Master's Thesis

# Interactive Vision Based Reconstruction of Statistical 3D Microstructure of Electrodes of Lithium-Ion Cells

Christian Mischitz

mischitz@student.tugraz.at

Institute for Computer Graphics and Vision (ICG)
Graz University of Technology
Inffeldgasse 16,
8010 Graz, Austria


Supervisor: Univ.-Prof. Dipl.-Ing. Dr. techn. Horst Bischof

Graz, June 2012

Masterarbeit

# Interaktive Bildbasierte Rekonstruktion der statistischen 3D Struktur von Lithium-Ionen Zellen

Christian Mischitz

mischitz@student.tugraz.at

---

Institut für Maschinelles Sehen und Darstellen (ICG)
Technische Universität Graz
Inffeldgasse 16,
8010 Graz, Österreich


Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Horst Bischof

Graz, Juni 2012

## Abstract

In the field of lithium-ion cells the investigation of the inner 3D microstructure of electrode materials is an current research topic. It is proven that the electrical characteristics of these materials are depending on their composition and their microstructure. The manufacturers of the cells provide statistical data about the composition, but to obtain the microstructure of the materials expensive procedures are needed, which involve acquiring hundreds of consecutive electron-microscope images.

This master's thesis provides an interactive segmentation tool which reconstructs a statistically identical 3D structure of lithium-ion electrodes based on just one image. To realize this, the arbitrary shaped particles are approximated by touching ellipsoids. The obtained 3D structure is used as a base for accurate modeling of lithium-ion cells.

In this master's thesis the framework of ferns, originally proposed for keypoint recognition, is adapted and used for a classification task. The user provides seed points which cover the characteristics of the different materials; the system uses the ferns to obtain a classification; a subsequent segmentation based on the discrete Potts model is performed. An ellipse fitting algorithm, tuned for the special purpose of separating particle shapes which are common in lithium-ion electrodes, approximates the segmented areas with ellipses. Based on these, ellipsoids defined by their sizes, rotations and relative frequencies are reconstructed and arranged in a 3D volume.

In the end, comprehensive evaluations of the adaptations of the fern framework and the reconstructing algorithm are given. In addition, the performance of the proposed system is evaluated on natural images and compared to other up-to-date segmentation frameworks.



**Keywords:**

Lithium-Ion Cell, Electrode Microstructure, Statistical Reconstruction, Interactive Segmentation, Ferns, Ellipse Fitting

# Kurzfassung

In der Lithium-Ionen-Zellen Forschung spielt die Rekonstruktion der inneren Mikrostruktur der Zellen eine immer wichtigere Rolle. Diese Zellen bestehen aus verschiedenen Materialien, welche mikroskopische Partikel bilden und sich zu einer porösen Struktur zusammenschließen. Es ist erwiesen, dass die räumliche Größe dieser Partikel die elektrischen Eigenschaften der Zelle maßgeblich beeinflussen. Die Produzenten der Zellen geben die statistische Zusammensetzung der Materialien an, um jedoch Information über die räumliche Mikrostruktur der Zelle zu erhalten, sind aufwendige Verfahren nötig. Eine weit verbreitete Methode ist es, hunderte von aufeinander folgende Einzelbilder zu generieren und daraus die 3D Struktur abzuleiten.

Das Ergebnis dieser Arbeit ist ein interaktives Softwaretool, welches es dem User ermöglicht, die statistische 3D Struktur von Lithium-Ionen Zellen anhand nur eines einzelnen Mikroskopbildes zu rekonstruieren. Die Partikel, welche in der Zelle willkürliche Formen annehmen können, werden dabei mit sich schneidenden Ellipsoiden approximiert. Die erstellte 3D Struktur wird für die genauere Modellbildung von Lithium-Ionen Zellen verwendet.

Für die Klassifizierung der verschiedenen Materialien einer Zelle wird das Fern-Konzept verwendet. Ferns wurden ursprünglich eingeführt als schnelle und robuste Methode, charakteristische Punkte in Bildern zu detektieren. In dieser Arbeit wird ihr Konzept erweitert und für die Klassifizierung von verschiedenen Materialien verwendet. Der User markiert charakteristische Stellen für die verschiedenen Materialien, diese Information wird von den Ferns gelernt und für die Klassifizierung des restlichen Bildes verwendet. Anschließend erfolgt eine Segmentierung basierend auf dem diskreten Potts-Model. Ein neuartiger Algorithmus um die gefundenen Partikel mit Ellipsen zu approximieren, wird vorgestellt. Dieser nutzt das hohe vorhandene Wissen über die spezifische Aufgabe, Partikel in Lithium-Ionen Zellen zu detektieren. Aus den gefundenen Ellipsen wird auf die 3D Ellipsoide geschlossen (Größe, Anzahl, Rotation im Raum), welche abschließend in einem diskreten Volumen zufällig angeordnet werden.

Zum Schluss werden ausführliche Evaluierungen aufgezeigt, welche die Adaptierung des Fern-Frameworks und des Rekonstruktions-Algorithmus betreffen. Außerdem zeigen wir, dass das vorgestellte System auch mit natürlichen Bildern Ergebnisse liefert, welche im Bereich aktueller Segmentierungs-Frameworks liegen.

**Schlüsselwörter:**

Lithium-Ionen Zelle, Mikrostruktur von Elektroden, Statistische 3D Rekonstruktion, Interaktive Segmentierung, Ferns, Ellipsen Approximation

## Statutory Declaration

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

## Eidesstattliche Erklärung

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

| | | |
|---|---|---|
| Ort | Datum | Unterschrift |

## Acknowledgements

This master's thesis was carried out at the Institute for Computer Graphics and Vision, Graz University of Technology, in a close cooperation with the research center Virtual Vehicle, located in Graz. I want to thank Prof. Dr.techn. Horst Bischof for supervising this master's thesis from the side of the university and Dr. Alexander Thaler, group-leader at the research center, who gave me the possibility to carry out this very interesting work. I also want to thank Peter Kontschieder and Markus Unger, who provided me with a lot of input which helped me to achieve the aims of this master's thesis. Without them it would have been a lot more difficult to fulfill all requirements of this work. Furthermore I want to thank my colleagues at Virtual Vehicle for inspiring technical talks and especially Wenzel Prochazka for supporting me with many good advices.

At this point I also want to thank my family, especially my father, who always encouraged me to reach my goals and provided me with all the necessary support to finish my studies. Finally, special thanks to my friends and fellow students - as single player I would not have finished my studies by now.

Graz, June 2012 Christian Mischitz

# Contents

# List of Tables

# List of Figures

# 1 Introduction

Lithium-ion cells play an important role in the field of electric vehicles as they are used as the main energy storage. This master thesis is done in cooperation with the research center 'Virtual Vehicle', located in Graz, Austria, where the modeling of performance of lithium-ion cells is one main research topic. The outcome of this work is an interactive segmentation tool, which enables the investigation of the inner structure of lithium-ion cells faster and more economical than with actual methods.

## 1.1 Motivation and Goal

Lithium-ion cells are considered to be an important power source of future electric vehicles. One main advantage is their achievable high energy density, which allows to store more electrical energy than with other feasible electrical materials for a fixed sized battery [WB04]. It is a well studied fact that the chemical composition and the microstructure of a cell determine its performance [CWL$^+$07, FCK09, HLC08]. During the aging process, chemical reactions change the composite and the structure of the cell material, which increases the inner resistance [IU05]. These morphological changes are also expected to impact the capacity and discharging current rate. The modification of the cell material is therefore an active research topic in the field of lithium-ion research [EJCIT11, WCBH10].

An actual research topic is also the simulation of the aging process of a lithium-ion cell, which is not feasible by now, because accurate data about the internal structure of the cells is missing. The manufacturers of the cells typically provide statistical data about the composition of the used materials. Based on that, macro-homogenous models of batteries model the active material as homogenous, isotropic and flawless particles [SN04]. A disadvantage of such a model is provided in [CWL$^+$07], showing that composition-wise statistically identical materials may have different electrical characteristics because of their spatial microstructure.

To model the aging process of a cell more accurately, the used materials need to be split into their component parts and all of them need to be modeled separately according to their electrical characteristics. For this purpose, information about the inner 3D structure is needed. In [EJCIT11] and [WCBH10] the data about the internal structure is obtained by slicing and imaging consecutive parts of a cell. The principal constituents (active material, binder material and pores) are segmented based on their grayscale intensities and via stacking the 3D structure is reconstructed. A drawback of this method is the high amount of needed images, for example 200 in the case of [EJCIT11], which are expensive to produce. In [WCBH10] just the structure of the active material is reconstructed, because of limited contrast of the images.

The goal of this work is to reconstruct the statistical distribution of the volume elements of a lithium-ion cell based on just one image. In [Wal10] it is shown, that the reconstruction is possible, if the image is labeled properly. They focused on the reconstruction and used a basic watershed segmentation algorithm to extract the data of exactly one kind of lithium-ion material. In this work, an interactive segmentation tool is presented, which supports specialists in the field of lithium-ion cells in investigating the microstructure of different kinds of used electrode material. At first a classification algorithm based on ferns [OCLF10] is applied which learns class-specific information from user scribbles. The output of the classifier is used to initialize a subsequent segmentation algorithm, based on a Markov Random Field (MRF) model. To solve the segmentation problem, we use graph cuts for minimizing the energy term formulated via the Potts model. The segmented particles are approximated with ellipses, where a new algorithm for this special purpose is proposed. Based on these ellipses a statistical analysis is performed and the ellipsoids, where the ellipses originate from, are reconstructed. Finally, these ellipsoids, defined by their sizes, rotations and relative frequencies, are randomly positioned in a discrete 3D volume.

## 1.2 Outline

In the next section related work is presented. It starts with the basic function of a lithium-ion cell and gives a short overview about actual research work in the field of lithium-ion cell microstructure investigation. Then, in the computer vision section, the principles of trees [CSK11] and ferns [OCLF10] are reviewed. In the second part of this section Markov Random Fields, the Potts model and graph cuts are explained and how these concepts can be used for image segmentation. The stochastic section shows, how in [Wal10] circles are reconstructed based on cut length distributions. The findings there are extended in Chapter 3.2.1 for reconstructing ellipsoids based on ellipse distributions. The last section of the related work introduces an algorithm based on [Wal10] which arranges given ellipsoids in a discrete 3D volume.

After the related work section, the problem specific work is presented. The section starts with the adaptations, which were needed to use the ferns for image classification. The investigated feature channels and the expanded binary tests compared to the original version of ferns in [OCLF10] are explained in more detail. Then, an ellipse fitting algorithm is proposed, which exploits the strong knowledge about the specific problem domain. The last part of this section shows how the findings in [Wal10] can be extended to reconstruct ellipsoids based on a discrete ellipse distribution.

In Chapter 4 comprehensive tests for image classification and segmentation, as well as for the ellipsoids reconstruction, are presented. It is shown, how the parameters of the final system are determined and how they influence the result. Although the entire system is tuned to segment different materials present in lithium-ion cells, good results are also obtained on most natural images. We show that the performance on the public available ICG-Bench is comparable to other actual segmentation frameworks [San10].

Finally, a conclusion and ideas for future work are given.

# 2 Related Work

The related work section of this master's thesis is structured as follows: First, a basic explanation is given about how lithium-ion batteries work and a short overview about the found methods to investigate the microstructure of a cell with the help of image processing algorithms is provided. Second, the classification and segmentation algorithms, which were investigated for this problem domain, are described in detail. The segmented materials are approximated by ellipses; the obtained ellipse distribution is used to reconstruct the ellipsoids they originate from. In this section the principle of reconstructing circles based on a cut length distribution is given. Finally, the algorithm is introduced, which creates the discrete 3D volume of ellipsoids.

## 2.1 Lithium-Ion Cell

This chapter gives an overview about how lithium-ion cells work, what parts they consist of and why it is necessary to gain information about their inner structure. A short introduction about actual methods for investigating this structure with their proper advantages and drawbacks is provided.

### 2.1.1 Working Principle

Primary cells store electrochemical energy and transform it to electrical energy if the external current circuit is closed. Such cells are meant for one-time use, whereas the electrochemical processes of rechargeable cells can be reversed. In the rest of this document the term cell is used to refer to rechargeable ones. The main parts of a cell are two electrodes, an electrolyte and the housing. If the cell is charged, chemical energy is stored. If the external electrical circuit is closed, a chemical reaction starts which consists of reduction and oxidation at the according electrodes. The negative pole during discharging is commonly called 'anode' and the positive pole 'cathode'. For the rest of the document this notation will be used. During the discharge process, electrons are produced via an oxidation process at the anode and move via the external electrical circuit to the cathode. There, the electrons are absorbed by a reduction process. Within the cell the electrical circuit is closed by ions which move from the anode to the cathode via the electrolyte. The electrolyte can be solid or liquid, but has to be highly conductive for ions and an isolator to electrons. The two electrodes of the cell are isolated by a separator. The separator can be penetrated by ions but is impermeable for electrons, which prevents the cell from an internal short-circuit. In Figure 2.1 the scheme of a cell is illustrated. Important characteristics of a cell are capacity, energy density and

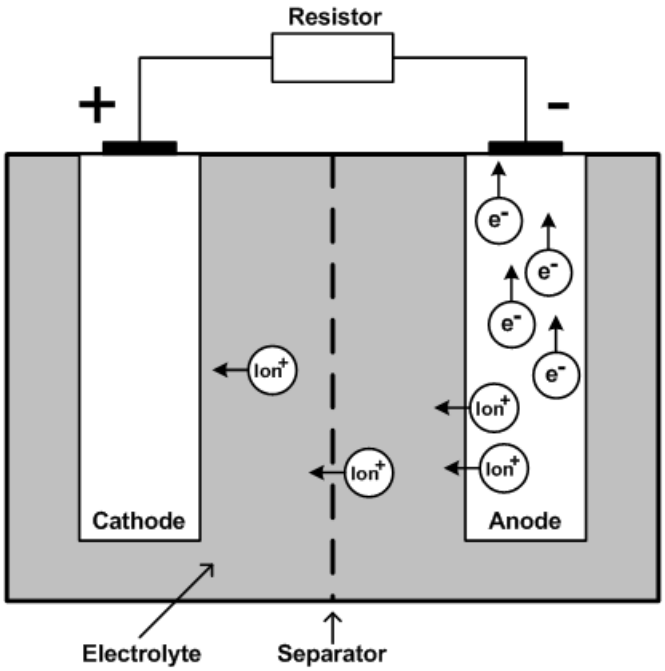



Figure 2.1: Scheme of a discharging cell with external circuit.

achievable power capability. The capacity defines the amount of electrical charge the cell can store and is given in ampere-hours $[Ah]$. As this value is highly dependent on the magnitude of the discharging current, manufacturers often rate the capacity according to a one-hour discharging current. The energy density defines the amount of energy that can be stored in a particular unit volume of the cell and is given in watt-hours per liter $[Wh/l]$. The higher the energy density is, the more energy can be stored in a cell with a given geometrical standard. The achievable power capability defines the highest current rate in ampere $[A]$ which can be performed by the cell without getting damaged [MW05].

In lithium-ion cells, electrodes consist of materials, which enable intercalation of lithium-ions. During the charging process, ions are intercalated within the mesh-structure of the anode, during the discharge process ions are produced at the anode and intercalated in the cathode. An important aspect of the intercalation is that the main-structure of the material is not destroyed, making the process reversible. During the first charging cycles, a passivating layer develops on the surface of the electrodes mainly from a reduction decomposition of the electrolyte. This layer is called solid electrolyte interphase (SEI) and is permeable for the small lithium ions, but preserves the electrolyte from further decomposition [BW04, MW05].

The electrode is made of a porous structure which can be penetrated by the liquid electrolyte. The solid part of the porous structure consists of active material and binder material. The active material is capable of intercalating the ions and therefore determines the capacity of the cell. The more active material is present in a cell of given geometry, the higher the energy density becomes. The achievable power capability is constrained by the diffusion at

the electrodes. The more electrons and respectively ions can be transported within a certain amount of time, the higher the power capability becomes. As the transportation of ions in the solid is slower than in the electrolyte, the diffusion paths should be kept short to reach a high power rate. This is achieved by the porous structure of the material. As the porosity of the material is increased, the diffusion paths become shorter and a higher power capability is achieved. On the other hand an increased porosity implies less active material given a certain unit volume. As the current drain increases also the inner resistance of the electrodes and the transfer resistance between electrolyte and electrode surface have to be taken into account. To lower these, highly conductive binder material is added and enables a homogenous use of the active material and increases its inner conductivity. Unfortunately, adding this binder material leads again to a decrease of the energy density and therefore to a lower capacity of the cell by a given unit volume [MW05, CWL$^+$07].

As these correlations show, the ratio of active material, binder material and porosity is determining the electrical attributes of a lithium-ion cell. In [CWL$^+$07] and [FCK09] the influence of different materials and ratios is further investigated.

### 2.1.2 Vision Based Research

The widely used macro-homogenous model for lithium-ion batteries assumes the material of the cells to consist of isotropic, homogenous, spherical particles, small compared to the electrode thickness. This model is capable of simulating electrode properties such as porosity and SEI-film-thickness. However, the model is not suitable to analyze the full degradation process of a cell. It is a well known fact that during the lifecycle of a cell the inner material structure changes which correlates with the changing of the electrical characteristics [IU05]. To investigate these correlations two basic methods were found. Either just single images were taken to show the existence of morphological changes or an array of consecutive images was taken to reconstruct the full 3D structure.

In [IU05] the correlation of morphological changes and decreasing performance of lithium-ion batteries is investigated. They examined lithium-nickel-cobalt-oxid based materials, used as positive electrode, but their findings are believed to be true for other materials as well. They observed the cells using a microscope equipped with focused ion beam (FIB) technique and a X-ray absorption fine-structure spectroscopic (XAFS) analysis method. To examine the morphology, single images were taken at initial state, after 500 and after 800 charging-cycles. These 2D images show that the primary active particles change their microstructure during the lifecycle. They seem strongly connected at initial state; after 500 charging cycles cracks and clearances between the particles are found and after 800 cycles they are pulverized. In [IU05] a FIB technique is used to investigate the fact that changes of the microstructure correlate with the loss of performance during the lifecycle of a cell, but they do not examine the 3D structure of the material.

In [EJCIT11] the 3D microstructure of lithium-ion electrodes is reconstructed via an array of consecutive focused ion beam-scanning electron microscopy (FIB-SEM) images. They produced 300 SEM images and finally used 200 of them, showing the three main constituent parts of a lithium-ion cell (active material, binder material and pores), which differ in the

image just in their grayscale intensities. The histograms of the images were equalized, combined with median noise filtering and subsequent sharpening. To identify the optimal threshold values, for separating pores, binder and active material from each other, a specialized algorithm is presented. This algorithm estimates the distribution of the grayscale values, automatically identifies the threshold values and applies them to every image. To reconstruct the 3D structure the images were combined via a 3D stacking method. They were able to reconstruct the microstructure of the three main constituent materials of the cell. In that particular case the active material was $LiFePO_4$ and the conductivity additive material carbon black.

In [WCBH10] the same principles were used to investigate the microstructure of a $LiCoO_2$ positive electrode. The authors faced a problem concerning the contrast of their images and therefore just reconstructed the active material $LiCoO_2$. A differentiation between pores and carbon black (binder material) was not feasible.

The main drawback of the 3D stacking method is the high amount of needed images, which are expensive to produce. As in the examples above described, about 300 images need to be produced to be able to reconstruct a representative volume element. As researchers try to simulate the aging process of a cell, which includes the degradation process of the inner structure, more than just one reconstruction during a lifecycle is needed. Therefore some thousand images are needed to get an accurate insight into the morphological changes of a cell.

This work tries to overcome this problem by reconstructing an approximation of the 3D structure out of a single FIB-SEM image. After the segmentation is done, the active material is approximated with ellipses and the statistical 3D structure is determined based on methods described in Chapter 2.4 and [Wal10].

## 2.2 Computer Vision

The first step in reconstructing a statistical 3D microstructure of lithium-ion electrodes is to separate the different materials of such electrodes. As the final system should be able to handle different kinds of lithium-ion cell materials, this chapter presents supervised learning algorithms. It is divided in the two main parts of the separation process. First, the process of classification is explained with the focus on the methods 'classification forest' and 'ferns'. Second, the segmentation process based on energy functions, which are minimized via graph cuts, is described.

### 2.2.1 Classification Algorithms

The aim of supervised learning algorithms is to analyze given training data and to create a classifier or a regression function. Classification refers to the process of associating a discrete label out of a predefined list to any given data point. In the case of regression the algorithm outputs a continuous variable, whos interpretation depends on the specific application. As for

this work relevant, the focus of this chapter is put on classification. In both cases, a sample is a feature vector derived from the basic instance of the sample. Often the features are characteristics which can be calculated from an image, but arbitrary information about a sample can be added as feature. In general a classification algorithm is divided into a training phase and a runtime phase. During training, which is often done offline, the algorithm learns a model which should predict correct labels for unseen samples. If no online learning is implemented, this model does not change during runtime and predicts the output according to the previously learned training data statistics. In the next paragraphs a short introduction to the popular classification methods 'support vector machine' (SVM) and 'boosting' is given, followed by a detailed description of classification forests and ferns in the following subchapters.

Support vector machines are in their standard version a binary, linear classifier, which tries to separate the training examples by the concept of a maximum margin [CV95]. The definition of the margin is the minimal distance between the decision boundary and the nearest training sample. A SVM defines the decision boundary such that the margin for both classes is maximized, which gives the smallest generalization error. The training samples which are the nearest to the decision boundary are called support vectors. They define the boundary and therefore just these need to be stored, all other samples can be discarded. To split training sets which are originally not linearly separable, the data points are mapped to a higher dimensional feature space. There, a linear decision boundary is represented by a maximum margin hyperplane, which leads to a non-linear decision boundary when remapped to the original feature space. This mapping is in general computationally intensive and therefore not done explicitly. The concept of kernel substitution is used, which lowers the computational load and is described in detail in [Bel06]. If the training set cannot be split perfectly, in [CV95] a modified version of the maximum margin idea known as soft margin hyperplane is proposed, which tries to separate the training set with a minimal number of errors. Unseen samples are labeled according to their position concerning the hyperplane. A SVM is originally designed as a binary classifier, but several extensions exist to use them also for more advanced problems. A comparison of the different multi-label SVM methods is given in [DK05]. Also a regression model was proposed in [DBK$^+$97] which is known as support vector regression machine. In [THJA04] support vector machines are adapted to a structured output space and [Pla99] shows how support vector machines can be used for probabilistic outputs.

Support vector machines try to find one decision function, which is strong enough to classify new samples correctly. By contrast, other approaches exist, which do not rely on one decision function, but combine several of them. An advantage of the combination is that every decision function itself does not need to be that strong. In some cases such strong classifiers are not available, or their computational effort is significantly higher than for a group of weaker ones. One method of combining several classifiers is called boosting and was first introduced in [Sch90]. The term boosting generally refers to algorithms which train weak learners consecutive and combine them to a strong one. A weak learner outputs a hypothesis that performs only slightly better than random guessing. A strong learner performs significantly better than any of the weak ones [Sch90]. Whereas the original algorithms were not adaptive, following versions trained the weak learners in a consecutive way to gain better results. The

main principle is to check the performance of every learner and put the focus on misclassified training samples for the subsequent learners. The final strong learner classifies an unseen sample by averaging the predictions of the weak learners, which are weighted according to the results of the learning process. The differences among the boosting algorithms are the way how they use the training samples and according to what rules the combination is carried out. Popular versions of boosting algorithms are AdaBoost [FS95], Gradient boosting [Fri02] and LPBoost [DBST02].

Boosting algorithms train their learners consecutively, but average the outputs of these learners while classifying. Another way of combining learners is to apply them also during the classification phase in a consecutive manner. One example of such classification concepts are decision trees.

### Classification Forest

This chapter describes the principle of decision trees and how they are assembled to a classification forest. Decision trees were introduced in [BFSO84], and in [Ho95] several of them were combined to form a so called forest. In [Ho95] also the method of using randomly selected features to train the trees is proposed. In [CSK11] a comprehensive summary of different forest models is given, which is the base of this chapter.

Classification forests are a common method to solve classification tasks. Important properties of them include [CSK11]:

- naturally handle multi-label problems;
- provide a probabilistic output;
- generalize well to previously unseen data;
- can be efficiently implemented because of their parallelism.

A decision tree is a hierarchical structure of nodes and edges. Nodes have two or more successors and always one predecessor. In this work the focus is put on binary trees where each internal node has exactly two outgoing edges. Figure 2.2 shows a binary decision tree; the top node is called root node and nodes without a successor are called terminal nodes or leaf nodes. To label a sample with a decision tree, it is injected to the root node. The root node, and every internal node, passes the sample to one of its successors according to a decision function associated to this node. When the sample reaches a leaf node, the tree predicts an output for this sample.

***Training:*** The aim of the training phase is to set up the tree structure, find the decision functions and assign predictors to the leaf nodes. The depth $D$ of a tree is defined during training phase, at which several criteria can be used to determine it. To select proper decision functions, their quality is measured by the achieved information gain. These concepts will be described later in this chapter in detail.

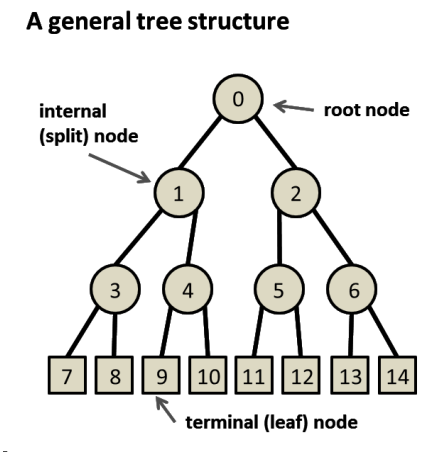



Figure 2.2: Binary decision tree. Image taken from [CSK11].

Data points are represented as feature vectors $\boldsymbol{v}$, which may consist of a large number of feature responses. Training data is also associated with a ground truth label. The decision functions associated with each node will be called split functions $h$. A split function passes every input data $\boldsymbol{v}$ of a node either to its left or right child. This implies that in general every node, except of the root node, is reached by a subset of the entirety of the training samples. The subset reaching a node as input data is denoted by $S_i$; the subsets which are passed to the left and right successor are denoted by $S_l$ and $S_r$, respectively. For each split node the following properties apply: $S_i = S_l \cup S_r, S_l \cap S_r = \emptyset$. These subsets are used to determine the quality of the induced splits by calculating the information gain. The information gain is the most common criterion used to select the decision function. Before talking about information gain the term entropy needs to be defined, whereby in this work it refers to the Shannon entropy. Shannon introduced in [Sha01] a concept to measure the uncertainty of the state of a discrete random variable. This concept is called entropy and can be used to determine the information content of a randomly chosen variable out of a set $B$ of data points. With $\kappa$ as the space of classes and $p(c)$ as the likelihood of class $c$ in the distribution, the entropy $H$ of the set $B$ is mathematically defined as:

$$H(B) = -\sum_{c \epsilon \kappa} p(c) \cdot \log_2(p(c)). \tag{2.1}$$

If in a set one class is predominant, and the amount of all other classes is comparable small, the entropy also will be very small. The information content of a randomly chosen variable out of this set is small, because it is very likely to choose the dominant class. If all classes are equally distributed, the entropy has the highest value because no prediction about a preferred class can be made.

In the context of decision trees we want to split the training samples into subsets such that the resulting entropy value of each subset is minimized. To reach this state, after every split the

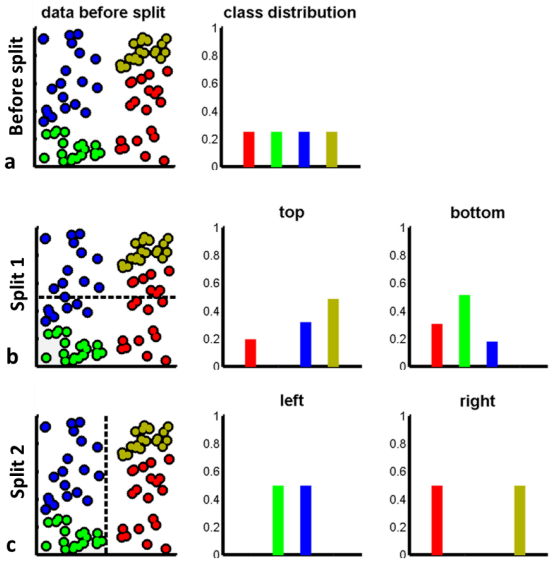



Figure 2.3: Information gain for discrete distributions: (a) Dataset S before a split. (b) After a horizontal split the dataset is not well separated, which leads to a lower information gain. (c) A vertical split separates the dataset better and therefore leads to a higher information gain. Image taken from [CSK11].

created subsets should have lower entropy than the input set. With such a split, information about the classes in the sets is gained, and therefore this concept is called information gain. With $H(S)$ being the entropy of a set $S$, which is split in the subset $S_l$ and $S_r$, the information gain $I$ is computed as:

$$I = H(S) - \sum_{i \in \{l,r\}} \frac{|S_i|}{|S|} \cdot H(S_i). \tag{2.2}$$

In Figure 2.3 an illustrative example is given. Figure 2.3(a) shows data points in a 2D feature space; colors indicate the different classes. The number of points for each class is the same, therefore a uniform class distribution results. If a horizontal split like in Figure 2.3(b) is applied, each of the resulting subsets contains just three different classes. This is associated with lower entropy compared to the initial set in Figure 2.3(a). The achieved information gain is according to [CSK11] $I = 0.4$. If this set is split vertically, like in Figure 2.3(c), the data is separated better and each subset contains just two classes. Compared to the horizontal split the entropy of these subsets is even lower, which yields a higher information gain of $I = 0.69$. With this concept the optimal split function for every node can be selected; the detailed procedure will be described later in this chapter.

*Split function*: A split function $h$ is defined as

$$h(\boldsymbol{v}, \theta) \in \{0, 1\}, \tag{2.3}$$

with $\boldsymbol{v}$ as the input data vector; $\theta$ are the parameters of the function and the output 0 and 1 indicates whether the processed data is passed to the left or right child. $\theta = (\phi, \psi, \tau)$ and characterizes the split function. $\phi$ is a filter function and selects some features out of the feature vector $\boldsymbol{v}$. This lowers the complexity of the split function and leads to the hierarchical test structure which is inherent for decision trees. $\psi$ defines the geometric primitive used to separate the data. In Figure 2.3(b) and (c) axis-aligned linear separators are shown for a 2D case, but arbitrary decision boundaries can be chosen, as long as they can be defined by parameters. In higher dimensional spaces the equivalents are hyperplanes or general surfaces. $\tau$ defines a vector of thresholds for the decision boundaries.

While setting up the tree the training algorithm selects for every node the optimal parameters $\theta^*$ out of a random subset of all available parameters. The concept of randomness is described in detail later in this chapter. $\theta^*$ is determined by maximizing an information gain objective function

$$\theta^* = \arg\max_{\theta} I(S_i, S_l, S_r, \theta), \tag{2.4}$$

with $I$ depending on the input data set $S_i$, the subsets after the split $S_l$ and $S_r$, and the parameters $\theta$. This maximizing of the information gain is the reason why training of trees is computationally expensive. To calculate the information gain, the whole input set needs to be processed according to the split function. To find the optimal settings, the procedure needs to be repeated for every set of parameters $\theta$ out of the random subset. If the optimal split function is found the according subsets are passed to the successor nodes, which again try to optimize their split functions.

The procedure can be repeated until every leaf node contains just one training sample, which is a subset that cannot be split any more. This would lead to a decision tree which separates the training samples perfectly but shows highly overfitting behavior. To avoid this, several criteria can be used to decide when to stop growing each individual branch. A content independent choice would be to stop after a maximum tree depth $D$ (level of consecutive nodes of a branch) is reached. Other versions take the processed data into account and stop when a split does not achieve more than a minimum information gain. Another version is to stop when a node contains less than a minimum number of data points.

*Leaf nodes*: A decision tree consists of several internal split nodes with the associated split functions and several leaf nodes, which holds information for the final output. In a classification case, a leaf node may store the empirical distribution over the classes associated to the training data points which reached that node during training. The probabilistic leaf predictor model is then

$$p(c|\boldsymbol{v}), \tag{2.5}$$

with $c \in \kappa$ indexing the class and $\boldsymbol{v}$ being the input data vector. During runtime a data vector is inserted into the root node of the tree, passed through the tree according to the learned split functions, and in the end reaches a leaf node. With the leaf predictor model a probability $p$ of vector $\boldsymbol{v}$ belonging to a class $c$ is given.

***Forest***: To achieve generalization and robustness against noisy data, not just one decision tree is used to classify data samples, but several of them are combined to an ensemble called forest. The input data vector $\boldsymbol{v}$ is processed by every tree separately which leads to several predictions $p_t(c|\boldsymbol{v})$, with $t = 1, ..., T$ and $T$ being the number of trees in the forest. The final prediction $p(c|\boldsymbol{v})$ of a forest can be calculated either by averaging the predicted distributions of the trees:

$$p(c|\boldsymbol{v}) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|\boldsymbol{v}), \tag{2.6}$$

or by multiplying the outputs together:

$$p(c|\boldsymbol{v}) = \frac{1}{Z} \prod_{t=1}^{T} p_t(c|\boldsymbol{v}). \tag{2.7}$$

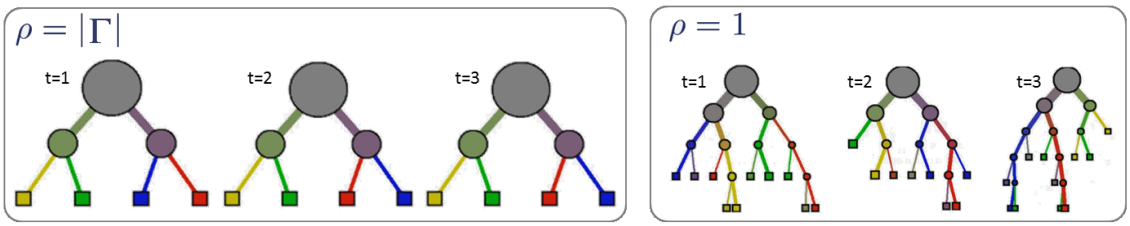
There, $Z$ is a partition function ensuring probabilistic normalization. The advantage of the averaging version is that outliers, for example caused by noisy training data, do not have a significant impact on the result. The multiplicative approach is far more influenced by the individual outputs of each tree. For example, a single tree with a very small predicted probability for a sample belonging to a certain class, may lower the overall probability of the forest significantly, even if the output of all other trees predict the opposite. Depending on the application this behaviour can be desirable, if for example just samples with a high overall confidence should be labelled.

***Randomness***: A forest outperforms a single tree just if the individual trees in the forest are different to each other. This is achieved by introducing randomness in the training phase. Two of the most popular methods are 'randomized node optimization' and 'bagging'. Bagging was first introduced in [Bre96] and is an acronym for 'bootstrap aggregating'. The idea is to train every tree $t$ with a different set of training data. This is done by randomly sampling a subset $L_t \subseteq L$ for every tree, with $L$ denoting the original training set.

An alternative way of inserting randomness to the training procedure is randomized node optimization. The training set $L$ is the same for all nodes, but for the parameters $\theta$ of the split function just a randomly selected subset is available. The training procedure is done like described, but from the entire set $\Gamma$ of possible parameters $\theta$, for every node $j$ just a subset $\Gamma_j \subseteq \Gamma$ is available to optimize it. The objective function becomes

$$\theta_j^* = \arg \max_{\theta_j \in \Gamma_j} I_j, \tag{2.8}$$

with




(a) High tree correlation (b) Low tree correlation

Figure 2.4: Figure (a) shows a forest with a large value of $\rho$, which results in trees which are quite similar to each other. In Figure (b) $\rho$ is small and the resulting trees show a higher variation than in (a). Images taken from [CSK11].

$$I_j = I(S_j^i, S_j^l, S_j^r, \theta_j). \tag{2.9}$$

$S_j^i$ denotes the input set for node $j$; $S_j^l$ and $S_j^r$ are the subsets passed to the sucessors of the node. The amount of randomness is controlled by the ratio $|\Gamma_j|/|\Gamma|$. For easier notation the parameter $\rho = |\Gamma_j|$, $\rho = 1, ..., |\Gamma|$, is used. If $\rho = |\Gamma|$, all trees are trained with the same set $\Gamma$ and therefore the correlation between the trees is the highest. This leads to a behaviour of the forest very much like a single tree. Figure 2.4(a) shows an example of such trees which are highly correlated. If $\rho = 1$, maximum randomness is achieved and the trees are very different to each other, which results in better generalization behaviour. An example of such highly uncorrelated trees is shown in Figure 2.4(b).

### Ferns

In [OCLF10] a new framework for keypoint recognition in images called 'random ferns' is proposed. Ferns are related to random trees, but are made of a non-hierarchical structure and their output is combined in a Semi-Naive Bayesian manner. This chapter describes the principle of the ferns like proposed in [OCLF10] and shows how they are used for a classification task.

To stress the differences between random ferns and random trees a short example will be given how a classification task is handled by a classification forest. Images of different objects in various view points are given and should be classified by a forest. During training phase several trees are constructed which are tuned to split the entire training set as good as possible. The feature space is divided step by step in a hierarchical manner. For the classification of a test image the output of all trees is averaged and the label with the maximum likelihood is assigned to it. In [AG97] it is shown that very simple binary tests as decision functions are sufficient to reach acceptable performance for shape quantization. Random ferns are also based on simple binary tests, but these tests are combined non-hierarchically and chosen randomly, without checking their suitability. To determine a label for a test image, the output of the ferns is combined in a Semi-Naive Bayesian way.

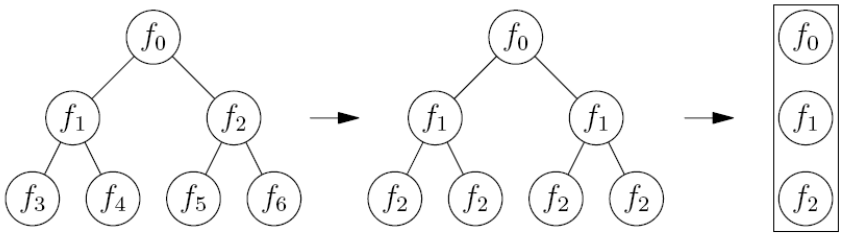



Figure 2.5: Ferns vs. trees: trees consist of several branches with unique tests. Ferns perform the same test on every level of the tree, therefore a tree structure is not needed anymore and a sequence of tests is sufficient. Image taken from [OCLF10].

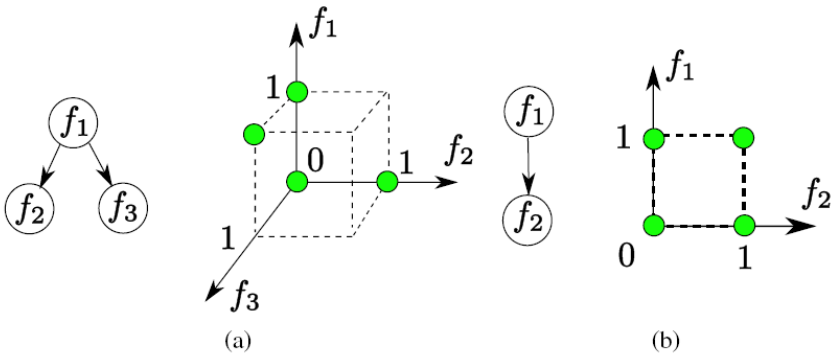



Figure 2.6: Feature space of trees and ferns: A tree of depth 2, shown in Figure (a), offers 4 possible combinations of feature values (denoted by the green circles). A fern of size 2, shown in Figure (b), offers also 4 possible combinations. A detailed description is given in the text. Image taken from [OCLF10].

***Comparison with trees***: Ferns are a simplification of trees. To compare them, we assume that both methods use simple binary tests as decision functions. These binary tests, according to [OCLF10] denoted by $f$, are the equivalents of the split functions $h$ of decision trees. Whereas a tree structure consists of unique branches defined by its binary tests $f$, ferns perform the same binary test $f$ in every node of one level. As the branches are identical they are in fact not needed anymore and the tree structure can be reduced to a sequence of binary tests. This is illustrated in Figure 2.5. In [OCLF10] it is shown that the simplified structure does not entail any performance loss. Figure 2.6 illustrates that ferns and trees of the same depth $D$ offer the same amount of combinations in their feature space. The tree in Figure 2.6(a) has a depth of 2 and offers 4 possible combinations of feature values. Although 3 features are used, just the 4 combinations denoted by the green circles are possible. Other combinations are not feasible because just 2 out of the 3 features can be evaluated in one branch. The fern in Figure 2.6(b) with size (equivalent to the depth of trees) 2 offers also 4 possible combinations of feature values but has a much simpler structure.

Ferns share important properties with trees, including:

- naturally handling multi-label problems;
- providing a probabilistic output;
- generalizing well to previously unseen data;
- efficiently implementations possible because of their parallelism;

but have two main advantages. The training of ferns is faster, because no optimization of the binary tests, like done with the decision function of trees, is performed. Furthermore, [OCLF10] shows that ferns outperform trees, especially if the number of classes increases.

***Semi-Naive Bayesian method***: Recently it was shown in [OCLF10] that the power of a classification forest does not derive from the tree structure itself but from the advantage of combining groups of binary tests, which improves the classification rate. In [OCLF10] just one kind of a simple binary test of the form

$$f = \begin{cases} 1, & \text{if } Int(\boldsymbol{p}_1) < Int(\boldsymbol{p}_2) \\ 0, & \text{otherwise} \end{cases} \tag{2.10}$$

is used. $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are two randomly chosen pixel locations within the image. $Int(\boldsymbol{p}_1)$ represents the intensity value of the location $\boldsymbol{p}_1$. These tests are referred to as binary feature tests $f$ and indicate if the intensity at location $\boldsymbol{p}_1$ is lower than at $\boldsymbol{p}_2$. In a classification task with $\kappa$ as the space of classes, we assume $c_i, i = 1, \ldots, |\kappa|$ to be the set of classes and $f_j, j = 1, \ldots, N$ to be the set of binary features calculated for an image. To assign a label to an unseen image, the maximum likelihood of this image belonging to class $c_i$, accounting the associated features $f_1, f_2, \ldots, f_N$, need to be found:

$$\hat{c_i} = \arg\max_{c_i} P(C = c_i | f_1, f_2, \ldots, f_N), \tag{2.11}$$

where $\hat{c_i}$ denotes the assigned class label and $C$ is a discrete random variable that represents the class. Applying Bayes' Formula yields

$$P(C = c_i | f_1, f_2, \ldots, f_N) = \frac{P(f_1, f_2, \ldots, f_N | C = c_i) P(C = c_i)}{P(f_1, f_2, \ldots, f_N)}. \tag{2.12}$$

As the denominator is just a scaling factor independent from the class and $P(C = c_i)$ can be assumed as a uniform prior, the problem reduces to finding:

$$\hat{c_i} = \arg\max_{c_i} P(f_1, f_2, \ldots, f_N | C = c_i). \tag{2.13}$$

To reach accurate classification based on simple binary features, many of them are required. For example in the implementation of the framework in [OCLF10] $N = 300$ binary features were used. To represent the complete joint probability of equation (2.13), $2^N$ entries would

need to be estimated and stored for each class. As this is not feasible, simplifications are needed. One method would be to assume complete independence among the features, which leads to:

$$P(f_1, f_2, \ldots, f_N | C = c_i) = \prod_{j=1}^{N} P(f_j | C = c_i). \tag{2.14}$$

As the features are correlated to each other, this method does not give acceptable results. A good compromise between computational effort, storing effort and modeling the dependencies of the features is to partition them into groups. The $N$ features are grouped into $M$ groups of size $S = N/M$. These groups are called Ferns and the joint probability of the features within a group is determined. The groups are assumed to be independent from each other and the conditional probability becomes

$$P(f_1, f_2, \ldots, f_N | C = c_i) = \prod_{k=1}^{M} P(F_k | C = c_i). \tag{2.15}$$

$F_k = \{f_k^1, f_k^2, \ldots, f_k^S\}$ is the set of features used for the $kth$ fern. $f_k^x$ represents the $xth$ feature of the $kth$ fern and is chosen randomly out of the $N$ binary features. Equation (2.15) shows a Semi-Naive Bayesian approach by modeling only some of the dependencies between all features. That approach decreases the initially amount of $2^N$ entries to $M \times 2^S$. The practical implementation presented in [OCLF10] gave good results with $M = 50$ and $S = 11$. $M \times 2^S$ is therefore in the order of $10^5$ and much smaller than $2^N$ which is in the order of $10^{165}$.

***Training***: Analog to the training of random trees the aim of the training phase of ferns is to estimate the class conditional probabilities $P(F_k | C = c_i)$ used by equation (2.15). For a simple notation (and implementation), all binary features $f$ of a fern are stacked together in a fixed sequence which forms a binary vector of size $S$. This vector is interpreted as a decimal number, which is called the value $k$ of a fern. The probability $p_{k,c_i}$ of a fern $F_m$, $m = 0, \ldots, M$, responding with the value $k$, $k = 0, \ldots, 2^S - 1$, for a class $c_i$ can therefore be written as:

$$p_{k,c_i} = P(F_m = k | C = c_i). \tag{2.16}$$

Ferns can take $K = 2^S$ values and for every class $c_i$ the constraint

$$\sum_{k=0}^{K-1} p_{k,c_i} = 1 \tag{2.17}$$

needs to be fulfilled. A simple approach would be to assign the maximum likelihood estimates from the training samples to these parameters $p_{k,c_i}$:

$$p_{k,c_i} = \frac{N_{k,c_i}}{N_{c_i}}. \tag{2.18}$$

$N_{k,c_i}$ represents the number of training samples of class $c_i$ that evaluated for one particular fern to value $k$. $N_{c_i}$ is the total number of samples for class $c_i$. In practice this version has a main drawback. If no training sample at all evaluates to a particular value $k$, the according $N_{k,c_i}$ and consequently $p_{k,c_i}$ would be zero for one fern. As the outputs of all ferns are combined multiplicatively, the resulting probability for the class $c_i$ would be zero, no matter what the other ferns respond. Beside the fact that the vote of a single fern should never be that strong to cancel out a possible label for an image, the reason for such probabilities $p_{k,c_i} = 0$ may simply be an artifact of the limited size of the training set. In practice training sets are not arbitrary big and for some classes $c_i$ there simply might be no instances which evaluates to some particular values of $k$. To overcome this problem $p_{k,c_i}$ is defined as

$$p_{k,c_i} = \frac{N_{k,c_i} + N_r}{N_{c_i} + K \times N_r}, \tag{2.19}$$

where $N_r$ represents a regularization term. If during training no sample evaluates to a specific value of $k$ ($N_{k,c_i} = 0$), this definition will still assign a non-zero value to the corresponding probability. [OCLF10] investigated the influence of the parameter $N_r$ and states that a value of $N_r = 1$ works well for practical implementations.

***Practical Realization***: To clarify the theoretical approach, a descriptive example will be given how ferns are used to classify images. We assume that the classification algorithm should consist of $M$ ferns of size $S$. During initialization phase for every fern $S$ decision functions, referred to as binary tests, are chosen and put in sequence. Every test is associated to a certain feature, which is picked randomly out of a predefined feature space. In [OCLF10] a simple compare method is used for the binary tests but in fact arbitrary functions, even feature combining ones, can be used. The pixel locations are also picked randomly within a reasonable limit. Ones all binary tests are set they do not change anymore during the rest of the procedure.

A training set consists of several images with their associated ground truth labeling. A fern applies all of its tests to a training sample, where the results of the tests lead to a particular fern value $k$. According to the associated ground truth label, the discrete class distribution of samples leading this particular value $k$ is updated. After all training samples have been processed, for every value $k$ a discrete distribution of classes is given. To realize the definition of equation (2.19), the discrete distributions of the classes are initialized with 1, which assigns at least one training image of every class to every possible value of $k$.

During runtime the unseen image is evaluated by every fern, which means the value $k$ of the fern accounting to the test results for the image is calculated. Every fern outputs its learned distribution for this value and these outputs are combined in the Semi-Naive Bayesian manner described.

### 2.2.2 Segmentation Algorithms

The aim of image segmentation in general is to partition an image into disjoint, meaningful parts. If also a label should be assigned to each part to associate it with logical properties, it is referred to as semantic image segmentation.

In Chapter 2.2.1, classification algorithms are described which can be used to provide a probability distribution over possible labels for each pixel in an image. Based on these distributions a simple labeling can be obtained by assigning the label with the maximum likelihood to each pixel. One drawback of this method is that it assumes independence between all pixels, which leads to highly scattered results.

The segmentation algorithms introduced in this chapter are designed to produce a smooth labeling of a given image, taking local characteristics of the image and the neighborhood of each pixel into account. At first graphical models are described and how they can be used to formulate problems in the field of computer vision. Second, methods to solve such problems, with the focus on the graph cut method, are described.

#### Graphical Models

Graphical models are used to describe the relationships between multiple random variables by means of a graph. They allow us to determine the joint or conditional probability distribution over a set of configurations of the variables in a very effective way. In the field of computer vision many segmentation algorithms are based on such models. Two very common models are the Bayesian network and the Markov random field (MRF). Both are represented by a graph $G = (V, \mathcal{E})$ which consists of nodes $V$ and edges $\mathcal{E}$. Every node $i$ is associated with a random variable $Y_i$ and a certain realization of it is denoted by $Y_i = y_i$. The joint realization over all nodes, also called configuration, is denoted by $Y = y$. In this chapter just discrete values for the random variables are assumed, but the explanations can be straightforwardly extended to the continuous case.

The Bayesian network is a graphical model where the graph is acyclic and directed. For Bayesian networks the probability $p(Y = y)$ can be factorized by following formula:

$$p(Y = y) = \prod_{i \in V} p(y_i | y_{parents(i)}), \tag{2.20}$$

where $p(y_i|y_{parents(i)})$ is a conditional probability distribution, and $parents(i)$ denotes the set of parents of node $i$. In Figure 2.7 a Bayesian network of four elements and the according factorization is given.

The focus of this chapter is put on undirected graphical models, also known as Markov random fields (MRF). The graph $G = (V, \mathcal{E})$ of a MRF consists of nodes $V$ and undirected edges $\mathcal{E}$. The edges indicate a dependency between the random variables of the associated nodes. In Figure 2.8 a MRF with four nodes is shown [NL11].

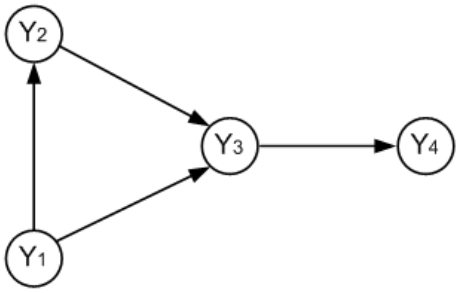



Figure 2.7: A Bayesian network consisting of 4 nodes with following factorization: $p(Y = y) = p(Y_4 = y_4|Y_3 = y_3) \cdot p(Y_3 = y_3|Y_1 = y_1, Y_2 = y_2) \cdot p(Y_2 = y_2|Y_1 = y_1) \cdot p(Y_1 = y_1)$.

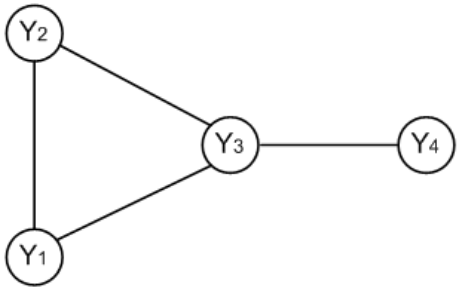



Figure 2.8: A Markov random field consisting of 4 nodes with following factorization: $p(Y = y) = \frac{1}{Z} \cdot \vartheta_{123}(y_1, y_2, y_3) \cdot \vartheta_{34}(y_3, y_4)$.

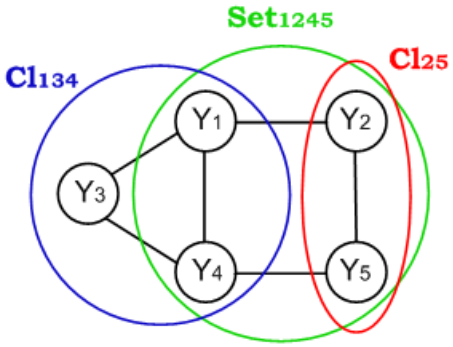



Figure 2.9: Definition of cliques: Two maximal cliques are marked in the graph: $Cl_{134}$ and $Cl_{25}$. The subset formed by the nodes $\{Y_1, Y_2, Y_4, Y_5\}$ is not a clique, because connections between $Y_1$ and $Y_5$, and between $Y_2$ and $Y_4$ are missing. Note: Every connected pair of nodes forms a clique, but in the image just $Cl_{25}$ is given as an example.

The factorization of the probability of a certain realization $p(Y = y)$ of a MRF is determined by help of the graphical concept 'clique'. A clique is defined as a subset of nodes in which every node is connected to each other node (clique property). In terms of probability theory this means that each node is dependent on all other nodes of the clique. A maximal clique is defined as a subset of nodes of the graph, where it is not possible to include any other node of the graph without canceling the clique property. An illustrative example is given in Figure 2.9. The factorization of $p(Y = y)$ for a MRF is then given by:

$$p(Y = y) = \frac{1}{Z} \prod_{cl \in Cliques(G)} \vartheta_{cl}(y_{cl}). \tag{2.21}$$

There, $cl$ denotes a clique, $Cliques(G)$ is the set of all maximal cliques of the graph $G$ and $y_{cl}$ is the specific node configuration of this clique. The factors $\vartheta_{cl}()$ are also called potential functions and model the relationship between the variables (nodes) in the clique $cl$. A factor can be realized by an arbitrary non-negative function. In general the product of the factors will not be correctly normalized, that is why the partition function $Z$ as normalization factor is needed:

$$Z = \sum_{y \in \Upsilon} \prod_{cl \in Cliques(G)} \vartheta_{cl}(y_{cl}). \tag{2.22}$$

The partition function calculates the sum over all possible configurations of $y \in \Upsilon$, where $\Upsilon$ is the entire solution space. In Figure 2.8 a Markov random field with its factorization is given. A more convenient way to explicit denote the factorization is to use a factor graph [NL11, Bel06].

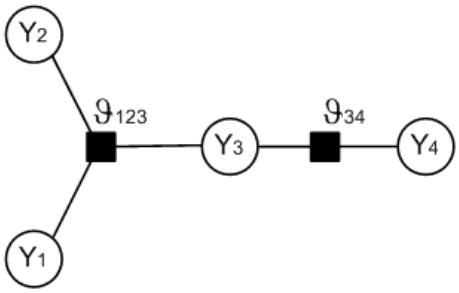



Figure 2.10: A factor graph of the Markov random field in Figure 2.8. The nodes $Y_1, \ldots, Y_4$ are the variable nodes with the factors $\vartheta_{123}, \vartheta_{34}$ of the maximal cliques.

***Factor Graphs:*** A factor graph directly encodes the factorization of the probability function in the graph structure. The factor graph $FG = (V, \mathcal{F}, \mathcal{E})$ consists of nodes $V$ which represent the random variables and nodes $\mathcal{F}$ which represent the factors. The edges $\mathcal{E}$ are undirected and connect always a variable node and a factor node. A factor determines the relation between the adjacent variable nodes. The probability $p(Y = y)$ of a certain configuration $y$ then factorizes by following formula:

$$p(Y = y) = \frac{1}{Z} \prod_{i \in \mathcal{F}} \vartheta_i(y_{\mathcal{N}(i)}). \tag{2.23}$$

There, $\mathcal{N}(i)$ denotes the adjacent variable nodes of the factor node $i$. The partition function $Z$ becomes:

$$Z = \sum_{y \in \Upsilon} \prod_{i \in \mathcal{F}} \vartheta_i(y_{\mathcal{N}(i)}). \tag{2.24}$$

In Figure 2.10 a factor graph over the maximal cliques of the Markov random field in Figure 2.8 is given [NL11, Bel06].

***Energy Minimization:*** To find the best solution for a problem which is defined by a factor graph, we need to determine a configuration $y^* = \text{argmax}_{y \in \Upsilon}\, p(Y = y)$ with the maximum probability. This can be achieved by defining a function, the so called 'energy', which measures the quality of a configuration. To get the desired solution this energy function needs to be minimized. The configuration which leads to the minimum energy has also the maximum probability. Now this relation will be described in detail.

For every factor node $i \in \mathcal{F}$ an energy function $E_i(y_{\mathcal{N}(i)})$ is defined. The energy function of factor node $i$ depends on the configuration $y$ of the adjacent variable nodes $\mathcal{N}(i)$ and provides a quality measure for this configuration. The factors $\vartheta_i$ in equation (2.23) are defined as:

$$\vartheta_i(y_{\mathcal{N}(i)}) = \exp(-E_i(y_{\mathcal{N}(i)})), \tag{2.25}$$

where the exponential representation is called the Boltzmann distribution. The probability $p(Y = y)$ in equation (2.23) becomes [NL11]:

$$\begin{aligned} p(Y = y) &= \frac{1}{Z} \prod_{i \in \mathcal{F}} \vartheta_i(y_{\mathcal{N}(i)}) \\ &= \frac{1}{Z} \prod_{i \in \mathcal{F}} \exp(-E_i(y_{\mathcal{N}(i)})) \\ &= \frac{1}{Z} \exp(-\sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)})). \end{aligned} \tag{2.26}$$

The partition function $Z$ becomes:

$$Z = \sum_{y \in \Upsilon} \exp(-\sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)})). \tag{2.27}$$

Based on equation (2.26) the finding of the configuration $y^* = \operatorname{argmax}_{y \in \Upsilon} p(Y = y)$ with the maximum probability turns into an energy minimization problem [NL11]:

$$\begin{aligned} y^* = \operatorname*{argmax}_{y \in \Upsilon} p(Y = y) &= \operatorname*{argmax}_{y \in \Upsilon} \frac{1}{Z} \exp(-\sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)})) \\ &= \operatorname*{argmax}_{y \in \Upsilon} \exp(-\sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)})) \\ &= \operatorname*{argmax}_{y \in \Upsilon} (-\sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)})) \\ &= \operatorname*{argmin}_{y \in \Upsilon} (\sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)})). \end{aligned} \tag{2.28}$$

The final equation (2.28) states that the finding of the solution with the maximum probability is equivalent to minimizing the sum of the energy functions $E_i(y_{\mathcal{N}(i)})$. The next chapter shows how this relation offers a convenient way to solve image segmentation tasks by means of conditional random fields (CRF). Another important fact is that the partition function $Z$ does not influence the energy minimization of equation (2.28). This is a big advantage, as in practice the explicit calculation of $Z$ is in general not feasible because of the big solution space $\Upsilon$.

***Markov Random Fields used for Image Denoising:*** Markov random fields are undirected graphical models which describe the interaction between random variables. The joint probability of a MRF can be determined by help of a factor graph and appropriate energy functions. If the random variables of the MRF are conditionally dependent on given observations, this can be expressed using conditional random fields.

The Ising model is a basic Markov random field and was originally used by physicists to model the interaction between atom spins. The model turned out to be useful for other binary

problems too and was mentioned in the field of computer vision first in [GG84]. To clarify the relation of factor graphs, energy minimization, Markov random fields and conditional random fields, a simple example for the task of image denoising based on the Ising model is given. We assume a binary image which is corrupted by salt and pepper noise. The graph-structure of the MRF is the pixel grid of the image; every pixel is associated with a node and a 4-neighborhood is assumed. Every maximal clique $cl$ consists of two adjacent pixels (nodes) $i, j$ and the factor of a clique is defined by equation (2.25). The random variables of the nodes are in the binary case 0 and 1 and represent the pixel values of the image. The aim is to find a configuration $Y = y$, which represents the original image the best. As we want to remove salt and pepper noise, we need energy functions $E_i(y_{\mathcal{N}(i)})$ which penalize configurations of cliques where the associated variables of the involved nodes are unequal. A very simple version would be a logical XOR function. Intuitively it could be expected that this model would remove the noise, because single nodes with a value different to the surrounding area would be forced to change their value to achieve a smooth area. The problem hereby is that an image which does not have any transitions of different valued nodes would be associated with the smallest energy and therefore with the maximum probability. The result would be either a totally black or totally white image. To address this problem some conditions need to be added, which leads to the conditional random field [Li95, Bel06].

***Conditional Random Field:*** A conditional random field (CRF) is defined like a Markov random field, with the extension that nodes can be conditioned by an observation $X$. This means the probability distribution $p(Y = y)$ turns into the conditional distribution $p(Y = y|X = x)$, where $X = x$ denotes a certain instance of the observation. Equation (2.23) becomes [NL11]:

$$p(Y = y|X = x) = \frac{1}{Z(x)} \prod_{i \in \mathcal{F}} \vartheta_i(y_{\mathcal{N}(i)}; x_{\mathcal{N}(i)}), \tag{2.29}$$

where $x_{\mathcal{N}(i)}$ denote the adjacent observation nodes of a factor node $i$. The normalizing constant $Z(x)$ is also depending on the observation and becomes [NL11]:

$$Z(x) = \sum_{y \in \Upsilon} \prod_{i \in \mathcal{F}} \vartheta_i(y_{\mathcal{N}(i)}; x_{\mathcal{N}(i)}). \tag{2.30}$$

An illustrative example of a conditional random field is given in Figure 2.11. Every node $Y$ of the previous described Markov random field is connected with one node of the observation. In Figure 2.11 all $X$ nodes are fix valued and represent the observation. This leads to two different cliques in the graph. In a graph with a 4-neigborhood on the $Y$-level and one observation node connected to each $Y$ node, each maximal clique consists of two nodes. Type 1 cliques are formed of one $Y$ node and one $X$ node and model the dependency of the $Y$ nodes on the observation. Type 2 cliques are formed between $Y$ nodes and model the relation of neighborhood pixels. The factor of each clique is represented by factor nodes of the form (2.25).

Concerning the previous example of image denoising, the corrupted image is the observation $X$. The configuration $Y = y$ represents the reconstructed image. To find the solution $Y = y^*$

with the maximum probability we will use equation (2.28). We denote the sum of the energy functions $E_i(y_{\mathcal{N}(i)}; x_{\mathcal{N}(i)})$ with $E(Y = y|X = x)$, which shows that the overall energy depends on the observation. For the two different type of factors two energy functions $U_i(y_i, x_i)$ and $PW_{i,j}(y_i, y_j)$ are defined:

$$E(Y = y|X = x) = \sum_{i \in \mathcal{F}} E_i(y_{\mathcal{N}(i)}; x_{\mathcal{N}(i)}) \tag{2.31}$$

$$E(Y = y|X = x) = \sum_{i=0}^{|Y|} U_i(y_i, x_i) + \sum_{\{i,j\} \in Ne(Y)} PW_{i,j}(y_i, y_j). \tag{2.32}$$

There, $U_i(y_i, x_i)$ is called unary term and defines the energy functions for the factors of cliques of type 1 (see Figure 2.11). These functions model the compatibility of the random variables $y_i$ and the observed variables $x_i$. $PW_{i,j}(y_i, y_j)$ is called pairwise term and models the energy between two adjacent nodes $Y_i$ and $Y_j$. $Ne(Y)$ indicates all present pairs of neighbors of the $Y$-nodes and corresponds to the cliques of type 2 in Figure 2.11. The energy given in equation (2.32) corresponds for the binary case to the Ising model [NL11, Bel06, SM10].

To determine the configuration with the maximum probability $y^*$, which represents the denoised image, equation (2.28) needs to be solved:

$$y^* = \underset{y \in \Upsilon}{\operatorname{argmax}}\, p(Y = y|X = x) = \underset{y \in \Upsilon}{\operatorname{argmin}}\, E(Y = y|X = x). \tag{2.33}$$

***Parameterization:*** Equation (2.32) shows a general form of an energy function that can be represented by a conditional random field. To specify more precisely the unary and pairwise terms and how they interact, learnable parameters can be added. These parameters are presented as a $W$-dimensional parameter vector $w \in \mathbb{R}^W$, and the energy $E(Y = y|X = x)$ in equation (2.32) changes to $E(Y = y|X = x; w)$ to denote the dependence on these parameters. If a training set is available, the parameters of the model can be tuned automatically according to this set. As parameter tuning is not used in this work, the process will not be described in detail. For more information about parameterization see [NL11].

### Graph Cut

In the previous chapter we showed how an energy function can be defined to formulate the common problem of image denoising. In practice many problems in the field of computer vision, like image segmentation or stereo matching as example, can be formulated as energy minimization problems. Formula (2.33) states that minimizing an appropriate energy function leads to a desired solution, which is in general an optimization problem. Solving such optimization problems is in general NP-hard. To find solutions in practice, particular algorithmic properties need to be discarded. These properties include: generality, optimality, worst-case complexity, integrality and determinism. Several algorithms exist to solve problems by giving up one or more of the previous listed properties. For a detailed characterization of

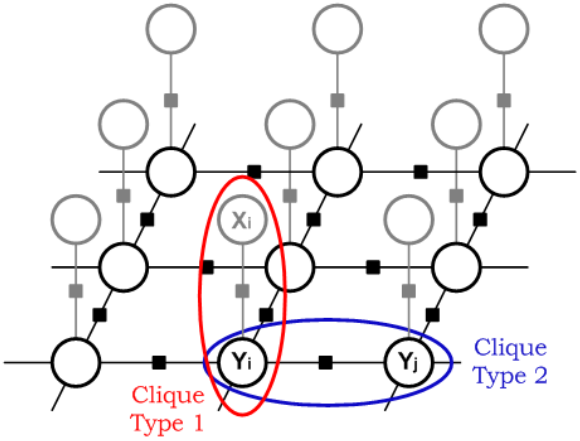



Figure 2.11: A CRF used for image denoising: The corrupted image is represented by the observation $X$. The aim is to find a configuration for the $Y$-nodes, which represents the original image (without noise) the best. Cliques of type 1 define the relation between the given corrupted image and the reconstructed one. Cliques of type 2 define the relation between adjacent pixel.

them with according algorithms see [NL11]. In this work the focus will be put on the graph cut method, which is capable to minimize energies of the form (2.34), but optimality and generality cannot be guaranteed at the same time. In the case of multilabel graph cuts even both properties need to be discarded.

***Binary Graph Cut:*** We assume an optimization task, where the problem can be formulated as the minimization of an energy function. If the generality of this energy function is given up for the restricted class of binary energy functions of the form:

$$E(Y = y|X = x) = \sum_{i=0}^{|Y|} U_i(y_i, x_i) + \sum_{\{i,j\} \in Ne(Y)} PW_{i,j}(y_i, y_j), \tag{2.34}$$

the energy can be minimized globally by use of the binary graph cut method. In equation (2.34) $U_i()$ denotes unary factors and $PW_{i,j}()$ pairwise factors. $Y$ denotes the set of random variables and $Y = y$ is a particular realization of it. $Ne(Y)$ indicates all present pairs of neighbors of the $Y$ nodes. The resulting energy of the random variables $Y$ is depending on a condition $X$ with a certain realization $X = x$. For the energy minimization problem an optimal solution can be found if the unary term satisfies:

$$U_i(y_i, x_i) \geq 0 \quad \forall i \in Y. \tag{2.35}$$

The pairwise term must satisfy:

$$PW_{i,j}(y_i, y_j) = 0, \quad \text{if } y_i = y_j, \tag{2.36}$$
$$PW_{i,j}(y_i, y_j) = PW_{i,j}(y_j, y_i) \geq 0, \quad \text{otherwise.} \tag{2.37}$$

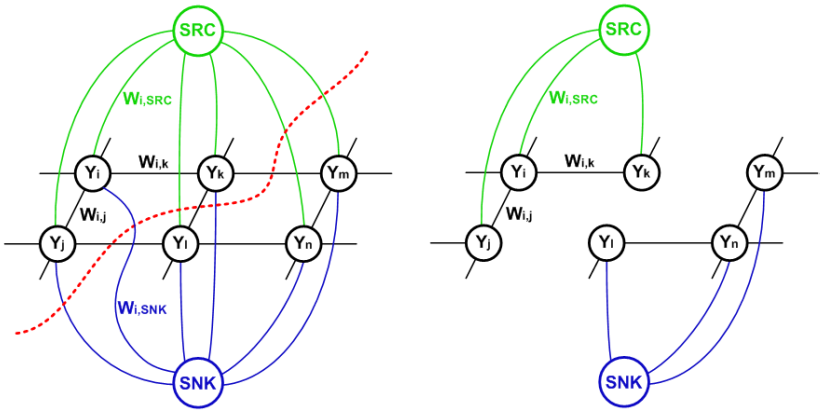
In practice the class of energy functions which fulfill these restrictions can be extended by energy functions which are transformable to appropriate functions. For energy functions with negative unary terms in their original representation, equation (2.35) can be fulfilled by adding a sufficiently large constant to every unary term. This will increase the overall energy, but the configuration which causes the minimum energy does not change. For the restrictions of the pairwise term similar transformations exist. In [KZ04] a characterization of energy functions which can be minimized by graph cuts is given. For a binary energy function of the form (2.34) they show the theorem that the pairwise energy functions must be regular to be representable and solveable by a graph. An energy function of the form (2.34) is called regular if the pairwise term $PW_{i,j}(y_i, y_j)$ satisfies for all $\{i, j\} \in Ne(Y)$:

$$PW_{i,j}(0,0) + PW_{i,j}(1,1) \leq PW_{i,j}(0,1) + PW_{i,j}(1,0). \tag{2.38}$$

In practice this means that the labeling of two adjacent pixels with the same value must result in an equal or lower energy than the labeling with two different values.

To solve the energy minimization problem a graph as shown in Figure 2.12(a) is constructed. The $Y$ nodes represent the random binary variables, which can be labeled with $Y = 0$ or $Y = 1$. For every pairwise factor an edge between the according $Y$ nodes with the edge weight $w_{i,j} = PW_{i,j}(y_i, y_j)$ is inserted. An additional source node $SRC$ and a sink node $SNK$ is added, which both are connected to every $Y$ node. The weights of these edges are defined by the unary term of the energy function. $w_{i,SRC}$ defines the costs for $Y_i = 1$ and $w_{i,SNK}$ for $Y_i = 0$. A cut is defined as the set of edges, that causes the nodes $SRC$ and $SNK$ to be separated if this set is removed. The cost of the cut equals the sum of the weights of the involved edges. A minimum cut separates the source from the sink with the least costs. To determine the configuration of the $Y$ nodes which minimizes the energy, the minimum cut needs to be found. Finding minimum cuts in graphs is a well studied field; a review is given in [BK04] and they also propose a new algorithm for solving the minimum cut problem especially in the field of computer vision. After the minimum cut is found and the according edges removed, the label of each $Y$ node depends whether it is connected to the $SRC$ or the $SNK$ node. Nodes connected to the source get labeled with $Y = 0$, nodes connected to the sink with $Y = 1$. In Figure 2.12(a) the red dashed line shows the minimum cut. Figure 2.12(b) shows the graph after the cut [NL11].

***Multilabel Graph Cut:*** In the previous paragraph we showed that optimization problems which can be formulated as an energy minimization problem can be solved optimally and efficiently, if the energy function is regular and defined on binary variables. In practice many problems exist which requires the use of multilabel variables. For example when segmenting an image it is natural to divide the image into more than two disjoint regions. In [BVZ01] efficient algorithms, namely swap- and expansion-algorithm, are proposed to solve discrete




(a) Subgraph with minimum cut

(b) Resulting subgraph after minimum cut

Figure 2.12: Figure (a) shows the constructed graph with binary $Y$ nodes and additional source and sink node. The dashed line shows the minimum cut and marks all edges which will be removed. Figure (b) shows the graph after the minimum cut is done. Nodes connected to the $SRC$ node are labeled with 0, nodes connected to the $SNK$ node are labeled with 1. Image (a) is a modification from an image in [NL11].

multilabel problems based on graph cuts, but unfortunately these algorithms cannot guarantee an optimal solution. In addition the energy functions used have to fulfill certain criteria, which leads again to a loss of generality. In practice the proposed algorithms in [BVZ01] are very popular, because they are efficient and many practical tasks can be formulated as appropriate energy functions. Also the provided local optimal solutions are often sufficient. When a practical problem is formulated as a discrete optimization problem, it is in general simplified. This leads to a difference between the original and the constructed model, which is also known as model uncertainty. A consequence of this is that the optimal solution of the optimization problem is not necessarily the optimal solution of the original one. Therefore it is sufficient if the provided solution is within a known factor of the optimal solution, which is fulfilled by the algorithms proposed in [BVZ01].

The swap- and the expansion-algorithm [BVZ01] belong to the group of algorithms which perform local search. They find a local optimal solution by iteratively improving the actual solution step by step until no enhancement can be achieved anymore. The search domain changes from step to step but it is restricted to the neighborhood of the actual solution, which leads to a local optimum. Another local search method is Iterated Conditional Modes (ICM), introduced in [Bes86]. ICM is an algorithm which is based on a factor graph and iteratively updates a solution by changing one variable at a time and keeping all others fixed. The method is local optimal with respect to the neighborhood, which means in every step the variable which causes the highest increases in the quality of the solution, is changed. In [BVZ01] changing single variables at a time is referred to as standard move and large

moves refer to a change of a sample of variables at a time. In general optimization methods using standard moves are slower and they tend to produce worst solutions compared to methods using larger moves. When using standard moves the search domain is smaller and the algorithm converges to less dominant local optima [BVZ01].

The swap- and the expansion-algorithm proposed in [BVZ01] use large moves and efficiently solve multilabel problems based on binary graph cuts. The restrictions for the use of them will be described now in detail. Energy functions which can be approximately minimized need to be of the form:

$$E(Y = y|X = x) = \sum_{i=0}^{|Y|} U_i(y_i, x_i) + \sum_{\{i,j\} \in Ne(Y)} PW_{i,j}(y_i, y_j). \tag{2.39}$$

There, $U_i()$ denotes unary factors and $PW_{i,j}()$ pairwise factors. $Y$ denotes the set of random variables and $Y = y$ is a particular realization of it. The variables can be assigned a label in some finite set $\mathcal{L}$. $Ne(Y)$ indicates all present pairs of neighbors of the $Y$ nodes. The resulting energy of the random variables $Y$ is depending on a condition $X$ with a certain realization $X = x$. The unary terms $U_i(y_i, x_i)$ can be arbitrary but nonnegative:

$$U_i(y_i, x_i) \geq 0 \quad \forall i \in Y. \tag{2.40}$$

The pairwise terms must be either a metric or a semi-metric on the space of labels $\mathcal{L}$. $PW_{i,j}(y_i, y_j)$ is called a metric if for any labels $y_i, y_j, y_k \in \mathcal{L}$ it satisfies:

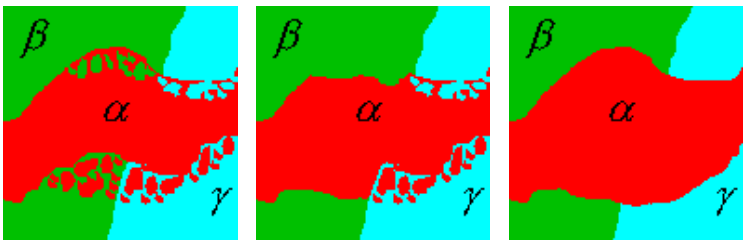
$$PW_{i,j}(y_i, y_j) = 0 \Leftrightarrow y_i = y_j, \tag{2.41}$$

$$PW_{i,j}(y_i, y_j) = PW_{i,j}(y_j, y_i) \geq 0, \tag{2.42}$$

$$PW_{i,j}(y_i, y_j) \leq PW_{i,j}(y_i, y_k) + PW_{i,j}(y_k, y_j). \tag{2.43}$$

If $PW_{i,j}(y_i, y_j)$ just statisfies (2.41) and (2.42) it is called a semi-metric. Both algorithms iteratively improve their solution with respect to a local search domain.

*$\alpha - \beta$ swap move:* If the unary term of the energy function satisfies (2.40) and the pairwise term is a semi-metric, the swap-algorithm can be used to approximately minimize the energy. We assume a set of variables $Y$ and an actual labeling (certain configuration) of them. A subset $A \subset Y$ contains nodes labeled $\alpha$ or $\beta$ and depends on the actual search domain. When an $\alpha - \beta$ swap move is applied, the variables of set $A$ are free to change their label either to $\alpha$ or $\beta$. Given an energy function and initial labeling of $Y$, the swap algorithm minimizes the overall energy in every cycle by applying the optimal $\alpha - \beta$ swap with respect to the actual search domain. If the energy cannot be decreased any further by any $\alpha - \beta$ swap, the algorithm converges. In every cycle all pair of labels in $\mathcal{L}$ are checked, but every $\alpha - \beta$ swap includes just two labels, whereby these sub-problems can be efficiently solved by the binary graph cut method introduced earlier. In Figure 2.13(b) an illustrative example of an $\alpha - \beta$ swap is shown [BVZ01, NL11].




(a) initial labeling (b) $\alpha - \beta$ swap (c) $\alpha$ expansion

Figure 2.13: Figure (a) shows a fictive initial labeling. Figure (b) shows the result after a swap move was applied. Figure (c) shows the result after an expansion move. Images taken from [BVZ01].

$\alpha-$*expansion move:* If the pairwise energy term $PW_{i,j}()$ of (2.39) is further restricted to be a metric and the unary term satisfies (2.40), the expansion-algorithm can be used to approximately minimize the energy. For the costs of a limited domain of energy functions the expansion algorithm guarantees a solution within a known factor of the global optimal solution. The algorithm minimizes the energy in every cycle by $\alpha$-expansion moves. Given a set of variables $Y$ and an actual labeling of them, an $\alpha$-expansion move refers to changing the labels of all variables of a subset $A \subset Y$ to $\alpha$. The variables of set $A$ can be labeled with any label in $\mathcal{L}$. The algorithm chooses in every cycle the optimal $\alpha-$expansion move with respect to the actual search domain to decrease the overall energy. Given the subset $A \subset Y$, all variables in $A$ either keep their actual label or change it to $\alpha$. This binary problem can be solved again efficiently via the binary graph cut method. To determine the optimal $\alpha-$expansion move the algorithm checks the moves for all labels in $\mathcal{L}$. Figure 2.13(c) shows an illustrative example of an $\alpha-$ expansion move. Although it is a local search method, where every move is restricted to a certain search domain, the final solution is guaranteed to be within a known factor of the optimal solution. For further details see [BVZ01].

#### Discrete and Continuous Potts-Model:

The Potts-model [Pot52] is a generalization of the binary state Ising model (2.32), introduced in Chapter 2.2.2, to multiple states. If the energy function in (2.39) is based on the Potts model, it becomes:

$$E_P(Y = y|X = x) = \sum_{i=0}^{|Y|} U_i(y_i, x_i) + \sum_{\{i,j\} \in Ne(Y)} PW_{i,j} \cdot \delta(y_i, y_j). \tag{2.44}$$

There, $E_P()$ denotes that it is an energy function based on the Potts model. The unary factors $U_i()$ do not change, but the pairwise factors split into two parts. $PW_{i,j}$ denotes a spatial weight factor, in practice often depending on the gradient of the image. $\delta()$ is defined as:

$$\delta(y_i, y_j) = \begin{cases} 0, & \text{if } y_i = y_j \\ 1, & \text{if } y_i \neq y_j \end{cases} \tag{2.45}$$

The definition of the pairwise term implies that discontinuities between any pair of labels are penalized equally. It also implies that the energy in (2.44) is highly influenced by the boundary length between regions of different labels. A minimization of (2.44) results practically in an ensemble of piecewise constant label regions with shortest possible boundary lengths with respect to all factors of the energy function. In case of image segmentation, $PW_{i,j}$ is often depending on the gradient of the image and forces the boundaries of the regions to be located on, or at least near, edges in the image. The discrete version of it can be solved efficiently by the $\alpha$-expansion moves introduced in [BVZ01]. If a 4-neighborhood is assumed among the pixels, the solution suffers from metrication errors [KSK+08]. Practically, region boundaries tend to proceed along the underlying grid structure. To improve the solution a higher connected neighborhood can be assumed, but this results in a major increase of needed memory and therefore makes this method practically unfeasible [KSK+08].

In [PCCB09] a convex relaxation of the Potts model which is formulated in a spatially continuous setting is proposed. The spatially continuous energy has the form:

$$E_{cont} = \sum_{l=0}^{k} Per(R_l; \Omega) + \sum_{l=0}^{k} \int_{R_l} f_l(y) \, \mathrm{d}y, \tag{2.46}$$

$$\text{such that} \bigcup_{l=0}^{k} R_l = \Omega, \; R_a \cap R_b = \emptyset \; \forall \, a \neq b. \tag{2.47}$$

There, $\Omega$ is a domain, for example an image, which should be partitioned into $k+1$ pairwise disjoint regions $R$. A region is defined as an area of connected elements with the same label. $Per(R_l; \Omega)$ denotes the perimeter of region $R_l$ in the domain $\Omega$. The integral $\int_{R_l} f_l(y) \, \mathrm{d}y$ is used to model the unary factors. It denotes the entirety of non-negative weight functions $f_l()$ of a region $R_l$. The weight function for every element is depending on the assigned label. In [PCCB09] a primal dual algorithm is offered which can be paralyzed efficiently and therefore provides a major speedup if appropriate hardware is used. The algorithm is based on a Total Variation functional and therefore does not suffer from the metrication errors like the discrete model. For more details see [PCCB09]. In Chapter 4 the performance of the ferns, described in Chapter 2.2.1, combined with both the discrete and the continuous Potts model is compared.

## 2.3 Stochastic

The aim of this work is to provide a tool for specialists in the field of lithium-ion cells which creates a statistical volume element out of one given image of a lithium-ion electrode. Such electrodes consist of different materials which form particles of arbitrary shapes. For more

details about the function of lithium-ion cells see Chapter 2.1.1. To reconstruct a 3D volume out of a single image some approximations about the shape of the particles need to be made. We assume the particles are ellipsoids defined by their three semi-principal axes, their location and the yaw angle which represents their rotation about the z-coordinate. After the different materials in the image are segmented with help of the methods described in Chapter 2.2, the found particles are approximated with ellipses. The goal is to reconstruct ellipsoids based on this distribution of ellipses.

This chapter reviews how in [Wal10] circles are reconstructed (radii and relative frequencies) from a given distribution of cut lengths. In chapter 3.2.1 these methods are expanded to reconstruct ellipsoids out of a distribution of ellipses.

### 2.3.1 Reconstruction of Circles from Cut Length Distributions

We assume a 2D binary image consisting of circles with different radii. A one dimensional cut is made through the image and all cut lengths are counted. The resulting distribution is interpreted as the sum of weighted cut length density functions and the relative frequencies and radii of the original circles are reconstructed. In this section the main steps are presented, for more details see [Wal10]. First the derivation of the cut length density function is explained. Then it is shown how the density function can be used to reconstruct the circles. An implementation of the proposed algorithm in [Wal10] has been realised and the achieved results are illustrated.

#### Derivation of Cut Length Probability Density

A unit circle in the Cartesian coordinate system is defined by:

$$x^2 + y^2 = 1. \tag{2.48}$$

The length of a cut in the y-coordinate direction is given by:

$$cut\_length_y(x) = 2 \cdot \sqrt{1 - x^2}, \tag{2.49}$$

and varies in the range from 0.0 to 2.0. Therein, $x$ denotes the position of the cut on the x-coordinate. The possible cut positions are uniformly distributed over the x-coordinate. These facts are illustrated in Figure 2.14. The upper diagram shows the cut lengths of a unit circle in y-direction, defined by equation (2.49), depending on the x-position. In the lower diagram the uniform distribution density $f_X(x)$ for the cut position is given.

For easier formulation we introduce a random variable $Y$ which represents the cut length. A certain value for it is denoted by $cut\_length_y$. The cut length distribution function $F_Y(cut\_length_y)$ (in [Wal10] denoted as 'Kreisverteilungsfunktion') is defined in the range from $cut\_length_y = 0.0$ to 2.0 (unit circle, radius = 1.0) and denotes the probability that

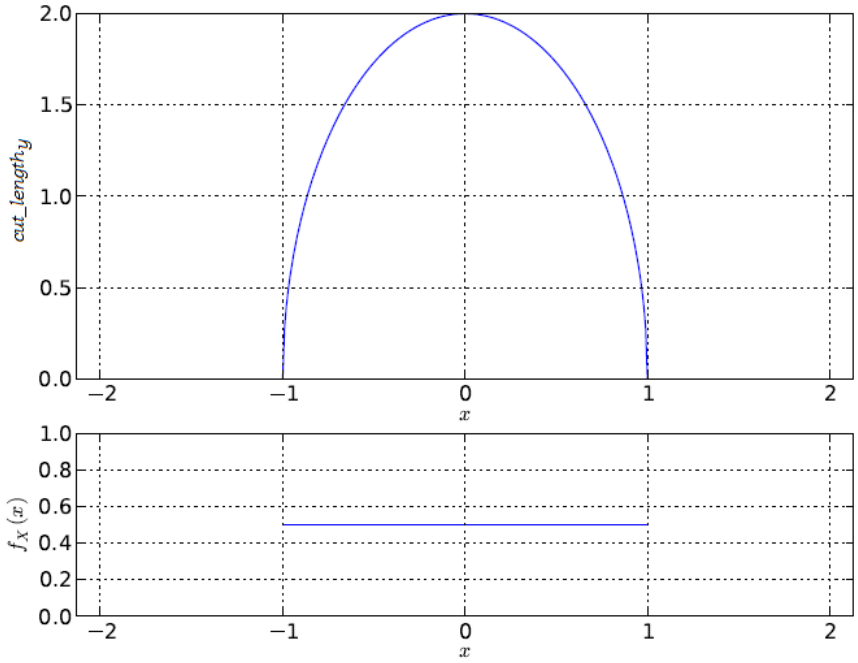



Figure 2.14: The upper diagram shows the resulting cut length from a cut performed normal on the x-axis; depending on the cut position on the x-axis. The lower diagram shows the uniform distribution density for the cut position. Image taken from [Wal10].

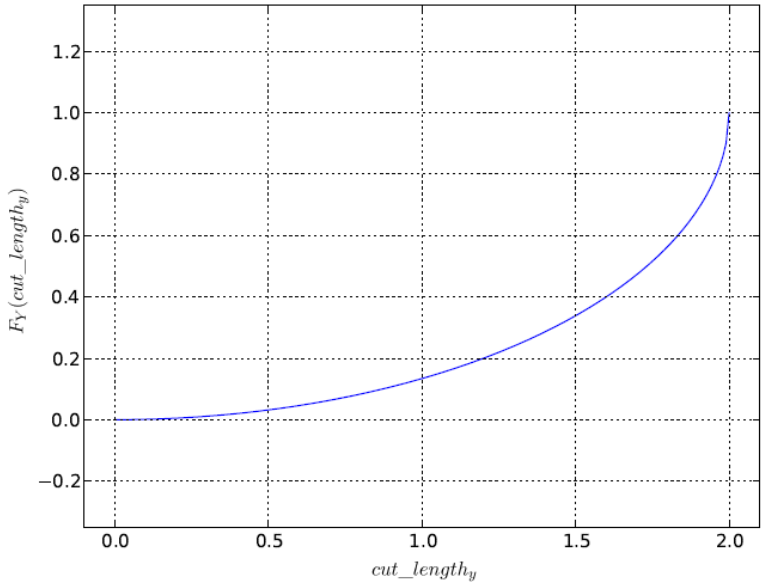



Figure 2.15: The diagram shows the cut length distribution function of a unit circle. The function value $F_Y(cut\_length_y)$ denotes the probability that a cut of the circle results in a cut length of $cut\_length_y$ or less. Image taken from [Wal10].

a uniformly distributed cut performed on the original circle has a length of $cut\_length_y$ or less:

$$F_Y(cut\_length_y) = P(Y \leq cut\_length_y) = 1 - P(Y > cut\_length_y). \tag{2.50}$$

Given a certain cut length, the straightforward reformulation of equation (2.49) results in two possible cut positions $x_1, x_2$ on the x-axis (upper diagram of Figure 2.14). $P(Y > cut\_length_y)$ is then defined as the area under $f_X(x)$ between $x_1$ and $x_2$ (lower diagram of Figure 2.14):

$$F_Y(cut\_length_y) = 1 - P(Y > cut\_length_y) \tag{2.51}$$

$$F_Y(cut\_length_y) = 1 - \int_{x_1}^{x_2} f_X(x)\,\mathrm{d}x \tag{2.52}$$

$$F_Y(cut\_length_y) = 1 - \sqrt{1 - \left(\frac{cut\_length_y}{2}\right)^2} \tag{2.53}$$

The graph of $F_Y(cut\_length_y)$ is shown in Figure 2.15.

The cut length density function is defined as the derivative of the cut length distribution function $F_Y(cut\_length_y)$. As for this work the cut lengths are in a discrete form, the density

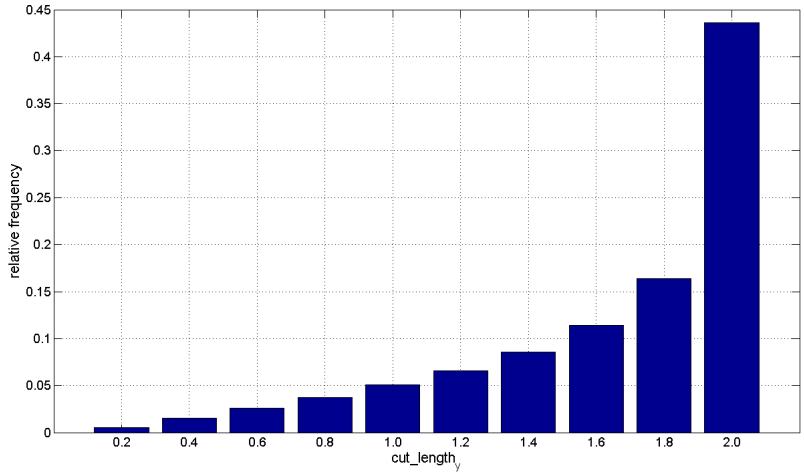



Figure 2.16: A discrete cut length density function of the unit circle with 10 bins.

function also needs to be discretised. Figure 2.16 shows a discrete cut length density function of the unit circle with 10 bins.

The extension to the general case of a circle with radius $r$ is straightforward and equation (2.53) becomes:

$$F_Y(cut\_length_y) = 1 - \sqrt{1 - \left(\frac{cut\_length_y}{2 \cdot r}\right)^2}. \tag{2.54}$$

### Reconstruction of Circles

We assume a binary image consisting of circles of different size. A one dimensional cut through the image gives various cut lengths which are binned into a histogram. This discrete distribution can be interpreted as a weighted mixture of the cut length densities of the original circles of the image. For the reconstruction algorithm a help function needs to be defined which calculates the discrete cut length density for a given radius. Algorithm 1 shows the help function and algorithm 2 the reconstruction algorithm in pseudocode.

**Algorithm 1** Generating the cut length density

**Function:** generate_cut_length_density(radius r)
**Description:** Generates the discrete cut length density based on equation (2.54) for a circle of radius r.
**Parameter:** r: radius of circle
**Returns:** vector representing the cut length density (a plot of such a vector is shown in Figure 2.16)

Description of algorithm 2 - reconstruction of circles: The indices of the mixture vector $mv$ represent the discretised cut lengths. Starting from the longest cut it is checked for every cut

**Algorithm 2** Reconstruct circles from a cut length distribution

**Variables**:
Input: $mv$: mixture vector; representing the histogram of cut lengths
Output: $rel\_f$: vector containing the relative frequencies of the reconstructed circles
$rec\_r$: radii of the reconstructed circles

**Algorithm:**
**for** $cut\_len = mv.end$ to 1 **do**
  **if** $mv(cut\_len) > threshold$ **then**
    - determine density of cut length with radius $cut\_len/2$:
      $act\_dens$ = generate_cut_length_density($cut\_len/2$)
    - save relative frequency $f = mv(cut\_len)/act\_dens.end$ in $rel\_f$
    - save $cut\_len/2$ in $rec\_r$
    - remove reconstructed density: $mv = mv - f \cdot act\_dens$
  **end if**
**end for**
- normalize relative frequencies to: $sum(rel\_f) = 1$

length if its frequency is higher than a certain threshold. The threshold affects the amount of the reconstructed circles and is adjustable by the user. If the frequency of a cut length *cut_len* is significant enough, a circle with radius half of the cut length is found. The according cut length density is calculated via the help function *generate_cut_length_density(r)* and saved as vector *act_dens*. In Figure 2.16 the plot of such a vector is shown. To determine the relative frequency of the found circle, the frequency of the actual cut length *cut_len* in the mixture vector *mv* needs to be divided by the discrete density value indexed by *cut_len* (vector *act_dens*). After the relative frequency and the radius of the reconstructed circle are saved, the according cut length density of the reconstructed circle needs to be removed from the mixture vector weighted by the relative frequency. The last step is the normalization of the relative frequencies.

The algorithm was implemented to test its functionality. For the experiment 2000 circles with radii of $9, 24, 25$ and 30 with different relative frequencies were cut at a random position. The resulting histogram of the cut lengths is shown in Figure 2.17. Given this distribution the algorithm was executed and the results are listed in Table 2.1. The algorithm was capable to reconstruct all radii present in the original distribution. The reconstructed relative frequencies are within a small range around the original ones. As this experiment should just proof that the algorithm is working, no further investigation of the error rate was done. In Chapter 3.2.1 this method will be expanded to reconstruct ellipsoids from a distribution of ellipses.

## 2.4 Geometry Reconstruction

Based on the approximated ellipses of the segmentation result, ellipsoids are derived via the algorithm described in 3.2.1. Each ellipsoid is defined via its size and relative frequency, but

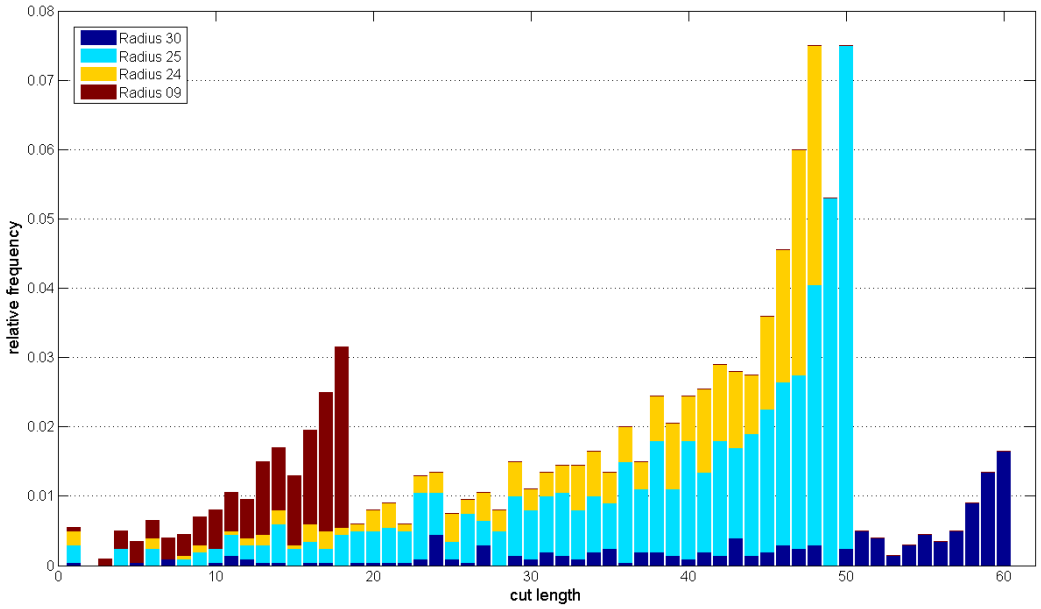



Figure 2.17: Histogram of cut lengths: Circles with radii 9, 24, 25 and 30 were cut at random positions. The colors indicate the ratio of every group of circles for the respective bars.

| | Radius | Relative Frequency |
|---|---|---|
| Original: | 9 | 0,125 |
| | 24 | 0,250 |
| | 25 | 0,500 |
| | 30 | 0,125 |
| Reconstructed: | 9 | 0,106 |
| | 24 | 0,257 |
| | 25 | 0,509 |
| | 30 | 0,128 |

Table 2.1: Example of radii reconstruction: The table shows the radii and the according relative frequencies of the original and the reconstructed distribution. The unit of the radii is irrelevant, as it depends on the interpretation of the user.

to model lithium-ion electrodes a discrete 3D volume is needed. This chapter describes how such ellipsoids are arranged in a certain volume element based on [Wal10].

### 2.4.1 Absolute Frequency

The statistical analysis described in 3.2.1 gives the sizes and relative frequencies of ellipsoids, which should be placed randomly in a 3D volume. This 3D volume is realized as a cube and the user defines its size. The relative frequency of an ellipsoid is denoted by $h$; its size is defined by its semi-principal axes $a, b, c$. The volume $V_{ell}$ of an ellipsoid is calculated via:

$$V_{ell} = \frac{4}{3} \cdot \pi \cdot a \cdot b \cdot c. \tag{2.55}$$

The user defined overall volume size is denoted by $V_{all}$. The manufacturers of the lithium-ion cells provide the relative frequencies of the different materials in the cells. With the relative frequency $f_{active}$ of the active material, the overall volume $V_{AM}$ of active materials is calculated via:

$$V_{AM} = f_{active} \cdot V_{all}. \tag{2.56}$$

The sum of all $n$ ellipsoid volumes must be equal to the overall active material volume:

$$V_{AM} = \sum_{i=1}^{n} H_i \cdot V_{ell_i}. \tag{2.57}$$

There, $V_{ell}$ is calculated via equation (2.55) and $H$ denotes the absolute frequency of a certain ellipsoid size. The absolute frequency $H$ can be calculated by multiplying the relative frequency $h$ with a factor $k$:

$$H = k \cdot h. \tag{2.58}$$

Equation (2.57) becomes:

$$V_{AM} = k \cdot \sum_{i=1}^{n} h_i \cdot V_{ell_i}. \tag{2.59}$$

The factor $k$ is then determined by:

$$k = \frac{V_{AM}}{\sum_{i=1}^{n} h_i \cdot V_{ell_i}}, \tag{2.60}$$

and the absolute frequencies can be calculated for every ellipsoid by equation (2.58) [Wal10].

### 2.4.2 Arrange Ellipsoids in Volume

Chapter 2.4.1 showed how the absolute frequencies of ellipsoids for a given volume size are derived from their relative frequencies. These ellipsoids need to be placed randomly in the volume with small overlaps. For the special purpose of reconstructing the lithium-ion electrode geometry it is important that all reconstructed particles are connected. Beside the fact that the simulation tools which are used to model the cell need a connected structure, the overlapping ellipsoids create arbitrary shaped particles instead of separated perfect ellipsoids. For arranging the ellipsoids an algorithm introduced in [Wal10] is used.

We assume a user defined cubic volume and a list of ellipsoids which should be placed in this volume. The ellipsoids where determined with the algorithms described in this work and are defined by their size (semi-principal axes lengths) and a rotation angle around the y-coordinate of a Cartesian coordinate system. For further details about how the rotation angle is determined see chapter 4.3. To create the discrete 3D structure all ellipsoids are inserted in the volume and rearranged to achieve just small overlaps.

An ellipsoid in canonical form in the Cartesian coordinate system is defined by:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1. \tag{2.61}$$

There, $a, b, c$ are the semi-principal axes which align with the coordinate directions and the ellipsoid is centered at the origin. A line through the origin can be defined by a direction vector $\vec{d}$ (defined by $d_x, d_y$ and $d_z$) multiplied by a scalar:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \lambda \cdot \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} \tag{2.62}$$

The intersection points of the line with the ellipsoid are determined by combining equations (2.61) and (2.62) and solving it for the parameter $\lambda$:

$$\frac{\lambda^2 d_x^2}{a^2} + \frac{\lambda^2 d_y^2}{b^2} + \frac{\lambda^2 d_z^2}{c^2} = 1 \tag{2.63}$$

$$\lambda_{1,2} = \pm \sqrt{\frac{1}{\frac{d_x^2}{a^2} + \frac{d_y^2}{b^2} + \frac{d_z^2}{c^2}}} \tag{2.64}$$

The intersections points $\vec{p_1}$ and $\vec{p_2}$ are then determined by:

$$\vec{p_1} = \lambda_1 \cdot \vec{d}, \tag{2.65}$$

$$\vec{p_2} = \lambda_2 \cdot \vec{d}. \tag{2.66}$$

The lengths of these vectors are the distances from the origin to the hull of the ellipsoid in direction of $\vec{d}$, respectively -$\vec{d}$. As the ellipsoid is in canonical form and the line crosses the origin, both vectors have the same length. This method of determining the dimension of an ellipsoid in a certain direction can also be used if the ellipsoid is rotated but still centered at the origin. In this case the ellipsoid is assumed to be in canonical form and the direction vector $\vec{d}$, and respectively the according line, is rotated in the opposite direction. Then the dimension of the rotated ellipsoid in the direction of $\vec{d}$ can again be determined with the described method [Wal10].

We assume a given cubic volume and a list of ellipsoids, which are defined by their sizes and rotations. At first the ellipsoids are placed randomly in the volume, without checking for overlaps. Then all the ellipsoids are rearranged within the volume in such a way that the overlapping volume is small compared to the ellipsoid volume. The pseudocode of the used algorithm is given in algorithm 3 [Wal10].

Description of algorithm 3 - rearranging ellipsoids: The input of the algorithm is a list of ellipsoids $l_{ell}$ which are randomly placed within a defined volume size $V_{dim}$. At first a list of vectors $l_{move}$ is defined, which holds for every ellipsoid an according movement vector. Then every ellipsoid in $l_{ell}$ is checked against all subsequent ellipsoids in the list. For every pair of ellipsoids it is checked if the distance $dist_c$ between the two centers of these ellipsoids is smaller than the sum $dist_{req}$ of the dimensions of these ellipsoids in direction of the opposite center. If this is the case, the ellipsoids need to be moved apart. The moving distance is split among the two ellipsoids inversely proportional to their volume. This will cause small ellipsoids which are within bigger ones to move further and bigger ones to tend to stay in place, except they intersect with an ellipsoid of their size. All movements are summed up in the list of movement vectors $l_{move}$. After a whole cycle the ellipsoids are moved according to the determined vectors, if the resulting position is still within the fixed volume size. The algorithm is repeated for $number\_of\_runs$ times, lowering the overlapped volume within every cycle.

The algorithm is capable of separating two spheres, but if it is applied to rotated ellipsoids configurations exist where the ellipsoids will still overlap after the movement step. In such a case the overlapping volume is small compared to the ellipsoid volumes which results in an arbitrary shaped form where the original ellipsoids are still existent. For reconstructing the lithium-ion cell microstructure this behavior is desirable, that is why this algorithm was chosen to create the discrete volume. An illustrative example for 2D shapes is given in Figure 2.18.

**Algorithm 3** Rearranging ellipsoids in 3D volume

**Variables**:
Input: $V_{dim}$: size (x, y, z direction) of cubic volume
$l_{ell}$: list of ellipsoids (size, position and rotation)
$number\_of\_runs$: determines how often the algorithm is repeated
Output: $l_{ell}$: new positions of ellipsoids in the volume

**Algorithm**:
- create list $l_{move}$ of movement vectors of size($l_{ell}$)
**for** runs = 1 to $number\_of\_runs$ **do**
  **for** every ellipsoid $e_1$ in $l_{ell}$ **do**
    $\vec{c_1}$ = center of $e_1$
    **for** every ellipsoid $e_2$ in $l_{ell}$, where index($e_2$) > index($e_1$) **do**
      - $\vec{c_2}$ = center of $e_2$
      - determine $dist_c$ = distance between $\vec{c_1}$ and $\vec{c_2}$
      - determine direction vector $\vec{d} = \vec{c_2} - \vec{c_1}$
      - determine dimension of $e_1$ from $\vec{c_1}$ in direction of $\vec{d}$ as $dist_1$
      - determine dimension of $e_2$ from $\vec{c_2}$ in direction of -$\vec{d}$ as $dist_2$
      - determine required distance between centers: $dist_{req} = dist_1 + dist_2$
      **if** $dist_c < dist_{req}$ **then**
        - $dist_{move} = dist_{req} - dist_c$
        - split $dist_{move}$ in $dist_{m1}$ and $dist_{m2}$ inversely proportional to the volumes of $e_1$ and $e_2$
        - scale $\vec{d}$ to the length $dist_{m1}$ and add it to the according movement vector of $e_1$ in $l_{move}$
        - scale -$\vec{d}$ to the length $dist_{m2}$ and add it to the according movement vector of $e_2$ in $l_{move}$
      **end if**
    **end for**
  **end for**
  **for** every position $\overrightarrow{pos}$ in $l_{ell}$ **do**
    **if** $\overrightarrow{pos}$ + according movement vector in $l_{move}$ is inside of $V_{dim}$ **then**
      - move $e_1$ according to movement vector in $l_{move}$
    **end if**
  **end for**
**end for**

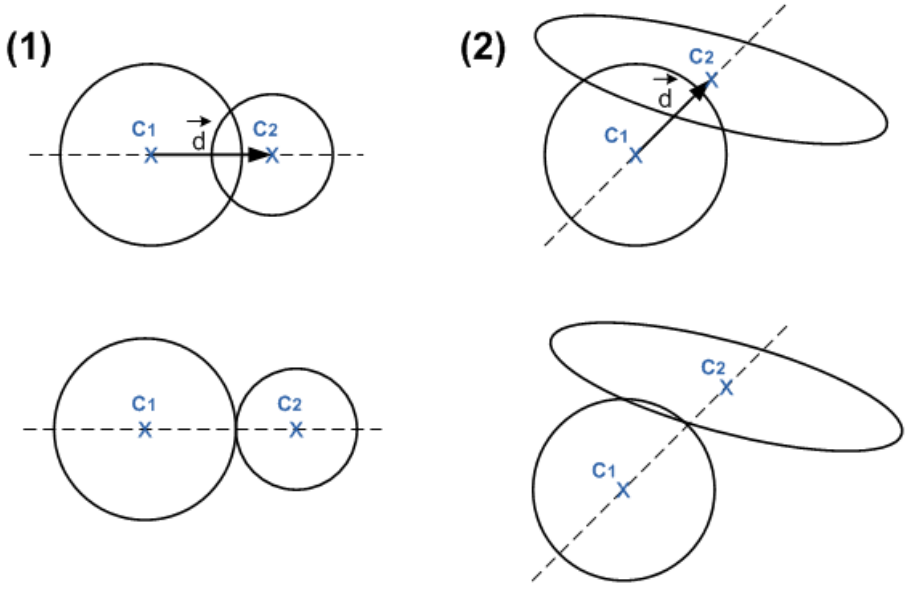



Figure 2.18: Illustration of the rearranging algorithm for a 2D example: $C_1$ and $C_2$ denotes the centers of the shapes. $\vec{d}$ is the direction vector. Case 1 shows the separation of 2 circles. They will be moved on the connection line (dashed) of the two centers, which results in two non overlapping circles. Case 2 shows the result of the algorithm applied to a circle and a rotated ellipse. After the movement step the two shapes show a small overlapping area, but both original shapes are recognizable. The algorithm moved the shapes in such a way that they do not overlap on the connection line.

# 3 Problem Specific Work and Modifications

This chapter shows implementation details, modifications and expansions of the related work. At first the whole system is described to give a proper overview about the included work steps. Then the sub sections are described in more detail.

The aim of this work is to provide a tool for specialists in the field of lithium-ion cells which creates a statistical volume element out of one given image of the inner structure of a lithium-ion electrode. Such electrodes consist of different materials which form particles of arbitrary shapes. For more details about the function of lithium-ion cells see Chapter 2.1.1. At first the different materials of the electrode need to be classified. For this the concept of ferns, described in Chapter 2.2.1, is used. In chapter 3.1.1 the modifications according to the original version of ferns proposed in [OCLF10] are described. The ferns provide a probability distribution over possible labels for each pixel in the image. This result is used as the base for the segmentation via the graph cut algorithms proposed in [BVZ01, BK04, KZ04]. To reconstruct a 3D volume out of a single image some approximations about the shape of the particles need to be made. We assume the particles are ellipsoids, defined by their three semi-principal axes $a, b, c$, their location and the yaw angle which represents their rotation about the z-coordinate. We also assume that the semi-principal axis $c$ equals the longest of the remaining axes. A horizontal cut is done through the 3D volume which results in an image showing ellipses of different size and rotations. As the particles are in fact no ellipsoids, the segmented image does not show ellipses. The arbitrary shaped segments need to be approximated by ellipses. The algorithm used for this purpose is described in chapter 3.1.2. These ellipses are defined by their major- and minor axis, their rotation in the image and their location; but the location is not needed for further processing. Chapter 2.3.1 of the related work section shows how circles can be reconstructed (radii and relative frequencies) out of a given cut length distribution. In Chapter 3.2.1 the described methods of 2.3.1 are expanded to reconstruct ellipsoids out of the given distribution of the approximated ellipses. The last step is to fill a 3D volume with the determined ellipsoids. The used method for this is described in 2.4.2. Chapter 4.3 shows all intermediate results of the reconstruction work flow.

## 3.1 Computer Vision

### 3.1.1 Fern Implementation

To realize the classification task the concept of ferns was chosen. The user sets some seed points which should mark the specific characteristic of the according material. Based on

these scribbles the ferns are trained and for the whole image a probability distribution over all possible labels is determined. The theory of ferns is given in Chapter 2.2.1 and the implementation follows these descriptions. In [OCLF10] ferns are proposed as a new method for keypoint recognition. In this work we solve a classification task based on them. To get reasonable results some modifications need to be realized. In the following part of this paragraph just these modifications are described.

#### General System

The classes to segment are given by the different materials present in the image. For every seed point a patch is extracted around it which represents an instance of the certain class. After the ferns are trained on these instances, a patch is extracted for every pixel to classify the rest of the image. Border-pixels at which no full patch can be extracted are not classified. The subsequent segmentation based on the Potts model can handle unclassified pixels as well. The final system includes 30 ferns of size 10 and uses a patch of size $7 \times 7$. In Chapter 4.1 the experimental results are listed which led to these settings.

#### Feature Channels

In [OCLF10] it is proposed that just the gray level intensity of an image is sufficient as feature channel to obtain robust keypoint recognition. Our experiments showed that for image classification better results can be obtained if more feature channels are used. The scanning-electron-microscopy images, representing the focus of this work, are gray scaled and the materials differ mainly in their structure. Therefore a wider range of feature channels is investigated. The investigated feature channels include: the gray value intensity, the first and second order derivatives in both coordinate directions and histogram of oriented gradient (HOG) features with 9 bins. For a detailed description see Table 3.1. These features have been filtered with a $3 \times 3$ minimum- and a $3 \times 3$ maximum-filter, which leads to a total of 30 feature channels. In Figure 3.1 illustrative examples of these feature channels are given. For better legibility just 8 channels are displayed and the images are contrast enhanced. The images show the original patch and the magnitudes of the first order derivatives and of 5 HOG-features, which were randomly selected. All images show the same patch of a C6-electrode sample image. To determine the best of them for the special purpose of material segmentation of lithium-ion electrodes, at first the performance of every single feature channel on the test images was determined. Based on these results we combined certain feature channels and compared the performance among them. The experimental results are listed in 4.1.1. Our findings showed that the HOG-like features do not offer an increase in performance compared to the derivatives. The reason for this is that the different material structure does not show significant oriented gradients (see Figure 3.1). The vertical artifacts in the images originate from the focus ion beam milling method which is used to slice the electrodes. Because of these the HOG-features with bin-centers close to the x-coordinate direction ($+20°$, $0°$ and $-20°$) perform even significant worse compared to the others. For detail results see Chapter 4.1.1.

| Acronym | Meaning |
|---|---|
| I | Gray value intensity |
| Ix | Magnitude of first order derivative in direction of x-coordinate |
| Iy | Magnitude of first order derivative in direction of y-coordinate |
| Ixx | Magnitude of second order derivative in direction of x-coordinate |
| Iyy | Magnitude of second order derivative in direction of y-coordinate |
| Ixy | Magnitude of gradient |
| | Bins of histogram of oriented gradient: |
| HOG1 | Bin-center: 80° |
| HOG2 | Bin-center: 60° |
| HOG3 | Bin-center: 40° |
| HOG4 | Bin-center: 20° |
| HOG5 | Bin-center: 0° |
| HOG6 | Bin-center: -20° |
| HOG7 | Bin-center: -40° |
| HOG8 | Bin-center: -60° |
| HOG9 | Bin-center: -80° |

Table 3.1: Description of feature channels and acronyms used in Figure 4.2.

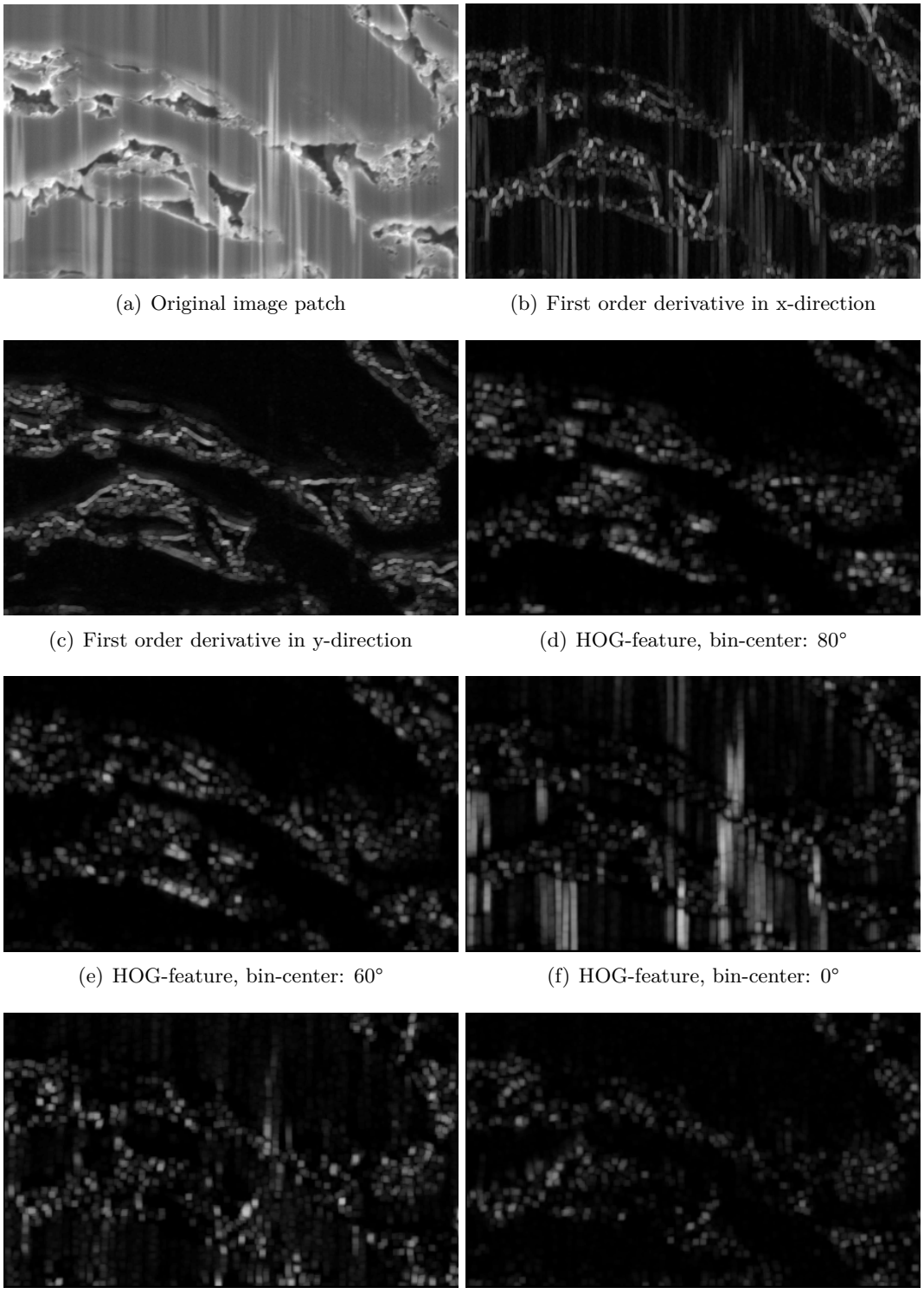
The combination of the derivatives to determine the smoothness of the cutting surface, the gradient, and the gray value intensity lead to the best results. The correlations between them are modeled implicitly by the ferns. Chapter 4.1.3 shows that these feature channels are also suitable for classification tasks performed on natural images. With this settings and a subsequent segmentation based on graph cuts, the results obtained are within the range of other modern segmentation methods, like proposed in [San10]. For detailed results see Chapter 4.1.3.

#### Binary Tests

Ferns use binary tests which are combined in a fixed sequence to form a vector. These tests are applied to the extracted patches and the resulting binary numbers characterize each patch. Arbitrary tests can be used as long as the output is binary, but for performance reasons they should be simple and efficient to implement. In the original version of [OCLF10] just one kind of binary test $f$ was used, a compare of two pixel values:

$$f = \begin{cases} 1, & \text{if } Int(\boldsymbol{p}_1) < Int(\boldsymbol{p}_2) \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

There, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are two randomly chosen pixel locations within each patch. $Int(\boldsymbol{p}_1)$ represents the intensity value of the location $\boldsymbol{p}_1$. Table 4.2 of the experimental results section shows that especially for the NCA lithium-ion electrodes the results obtained by using this



(a) Original image patch

(b) First order derivative in x-direction

(c) First order derivative in y-direction

(d) HOG-feature, bin-center: 80°

(e) HOG-feature, bin-center: 60°

(f) HOG-feature, bin-center: 0°

(g) HOG-feature, bin-center: -20°

(h) HOG-feature, bin-center: -40°

Figure 3.1: Samples of $3 \times 3$ maximum filtered feature patches for a C6-electrode. For a detailed description see the according text.

binary test are just slightly better than random guessing (average recall score: 0.57). For the classification task it is necessary to use more advanced tests which should be still efficient to implement. The following four tests $f_1, \dots f_4$ are used in this work:

$$f_1 = \begin{cases} 1, & \text{if } V(\boldsymbol{p}_1) > \text{thres} \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

$$f_2 = \begin{cases} 1, & \text{if } (V(\boldsymbol{p}_1) - V(\boldsymbol{p}_2)) > \text{thres} \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

$$f_3 = \begin{cases} 1, & \text{if } (V(\boldsymbol{p}_1) + V(\boldsymbol{p}_2)) > \text{thres} \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

$$f_4 = \begin{cases} 1, & \text{if } abs(V(\boldsymbol{p}_1) - V(\boldsymbol{p}_2)) > \text{thres} \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

There, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the two randomly chosen pixel locations within each patch. $V(\boldsymbol{p})$ represents the value at location $\boldsymbol{p}$ of the feature channel where the test is applied to. *thres* is a randomly chosen threshold value within the range of the minimum and maximum values for this test, determined on all training samples. The results obtained using these tests are significantly better than with the simple binary test proposed in the original fern version of [OCLF10]. For details see Chapter 4.1.1.

### 3.1.2 Ellipse Fitting

To reconstruct a statistically identical 3D microstructure of a lithium-ion electrode based on just one image, either strong knowledge about the original 3D structure is needed or approximations need to be made. The different materials in an electrode form particles of arbitrary shapes, which we assume to be ellipsoids. To create more complex forms in the reconstructed volume, these ellipsoids are grouped to ensembles, touching each other. The relative frequencies and sizes of the ellipsoids are determined based on an ellipse distribution. For details see Chapter 3.2.1 . Therefore the segmented material needs to be approximated by ellipses. Parameter based methods like the Hough-transform [Bal81] are not suitable for this problem, because arbitrary shapes need to be approximated by ellipses. In this chapter a specialized algorithm for ellipse fitting is introduced, which exploits the given strong knowledge about the problem domain.

#### Ellipse Fitting Algorithm

The input for the algorithm is a binary mask, which defines foreground and background. The target is to split the foreground into single areas and approximate them with ellipses. The foreground consists of particles which touch each other in various ways. The shape of each particle is arbitrary; the touching area ranges from small connections up to the size of the

main axis of the particle. Often free space is enclosed within particles, which also need to be detected. Algorithm 4 shows the pseudocode for the ellipse fitting.

**Algorithm 4** Ellipse fitting

**Variables**:
Input: $mask\_fg$: foreground mask
Output: $list\_el$: saves all found ellipses

**Algorithm:**
**while** $mask\_fg$ contains foreground pixels **do**
  - find contours of foreground
  **for** every contour **do**
    - find ellipse that approximates contour best
    - calculate ellipse-similarity-factor $ESF$
    - calculate area-approximation-error $AAE$
    **if** $ESF > ESF\_min$ and $AAE < AAE\_max$ **then**
      - save ellipse in $list\_el$
      - delete actual contour from $mask\_fg$
    **else**
      - find and do best cut on actual contour
    **end if**
  **end for**
**end while**

Description of algorithm 4 - ellipse fitting: The foreground mask is divided into connected parts and the contour of each part is determined. For every contour an ellipse is created which fits the contour best in a least-squares sense. To determine if the fitted ellipse approximates the area under the contour well enough according to the user settings, the ellipse-similarity-factor $ESF$ and the area-approximation-error $AAE$ are defined. If both parameters fulfill the user requirements the ellipse is saved and the contour which is approximated by it is deleted from the mask. If one of the parameters is not in the required range it means the contour cannot be approximated with one single ellipse and needs to be cut.

*Ellipse-Similarity-Factor:* If an ellipse is fitted to an arbitrary contour in a least-square sense it always yields a result. Therefore the ESF is introduced to determine if the contour shape is similar to an ellipse. The factor equals the cross-ratio of the area-to-perimeter ratio of the contour and the ellipse. The area of the ellipse is calculated via equation (3.6), at which $a$ and $b$ are the ellipse's semi-major and semi-minor axes. The perimeter is calculated via Euler's approximation function given in equation (3.7). For the contour parameters $perimeter_{con}$ and $area_{con}$ the build-in functions of the OpenCV-Library are used. The calculation of the ESF is shown in equations (3.8) - (3.10). During the process of splitting the foreground into ellipses all chosen ellipses need to have an ESF bigger than a given minimum value. This minimum value is highly depending on the particular problem, therefore no evaluation for it will be proposed. In practice it is realized as an adjustable parameter.

$$area_{ell} = \pi \cdot a \cdot b; \tag{3.6}$$

$$perimeter_{ell} = \pi \cdot \sqrt{2 \cdot (a^2 + b^2)} \tag{3.7}$$

$$ellipse_{a-p-ratio} = \frac{area_{ell}}{perimeter_{ell}} \tag{3.8}$$

$$contour_{a-p-ratio} = \frac{area_{con}}{perimeter_{con}} \tag{3.9}$$

$$ESF = \frac{min(ellipse_{a-p-ratio}, contour_{a-p-ratio})}{max(ellipse_{a-p-ratio}, contour_{a-p-ratio})} \tag{3.10}$$

*Area-Approximation-Error:* The ESF detects contours which do not have an elliptic shape, but in Figure 3.2 an example is given where the ESF is not enough to detect the bad approximation. When the contour of a connected part is extracted just the outer contour is taken into account. Embedded holes are not detected, although the approximation is not acceptable in that case. To detect these kinds of bad approximations the area-approximation-error (AAE) is introduced. It is defined in equation (3.11) as the ratio of the not overlapping area to the contour area. Figure 3.3 shows the graphical interpretation. Every accepted contour-ellipse pair has to have an AAE smaller than a maximum value. This maximum value is an adjustable parameter and has to be set by the user for every particular problem. As it is highly depending on the particular problem, no evaluation for it is given.

$$AAE = \frac{area_{ell} \text{ XOR } area_{con}}{area_{con}} \tag{3.11}$$

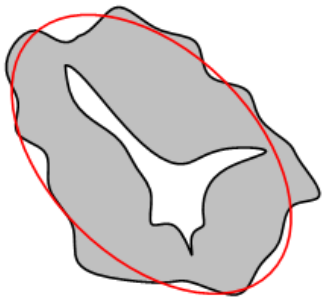


Figure 3.2: The outer contour is approximated by the red ellipse. The ESF is not capable to detect bad approximation caused by the enclosed hole.

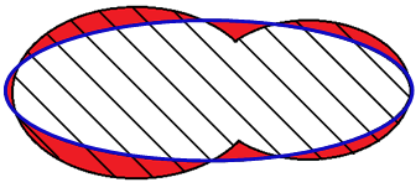


Figure 3.3: Definition of the AAE: Ratio of the not overlapping area, marked in red, to the total area of the contour, marked with stripes.

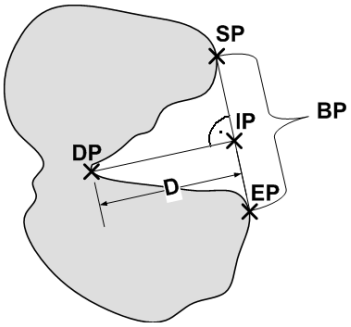



Figure 3.4: Illustration of a convexity defect. The features are: $DP$, Depth-Point; $IP$, Intersection-Point; $SP$, Start-Point; $EP$, End-Point; $BP$, Border-Points (all points between the $SP$ and the $EP$); $D$, Depth of the defect.

*Cutting Ellipses:* If a contour cannot be approximated well enough by an ellipse, which means its AAE is higher than the user set threshold or its ESF is lower than the according user set threshold, it needs to be cut. This gives two new particles which have a lower AAE than the original one. In the Experimental Results section in chapter 4.1.4 an investigation of this is given. For the proposed cutting algorithm the existent strong knowledge about the problem domain is exploited. A part which should be cut is assumed to consist of two or more single particles, where every particle for its own can be approximated by an ellipse. Before the cutting algorithm is explained the term convexity defect needs to be defined.

*Convexity Defect:* Given an arbitrary shape and its convex hull, regions where the convex hull does not align with the border of the shape are called convexity defects. In this work a convexity defect is defined by its depth, border-points of the defect, and an additional start-, end-, depth- and intersect-point. Figure 3.4 illustrates these features graphically. To extract these features build-in functions of the OpenCV-Library are used and expanded for this particular purpose.

The cutting algorithm is applied to every particle which cannot be approximated well enough by one ellipse according to the user regulations. The pseudocode is given in algorithm 5.

According to our problem domain, a deep convexity defect indicates possible good cuts to separate two particles. If the particles are of an easy shape, the best cut directions leads from the intersect-point to the depth-point. Because of the definition of the convexity defect, the direct line between these points could intersect with the particle. In that case the cut-line needs to be moved to get the best possible cut direction. Possible directions lead from any point of the convex hull between the start- and end-point to the depth-point. The best cut direction is defined as the cut-line which intersects the least with the particle. Figure 3.5 shows two particles with its best cutting lines. In Figure 3.5(a) the dashed arrow shows the initial cut direction from the intersection-point to the depth-point. Before the cut is finally done, an isosceles triangular area with the depth-point as corner and a given opening angle is checked for the best cut. This means all possible directions within this triangular are taken into account and the one with the shortest cut length is chosen. In Figure 3.5(a) the

**Algorithm 5** Cutting particles

**Variables**:
Input: *act_contour*: contour of particle which should be cut

**Algorithm:**
- find deepest convexity defect of act_contour
- set cut-line = intersect_point → depth_point
**if** cut-line intersects with contour **then**
    **for** every border_point **do**
        - set cut-line = border_point → depth_point
    **end for**
    - select cut-line which intersects least with contour
**end if**
- check opening angle for best cut
- do the cut

solid arrow indicates the resulting cutting direction. In Figure 3.5(b) the direct line from the intersection-point to the depth-point crosses the particle, which indicates that the cut direction might not be the best. The cutting line is moved to a position where it intersects with the particle the least. The original position is indicated by the dashed arrow. The solid arrow shows the resulting cutting direction after adjustment.

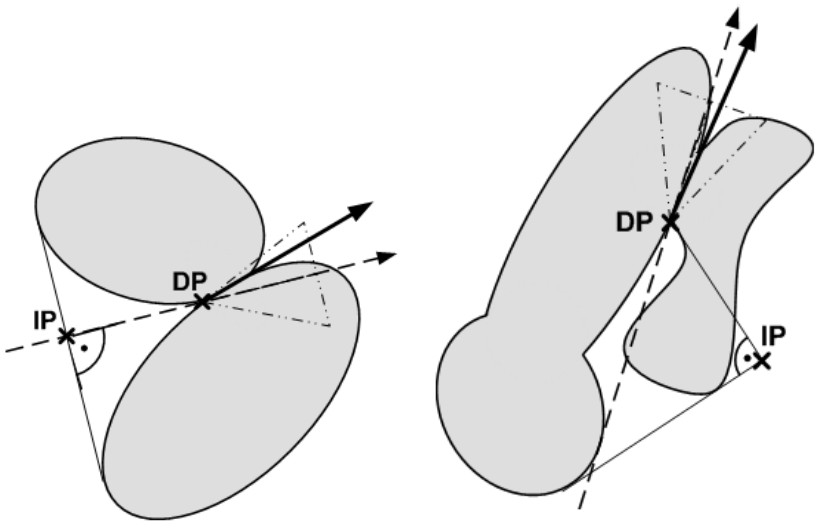
## 3.2 Stochastic

In Chapter 2.3.1 of the related work section it is shown how the radii and the relative frequencies of circles can be reconstructed from a histogram of cut lengths. In [Wal10] this method is the base to reconstruct ellipsoids out of a binary image which shows ellipse like contours. The main idea is to perform a cut at every pixel position on the x- and y-axes and combine the detected cut lengths.

In this work the segmented materials in the images are already approximated by ellipses. This means the proposed reconstruction algorithm for ellipsoids in [Wal10] is not suitable. This chapter shows how the findings in [Wal10] can be expanded to the 2D case and how the size and the relative frequencies of ellipsoids are reconstructed based on a histogram of ellipses.

### 3.2.1 Reconstruction of Ellipsoids from Ellipse Distributions

We assume a 3D volume which contains ellipsoids with certain restrictions. The ellipsoids are defined by their three semi-principal axes $a, b, c$, their location and their rotation about the z-coordinate. No rotation about the x- or y-coordinate is allowed. We also assume that the semi-principal axis $c$ equals the longest of the remaining semi-principal axes. A horizontal cut through this volume is done which results in an image showing different sized ellipses




(a) Easy separation (b) Advanced separation

Figure 3.5: Figure (a) and (b) show two common constellations of touching particles. In the figures the intersect-point $IP$, the depth-point $DP$, the initial cutting direction as dashed arrow and the final cutting direction as solid arrow are shown. A detailed explanation is given in the text.

with rotations. Based on these ellipses in the image the original distribution of ellipsoids, consisting of their size and relative frequencies, should be reconstructed. The positions of the ellipses can be discarded, because the reconstructed ellipsoids will be placed randomly in a certain volume element. The distribution of the rotation angles of the ellipses is determined separately and for the final reconstruction of the volume element combined with the associated ellipsoids. To determine the sizes and relative frequencies of the original ellipsoids just the dimensions of the ellipses are needed. The 2D histogram of the ellipse dimensions can be interpreted as the weighted sum of the ellipse density functions of the original ellipsoids. An example of such a histogram is given in Figure 3.6. The next section shows the derivation of the ellipse density function and how the ellipsoids are reconstructed.

#### Derivation of Ellipse Probability Density

An ellipse in canonical form in the Cartesian coordinate system is defined by:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1. \tag{3.12}$$

There, $a$ and $b$ denote the semi-major and semi-minor axis of the ellipse. In canonical form, $a$ aligns with the x-coordinate and $b$ with the y-coordinate; it must hold that $a \geq b$. It is obvious that a circle is a special form of an ellipse (see equation (2.48)). Therefore the

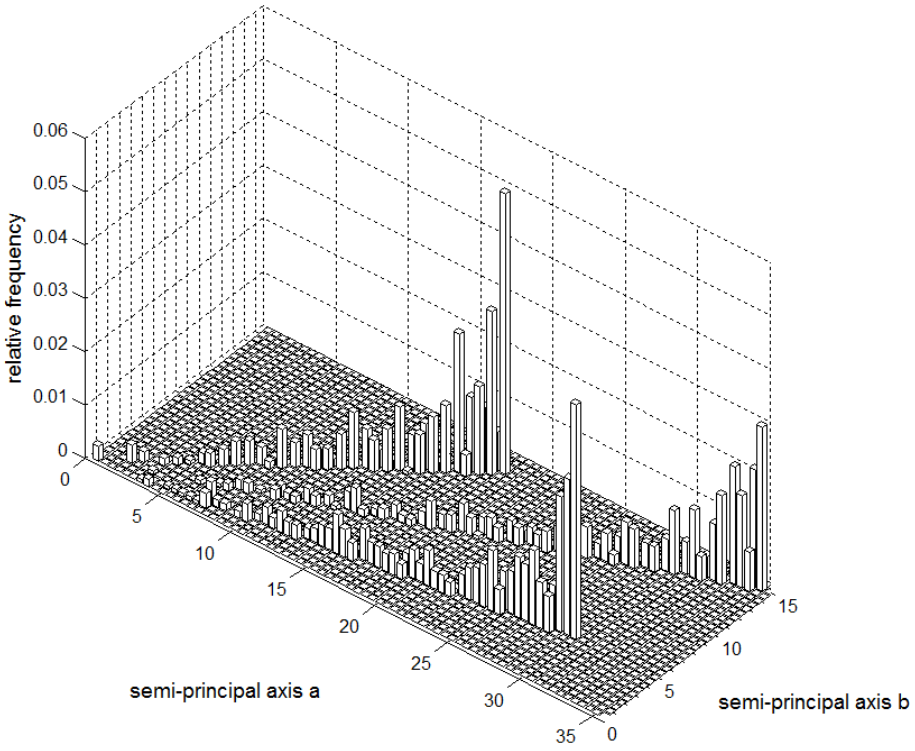



Figure 3.6: 2D histogram of ellipse distribution: Three ellipsoids have been cut at random positions. The sizes of their semi-principal axes $a, b, c$ are: 30.0, 5.0, 30.0; 35.0, 15.0, 35.0 and 18.0, 14.0., 18.0.

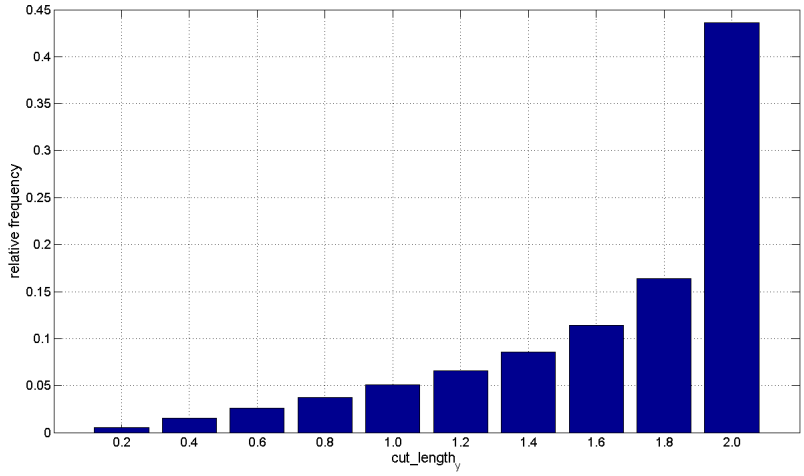



Figure 3.7: A discrete cut length density function of the unit circle with 10 bins.

findings of Chapter 2.3.1 will be used to generate an ellipse density matrix, which represents the 2D analog of the discrete cut length density vector. For convenience the discrete cut length density vector of Chapter 2.3.1 is shown again in Figure 3.7.

The length of a cut in y-coordinate direction becomes:

$$cut\_length_y(x) = 2 \cdot \sqrt{\left(1 - \frac{x^2}{a^2}\right) \cdot b^2}. \tag{3.13}$$

The cut length in y-coordinate direction varies from 0.0 to $2b$ and $x$ denotes the position of the cut on the x-coordinate. The possible cut positions are uniformly distributed over the x-coordinate. This leads to a uniform distribution density $f_X(x)$:

$$f_X(x) = \frac{1}{2 \cdot a}. \tag{3.14}$$

Figure 3.8 shows an ellipse, the resulting cut length distribution $cut\_length_y(x)$ and the uniform distribution density $f_X(x)$.

Similar to Chapter 2.3.1 we introduce a random variable $Y$ which represents the cut length in y-coordinate direction. A certain value for it is denoted by $cut\_length_y$. The cut length distribution function $F_Y(cut\_length_y)$ is now defined in the range from $cut\_length_y = 0.0$ to $2b$ and denotes the probability that a uniformly over the x-coordinate distributed cut has a length of $cut\_length_y$ or less. A given $cut\_length_y$ results according to equation (3.13) in two possible positions $x_1, x_2$ on the x-coordinate:

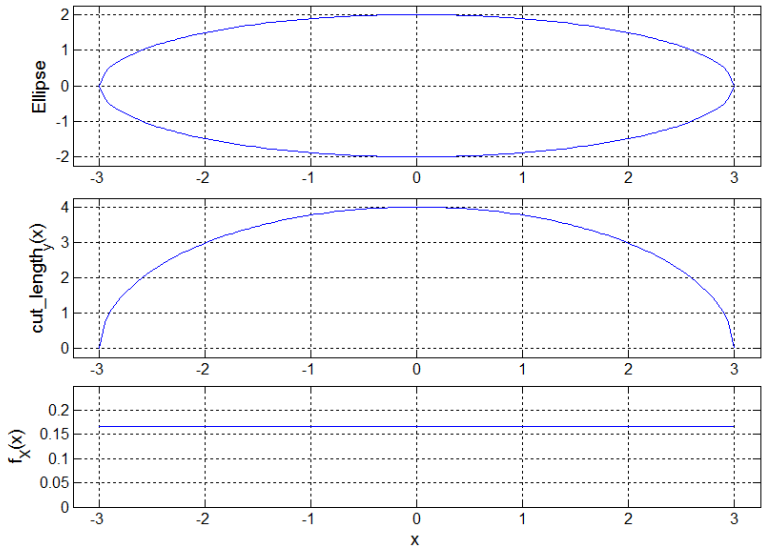



Figure 3.8: The upper diagram shows an ellipse with $a = 3.0$ and $b = 2.0$. The middle diagram shows the resulting cut length in y-coordinate direction depending on the cut position on x. In the lower diagram the uniform distribution density $f_X(x)$ is shown.

$$x_{1,2} = \pm\sqrt{\left(1 - \left(\frac{y}{2 \cdot b}\right)^2\right) \cdot a^2}. \tag{3.15}$$

Using equation (2.50) of Chapter 2.3.1:

$$F_Y(cut\_length_y) = P(Y \leq cut\_length_y) = 1 - P(Y > cut\_length_y), \tag{3.16}$$

and $x_1, x_2$ from equation (3.15) as limits for equation (3.14), $F_Y(cut\_length_y)$ becomes:

$$F_Y(cut\_length_y) = 1 - P(Y > cut\_length_y) \tag{3.17}$$

$$F_Y(cut\_length_y) = 1 - \int_{x_1}^{x_2} f_X(x)\,\mathrm{d}x \tag{3.18}$$

$$F_Y(cut\_length_y) = 1 - \sqrt{1 - \left(\frac{cut\_length_y}{2 \cdot b}\right)^2} \tag{3.19}$$

$F_Y(cut\_length_y)$ denotes the probability that a cut performed on the ellipse normal to the x-coordinate has a length of $cut\_length_y$ or less. It is noticeable that equation (3.19) has the

same form as the cut length distribution function of a general circle (2.54). The derivation of $F_X(cut\_length_x)$ is straightforward and yields:

$$F_X(cut\_length_x) = 1 - \sqrt{1 - \left(\frac{cut\_length_x}{2 \cdot a}\right)^2}. \tag{3.20}$$

There, $X$ is a random variable which represents the cut length in direction of the x-coordinate and $cut\_length_x$ represents a certain value for it. $F_X(cut\_length_x)$ is defined in the range from $cut\_length_x = 0.0$ to $2a$ and represents the possibility that a cut performed normal to the y-coordinate has a length of $cut\_length_x$ or less.

The derivation of $F_Y(cut\_length_y)$ and $F_X(cut\_length_x)$ gives the cut length density functions for both coordinate directions, denoted by:

$$f_x(cut\_length_x) = F'_X(cut\_length_x) \tag{3.21}$$
$$f_y(cut\_length_y) = F'_Y(cut\_length_y) \tag{3.22}$$

As the cut lengths are in a discrete form, the density functions also need to be discretised. In Figure 3.7 such a discretised density function is shown as a vector of 10 elements. For further explanations we assume $f_x()$ and $f_y()$ to be vectors and representing the discrete density functions of the cut lengths in x- and y-coordinate direction. To create the ellipse density matrix, which is the 2D analog of a cut length density vector (Figure 3.7), $f_x()$ and $f_y()$ need to be combined. The ideal ellipse density matrix has a size of $2a \times 2b$, which equals the lengths of the according density vectors $f_x()$ and $f_y()$, and consists of zeros except on the main diagonal. It is straightforward to prove that an ellipse density matrix has to have this shape if it originates from an ellipsoid according to the regulations set and all cuts are done horizontal. An example of an ideal ellipse density matrix is shown in Figure 3.9(c). The main diagonal values are a product of the according relative frequencies of the density vectors. In an experimental way this matrix can be produced if an ellipsoid in canonical form defined by $a, b, c$, with $a \geq b, c = a$, is intersected by a horizontal plane several times. The position of the plane on the z-axis is normally distributed over the height of the ellipsoid. The 2D histogram of the resulting combinations of the major and minor axes of the produced ellipses is an ideal ellipse density matrix. Because of the discretisation of the main diagonal and the whole ellipse distribution (see Figure 3.6 as example) the ideal ellipse density matrix cannot be used for the reconstruction of the ellipsoids. Values near the main diagonal also need to be determined to model discretisation uncertainties. At first a so called full ellipse density matrix of size of the ideal ellipse density matrix is created. Each element is calculated via the multiplication of the according values of the cut length density vectors. An example of a full ellipse density matrix is shown in Figure 3.9(d). As we assume the cuts to be horizontal, and the ellipsoids in an upright position, elements with a big distance to the main diagonal are not possible, but the full ellipse density matrix does not consider this fact. Therefore the elements of the matrix are weighted with a Gaussian distribution normal to the main diagonal. Figure 3.9(e) shows such a Gaussian distribution matrix which is entry-wise multiplied with

the full ellipse density matrix. The result is a weighted ellipse density matrix shown in Figure 3.9(f).

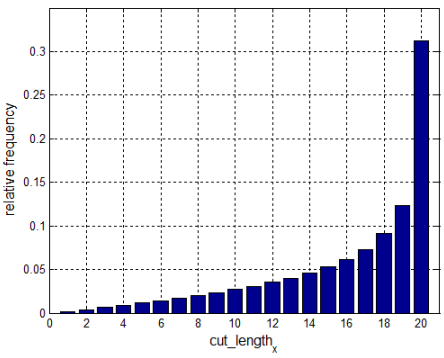
### Reconstruction of Ellipsoids

This section shows the algorithms which are used to reconstruct the ellipsoids. We assume an original 3D volume, consisting of ellipsoids of different size, which are located at random positions within the volume. The ellipsoids are defined by their semi-principal axes $a, b, c$, with $a \geq b, c = a$, and a rotation angle about the z-coordinate. We assume no rotations about the x- or y-coordinate. A horizontal cut through the original volume is done which results in an image with various ellipses. Based on this image the sizes and relative frequencies of the ellipsoids in the original volume are reconstructed. The positions of these ellipses are not analyzed, because the reconstructed ellipsoids are placed randomly in the volume. The rotation angles of the ellipses are treated separately, so their distribution is recorded and taken into account when the discrete ellipsoids are created. The combinations of the major and minor axes are binned into a 2D histogram (Figure 3.6). This histogram is interpreted as the sum of weighted ellipse density matrices and forms the base of the reconstruction algorithm. For the reconstruction of ellipsoids the same help function as for the reconstruction of circles is needed. For convenience the pseudocode is given in algorithm 6 again. The function generates a cut length density vector for a circle with given radius $r$. For the ellipse density matrix two of these vectors are combined. Algorithm 7 shows the pseudocode of the reconstruction algorithm.

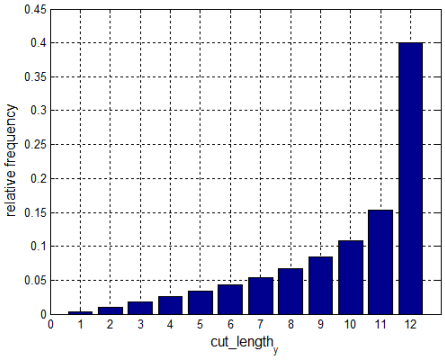
**Algorithm 6** Generating the cut length density

**Function:** generate_cut_length_density(radius r)
**Description:** Generates the discrete cut length density based on equation (2.54) for a circle of radius r.
**Parameter:** r: radius of circle
**Returns:** vector representing the cut length density (a plot of such a vector is shown in Figure 2.16)

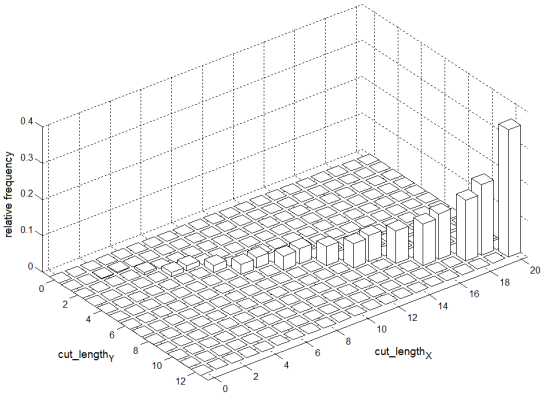
Description of algorithm 7 - reconstruction of ellipsoids: The indices $row, col$ of the ellipse mixture matrix $emm$ represent the principal axes of the according ellipses. Starting from the ellipse with the biggest size, it is checked if the relative frequency of the ellipse is higher than a certain threshold. This threshold affects the amount of the reconstructed ellipsoids and is adjustable by the user. If a principal axes combination is significant enough, the according weighted ellipse density matrix is created. At the beginning the cut length density vectors in x- and y-coordinate direction $f_x, f_y$ are generated. Then these vectors are combined to create the full ellipse density matrix $fedm$. Given the size of this matrix, the Gaussian distribution matrix $gdm$ is created. It is a symmetrical matrix which represents a Gaussian distribution normal to the main diagonal of $fedm$. The weighted ellipse density matrix $wedm$ is created as a per element multiplication of the full ellipse density matrix and the Gaussian distribution matrix. The relative frequency of the actual reconstructed ellipsoid is determined by dividing the value of the ellipse mixture matrix $emm$ by the according value of



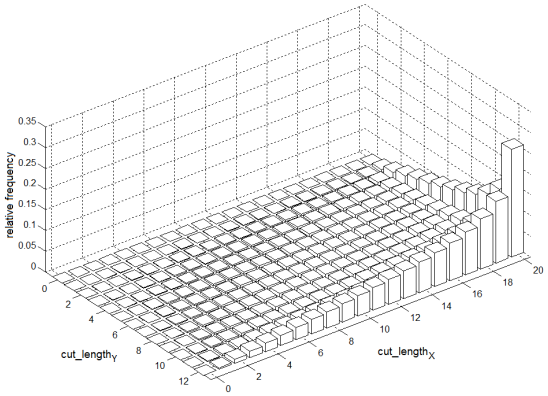

(a) Cut length density x



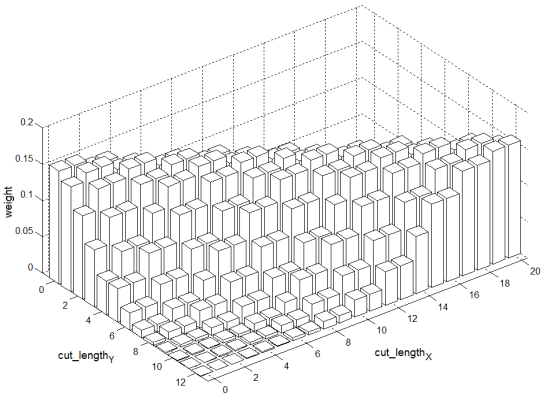

(b) Cut length density y



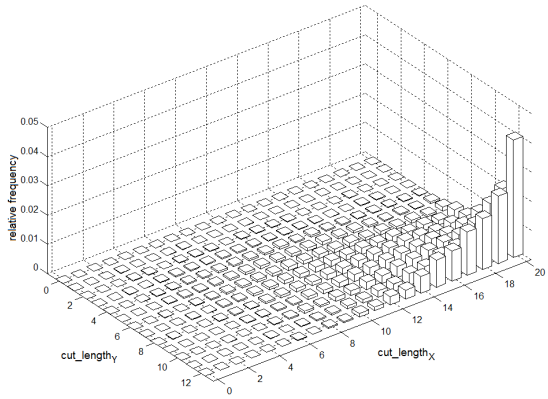

(c) Ideal ellipse density matrix




(d) Full ellipse density matrix




(e) Gaussian distribution matrix




(f) Weighted ellipse density matrix

Figure 3.9: Derivation of the ellipse density matrix of an ellipse with $a = 10.0$ and $b = 6.0$. Figures (a) and (b) show the cut length density vectors in x- and y-coordinate direction. Figure (c) shows the resulting ideal ellipse density matrix as a combination of them. In Figure (d) the full ellipse density matrix is shown. The values result from an entry-wise multiplication of the according density vector values. Figure (e) shows a Gaussian distribution normal to the main diagonal. The matrix is multiplied entry-wise with the full ellipse density matrix of Figure (d). The result is shown in Figure (f). The weighted ellipse density matrix is used in the process of reconstructing the ellipsoids. The relative frequencies of Figure (c), (d) and (f) have been scaled logarithmical for better legibility.

**Algorithm 7** Reconstruction of ellipsoids

**Variables**:

Input: $emm$: ellipse mixture matrix; representing the 2D histogram of the major and minor axes combinations

Output: $rel\_f$: vector containing the relative frequencies of the reconstructed ellipsoids
$rec\_ell$: reconstructed ellipsoids, defined by their semi-principal axes $a, b, c$

**Algorithm:**

**for** row = (rows of emm) to 1 **do**
  **for** col = (columns of emm) to 1 **do**
    **if** emm(row,col) > threshold **then**
      - determine cut length density in x-coordinate direction:
        $f\_x$ = generate_cut_length_density(row/2)
      - determine cut length density in y-coordinate direction:
        $f\_y$ = generate_cut_length_density(col/2)
      - create full ellipse density matrix $fedm$ as combination of
        $f\_x$ and $f\_y$ and normalize it
      - create Gaussian distribution matrix $gdm$
      - create weighted ellipse density matrix:
        $wedm = fedm$ per element multiplied by $gdm$
      - save actual relative frequency
        $act\_f = emm(row, col)/wedm(row, col)$ in $rel\_f$
      - save reconstructed ellipsoid $a = c = col/2$, $b = row/2$ in $rec\_ell$
      - remove reconstructed density: $emm = emm - act\_f \cdot wedm$
    **end if**
  **end for**
**end for**
- normalize relative frequencies to: $sum(rel\_f) = 1$

the weighted ellipse density matrix *wedm*. The size of the reconstructed ellipsoid is given by the actual processed indices of the matrices. The reconstructed density needs to be removed from the ellipse mixture matrix weighted by the according relative frequency. As last step the relative frequencies need to be normalized.

The quality of the algorithm is mainly depending on the amount of ellipses in the ellipse mixture matrix being large enough. An investigation of the error rate depending on the number of given ellipses is shown in Chapter 4.2.1.

# 4 Experimental Results

This chapter shows the experimental results of the three main parts of this work, namely classification, segmentation and reconstruction. First, the intermediate results which led to the final parameter configuration for the ferns are shown. The final classification sub-system is also compared to the fern version for keypoint recognition proposed in [OCLF10]. Second, the performance of the sub-system 'classification combined with segmentation' on natural and task-specific images is determined. Then the quality of the reconstruction algorithm depending on the number of detected ellipses is investigated. At last, a whole work flow of the final system is illustrated with all intermediate results and images.

## 4.1 Computer Vision

The outcome of this work is a tool to reconstruct the 3D microstructure of lithium-ion electrodes. Therefore the evaluation of different fern parameters was performed just on the task-specific images. As no public data is available about the performance of other segmentation systems on lithium-ion electrode images, the performance of the proposed system on the public available Icg-bench, proposed in [San10], is also shown. Although the images of this dataset are colored and showing natural scenes, in contrast to the grayscale cell images which show materials with different structure, the performance reached is comparable to the multi-label segmentation framework in [San10].

The scores used to measure the performance are the global pixel accuracy, the average recall and the Dice score. The global pixel accuracy denotes the ratio of the correctly classified pixels in percent of the entire set of classified pixels:

$$global\_pixel\_accuracy = \frac{number\_correctly\_labeled\_pixels}{total\_number\_of\_pixels}. \tag{4.1}$$

The term 'recall' refers to the correctly classified pixels of a certain class. The average recall is the average over all present classes in the image:

$$average\_recall = \frac{\sum_{i=0}^{classes-1} \frac{number\_correctly\_labeled\_pixels\_of\_class\_i}{total\_number\_of\_pixels\_of\_class\_i}}{number\_of\_classes}. \tag{4.2}$$

In [San10] the Dice score, based on [Dic45], is used to measure the quality of multi-label segmentation results. This score is defined as:

$$dice(E_1, E_2) = \frac{2 \cdot |E_1 \cap E_2|}{|E_1| + |E_2|}, \tag{4.3}$$

where $|E_i|$ denotes the area of segment $E_i$. To determine the Dice score of one image, the average over all segments related to the ground-truth labeling is calculated:

$$dice\_score = \frac{1}{N} \cdot \sum_{i=1}^{N} dice(E_i, GT_i), \tag{4.4}$$

where $GT_i$ is the ground-truth labeling of segment $i$. The overall benchmark score is the average over all image scores. The advantage of the Dice score is that the sizes of the areas are taken into account. If a large area is labeled inaccurately at the border, the resulting score for this specific element will be quite high. If a small area is not labeled correct at all, the according score will be zero, which lowers the entire score of the image significantly. If in both cases the absolute number of misclassified pixels is the same, the global pixel accuracy would not show any difference, although the missing of an entire label in the result is much worse [San10].

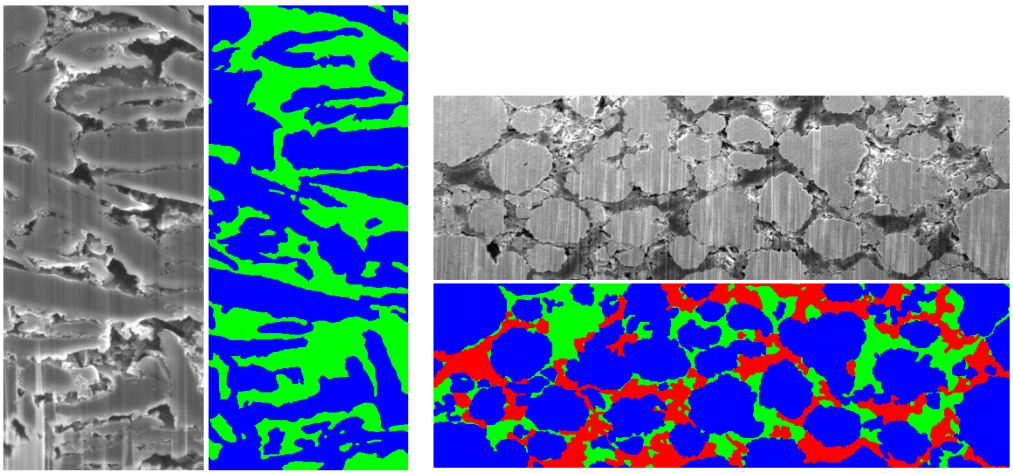
In this section the average recall and the global pixel accuracy are used to determine the performance of the different fern parameters on the lithium-ion electrode images. For all experiments done with the Icg-bench the Dice score is additionally shown.

### 4.1.1 Classification

In this work the concept of ferns, like proposed in [OCLF10], is used to classify different materials in scanning electron microscopy (SEM) images. These images show cuts through electrodes of lithium-ion batteries. The experiments led to an ensemble of 30 ferns of size 10 and a patch size of $7 \times 7$ for the final system. As features the gray value intensities and the first and second order derivatives were chosen. First it is shown how these values have been determined. Then a comparison with the proposed version in [OCLF10] is given.

The outcome of this work should help experts in the field of lithium-ion batteries to examine the inner structure of lithium-ion cells. For this reason all the experiments have been done with SEM-images of different electrode materials. A drawback of this is, that at the time when this master thesis was written, just few of them were available to the author. Figure 4.1 shows two representative images with according ground truth labeling which were used for the experiments. Figure 4.1(a) shows an image of a $C_6$-electrode, which should be separated in active material and pores. In Figure 4.1(b) a NCA-electrode is shown, where three materials should be detected: active material, binder material and pores.

Every test-run was realized via a 5-fold cross validation, at which every subpart represents an average of 20 turns, to average the influence of randomness. The results are visualized via a boxplot which shows the following for every group of data points (the meaning of the data points is described in detail for every figure): the median; the 25th and 75th percentile as



(a) $C_6$-electrode material with ground truth

(b) NCA-electrode material with ground truth

Figure 4.1: Figure (a) shows a $C_6$-electrode and the corresponding ground truth labeling. Blue marks the active material and green the pores. Figure (b) shows the labeling of a NCA-electrode: active material in blue, pores in green and binder material in red.

the borders of the box; the most extreme data points which are inside an interval of 99.3% coverage of the data via whiskers; and if existent, outliers outside this interval.

For the runtime evaluations images of size $737 \times 668$ for the two-label problem associated with the $C_6$-electrode, and images of size $1016 \times 221$ for the three-label problem according to the NCA-electrode, were used.

All experiments were done on a PC with an Intel dual core processor with a clock speed of 2.53 GHz, and 4 GB RAM. The implementation is CPU-based to not restrict the use of the final system to certain hardware requirements.

### Feature Channels

The original framework introduced in [OCLF10] for keypoint recognition uses just the gray value intensity of the image as feature channel. In this work different materials should be separated which mainly differ in their structure (see Figure 4.1), therefore a wider range of feature channels is investigated. The investigated feature channels include: the gray value intensity, the first and second order derivatives in both coordinate directions and histogram of oriented gradient (HOG) features with 9 bins. For a detailed description see Table 3.1. The filtering of these features with a $3 \times 3$ minimum- and a $3 \times 3$ maximum-filter leads to a total of 30 feature channels.

At first the performance of every single feature channel was determined. Then the findings were used to combine certain feature channels and compare their performance among them.

| | Mean value of scores | |
|---|---|---|
| | Global | Average Recall |
| Best 4 channels | 0,7842 | 0,7671 |
| Best 10 channels | 0,7634 | 0,7476 |
| First 6 channels | 0,7793 | 0,7631 |

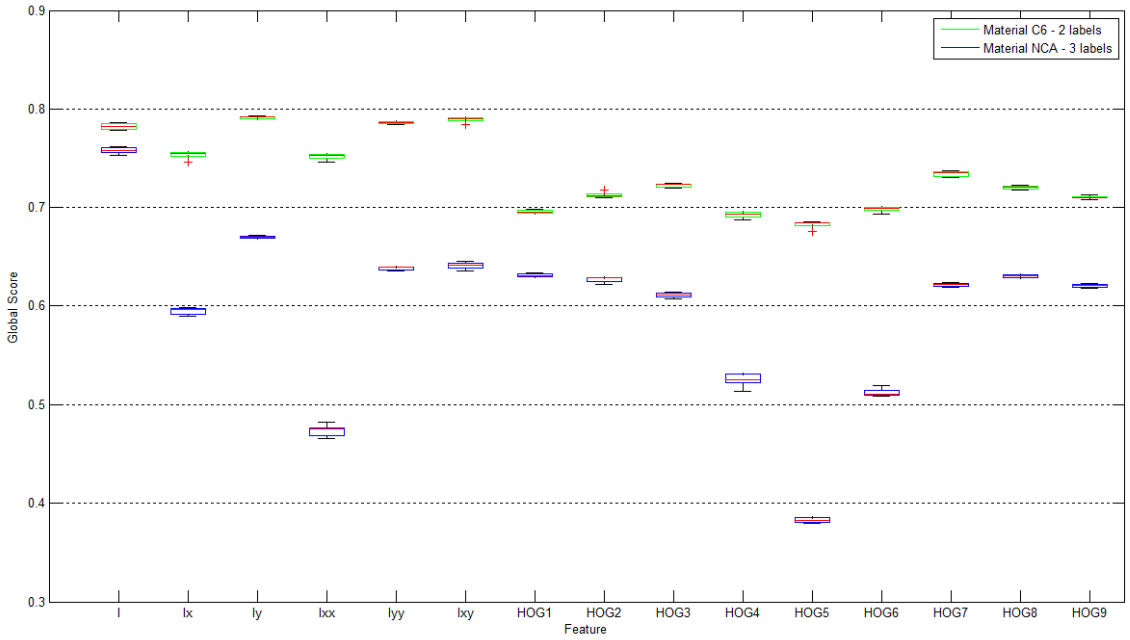
Table 4.1: The table shows the average performance of certain combinations of feature channels for the NCA material. The combinations are (see Figure 4.2 and Table 3.1): Best 4 channels: I, Iy, Iyy, Ixy; best 10 channels: I, Iy, Iyy, Ixy, HOG1, HOG2, HOG3, HOG7, HOG8, HOG9 and first 6 channels: I, Ix, Iy, Ixx, Iyy, Ixy.

Figure 4.2 shows the achieved scores of the maximum filtered feature channels. As the minimum filtered channels show the same pattern, they are not listed for better legibility. The first six feature channels, including the gray value intensity and the derivatives, show significantly better results over the HOG features for the $C_6$ material with two labels. For the NCA material with three labels that clear divergence is not visible. It is noticeable that the derivatives in direction of the x-coordinate as well as the HOG features, which represent directions close to the x-coordinate ($+20°$, $0°$ and $-20°$) perform significantly worse than the other features. The reason for this behavior are the vertical artifacts in the images (see Figure 3.1) which arise from the focused-ion-beam milling method. Three combinations of feature channels for the NCA material have been further investigated: The best 4 (gray value intensity, derivatives in direction of y-coordinates and magnitude of gradient), the best 10 (include the best 4 and the 6 HOG features which perform the best) and the first 6 (gray value intensity, all derivatives and the magnitude of the gradient). The results are shown in Table 4.1. The tests show that the HOG features are not suitable for the segmentation of different materials in lithium-ion electrodes. They do not offer an increase in performance compared to the derivatives. This originates from the fact that the materials mainly differ in their structure, but do not show significant directions in their gradients.

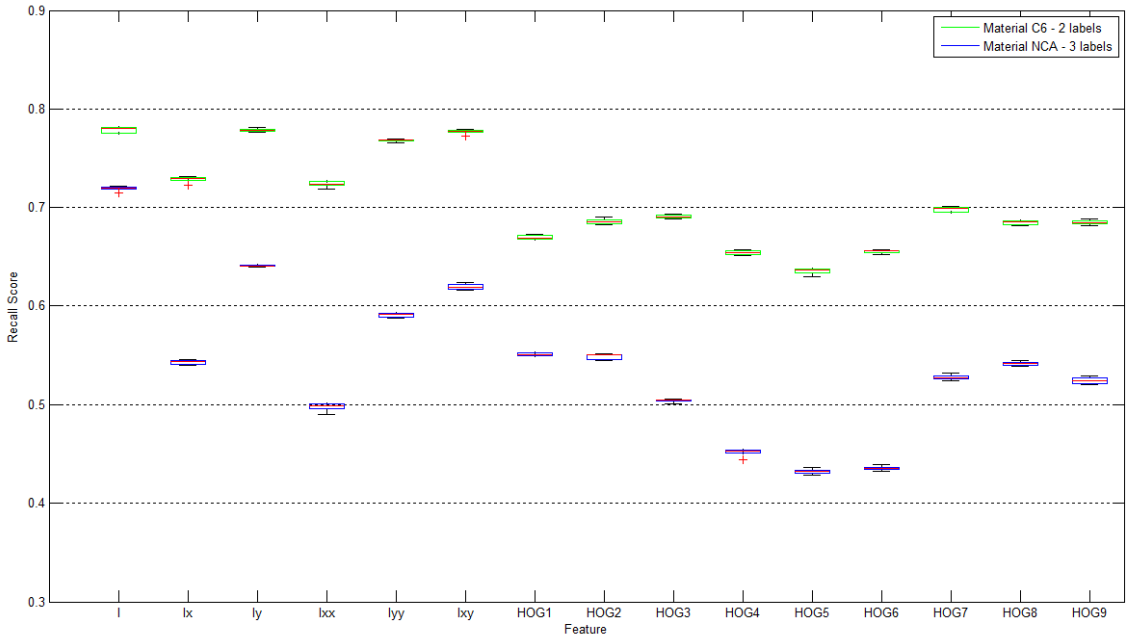
For the final system the combination of the first 6 feature channels is chosen. They perform best for the $C_6$ material and for the NCA material their performance is insignificant worse than the performance of the 4 best feature channels for this material.

#### Binary Tests

In [OCLF10] the comparison of the gray level intensities of two pixel locations is used as binary test for the ferns. For the specific task of image classification in this work this test is not suitable. Especially when used to classify images of an NCA electrode the results are just slightly better than random guessing. Table 4.2 compares the obtained results for using the binary test of the original fern version of [OCLF10] and the four binary tests used in this work. They include the comparison of a pixel value with a threshold; the comparison of the sum, respectively difference, of two pixel values with a threshold and the absolute difference of two pixel. For a detailed mathematical description see Chapter 3.1.1. Table 4.2 shows that the proposed tests outperform the original one, especially for the NCA-material images. The




(a) Global Score




(b) Average Recall

Figure 4.2: Scores depending on the features: Figure (a) shows the global score; in Figure (b) the average recall is plotted.

| Binary test version | Mean value of scores | | | |
|---|---|---|---|---|
| | $C_6$ | | NCA | |
| | Global | Average Recall | Global | Average Recall |
| Original | 0,7817 | 0,7862 | 0,6063 | 0,5689 |
| Proposed | 0,8241 | 0,8328 | 0,8004 | 0,7850 |

Table 4.2: The table shows the average performance for different binary tests. The original version includes just one test based on the comparison of gray value intensities. The proposed tests are described in detail in the according text and in Chapter 3.1.1.

listed score values are the results of a 5-fold cross validation as described in the beginning of this chapter. For the experiments the first 6 feature channels of table 3.1, 30 ferns of size 10 and a patch size of $7 \times 7$ were used.

### Fern Parameter

This chapter shows the influence of the fern size, the number of ferns and the patch size on the computation time and the performance. It should be mentioned that a fast computation time was not a goal of the implementation.

***Number of Ferns:*** Figure 4.4 shows the achieved global and recall scores for the introduced images depending on the number of ferns. The plots show that with increasing number of ferns the associated scores ameliorate and the scatter among the individual runs decreases. From the amount of 30 ferns upwards the gain of performance is insignificant compared to the higher computation time needed. In Figure 4.3(b) the classification output for the system with just 5 ferns is shown. The smaller number of ferns leads to a highly scattered result. Table 4.3 shows the achieved average runtime of the algorithm depending on the number of ferns. For this evaluation a patch size of $9 \times 9$ and a fern size of 11 was used.

***Size of Ferns:*** Figure 4.5 shows the global and the average recall performance depending on the size of the ferns. The graphs show, that an increasing fern size leads to an increase of the performance. Figure 4.3(c) shows the classification result for ferns of size 4. A small fern size reduces the ability of the system to distinguish classes which are similar to each other. In the given example this yields a misclassification of the vetical artifacts. In table 4.4 the according average runtime of the algorithm is given. A significant increase of runtime is detected above a fern size of 11. This evaluation was done with 30 ferns and a patch of size $9 \times 9$.

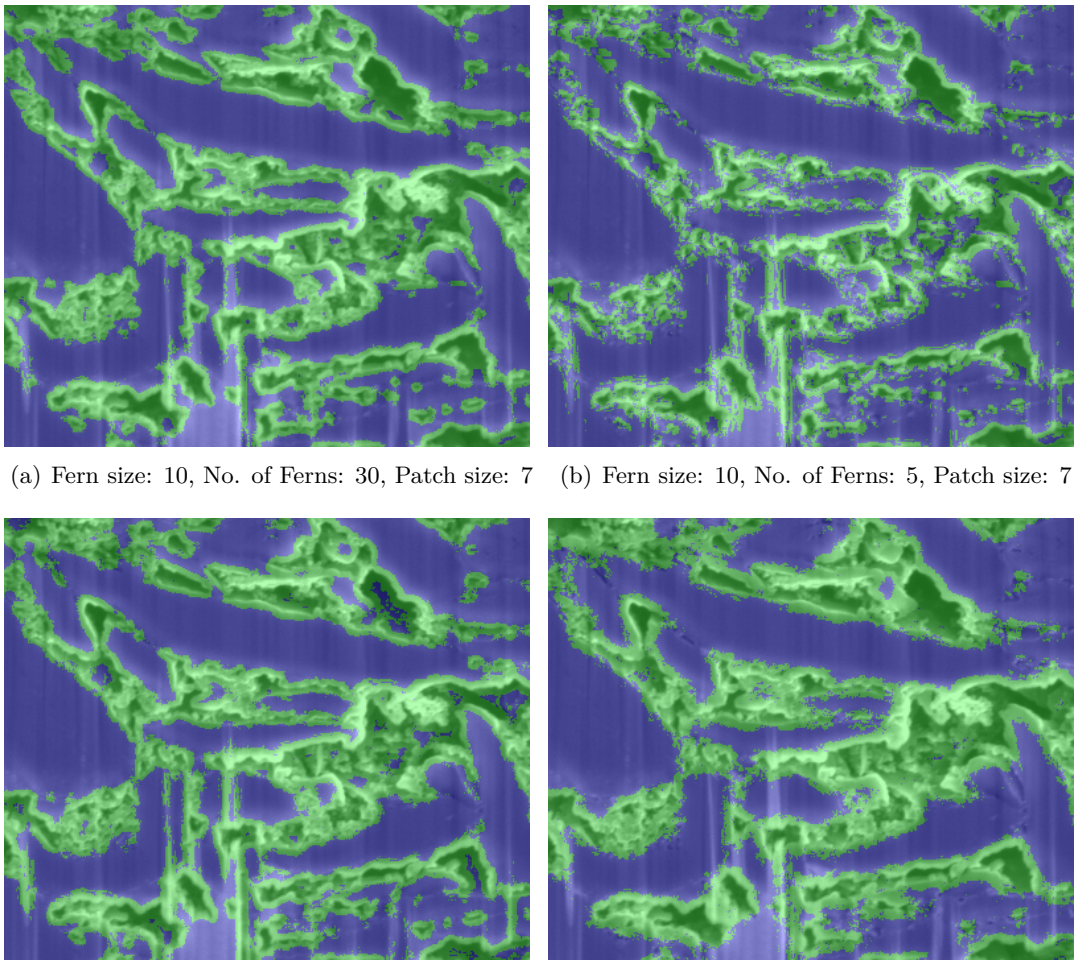
***Patch Size:*** Figure 4.6 shows the global and the average recall score depending on the size of the used patch. The peak of the recall score is reached with a patch size of $7 \times 7$ and the global score shows an insignificant degradation of a size of $5 \times 5$ compared to $7 \times 7$. Therefore a patch size of $7 \times 7$ was chosen for further evaluation of the system. Figure 4.3(d) shows the classification result for a patch size of $19 \times 19$. A too big patch size results in regions with scattered borders and the loss of spatially small segments. For all tests of this evaluation, 30

| | Runtime / Image [sec] | |
|---|---|---|
| Number of Ferns | $C_6$ | NCA |
| 5 | 5,7 | 3,2 |
| 10 | 8,5 | 5,6 |
| 20 | 14,4 | 8,8 |
| 30 | 20,1 | 12,1 |
| 40 | 25,7 | 15,6 |
| 50 | 31,1 | 18,8 |
| 100 | 59,6 | 36,2 |
| 200 | 113,9 | 71,2 |

Table 4.3: The average runtime per image for an increasing number of ferns is given. The $C_6$-image has a size of $737 \times 668$ and involves two labels, the NCA-image has a size of $1016 \times 221$ and involves three labels.

| | Runtime / Image [sec] | |
|---|---|---|
| Size of Ferns | $C_6$ | NCA |
| 4 | 11,6 | 5,9 |
| 5 | 12,2 | 6,7 |
| 6 | 12,6 | 7,4 |
| 7 | 14,0 | 7,4 |
| 8 | 14,9 | 8,0 |
| 9 | 15,7 | 8,5 |
| 10 | 17,3 | 9,2 |
| 11 | 19,6 | 12,2 |
| 12 | 26,7 | 21,3 |
| 13 | 54,7 | 59,0 |

Table 4.4: The average runtime per image for an increasing fern size is given. The $C_6$-image has a size of $737 \times 668$ and involves two labels, the NCA-image has a size of $1016 \times 221$ and involves three labels.



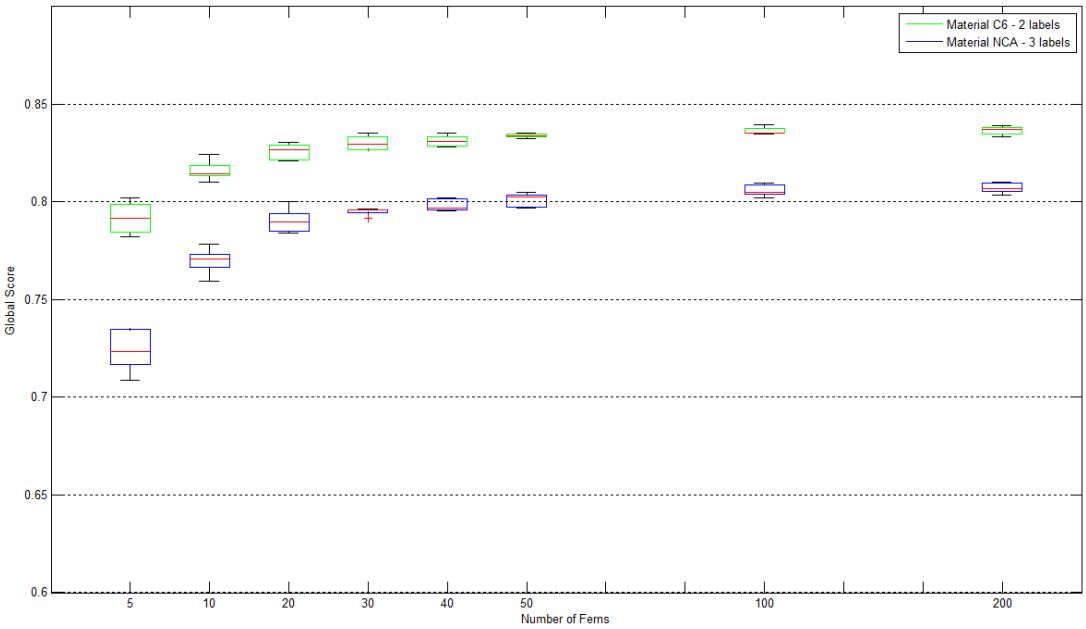
(a) Fern size: 10, No. of Ferns: 30, Patch size: 7

(b) Fern size: 10, No. of Ferns: 5, Patch size: 7

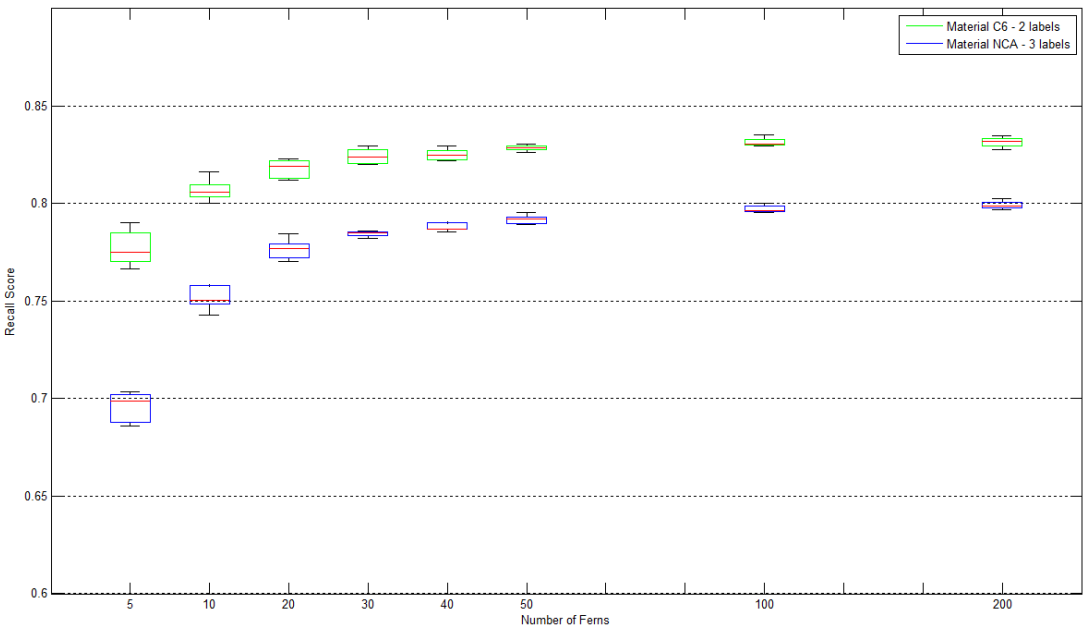
(c) Fern size: 4, No. of Ferns: 30, Patch size: 7

(d) Fern size: 10, No. of Ferns: 30, Patch size: 19

Figure 4.3: Cut-outs of the classification results for the $C_6$-material. Figure (a) shows the result for the best configuration. Figure (b) and (c): if too less ferns are used or the size of each fern is too small, the final classification is more scattered. Figure (d): a too big patch size leads to scattered borders and loss of spatially small segments.

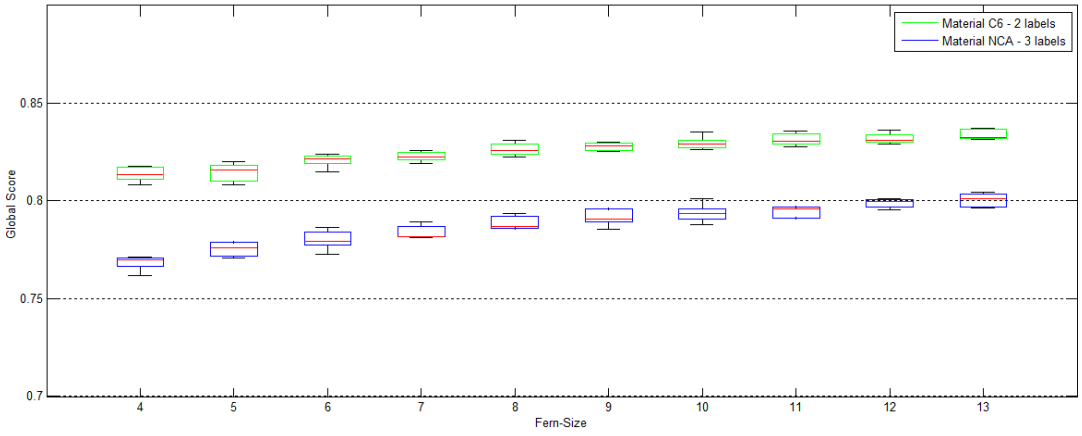



(a) Global Score

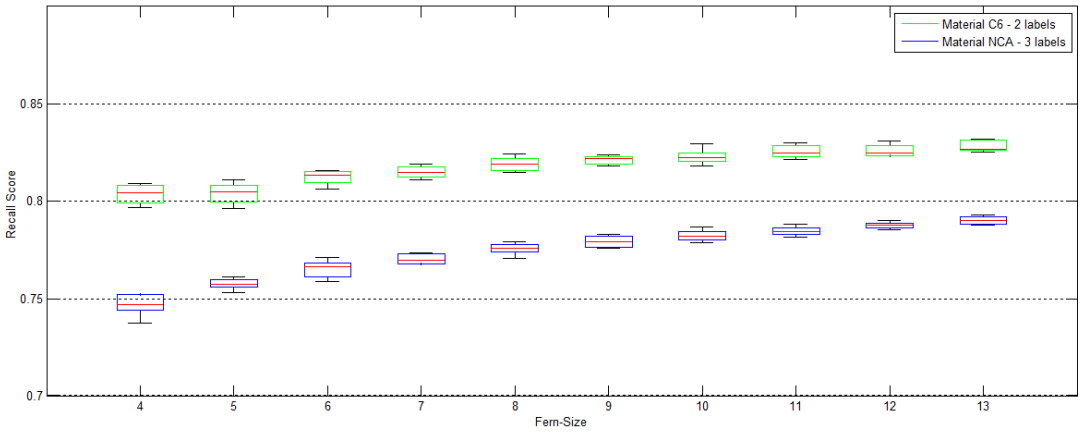



(b) Average Recall

Figure 4.4: Scores depending on the number of ferns: Figure (a) shows the global score; in Figure (b) the average recall is plotted. Both plots show, that an increase in the number of ferns up to 30 leads to significant better results. A further increase improves the score just minimal.




(a) Global Score




(b) Average Recall

Figure 4.5: Scores depending on the size of ferns: Figure (a) shows the global score; in Figure (b) the average recall is plotted. Both plots show, that an increase of the fern size leads to an increase of performance.

ferns with a size of 11 were used. No runtime evaluation was done because the patch size does not influence the runtime significantly.

### 4.1.2 Performance Gain Compared to the Original Fern Version

The original fern version proposed in [OCLF10] for keypoint recognition uses just the gray value intensity as feature channel and the comparison of two pixel values as binary test. It turned out that this version is not suitable for classification tasks. Without the modifications proposed in this work the result seems randomly scattered. Figure 4.7 shows illustrative examples for natural- and lithium-ion cell images.

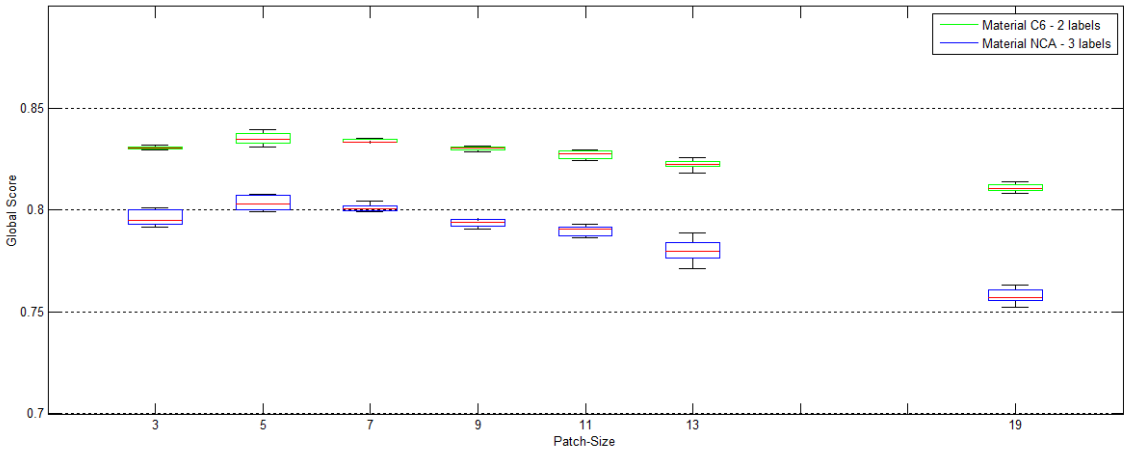
Based on the previous evaluations following parameters were chosen for the proposed system: A number of ferns of 30, a fern size of 10 and a patch size of $7 \times 7$. All in all 12 feature channels are used, which include the gray value intensity, the first and second order derivatives and the magnitude of the gradient. All channels are used twice, once filtered with a $3 \times 3$ maximum filter and once with a $3 \times 3$ minimum filter. The correlation between these feature channels is modeled implicitly by the ferns. The binary tests of the ferns include the comparison of a pixel value with a threshold; the comparison of the sum, respectively difference, of two pixel values with a threshold and the absolut difference of two pixel.

Figure 4.7 shows a comparison of the proposed and the original version of ferns via practical examples. The seed points, the ground truth labeling and the results of both versions are shown for a sample image of a lithium-ion electrode and a natural image from the Icg-bench. Figure 4.8 visualizes the performance gain achieved with the lithium-ion electrode images and the Icg-bench. All subfigures show a significant increase in performance of the proposed version of this work compared to the original one in [OCLF10]. Figure 4.8(a) and 4.8(b) are the results of a 5-fold cross validation as described in the beginning of 4.1. In Figure 4.8(c) the results of all images of the Icg-bench were used to create the boxplot. As this dataset contains a wide range of different images, the boxes are accordingly larger. The meaning of the size of the boxes is explained in the beginning of 4.1.

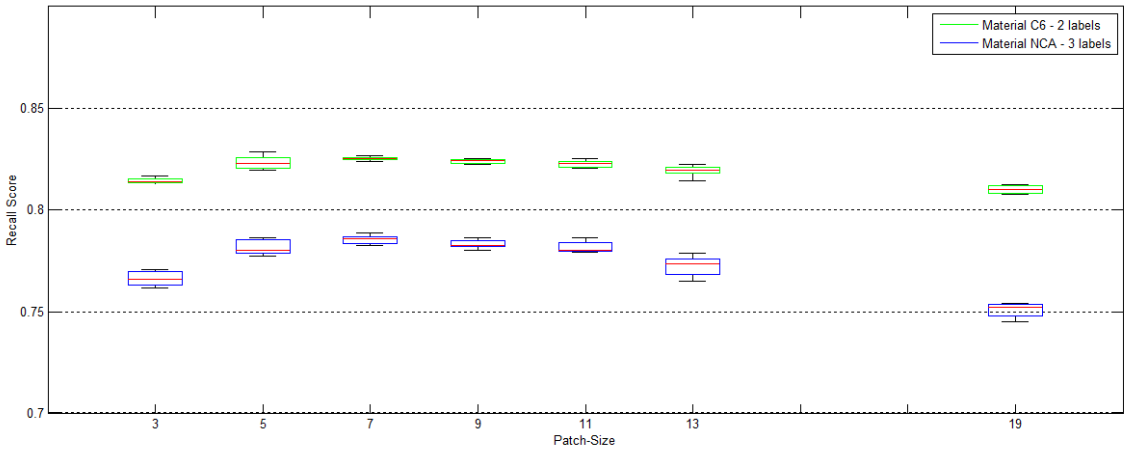
### 4.1.3 Segmentation

In the previous chapter the parameters for the ferns were determined and the performance of the classification was investigated. For a smooth result image a subsequent segmentation is necessary. The performance of the discrete and the continuous Potts model will be compared in this section. To solve the discrete Potts model the graph cut algorithm described in chapter 2.2.2 is used. For the analyses a license free implementation of the algorithms described in [BVZ01, BK04, KZ04] is used. To solve the continuous Potts model, a CUDA-based implementation of the algorithm proposed in [PCCB09] is used.

This chapter is divided into two parts. In the first section the performance of the proposed system on lithium-ion electrode FIB-SEM images is investigated. At the time when this master thesis was written just few of lithium-ion electrode images were available to the author. Also no public dataset for the special purpose of segmenting materials in lithium-ion electrode

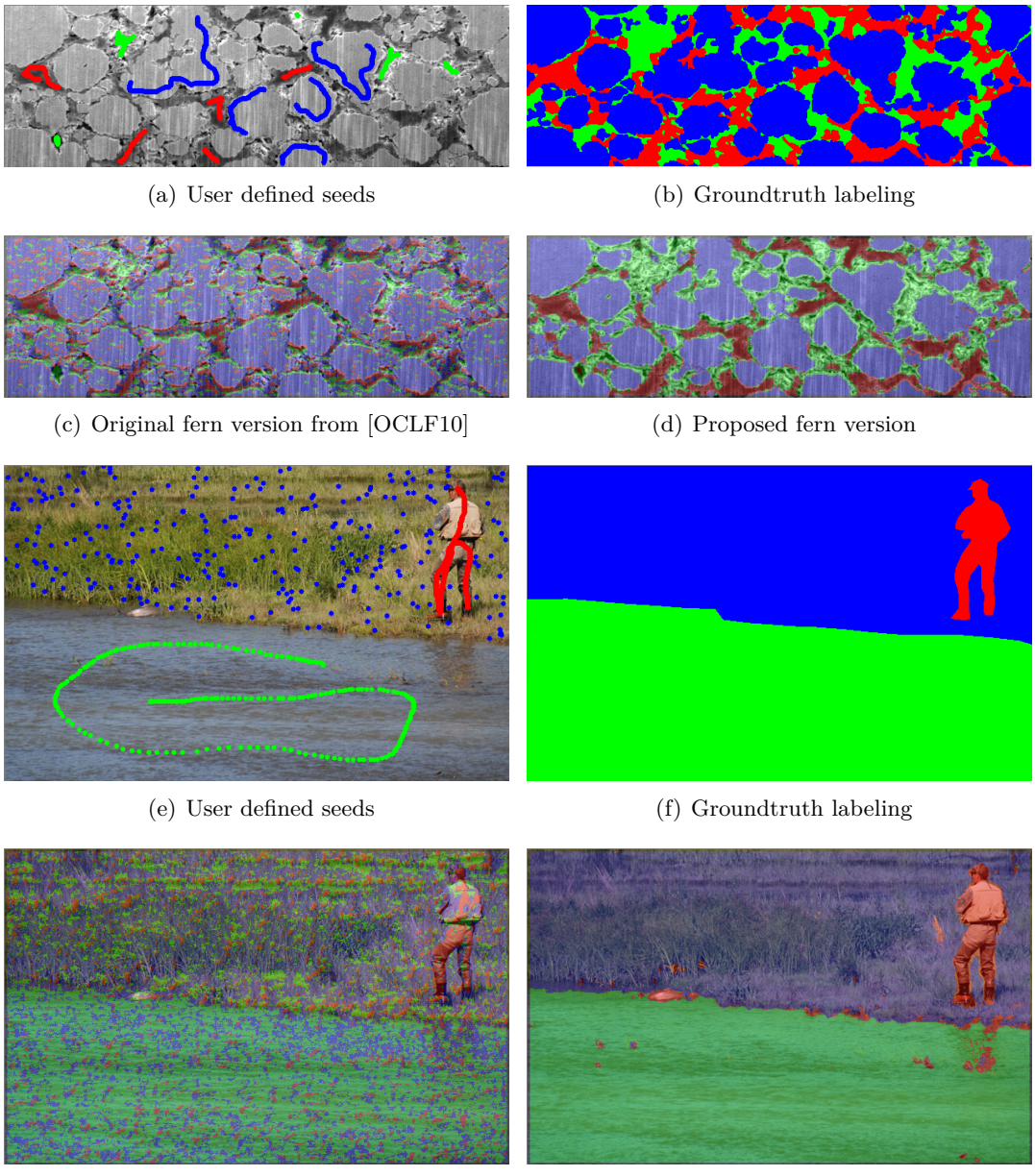



(a) Global Score




(b) Average Recall

Figure 4.6: Scores depending on the patch size: Figure (a) shows the global score; in Figure (b) the average recall is plotted.



(a) User defined seeds

(b) Groundtruth labeling

(c) Original fern version from [OCLF10]

(d) Proposed fern version

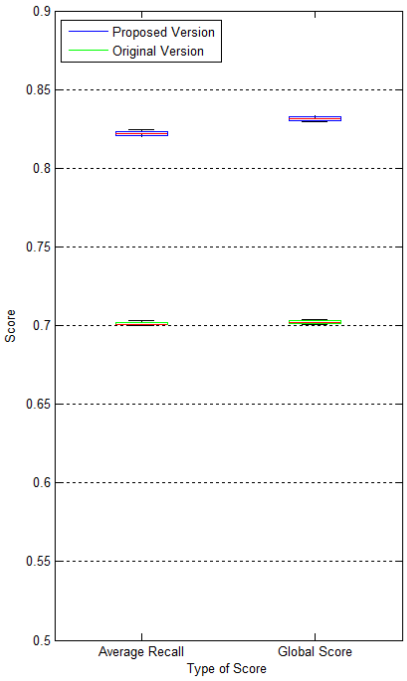
(e) User defined seeds

(f) Groundtruth labeling

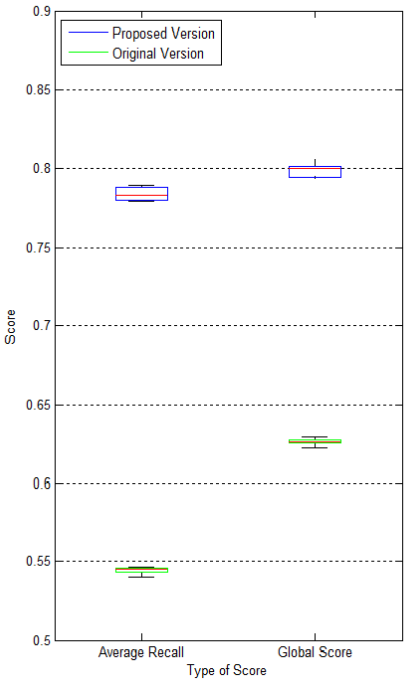
(g) Original fern version from [OCLF10]

(h) Proposed fern version

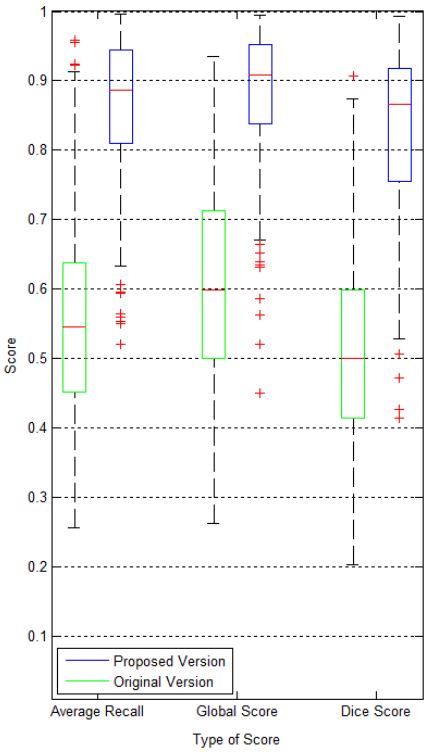
Figure 4.7: Classification results of the original fern version from [OCLF10] and the fern version proposed in this work. Figures (a) - (d) show an example image of a NCA-electrode. Figures (e) - (h) show one example of the Icg-bench. For both versions the same number of ferns, fern size and patch size was used. It is obvious that the unmodified version is not suited for classification tasks.




(a) $C_6$-electrode




(b) NCA-electrode




(c) ICG-Bench

Figure 4.8: Performance of the original fern version from [OCLF10] and the proposed fern version: On both cell materials (Figure (a), (b)) and on the ICG-Bench the proposed version of this work outperforms the original one. For a detailed description see the according text.

| | Model | |
|---|---|---|
| Score | discrete Potts | continuous Potts |
| Global pixel accuracy: | 0,8549 | 0,8578 |
| Average recall: | 0,8374 | 0,8377 |
| Dice score: | 0,8236 | 0,8263 |

Table 4.5: The table shows the average performance of the proposed system on images of lithium-ion electrodes.

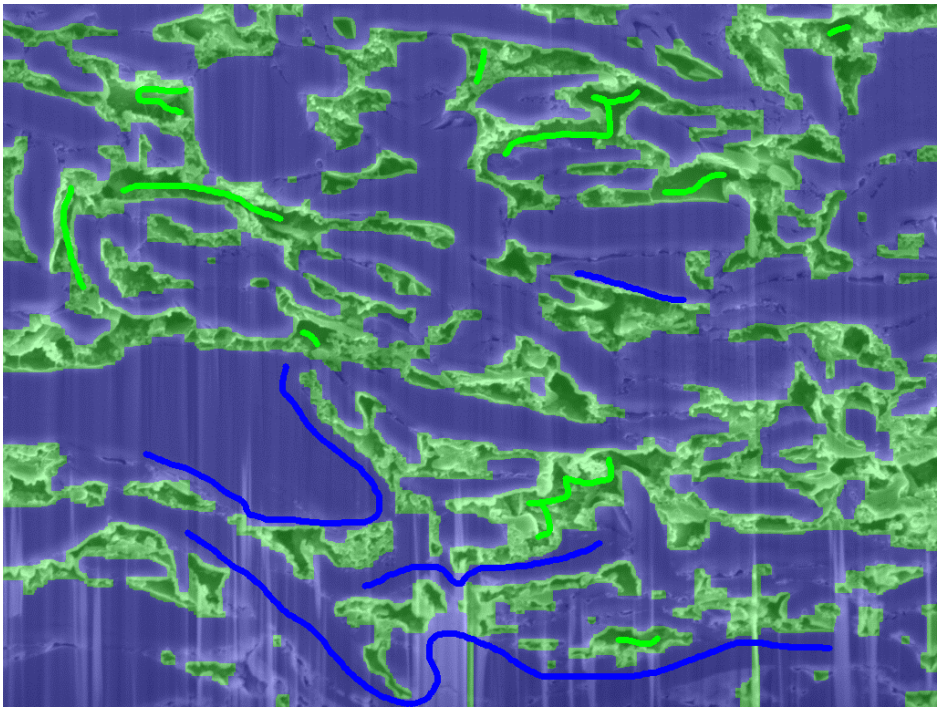
images is presently available. That is why the performance of the proposed system is also tested on natural images. The second part of this chapter shows the performance on the public available Icg-bench, proposed in [San10]. As the Icg-bench includes color images, the gray value intensity feature channel is replaced by three channels representing the image in the *Lab* color space.

### Lithium-Ion Cell

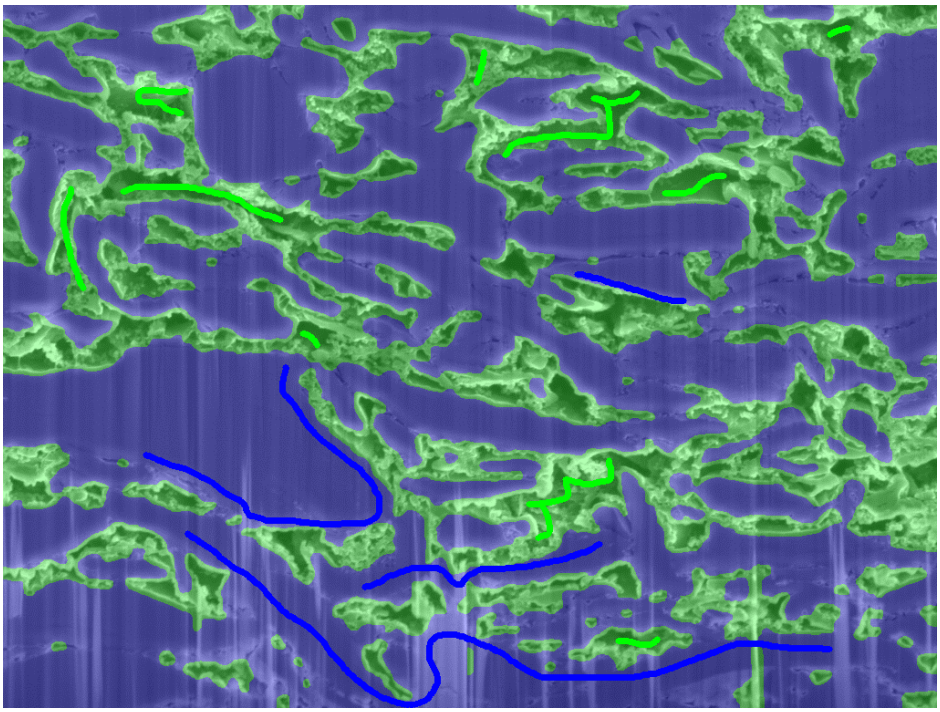
The available images show different lithium-ion electrodes which should be separated either in two or three materials. Experts in the field of lithium-ion batteries placed the seed points and labeled the ground truth image. For the evaluation the seed points of each image were used to train the ferns. Then the produced probability distribution was taken as input for the discrete and the continuous Potts model. For both implementations the best parameter configuration was chosen via a brute force search. In Table 4.5 the average of the best values for the global pixel accuracy, the average recall and the Dice score is given. In Figure 4.9 and 4.10 segmentation results for both versions are given. Figure 4.9 shows the result for a $C_6$ lithium-ion electrode. The artifacts caused by the discrete Potts model are well visible. For further processing (ellipse fitting) both segmentation results are equally suitable. Figure 4.10 shows sample results for the NCA-material. As the segmented particles are smaller compared to the $C_6$-material, the artifacts caused by the discrete Potts model are not that visible.

### ICG Bench

A record of the Icg-bench consists of the colored image in jpg-format, some seed points placed by a user and the according ground truth labeling. At first the seed points of each image were used to train the ferns. Then the produced probability distribution was taken as input for the discrete and the continuous Potts model. For both implementations the according parameters, which are adjustable by the user, were varied in reasonable ranges and the average scores over all records were calculated. For the discrete Potts model the parameter alpha $\alpha$ is determining the trade-off of the unary and the pairwise energy factors (2.44). The parameter lambda $\lambda$ is a weight factor in the continuous Potts model and influences the length of the perimeter of the segmented regions (2.46). Figures 4.11 and 4.12 show the global score, the average recall and the Dice score depending on these parameters. In Table 4.6 the best values for these scores

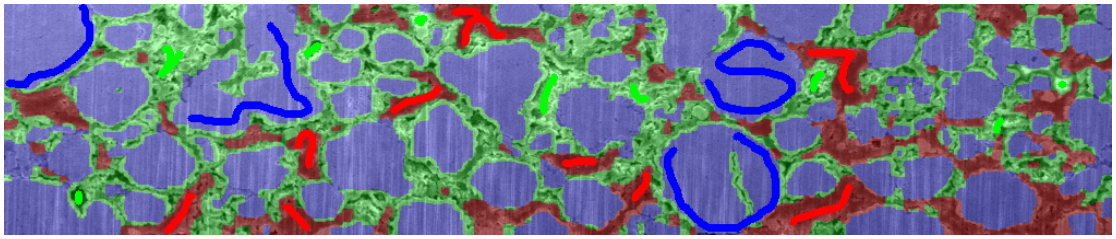


(a) $C_6$ sample segmentation with discrete Potts model

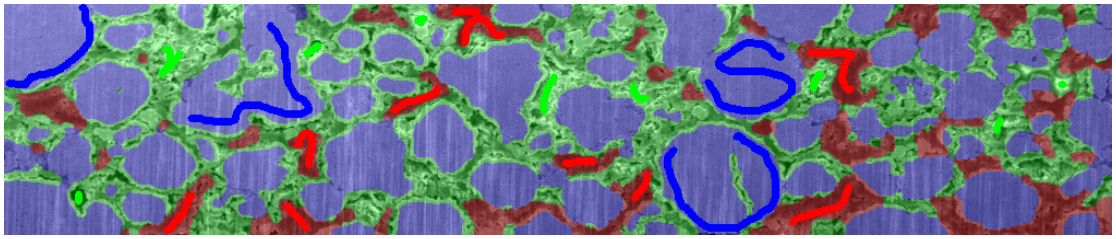


(b) $C_6$ sample segmentation with continuous Potts model

Figure 4.9: Segmentation results of the discrete and the continuous Potts model applied on an image of a $C_6$ lithium-ion electrode. In blue active material is marked and green are the pores. The artifacts of the segmented regions caused by the discrete model are well visible.



(a) NCA sample segmentation with discrete Potts model



(b) NCA sample segmentation with continuous Potts model

Figure 4.10: Segmentation results of the discrete and the continuous Potts model applied on an image of a NCA lithium-ion electrode. The active particles are marked in blue, the pores are green and the binder material is red. The particles are smaller compared to the $C_6$ material, that is why the artifacts of the discrete Potts model are not that visible.

| | Model | | | | framework of [San10] |
|---|---|---|---|---|---|
| | discrete Potts | | continuous Potts | | |
| | Score | $\alpha$ | Score | $\lambda$ | |
| Global pixel accuracy: | 0,9077 | 0.7 | 0,9066 | 1.65 | not available |
| Average recall: | 0,8781 | 0.7 | 0,8762 | 1.65 | not available |
| Dice score: | 0,8912 | 0.7 | 0,8898 | 1.65 | 0,9260 |

Table 4.6: The table shows the average performance of the proposed system on the Icg-bench.

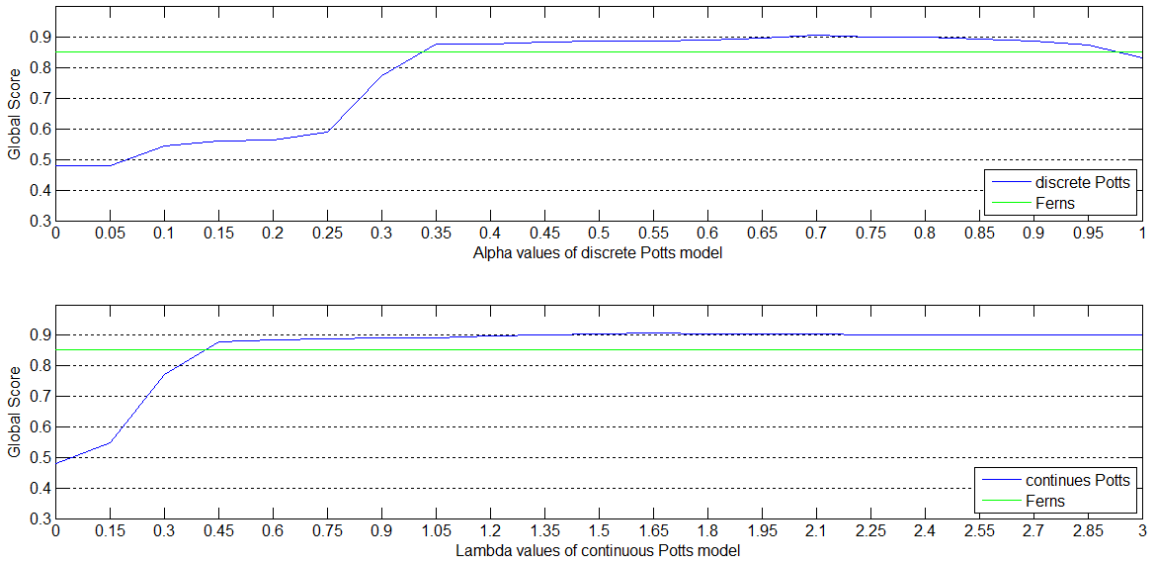
for both models are given. Also the Dice score of the interactive multi-label segmentation framework proposed in [San10] is shown. Although the proposed system of this work does not achieve the score of the framework in [San10], it still leads to comparable results. Figure 4.13 shows the segmentation results for the images with the highest performance. For every image the identification phrase within the Icg-bench and the achieved Dice score is given. It is also stated if the continuous or the discrete Potts model as segmentation framework led to the according score. For the proposed system of this work the difference between these two versions is insignificant. Figure 4.14 shows the segmentation results for the images with the lowest performance.

### 4.1.4 Ellipse Fitting

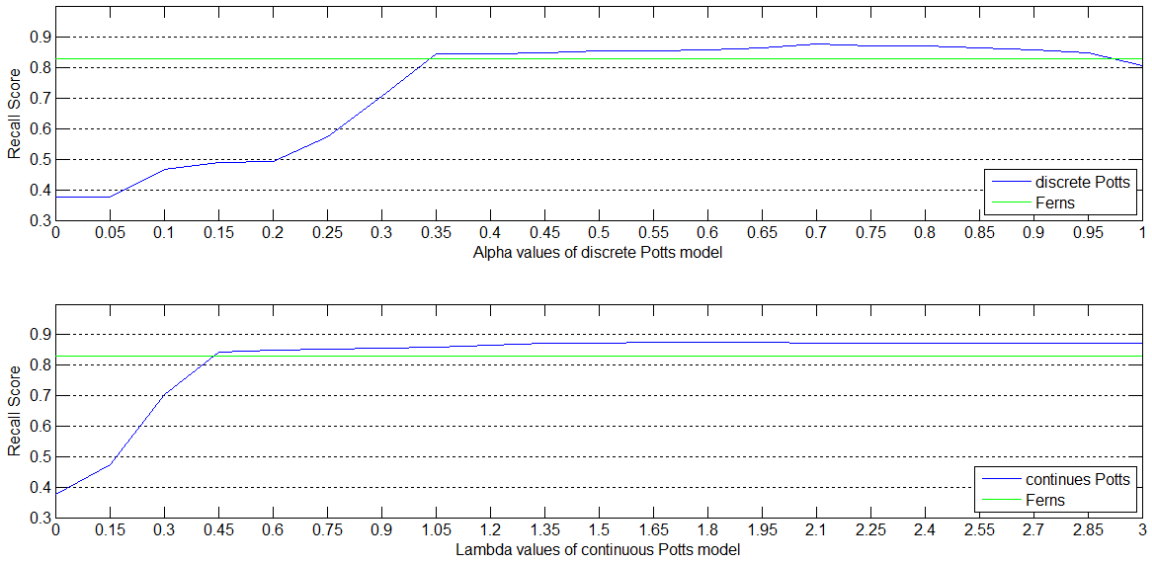
In section 3.1.2 two quality criteria, the Area-Approximation-Error $AAE$ and the Ellipse-Similarity-Factor $ESF$, are introduced to determine how good a particle is approximated by an ellipse. If an approximation does not fulfill certain user thresholds concerning these criteria, the approximated particle needs to be split with the proposed splitting algorithm. In general the two resulting particles should have a lower $AAE$ than the original one. The following experiment will show this fact. For a detailed description of the ellipse fitting algorithm see chapter 3.1.2, here just an overview of the procedure is given.

The aim of the statistical 3D reconstruction is to create particles of the size of the original ones but with a simplified shape. This means the difference in the size of the fitted ellipse to the size of the segmented particle can be used as quality measurement. In this work this difference is determined by the $AAE$. The ellipse fitting procedure works as follows: The user sets two thresholds; the maximum $AAE$ and the minimum $ESF$; then all found particles are approximated by ellipses. If an approximation does not give an $AAE$ lower than the maximum value, or an $ESF$ higher than the minimum value, the according particle is rejected and split in the next step. If an approximation fulfills all criteria the according ellipse is saved and the particle area is discarded for further processing.

Figure 4.15 shows the relation of the user set thresholds, the resulting average $AAE$ of all approximations and the number of ellipses. In the practical realisation the user adjusts both thresholds at the same time, for better traceability this experiment investigates the influence of every parameter on its own. Figure 4.15 displays the result for two different electrode materials. Each figure shows in the upper diagram the average $AAE$ depending on




(a) Global Score




(b) Recall Score

Figure 4.11: Each figure shows two diagrams, which show the achieved average score on the entire Icg-bench. The upper diagram shows the results for the discrete Potts model, depending on the weight parameter $\alpha$. The lower diagram shows the results for the continuous Potts model, depending on the weight parameter $\lambda$. For both segmentation methods the result of the preceding classification via Ferns is used as input. The achieved score of this classification is shown in green.

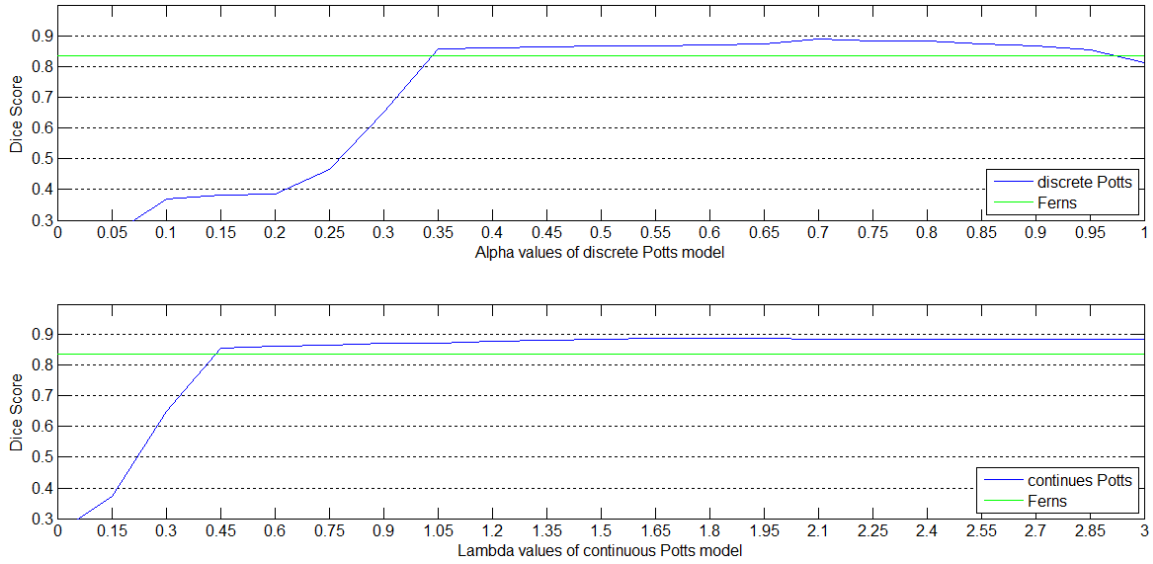
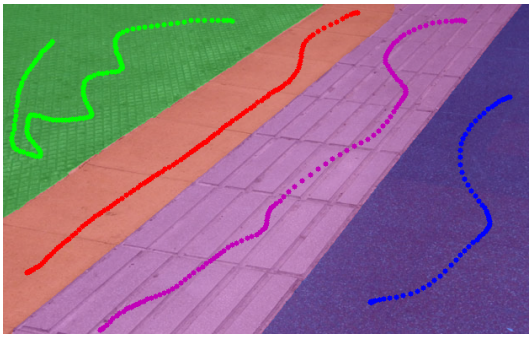
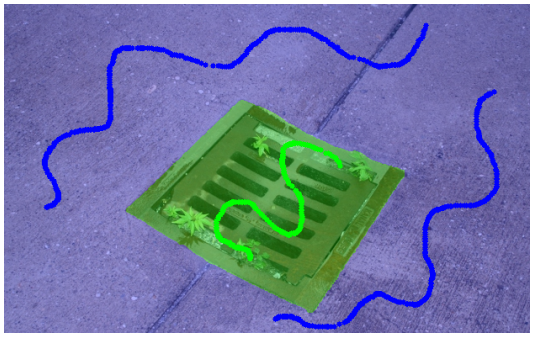



Figure 4.12: The two diagrams show the achieved average Dice score on the entire Icg-bench. The upper diagram shows the results for the discrete Potts model, depending on the weight parameter $\alpha$. The lower diagram shows the results for the continuous Potts model, depending on the weight parameter $\lambda$. For both segmentation methods the result of the preceding classification via Ferns is used as input. The achieved score of this classification is shown in green.


(a) ID: 0124_9000, Score: 0.9973, discrete

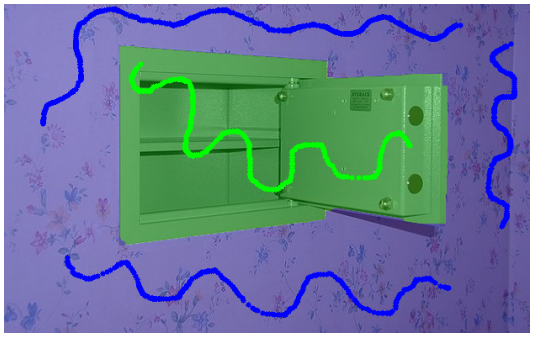


(b) ID: 0132_9000, Score: 0.9954, continuous

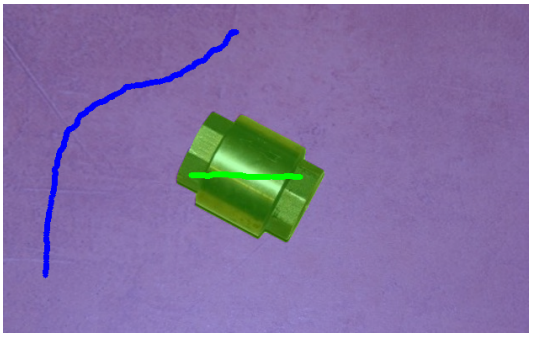


(c) ID: 0182_9050, Score: 0.9953, continuous

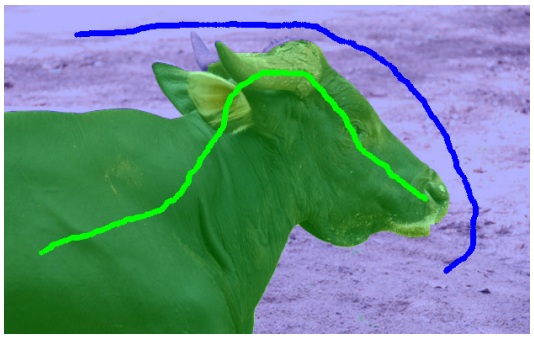
(d) ID: 0202_9050, Score: 0.9937, continuous



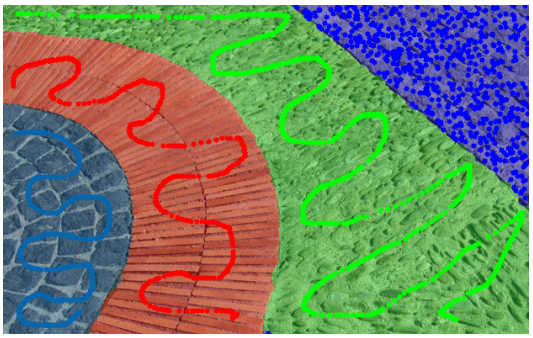
(e) ID: 0229_9050, Score: 0.9925, discrete



(f) ID: 0212_9060, Score: 0.9923, continuous

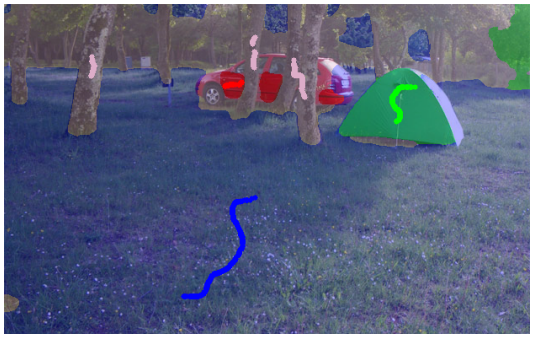


(g) ID: 0044_9060, Score: 0.9917, discrete

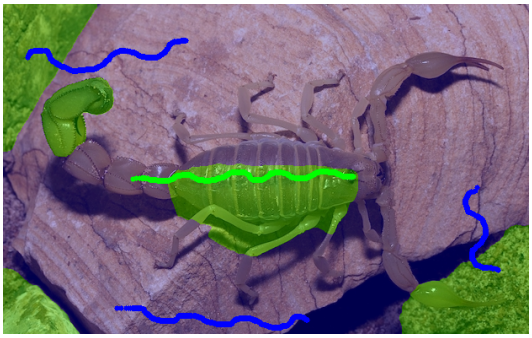


(h) ID: 0140_9025, Score: 0.9915, continuous

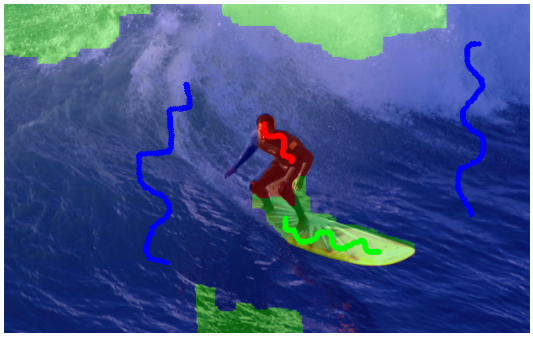
Figure 4.13: Segmentation results with the highest performance. For every image the identification phrase within the Icg-bench, the Dice score and the used version of the potts model (discrete or continuous) is given.



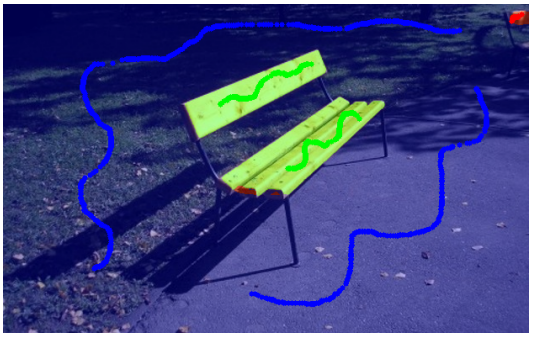
(a) ID: 0029_9050, Score: 0.6265, continuous



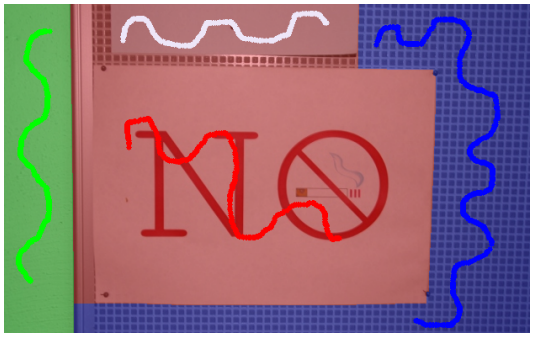
(b) ID: 0111_9050, Score: 0.6387, continuous



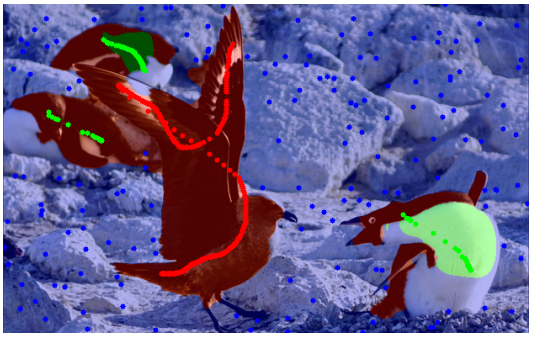
(c) ID: 0149_9050, Score: 0.6880, discrete



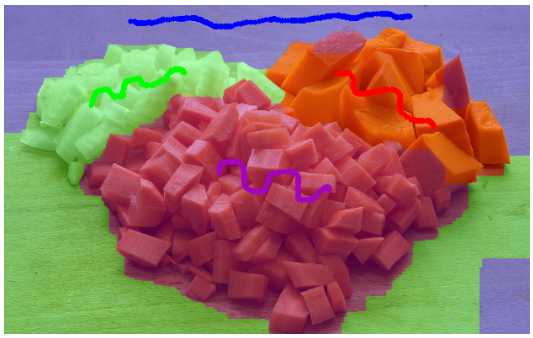
(d) ID: 0223_9050, Score: 0.6882, continuous



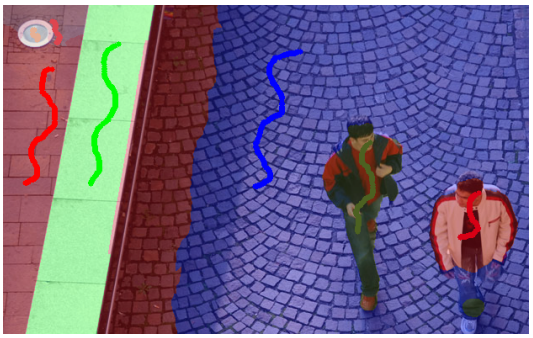

(e) ID: 0160_9050, Score: 0.6888, discrete



(f) ID: 0005_9020, Score: 0.6928, continuous



(g) ID: 0102_9050, Score: 0.7001, discrete



(h) ID: 0018_9050, Score: 0.7369, continuous

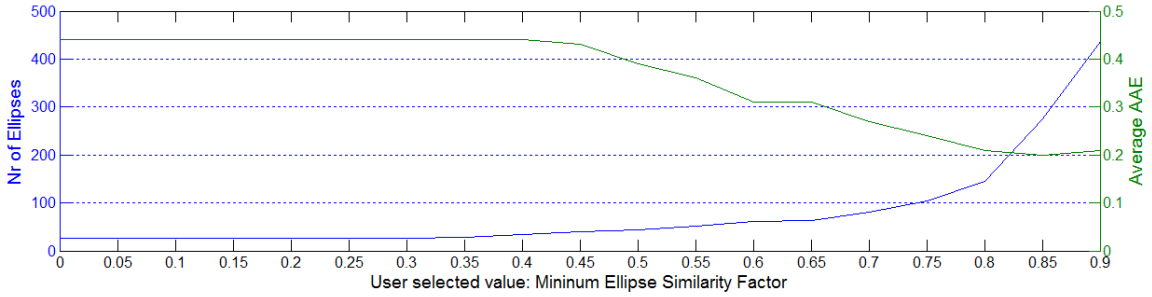
Figure 4.14: Segmentation results with the lowest performance. For every image the identification phrase within the Icg-bench, the Dice score and the used version of the potts model (discrete or continuous) is given.

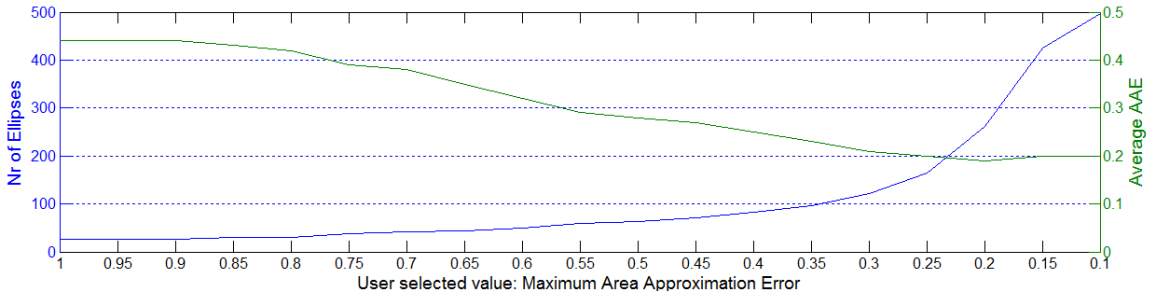
the minimum $ESF$ selected. The maximum $AAE$ value is set to 1 and therefore does not influence the result. It is visible that with increasing this minimum value, the average $AAE$ tends to decrease. This also means that the number of splits increases, seen in the figure implicitly by the increasing number of ellipses. The lower diagram of each figure shows the average $AAE$ depending on the maximum $AAE$ selected. Now the minimum $ESF$ value is set to 0, to not influence the result. The diagram shows, that a decreasing value for the maximum $AAE$ also decreases the achieved average $AAE$ of all ellipses by trend. As in the upper diagram this is reached with splitting the particles which leads to an increasing number of ellipses. For the two lowest values of the maximum $AAE$, in the diagram 0.15 and 0.1, the achieved average $AAE$ is higher than these thresholds. This results from the fact, that after a maximum number of turns the algorithm is stopped. This is necessary to prevent the system from getting into a practically infinite loop, if the user set thresholds can not be achieved. In general a two high number of ellipses is a sign of overfitting, which means in this particular problem domain that a large, nearly elliptic shaped, particle is split into several smaller ones. As it is an interactive framework, the user is in charge of selecting the proper thresholds for the particular use case.

## 4.2 Stochastic

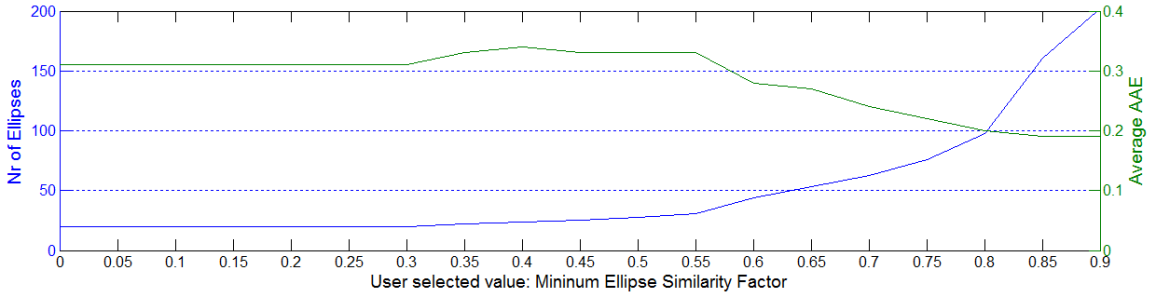
### 4.2.1 Reconstruction of Ellipsoids

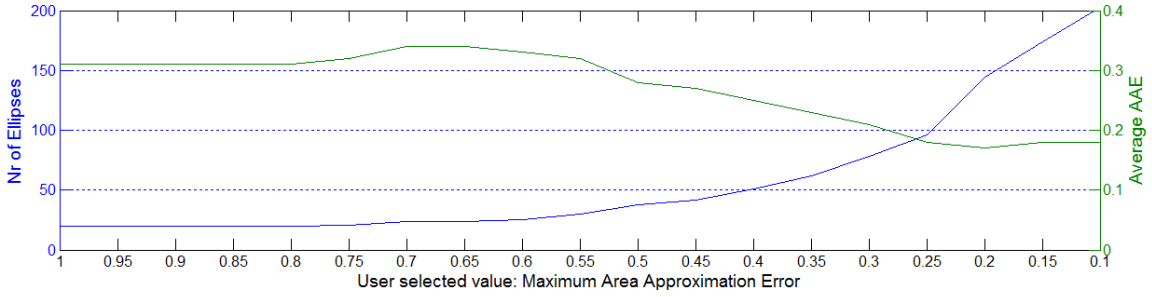
We assume a given distribution of ellipsoids in a certain volume. According to the assumptions of the proposed reconstruction algorithm the semi-principal axes $a, b, c$ of the ellipsoids have to fulfill $a \geq b, c = a$. The amount of each kind of ellipsoid is defined by its relative frequency. A horizontal cut is done through this volume and based on the resulting image of ellipses the sizes and relative frequencies of the ellipsoids are reconstructed. As $c = a$ per definition, for further investigations just the semi-principal axes $a$ and $b$ define the size of an ellipsoid. To measure the difference between the original and reconstructed distributions of ellipsoids two Euclidean distances are used, namely axes area error and relative frequency error. Each of the reconstructed ellipsoids is mapped to one of the original ellipsoids. This is done via a minimum distance calculation based on the lengths of the principal axes, which implies that not all of the original ellipsoids have to have a counterpart in the reconstructed distribution. It also implies that more than one reconstructed ellipsoid can be assigned to one original. The axes area error evaluates the quality of the reconstructed sizes of the ellipsoids, not taking any frequency distribution into account. For every ellipsoid an according area $ellipsoid_{area} = a \cdot b$ is defined by its axes. The axes area error is the root of the sum of the squared errors of the ellipsoid areas. The relative frequency error measures the quality of the reconstructed frequencies, not taking any size information into account. It is defined as the root of the sum of the squared errors of the relative frequencies. If two or more reconstructed ellipsoids are assigned to one original one, their frequencies are summed up before the error is calculated. If an original ellipsoid is not assigned with any reconstructed one, the according area and relative frequency of the not present counterpart is set to zero. Equations (4.5) and (4.6) show the calculation of the two errors for an example distribution given in Figure 4.16.




(a) Material $C_6$




(b) Material NCA

Figure 4.15: The two figures show the Area-Approximation-Error depending on the user set thresholds for two electrode materials. The upper diagram of each figure shows the dependency on the mininum Ellipse-Similarity-Factor, the lower diagram the dependency on the maximum Area-Approximation-Error. For more details see the according text.

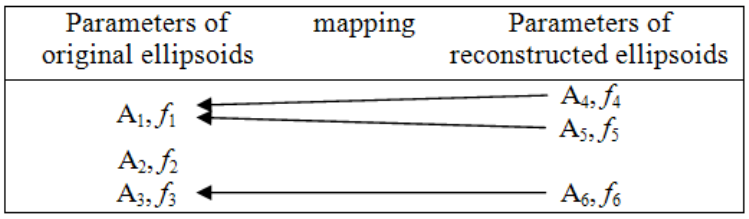


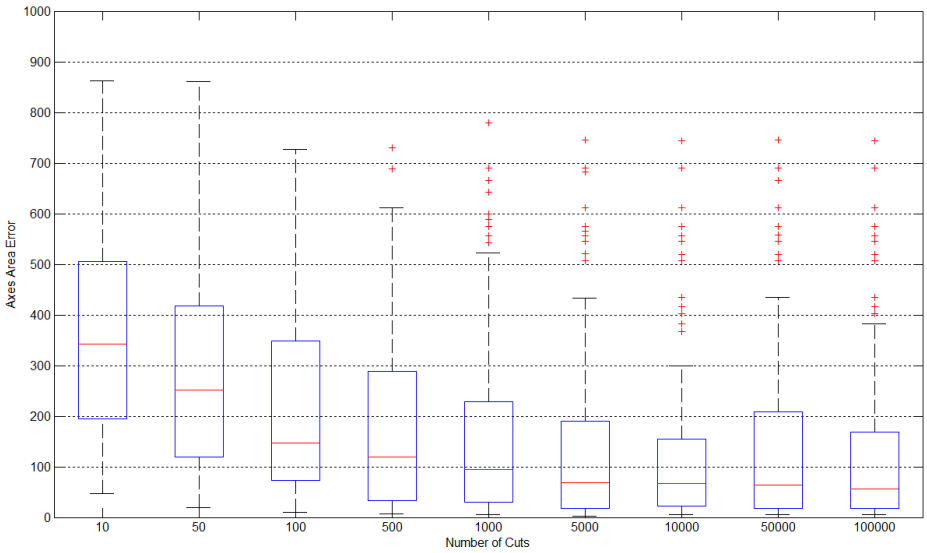

Figure 4.16: The graphic shows the parameters used for the error calculation. On the left side the original areas $A_i$ and relative frequencies $f_i$ are given. On the right side the parameters of the reconstructed ellipsoids are shown. The arrows denote the mapping of the ellipsoids resulting from a minimum distance calculation based on the lengths of the principal axes. The calculation of the according area axes error and relative frequency error is given in equations (4.5) and (4.6).

$$axes\_area\_error = \sqrt{(A_1 - A_4)^2 + (A_1 - A_5)^2 + {A_2}^2 + (A_3 - A_6)^2} \quad (4.5)$$

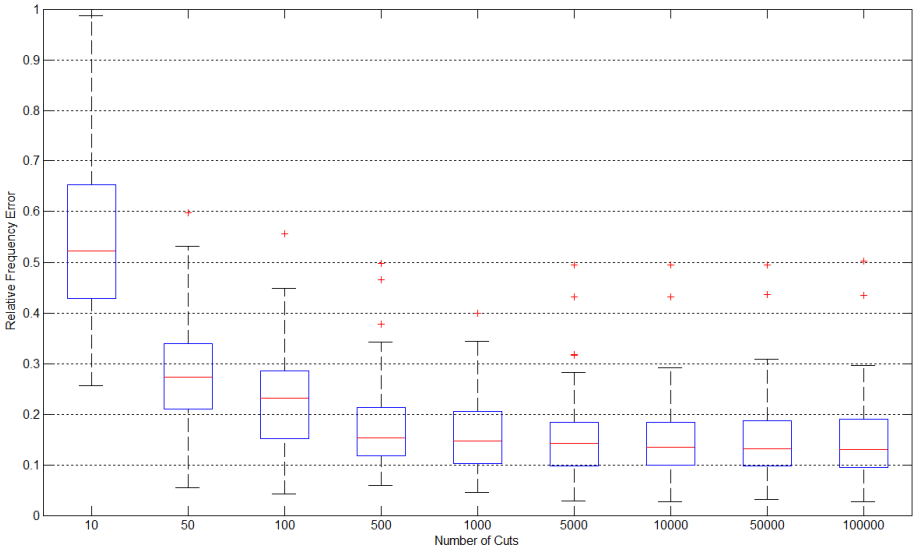
$$relative\_frequency\_error = \sqrt{(f_1 - [f_4 + f_5])^2 + f_2^2 + (f_3 - f_6)^2} \quad (4.6)$$

The quality of the reconstruction algorithm described in 3.2.1 mainly depends on the number of ellipses which were used to create the 2D histogram of the major and minor axes combinations. For this experiment 5 ellipsoids, defined by their semi-principal axes $a, b, c$ and relative frequencies, were randomly created. According to the assumptions of the algorithm the ellipsoids have to fulfill $a \geq b, c = a$. To keep the experiment manageable the semi-principal axis $a$ was restricted to $5.0 \leq a \leq 30.0$. These ellipsoids have been cut normally distributed over their height with a horizontal plane, which leads to one ellipse per cut. The number of cuts per ellipsoid was set according to the relative frequencies of them. Based on the resulting distribution of ellipses the proposed algorithm was used to reconstruct the ellipsoids. The sum of all cuts varied from 10 to 100000. In Figure 4.17 the resulting axes area error and relative frequency error is plotted as a function of the total number of cuts. For every number of cuts the experiment was done 100 times. Figure 4.17 shows the results visualized via a boxplot which shows for every number of cuts: the median; the 25th and 75th percentile as the borders of the box; the most extreme data points which are inside an interval of 99.3% coverage of the data via whiskers; and outliers outside this interval.

Figure 4.17 shows how big a sample histogram of ellipses should be to achieve the best results. Figure 4.17(b) shows that the relative frequency error decreases significantly until an amount of 500 ellipses. From this point upwards the error rate stabilizes. The axes area error also decreases significantly at the beginning. It stabilizes from about 1000 ellipses upwards. Practically this values can be used to determine how spatially big a cut through a lithium-ion electrode should be to achieve the best possible results with the proposed algorithm. The introduced sample images show about 100 ellipses for the NCA-material and about 120 ellipses for the $C_6$-material. The number of the ellipses is depending on the user settings; therefore no exact value can be stated.




(a) Axes Area Error

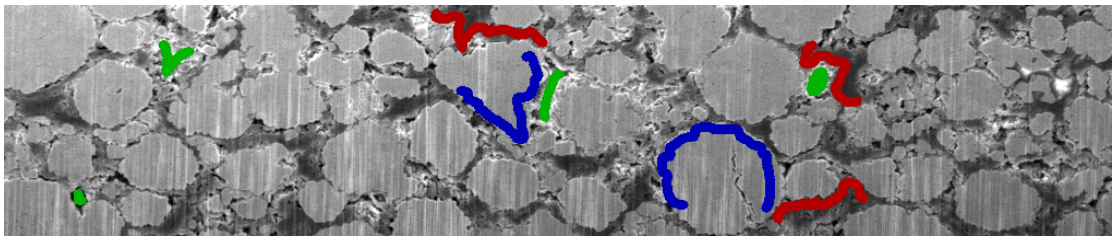



(b) Relative Frequency Error

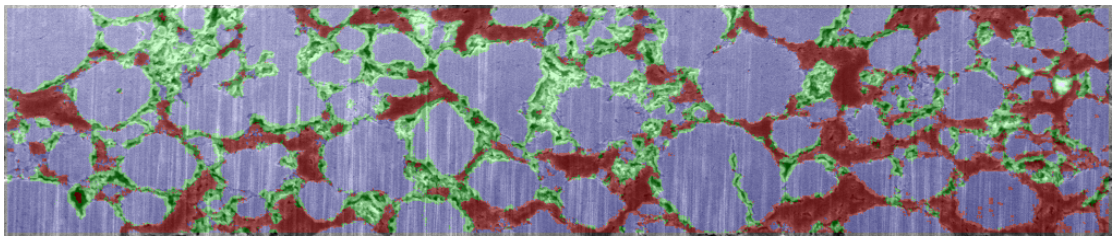
Figure 4.17: Reconstruction errors depending on the number of cuts: Figure (a) shows the axes area error; in Figure (b) the frequency error is plotted. For a detailed description see the according text.

## 4.3 Final System

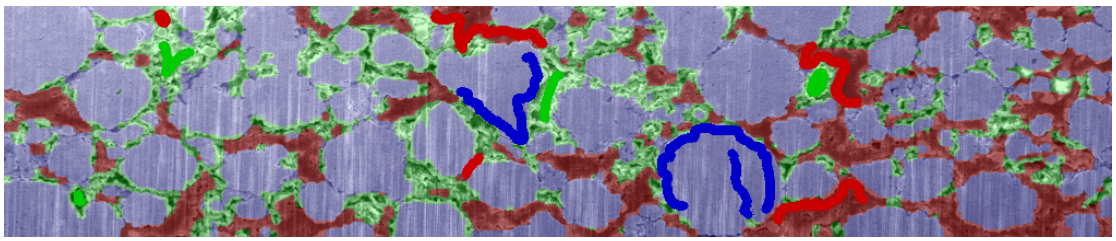
This section shows a complete work flow of the reconstruction of the 3D microstructure of a lithium-ion cell. In Figure 4.18(a) a sample image of a NCA-electrode is given. At the beginning the user marks characteristic areas for the three materials: active particles, binder material, respectively conductivity additives, and pores. Based on these seed points the ferns are trained and a probability distribution over all 3 labels for every pixel of the image is determined. Figure 4.18(b) shows the classification result, which is created by assigning every pixel the label with the highest probability. Then the probability distributions are used as input for the segmentation algorithm which is based on graph cuts. Figure 4.18(c) shows the segmentation result. The user can adjust the parameters of the graph cut algorithm and add some more seed points as hard constraints. If the user is satisfied with the segmentation result the ellipse fitting algorithm is started. The user has again the possibility to adjust the according parameters for the algorithm. It is also possible to set cuts through particles or define whole ellipses per hand. Figure 4.18(d) shows the segmented ellipses of the active material. The found ellipses are analysed and the according ellipsoids are reconstructed. Table 4.7 shows the result for the shown sample image. There, $a$, $b$, $c$ are the semi-principal axes in direction of the $x$, $y$, $z$ coordinate. The unit of the radii are pixels, which is sufficient for the statistical reconstruction. For further processing it is important to know the scale of the microscope images. As in this work described just one rotation angle is taken into account. When the ellipsoids are positioned in the discrete 3D volume, they are rotated randomly according to the normal distribution defined by the given $\mu$ and $\sigma$ values. For the final reconstruction the user defines a volume size, the system creates the ellipsoids according to the data shown in Table 4.7 and places them randomly within the volume. The last step is to rearrange the ellipsoids which is done with the algorithm described in Chapter 2.4.2. The result is shown in Figure 4.19(a). Through this 3D volume a vertical cut was made to get a visible comparison with the original image. Figure 4.19(b) shows the result for this cut. For better legibility the resulting ellipses were filled with the colours according to the labels of the segmentation. In Figure 4.19(c) the original sample image of the NCA-electrode is given for comparison.



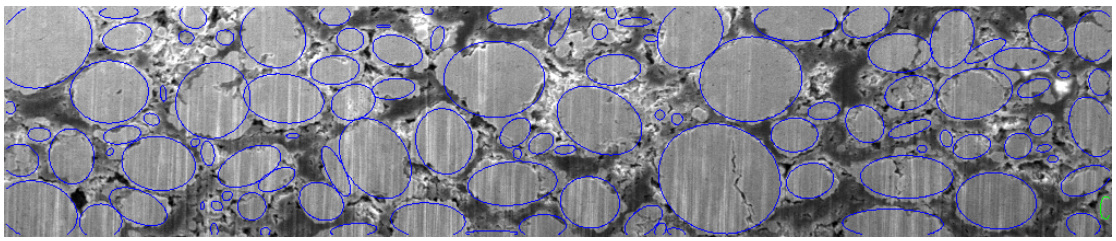
(a) Seed points



(b) Classification based on ferns

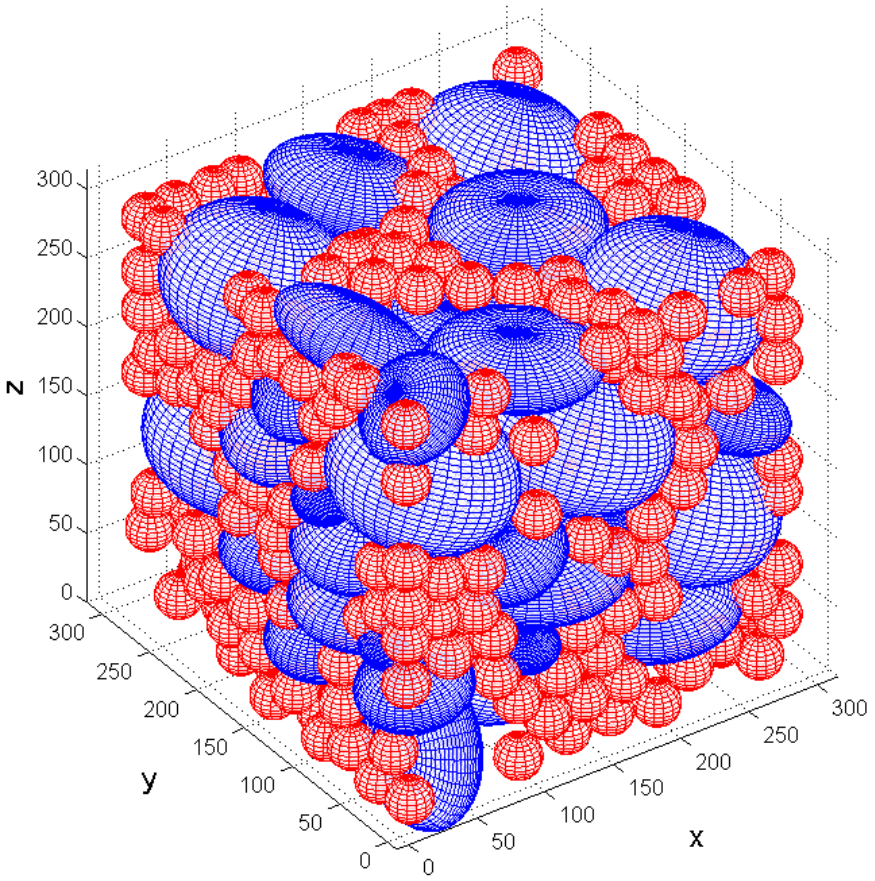


(c) Segmentation result

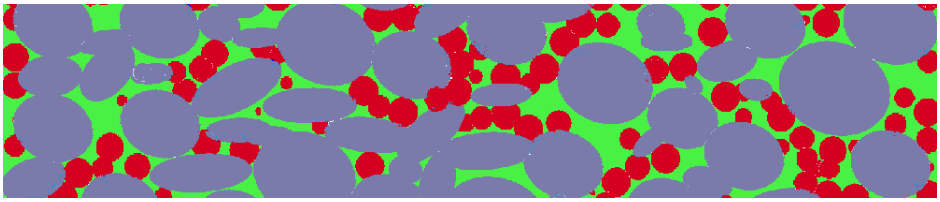


(d) Segmented ellipses

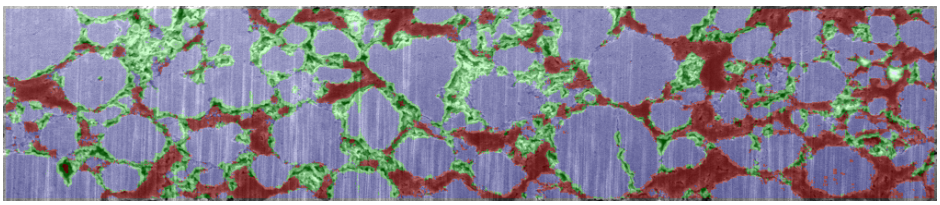
Figure 4.18: The Figure shows the first part of the work flow of the final system. Figure (a) shows a sample image for a NCA-electrode with user defined seed points. Active particles are marked in blue, binder material in red and pores in green. Figure (b) shows the classification result of the ferns. In Figure (c) the output of the segmentation algorithm is given. It is visible that the user added some more seed points, which are used as hard constraints for the segmentation algorithm. Figure (d) shows the segmented ellipses.




(a) Reconstruction result



(b) Cut through the reconstructed volume element



(c) Original sample image classified

Figure 4.19: Reconstruction result: In Figure (a) a reconstructed volume element is shown. The active particles are blue and sized and rotated according to the data of Table 4.7. Binder material is realized as smaller spheres and colored red. Through this volume a cut was made and filled with color for better legibility. Figure (b) shows this cut: active particles in blue, binder material in red and pores in green. Figure (c) shows the original classified image for comparison.

| radii [pixels] | | | relative frequency | angle y-coordinate | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | | $\mu$ | $\sigma$ |
| 55.5 | 55.5 | 48.0 | 0.26 | 31.1 | 18.2 |
| 45.5 | 45.5 | 12.5 | 0.11 | 0.6 | 2.2 |
| 44.5 | 44.5 | 22.0 | 0.15 | 29.4 | 37.6 |
| 40.5 | 40.5. | 27.0 | 0.16 | -3.0 | 18.1 |
| 29.5 | 29.5. | 28.5 | 0.14 | 6.8 | 43.5 |
| 18.0 | 18.0. | 15.0 | 0.18 | 8.8 | 34.0 |

Table 4.7: The table shows the data of the reconstructed ellipsoids from the sample image shown in Figure 4.18. $a$, $b$, $c$ are the semi-principal axes, $\mu$ and $\sigma$ define a normal distribution which is used to determine the angle for every created ellipsoid. For a detailed description see the according text.

# 5 Conclusion And Future Work

In this chapter a conclusion about the findings and outcomes of this work is given. Finally, future prospects are listed.

## 5.1 Conclusion

In this work we propose a novel method to reconstruct a statistically identical 3D volume of the microstructure of electrodes of lithium-ion cells. For the cost of approximating arbitrary shaped particles with an ensemble of touching ellipsoids, we are able to reconstruct the statistically identical microstructure based on just few scanning electron microscope images, depending on the spatial size of them. The created volume consists of ellipsoids with sizes and relative frequencies according to the particles in the original material. Such statistical 3D volumes are used in the field of lithium-ion cell research as base for different models of batteries. The costs of investigating the microstructure of lithium-ion electrodes can be lowered significantly, because just few SEM images are needed instead of few hundreds, like with other actually used methods.

In our work we used the concept of ferns, originally proposed for keypoint recognition in [OCLF10], for the classification of different materials in lithium-ion electrodes. We adapted the ferns for the classification task and achieved a significant performance gain. A comprehensive evaluation of the adaptations, especially of the used feature channels, is given. The proposed system uses a combination of classification via ferns and a subsequent segmentation based on the Potts model. Although the system is designed for segmenting different materials of lithium-ion cells, we achieve performance scores comparable to other current segmentation tools ([San10]) on publicly available data benches. We present an ellipse fitting algorithm to approximate the segmented materials with ellipses, which exploits the strong knowledge about the specific problem domain. We derived a method to reconstruct the sizes and relative frequencies of ellipsoids out of a distribution of ellipses, which originate from horizontal cuts through these ellipsoids. Finally the reconstructed ellipsoids are randomly arranged in a representative 3D volume element, representing the statistical identical 3D microstructure.

All findings have been implemented in a CPU-based interactive segmentation tool, which enables user interaction after every step. The user defined seed points are used to train the ferns for the classification of the image. The classification result, and if needed more hard constraints defined by the user, are used as input for the segmentation algorithm. The ellipse fitting algorithm can be influenced by two parameters which adjust the quality of the found ellipses. Furthermore the user can specify ellipses by hand or discard them. For the reconstruction a user defined threshold needs to be set which controls the amount and

similarity of the reconstructed ellipsoids. Finally, the user specifies the size of a volume and the microstructure is created according to the previous findings within it.

## 5.2 Future Work

The implementation of the proposed system is CPU-based to not be restricted to hardware configurations. The aim of this work was to show the feasibility of the reconstruction algorithm and the use of ferns for image classification. This has the drawback of a long computation time; for user comfort a GPU-based implementation would be desirable.

For the reconstruction active particles and binder material are approximated via ellipsoids. The sizes and frequencies of the active particles are derived from the found ellipses, which leads to a good approximation. The binder material is realized via small spheres and arranged between the active particle ellipsoids. As future work the shape of this binder material should be reconstructed more accurate. In fact this material does not have a geometrical form; it is pressed between the active particles and gathers mainly at narrow gaps among them. To get a more accurate model of a lithium-ion electrode, this fact should be considered while reconstructing the 3D volume.

The positioning of the ellipsoids in the discrete 3D volume is realized via the rearranging algorithm described in Chapter 2.4. This algorithm moves ellipsoidal particles apart and tends to keep small overlapping volumes. The size of these volumes is neither controllable nor is any information given about it. To get a reconstruction which is more accurate, a method should be used which offers the possibility to adjust the ratio of overlapping volumes and measures the overlapping volume size. To find such a method was not in the scope of this master's thesis. This results from the fact that the focus of this work was put on the segmentation of the active particles and the statistical analysis and reconstruction.

# List of Abbreviations

| | |
|---|---|
| **FIB** | Focus ion beam |
| **SEM** | Scanning electron microscopy |
| **SEI** | Solid electrolyte interphase |
| **XAFS** | X-ray absorption fine-structure spectroscopic |
| **SVM** | Support vector machine |
| **MRF** | Markov Random Field |
| **CRF** | Condiotional Random Field |
| **ICM** | Iterated Conditional Modes |
| **ESF** | Ellipse Similarity Factor |
| **AAE** | Area Approximation Error |

# Bibliography


[AG97] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[Bal81] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[Bel06] C.M. Bishop and SpringerLink (Service en ligne). *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[Bes86] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986.

[BFSO84] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[Bre96] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[BVZ01] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[BW04] P.B. Balbuena and Y. Wang. *Lithium-ion batteries: solid-electrolyte interphase*. Imperial College Pr, 2004.

[CSK11] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. TechReport MSR-TR-2011-114, Microsoft Research, 2011.

[CV95] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CWL+07] Y.H. Chen, C.W. Wang, G. Liu, X.Y. Song, VS Battaglia, and A.M. Sastry. Selection of Conductive Additives in Li-Ion Battery Cathodes. *Journal of the Electrochemical Society*, 154:978–986, 2007.

[DBK+97] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

[DBST02] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1):225–254, 2002.

[Dic45] L.R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[DK05] K.B. Duan and S. Keerthi. Which is the best multiclass svm method? an empirical study. *Multiple Classifier Systems*, pages 732–760, 2005.

[EJCIT11] Moses Ender, Jochen Joos, Thomas Carraro, and Ellen Ivers-Tiffee. Three-dimensional reconstruction of a composite cathode for lithium-ion cells. *Electrochemistry Communications*, 13(2):166 – 168, 2011.

[FCK09] G.T.K. Fey, Y.G. Chen, and H.M. Kao. Electrochemical properties of LiFePO4 prepared via ball-milling. *Journal of Power Sources*, 189(1):169–178, 2009.

[Fri02] J.H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[FS95] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[GG84] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[HLC08] K.C. Hsiao, S.C. Liao, and J.M. Chen. Microstructure effect on the electrochemical property of Li4Ti5O12 as an anode material for lithium-ion batteries. *Electrochimica Acta*, 53(24):7242–7247, 2008.

[Ho95] T.K. Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[IU05] Y. Itou and Y. Ukyo. Performance of LiNiCoO2 materials for advanced lithium-ion batteries. *Journal of power sources*, 146(1-2):39–44, 2005.

[KSK+08] M. Klodt, T. Schoenemann, K. Kolev, M. Schikora, and D. Cremers. An experimental comparison of discrete and continuous shape optimization methods. *Computer Vision–ECCV 2008*, pages 332–345, 2008.

[KZ04] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[Li95] SZ Li. Markov random field models in computer vision. *Lecture Notes in Computer Science. Springer*, 1995.

[MW05] Kai Moeller and Martin Winkler. *Script Primaere und wiederaufladbare Lithium-Batterien*. Script fuer Praktikum Anorganisch-Chemische Technologie, 2005.

[NL11] S. Nowozin and C.H. Lampert. *Structured learning and prediction in computer vision*. Now Publishers, 2011.

[OCLF10] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):448–461, 2010.

[PCCB09] T. Pock, A. Chambolle, D. Cremers, and H. Bischof. A convex relaxation approach for computing minimal partitions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 810–817. IEEE, 2009.

[Pla99] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[Pot52] R.B. Potts. Some generalized order-disorder transformations. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109. Cambridge Univ Press, 1952.

[San10] Jakob Santner. *Interactive Multi-Label Segmentation*. PhD thesis, Graz University of Technology, October 2010.

[Sch90] R.E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[Sha01] C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[SM10] C. Sutton and A. McCallum. An introduction to conditional random fields. *Arxiv preprint arXiv:1011.4088*, 2010.

[SN04] Venkat Srinivasan and John Newman. Discharge model for the lithium iron-phosphate electrode. *Journal of The Electrochemical Society*, 151(10):A1517–A1529, 2004.

[THJA04] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[Wal10] Gernot Walzl. *Stochastische Rekonstruktion der 3-dimensionalen Mikrostruktur von Lithium-Ionen-Zellen*. Master thesis, Technical University Graz, 2010.

[WB04] M. Winter and R.J. Brodd. What are batteries, fuel cells, and supercapacitors? *Chemical reviews*, 104(10):4245–4270, 2004.

[WCBH10] J.R. Wilson, J.S. Cronin, S.A. Barnett, and S.J. Harris. Measurement of three-dimensional microstructure in a licoo2 positive electrode. *Journal of Power Sources*, 2010.