prior knowledge as well. This may produce undesired segmentation results.

6.6 Summary

In this chapter, results are reported for our image segmentation approach using channel-based probability formulations for multi-label segmentation. Our contribution is introducing the channel framework to solve an interactive multi-label segmentation problem. The channel framework is already used with Potts model for multi-label segmentation in the work by Wallenberg *et al.* [93], but with a different setup.

Performance of the method is examined on tobacco plant images from the data-set given in Appendix B but with scribbles added. While it is now possible to differentiate leaves from background and from one another, parts of leaves are often merged to other leaves. This is not surprising, as color distributions of different leaves are quite similar. To avoid this problem, additional constraints should be introduced.

Our method is also evaluated on the Graz data-set and compared to methods presented in papers by Santner *et al.* [82, 80], Nieuwenhuis and Cremers [69]. While our method does not surpass performance of state-of-the-art methods, it outperforms both the basic Santner *et al.* [82] approach and the spatially constant approach by Nieuwenhuis and Cremers [69]. At the same time it achieves comparable dice-score performance to the spatially variant version of Nieuwenhuis and Cremers' [69] approach.

The multi-label segmentation problem is studied only preliminary. This topic, however, deserves much more research to be performed. The PDF estimation is performed using only MATLAB implementation. The channels offer the same complexity advantages to KDE methods as in previous chapters and offer an ideal setup to be able to parallelise the PDF calculation. However, parallel implementation of the method and implementation on the GPU are left for future work.

Chapter 7

A stochastic approach to reconstruct a smooth image and its edge-map

7.1 Introduction

In this chapter, we present another possible direction in future development of image segmentation algorithms. The methods presented previously in this thesis can be used for coarse detection of an object. Then, the result can be used as a defined region of interest for the shape fitting method. A shape prior, as introduced subsequently, can then be used to refine the detection of the object.

Approaches of previous chapters divide an image into a number of regions according to their gray-value or color distribution. Usually, results differ from how a human would segment the same image, and a way to make it more similar is to include additional object features to be recognized. Humans can trace a shape of a targeted object and having this in mind, methods have been developed targeting consideration of shape priors.

In practice, edge-detection is performed using various filters (a description of filters can be found in *e.g.* [83]) on an image (Gradient, Laplace, Sobel[87]) and as a result give

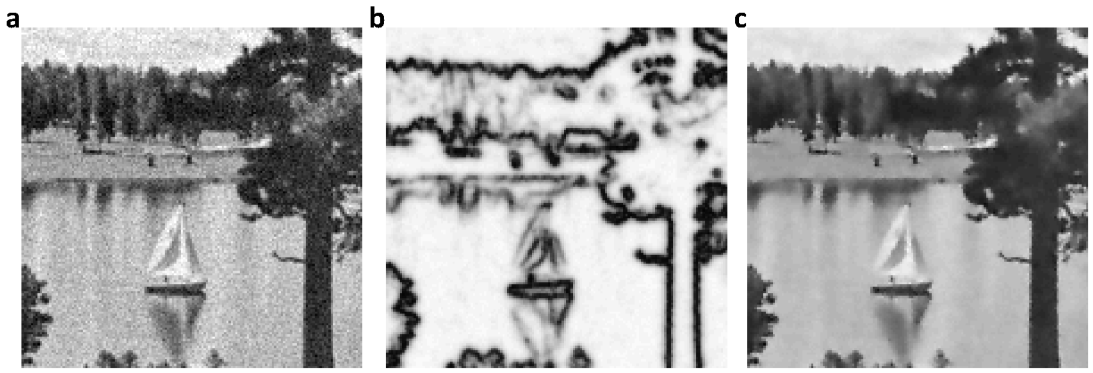


Figure 7.1: The goal of the Ambrosio-Tortorelli segmentation is to estimate the edge-map on the image and give its cartoon/smooth reconstruction. **a** original images degraded with a Gaussian noise; **b** estimated edge maps; **c** smoothed images.

often a binary image (Canny[18]).

In this chapter another approach to edge-detection of a gray-scale image is given[1]. Here, the edge-map is obtained by reaching a minimum of an energy functional. While it requires more time to process images, its parameters and results are more intuitive and better understood than using the filter approaches. *E.g.* parameters of our approach are the thickness of an edge and its length, while a result is a gray-scale image whose intensity shows us the probability that there is an edge at some location. Figure 7.1 shows a result of our method. Therefore, it is worth examining this method as well, as future research will lead to possibility to perform calculations with a high speed and therefore allow this method to compete with the previously mentioned ones.

Our method presented in this chapter is based on an another modification of the Mumford-Shah functional, the Ambrosio-Tortorelli (AT) functional [5] which is introduced in Section 2.4.4. Most state-of-the-art methods considering the AT functional minimize it using the gradient descent approach and result in finding only the local minimum. Some of these approaches, like the original paper [5], deliver a binary image as an edge-map result and do not offer any confidence measure of the edge recognition.

We offer an approach that is independent on initialization and as a result gives a confidence of edge-location. Our approach is based on an estimation theoretical point of view of the problem. The AT functional is interpreted as the energy of a posterior probability density function. The smooth reconstruction u of an image g and the edge-map reconstruction v are treated as random variables and their true values are obtained applying different estimators, *the minimum mean square* and *the minimum median estimator* on their samples. Samples are obtained using the block-Gibbs-sampler. This approach is similar to the work by Geman-Geman [34], with a difference that our approach uses the block-Gibbs sampler instead of the pixel-wise Gibbs sampler. This modification provides the increase of the speed of the method as the block-Gibbs sampler converges quicker to the targeted distribution.

7.2 Related work

The method presented in this chapter is similar to the work by Geman-Geman [34] which was the first paper to use pixel-wise Gibbs sampling in image processing. In the paper by Rue and Held *et al.* [79], the block Gibbs sampler of Gaussian Markov random fields is introduced which converges faster to the target distribution than the pixel-wise sampler. The reason for this is that the block Gibbs approach samples whole vectors at once. While Rue and Held concentrate on numerical implementation of the sampler, in this thesis, the block Gibbs sampler has been introduced to an image smoothing and an edge detection method.

Other important related methods are those by Pätz and Preusser [72] and Pock and Cremers [74]. In this chapter, we compare results of our experiments to those methods.

[1]This work has been published in papers by Kai Krajsek, Ines Dedović and Hanno Scharr [51] and [52] where the mathematical derivation and implementation were contributed by the author of this thesis. Note that experiments shown in this chapter as well as an amount of the text are exactly the same as in the related publications.

Pätz and Preusser [72] use the AT functional with stochastic images. They aim to give a confidence of edge position by modeling distributions, but they solve the functional implementing the gradient descent approach to Euler-Lagrange equations.

Pock and Cremers [74] apply a fast primal-dual algorithm to compute the solution of a convex relaxation of the MS functional. They compare results of their method to results obtained using the AT functional. Finally, they demonstrate the advantage of their method to the AT method in terms of avoiding artifacts and offering a method that is independent on initialization. We perform the same experiments with our method in order to show that our approach is not prone to get stuck in local minima and can avoid artifacts (where a smooth reconstruction image contains noise close to edges) as well, with an appropriate choice of parameters. In addition to an estimation result, we are able to give information on distribution of edges, *e.g.* error variances of the edge-map.

7.3 The AT energy in a discrete setting

The AT estimation problem, Equation (2.16), is formulated on a regular discrete image domain Ω_h with grid size h.

$$E_{AT}(\boldsymbol{u},\boldsymbol{v}) = \sum_{i=1}^{N_p} \beta\,(u_i - g_i)^2 + \left(\alpha\,(1-v_i)^2\,|\,(\nabla u)_i\,|^2\right) + \left(\frac{\rho}{2}|(\nabla v)_i|^2 + \frac{v_i^2}{2\rho}\right) \tag{7.1}$$

Here, $N_p \in \mathbb{N}$ denotes the total number of pixels. Discrete images can be represented as column vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g} \in \mathbb{R}^{1\times N_p}$, respectively. Gradients ∇u, ∇v are approximated by finite difference operators that can be described by matrices acting on the column vectors. The central difference is avoided as such approximation leads to checkerboard artifacts as shown in [84]. Therefore, gradients are approximated by forward and backward differences and taking their average:

$$\begin{aligned} \sum_{i=1}^{N_p} |(\nabla u)_i|^2 &= \boldsymbol{u}^T \boldsymbol{D}(\boldsymbol{I})\boldsymbol{u}; \\ \sum_{i=1}^{N_p} |(\nabla v)_i|^2 &= \boldsymbol{v}^T \boldsymbol{D}(\boldsymbol{I})\boldsymbol{v}; \\ \boldsymbol{D}(\boldsymbol{I}) &= \frac{1}{2}\sum_{j=1}^{2} \boldsymbol{B}_{x_j^+}^T \boldsymbol{I} \boldsymbol{B}_{x_j^+} + \boldsymbol{B}_{x_j^-}^T \boldsymbol{I} \boldsymbol{B}_{x_j^-} \end{aligned} \tag{7.2}$$

where $\boldsymbol{B}_{x_j^+}$ and $\boldsymbol{B}_{x_j^-}$ denote matrices whereby multiplication gives forward and backward first order finite differences, respectively, along the dimension x_j.

Images are interpreted as random vectors, *i.e.* each element of the vector is a realization of a random variable. The task is to estimate the image $\boldsymbol{u}$ as well as the edge indicator $\boldsymbol{v}$ from *e.g.* noisy or incomplete observation $\boldsymbol{g}$. AT functional (7.1) is considered to be the energy of the posterior PDF $p(\boldsymbol{u},\boldsymbol{v}|\boldsymbol{g})$ of image $\boldsymbol{u}$ and edge indicator function $\boldsymbol{v}$,

$$p(\boldsymbol{u},\boldsymbol{v}|\boldsymbol{g}) \propto e^{-E_{AT}(\boldsymbol{u},\boldsymbol{v})}. \tag{7.3}$$

True values of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are then estimated from the posterior distribution function (PDF) $p(\boldsymbol{u}, \boldsymbol{v}|\boldsymbol{g})$, using the estimators described in the following section.

7.4 Bayesian Estimation Theory

An overview of Bayesian estimation theory can be found in [47], Chapter 11. Let $\boldsymbol{z}$ be a variable whose true value we need to estimate. Let $\boldsymbol{\varepsilon} = \hat{\boldsymbol{z}} - \boldsymbol{z}$ denote the error between the estimated vector $\hat{\boldsymbol{z}}$ and the particular true realization $\boldsymbol{z}$ of the target vector.

To estimate the true value, we define a loss function $L : \mathbb{R}^{2N_p} \rightarrow \mathbb{R}^{+}$. There are different loss functions, and the choice of a loss function leads to different estimators. The expectation of a loss function L with respect to the posterior PDF $p(\boldsymbol{z}|\boldsymbol{g})$ is called a *risk* $\mathbf{R}(\hat{\boldsymbol{z}}) = \mathbb{E}[L]$. The Bayesian estimator gives the true value of a variable by minimizing its risk function.

$$\hat{\boldsymbol{z}} = \arg_{\tilde{\boldsymbol{z}}} \min \mathbf{R}(\tilde{\boldsymbol{z}}) \tag{7.4}$$

In the following text several estimators are presented:

The Minimum Mean Squared Error Estimator (MMSEE) is obtained using the quadratic loss function.

$$\begin{aligned} L(\boldsymbol{\varepsilon}) &= |\boldsymbol{\varepsilon}|^2 \\ \hat{\boldsymbol{z}} &= \int \boldsymbol{z} p(\boldsymbol{z}|\boldsymbol{g}) d\boldsymbol{z} \end{aligned} \tag{7.5}$$

The Minimum Median Error Estimator (MMEE) is obtained using the absolute error loss function.

$$\begin{aligned} L(\boldsymbol{\varepsilon}) &= \sum_{j=1}^{2N_p} |\varepsilon_j| \\ \hat{z}_j &= \text{median}\big(p(z_j|\boldsymbol{g})\big) \\ &= \text{median} \int p(\boldsymbol{z}|\boldsymbol{g}) d\boldsymbol{z}_{\dashv j} \end{aligned} \tag{7.6}$$

where $\boldsymbol{z}_{\dashv j}$ denotes the random vector without element z_j, i.e. $p(z_j|\boldsymbol{g})$ is marginalization of $p(\boldsymbol{z}|\boldsymbol{g})$ with respect to all dimensions but j.

It is either impossible to analytically solve these integrals, or they require complex calculations. To calculate them, it is necessary to use the Markov Chain Monte Carlo integration.

7.5 Introduction to the Monte Carlo integration

An introduction to Markov Chain Monte Carlo methods can be found in [66, 89, 23, 94]. The idea of Monte Carlo integration is to calculate complex integrals by decomposing the integrated function to a multiplication of some other function and some density function defined on $\mathbb{R}^n$. This way, the integral can be calculated as the expectation of the function over the density distribution function from its samples.

Let $I(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$ be an integral which is impossible to solve analytically:

$$I(\boldsymbol{x}) = \int f(\boldsymbol{x}) d\boldsymbol{x} \tag{7.7}$$

If it is possible to decompose $f(\boldsymbol{x}) = h(\boldsymbol{x})p(\boldsymbol{x})$, where $p(x)$ is the density function defined on $\mathbb{R}^n$, the integral in Equation (7.7) can be solved using *Monte Carlo integration* from n samples:

$$\begin{aligned} I(\boldsymbol{x}) = \int f(\boldsymbol{x}) d\boldsymbol{x} &= \int h(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \\ &\approx \frac{1}{n} \sum_{j=1}^{n} h(x_j) \end{aligned} \tag{7.8}$$

The integral can be expressed as an expectation of $h(x)$ over the density $p(x)$ and calculated using a sufficient number of samples $x_1, \ldots, x_n$ from the distribution $p(x)$.

Let us combine the smooth reconstruction of an image and its edge map into variable $\boldsymbol{z} = (\boldsymbol{u}^T, \boldsymbol{v}^T)^T$. The MMSEE and the MMEE, presented in Equations (7.5) and (7.6) can be calculated by generating n samples $\boldsymbol{z}^j$ from the posterior PDF $p(\boldsymbol{z}|\boldsymbol{g})$. Here, the upper index denotes the sample ordinal number, meaning $\boldsymbol{z}^j$ is the j-th sample of $\boldsymbol{z}$. The lower index shows the position of an element in a vector, *e.g.* z_j is the j-th element of $\boldsymbol{z}$. The MMSEE is then approximated by taking the mean of the variable $\boldsymbol{z}$ and the MMEE by taking the median of samples' values.

The estimated variance of each variable serves as a reliability measure of the point estimates.

7.6 The Block-Gibbs sampler

It is impossible to sample directly from the distribution $p(\boldsymbol{z}|\boldsymbol{g})$ in Equations (7.5)-(7.6). Therefore, Markov Chain Monte Carlo methods are used for approximate sampling from some complex distribution.

Metropolis [63] and Hasting [42] propose algorithms to generate samples from some complex distribution by considering samples from another suitable proposal distribution. These samples form a Markov Chain. However, first $b_n - 1$ samples are considered to be *a burn-in phase* in which a chain has not yet converged to its stationary distribution. Therefore, they are discarded. While some convergence tests exist to determine if a chain has reached its stationary state, the number of burn-in samples is usually determined experimentally. Brief overviews of some of these tests are given in [94].

The Gibbs sampler is a special case of the algorithm proposed by Metropolis and Hastings. It was first introduced to image processing by Geman and Geman [34]. Within this chapter, *a block Gibbs sampler* is used. It is different than the stochastic approach presented in [34] because whole vectors are sampled at once, instead sampling each element of vectors at one time. If there is a partition of the target vector $\boldsymbol{z} = (\boldsymbol{a}, \boldsymbol{b})$ such that it is possible to sample from the conditional PDFs $p(\boldsymbol{a}|\boldsymbol{b}, \boldsymbol{g})$ and $p(\boldsymbol{b}|\boldsymbol{a}, \boldsymbol{g})$ in turn, it is possible to approximately sample from $p(\boldsymbol{a}, \boldsymbol{b}|\boldsymbol{g})$.

$$\begin{aligned} \boldsymbol{a}^{j+1} &\sim p(\boldsymbol{a}|\boldsymbol{b}^j, \boldsymbol{g}) \\ \boldsymbol{b}^{j+1} &\sim p(\boldsymbol{b}|\boldsymbol{a}^{j+1}, \boldsymbol{g}), \quad j = b_n, \ldots, K; \end{aligned} \tag{7.9}$$

Hastings [42] claims that, after a considerable *burn-in-phase*, each sample of (7.9) is an approximative sample from the posterior PDF $p(\boldsymbol{a}, \boldsymbol{b}|\boldsymbol{g})$. First $b_n - 1$ samples are considered to be a burn-in-phase and they are discarded. This approach is denoted as *the block-Gibbs sampler* [78].

Conditional PDFs $p(\boldsymbol{a}|\boldsymbol{b}, \boldsymbol{g})$ and $p(\boldsymbol{b}|\boldsymbol{a}, \boldsymbol{g})$ can be derived from the full posterior PDF $p(\boldsymbol{a}, \boldsymbol{b}|\boldsymbol{g})$ by means of the basic multiplicative rule of probability theory as

$$\begin{aligned} p(\boldsymbol{a}|\boldsymbol{b}, \boldsymbol{g}) &= \frac{p(\boldsymbol{a}, \boldsymbol{b}|\boldsymbol{g})}{p(\boldsymbol{b}|\boldsymbol{g})} \\ p(\boldsymbol{b}|\boldsymbol{a}, \boldsymbol{g}) &= \frac{p(\boldsymbol{a}, \boldsymbol{b}|\boldsymbol{g})}{p(\boldsymbol{a}|\boldsymbol{g})} \end{aligned} \tag{7.10}$$

Thus, conditional PDFs $p(\boldsymbol{a}|\boldsymbol{b}, \boldsymbol{g})$ and $p(\boldsymbol{b}|\boldsymbol{a}, \boldsymbol{g})$ can be obtained from the posterior PDF $p(\boldsymbol{a}, \boldsymbol{b}|\boldsymbol{g})$ by setting one variable fixed combined with a suitable normalization.

By investigating the posterior PDF $p(\boldsymbol{u}, \boldsymbol{v}|\boldsymbol{g})$ of the AT energy, we recognize that, by fixing $\boldsymbol{u}$ or $\boldsymbol{v}$ the resulting AT energy becomes a quadratic function of the other variable. Consequently, the corresponding conditional PDFs become Gaussian distributions from which samples can be obtained directly.

If the edge indicator $\boldsymbol{v}$ is fixed, the energy of the conditional PDF becomes (please note that the gradient is approximated by first order forward and backward differences and taking their average Eq. (7.2)):

$$\begin{aligned} E_{AT}(\boldsymbol{u}) &= \beta|\boldsymbol{g} - \boldsymbol{u}|^2 + \alpha \sum_{i=1}^{N_p} \boldsymbol{u}^T \boldsymbol{D}(\boldsymbol{V}) \boldsymbol{u} + c_1; \\ c_1 &= \sum_{i=1}^{N_p} \left(\frac{\rho}{2} |(\nabla v)_i|^2 + \frac{v_i^2}{2\rho} \right); \\ \boldsymbol{V} &= (\boldsymbol{I} - \mathrm{diag}(\boldsymbol{v}))^2 \\ \boldsymbol{D}(\boldsymbol{V}) &= \sum_{i=1}^{N_p} \boldsymbol{B}_{x_i^+}^T \boldsymbol{V} \boldsymbol{B}_{x_i^+} + \boldsymbol{B}_{x_i^-}^T \boldsymbol{V} \boldsymbol{B}_{x_i^-} \end{aligned} \tag{7.11}$$

Quadratic complementation allows to transform the energy:

$$\begin{aligned}
E_{AT}(\boldsymbol{u}) &= (\boldsymbol{u}-\boldsymbol{m}_u)^T\boldsymbol{Q}_u(\boldsymbol{u}-\boldsymbol{m}_u)+c_1+c_2;\\
\boldsymbol{Q}_u &= \beta\boldsymbol{I}+\alpha\boldsymbol{D}(\boldsymbol{V});\\
\boldsymbol{m}_u &= \beta\boldsymbol{Q}_u^{-1}\boldsymbol{g};\\
c_1 &= \sum_{i=1}^{N_p}\left(\frac{\rho}{2}|(\nabla v)_i|^2+\frac{v_i^2}{2\rho}\right);\\
c_2 &= \boldsymbol{g}^T\beta\boldsymbol{I}\boldsymbol{g}-\beta^2\boldsymbol{g}^T\boldsymbol{Q}_u^{-1}\boldsymbol{g}
\end{aligned} \tag{7.12}$$

and finally results in the Gaussian Markov random field (GMRF):

$$p(\boldsymbol{u}|\boldsymbol{v},\boldsymbol{g}) = \frac{1}{2\pi^{\frac{N_p}{2}}}|\boldsymbol{Q}_u|^{1/2}\,\mathsf{exp}\left(-\frac{1}{2}(\boldsymbol{u}-\boldsymbol{m}_u)^T\boldsymbol{Q}_u(\boldsymbol{u}-\boldsymbol{m}_u)\right) \tag{7.13}$$

Applying the same procedure when holding $\boldsymbol{u}$ fixed yields the GMRF:

$$\begin{aligned}
p(\boldsymbol{v}|\boldsymbol{u},\boldsymbol{g}) &= \frac{1}{2\pi^{\frac{N_p}{2}}}|\boldsymbol{Q}_v|^{1/2}\,\mathsf{exp}\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{m}_v)^T\boldsymbol{Q}_v(\boldsymbol{v}-\boldsymbol{m}_v)\right);\\
\boldsymbol{L} &= \alpha\left(\mathsf{diag}\left(\nabla\boldsymbol{u}\right)\right)^2;\\
\boldsymbol{e} &= (1,\dots,1)^T;\\
\boldsymbol{Q}_v &= \boldsymbol{L}+\frac{\rho}{2}\boldsymbol{D}(\boldsymbol{I})+\frac{1}{\rho}\boldsymbol{I};\\
\boldsymbol{m}_v &= \boldsymbol{Q}_v^{-1}\boldsymbol{L}\boldsymbol{e}
\end{aligned} \tag{7.14}$$

Having obtained an appropriate number of samples using the scheme (7.9) with the conditional PDFs (7.13) and (7.14) allows us to approximate different estimators as described in Section 7.4.

7.7 Implementation details

The challenge is to sample from Gaussian distributions given in Equations (7.13) and (7.14) with known mean values $\boldsymbol{m}_u, \boldsymbol{m}_v$ and precision matries $\boldsymbol{Q}_u, \boldsymbol{Q}_v$ respectively. As the first order backward and forward differences are used to approximate gradients (Equ. 7.2), precision matrices in Equations (7.13) and (7.14), $\boldsymbol{Q}_u, \boldsymbol{Q}_v$ are sparse symmetrical matrices that have non-zero values for elements on the main diagonal, the first diagonal below and the first diagonal above the main one.

With an appropriate selection of parameters α, β and ρ matrices $\boldsymbol{Q}_u$ and $\boldsymbol{Q}_v$ are positive definite. As described in Rue and Held [78], this allows us to reduce Gaussian distributions as *Gaussian Markov random fields (GMRF)*. A GMRF has, in contrast to a general Gaussian distribution, a band-limited precision matrix and this leads to an efficient Cholesky decomposition which we use in our sampling scheme.

Next, we apply the block-Gibbs sampler in the following way:

1. Sampling starts from a zero mean Gaussian random vector $\boldsymbol{q}$ with identical covariance matrix, *i.e.* each element of $\boldsymbol{q}$ is the realization of an independent zero mean Gaussian random variable. Gaussian random variables can be generated from uniformly distributed variables using the Box-Muller algorithm [13]. We used the Matlab function `'randn'`.

2. *The Cholesky decomposition* $\boldsymbol{M}$ of the precision matrix $\boldsymbol{Q}$ is computed, *i.e.* $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{M}^T$.

3. The linear equation system $\boldsymbol{M}^T\boldsymbol{y} = \boldsymbol{q}$ is solved by back-substitution.

4. The calculated value of $\boldsymbol{y}$ is added then to the mean $\boldsymbol{m}$. Inserting $\boldsymbol{p} = \boldsymbol{m} + \boldsymbol{M}^{-T}\boldsymbol{q}$ in the definition of the precision matrix directly proves that $\boldsymbol{p}$ is a sample from $\mathcal{N}_{1,N_p}(\boldsymbol{m}, \boldsymbol{Q}^{-1})$.

7.8 Experiments

We implemented the Gibbs sampler in MATLAB using standard built-in functions and performed all experiments on an AMD Phenom II X6 1055T processor running at 3.5 Ghz. For an 128×128 image our block-Gibbs sampler requires about $0.3s$ for each sample, *i.e.* a sample of the image or the edge indicator function. A typical estimate of $K - b_n = 1000$ samples with $b_n = 500$ burn-in samples thus requires about $2 \cdot 0.3(1000 + 500)s = 900s$ which is in the same time range reported for the estimator proposed in [74].

However, in [74] a GPU implementation is considered running on a NVIDIA Tesla C1060, such that we conclude that our algorithm has less complexity. The implementation of our approach on parallel optimized hardware/ software is topic of future research. We compute the MMSEE and MMEE for the $K - b_n$ samples. As we found no significant differences between them, we report only results of the MMSEE in the following.

7.8.1 Experiment 1: Stochastic vs. deterministic AT

In Figure 7.2, results for a noise reduction and edge detection at a triple junction are shown using the deterministic AT [5] and our stochastic approach. We observe that the deterministic approach gets stuck in a local minimum when initializing with the observed images (**k**, **l**), as well as, when initialized with random images (**c**, **d**).

Our stochastic approach (**i**, **j**, **e**, **f**) correctly finds edges and in addition gives information about the underlying distribution irrespective of its initialization. Our approach also allows to estimate standard deviations of image (**g**) and edge-map (**h**). As expected, variances are high for edge positions in the smooth image reconstruction. Variances are low for edge positions in the edge map where the edge signal is highest and most certain.

7.8.2 Experiment 2: Piecewise smooth images and crack-tip problem

Pock and Cremers [74] compare the original AT approach to their MS method, which we consider to be the current reference method concerning achievable segmentation

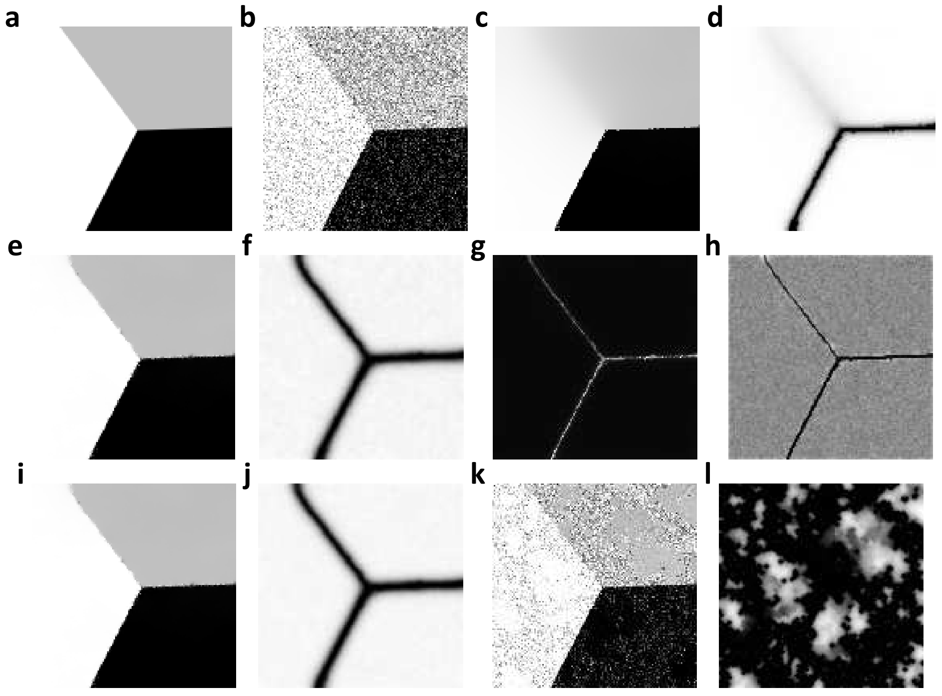


Figure 7.2: Stochastic vs. deterministic AT. The edge map is inverted to its formulation in Equation (7.1), such that variances would be more visible to a human observer. Here, instead of the vector v, vector $1-v$ is shown. The deterministic estimator uses traditional gradient descent minimization. **a** original image; **b** degraded with 20% Gaussian noise; **c**, **d** results obtained using the deterministic AT approach initialized with a random image and edge indicator function, **e**,**f** mean vectors of the estimated image and the inverted edge-map (MMSEE estimator), **g**,**h** standard deviations of estimated image and edge-map of **e** and **f**, respectively, **i**,**j** median vectors of the estimated image and the inverted edge-map (MMEE estimator), **k**, **l** another example of the local minimum obtained while using the deterministic approach (image reconstruction and the edge-map) (parameters: $\alpha = 6000$, $\beta = 12.5$, $\rho = 0.8$).

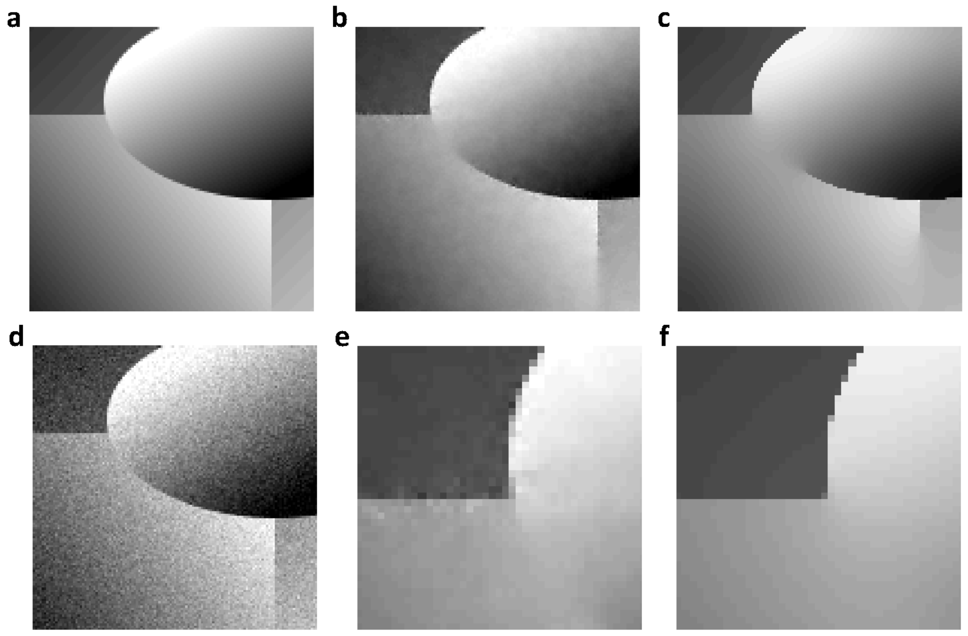


Figure 7.3: **a**, **d** original and degraded image with 5% Gaussian noise; **b**, **e** smoothed image and close up with parameters: $\alpha = 1000$, $\beta = 200$, $\rho = 1.2$; **c**, **f** parameters: $\alpha = 4000, \beta = 70, \rho = 0.3$

quality. They show two experiments which we redo in the following. The first experiment using a synthetic piece-wise smooth 128×128-image with 5% Gaussian noise added (see Figure 7.3 **a** and **d**, and [74], Figure 4) focuses on visual noise artifacts.

Pock and Cremers state worse results for the deterministic AT approach, visible as noise pattern close to image edges. In Figure 7.3, the same experiment is redone for two different choices of ρ, however now using our stochastic AT approach. In the close-up views in Figures 7.3 **e** and **f**, we see that using $\rho = 1.2$ we get similar artifacts as reported in [74], which nicely vanish if ρ is reduced to 0.3 and the smoothness weight α is increased.

The next experiment is the so-called Crack-Tip experiment (*cmp.* Figure (7.4) and [74], Figure 6), an inpainting experiment demonstrating how the deterministic AT optimization approach gets stuck in local minima, depending on initial conditions. The synthetic image I, Figure 7.4**a**, is given by $I(x, y) = \sqrt{r(x, y)}\,\text{sin}(\theta(x, y)/2)$ where $r(x, y)$ is the Euclidean distance of a point (x, y) to the image center and $\theta(x, y)$ is the angle of the point (x, y) to the horizontal line.

The red circle in Figure 7.4 **d** represents the covered/ missing part of the image. Inside the circle, the parameter β is set to be 0, and outside it is defined as ∞. The targeted image, Figure 7.4 **a** is a global minimum of the Mumford-Shah problem [65]. Pock *et al.* [74] use this example to show that their approach reaches the global minimum. The classical AT method works well in the case of good initial conditions, but in other cases it does not always come close to the global minimum of the AT energy, Figure 7.4 **b**, **e**.

Our stochastic approach visibly comes close to the true underlying image independent of initial conditions, even locating the tip of the crack close to the center, Figure 7.4

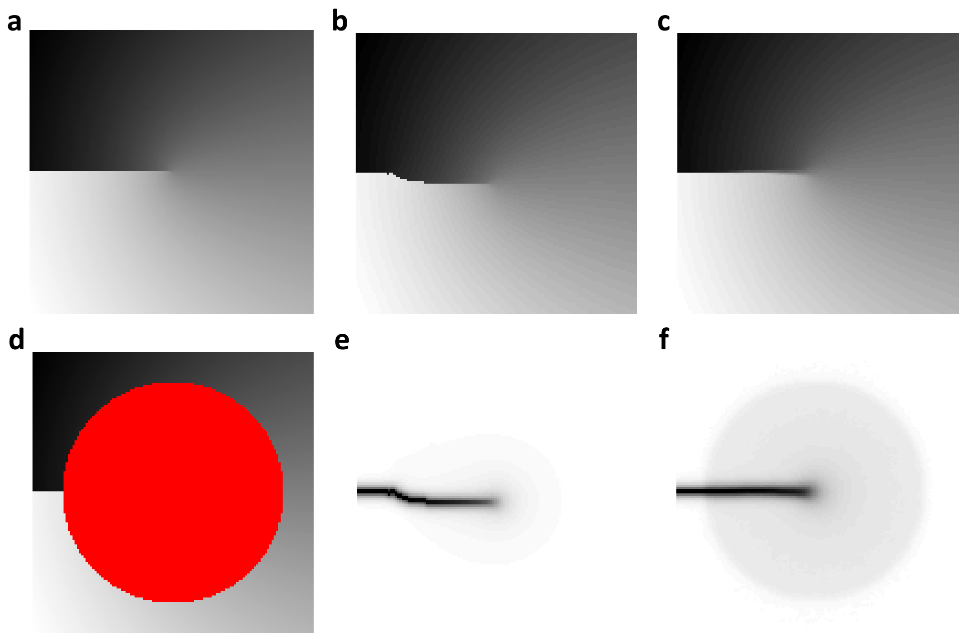


Figure 7.4: **a**, **d** targeted global minimum of MS functional image, and observed image with a circular gap in the middle; **b**, **e** image and border using deterministic AT approach and a random image as an initialization,**c**, **f** our result

c, **f**. Thus, we conclude that when applying a suitable estimator, the AT functional delivers highest quality results, well comparable to the ones reported in [74].

7.9 Summary

We presented the block-Gibbs-sampling approach to the AT smooth image and edge-map reconstruction. The work presented in this chapter was published in papers by Kai Krajsek, Ines Dedović and Hanno Scharr [51, 52].

Geman and Geman [34] introduced MCMC algorithms to image segmentation. They solve an energy functional for restoring a degraded image using a stochastic approach by applying a pixel-wise Gibbs sampler within a discrete energy model. In our approach, the usual AT energy is interpreted as a posterior energy and an efficient block-Gibbs-sampler is used. The block-Gibbs-sampler allows us to sample the whole image or the whole edge-map at once.

MMSEEs and the MMEEs are subsequently obtained by the sample mean and sample median, respectively. No minimization of non-convex functionals is necessary within this estimation framework. In addition, obtaining samples allows us to calculate standard deviations of images and edge-map as a confidence measure easily. This is also achieved by the method of Pätz and Preusser [72], however, using a gradient descent scheme sensitive to local minima.

Our results are visually of the same quality as the ones of Pock and Cremers [74], and also independent of the initial condition. In contrast to Pock and Cremers' approach, we do not need to quantize gray values, allowing for better results on continuous-valued images. We conclude, that we presented the first AT method combining high quality results, independence of initial conditions, and error estimation.

Chapter 8

Conclusion and outlook

The aim of this thesis is to introduce *the channel framework* to Computer Vision problems, mainly image segmentation. With this aim, different methods mainly developed to be used together with KDE are modified to use the channel framework.

8.1 The channel framework

The channel framework is designed as an alternative to KDE methods. It can even be considered as a special case of a discrete implementation of kernel density estimators where kernels are not calculated on data-points but on fixed locations in space. This way, calculations of kernels are completely avoided, but values are just read from different look-up tables. It can also be considered as a special case of histograms where bins are overlapping and usually not rectangle functions. This allows using sub-bin accuracy and for achieving similar results, considerably less bins are needed than for histogram implementations. In contrast to KDE, the idea of the channel framework is to avoid complex calculations, but instead using look-up tables to read pre-calculated coefficients and only simple multiplications and summations are required for obtaining the final PDF. Optimization of look-up tables is a large research area and is outside of the scope of this thesis.

Within this work, we limit an implementation to only uniformly distributed channels on the space of possible pixel values and using the same basis function on all positions. Still, even with this setup we are able to obtain results comparable to state-of-the-art methods. Further investigation will be concentrated on making more advanced look-up tables specialized for different types of images. Together with this, rules on choosing an appropriate look-up table should be developed. Look-up tables are particulary useful when segmenting a set of images of the same type as it is possible to load a look-up table only once and then segment the whole set of images, obtaining similar performance and runtime.

8.2 Image segmentation methods

The following segmentation methods are considered:

- an unsupervised two-region segmentation method,
- a supervised 2-region segmentation method with different user hints,
- an interactive multi-label segmentation method of color images.

Within the experiments, we show how our method competes with others from different points of view: Speed, initialization dependency, sensitivity to *e.g.* bimodal Gaussian distributions, image size, *etc.* .

Most of our methods give similar performance to state-of-the-art methods'. An exception is the bounding-box problem in Section 5.5.4 where our method fails to obtain even close performance to the method by Lempitsky *et al.* [55]. Therefore, this method requires further investigation, as bounding boxes are inappproprate to describe the PDF of objects closely enough.

8.3 Possible extensions

Our methods can be extended to full object detection. In order to achieve this, more features are needed to be considered.

Also, other image processing methods need to be added to include different cues. *E.g.* for the application investigated within the Garnics project, shape priors could be introduced which could improve segmentation of occluded leaves. In the paper by Erdem *et al.* [28] it is shown that a shape prior can easily be introduced to Ambrosio and Tortorelli segmentation. The method presented in the paper has high dependency on initialization and therefore a good choice of region of interest is required.

Another possible research direction is the minimization of functionals. In Chapter 4, the unsupervised 2-region segmentation, includes a novel minimization method. While the non-linear Richardson fixed-point iteration method is quite fast, it is not immune to getting stuck in local minima and its convergence is not proven. The NRI method is also limited to 2-label segmentation, while no minimization method for multi-label segmentation is developed so far.

And finally, the future work can be devoted to developing data-sets and tests for grading the performance. Within this thesis, a number of different test data-sets have been presented. The reason for not considering only one data-set is that all data-sets presented in related papers are targeted to show advantages of a specific type of segmentation method and do not offer a possibility for evaluating different characteristics of segmentation methods. A perfect evaluation data-set would contain different tests that would grade different characteristics of a segmentation method, requiring similar inputs for different tests and reporting results in the same format for all tests.

Appendices

Appendix A

Notation

A.1 Notation

A.2 General notation regarding scalars, vectors and tensors

Scalar values are noted with a regular font. If a scalar value depends on coordinates or is a scalar function of several variables it is noted as:

1. s - scalar value
2. $s(x, y), s(\boldsymbol{g})$ - scalar value dependent on coordinates, or some other variable, *e.g.* $\boldsymbol{g}$

Images are 2D signal values and are usually stored as matrices, or vectors, *e.g.* variable $\boldsymbol{g}$. Matrices and vectors are noted with bold text. Elements of vectors are scalar values and therefore, are noted with the regular font.

1. $\boldsymbol{c}(s)$ - vector function of a signal s
2. $\boldsymbol{c}(s) = [c_1(s), c_2(s), \dots, c_k(s)]$ - individual scalar elements of the vector

Tensors are noted in a similar way. To access individual scalar values of a tensor at some position following notation is used:

1. $\hat{\boldsymbol{c}}(\boldsymbol{s})$ - tensor function of a vector variable $\boldsymbol{s}$
2. $\hat{c}(\boldsymbol{s}, k, i, j)$ - individual scalar elements of the tensor at position (k, i, j)

A.3 The Mumford-Shah functional

The Mumford Shah functional is presented in Section 2.4.1.

- $\Omega \in \mathbb{R}_+^2$ - image domain
- $\Omega_i, \cup_i \Omega_i = \Omega$ - regions in which domain is partitioned

- $g(x, y)$ - the original image
- $u(x, y)$ - smooth reconstruction of the given image $g(x, y)$.
- C - the edge set
- $\lambda, \nu \in \mathbb{R}_+$ - regularization parameters

A.4 The Chan-Vese functional

A.4.1 General notation

The Chan-Vese functional is presented in Section 2.4.2.

- $\Omega \in \mathbb{R}^2_+$ - image domain
- Ω_f, Ω_b - foreground and background region
- $g(x, y)$ - original image
- $u_i \in \mathbb{R}_+$ - constant value within region $\Omega_i, i \in \{f, b\}$
- $\phi(x, y)$ - the level-set function that divides regions
- $C = \{(x, y) \in \Omega \subset \mathbb{R}^2 : \phi(x, y) = 0\}$ - the edge-set

It is implemented with the gray-scale images in Chapter 4, Unsupervised 2-region segmentation of gray-value images.

- $\boldsymbol{c}(x), x \in \mathbb{R}$ - channel vector representation of a value x
- $\boldsymbol{q}_f, \boldsymbol{q}_b$ - channel density estimation of foreground and background respectively
- $p(g(\boldsymbol{x})|\Omega_f), p(g(\boldsymbol{x})|\Omega_f)$ - probability distribution function of foreground and background respectively
- $H(x), \delta(x), x \in \mathbb{R}$ - Heaviside and Dirac function of x
- $H_\epsilon(x), \delta_\epsilon(x), x \in \mathbb{R}$ - Regularized Heaviside and Dirac function of x

The NRI method is presented in the matrix notation:

- $\boldsymbol{\phi}$ - vector representation of the level-set function $\phi(x, y)$
- $\boldsymbol{\phi}_m$ - the m-th iteration of the level set function using the NRI method
- $\boldsymbol{G}(\boldsymbol{\phi}_m), \boldsymbol{r}(\boldsymbol{\phi}_m)$ - transformation functions used within NRI method

In Chapter 5 Chan-Vese approach is used on color images. This means that CDEs are tensors in this case. In addition, prior knowledge is used to adjust the importance of certain channels in CDEs. Therefore, a weight coefficient ω that adjusts the influence of prior knowledge is included.

- $\hat{\boldsymbol{c}}(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^n, n > 1$ - tensors representing channel tensor representation of a vector value $\boldsymbol{x}$
- $\hat{\boldsymbol{q}}_{\phi>0}, \hat{\boldsymbol{q}}_{\phi<0}$ - tensors representing channel density estimation of foreground and background, independent of prior knowledge respectively
- $\hat{\boldsymbol{q}}_{fu}, \hat{\boldsymbol{q}}_{bu}$ - tensors representing channel density estimation of foreground and background, using only prior knowledge on regions
- $\hat{\boldsymbol{q}}_f, \hat{\boldsymbol{q}}_b$ - channel density estimation of foreground and background respectively
- ω_f, ω_b - regularization parameters that adjust the trade-off between the supervised and the unsupervised approach for foreground and background respectively
- $p(g(\boldsymbol{x})|\Omega_f), p(g(\boldsymbol{x})|\Omega_f)$ - probability distribution function of foreground and background respectively.

A.5 The Convex Relaxed Potts model

A.5.1 General notation

The Convex Relaxed Potts model is presented in Section 2.4.3

- $\Omega \in \mathbb{R}_+^2$ - image domain
- $\Omega_i, \cup_i \Omega_i = \Omega$ - regions in which domain is partitioned
- $g(x)$ - original image
- $\rho_i(x) \in [0, 1]$ - measure of pixel x belonging to region Ω_i
- $C_i(x)$ - cost function for region Ω_i
- $\omega(x)$ - penalty function for region edges; within the thesis the following equation is used: $\omega(x) = \lambda \cdot e^{-\gamma|\nabla g(x)|}, \lambda, \gamma \in \mathbb{R}_+$

To estimate regions on an image we need to calculate CDEs of regions and PDFs of regions. CDEs are calculated on areas with user-defined scribbles.

- $N \in \mathbb{N}$ - number of different regions
- $\Omega_i^{us}, i = 1, \ldots, N$ - area on image domain Ω marked by scribbles as a prior knowledge for the region Ω_i
- $\hat{\boldsymbol{q}}_i, i = 1, \ldots, N$ - tensor representing channel density estimation of the region Ω_i
- $p(g(\boldsymbol{x})|\Omega_i)$ - probability distribution function of the region Ω_i.

A.6 The Ambrosio and Tortorelli approach

A.6.1 General notation

The Ambrosio and Tortorelli functional is presented in Section 2.4.4

- $\Omega \in \mathbb{R}_+^2$ - image domain;
- $g(x, y)$ - original image;
- $N_p \in \mathbb{N}$ - the total number of pixels in the targeted image;
- $u(x, y)$ - smooth reconstruction of an image;
- $v(x, y) : \mathbb{R}^2 \to [0, 1]$ - a smooth edge indicator function with $v(x, y) \approx 1$ on the edges, and $v(x, y) \approx 0$ on smooth regions;
- $\boldsymbol{g}, \boldsymbol{u}, \boldsymbol{v}$ - matrix representation of functions $g(x, y), u(x, y), v(x, y)$ respectively;
- $\alpha, \beta, \rho, h \in \mathbb{R}^+$ - regularization parameters;
- $\boldsymbol{B}_{x_j^+}$ and $\boldsymbol{B}_{x_j^-}$ - matrices whereby multiplication gives forward and backward first order finite differences, respectively, along the dimension x_j
- $\boldsymbol{D}$ - matrix gradient operator, used in the following way:

$$\begin{gathered}\sum_{i=1}^{N_p} |(\nabla v)_i|^2 = \boldsymbol{v}^T \boldsymbol{D}(\boldsymbol{I}) \boldsymbol{v}; \\ \boldsymbol{D}(\boldsymbol{I}) = \frac{1}{2} \sum_{j=1}^{2} \boldsymbol{B}_{x_j^+}^T \boldsymbol{I} \boldsymbol{B}_{x_j^+} + \boldsymbol{B}_{x_j^-}^T \boldsymbol{I} \boldsymbol{B}_{x_j^-}\end{gathered} \tag{A.1}$$

- $\boldsymbol{Q}_u$ and $\boldsymbol{Q}_v$ - precision matrices of conditional PDFs $p(\boldsymbol{u}|\boldsymbol{v}, \boldsymbol{g})$ and $p(\boldsymbol{v}|\boldsymbol{u}, \boldsymbol{g})$ respectively;
- $\boldsymbol{m}_u$ and $\boldsymbol{m}_v$ - mean matrices of conditional PDFs $p(\boldsymbol{u}|\boldsymbol{v}, \boldsymbol{g})$ and $p(\boldsymbol{v}|\boldsymbol{u}, \boldsymbol{g})$ respectively;
- $\boldsymbol{q}$ - the realization of an independent zero mean Gaussian random variable;
- $\boldsymbol{p}$ - obtained sample from $\mathcal{N}_{1,N_p}(\boldsymbol{m}, \boldsymbol{Q}^{-1})$.

Appendix B

Evaluating segmentation result

B.1 Introduction

After performing segmentation, it is necessary to evaluate results and grade the quality of segmentation methods. For this purpose, segmentation methods are compared on the same set of images and evaluated using the same evaluation criteria.

To evaluate methods, we need *the correct segmentation outcome*, so that we can compare results of segmentation methods to it. The correct segmentation outcome is usually provided by human subjects and is called *the ground-truth segmentation*.

Therefore, data-sets for evaluation consist of a set of images that need to be segmented and a set of their ground-truth segmentation. Data-sets for evaluation can be aimed at unsupervised and supervised segmentation methods. An example of a data-set aimed at unsupervised segmentation is the one developed for the paper by Alpert *et al.* [4]. Data-sets aimed at interactive segmentation contain also a set of hints/user input for segmentation. Examples of such data-sets are *e.g.* [76, 81, 69].

Next, to evaluate a segmentation method, a measure of segmentation success is required. This chapter provides several measures, *e.g.* a Receiver Operating Characteristic curve (ROC curve), Precision-Recall curves (PR-curves), and F-measure/dice-score. Measures are described in detail in Section B.3.

Data-sets are usually divided in two sets:*a training set* and *a test set*. A training set serves as a guidance for tuning parameters of a segmentation method. It contains representative samples from the data-set. The rest of data forms a test set that serves to evaluate a method's performance.

For needs of this thesis, a tobacco-plant data-set is developed. It contains 235 top-view images of tobacco plants with different size and different number of leaves on each of the images. They are hand-segmented in the following way: On each image all leaves are marked with positions of their tips and bases. Each leaf also contains a flag if it is occluded or not. In Chapter 5 this data-set is used to test if the method is able to separate the plant object from background using a color distribution as a criterion. Segmentation methods are trained on a part of a foreground on an image in Experiment 3 and on a separate leaf collection in Experiment 4. In Chapter 6 scribbles are added to leaves, and the segmentation method is tested if it is possible to separate different leaves. To evaluate methods on the tobacco-plant data-set, the dice score is used.

B.1.1 Data-sets for unsupervised segmentation methods

Data-sets can be also designed to evaluate just a specific class of images, or can be targeted at general segmentation problems. If they are targeted at a general segmentation problem, it is required that they contain a large diversity of images.

Such data-sets are developed for the paper by Alpert *et al.* [4]. The first data-set contains natural images containing one object on the background. The second data-set contains images with two distinguishable objects on background. Both data-sets are aimed at unsupervised segmentation. Each image is segmented by three different subjects. Pixels that are marked to belong to some region by at least two subjects are considered to belong to the region. Data-sets are evaluated using the F-measure/dice-score. As within this thesis unsupervised method is presented, the above mentioned data-set is used for its evaluation. This way, the performance of our method is compared to state-of-the-art methods for unsupervised segmentation. This data-set is chosen from available data-sets since it handles gray scale natural images, segmentation to two regions and compares region-based segmentation methods. These are all targeted characteristics of the method developed in Chapter 4.

B.1.2 Ground-truth data-sets for interactive segmentation methods

Methods developed and presented in Chapters 5 and 6 target interactive region based segmentation methods for color images that segment images in two or more regions. In addition, the aim of this work is to compare CDE to KDE performance as the KDE approach is the most similar one to CDE and at the same time gives state-of-the-art results. Therefore, data-sets that test KDE approaches to mentioned segmentation problems/tasks are considered.

The Grab-Cut data-set was developed for the work by Rother *et al.* [76] and is also used in papers [9, 56, 69]. This data-set is aimed at interactive segmentation. The data-set contains a set of images, a set describing ground-truth segmentation, a bounding-box prior and trimaps resembling a lasso tool selection. The data-set is evaluated using the E_{miss} measure.

Another data-set for interactive segmentation is presented in the PhD work by Santner [80]. This data-set serves for evaluating interactive multi-label segmentation methods. The data-set includes a variety of natural images, scribbles from which they calculate a prior knowledge on an object and a ground-truth segmentation. The dice-score is used for comparing segmentation results to the ground-truth. This data-set is also used in the paper by Nieuwenhuis and Cremers [69].

B.2 Description of the tobacco-plant image data-set

Within the Garnics project, image segmentation is used for locating tobacco plants. Therefore, 235 images containing top views of tobacco plants are chosen to form a ground-truth data set. Examples of images are given in Figure B.1.

On each image, outlines of leaves are marked and a binary mask is formed for each leaf. Each mask has value 1 on positions where the leaf surface is present, and 0 otherwise. For each leaf its base and tip are marked. This way, a leaf angle is calculated and it is possible to extract leaves, and normalize them to same 2D orientation. Each leaf also contains a flag if it is occluded or not. The process of hand-segmentation is given in Figure B.2. Each data-set entry corresponds to one image file and is a `*.mat` file containing a variable `plant` with the following structure:

- `image` - original image
- `leafNum` - number of segmented leaves
- `leaves` - array of leaf structures.

Each `plant.leaves{#}` entry has the following structure:

- `mask` - a leaf mask
- `base` - coordinates of base of a leaf
- `tip` - coordinates of tip of a leaf
- `occlusion` - a flag showing if a leaf is occluded or not
- `angle` - an angle of a leaf

Leaves from images are extracted, normalized to the same 2D orientation and centered on 256×256 px sized images. Examples of extracted leaves are given in Figure B.3.

There are many plant features that can be used in the plant recognition process. We give several examples of plant features that can be used:

Color distribution Color distribution of a plant object is used in Chapters 5 and 6 for plant recognition. Leaves are similar in color distribution, therefore using it as only criterion allows us easily to segment a plant from its background, but distinguishing leaves can be difficult.

Depth information In papers [85, 27, 3, 92, 93] images of plants are paired with depth-maps to enable separation of leaves. Depth-maps allow to separate leaves considering that leaves should be smooth surfaces and singularities in depth maps can respond to object borders. Regions that quite differ in depth information probably belong to different objects. Depth-maps also allow a 3D reconstruction of an image.

Leaf outline and vein structure Shape prior of objects is implemented in segmentation methods *e.g.* [25, 28]. Leaf shape seems like a good idea to consider as a feature within a segmentation method. The problem of using a leaf shape is that the shape is rather simple, and, therefore, influences many false recognitions. Within the Garnics project ellipses are fitted to leaves to separate them [7]. Still, more accurate results are required, as ellipses turn out to be inadequate for fitting occluded leaves. Occluded leaves are most of time merged with the front-most leaf or not recognized at all. If it is possible to locate the vein structure of leaves, distinguishing and separating them would become much easier. Within this thesis, shape priors are recommended for future research.

Other Additional hints about the plant characteristic *e.g.* the possible leaf size, leaf arrangement on stems could improve segmentation results.

The data-set can be used in the following way:

- A segmentation method can be trained on a small number of data-set entries, *e.g.* 20%.
- Prior knowledge on leaves and background is constructed and parameters are tuned to achieve the best segmentation performance.
- With this setup, the segmentation method is run on the whole data-set.
- Leaf collection constructed from extracted leaves can also be used as a training data instead.

The data-set is used in Chapter 5. Since just color distribution is used within our method as a criterion, the task is to locate the whole plant and segment it from background.

In Chapter 6, scribbles are added to leaves and background and the method is tested to distinguish them.

B.3 Evaluating segmentation results

The success of segmentation needs to be quantified, such that segmentation methods can be graded and compared. Metrics used for evaluating segmentation methods originate from evaluation methods for information retrieval [90, 59]. Segmentation results are compared to ground-truth segmentation.

An overview of metrics can be found in [59]. First, we define additional terms. *A true positive (hit)* t_p is a pixel that is correctly assigned to a region. Pixels that are correctly addressed as not belonging to a region are *true negatives (correct rejection)* t_n. Pixels that are assigned to some region and they do not belong to that region according to ground-truth segmentation are *false positives (false alarm)* f_p. *False negatives (miss)* are pixels that are wrongly labeled as not belonging to some region f_n.

Precision P, *recall* R and *fall-out (false positive rate)* F are defined in the following way: Precision gives the percentage of correctly recognized pixels from all recognized pixels as a part of a region, but does not show how many pixels are missed. Recall gives the percentage of detected ground-truth, but does not show how many pixels are wrongly assigned as a part of a region. Fall-out is known as $(1 - specificity)$, where specificity

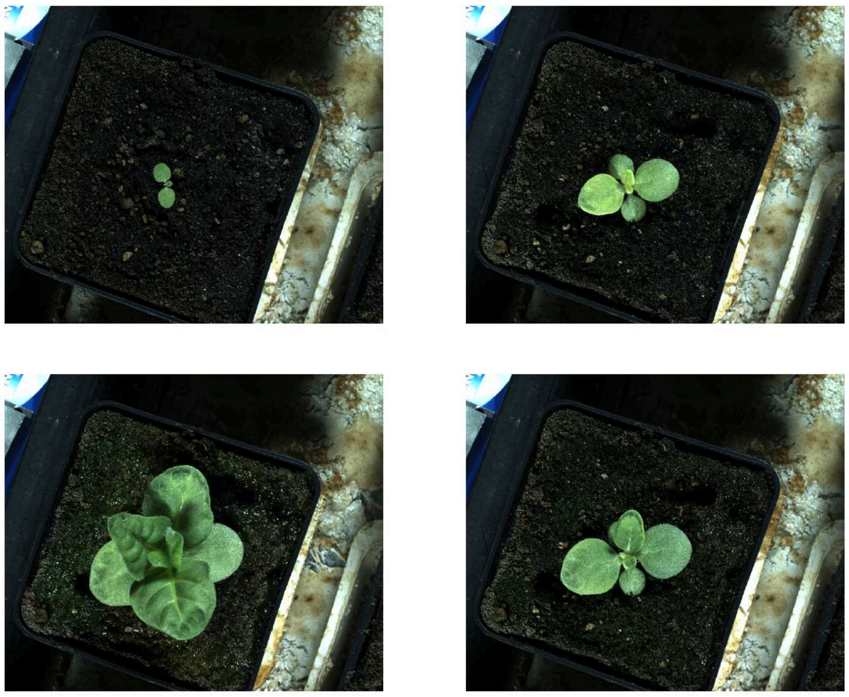


Figure B.1: Examples of tobacco plant images.

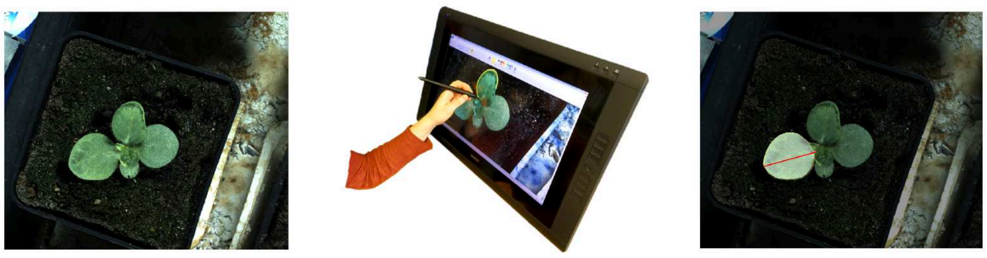


Figure B.2: Process of plant segmentation.

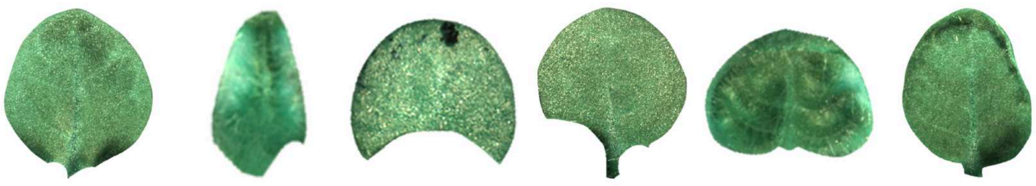


Figure B.3: Examples of extracted leaves.

shows the percentage of correctly unassigned pixels to a region from all unassigned pixels [96].

$$P = \frac{t_p}{t_p + f_p} \tag{B.1}$$

$$R = \frac{t_p}{t_p + f_n} \tag{B.2}$$

$$F = \frac{f_p}{f_p + t_n} \tag{B.3}$$

B.3.1 The Precision-recall curve (The PR-curve)

The precision and recall curve is a plot for evaluating edge-detection and segmentation results. The PR-curve was first introduced to edge-detection methods in the work by Abdou and Pratt [2] and is also used in works by Martin *et al.* [61, 62].

In [60, 61, 62], the authors use PR-curves to evaluate edge-detection methods. A ground-truth is formed from human edge-detection results. These results are combined into a ground-truth map. This map is then smoothed to increase tolerance to small local errors in detection. Next, values for the threshold level are chosen as described in Figure B.4.

Also, for a set of images that present a ground-truth, a histogram can be built that shows how often pixels are chosen to *e.g.* belong to a region Ω_i. Threshold level values are then chosen. For each threshold level value, pixels, that have frequency of being chosen as belonging to the region Ω_i less or equal to the threshold value, are considered as correctly segmented.

The PR curve gives a trade-off between the precision and the recall. To form a plot, precision and recall are calculated for different threshold level values. The perfect PR-curve has the highest precision value on all chosen threshold level values, $P = 1$. The correct edge-detection should match as much as possible the map of high threshold level values. Therefore, the recall should increase as with an increase of the threshold level value as the number of false negatives f_n declines. Once all pixels that correctly belong to Ω_i are obtained, precision should decline to $P = 0$ as a high threshold level value detects no pixels and there are no pixels to be correctly assigned as t_p. On the other hand, any pixel classified as a part of the region Ω_i is a f_p. Recall should stay $R = 1$. This is shown in Figure B.5 **a)**.

B.3.2 The Receiver Operating Characteristic curve (ROC)

The receiver operating characteristic curve (ROC) is another plot for evaluating edge-detection and segmentation methods. It is used in *e.g.* paper by Yitzhaky and Peli [96] and a paper by Bowyer *et al.* [12] The ROC curve gives a trade-off between recall and fall-out.

The ideal ROC curve has a zero value for a fall-out $F = 0$ for all threshold level values until all edge-pixels are retrieved. This is because pixels should not be misclassified as

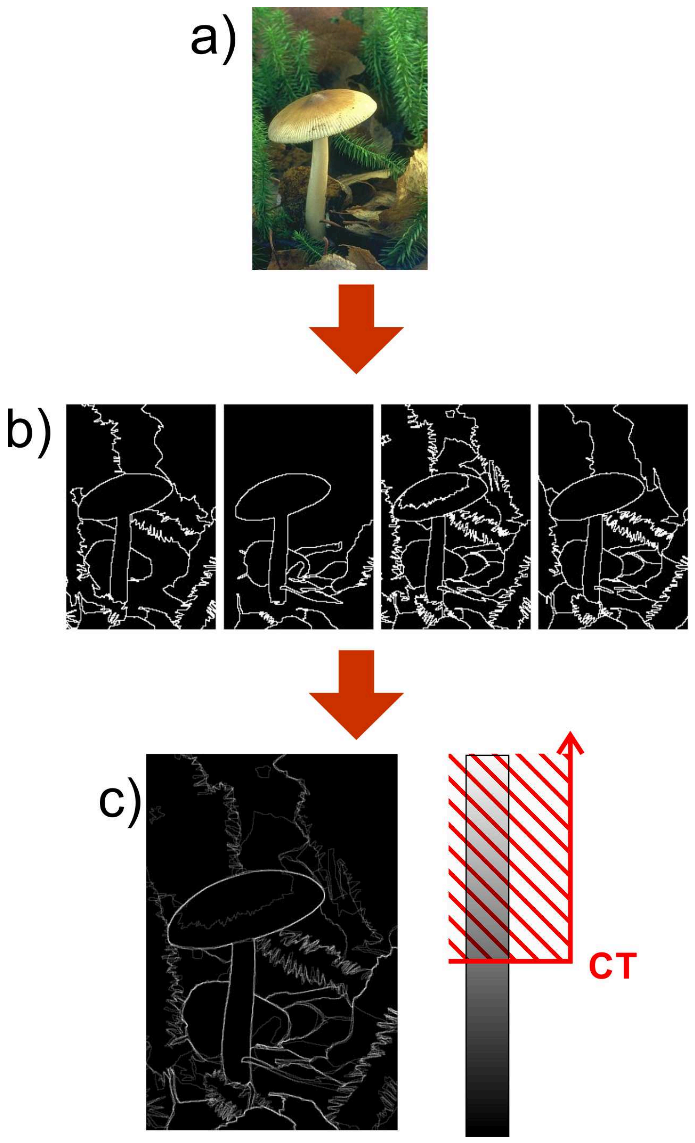


Figure B.4: An example of calculating threshold-levels for ground-truth entries in the Berkeley image data-set [61, 62]. The task is to detect edges in the image **a)**. For the image **a)** a set of different ground-truth results is created **b)**. From this set, a gray-value ground-truth map is formed **c)** by *e.g.* summing up and normalizing all ground-truth results. The map will have lower values for pixels that are chosen as foreground by only few users, while pixels that are chosen as foreground in all human segmentation results have the highest values. For calculating values on a precision-recall (PR) curve, different values for the threshold level are chosen. For each value of the threshold level, pixels on the ground-truth map that have higher values are considered edge pixels.

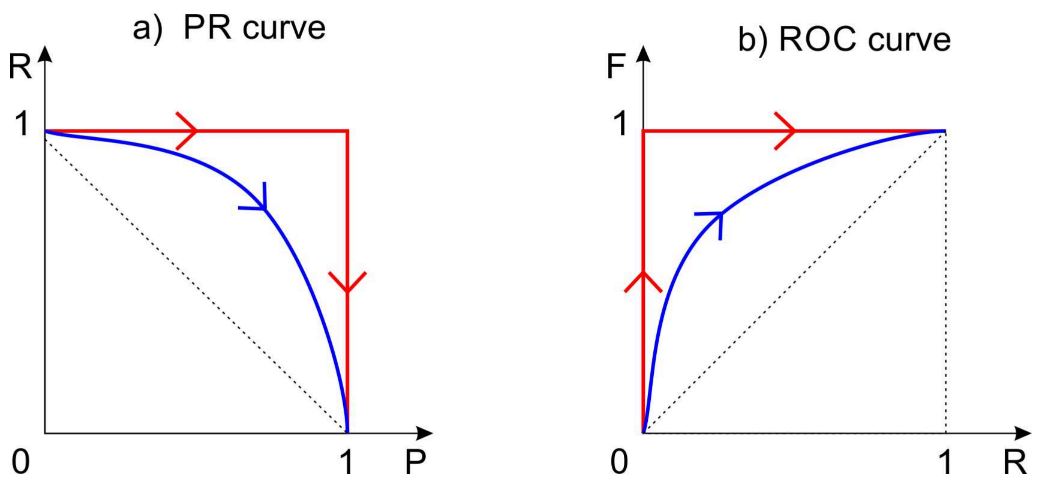


Figure B.5: Examples of a PR curve and a ROC curve. The red plot shows ideal PR and ROC curves. The blue plot gives an example of some PR and ROC curves, as they typically arise in experiments. Arrows show the increase of the threshold level value. Equations for calculating precision, recall and fall-out are given in (B.3)

positives $f_p = 0$. Recall increases as the number of false negatives declines. After the highest threshold level value defined by ground-truth is reached, any higher level contains less pixels that respond to correct edge-detection. Therefore, Fall-out increases as number of false positives increases. This is shown in Figure B.5 **b)**.

In the paper by Yitzhaky and Peli [96], ground-truth results are not provided by human subjects. Instead, ground-truth edge-detection is formed by performing N different edge-detection set-ups. These N experiments produce ground-truth with N possible *correspondence threshold (CT) levels*. Each level consist of pixels with the same or higher frequency of being assigned as an edge. Edge-detection results are calculated for each threshold level value and an ROC curve is formed.

In the paper by Martin *et al.* [60], authors dismiss ROC curves as inefficient for evaluating edge-detectors since fall-out is sensitive to a resolution change. They consider edges to be $1D$ and therefore, while the number of true positives grows linearly as the image resolution increases, the number of true negatives grows quadratically.

B.3.3 The F-measure or a dice-score

For evaluating segmentation algorithms instead of plots, usually precision and recall are combined into one number, *the F-measure*, also referred to as the F1-measure or the dice-score [4, 81, 69]. The advantage of a dice-score is that it gives the grade of a method in just one number.

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{B.4}$$

In Alpert *et al.* [4] unsupervised methods for dividing an image in two regions are evaluated. Each image in the data-set is segmented by three persons and pixels that are marked as a foreground by at least two persons is considered ground-truth foreground. The Grab-cut [76] data-set is also evaluated using dice-score.

In Santner *et al.* [81] authors extend the use of a dice-score for evaluating multi-label segmentation methods. If Ω_{gt} is a ground-truth region and Ω_{al} is the corresponding region calculated by some segmentation method, the score is measured as:

$$F = \sum_{i=1}^{N} dice\left(\Omega_{gt}^{i}, \Omega_{al}^{i}\right) = \sum_{i=1}^{N} \frac{2\left|\Omega_{gt}^{i} \cap \Omega_{al}^{i}\right|}{\left|\Omega_{gt}^{i}\right| + \left|\Omega_{al}^{i}\right|}. \tag{B.5}$$

This data-set and evaluation is used also in paper by Nieuwenhuis and Cremers [69]

Segmentation methods in Chapters 6, 7 and 8 are evaluated using the dice-score measure.

These measures are developed to give a quantity measure to a segmentation method. To discover the nature and the source of the introduced error, additional visual inspections are required.

B.3.4 The misclassification measure

For evaluating segmentation results on trimaps, Rother *et al.* [76] suggest the use of the misclassification measure E_{miss} for the GrabCut data-set. The E_{miss} measure is also used in papers [76, 9, 56, 69].

As mentioned in Chapter 2, trimaps can consist of sets of pixels that are definitive foreground and background, and also a set of pixels that have yet to be classified to either foreground or background. Unclassified pixels are referred to as an unknown area. The misclassification measure shows the percentage of misclassified pixels on an unknown area of an image:

$$E_{miss} = \frac{\text{no. of misclassified pixels}}{\text{no. of pixels in unclassified trimap region}} \tag{B.6}$$

As the number of misclassified pixels is normalized on an area that is given with a trimap, sometimes, E_{miss} measure can be misleading. To explain this, we observe images in Figure B.6. Images depict segmentation results from Chapter 7, Experiment 1. Gray-value coding of trimaps form the GrabCut data-set is given in Table B.1. Segmentation results of images **a)** and **b)** in Figure B.6 produce similar E_{miss} results. However, the trimap for image **a)** has a small unknown area compared to the size of the whole image. Therefore, the poor E_{miss} performance can be a result of a poor behavior of segmentation method near borderline. On the other hand, image **b)** has a large unknown area, and a low E_{miss} value is a result of a failed segmentation as a large part of the foreground is not segmented.

B.4 Summary

In this annex different ground-truth data-sets were presented. They are used in the chapters of the thesis for evaluation of segmentation methods. The data-set developed in

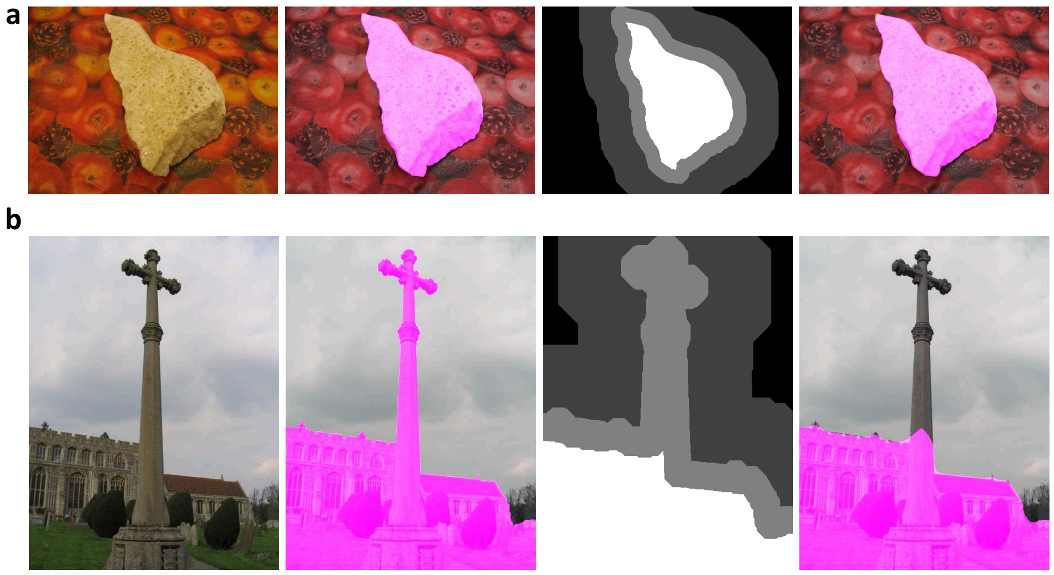


Figure B.6: Results of the method in Chapter 7, Experiment 1: The first image in a row depicts the original image to be selected, the second image is the ground-truth segmentation, the third image is the trimap that is used for segmentation and the fourth is the segmentation result of our method. Coding of trimaps is given in Table B.1 **a)** $E_{miss} = 23.8\%$ **b)** $E_{miss} = 24.6\%$ Both images have a similar misclassification error, while to an eye the error of the first image is 'only' on borders and in the second image the whole part of an object is missing.

Gray value	Meaning
0	background
64	background area used for model training
128	unknown region
255	foreground area used for model training

Table B.1: Gray value coding of lasso-selection trimaps in the GrabCut data-set [76]

the paper by Alpert *et al.* [4] is used in Chapter 4 for evaluating automatic segmentation method in two regions based on a gray-value distribution. The Grab-Cut data-set [76] is used in Chapter 5 for evaluating interactive foreground extraction. The data-set developed within the work of Santner *et al.* [81] is used in Chapter 6 for evaluating interactive multi-label segmentation method. A special data-set for the Garnics project has been developed in the context of the thesis. This data-set contains top-view images of tobacco-plants. It is also used in Chapters 5 and 6.

Different evaluation measures are suggested for evaluating segmentation results. While PR-curves and ROC-curves are used mostly for edge-detection methods [2, 61, 62, 12, 96], F-measure (dice-score) is usually used for region based segmentation methods *e.g.* [81, 69]. E_{miss} measure is used to evaluate algorithms on the GrabCut data-set [76, 56, 69] These measures give an idea of quality of a segmentation method, but segmentation results need to be analyzed for the nature of the segmentation error.

List of Figures

List of Tables

Bibliography

[1] "MATLAB Version 8.3.0.532 (R2014a), help," Natick, Massachusetts, February 11, 2014, http://www.mathworks.de/de/help/matlab/.

[2] I. Abdou and W. Pratt, "Quantitative design and evaluation of enhancement/thresholding edge detectors," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 753–763, May 1979.

[3] G. Alenya, B. Dellen, and C. Torras, "3d modelling of leaves from color and ToF data for robotized plant measuring," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3408–3414.

[4] S. Alpert, M. Galun, A. Brandt, and R. Basri, "Image segmentation by probabilistic bottom-up aggregation and cue integration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 315–327, 2012.

[5] L. Ambrosio and V. M. Tortorelli, "Approximation of functional depending on jumps by elliptic functional via Γ–convergence," *Communications on Pure and Applied Mathematics*, vol. 43, no. 8, pp. 999–1036, Dec. 1990. [Online]. Available: http://dx.doi.org/10.1002/cpa.3160430805

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2010.161

[7] T. Ardeshiri, F. Larsson, F. Gustafsson, T. B. Schön, and M. Felsberg, "Bicycle Tracking Using Ellipse Extraction," in *Fusion 2011*, 2011. Accepted.

[8] S. Bagon, O. Boiman, and M. Irani, "What is a good image segment? A unified approach to segment extraction," in *Computer Vision – ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds. Springer Berlin Heidelberg, 2008, vol. 5305, pp. 30–44. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88693-8_3

[9] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive GMMRF model," in *in ECCV*, 2004, pp. 428–441.

[10] A. Bonnet, “On the regularity of edges in image segmentation,” *Annales de l’institut Henri Poincaré (C) Analyse non linéaire*, vol. 13, no. 4, pp. 485–528, 1996. [Online]. Available: http://eudml.org/doc/78389

[11] A. Bonnet and G. David, *Cracktip is a global Mumford-Shah minimizer*, ser. Astérisque ; 274 ; Astérisque. Paris: Société Mathématique de France, 2001, no. 274.

[12] K. Bowyer, C. Kranenburg, and S. Dougherty, “Edge detector evaluation using empirical ROC curves,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, 1999, pp. –359 Vol. 1.

[13] G. E. P. Box and M. E. Muller, “A note on the generation of random normal deviates,” *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 06 1958. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177706645

[14] Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images,” in *ICCV*, 2001, pp. 105–112.

[15] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” in *EMMCVPR*, 2001, pp. 359–374.

[16] ——, “Computing geodesics and minimal surfaces via graph cuts,” in *ICCV*, 2003, pp. 26–33.

[17] E. S. Brown, T. F. Chan, and X. Bresson, “Completely convex formulation of the Chan-Vese image segmentation model,” *International Journal of Computer Vision*, vol. 98, no. 1, pp. 103–121, 2012.

[18] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.1986.4767851

[19] A. Chambolle, “An algorithm for total variation minimization and applications,” *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 89–97, 2004. [Online]. Available: http://dx.doi.org/10.1023/B%3AJMIV.0000011325.36760.1e

[20] T. F. Chan and L. A. Vese, “Active contours without edges,” *Trans. Img. Proc.*, vol. 10, no. 2, pp. 266–277, Feb. 2001. [Online]. Available: http://dx.doi.org/10.1109/83.902291

[21] T. F. Chan, S. Esedoglu, and M. Nikolova, “Algorithms for finding global minimizers of image segmentation and denoising models,” *SIAM Journal of Applied Mathematics*, vol. 66, no. 5, pp. 1632–1648, 2006.

[22] T. F. Chan, B. Y. Sandberg, and L. A. Vese, “Active contours without edges for vector-valued images,” *Journal of Visual Communication and Image Representation*, vol. 11, pp. 130–141, 2000.

[23] S. Chib and E. Greenberg, “Understanding the Metropolis-Hastings algorithm,” *The American Statistician*, vol. 49, no. 4, pp. 327–335, Nov. 1995. [Online]. Available: http://www.jstor.org/stable/2684568

[24] D. Comaniciu, P. Meer, and S. Member, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[25] D. Cremers, S. Osher, and S. Soatto, “Kernel density estimation and intrinsic alignment for knowledge-driven segmentation: Teaching level sets to walk,” in *Pattern Recognition*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3175, pp. 36–44.

[26] T. Davis, *Direct Methods for Sparse Linear Systems*, ser. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, 2006. [Online]. Available: http://books.google.de/books?id=TvwiyF8vy3EC

[27] B. Dellen, G. Alenyà, C. S.Foix, and Torras, “Segmenting color images into surface patches by exploiting sparse depth data,” in *WACV*, 2011.

[28] E. Erdem, A. Erdem, and S. Tari, “Edge strength functions as shape priors in image segmentation,” in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, ser. Lecture Notes in Computer Science, A. Rangarajan, B. Vemuri, and A. Yuille, Eds. Springer Berlin Heidelberg, 2005, vol. 3757, pp. 490–502. [Online]. Available: http://dx.doi.org/10.1007/11585978_32

[29] M. Felsberg, “Incremental computation of feature hierarchies,” in *Pattern Recognition: 32nd DAGM Symposium, Darmstadt, Germany, September 22-24, 2010. Proceedings*, M. Goesele, S. Roth, A. Kuijper, B. Schiele, and K. Schindler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 523–532. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15986-2_53

[30] M. Felsberg, P.-E. Forssén, and H. Scharr, “Channel Smoothing: Efficient Robust Smoothing of Low-Level Signal Features,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 2, pp. 209–222, 2006.

[31] P.-E. Forssén, “Low and medium level vision using channel representations,” Ph.D. dissertation, Linköping University, Sweden, SE-581 83 Linköping, Sweden, March 2004, dissertation No. 858, ISBN 91-7373-876-X.

[32] M. Galun, E. Sharon, R. Basri, and A. Brandt, “Texture segmentation by multiscale aggregation of filter responses and shape elements.” in *ICCV*. IEEE Computer Society, 2003, pp. 716–723.

[33] M. Gazzaniga, R. Ivry, and G. Mangun, “Cognitive neuroscience: The biology of the mind,” 1998.

[34] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution and the Bayesian restoration of the images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 877–885, 1984.

[35] N. E. Gibbs, W. G. Poole, Jr., and P. K. Stockmeyer, "A comparison of several bandwidth and profile reduction algorithms," *ACM Trans. Math. Softw.*, vol. 2, no. 4, pp. 322–330, Dec. 1976. [Online]. Available: http://doi.acm.org/10.1145/355705.355707

[36] J. R. Gilbert, C. Moler, and R. Schreiber, "Sparse matrices in MATLAB: Design and implementation," *SIAM J. Matrix Anal. Appl*, vol. 13, pp. 333–356, 1992.

[37] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006.

[38] G. H. Granlund, "An associative perception-action structure using a localized space variant information representation," in *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.

[39] G. H. Granlund and A. Moe, "Unrestricted recognition of 3-D objects for robotics using multi-level triplet invariants," *Artificial Intelligence Magazine*, vol. 25, no. 2, pp. 51–67, 2004.

[40] J. Hadamard, *Lectures on Cauchy's Problem: In Linear Partial Differential Equations*, ser. Silliman memorial lectures. Yale University Press, 1923. [Online]. Available: http://books.google.de/books?id=h-_PAAAAMAAJ

[41] ——, "Sur les problèmes aux dérivés partielles et leur signification physique," *Princeton University Bulletin*, vol. 13, pp. 49–52, 1902.

[42] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970. [Online]. Available: http://biomet.oxfordjournals.org/cgi/content/abstract/57/1/97

[43] N. Houhou, J. Thiran, and X. Bresson, "Fast texture segmentation based on semi-local region descriptor and active contour," *Numerical Mathematics: Theory, Methods and Applications*, vol. 2(4), pp. 445–468, 2009.

[44] S. Jehan-Besson, M. Barlaud, and G. Aubert, "DREAM2S: Deformable regions driven by an Eulerian accurate minimization method for image and video segmentation," *International Journal of Computer Vision*, vol. 53, no. 1, pp. 45–70, 2003.

[45] B. Johansson, "Low level operations and learning in computer vision," Ph.D. dissertation, Linköping University, Sweden, SE-581 83 Linköping, Sweden, December 2004, dissertation No. 912, ISBN 91-85295-93-0.

[46] E. Jonsson and L. universitet. Institutionen för systemteknik, *Channel-coded Feature Maps for Computer Vision and Machine Learning*, ser. Linköping studies

in science and technology: Dissertations. Department of Electrical Engineering, Linköpings universitet, 2008. [Online]. Available: http://books.google.de/books?id=mbMCNQAACAAJ

[47] S. M. Kay, *Fundamentals of statistical signal processing: estimation theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[48] J. Kim, J. W. F. III, A. J. Yezzi, M. Çetin, and A. S. Willsky, "A Nonparametric Statistical Method for Image Segmentation Using Information Theory and Curve Evolution," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1486–1502, 2005.

[49] P. Kohli, M. P. Kumar, and P. H. S. Torr, "P & beyond: Move making algorithms for solving higher order functions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1645–1656, 2009.

[50] V. Kolmogorov and Y. Boykov, "What metrics can be approximated by geo-cuts, or global optimization of length/area and flux," in *ICCV*, 2005, pp. 564–571.

[51] K. Krajsek, I. Dedović, and H. Scharr, "An estimation theoretical approach to Ambrosio-Tortorelli image segmentation," in *Lecture Notes in Computer Science, DAGM 2011, / ed. R. Mester, M. Felsberg Berlin, Heidelberg, Springer, 2011. - 6835. S. 41 - 50*, 2011, record converted from VDB: 12.11.2012. [Online]. Available: http://juser.fz-juelich.de/record/14902

[52] ——, "An estimation theoretical view on Ambrosio-Tortorelli image segmentation," in *83rd Annual Meeting of the International Association of Applied Mathematics and Mechanics*, 2012, record converted from VDB: 12.11.2012. [Online]. Available: http://juser.fz-juelich.de/record/20482

[53] K. Krajsek, M. I. Menzel, and H. Scharr, "Riemannian Bayesian estimation of diffusion tensor images," in *The 12th IEEE International Conference on Computer Vision (ICCV 2009)*, 2009, record converted from VDB: 12.11.2012. [Online]. Available: http://juser.fz-juelich.de/record/4878

[54] K. Krajsek and H. Scharr, "Diffusion filtering without parameter tuning: Models and inference tools," in *CVPR*, 2010, pp. 2536–2543.

[55] V. S. Lempitsky, P. Kohli, C. Rother, and T. Sharp, "Image segmentation with a bounding box prior," in *ICCV*, 2009, pp. 277–284.

[56] V. S. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake, "Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography," in *FIMH*, 2009, pp. 447–456.

[57] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850924.851523

[58] J. Malik, S. Belongie, T. Leung, and J. Shi, “Contour and texture analysis for image segmentation,” *IJCV*, vol. 43, no. 1, pp. 7–27, June 2001.

[59] O. Marques, *Practical Image and Video Processing Using MATLAB*. Wiley, 2011. [Online]. Available: http://books.google.de/books?id=xzD25QEo8qYC

[60] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proc. 8th Int’l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.

[61] D. R. Martin, C. Fowlkes, and J. Malik, “Learning to detect natural image boundaries using brightness and texture,” in *NIPS*, 2002, pp. 1255–1262.

[62] D. Martin, C. Fowlkes, and J. Malik, “Learning to detect natural image boundaries using local brightness, color, and texture cues,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 5, pp. 530–549, May 2004.

[63] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *J. of Chem. Phys.*, vol. 11, no. 21, pp. 1087–1091, 1953.

[64] E. N. Mortensen and W. A. Barrett, “Intelligent scissors for image composition,” in *In Computer Graphics, SIGGRAPH Proceedings*, 1995, pp. 191–198.

[65] D. Mumford and J. Shah, “Optimal approximations by piecewise smooth functions and associated variational problems,” *Communications on Pure and Applied Mathematics*, vol. 42, no. 5, pp. 577–685, 1989.

[66] R. M. Neal, “Probabilistic inference using Markov chain Monte Carlo methods,” 1993.

[67] K. Ni, X. Bresson, T. Chan, and S. Esedoglu, “Local histogram based segmentation using the Wasserstein distance,” *International Journal of Computer Vision*, vol. 84, no. 1, pp. 97–111, 2009. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0234-0

[68] C. Nieuwenhuis, E. Toeppe, and D. Cremers, “A survey and comparison of discrete and continuous multi-label optimization approaches for the Potts model,” *International Journal of Computer Vision*, 2013.

[69] C. Nieuwenhuis and D. Cremers, “Spatially varying color distributions for interactive multilabel segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1234–1247, May 2013. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2012.183

[70] M. Nikolova, S. Esedoglu, and T. F. Chan, “Algorithms for finding global minimizers of image segmentation and denoising models,” *SIAM Journal of Applied Mathematics*, vol. 66, no. 5, pp. 1632–1648, 2006.

[71] N. Papenberg, H. Schumacher, S. Heldmann, S. Wirtz, S. Bommersheim, K. Ens, J. Modersitzki, and B. Fischer, “A fast and flexible image registration toolbox,” in *Bildverarbeitung für die Medizin*, 2007, pp. 106–110.

[72] T. Pätz and T. Preusser, “Ambrosio-Tortorelli segmentation of stochastic images,” in *Proceedings of the 11th European conference on Computer vision: Part V*, ser. ECCV’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 254–267.

[73] T. Pock, A. Chambolle, D. Cremers, and H. Bischof, “A convex relaxation approach for computing minimal partitions,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[74] T. Pock, D. Cremers, H. Bischof, and A. Chambolle, “An algorithm for minimizing the Mumford-Shah functional,” in *IEEE International Conference on Computer Vision (ICCV)*, 2009, to appear.

[75] R. B. Potts, “Some generalized order-disorder transformations,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 48, pp. 106–109, 1 1952. [Online]. Available: http://journals.cambridge.org/article_S0305004100027419

[76] C. Rother, V. Kolmogorov, and A. Blake, ““GrabCut“: Interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015720

[77] M. Rousson, T. Brox, and R. Deriche, “Active unsupervised texture segmentation on a diffusion based feature space,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Jun. 2003, pp. 699–704. [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2003/Bro03b

[78] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*. Chapman and Hall, 2005.

[79] H. Rue and H. Rue, “Fast sampling of Gaussian Markov random fields with applications,” *Journal of the Royal Statistical Society, Series B*, vol. 63, pp. 325–338, 2000.

[80] J. Santner, “Interactive multi-label segmentation,” Ph.D. dissertation, Graz University of Technology, October 2010.

[81] J. Santner, T. Pock, and H. Bischof, “Interactive multi-label segmentation,” in *Proceedings 10th Asian Conference on Computer Vision (ACCV), Queenstown, New Zealand*, November 2010.

[82] J. Santner, M. Unger, T. Pock, C. Leistner, A. Saffari, and H. Bischof, “Interactive texture segmentation using random forests and total variation,” in *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, September 2009.

[83] H. Scharr, *Optimal operators in digital image processing*, 2000. [Online]. Available: https://books.google.de/books?id=DJoBfAEACAAJ

[84] H. Scharr, M. J. Black, and H. W. Haussecker, “Image statistics and anisotropic diffusion,” in *ICCV*, 2003, pp. 840–847.

[85] T. Schuchert, *Plant Leaf Motion Estimation Using A 5D Affine Optical Flow Model*. Forschungszentrum Jülich, Energy & Environment, Volume 57: Forschungszentrum Jülich, 2010.

[86] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt, “Hierarchy and adaptivity in segmenting visual scenes,” *Nature*, vol. 442, no. 7104, pp. 810–813, Jun. 2006.

[87] I. Sobel, “History and Definition of the so-called “Sobel operator“,more appropriately named the Sobel-Feldman operator,” 02 2014.

[88] B. T. and C. D., “On local region models and a statistical interpretation of the piecewise smooth Mumford-Shah functional,” *International Journal of Computer Vision*, vol. 84, no. 2, pp. 184–193, Jun. 2009. [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2009/Bro09

[89] L. Tierney, “Markov chains for exploring posterior distributions,” *Annals of Statistics*, vol. 22, pp. 1701–1762, 1994.

[90] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979.

[91] L. N. Vaserstein, “Markov processes over denumerable products of spaces, describing large systems of automata,” *Problemy Peredachi Informatsii*, vol. 5, no. 3, pp. 64–72, 1969.

[92] M. Wallenberg, M. Felsberg, and P.-E. Forssén, “Leaf segmentation using the Kinect,” in *Proceedings of SSBA 2011 Symposium on Image Analysis*. SSBA 2011, Linköping 14-18 March 2011, 2011.

[93] M. Wallenberg, M. Felsberg, P.-E. Forssén, and B. Dellen, “Channel Coding for Joint Colour and Depth Segmentation,” in *Proceedings of Pattern Recognition 33rd DAGM Symposium, Frankfurt/Main, Germany, August 31 - September 2*, vol. LNCS 6835, no. 306-315. 33rd DAGM Symposium, Frankfurt 29 August - 2 September 2011, 2011.

[94] B. Walsh, “Markov chain Monte Carlo and Gibbs sampling,” 2004.

[95] D. Weiler and J. Eggert, “Multi-dimensional histogram-based image segmentation,” in *Neural Information Processing, 14th International Conference, ICONIP 2007, Kitakyushu, Japan, November 13-16, 2007, Revised Selected Papers, Part I*, ser. Lecture Notes in Computer Science, vol. 4984/2. Springer, 2008, pp. 963–972. [Online]. Available: http://tubiblio.ulb.tu-darmstadt.de/43798/

[96] Y. Yitzhaky and E. Peli, “A method for objective edge detection evaluation and detector parameter selection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1027–1033, 2003.

[97] J. Yuan, E. Bae, X. Tai, and Y. Boykov, “A continuous max-flow approach to Potts model,” in *ECCV 2010*.

[98] J. Yuan, C. Schnörr, and G. Steidl, “Simultaneous optical flow estimation and decomposition,” *SIAM J. Scientific Computing*, vol. 29, no. 6, pp. 2283–2304, 2007.

[99] J. Yuan, E. Bae, and X.-C. Tai, “A study on continuous max-flow and min-cut approaches,” in *CVPR 2010*.

[100] R. S. Zemel, P. Dayan, and A. Pouget, “Probabilistic interpretation of population codes,” in *NIPS*, 1996, pp. 676–684.

[101] S. C. Zhu and A. L. Yuille, “Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation.” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 9, pp. 884–900, 1996. [Online]. Available: http://dblp.uni-trier.de/db/journals/pami/pami18.html#ZhuY96