Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.

http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Over-Segmentation of 3D Medical Image Volumes based on Monogenic Cues

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Markus Holzer
Matrikelnummer 0406109

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: a.o.Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Assoc.Prof. Dipl.-Ing. Dr.techn. Georg Langs

Wien, 12.04.2014

(Unterschrift Markus Holzer)

(Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Over-Segmentation of 3D Medical Image Volumes based on Monogenic Cues

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Medical Computer Science

by

Markus Holzer
Registration Number 0406109

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: a.o.Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: Assoc.Prof. Dipl.-Ing. Dr.techn. Georg Langs

Vienna, 12.04.2014

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Markus Holzer
Donaustadtstraße 30/6/31, 1220 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Markus Holzer)

Acknowledgements

I would like to express my gratitude for everyone supporting me in completing the thesis. Especially René Donner and Georg Langs for assisting me with expert advices and my colleagues, Matthias, Markus, Joachim, Johannes, Michaela, Albert and Sarah, who helped me solving many problems and made the numerous working ours at the office a very pleasant time. I would also like to thank Robert Sablatnig for supervising this thesis.

Furthermore i am very grateful for the support and patience of my family during my studies, encouraging me and helping me to focus on my thesis. I would also like to thank Markus, Martin and Martin for giving me motivation during my studies and finally i want to thank Bianca for her encouragement and sympathy that motivated me to finish writing this thesis.

Abstract

In this thesis a novel approach to compute a 3D over-segmentation (supervoxels) for radiological image data is presented. It allows to cope with the high levels of noise and low contrast encountered in clinical data such as Computed Tomography (CT), Optical Coherence Tomography (OCT) and Magnetic Resonance (MR) images.

The method, *MonoSLIC*, employs the transformation of the image content to its monogenic signal as primal representation of the image. The phase of the monogenic signal is invariant to contrast and brightness and by selecting a kernel size matched to the estimated average size of the superpixels it highlights the locally most dominant image edge. Employing an agglomeration step similar to the one used in SLIC-superpixels yields superpixels/-voxels with high fidelity to local edge information while being of regular size and shape.

The proposed approach is compared to state of the art over-segmentation methods on the real-world images of the 2D Berkley Segmentation Dataset (BSD) converted to gray-scale, as well as on challenging 3D CT and MR volumes of the VISCERAL dataset. It yields a highly regular, robust, homogeneous and edge-preserving over-segmentation of the image/volume while being the fastest approach. For 3D volumes the method is 3 times faster than the state of the art. Due to its invariance to contrast and brightness it yields 11% higher recall rate when dealing with MR and CT volumes. There is also no parameter that needs to be tuned, increasing the usability of the method.

Kurzfassung

Diese Arbeit präsentiert den neuartiger Ansatz *MonoSLIC* zur Übersegmentierung von radiologischen Bild- und Volumsdaten mit Fokus auf Geschwindigkeit, Robustheit und Benutzerfreundlichkeit. Das Ziel ist die einfache Anwendung auf Daten die von Computer Tomographien (CT), Optischer Kohärenten Tomographien (OCT) sowie von Magnet Resonanz (MR) Tomographien erzeugt werden.

Dazu werden die Daten als Monogenic-Signal in eine neue Repräsentationsebene transformiert, welche aus den 3 unabhängigen Teilen Amplitude, Orientierung und Phase besteht. Zur Berechnung der Übersegmentierung wird nur die Phase, welche Information über die lokalen Strukturen erhält, verwendet. Diese wird so abgebildet, dass die lokal dominierende Kante sichtbar ist. Das anschließende Segmentierungsverfahren ist ähnlich dem SLIC-Superpixels Algorithmus. Mit speziell optimierter k-means Clusterbildung werden Superpixel/-voxel erstellt die eine regelmäßige Größe und Form haben und die Objektkanten erkennt.

Zur Auswertung wird der Ansatz mit anderen Algorithmen die dem neuesten Stand der Technik entsprechen auf den Bildern des Berkley Segmentation Datensatzes (BSD) sowie den medizinischen Bildern und Volumen des VISCERAL Datensatzes verglichen. Die Resultate zeigen, dass der Ansatz eine regelmäßige und einheitliche Übersegmentierung erstellt, welche auf den medizinischen Volumen um 11% genauer und 3 mal schneller als vergleichbare Methoden ist. *MonoSLIC* benötigt außerdem keinen zusätzlichen Parameter zu der Anzahl der gewünschten Segmente und ist unempfindlich zu Bildverschmutzungen durch Rauschen.

Contents

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Problem Statement 4
- 1.3 Aim and Methodical Approach 5
- 1.4 Structure of Work 5

2 State of the Art **7**
- 2.1 Overview 7
  - 2.1.1 Definitions 9
  - 2.1.2 Images 9
- 2.2 Discussion 10
- 2.3 Graph-Partition-Based Methods 11
  - 2.3.1 Felzenswalb - Efficient Graph-Based Image Segmentation 11
  - 2.3.2 Mori - Normalized Cuts 13
  - 2.3.3 Veksler - Superpixels in an Energy Optimization Framework 17
- 2.4 Gradient-Ascent-Based Methods 20
  - 2.4.1 Engel - Medial Features for Superpixel Segmentation 20
  - 2.4.2 Achanta - Simple Linear Iterative Clustering 23
- 2.5 Other Superpixel Algorithms 25
- 2.6 Comparison of the State of the Art Approaches 25
- 2.7 Summary 26
- 2.8 Limitations of State of the Art 27

3 Monogenic SLIC **29**
- 3.1 Preprocessing 29
- 3.2 Information Representation 32
- 3.3 Segmentation 33
- 3.4 Properties 34
- 3.5 Comparison 35
- 3.6 Summary 35

4 Results **41**
- 4.1 Evaluation Measures 41

4.2 Datasets 43
4.3 Parameter Choice 46
4.4 Results on 2D Images 47
  4.4.1 Berkley Image Segmentation Database - BSD 47
  4.4.2 Annotated Medical Images - 2D VISCERAL 53
4.5 Results on 3D Volumes 55
  4.5.1 3D Fetal 55
  4.5.2 3D Aorta 56
  4.5.3 3D VISCERAL 56
4.6 Runtimes 59
4.7 Summary 60

5 Discussion **63**
5.1 Monogenic SLIC Properties 63
5.2 Monogenic SLIC Performance Scaling 64
5.3 Recall, Noise and Regularity 66
5.4 Future Work 70
5.5 Summary 72

6 Conclusion **73**

Bibliography **75**

CHAPTER 1

Introduction

This chapter starts by introducing the motivation of the thesis, beginning with the properties of the human perceptual system, the difficulties of humans finding the correct segmentation of an image and how image processing methods address these problems. In the next section the exact problem of over-segmentation is defined, followed by the aim of the work and the methodical approach. The chapter then concludes with the structure of the work.

1.1 Motivation

The human perceptual system is a complex and powerful tool resulting from evolution [22]. Together with the experience stored in the individual's brain it allows to detect objects and understand the scene viewed within fractions of a second. Wertheimer 1923 [51] tries to understand and quantify the complex human perceptual system and states basic laws that he identifies as significant. The first is the *factor of proximity*, saying that underlying parts are organized by their proximity creating objects and groups. This is true not only for visual but also for auditory organization (e.g. rhythms in music). The second is named *factor of similarity*, describing that similar parts are organized together. In terms of auditory organization this is represented by the grouping of soft and loud beats. The third law is called the *factor of continuity*, that lets the human mind recognize objects (e.g. circle) even if they are partially occluded. Visual examples of the three laws are shown in Figure 1.1. All these factors are combined to either strengthen or weaken the grouping properties. Additional laws are stated by Wertheimer [51], which are out of scope of this thesis.

In [3] it is shown that the performance of humans carrying out visual perceptual tasks can be improved by training. This indicates that the visual system is based on every individual's experience. As a result of this, when presenting one image to different persons, giving them the visual perceptual task to segment all the objects in the image, no segmentation will look the same. This can be observed in Figure 1.2 where five people are given the task of annotating all object boundaries in the image. The fact that there already is variation in human perception

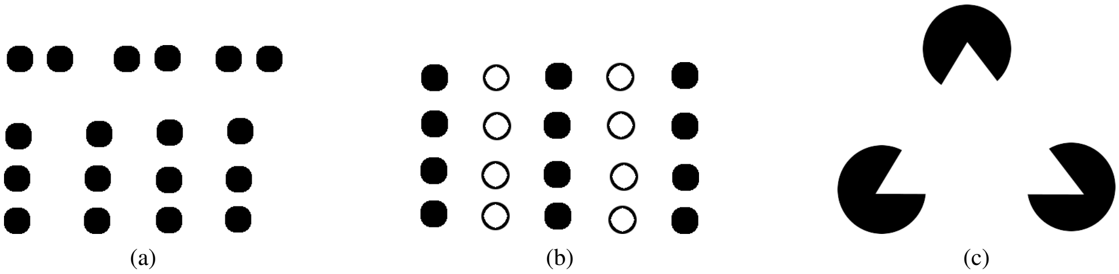

(a) (b) (c)

Figure 1.1: Examples for the basic laws of Wertheimer with (a) factor of proximity, (b) factor of similarity and (c) factor of continuity.

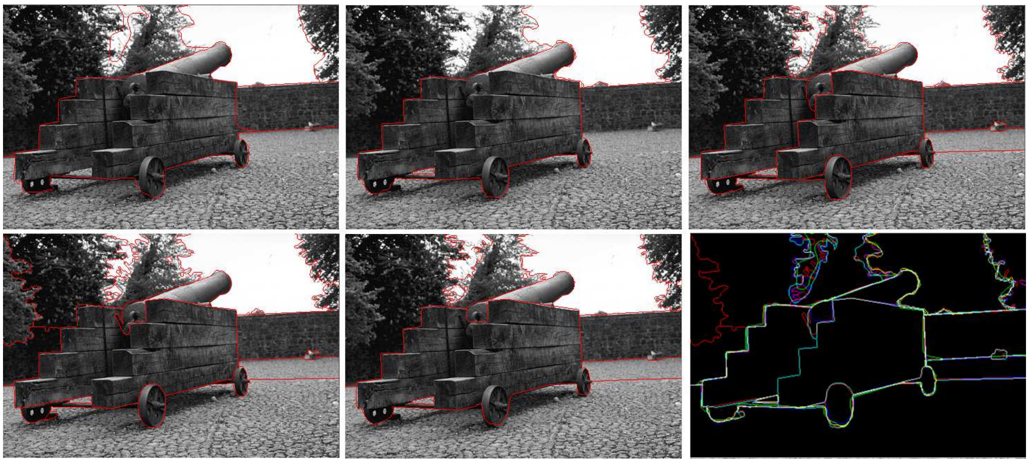


Figure 1.2: Object annotations of an image included in the Berkley Segmentation Database [34] from five different persons. In the image on the bottom right all segmentations are merged, each segmentation is shown in its own color to display variability.

between every individual is a hint to the complexity of designing a computer that attempts to do the same. The segmentations in Figure 1.2 are created as part of the Berkley Segmentation Dataset (BSD) [34] which is designed to quantify the quality of such computational attempts, namely automated segmentation methods.

The first design choice of such methods is for what domain the method should be used, as the real world can be digitally represented in different ways. A photograph of real world objects has different properties than medical images, originating from the technique used and the properties of objects that are recorded. Photographs can have multiple color channels, while medical recordings only record intensity values. In terms of objects the medical domain is restricted to the human body while a photograph is not. Each recording represents the real world in a mapping to pixel-wise intensity values, based on the resolution of the used technique. In medical recording techniques like Magnetic Resonance (MR) imaging, Optical Coherence Tomography (OCT) and Computed Tomography (CT) the resolution of the recordings increases

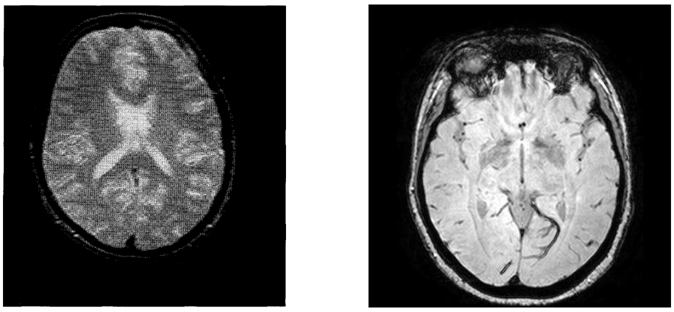

(a) (b)

Figure 1.3: Comparison of the quality of medical image recordings, where (a) shows a Head MR of a 1.5 Tesla machine (image taken from [52]) of 1997, (b) a Head MR of a 3 Tesla machine from the VISCERAL [27] dataset taken in 2013.

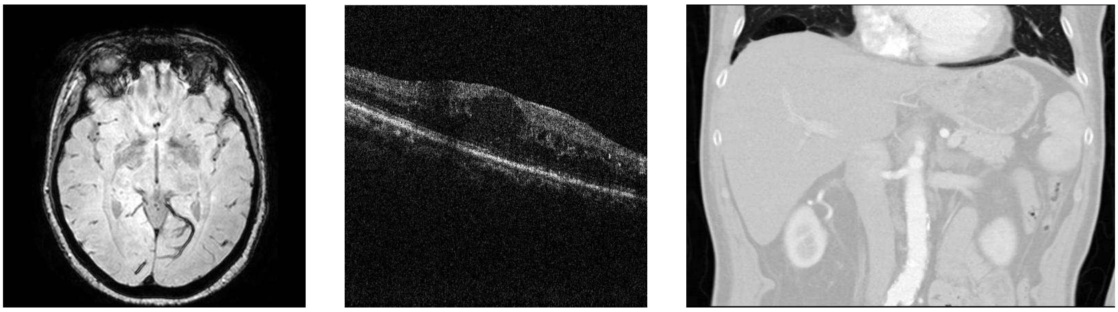

(a) (b) (c)

Figure 1.4: Example images for 3D medical volumes, where (a) shows a Head MR, (b) an OCT of the human eye and (c) an Abdomen CT from the VISCERAL dataset [27]

with the advances in technology [14]. This also increases the total number of pixels and voxels in a recording as shown in Figure 1.3 for a head MR taken in 1997 compared to one taken in 2013, where the latter is more detailed due to the higher resolution. In the following 3D recordings will be addressed as volumes and each pixel in such a volume is called voxel.

Examples for the medical imaging volumes can be seen in Figure 1.4, where each recording has its own properties. The MR (a) is used for visualizing different soft tissues, while the CT (c) detects how much energy is absorbed by objects such as bones [38]. OCT [25] (b) is used to examine the structure of an object with up to 3 mm depth creating a noisy representation of 3D structures of anatomical regions such as the retina [8]. These volumes create additional challenges when compared to images. Most obviously their size is larger by up to a factor of $1,000$, as for example the BSD images consist of 0.15 Mega Pixels (MP) while a full body CT can have a size of about 250 MP. Also the third dimension makes viewing of the recording a difficult task, as each layer blocks the sight on the subsequent one [13]. This property also causes problems when trying to annotate a volume, as volumes can either be viewed slice by slice or in complex 3D visualizations. Therefore manual annotation of structures in these volumes is more time intensive compared to images. For example, for a single volume with a size of $512 \times 512 \times 1000$, 1000 2D annotations have to be made for the segmentation. In order to create annotations for volumes more efficiently than doing it slice by slice, tools exist [26] that transport information across slices.

In order to segment an image or volume, for each of the pixels or voxels a decision must be made to which object it belongs. Making this decision for every single pixel is computationally expensive and therefore methods exist that reduce the number of pixels and the number of decisions necessary, by merging pixels into homogeneous groups of pixels while preserving object boundaries. These methods are called over-segmentation methods and are the main topic of this thesis.

The concept of over-segmentation reduces the number of pixels and enables the use of methods that would not be able to calculate in tractable time with current hardware. For example this reduces the effort of fitting a model of complexity $O(N^8)$, to an image of $1,000$ combined pixels instead of the original $500,000$ pixels [37]. Another usage is over-segmentation-based interest points in combination with bags of visual words for either medical image retrieval [23] or class-specified segmentation [30]. Over-segmentation is also used to combine the output of several different over-segmentation algorithms and merge them into one final segmentation of an image [29]. Additional approaches that use over-segmentation methods are [4] [21] [24].

1.2 Problem Statement

In order to detect objects in an image meta information or a priori learning is necessary. Algorithms exist that can detect certain simple objects [50] (e.g. face detection), but these are specifically trained and designed to the data. The problem addressed in this thesis is one step before the final object detection or classification. An image is over-segmented into homogeneous groups of pixels while preserving object boundaries, allowing other methods to detect any desired object in the image while only analyzing a fraction of the original number of pixels. Therefore it is important for the over-segmentation method to retain all object boundaries. Otherwise the subsequent algorithms cannot correctly delineate the object in the later stages.

Current state of the art over-segmentation methods are designed for 2D real world images. Only a few are used on medical images. The method of Andres 2008 [4] uses a modification of one of the first over-segmentation approaches [35] on volumes in combination with an hierarchical classifier, but there are no numeric results on the performance of over-segmentation methods on volumes. There are two reasons for this, (1) there is no over-segmentation code available for volumes with the exception of the code of Achanta 2010 [1] that a new method can be compared to and (2) until the starting of this thesis there is no public segmentation database available for volumes that is comparable to the BSD.

Due to their application on real world images, their performance on images with low contrast and brightness is not yet tested. These can occur in medical imaging data like the MR, which is used to investigate tissue and organs of the human body. Its signal is based on the single proton of the hydrogen nucleus [52]. The brightness and contrast of the recording depends on a combination of parameters (T1, T2, etc) used in order to distinguish specific soft tissues. If an over-segmentation has a parameter that handles the relation of brightness and contrast, it would have to be set specifically for each recording type. Creating a method that does not require a parameter to be tuned towards contrast and brightness would therefore increase its usability for medical imaging data.

The problems covered in this thesis can be summarized in the following points:

1. Over-segmentation methods are not yet compared on 2D medical images. Performance of state of the art over-segmentation methods are compared using the real world images of the BSD dataset. In this thesis 2D medical images and annotations are extracted from the VISCERAL [27] dataset and used for evaluation.

2. Over-segmentation methods are not yet compared on 3D medical imaging data. This thesis for the first time makes use of the 3D medical dataset named VISCERAL. This dataset contains expert anatomical annotations for more than 20 anatomical structures on 28 CT and MR volumes that can be used for volume segmentation evaluation.

3. There is no over-segmentation method available that has all the following properties:
    - Designed for low contrast/brightness medical imaging data like MR and CT.
    - Does not need parameter tweaking.
    - Over-segmentation of 3D medical imaging data within minutes.

1.3 Aim and Methodical Approach

The approach in this thesis is threefold. First state of the art methods for solving the over-segmentation problem are examined. The chosen over-segmentation algorithms cover the diverse directions of current state of the art methods. The features used for segmentation vary between gradient based, texture based or any combination of those features and the segmentation methods are based on graph partitioning or gradient ascent.

Secondly, after analyzing these methods a new method, *MonoSLIC* is developed and presented. It fills a gap that is discovered during the state of the art research by a new combination of feature extraction and segmentation and aims at the over-segmentation of medical image and volume recordings.

And thirdly, *MonoSLIC* and the state of the art methods examined [1] [15] [20] [37] [48] are compared on the 2D BSD dataset and, for the first time, also compared on the 2D medical images and 3D medical volumes of the VISCERAL [27] dataset.

1.4 Structure of Work

The structure of the remaining thesis is as follows. Chapter 2 presents a short history of over-segmentation methods, analyses the state of the art methods and discusses them. In Chapter 3 the new method *MonoSLIC* is presented, summarized and the relation to the state of the art methods is discussed All methods are then evaluated in Chapter 4 on image and volume segmentation datasets. In Chapter 5 the results of the *MonoSLIC* algorithm are discussed and compared to the other methods. Chapter 6 concludes the thesis with a brief summary and outlook on possible improvements and future work.

CHAPTER 2

State of the Art

In this chapter first an overview is given by discussing the properties of object boundaries in images and the history of image processing attempts to detect them. Then the definitions used to describe the state of the art methods are presented before analyzing them in detail. The chapter concludes with a comparison and summary.

2.1 Overview

The basic idea of over-segmentation is to merge pixels into homogeneous groups, while preserving object boundaries. In order to make a decision where such an object boundary is, we first look at how objects are defined in an image, and what properties define an object boundary.

Looking at the image in Figure 2.1 we follow the task of persons annotating the BSD dataset [34] and immediately recognize objects in the image like a cannon, trees, a wall, the ground and the sky. Analyzing the area where the sky and the wall meet we can observe a change in pixel intensity from bright to dark. This change of brightness between pixels is called gradient and is a first hint for an object boundary. Further investigating the cannon in more detail we notice that it is made of different parts, the wheels, the wooden basis and the tubular cannon body itself. Each of the parts is made of different material and based on their form and the light condition each creates a different structure in the image. We again take a closer look, but this time at the image boundary between the cannon and the wall. There is no obvious change in brightness between the pixels of the objects. In fact both structures have similar intensities. The only difference between them is how they are ordered, which as Wertheimer 1923 [51] stated is intuitively captured by the human mind. Therefore we can conclude that a change of structure is another hint for an object boundary, although it is more complex because more than two neighboring pixels need to be analyzed, compared to a gradient.

In image processing, a significant local change or discontinuity in brightness over a few pixels is called an edge. Although it is a simple way of defining an object boundary, the decision if the gradient is significant enough to be an edge is intensively studied in image process-

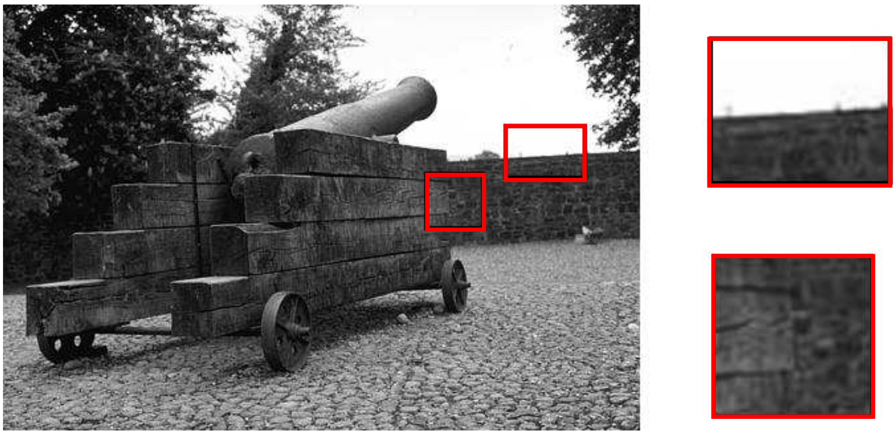


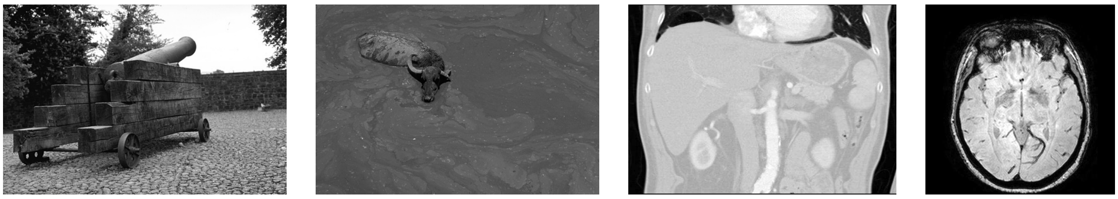
Figure 2.1: Cannon example from the BSD [34] dataset with two interesting object boundaries.

ing with methods called edge-detectors. An overview on edge-detection techniques is given in Ziou et al. 1998 [56] and explained in more detail in Section 2.4.

Taking into account texture information comes at a higher computational cost, because more than the direct neighbors of each pixel need to be analyzed in order to decide to which structure it belongs. Texture can be analyzed by convolution of an image with a filter bank. This is already a well-studied field and a comparative study on texture filtering can be found in Randen et al. 1999 [41]. More information about filter banks can be found in Section 2.3.

Another way of detecting edges in an image is not to look at object boundaries, but to combine pixels with the same properties. An early attempt to solve the problem of merging similar pixels is the use of the watershed algorithm for contour detection by Beucher et.al. 1979 [7]. Up to this time a traditional way of segmenting an image was to threshold the gradient of the image, which results in either thin but not closed contours, or closed but too thick. The watershed method provides closed contours for structures that differ by a high gradient using mathematical morphology. This is achieved by using the brightness values of pixels as topographic surface and starting to flood it from its minima. Whenever two minima meet, the merging of the waters is denied by a dam. These dams are then the final over-segmentation of the image. The major drawback of the watershed segmentation is severe over-segmentation, which is addressed by Beucher 1991 [6] introducing the marker driven watershed. Instead of starting flooding at minimas the flooding sources are defined by markers, restricting the number of sources and therefore decreasing the over-segmentation. Another change to the watershed algorithm is introduced by Meyer 1994 [35] using the geodesic distance function instead of the topographical to create a voronoi diagram where shortest path algorithms can create an output that is similar to the watershed. This watershed approach is still used in state of the art algorithms, for example in the method proposed by Engel et al. 2009 [15].

A synonym for over-segmentation is the term *superpixel* introduced by Malik et al. 2003 [42], which describes the grouping of pixels into homogeneous regions. The motivation for grouping pixels is that each pixel represents the real world in a discrete way, based on the resolution the recording technique is capable of. One pixel has semantically no meaning, only the combina-



(a) (b) (c) (d)

Figure 2.2: Example images used throughout this thesis. Cannon (a) and Ox (b) are taken from the BSD [34] dataset. The Abdomen CT (c) and the Head MR are taken from the VISCERAL dataset [27].

tion of pixels is the source of information in terms of segmentation and scene understanding. Although the output of previously mentioned edge-detection algorithms is closely related to the one of superpixel algorithms, they differ by the lack of restriction to closed contours as superpixels are constrained. Also superpixels are designed to have maximal homogeneity within each superpixel and maximum diversity between them.

The output of over-segmentation algorithms can be compared by the accuracy in terms of segmentation and by the regularity of the output. The latter means that if the size of one superpixel is not constrained this can result in the extreme case of only one superpixel as segmentation or on the contrary in as many superpixels as pixels. Also not restricting the shape of the superpixels could cause long and narrow superpixels or boundaries that are rough, which influences the performance of methods that are based on the superpixel boundary [45] and also is an indicator of over-fitting to the data (e.g. over-fitting to the noise in an OCT).

2.1.1 Definitions

In the remaining of this thesis the following definitions are used.

- For describing the upper bound of the complexity of algorithms the **Big O Notation** [12] is used with the symbol O.

- A graph is defined as followed. An undirected, connected **graph** $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is defined with **vertices** $\mathbf{V}$ and **edges** $\mathbf{E}$, where each vertex represents a pixel, and a weighted edge $\mathbf{e_{i,j}} = \mathbf{E}(\mathbf{v_i}, \mathbf{v_j})$ represents the connection between two corresponding pixels.

2.1.2 Images

The images shown in Figure 2.2 are taken from different sources and are used throughout this thesis. In panel (a) and (b) the Cannon and Ox images are shown and taken from the BSD [34] dataset. The abdomen CT (c) and the head MR (d) images are extracted from the VISCERAL dataset [27].

State of the Art						
Algorithm	Category	Parameter	Preprocessing	Information Representation		Segmentation
Veksler 2010 [48]	graph partition	superpixel count	-	Intensity	Energy Function represented as extended Grid Graph	Energy Minimization as Multi-Way-Graph Cut optimized by α-Expansion
Felzenswalb 2004 [20]	graph partition	significance value	-	Intensity	Grid Graph represented as Boundary Decision Rule	Disjoint-Set Forest optimized by Inverse Ackermann Function
Mori 2005 [37]	graph partition	superpixel count	texture and contour feature extraction	Features	Energy Function represented as Full Graph	Energy minimization as solving a General Eigenvalue System
Engel 2009 [15]	gradient ascent	seed points threshold	edge detection	Edge Information	Gradient Vector Flow Field to compute a Flux Flow Image	Watershed with Seedpoints and Height Map from Flux Flow Field
Achanta 2010 [1]	gradient ascent	superpixel count and regularity	-	Color Information	CIELab Color Space combined with Spatial Information	optimized k-means
Proposed Approach						
MonoSLIC 2013	gradient ascent	superpixel count	Monogenic Signal	Local Monogenic Phase	Monogenic Phase combined with Spatial Information	optimized k-means

Table 2.1: Overview of the methods discussed in this thesis.

2.2 Discussion

State of the art over-segmentation approaches are divided into different parts as shown in Figure 2.1. Achanta 2011 [2] proposed to categorize them into *graph-partition-based* and *gradient-ascent-based* approaches. In the first category a graph is constructed where each node represents a pixel and the edge weights are a similarity measure between two corresponding pixels. The graph is then cut such that dissimilar pixels are separated. The optimal cut is created by minimizing an energy function and the result is a superpixel segmentation of similar pixels. The methods of the second category create a rough initial segmentation of the image that is iteratively optimized resulting in a superpixel segmentation.

The work flow is similar for each over-segmentation approach. They first start off with the original image, followed by optional preprocessing of the image where features are extracted from the original image. The original image or the features or a combination of the two are used as input for each method. The extracted information is represented in different ways, either as a graph [48] [20] [37], as a higher dimensional features space [1] or as baseline to extract additional information [15]. In the final step the information is used to create the over-segmentation of the image by cutting the graph, clustering in the feature space or by using other methods like the watershed [5].

Each approach is reviewed using the same structure, starting with the image and ending with the final segmentation. First a short summary of the algorithm and methods it uses is presented together with an overview flow chart. Next we follow the work flow of over-segmentation methods guided by two example images, by first explaining the detailed **preprocessing**, followed by the **information representation** and **segmentation**. The last point details the **properties** of the algorithm containing information on parameters and runtime.

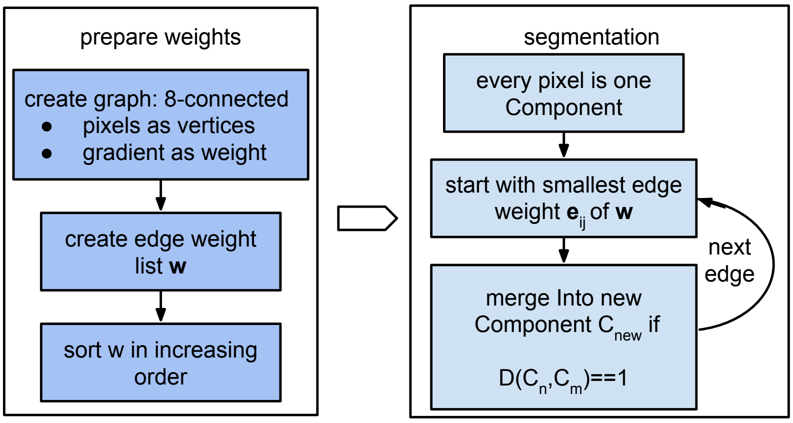
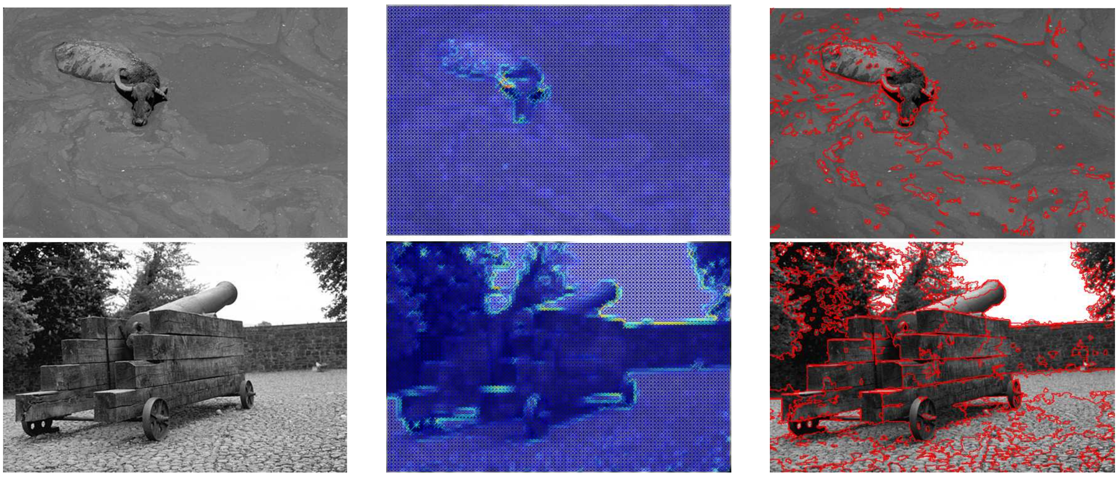

prepare weights
create graph: 8-connected
pixels as vertices
gradient as weight
create edge weight list **w**
sort w in increasing order
segmentation
every pixel is one Component
start with smallest edge weight $\mathbf{e}_{ij}$ of **w**
next edge
merge Into new Component C_{new} if
$D(C_n,C_m)==1$

Figure 2.3: Overview of the Efficient Graph-Based Image Segmentation method of Felzenswalb.



(a) (b) (c)

Figure 2.4: Method overview for Felzenswalb on the Ox and Cannon images. (a) Original input image, no features extracted. (b) Information representation illustration using the 8 connected neighborhood edge weights on a factor 0.2 downscaled version of the original image. (c) Segmentation on the example images.

2.3 Graph-Partition-Based Methods

The methods in this section use an information representation based on graphs.

2.3.1 Felzenswalb - Efficient Graph-Based Image Segmentation

The method of Felzenswalb 2004 [20] focuses on two important properties: (1) It captures perceptually meaningful regions and (2) is highly efficient. As summarized in Figure 2.3 a previously defined graph $\mathbf{G}$ is created and the edge weights are sorted in increasing order. Each pixel is assigned its own component C and starting with the smallest edge weight connecting two components C_n, C_m, the components are iteratively merged such that the internal weight $\mathrm{Int}(C_n)$ and $\mathrm{Int}(C_m)$ is low and the weight difference between two different component $\mathrm{Diff}(C_n, C_m)$ is high. The result is a superpixel segmentation of $\mathbf{V}$ into components $C \subset \mathbf{V}$. Figure 2.4 is used as guide following work flow of the method.

Preprocessing - Felzenswalb uses a Gaussian filter with $\sigma = 0.8$ for smoothing the original image Figure 2.4 (a). This is the only information used by this approach as no other features are extracted.

Information Representation - A grid graph $\mathbf{G}$ is created using a 8-connected neighborhood. The edge weights $\mathbf{e}_{ij}$ contain the absolute intensity difference between the two pixels i and j as shown in Figure 2.4 (b). In order to visualize the edge weights more clearly Figure 2.5 shows two different Gaussian distributions. Panel (a) shows the segmentation of two clearly distinguishable normalized distributions ($\mathcal{N}(mean = 0, std = 1)$ and $\mathcal{N}(4, 1)$), while in Panel (b) the distributions $\mathcal{N}(0, 1)$ and $\mathcal{N}(1, 1)$ are not clearly separable. Each pixel is connected to its neighbors according to the 8-connection criterion, where a low gradient is indicated in dark blue getting brighter with higher gradients until reaching yellow. The weights are increasingly ordered and to each pixel its own component C is assigned.

Segmentation - The segmentation process is started by taking the weight with the smallest value and the two components C_n, C_m it is connected to. The decision if these components should be merged is based on the relation of their internal weight to the weight between them. The internal weight of one component is measured as the maximum of all the edge weights within itself

$$\mathrm{Int}(C) = \max_{(v_i, v_j) \in C} \mathbf{e}_{ij}. \tag{2.1}$$

For a component size of $|C| = 1$ the internal weight is defined as $\mathrm{Int}(C) = 0$. A threshold function $\tau(C) = k/|C|$ based on the component size is introduced, so that the user has a general influence on the merging decision with the parameter k. For two components C_n, C_m a combined minima internal difference is defined as the minima of the internal weights added by the threshold function:

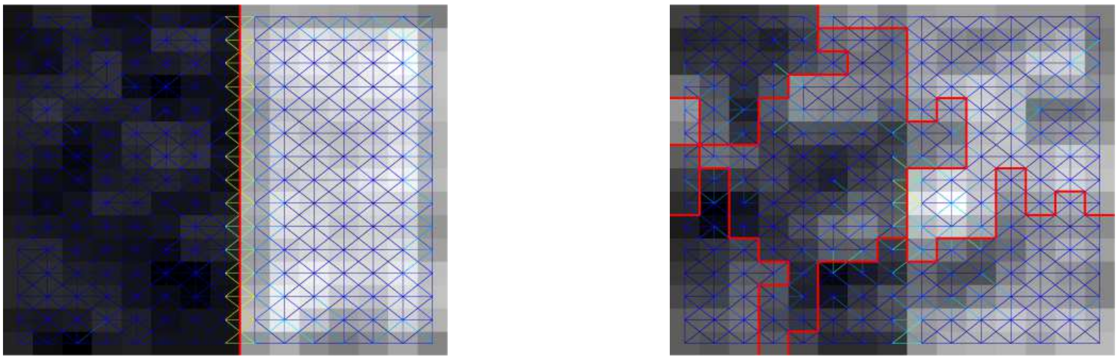
$$\mathrm{MInt}(C_n, C_m) = \min\left(\mathrm{Int}(C_n) + \tau(C_n), \mathrm{Int}(C_m) + \tau(C_m)\right). \tag{2.2}$$

In order to decide how different the two components $C_n, C_m \subseteq \mathbf{V}$ are, the difference is defined as the minima edge weight between them

$$\mathrm{Diff}(C_n, C_m) = \min_{v_i \in C_n, v_j \in C_m} \mathbf{e}_{ij}. \tag{2.3}$$

The final decision to create a boundary between the two components is based on whether the difference between them is larger than the minima internal difference.

$$D(C_n, C_m) = \begin{cases} 1 & \text{if } \mathrm{Diff}(C_n, C_m) > \mathrm{MInt}(C_n, C_m), \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$



(a) $\mathcal{N}(0,1)$ on the left and $\mathcal{N}(4,1)$ on the right.

(b) $\mathcal{N}(0,1)$ on the left and $\mathcal{N}(1,1)$ on the right.

Figure 2.5: Segmentation of two Gaussian-smoothed normal distributions. Color of the edges indicates the absolute intensity difference between corresponding pixels. Segmentation calculated with (a) parameter $k = 500$ and (b) parameter $k = 100$.

The segmentations of the example images are shown in Figure 2.4 (c) in red, calculated on the original size of the two images. To make the segmentation decision more we again look at the two normalized distributions of Figure 2.5. Panel (a) shows the segmentation in red of two clearly distinguishable normalized distributions using a parameter value of $k = 500$, while the distributions in Panel (b) are segmented using $k = 100$. Note that for (b) a parameter value of $k = 500$ would not create any segmentation, because of the insignificant edge weights compared to (a).

Properties - The runtime parameter k of the threshold function is used to control component size, where larger k corresponds to larger components, although smaller components will still be created when there is significant difference between neighboring ones. The segmentation output is irregular such that homogeneous regions like the sky and water have large superpixels and varying regions like the trees are segmented by small superpixels. The time complexity of the segmentation method is of $O(n \log n)$ [20]. Color images are treated as single monochrome image for each RGB-Channel and the algorithm is run three times. The final segmentation is an intersection of segmentations created for each color plane.

2.3.2 Mori - Normalized Cuts

The method of Mori 2005 [37], as summarized in Figure 2.6, first extracts a variety of perceptual information from the image, which is then grouped into contour and texture features. From these features an edge probability map is calculated, representing the probability for an object boundary at every pixel. In the second step the extracted features as well as the probability boundary map are used to calculate a texture and a contour weight matrix. These are combined into a final weight matrix W that reflects the similarity between pixels. The idea of features extracted and the design of the weight matrices is based on the paper of Malik 2001 [33].

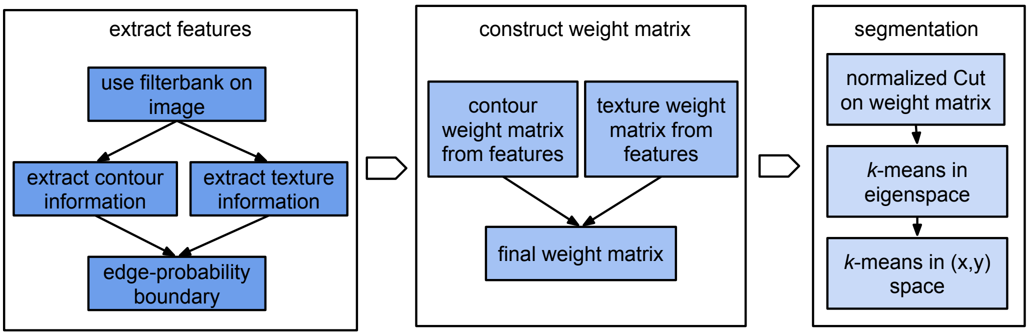

extract features
use filterbank on image
extract contour information
extract texture information
edge-probability boundary
construct weight matrix
contour weight matrix from features
texture weight matrix from features
final weight matrix
segmentation
normalized Cut on weight matrix
k-means in eigenspace
k-means in (x,y) space

Figure 2.6: Overview of the Normalized Cuts over-segmentation method of Mori.

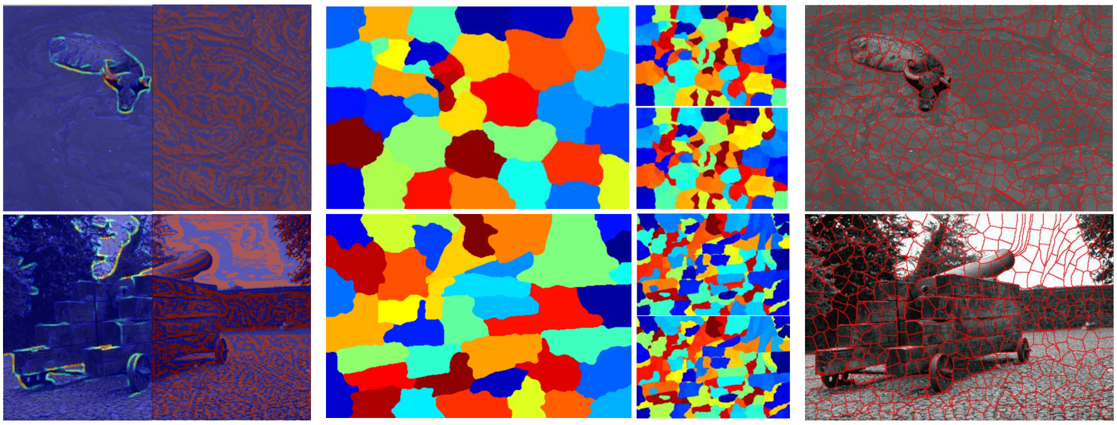

(a) (b) (c)

Figure 2.7: Method overview for Mori on the Ox and Cannon images. (a) Original input image overlaid with the extracted edge and texture features. (b) Segmentation procedure including *nCut* and subsequent k-means clustering. (c) Final Segmentation on the example images.

In a third an final step the weight matrix is segmented based on the *nCut* method proposed by Shi and Malik 2000 [32]. The result is a coarse pre-segmentation of the image, which is further subdivided using k-means clustering.

In the rest of this paragraph the single steps are explained in more detail, starting with the preprocessing and the information representation, followed by the segmentation section containing graph-cut and normalized-graph-cut methods. Each step is also visualized in Figure 2.7.

Preprocessing - Three color maps are created, one for each of the CIELab channels by reducing the number of unique values to a total of 32 for each. The reason for choosing this color-space is that for small color distances it is considered as perceptually uniform [1].

The texture map is calculated by filtering the image with a filter-bank consisting of $12x2$ (the first column are $13x13$ and the second $19x19$) Gaussian filters creating 24 feature responses for each pixel. These responses are compared with 64 pre-computed texture channels (each also consisting of 24 features) and the most similar one is assigned to each pixel creating a texture map with 64 labels.

The calculation from map to gradient is the same for both texture and color, and is done as follows. The texture map is used for texture gradient calculation while the color maps are used for the color gradients. A gradient is calculated by comparing two (upper and lower) map histograms for each pixel (orthogonal to the gradient orientation of the corresponding pixel) using the chi-squared-distance. If the map on both sides of the pixel is similar, then the pixel is within a texture or color and it's response is small. If the map is different, then the distance is high which this indicates a change in texture or a change in color.

The four (three color and one texture) gradients are then combined into an final edge probability map 2.7 (a) using the formula $pbi = \frac{1}{1+exp^{-x \times \beta}}$ where $x = [l, a, b, t]$ represents the four gradients and β weights each gradient with an empirical chosen value.

Information Representation - Intervening Contours [33] is used to construct a final weight matrix. In this method the weight between two pixels p_i and p_j is defined as the maximum pbi value on a direct line between them. Therefore a large weight in W_{IC} indicates that there is a boundary separating the two points. For segmentation W_{IC} is transformed to $W = exp^{-W_{IC}}$ where the value in W represents the costs to separate the two pixels. The weight matrix W is transformed to a previously defined graph $\mathbf{G}$.

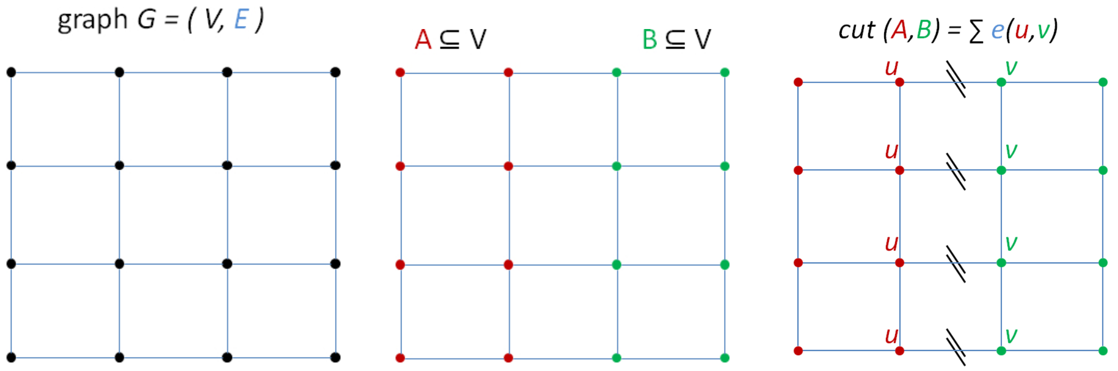
Segmentation - The goal is to create a segmentation of the image by partitioning $\mathbf{G}$ into disjoint sets.

Graph Cut - In the examples of Figure 2.8 (b) two disjoint sets A and B ($A \cup B = \mathbf{V}$, $A \cap B = \emptyset$) are created by removing the edges connecting the two sets. This is called the *cut* and its value is defined as the sum of the edge weights $\mathbf{e} \in \mathbf{E}$ removed from the graph. The definition of the graph-cut is visualized in Figure 2.8. Panel (a) shows the original graph, where each black dot represents a Vertex/Pixel and is connected in a 4-neighborhood to neighboring vertices' with an edge represented by a blue line. In (b) each vertex is assigned to a label A or B, creating two disjoint sets. The two sets are separated by removing the connecting edges as shown in (c), where the cut value is the sum of the edges removed.

$$cut(A, B) = \sum_{u \in A, v \in B} \mathbf{e}(u, v), \tag{2.5}$$

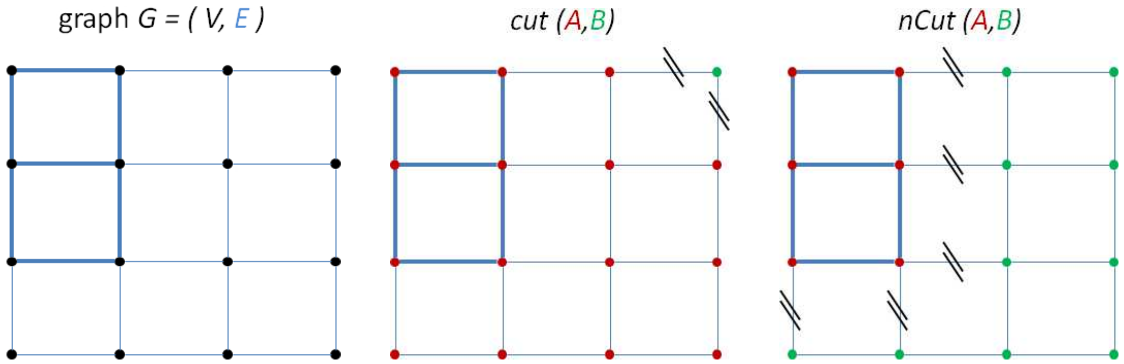
Normalized Graph Cut - The problem with the original *graph cut* is that the method favors cutting away a small number of vertices', because such a cut creates a small cut-value. Therefore *normalized graph cut* is introduced and to illustrate this a new graph is visualized in Figure 2.9 (a) where the weights correspond to the thickness of the edges, defining an object consisting of 8 vertices' in the top left corner. In (b) a minimum *cut* is shown that does not result in a desired segmentation result for the graph. To prevent removing small sets of points the cut value is normalized by the sum of edge weights $\mathbf{e}$ connecting the associated segment to $\mathbf{V}$ [32].

$$assoc(A, \mathbf{V}) = \sum_{u \in A, t \in V} \mathbf{e}(u, t) \tag{2.6}$$



(a) Original graph (b) Disjoint sets A and B (c) Calculation of the *cut* value

Figure 2.8: Illustration of the *Graph Cut* method.



(a) Original graph (b) Segmentation using *cut* (c) Segmentation using *nCut*

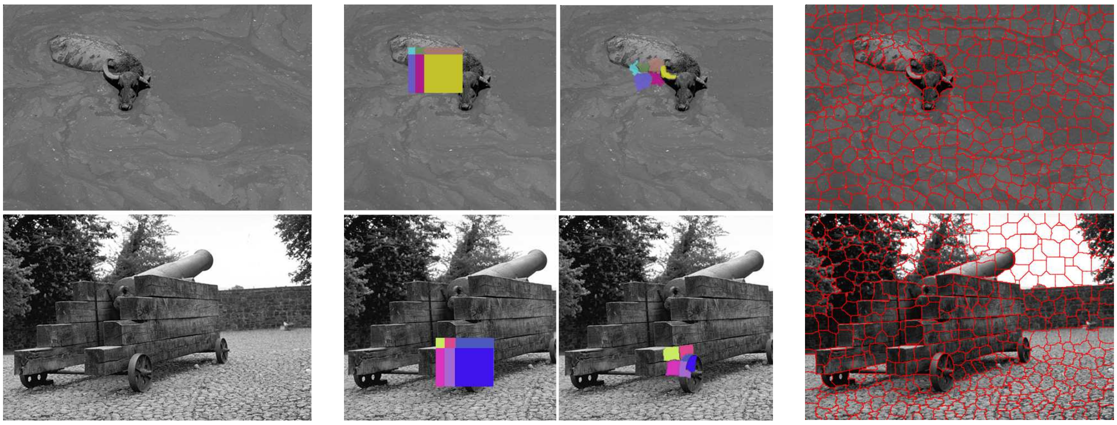
Figure 2.9: Difference of *cut* and *nCut* illustrated on an example graph.

With the final *normalized graph cut* definition

$$nCut(A, B) = \frac{cut(A, B)}{assoc(A, \mathbf{V})} + \frac{cut(A, B)}{assoc(B, \mathbf{V})} \tag{2.7}$$

the minimum *nCut* of the example graph in Figure 2.9 (c) reveals the object with the optimal segmentation.

Shi and Malik 2000 [32] also show that minimizing $nCut(A, B)$ is equal to solving the generalized eigenvalue system $(D - W)v = \lambda Dv$, where D is diagonal matrix with the sum of rows of the weight matrix W. Two methods on how the eigenvectors can be used for segmentation are described. (1) The first named *recursive 2-way cut* uses the eigenvector with the 2nd smallest eigenvalue to bipartition the graph. The splitting point for the eigenvector is chosen by calculating the minimum *nCut* value for l evenly spaced splitting points. The sub-partitions are segmented recursively until the the $nCut$ value exceeds a specified value. (2) The second, *simultaneous k-way cut*, uses the eigenvectors with the n smallest eigenvalues as n dimensional



(a) (b) (c)

Figure 2.10: Method overview for Veksler on the Ox and Cannon images. (a) Original input image, no features extracted. (b) Sketching of the initial patch distribution on the left and Patches after segmentation on the right. (c) Final Segmentation on the example image.

indicator vectors for each pixel. In this n dimensional space k-means is used to create an over segmentation of the image. These are then merged using either *greedy pruning* or *global recursive cut* to get the final segmentation.

Mori et al. 2005 [37] use an adapted version of the (2) *simultaneous k-way cut*. Its output for the two example images is visualized in Figure 2.7 (b). First n eigenvectors with the largest eigenvalues are used. The initial segmentation into n cluster is calculated by iteratively solving $X = VR$, where $R, RR' = I$ is a rotation matrix and V the n dimensional indicator matrix [55] and can be seen on the left. A second step uses k-means clustering on the *nCut* eigenvector coordinates to further subdivide the primary segmentation to Sp superpixels as shown on the top right. And in a third and final step k-means is used again for subdivision, although this time on the (x, y) pixel coordinates, creating the final segmentation of $Sp2$ superpixels (bottom right).

Label Cleanup - A final post processing step is necessary after the k-means algorithm, where disconnected clusters can be created. A new label is assigned to these disconnected regions. If one cluster is smaller than a defined minimum cluster size, it is merged with one of its neighbors.

Properties - The reason for the concatenating subdivision in the adapted normalized graph cut version of Mori is the high computational cost of *nCut* $O(N^{\frac{3}{2}})$ [28]. The default values for the parameters are suggested as $n = 40$ for the number of eigenvectors, $Sp = 100$ and $Sp2 = 200$ for the final segmentations. The results are regular superpixels in terms of shape and size.

2.3.3 Veksler - Superpixels in an Energy Optimization Framework

In Veksler 2010 [48] the segmentation task is formulated as an energy-minimization problem as summarized in Figure 2.11. The basic idea is that a number of overlapping patches L are equally distributed across the image, where this regular distribution also ensures the superpixel

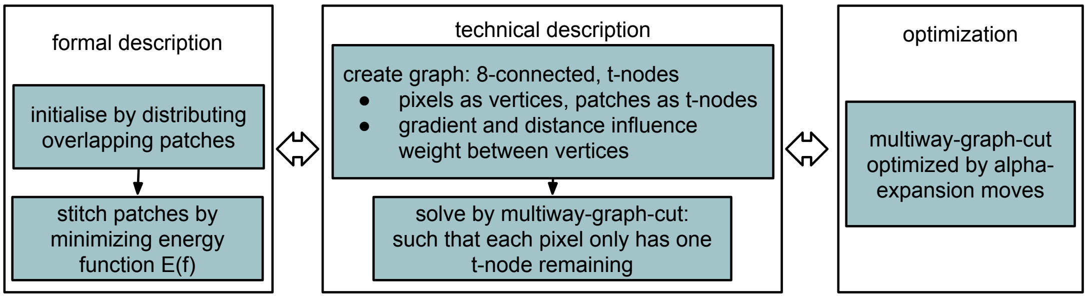

formal description
initialise by distributing overlapping patches
stitch patches by minimizing energy function E(f)
technical description
create graph: 8-connected, t-nodes
pixels as vertices, patches as t-nodes
gradient and distance influence weight between vertices
solve by multiway-graph-cut: such that each pixel only has one t-node remaining
optimization
multiway-graph-cut optimized by alpha-expansion moves

Figure 2.11: Overview of the Superpixels in an Energy Optimization Framework method of Veksler.

size. These patches are then stitched such that no overlapping patches remain and that the boundary between them aligns with a local brightness gradient in the image.

Preprocessing - The original image is the only information used by this approach as no other features are extracted.

Information Representation - From a programming point of view a graph $\mathbf{G}$ is constructed where each pixel is connected to its 8 neighbors. We now extend the previous definition of vertices $\mathbf{V}$ such that it also includes a terminal-nodes representing the overlaid patches. Each pixel is also connected to the terminal-nodes of the patches that the pixel is covered with. The edge weight $\mathbf{e}_{ij}$ between the pixels is influenced by the gradient and the spatial distance between them.

Energy Function - The energy function mentioned before consists of a data and a smoothing term

$$E(f) = \underbrace{\sum_{p \in P} D_i(f_i)}_{\text{data term}} + \lambda \underbrace{\sum_{\{p,q\} \in N} \mathbf{e}_{ij} \mathbf{V}_{ij}(f_i, f_j)}_{\text{smoothing term}} . \tag{2.8}$$

Each pixel $p \in P$ belongs to one or more corresponding patches $l \in L$. The assignment of pixel p_i to label l is denoted as $l = f_i$. The goal of the segmentation procedure now is to assign one pixel to only one label while so that the boundary between the patches creates a over-segmentation that preserves the object boundaries. This is also visualized in Figure 2.10 (b) where on the left the overlapping patches are regularly distributed over the image. On the right the patches are stitched using the described technique.

Segmentation - Veksler presents two versions of the algorithm, the first called *compact superpixels* is described in the next paragraph and the second, a modification of the first, named *constant intensity superpixels* is explained thereafter.

Compact Superpixels - The data term for the first version $D_i(f_i)$ is defined such that every pixel not belonging to a corresponding patch is penalized by setting its data term value to infinity

$$D_i(l) = \begin{cases} 1 & \text{if } p \in S(l), \\ \infty & \text{otherwise.} \end{cases} \tag{2.9}$$

The parameter λ is used to control the balance between data and smoothing term, but because the data term can only be 1 for the desired pixels, lambda can be chosen as $\lambda = 1$ without any effect on the optimization process [48]. The value of the smoothness term of two pixels $i, j \in P$ is nonzero only if the two pixels belong to different patches

$$\mathbf{V}_{ij}(f_i, f_j) = min(1, |f_i - f_j|^1). \tag{2.10}$$

For those pixels of different patches an edge weight difference $\mathbf{e}_{ij}$ is calculated influenced by the gradient I and the spatial distance $dist$

$$\mathbf{e}_{ij} = \exp\left(-\frac{(I_i - I_j)^2}{dist(i,j)2\sigma^2}\right). \tag{2.11}$$

This results in $\mathbf{e}_{ij} = 1$ for pixels with the same intensity measure and $0 \leq \mathbf{e}_{ij} < 1$ of dissimilar intensity measures, encouraging the stitching on highest gradients of the image. The *compact superpixels* boundaries adhere to high gradients, but there is no term in the energy function that forces smoothness throughout one superpixel. This allows a high gradient to appear within one superpixel, which is undesired in an over-segmentation approach. In order to improve this, the *constant intensity superpixels* approach is presented.

Constant Intensity Superpixels - There is one adaption to the *compact superpixels* method concerning the data term $D_i(l)$. Instead of assigning 1 to all pixels belonging to the patch $S(l)$, the data term value now depends on the gradient I between the pixel p and the patch center c, forcing constant brightness values throughout each patch

$$D_i(l) = \begin{cases} |I_i - I_{c(l)}| & \text{if } p \in S(l), \\ \infty & \text{otherwise.} \end{cases} \tag{2.12}$$

To ensure that label l of pixel p is the same as the cluster center $c(l)$, another term $T_{new}(f)$ is added to the energy function (2.8) penalizing different labels with ∞

$$T_{new}(f) = \sum_{p \in P} W(f_i, f_{c(f_i)}) \tag{2.13}$$

where

$$W(\alpha, \beta) = \begin{cases} \infty & \text{if } \alpha \neq \beta, \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

with $\alpha, \beta \in L$. The segmentations of the *constant intensity superpixels* are shown in Figure 2.10 (c) for the Cannon and Ox image of the BSD.

Properties - The benefit of the *constant intensity* superpixels is a more accurate segmentation at the cost of superpixel regularity [48]. What is not mentioned in the paper is that the change of the data term means that λ again has an influence on the smoothness of the superpixels, where larger λ creates smoother superpixels. The resulting superpixel are of regular shape and size. The runtime is close to linear $O(N)$ for a constant patch count, but is increasing with the number of patches [9] and therefore with the number of desired superpixel.

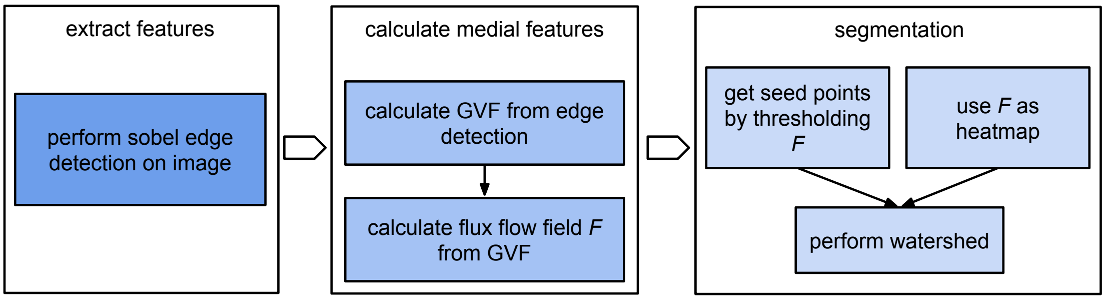

extract features
perform sobel edge detection on image
calculate medial features
calculate GVF from edge detection
calculate flux flow field F from GVF
segmentation
get seed points by thresholding F
use F as heatmap
perform watershed

Figure 2.12: Overview of the Medial Features for Superpixel Segmentation approach of Engel.

-1	0	1
a	0	a
-1	0	1

(a)

-1	a	-1
0	0	0
1	a	1

(b)

1	0
0	-1

(c)

0	1
-1	0

(d)

0	1	0
1	-4	1
0	1	0

(e)

Figure 2.13: (a) Horizontal and (b) vertical masks used by Prewitt and Sobel edge detectors. The masks used by Roberts (c), (d) and the mask for the Laplacian of Gaussians (e).

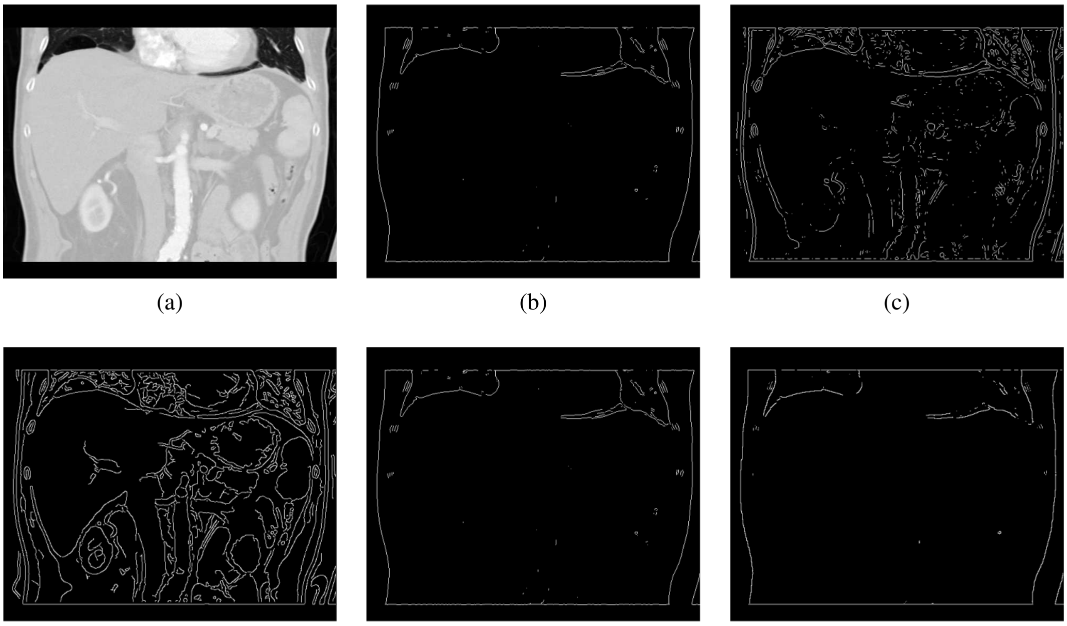
2.4 Gradient-Ascent-Based Methods

The methods in this section use gradient information based on the extracted features.

2.4.1 Engel - Medial Features for Superpixel Segmentation

The proposed method of Engel 2009 [15] makes use of two popular methods in image processing as summarized in Figure 2.12. One is extracting edge-cues using an edge-detection algorithm and the other is creating a segmentation using the watershed algorithm [35]. The heat-map and seed-points needed for the watershed are calculated from the extracted edges. First a *Gradient Vector Flow* (GVF) field based on the edge cues is calculated and in a subsequent second step a *flux flow* field ($\mathcal{F}$) is created from the GFV. $\mathcal{F}$ is used as the heat-map and it is also thresholded in order to create seed-points, which both are the input for the watershed segmentation. Each step is now explained in more detail with corresponding visualizations in Figure 2.15.

Preprocessing - *Edge-detection* methods can be grouped into first and second order derivative expressions [56]. First order derivative extract gradient information, while second order derivative extract the rate of change of the gradient. The derivatives are approximated by using masks that differ for each approach. The first order masks of Prewitt [40] and Sobel [31] are shown in Figure 2.13 (a,b) where $a = 1$ for Prewitt and $a = 2$ for Sobel. Due to the similarity of the masks also the segmentation output is similar as seen in Figure 2.14 (b) and (c). Another first-order approach by Roberts uses the *Robert's cross* Figure 2.13 (c,d), where the final gradient value for each pixel is the maximum of the two filter outputs [43]. The output is comparable to the one of the previous two methods Figure 2.14 (f). A second-order method is the Lapla-



(a) (b) (c)

(d) (e) (f)

Figure 2.14: Results of edge detection algorithms on the Abdomen CT slice (a), (b) Prewitt, (c) Laplacian of Gaussian, (d) Canny, (e) Sobel and (f) Roberts.

cian, estimated by the mask shown in Figure 2.13 (e), resulting in an edge response as seen in Figure 2.14 (c).

The method of Canny [10] can be split up into several steps, where the image is first convolved with a Gaussian filter to reduce noise. In the next step the intensity gradient is detected using four filters (horizontal, vertical and two diagonal). The filter responses are non-maxima suppressed before tracing the edges through the image using thresholding (with hysteresis) resulting in the detected edges of Figure 2.14(d).

Engel has chosen the Canny method for extracting the edge information as it performs best in terms of over-segmentation [15] and its result is the edge-image f that is also visualized in Figure 2.15 (a) for the two example images Ox and Cannon.

Information Representation - The edge cues extracted from the image are now further processed to create a information representation that will later be used for segmentation.

Gradient Vector Flow - First a vector field $V(p) = |u(p), v(p)|^T$, where $p = (x, y)$ is a pixel in the image, is defined. Each vector in V points to the direction of the largest brightness change. The length of the vector represents the magnitude of change in brightness. This vector field is used to minimize the adapted [53] energy function

$$\mathcal{E} = \iint \underbrace{|\nabla f|^2 |V - \nabla f|^2}_{\text{data term}} + 0.12 \underbrace{\nabla^2 V}_{\text{smoothing term}} \quad dxdy \tag{2.15}$$

which results in the GVF as shown in Figure 2.15 (b) on the top left. Similar to Equation 2.8 the

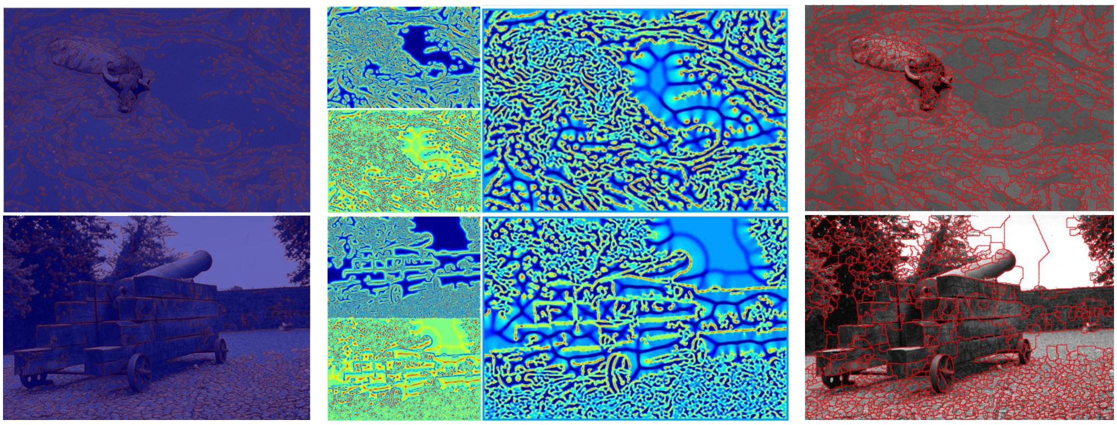

(a) (b) (c)

Figure 2.15: Method overview for Engel on the Ox and Cannon images. (a) Original input image overlaid with the Sobel edge-detection result as features. (b) Visualization of the GVF on the top left and the corresponding *flux flow field* below, The big image on the right is the visualization of the final height map that is used for segmentation. (c) Final Segmentation on the example image using the heat-map and the watershed segmentation.

energy function has a data and smoothing term, where the data term is influenced by the gradient of the edge image f and its difference to the vector field V. The smoothness constraint is given by the gradient of the vector field V and is weighted by a constant [53].

Flux Flow Field - The flux flow field $\mathcal{F}$ is calculated based on the $L_2 - norm$ normalized vector field $V_N(p) = V(p)/||V(p)||^2$ following the suggestions of [39].

$$\mathcal{F}(V_N(p)) = \frac{\oint \langle V_N, \mathcal{N} \rangle ds}{Area} \tag{2.16}$$

where $\mathcal{N}$ are the normals of a 7-pixel-ring used in the ring integral and is shown in Figure 2.15 (b) on the bottom left. The final segmentation is done using the watershed transform [35] with $\mathcal{F}$ as a heat map (Figure 2.15 (b) on the right) and a thresholded version of $\mathcal{F}$ as seed points.

Segmentation - The watershed algorithm is first described by Beucher 1979 et al. [7]. First the gradient of an image is calculated, which is then treated as a relief, where the height of the relief is equal to the value of the gradient. The relief is then flooded, starting from local minima. Whenever two different minima would merge, a dam is created. This results in a severe over segmentation of the image. Meyer 1994 et al. [35] adapted the first version, such that the flooding starts not at local minima, but at chosen marker positions. The final segmentations for the two test images are shown in Figure 2.15 (c).

Properties - The threshold for the seed points has to be chosen manually and is the only parameter for this method, therefore the number of superpixel cannot be chosen directly. The segmentation output is very irregular as seen in Figure 2.15 (c) where homogeneous regions like the sky and the water create large and varying regions, and the trees or the ox have small

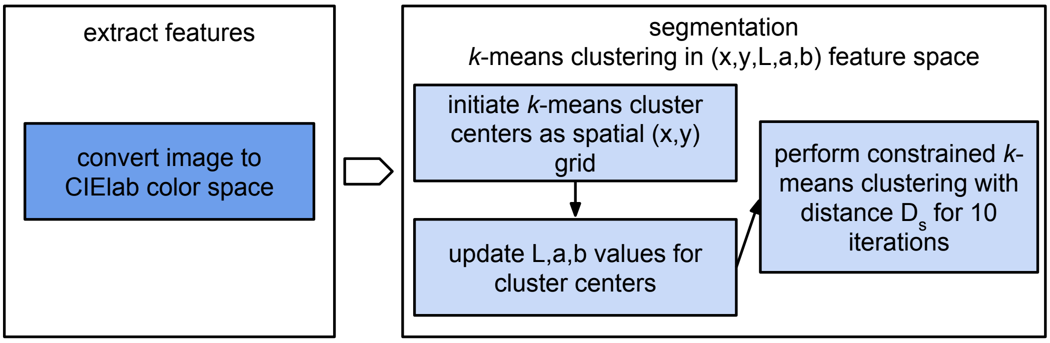

extract features
convert image to CIElab color space
segmentation
k-means clustering in (x,y,L,a,b) feature space
initiate k-means cluster centers as spatial (x,y) grid
update L,a,b values for cluster centers
perform constrained k-means clustering with distance D s for 10 iterations

Figure 2.16: Overview of the SLIC Superpixels approach of Achanta.

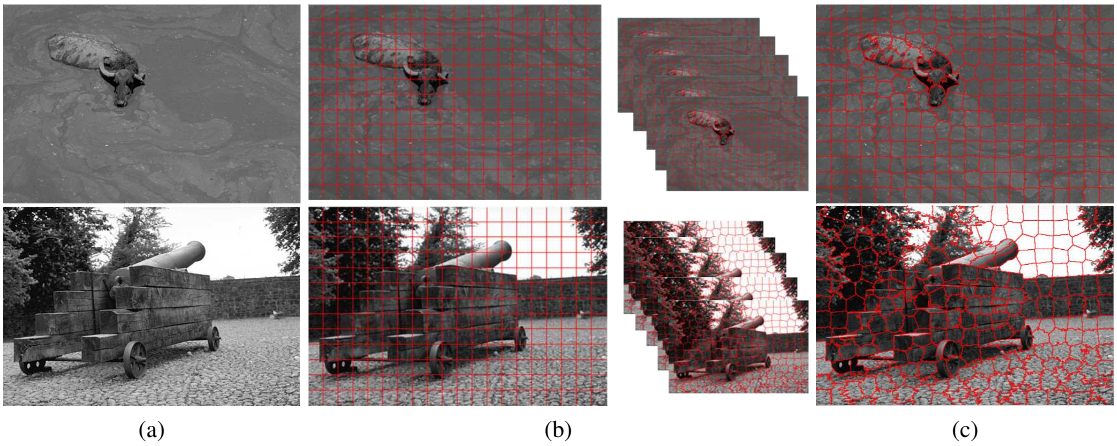

(a) (b) (c)

Figure 2.17: Method overview for Achanta on the Ox and Cannon images. (a) Original input image, no features extracted. (b) Sketching of the initial cluster distribution on the left and visualization of the k-means iterations on the right (c) Final segmentation on the example image.

and irregular superpixel. As each step of the computation consisting of edge-detection, gradient vector field, flux flow field and watershed takes O(N) [16], the combined complexity of the method is also O(N).

2.4.2 Achanta - Simple Linear Iterative Clustering

The *Simple Linear Iterative Clustering* (SLIC) approach of Achanta 2010 [2] uses local pixel information represented in the CIELab color space in combination with the euclidean distance between pixels as shown in Figure 2.16. Similar to [20] a 5D-space is defined, but this time using the *L,a,b* values of the CIELab color space together with the *x,y* pixel coordinates. k-means clustering is performed in the 5D-space, initialized by a spatial (x,y) grid and the corresponding cluster (L,a,b) values. Clustering is performed using a specified non-euclidean distance measure D_s, because compared to the spatial distance, the distance in the CIELab color space is limited [1]. Each cluster represents one superpixel and with a final post processing step, where disconnected or small cluster centers are merged, the finale over segmentation is created.

Preprocessing - The image is converted to the CIELab color space which limits the distance of two pixels when compared to the euclidean space. Other than that this is the only information used by this approach as no other features are extracted.

Information Representation - For 2D the information is represented in a 5D feature space spanned by the spatial coordinates (x,y) and the color space variables (L,a,b). The distance measure in this 5D space is divided into two parts. The color distance is represented as

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2\,(b_k - b_i)^2} \tag{2.17}$$

and the spatial euclidean distance as

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}\,. \tag{2.18}$$

Before the distances are added, the spatial distance d_{xy} is normalized by the value d_{cc} that is based on the total number of superpixels desired by the user.

$$D_s = d_{lab} + \frac{m}{d_{cc}} d_{xy} \tag{2.19}$$

With the parameter m the user can balance between color similarity and spatial proximity. The suggested default value for $m = 10$. The parameter used for the calculations used in this thesis is discussed in Section 4.3.

Segmentation As the segmentation method k-means is used for cluster analysis. Initially k cluster centers are placed in a regular spaced grid in the (x,y) plane, where k matches the desired number of superpixels as shown on the left in Figure 2.17 (b). The distance between the cluster centers is defined as $d_{cc} = \sqrt{\frac{N}{K}}$ where N is the number of pixels and K the number of desired superpixels. Before starting with the k-means algorithm each cluster center is first moved towards the lowest gradient in a 3×3 neighborhood. As part of the initialization process the (L,a,b) values of the cluster centers are set to the average of the (L,a,b) values of the pixels that are closest to the specific center. After that, repeatedly for each cluster center the best matching pixels from a $2S \times 2S$ square neighborhood around the center are calculated using the distance measure D_s. The cluster centers are being updated, until 10 iterations are reached as indicated in Figure 2.17 (b) on the right. As a final step *label cleanup* similar to the one explained in the method of [37] is performed. The final segmentation for the example images is visualized in Figure 2.17 (c).

Properties - The restriction of the k-means search space to a $2d_{cc} \times 2d_{cc}$ neighborhood instead of to the whole image and the maximum iterations to 10, reduces the original k-mean complexity from $O(NKI)$ to $O(N)$ where N is the number of pixel, K the number of cluster and I the number of iterations [2]. When the segmentation is done, cluster connectivity is enforced in a final step, taking only a fraction of the k-means runtime [1]. The superpixel size and regularity are based on the chosen parameter m, but due to the k-means search space restriction as well as the post processing, have a limited maximum and minimum size.

2.5 Other Superpixel Algorithms

The discussed algorithms are examples that illustrate different lines of solving the over-segmentation problem. However this selection is not complete. Additional approaches are proposed. They range from gradient-ascent based approaches of Vincent 1991 [49], Levenshtein 2009 [28] and the graph-partition-based approach of Moore et al. 2008 [36]. A comparison of those can be found in [2].

2.6 Comparison of the State of the Art Approaches

The algorithms are compared in terms of parameter, superpixel regularity and computational cost.

Parameters - In terms of parameters the methods are split into two groups, the first allows the user to directly choose the superpixel count for the segmentation, while the second is based on parameters that are method specific and cannot directly specify the number of superpixels.

Veksler and Achanta allow to directly set the number of initial centers for the consequent calculations. A post-processing step ensures that no superpixel smaller than a chosen size remain. These can occur as remnants of the iteration process [1]. In Achanta another parameter needs to be set which defines the regularity of the superpixel. Mori also chooses the number of superpixels directly, but compared to the previous methods achieves this by iteratively sub-partition large superpixel until the desired number of superpixels is reached.

Engel and Felzenswalb belong to the second parameter group. The parameter in Engel is a threshold that decides on the seed points for the watershed algorithm, while in Felzenswalb the user can set the significance necessary to divide a component into two.

Features - The information used for segmentation varies for each approach and is shown in Figure 2.18. Veksler and Felzenswalb use the unchanged intensity values of the image ((a) top). Achanta is based on the CIELab color information, but with gray-scale images this reverts to also using intensity values. The other algorithms use some sort of preprocessing before calculating the superpixel segmentation. Engel first extracts edge cues based on the Canny Edge detector ((a) bottom) and most complex information is extracted by Mori (b), where texture cues shown on the top and contour cues on the bottom are used.

Superpixel Regularity - With not being able to set the number of superpixels, and therefore the size of them, Engel and Felzenswalb produce irregular superpixel sizes and shapes Figure 2.19 (a) and (b). The additional parameter in Achanta decides to either capture more low gradient boundaries at the cost of irregularity, or capture less boundaries but having higher superpixel regularity (c). Veksler (d) and Mori (e) create very regular superpixel, with the latter having smoother boundaries.

Computational Cost - With a complexity of O($N^{3/2}$) Mori has the highest computational cost. Additionally it needs to store a matrix of size N^2, which is not feasible for large images, yet volumes. Felzenswalb has a complexity of O($N \times \log N$) elsewhere the methods of Achanta, Engel and Veksler have O(N). The complexity is also visualized in Figure 2.20 for increasing N.

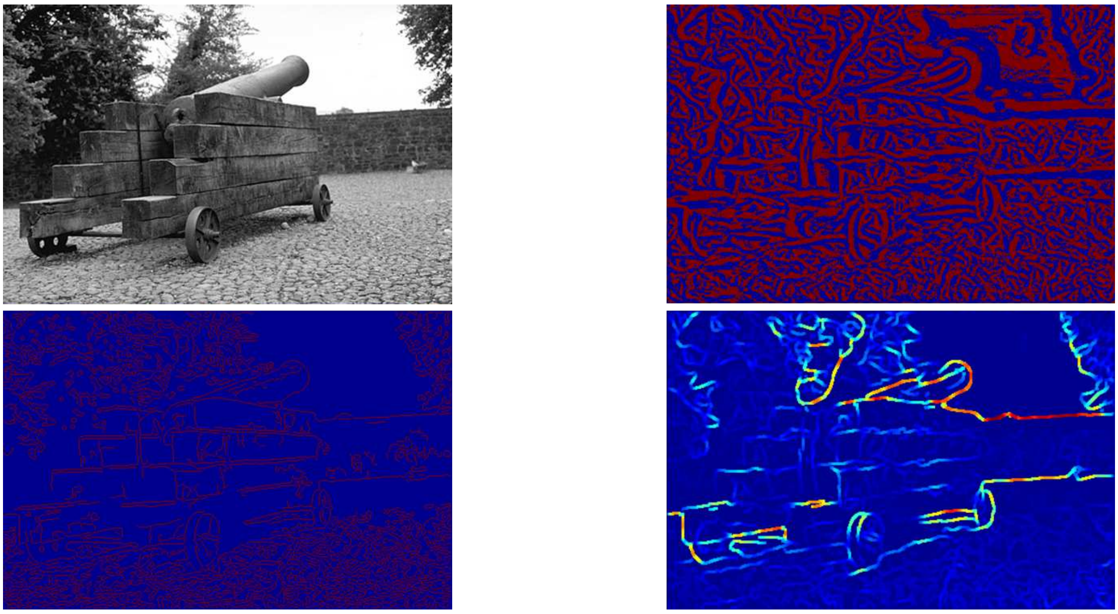

(a) (b)

Figure 2.18: Comparison of the extracted features of the algorithms on the Cannon image. (a) Original Cannon image on top. Canny edge features on the bottom. (b) Gradient (bottom) and texture (top) features extracted by Mori.

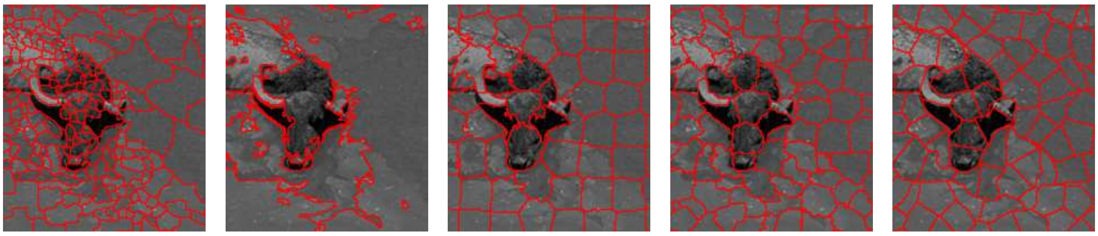

(a) (b) (c) (d) (e)

Figure 2.19: Comparison of the superpixel regularity and smoothness on a zoomed in version of the Ox image, where (a) Engel, (b) Felzenswalb, (c) Achanta, (d) Veksler and (e) Mori.

2.7 Summary

The method proposed by **Felzenswalb** [20] uses the original gray values of an image and creates superpixels by merging two components (e.g. pixels) if their difference is not significant enough to be an object boundary. There is a parameter that allows control over the significance needed, but it is not possible to create an over-segmentation with an exact number of superpixel. It therefore creates large superpixel at the homogeneous areas and small ones at heterogeneous.

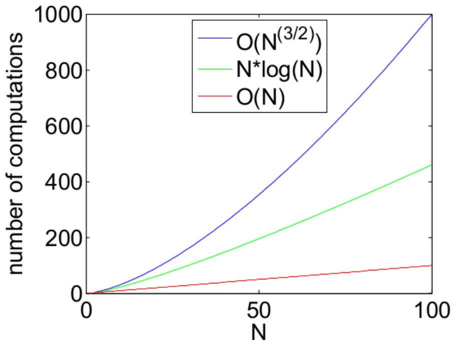


Figure 2.20: Visualization of the complexity.

Mori [37] extracts gradient and texture cues from the image and creates a large similarity matrix *W* of size O(N^2), which is then divided into superpixels by using the described graph-cut method of complexity O($N^{2/3}$). This is done until the desired number of superpixel is reached. It creates regular superpixels that highlight the dominant local texture and gradient difference.

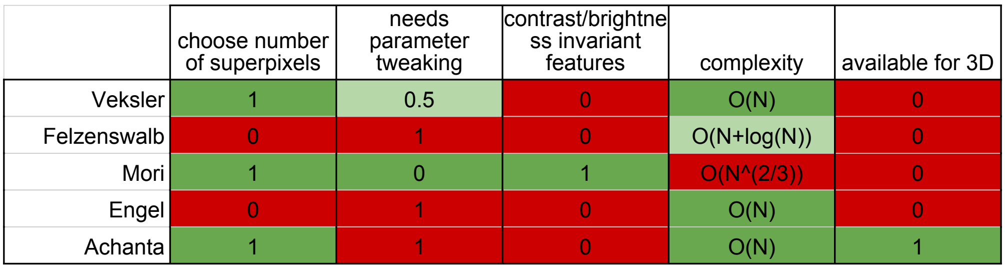
The approach of **Veksler** [48] uses the gradient information of the original image for segmentation. Overlaying patches are distributed across the image and are then stitched so that they align with the local dominant gradient. The number of patches matches the desired number of superpixels. In the used *constant intensity* approach has a parameter to tweak the desired smoothness of the superpixels. The superpixel are less regular than the one from Mori and highlight the local gradient difference.

The gradient of the original image detected by the Canny edge detector is the source information used by the approach of **Engel** [15]. It is used to generate a heat map and seed points for a subsequent watershed segmentation. The threshold for the seed-points is the only parameter for the method and therefore the exact number of superpixel cannot be chosen. Similar to the method of Felzenswalb it creates large superpixel at homogeneous areas and smaller at heterogeneous ones.

Achanta [1] converts the original image to the CIElab color space. Together with the spatial coordinates a higher dimensional feature space is created. A rectangular grid, where the number of rectangles matches the desired number of superpixels, is used as initialization for a k-means clustering algorithm using the previously described distance function. The relation between gradient and spatial information can be adapted using a parameter value. The resulting superpixels are similar to the one from Veksler and highlight the local dominant edge based on the parameter chosen.

2.8 Limitations of State of the Art

The limitations of the state of the art methods are explained with Table 2.2, where desired properties are marked in dark green and undesired ones in red. With the methods of Engel and Felzenswalb it is not possible to directly define the **number of superpixels** desired, as indicated in the first column. This number depends on brightness, contrast and homogeneity of the image



	choose number of superpixels	needs parameter tweaking	contrast/brightness invariant features	complexity	available for 3D
Veksler	1	0.5	0	O(N)	0
Felzenswalb	0	1	0	O(N+log(N))	0
Mori	1	0	1	O(N^(2/3))	0
Engel	0	1	0	O(N)	0
Achanta	1	1	0	O(N)	1

Table 2.2: Overview of the limitations of the state of the art over-segmentation methods.

and the parameter value chosen. In extreme cases this can create segmentation with only one superpixel, or segmentation with as many superpixels as pixels which would nullify the benefits of over-segmentation. This can be observed in Figure 2.4 (c) and Figure 2.15 (c) on the Cannon images, where the sky contains large superpixels or at the trees where very small superpixels are created. This makes it also difficult to compare the performance, which is based on comparing the over-segmentations at a certain number of superpixels.

Achanta and the *constant intensity* approach of Veksler create an over-segmentation with a defined number of superpixels, but similar to Felzenswalb and Engel they also have a **parameter** that needs to be tuned for the specific properties of an image (column two). Images that due to their homogeneity properties require a different parameter at different areas in the image cannot be properly over-segmented.

The method of Mori does allow choosing the number of superpixels and has no image specific parameter that needs tuning. It is also the only method that uses features that are **contrast and brightness invariant** (column three), but it uses the most **complex** segmentation approach (column four). This results in slow segmentation times, where an image of the BSD with 0.15 MegaPixels takes more than a minute to compute as shown in Section 4.6.

The method of Achanta is the only one where the available implementation supports to compute **3D volumes**, as indicated in the last column.

The overview in Figure 2.2 shows that each algorithm has limitations. The method presented in the next Chapter is designed to overcome these specific limitations of the state of the art in one single approach.

CHAPTER 3

Monogenic SLIC

As analyzed in Chapter 2 only one (Mori Section 2.3.2) of the chosen methods incorporate texture information as a segmentation cue and does not need an additional parameter to be tuned. The other methods solely make use of the brightness/gradient information and have a parameter that can be adapted to the contrast of the image.

The idea of the approach presented in this section originates from the method of Achanta [1] which, as described in Section 2.4.2, uses the color values of an image as features and combined with k-means clustering creates an over-segmentation of the image. The proposed method differs in preprocessing and the features used, which enables a contrast and brightness invariant method that also does not need an additional parameter and allows a more efficient k-means calculation.

The information used as feature is based on texture by using the phase of the monogenic signal [19]. This way structures are detected even though they only have a small gradient compared to their surroundings. From the use of the *mono*genic signal combined with the *S*imple *L*inear *I*terative *C*lustering the abbreviation *MonoSLIC* is used as a name of the presented method.

The new method is presented similar to the state of the art structure. First the **preprocessing** contains detailed information about the monogenic signal. This is followed by the main contributions of this thesis. First how it can be used as **information representation** for the purpose of image segmentation and second the **segmentation** describes how the clustering method k-means is adapted for creating the over-segmentation. Thereafter the **properties** of the method are explained. The chapter concludes with the comparison of the method to the state of the art and a summary.

In the following x is used as representation of a signal in the spatial domain and u as representation in the frequency domain. A bold variable indicates an n-dimensional signal $\mathbf{x} = (x_1, x_2, ...x_n)^\mathsf{T}$.

3.1 Preprocessing

In this section the preprocessing of the image is explained. The image is transformed from the original intensity based to an amplitude, phase and orientation based representation called monogenic signal.

Monogenic Signal - The *monogenic signal* proposed by Felsberg [19] originates from the 1D analytic signal [11], which extracts the local phase and local amplitude of a signal. The phase represents the local structure in the signal and the amplitude its magnitude. This set of features is independent to each other, meaning that a change in one does not affect the other and vice versa. This is called the *invariance-equivariance* property. Another property is that any signal can be uniquely represented by those two features. Having a set of features with both properties is also called *split of identity*. A popular example for this feature is the phase and amplitude of the Fourier Transformation and the analytic signal also full-fills this property. It is based on the Hilbert transform which, when applied to $cos(.)$ results in a $sin(.)$. The Hilbert transform can be understood as a *phase shifter*, shifting every sinusoidal function by -90 degrees. In the frequency domain, with the Fourier transformed signal u, the transfer function of the Hilbert transform can be defined as $H(u) = \frac{-iu}{|u|} = -i\, sgn(u)$, where $sgn()$ is the signum function [19]. The kernel can be written as the inverse Fourier transform of the transfer function [54]

$$f_H(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} H(u) e^{iux} du \tag{3.1}$$

and its combination with the original signal $f(x)$ is called the **analytic signal**

$$f_A(x) = f(x) - i f_H(x), \tag{3.2}$$

Both the Fourier and the Hilbert transform make use of complex numbers as the mathematical representation. The complex numbers are two dimensional, so the local amplitude and the local phase can be represented for a 1D signal analytic signal. When moving to a 2D signal (e.g. an image) then another feature is necessary that describes the local orientation of the structure in the 2D plane. One could calculate the Hilbert transform for a number of orientation and approximate the exact result, but this has shown to be inaccurate as well as time consuming [17].

Therefore Felsberg generalizes the Hilbert transform using the Riesz transform for n dimensional signals with the defined transfer function in the frequency domain as $H_2(\mathbf{u}) = \frac{\mathbf{u}}{|\mathbf{u}|} I_2^{-1}$ and the resulting transformed signal $F_R(\mathbf{u}) = \frac{i\mathbf{u}}{|\mathbf{u}|} F(\mathbf{u}) = H_2(\mathbf{u}) F(\mathbf{u})$. In the spatial domain this is written as the convolution $f_R(\mathbf{x}) = -\frac{\mathbf{x}}{2\pi|\mathbf{x}|^3} * f(\mathbf{x}) = h_2(\mathbf{x}) * f(\mathbf{x})$. Similar to the analytic signal the **monogenic signal** is also a combination of the original signal and, this time, its Riesz transformation

$$f_M(\mathbf{x}) = f(\mathbf{x}) - (i, j) f_R(\mathbf{x}). \tag{3.3}$$

Felsberg shows that the monogenic signal is isotropic and also performs a split of identity [19]. The orthogonal set of features the signal is decomposed into are the local amplitude A_f, the local phase φ and the local orientation θ.

The energetic information, based on brightness/contrast, is included by the local **amplitude** and is the norm of the monogenic signal.

$$A_f(\mathbf{x}) = |f_M(\mathbf{x})| = \sqrt{f^2(\mathbf{x}) + |f_R(\mathbf{x})|^2} \tag{3.4}$$

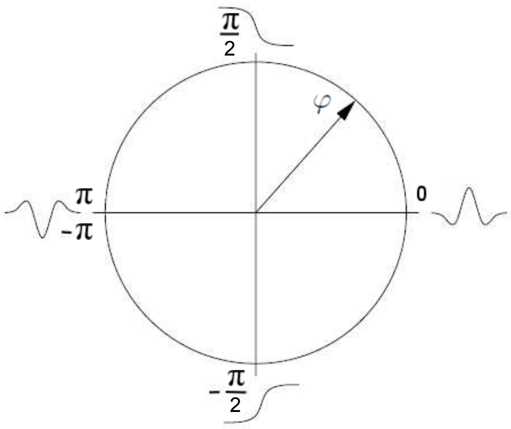


Figure 3.1: The sketched structure yielding a specific value of φ with a phase wrap from π to $-\pi$. Redrawn from [19].

The amplitude represents the local energy and is invariant to the local phase φ and local orientation θ.

The local **phase** represents the change of local structural information varying from $-\pi$ to π. The corresponding structure for a certain φ value is shown in Figure 3.1 for the 1D case. With the sketched structure one can see that the phase values of $|\frac{\pi}{2}|$ contains edge-like information.

The phase φ is defined as

$$\varphi\left(\mathbf{x}\right) = atan3(f_M(\mathbf{x})) = arg(f'_M(\mathbf{x}))\varphi \in [-\pi, \pi)\,, \tag{3.5}$$

where $\mathbf{f}_M$ is the vector field such that $f'_M = (i, j, 1)f_M$.

For completion the geometric information is represented by the **orientation** in the range of 0 to π

$$\theta\left(\mathbf{x}\right) = \arccos\left(f(\mathbf{x})\right)/A_f(\mathbf{x}))\,\theta \in [0, \pi) \tag{3.6}$$

The Matlab code used for calculating the Monogenic signal in this thesis is the implementation of Felsberg's monogenic filters [18], which is available at the official Matlab File Exchange Repository[1] at the submission of Manohar[2]. The implementation uses a combination of convolutions with complexity O(N). It is adapted such that unnecessary calculations are omitted (only calculating the monogenic phase) and to work in 3D. Additional improvements in terms of memory usage are applied. Only one filter at one scale is calculated and therefore the only parameter used is the wavelength of the filter. The parameter ratio of the standard deviation is fixed at the, by Manohars suggested, default value of 0.65. An example for the monogenic phase on a 2D medical image (Figure 3.2 (a)) can be seen in Figure 3.2 (b). How the phase can be used for the over-segmentation of an image is described in the next section.

---

[1] http://www.mathworks.com/matlabcentral/fileexchange/

[2] 38844-gabor-image-features/content/Gabor_Image_Features/monofilt.m

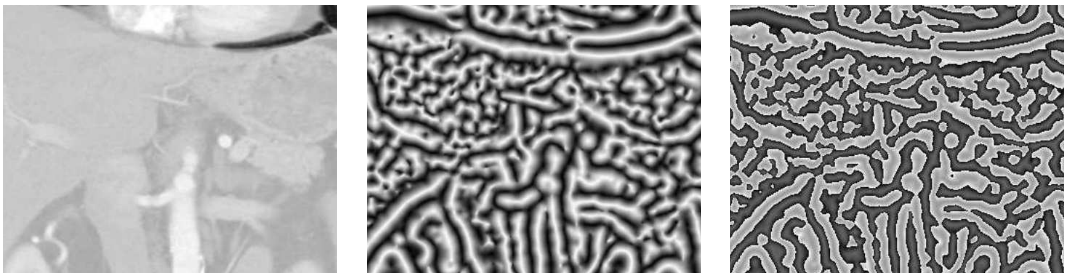

(a) (b) (c)

Figure 3.2: The monogenic phase calculated on the 512×624 Abdomen CT image with a filter wavelength of 18.3 (zoomed in view). (a) is the original image, (b) the local phase φ of the image and (c) the transformed signal Λ as used by our approach.

3.2 Information Representation

In the previous section we indicated that the phase contains edge-like information. We now propose how that information can be transformed and used for image segmentation.

Edge-cues from Monogenic Phase The idea behind how the monogenic phase information can be represented for segmentation is explained in Figure 3.3. In the first row two example 1D signals are shown, a simple sinusoid on the left(a) and one with different structure on the right (b). In the second row the corresponding phase φ is shown for both signals in (c) and (d). It can be observed that at a value of $|\frac{\pi}{2}|$ the phase correlates with the change in structure. Due to the non-symmetric rectangular part of the structured signal (b) the phase is forced to jump from $\frac{\pi}{2}$ to $\frac{-\pi}{2}$ as observed in (d,f). Taking the absolute value of φ removes the problems caused by the wrapping of the phase. We now create a horizontal line every time the phase intersects a value of $\pm\frac{-\pi}{2}$ (e,f). Using these lines as segmentations the original signals are now segmented into three structural parts (g,h).

The monogenic phase $\varphi(x)$ picks up the locally dominant structure in a n dimensional signal. Its values are between $-\pi, \pi$ and a value of $\pm\frac{\pi}{2}$ correlates with a strong edge, as previously illustrated in Figure 3.1 and Figure 3.3 and for a 2D image in Figure 3.2. We now want to map this information such that there is a high gradient when the phase has a value of $\pm\frac{\pi}{2}$. In other words, we want to get a representation with similar feature values within superpixels and value changes at their boundaries. This is done by first mapping the monogenic phase φ to $\varphi_{new} = |\varphi| - \frac{pi}{2}$ before calculating the final monogenic phase cue with

$$\Lambda\left(x\right) = \frac{sgn\left(\varphi_{new}\right)\exp^{\left(-|\varphi_{new}|\right)}}{2}, \Lambda \in \left[-0.5, 0.5\right]. \tag{3.7}$$

We divide by 2 to bring the range of the phase Λ to 1. This is done because it will be rescaled in the next stage. The result of the mapping can be seen in Figure 3.2 (c) where the cue now has a high contrast (brightness change from 0.5 to -0.5) at the change of structure of the original image. This is the only information about the image employed for segmentation.

It also has to be decided what filter wavelength λ, governing the scale of the monogenic signal should be used. We propose to link λ with the number of superpixels, as this influences the size of the smallest structure that can be detected in the image, when all superpixels have roughly the same size. The idea behind this is that a lower number of superpixels can only detect large structures and to capture larger structures the wavelength has to be increased. The wavelength therefore correlates to the average size of a superpixel denoted as d_{cc}. Figure 3.4 shows an experiment with the variation of the wavelength by a scaling factor s on the BSD dataset, where $\lambda = s * d_{cc}$. The results show that using a wavelength equal to d_{cc} has the best results in terms of recall.

3.3 Segmentation

We now use the transformed phase Λ as basis for image segmentation as shown in Figure 3.5 (a) on the right. Similar to Achanta 2010 [2], we perform k-means clustering on the image data to obtain the superpixels. In contrast the cluster centers are initialized spatially according to a hexagonal grid with cluster center distance $d_{cc} = \sqrt{\frac{N}{K}}$ as shown in Figure 3.5 (a) on the left, to avoid imposing too much of a directional preference compared to the rectangular grid.

Each pixel is represented in an $nD + 1$ space, with the first n dimensions being the original pixel or voxel coordinates and the additional dimension being the transformed monogenic phase information $\Lambda'(\mathbf{x}) = w d_{cc} \Lambda(\mathbf{x})$, with w specifying the value of how much the monogenic signal is weighted. The brightness and contrast invariant boundary representation based on the monogenic phase makes the regularization parameter of Achanta obsolete. The relation of monogenic phase cue to the spatial distance can be set at a fixed optimal value w. The optimal value was calculated by using variations of w from 1 to 3 on the BSD dataset, which is visualized in Figure 3.6. The recall rate does not change with $w > 2$, therefore the final choice is $w = 2$ which is used for all calculations presented in this thesis.

The final distance used for k-means in 2D is

$$D_m = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (\Lambda'_i - \Lambda'_c)^2}. \tag{3.8}$$

The resulting segmentation after 10 iterations overlaid with the monogenic phase cue can be seen in Figure 3.5 (b). The final segmentation overlaid in the image is shown in Figure 3.5 (c) where we can see that the local dominant structure is segmented by the presented method.

The homogeneity of the phase allows for another change to the k-means computation that decreases the computation time and is discussed in the following paragraph.

Adapting k-means - In this paragraph the k-means algorithm is adapted from the original one. Table 3.1 gives an overview on the number of calculations for three k-means versions, where the first is the original one, the second the adapted version of [2] and the third the one proposed for this thesis. For the rest of this paragraph I is the number of iterations, S the number of seed-points, P the number of pixels and D represents the dimensionality of the data. The variable M describes the maximum size of the region $R \subset P$, e.g. the maximum superpixel size. The adaption to k-means introduced by [1] reduces the number of calculations from ISP to ISR.

original	**Achanta**	**proposed adaption**	
ISP	ISR	$(I-1)SN+SR$	$P > R > N$

Table 3.1: Number of calculation steps needed for each of the k-means algorithms.

In the original version the distances between all seed-points S and pixels P are calculated. The adapted version restricts the distance calculation to a region R around S, more exactly $R = M^D P/S$ for a maximum superpixel size of $M_2 = 2$ times the average region size. Using the transformed monogenic phase response as k-means input makes it possible to further decrease the number of calculations necessary, because the information is represented in a smooth and noise-free way. Therefore instead of R samples only a reduced number of random samples $N \subset R$ is taken, although this time from $M_1 = 3$ times the average region size. Because this sub-sampling does not calculate the labels for all pixels, a final iteration taking SR calculations is added. The total number of calculations is $ISN + SR$ with the amount of random samples $N = nR$ (default value for $n = 0.1$).

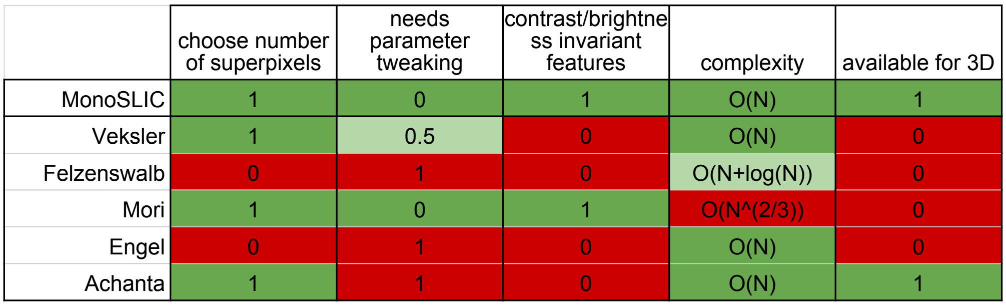
Putting the number of calculations of *MonoSLIC* and Achanta into relation, setting $I = 10$, substituting R and using M_1 for *MonoSLIC* and M_2 for Achanta gives the factor $\frac{1.9M_1^D}{10M_2^D}$. Using the predefined values for M results in a speed-up of factor 2.3 for 2D and 1.56 for 3D for *MonoSLIC* compared to Achanta. Due to the larger M of *MonoSLIC* the speed-up factor decreases with increasing dimensionality and the method would eventually ($D > 5$) need more calculations than Achanta. Setting $M = 2$ also for the *MonoSLIC* approach would result in a speedup of 5.2 independent of the dimension D.

For the first iteration of the clustering, the initial values for the $nD + 1$'st coordinate of the cluster center have to be estimated. These are set to the average of $\Lambda_{new}(\mathbf{x})$ of all pixels closest to that cluster center.

Label Cleanup - After the k-means algorithm a label cleanup is performed similar to Achanta and Mori [37]. This ensures that there are no disconnected clusters and the size of every cluster is larger than a defined minimal value that is based on the image size and the number of superpixels.

3.4 Properties

As both, the calculation of the monogenic signal and k-means clustering are generalized for any dimension, the method can also be used to segment 3D volumes. The user can chose the desired number of superpixels and there is no need to tune an additional parameter. Using the monogenic phase as feature makes the method invariant to contrast and brightness. The restriction of the k-means search space and the random sampling makes the k-means clustering at least 1.6 times faster than the version used by Achanta [1]. As all the parts of the presented method are of the same complexity, also the total complexity of *monoSLIC* is O(N).



	choose number of superpixels	needs parameter tweaking	contrast/brightness invariant features	complexity	available for 3D
MonoSLIC	1	0	1	O(N)	1
Veksler	1	0.5	0	O(N)	0
Felzenswalb	0	1	0	O(N+log(N))	0
Mori	1	0	1	O(N^(2/3))	0
Engel	0	1	0	O(N)	0
Achanta	1	1	0	O(N)	1

Table 3.2: Overview of the limitations of the state of the art over-segmentation methods compared with the presented method MonoSLIC.

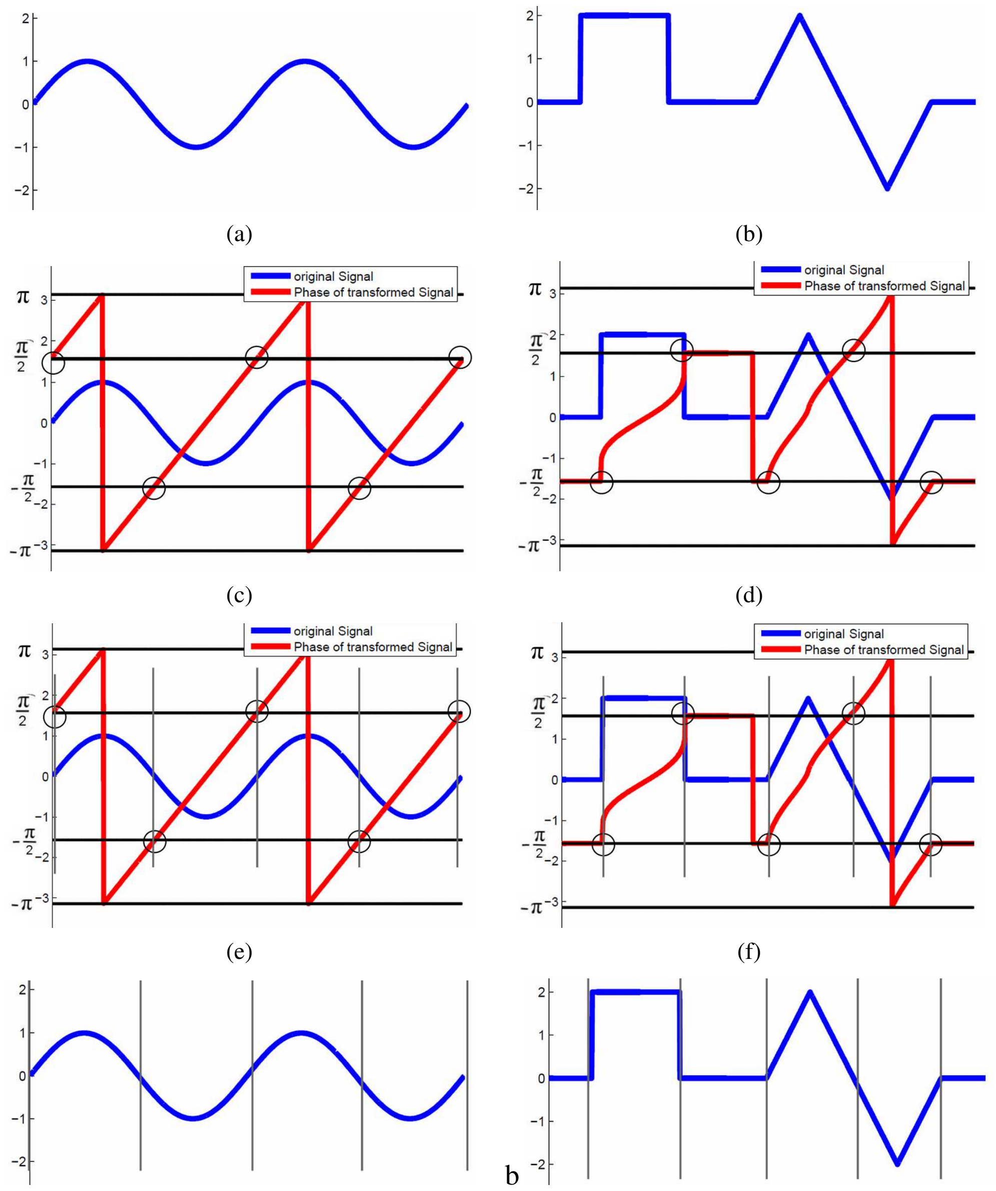
3.5 Comparison

Compared to the state of the art methods, in terms of feature extraction *monoSLIC* is similar to the method of Mori [37], but using only texture information in form of the phase of the monogenic signal and no gradient information. The creation of the segmentation is similar to Achanta [1], but instead of the rectangular grid k-means is this time initialized using a hexagonal grid. The distance function used by k-means is based on the spatial information and the transformed phase of the monogenic signal. Because the phase is independent of contrast and brightness and always has a value between 0 and π there is no need of an additional parameter as opposed to Achanta. The number of superpixels is based on the number of hexagons and can be therefore set directly. The result is a regular over-segmentation where the edges align with local dominant change in structure. The over-segmentation of the two example images Ox and Cannon from the BSD dataset are shown in Figure 3.7. Each image is divided into three parts where each part has a different number (from 750 to 150) of pixels per superpixels.

3.6 Summary

The method *monoSLIC* is summarized in Figure 3.8. It overcomes the limitations of the state of the art approaches as shown in Table 3.2 and is based on the representation of an image or volume using the monogenic signal. It decomposes the image into three orthogonal features called amplitude, orientation and phase. *MonoSLIC* uses the **contrast and brightness invariant** phase which indicates the locally most dominant structure as the only feature for segmentation. Before the segmentation procedure the phase is transformed to lie within the values of -0.5 and 0.5. A change from 0.5 to -0.5 indicates that at this location is an edge in the image. For segmentation, similar to Achanta [1], k-means clustering is used where the number of clusters corresponds to the **number of superpixels**. Because the value of an edge is always the same in the phase representation there is **no parameter tweaking** needed to balance the ratio between spatial and phase information. By using sub-sampling the computation time of k-means can be sped up by 2.3 for 2D and 1.5 for 3D. Finally label cleanup is performed similar to Achanta and

Mori [37], in order to ensure no disconnected clusters. The **complexity** of the presented method is similar to Achanta with O(N) and the implementation also supports **3D volumes**.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3.3: Visualization of the segmentation-idea based on the monogenic phase. The first row shows two example signals in blue with a sinusoid signal (a) and a more structured signal (b). In the second row (c,d) the phase of the transformed signal is shown in red with values between $-\pi$ and π. The points where the phase passes the value of $|\frac{\pi}{2}|$ (when no phase-wrap occurs) are marked with a black circle. The segmentation of the signal at these defined points is indicated by the gray lines in the third row (e,f) and the final division of the signal into its structural parts is shown in the last row (g,h).

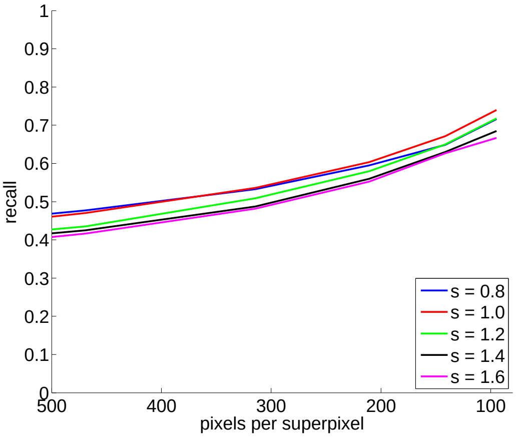


Figure 3.4: Recall for a variation of the scale parameter s on the BSD dataset (higher is better).

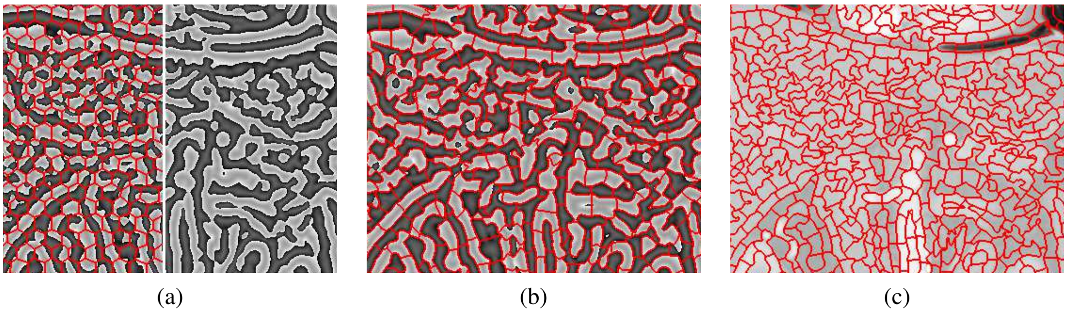

(a) (b) (c)

Figure 3.5: The k-means steps visualized on the zoomed in version Abdomen CT image. In (a) the transformed phase Λ is shown on the right side and the spatial hexagonal cluster center initialization is overlaid on the left. Panel (b) shows the final segmentation of the phase Λ and in (c) the segmentation is overlaid on the original image.

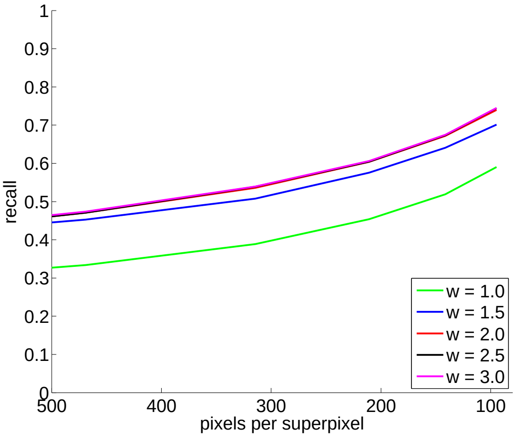
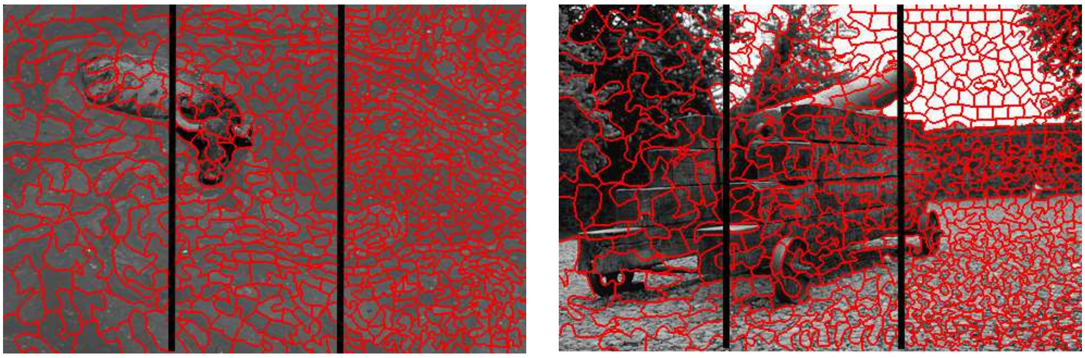


Figure 3.6: Recall for a variation of the weight parameter w on the BSD dataset (higher is better).



(a) (b)

Figure 3.7: Over-segmentation generated by MonoSLIC of the example images Ox (a) and Cannon (b), for 750, 300 and 150 pixels per superpixel.

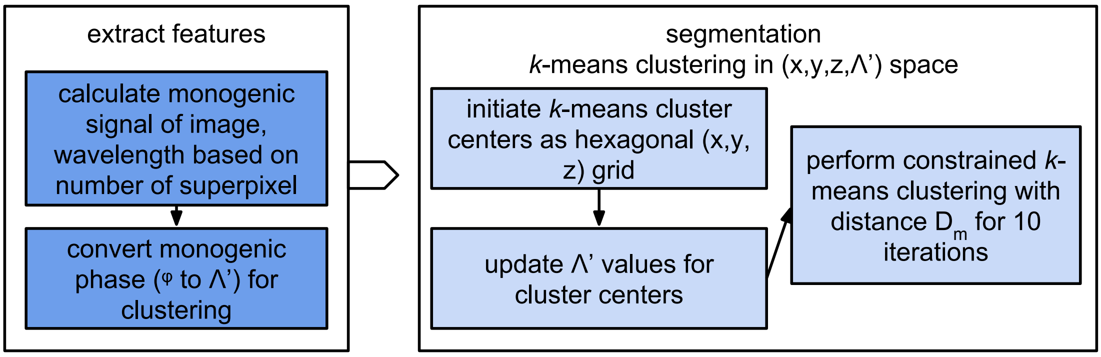

extract features
calculate monogenic signal of image, wavelength based on number of superpixel
convert monogenic phase (φ to Λ') for clustering
segmentation
k-means clustering in (x,y,z,Λ') space
initiate k-means cluster centers as hexagonal (x,y, z) grid
update Λ' values for cluster centers
perform constrained k-means clustering with distance D_m for 10 iterations

Figure 3.8: Overview of the MonoSLIC method.

CHAPTER 4

Results

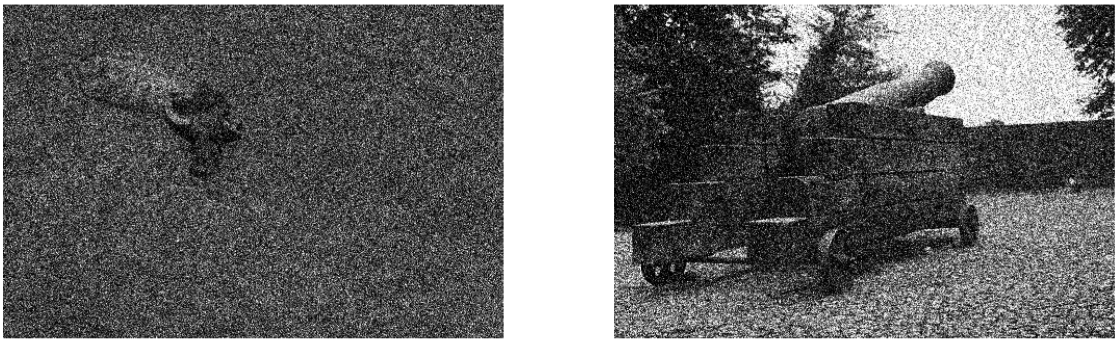
The compared state of the art algorithms and the presented method *MonoSLIC* are compared using two 2D (BSD and medical images) and *MonoSLIC* and Achanta are also compared on three 3D (Aorta, Fetal and VISCERAL) datasets. They are evaluated in terms of recall and precision, in percentage as well as the absolute boundary pixel error. Additionally regularity properties like variation in area and similarity to circle are compared. For reference purpose a rectangular and a hexagonal grid are added as baseline, to show how a simple over-segmentation would perform. In the figures of this thesis Grid stands for the rectangular grid, while Hexagon is the hexagonal grid. The algorithms are also compared in terms of Gaussian noise and *MonoSLIC* is explicitly compared to the Achanta method for a variation of Gaussian noise in the discussion. Finally also the runtime performance of the available implementation is also compared for 2D images and 3D volumes.

4.1 Evaluation Measures

This section describes the evaluation measures used for comparison. It contains accuracy rating in terms of recall and precision, robustness and regularity of the over-segmentation output.

Boundary Pixel Recall Error The boundary error measurement describes the accuracy of an algorithm. It is the average euclidean distance of the annotation of an image to the segmentation of the algorithm and is measured in pixels. The lower the boundary recall error the better the segmentation.

Boundary Pixel Precision Error The boundary precision measurement describes the efficiency of an algorithm. It is the average euclidean distance of the segmentation of the algorithm to the nearest annotation of the image and is measured in pixels. A higher boundary precision error indicates more unnecessary over-segmentation.



(a) Ox image (b) Cannon image

Figure 4.1: Two example images Ox and Cannon pollutated with Gaussian Noise with $mean = 0$ and $std = 0.22$.

Recall Rate The recall rate describes the percentage of how many of the annotated pixels were detected by the algorithms segmentation. A higher recall value indicates a better detection of the annotated boundaries. It is the most relevant measurement of over-segmentation algorithms, because every object boundary not detected is a loss of information. Before calculating the recall (or precision) the pixel wise annotation is symmetrically extended by 1 pixel. This is done because the real boundary cannot be exactly verified due to either aliasing effects or blurred object boundaries.

Precision Rate The precision rate describes the percentage of how many pixels in the algorithm's segmentation match the annotation of the image. A higher precision indicates less unnecessary over-segmentation of the algorithm, in other words how much noise in the segmentation output of the algorithm is. As the nature of discussed methods is over-segmentation they naturally generate more boundaries than object boundaries in the image and have a precision rate that is lower than methods that only segment the objects.

Precision-Recall Curve The precision-recall curve combines the two properties in one plot. It visualizes the correlation between precision and recall. Curves that are closer to the top right indicate a better precision-recall result.

Influence of Noise In order to measure the robustness against the influence of noise the images are disturbed with a Gaussian noise before segmented by the algorithms. The noise is added with a mean of 0 and increasing standard deviation from 0.032 to 0.32 as shown in the two example images in Figure 4.1. The values were chosen so that the result is similar to the noise in an OCT 1.4 (b). This test evaluates the robustness of the algorithms to noise, as it can occur in medical images and images under low light conditions.

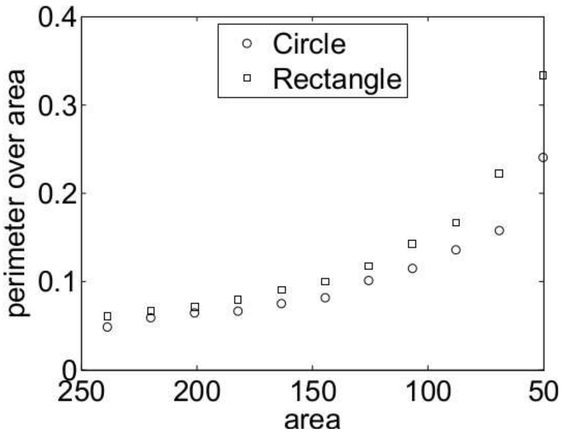


Figure 4.2: Perimeter over area for a rectangle and circle with decreasing size in pixels.

Regularity of Superpixel The regularity of superpixels can be described by the perimeter and area as description features. The perimeter describes the length of the boundary describing the superpixel, while the area represents the pixels filled by the boundary. The first idea was to take their direct relationship calculated by dividing the perimeter by the area. This is illustrated in Figure 4.2 for two shapes, a circle and an rectangle. The drawback of this is the dependency of the direct relationship on the size of each form, therefore a value suggested by Schick et al. 2012 [45] is used for comparison. The proposed value represents the similarity of the superpixel shape compared to a circle using Equation 4.1.

$$4\pi \frac{area}{perimeter^2} \tag{4.1}$$

4.2 Datasets

The two 2D and three 3D datasets used for evaluation of the over-segmentation methods are explained in this section.

2D - Berkley Segmentation Dataset - BSD The BSD [34] consists of a set of 500 natural color images with a resolution of 0.15 MP. The images were annotated by 30 different subjects such that each image has at least 5 annotations. Since the focus of this work is on medical images where no color exists, the color images are converted and used as gray-scale images. Two examples images, the Ox and the Cannon, and two of their corresponding annotations are shown in Figure 4.3 and Figure 4.4

2D - Annotated Medical Images - 2D VISCERAL The medical image dataset is extracted from the VISCERAL [27] dataset, which is described in more detail in the next paragraph. It consists of 28 extracted 2D coronal center slices with their corresponding annotation with an average size of 0.18 MP and will be called *2D VISCERAL* in the remainder of this thesis.

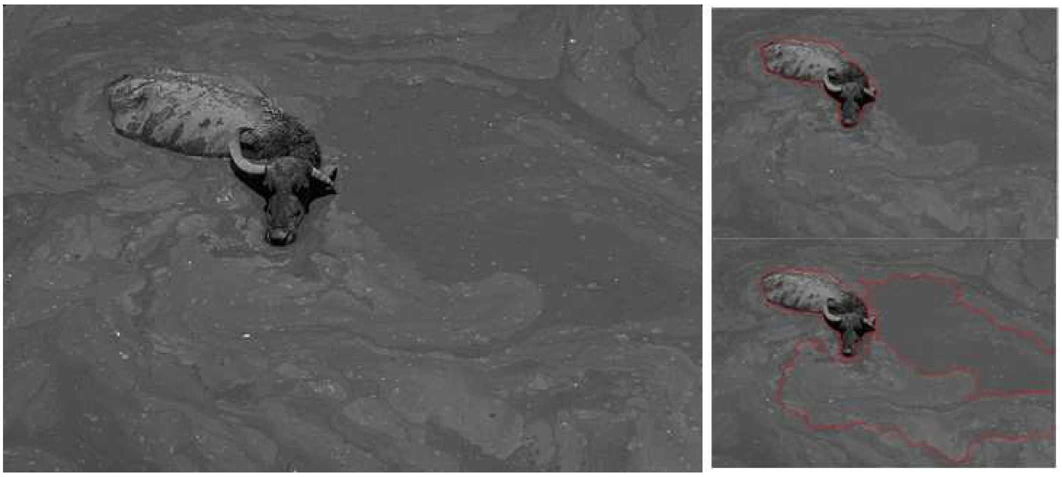


Figure 4.3: Original Ox image and two example annotations taken from BSD [34] overlayed in red.

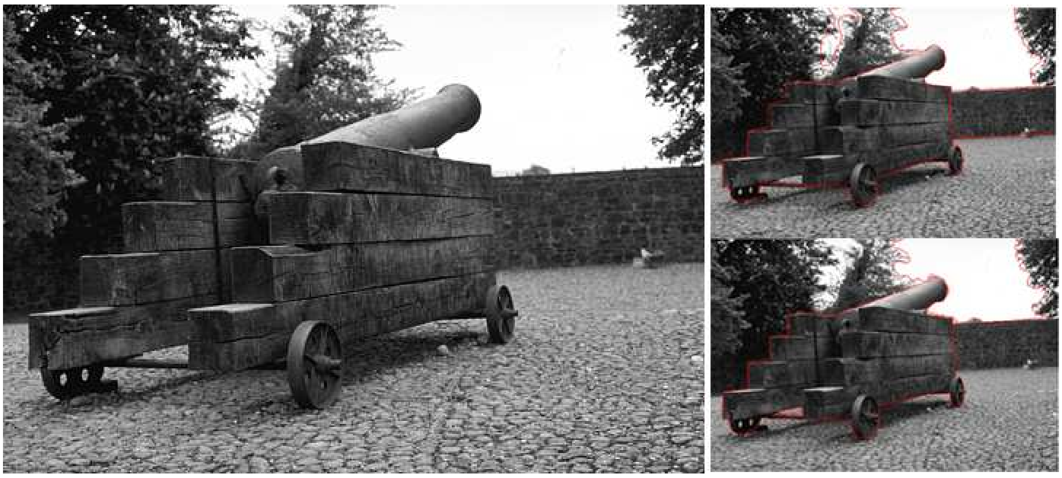
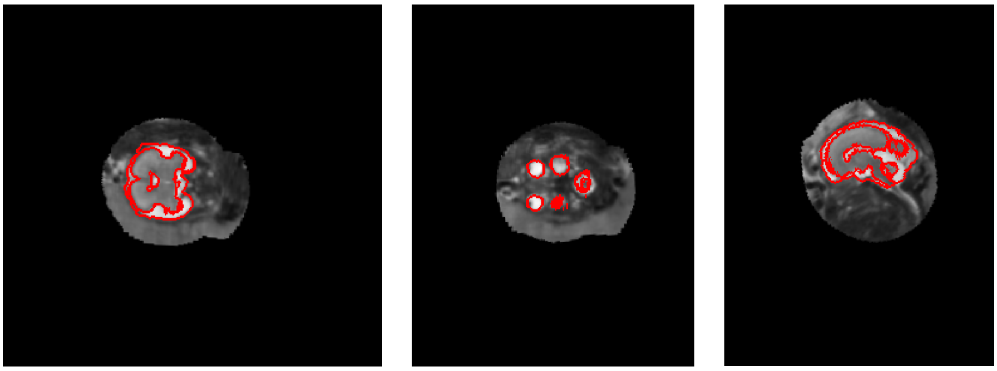


Figure 4.4: Original Cannon image and two example annotations overlayed in red from the BSD [34].

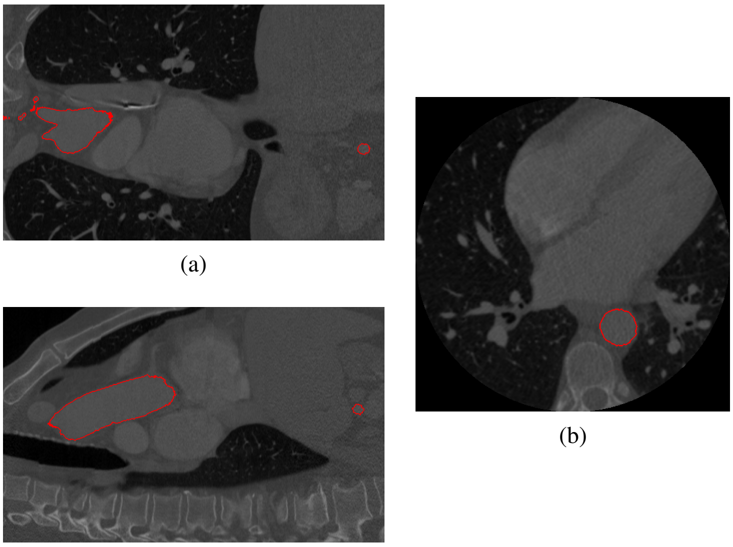
3D - Annotated Medical Volumes - Fetal The *Fetal* dataset was generated from Baby Brain Toolkit [44]. It consists of 33 isotropic Fetal MRI volumes with 1 mm voxel spacing and a resolution of 22.18 MP, where medical experts annotated the left and right eye, as well as the ventricles. The size of annotations is small when compared to the image size and resolution as the examples slices show in Figure 4.5.

3D - Annotated Medical Volumes - Aorta The *Aorta* dataset was created for a study performed by Schwartz et al [46]. The dataset consists of 9 thorax CT volumes with a slice resolution of 512×512 and pixel spacing of 0.27 mm to 0.39 mm. The number of slices varies,



(a) (b) (c)

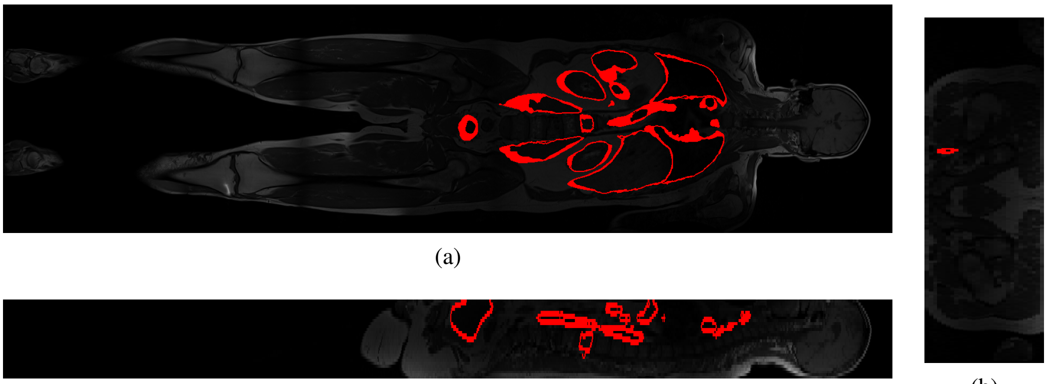
Figure 4.5: Three center slices of each axis for the Fetal MR [44] dataset with overlayed segmentation in red.



(a) (b) (c)

Figure 4.6: Three center slices of each axis for the Aorta CT [46] dataset with overlayed segmentation in red.

resulting in volume sizes from 189.27 to 252.45 MP with a slice spacing of 0.67 mm. In these volumes the thoracic aorta was annotated by medical experts, with center slice examples shown in Figure 4.6.



(a)

(c)

(b)

Figure 4.7: Three center slices of each axis for the VISCERAL MR dataset [27] with overlaid segmentation in red.

3D - Annotated Medical Volumes - 3D VISCERAL The *3D VISCERAL* dataset is named after the Visual Concept Extraction Challenge in Radiology Project[1] that provides an evaluation framework for big data [27]. It consists of 14 CT and 14 MR mixed thorax/abdomen and full-body volumes. The CT volumes have an average size of 167.27 MP with a slice spacing of 1.5 mm, while the MR volumes have an average of 14.02 MP with 4.75 mm slice spacing. Results are presented for CT, MR and a combination of both. It is the most diverse 3D dataset in terms of modality and annotation, as medical experts annotated 20 different anatomical regions in the Thorax and Abdomen area. Examples for the medical volumes and their annotations are shown for the CT in Figure 4.7 and for MR in Figure 4.7 volumes.

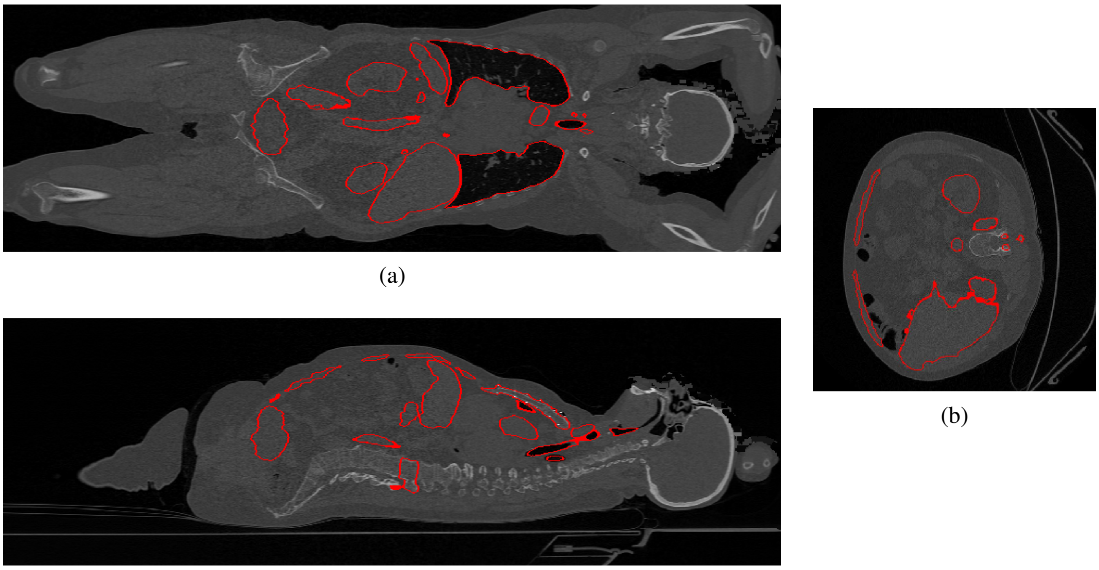
4.3 Parameter Choice

For the algorithm of Achanta an additional parameter p to the number of superpixel has to be set. As described in Section 2.4 it regulates the superpixel regularity at the cost of accuracy. In order to chose a good trade off between those properties, the algorithm is compared to the *MonoSLIC* method in terms of recall, perimeter over area mean and standard deviation. The values are calculated for the BSD dataset and for 100 pixels per superpixel. For each properties the parameter of Achanta for the corresponding *MonoSLIC* value is taken and the average of the resulting three parameters is the final parameter for evaluation. This is visualized in Figure 4.9, where panel (a) shows the perimeter over area mean with $p = 10$, panel (b) the corresponding standard deviation with $p = 22$ and in panel (c) the recall is shown with $p = 14$. The resulting final value for the parameter is $p = 15$, which is used throughout this thesis.

The parameters of Felzenswalb and Engel are varied in order to create segmentations that consist of 100 up to 500 pixels per superpixel.

---

[1] http://www.visceral.eu



(a)

(b)

(c)

Figure 4.8: Three center slices of each axis for the VISCERAL CT dataset [27]with overlaid segmentation in red.

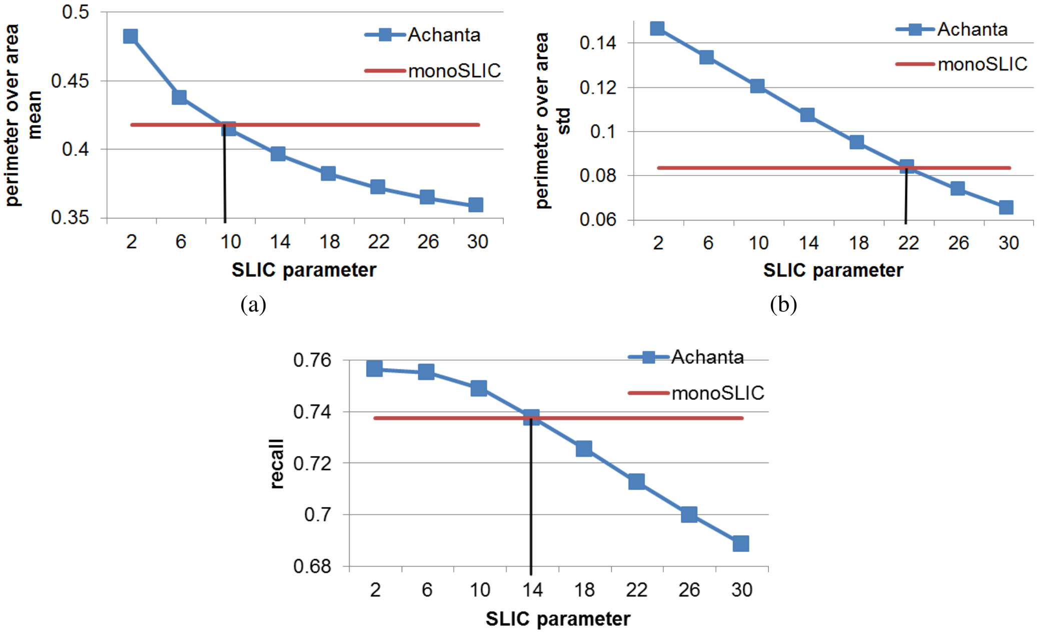
4.4 Results on 2D Images

In this section the performance results for the BSD and Medical Images dataset are presented for algorithms of *MonoSLIC*, Achanta [1], Veksler [48], Mori [37], Engel [15] and Felzenswalb [20]. Additionally the Hexagon and rectangular Grid are used as baseline. The results are calculated from 500 pixels per superpixel up to 90 pixels per superpixels. The results are compared at a value where the reference Hexagon reaches a recall rate of 0.5. This would be similar to make the decision if there is a boundary or not by flipping a coin at each pixel.

4.4.1 Berkley Image Segmentation Database - BSD

On the BSD the algorithm performance in terms of recall, precision, influence of noise and regularity is compared. The section concludes with a summary of the results.

Recall - In Figure 4.10 the results for recall (a) and the boundary pixel recall error (b) on the BSD [34] images are presented. The references of using a rectangular and hexagon grid show how a blind over-segmentation would perform. The slightly better results of 0.54 for the hexagon of are due to the naturally more meaningful shape compared to a rectangle 0.50 for 100 pixels per superpixel. The approaches of Mori 0.84 and Felzenswalb 0.77 have the highest recall accuracy followed by Achanta 0.73, Engel 0.72 and Veksler 0.71 for 100 pixels per superpixel (where the Hexagon has a recall rate of about 0.5. The *MonoSLIC* approach 0.74 has similar results to the latter ones when using a higher number of superpixels, put performs worse when



(a) (b)

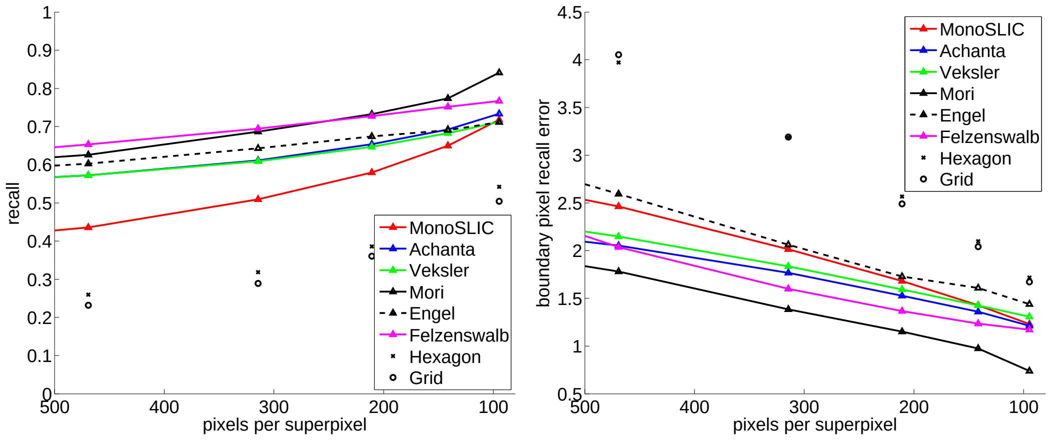
(c)

Figure 4.9: Decision on the choice of the regularity parameter for the Achanta algorithm used in the evaluation process based on the properties perimeter over area mean (a) and std (b), as well as recall (c) for 100 pixels per superpixel on the BSD dataset.

reducing the superpixel count. The ranking of the boundary pixel recall error is similar to the recall. It is lowest for Mori followed by Felzenswalb, Achanta and Veksler while Engel has the highest error. *MonoSLIC* behaves similar to the recall results.

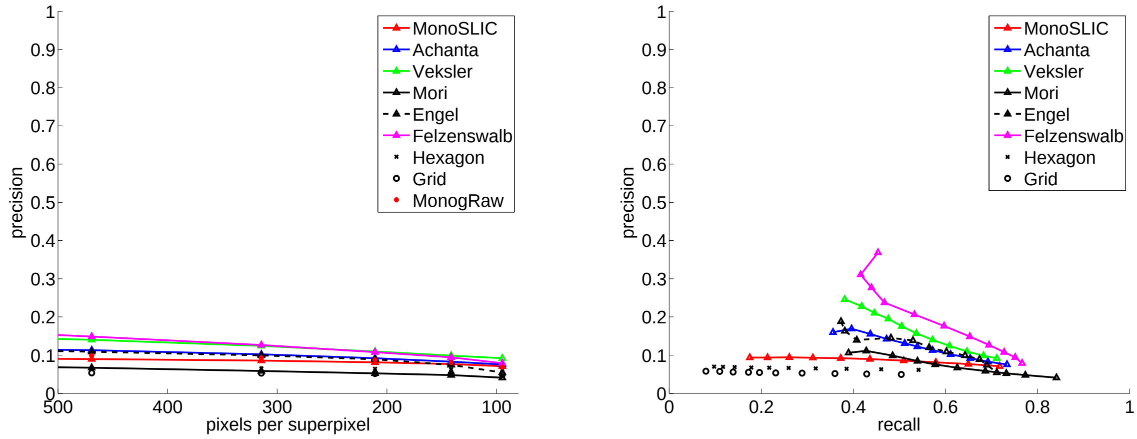
Precision - In Figure 4.11 the precision rate (a) and the precision over recall curve (b) are shown. Felzenswalb and Veksler have the highest precision performance, followed by Engel, Achanta, *MonoSLIC* and Mori. Due to the unconstrained area approach Felzenswalb outperforms the other approaches in terms of precision over recall (b). The precision generally decreases with increasing recall rates. The low precision rates of all algorithms are due to the design of over-segmentation methods.

Influence of Noise - In order to test the algorithms for their stability to Gaussian noise the BSD images were polluted with Gaussian noise with mean 0 and variance 0.05. The results in Figure 4.12 (a) show that Mori performs best for the noise polluted images with a recall of 0.74 for 150 pixels per superpixel. Felzenswalb 0.71 has similar performance when the number of pixels per superpixel is low, but performs poorly when the superpixels are larger. Engel has an average performance with a recall rate of 0.67 followed by *MonoSLIC* with 0.65. Veksler performs better when the superpixels are large, but has the worst results for smaller superpixels with 0.60. Achanta has a similar recall rate with 0.61, but it was not able to generate



(a) (b)

Figure 4.10: Recall (a) and boundary pixel recall error (b) for the BSD image segmentation dataset.



(a) (b)

Figure 4.11: Precision and precision over recall curve for the BSD image segmentation dataset.

a segmentation for less than 150 pixels per superpixel. Due to the Gaussian noise, even setting the number of superpixels to 100 pixels per superpixel returned a segmentation with 150 pixels per superpixel.

In Figure 4.12 (b) the results of the noise polluted images are compared to the original ones. The figure shows the decrease in % between the original and the noisy images. *MonoSLIC* is least affected by the added noise with a almost no recall rate decrease. The recall rate of Mori, Veksler and Achanta is at by about 80% for large to 90% for small superpixels. Felzenswalb is most affected, having only 60% for large superpixels. It is interesting to observe that for

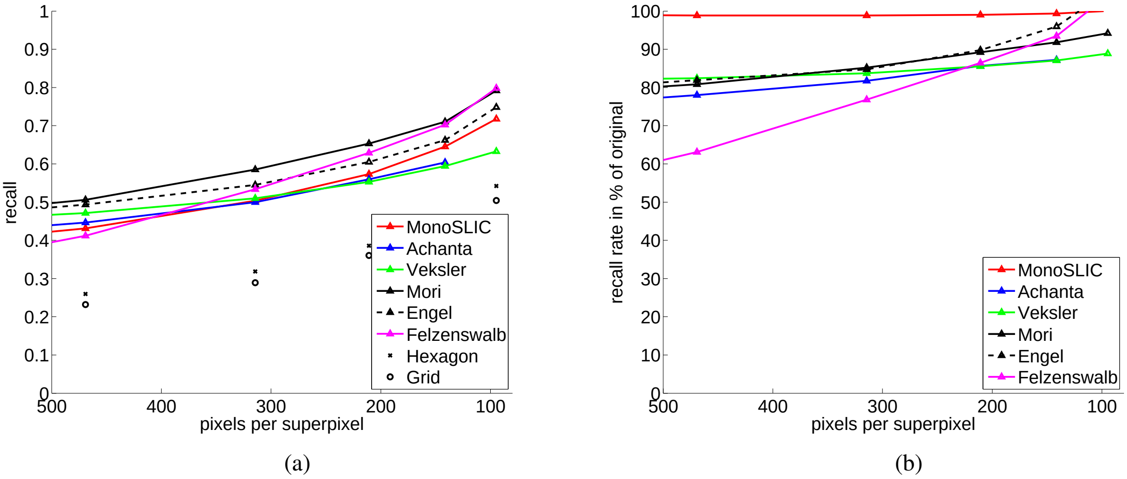
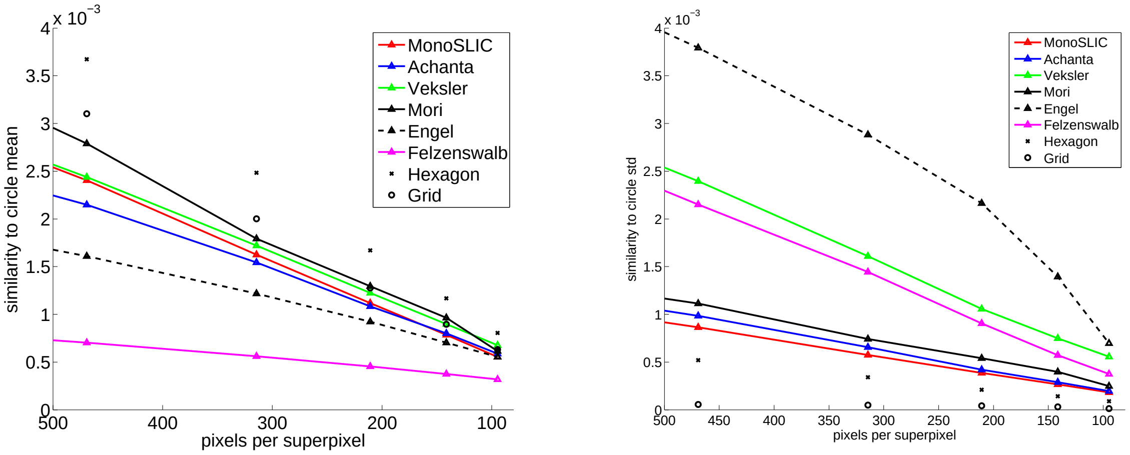


Figure 4.12: The recall rate with the applied Gaussian noise is shown in (a). In (b) the difference of the recall on the original and noisy images is visualized.

small superpixels the recall performance of Felzenswalb and Engel is actually higher than on the original images.

Regularity - The first feature for the regularity comparison is a value that describes how similar the average shape of a superpixel is to a circle. The mean regularity is calculated over all BSD images and represents the average regularity of the algorithm output. The corresponding standard deviation shows the variety of smoothness of the segmentation output for each algorithm. The results are shown in Figure 4.13, where both grids have the smoothest boundaries (a) and the smallest variation (b). Achanta, *MonoSLIC*, Mori and Veksler have the a similar average regularity, but *MonoSLIC* is the most and Veksler the least regular one in terms of variation. Engel and Felzenswalb have a higher irregularity due to their missing constraints on the superpixel size, with Engel having the highest variation. The results area also visualized using the Cannon example image from the BSD in Figure 4.14, where each superpixel is labeled with its corresponding similarity to the circle value. If the shape is similar to a circle it is marked as blue and red if its very irregular. The images reflect the previous numbers, with the reference Grids have the most similarity to a circle with no variation and Felzenswalb having the highest irregularities.

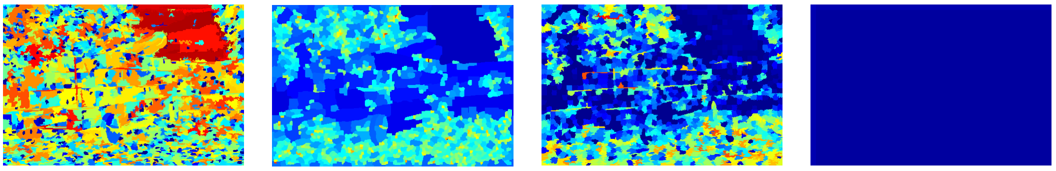
As second regularity feature the variance in area of the superpixels is analyzed. The results in Figure 4.15 (a) show the actual average pixels per superpixel for each algorithm. In the ideal case all the values would be on one line. The variance in (b) on the other hand represents the variation in superpixel size for each algorithm. As expected due to the missing constraints Felzenswalb shows the highest variation in area, followed by Engel. From the algorithms with regularity constraints Veksler performs worst. But they also presented a method with more regular superpixels as mentioned in Section 2.3.3. The other algorithms, as well as the Grid references, have a similar low variation in area. The results for the area are visualized with the Cannon image



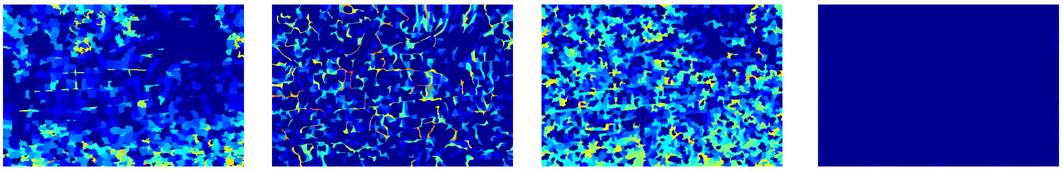
(a)

(b)

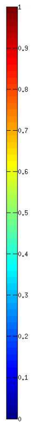
Figure 4.13: Mean (a) (higher is better) and standard deviation (b) (lower variation is better) the superpixel similarity compared to a circle on the BSD dataset.



(a) Felzenswalb (b) Engel (c) Achanta (d) Grid



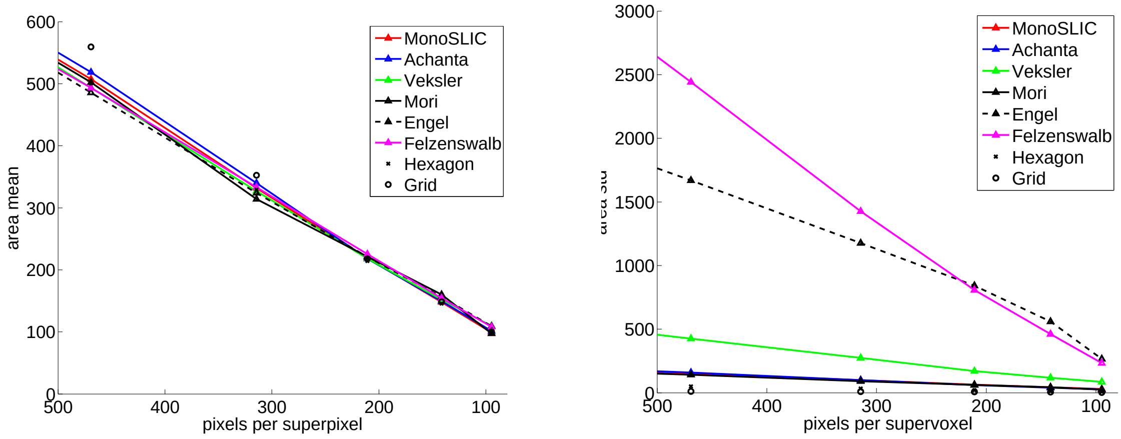
(f) Veksler (g) Mori (h) MonoSLIC (i) Hexagon



(e)

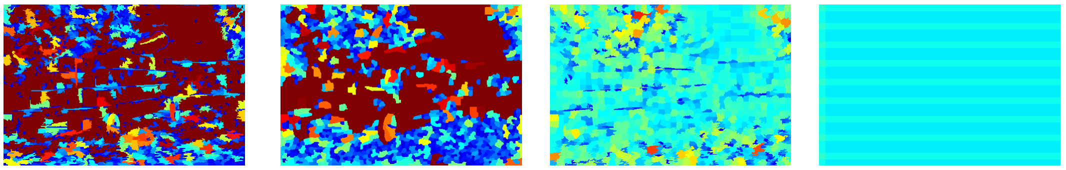
Figure 4.14: Segmentation for the Cannon image with about 200 pixels per superpixel, where each superpixel is labeled with its corresponding similarity to circle.

of the BSD in Figure 4.16, where each superpixel is labeled with its corresponding area value. The maximum area labeled are 400 pixels per superpixel, as a result all superpixels larger than the maximum value have the same color (red). Achanta, Mori and *MonoSLIC* have similar low variation in superpixel size. Veksler shows increased variation and the unconstrained methods of Felzenswalb and Engel in terms of superpixel size have wide variety of superpixel sizes, from very large (blue) to very small (red).

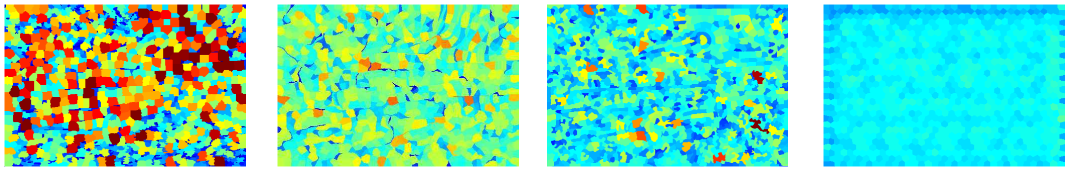


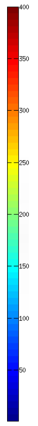
(a) (b)

Figure 4.15: Mean (a) and standard deviation (b) of the superpixel area on the BSD dataset.



(a) Felzenswalb (b) Engel (c) Achanta (d) Grid



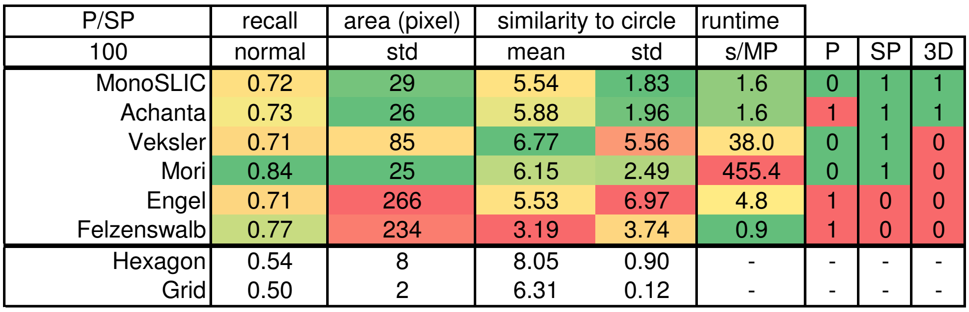


(e)

(f) Veksler (g) Mori (h) MonoSLIC (i) Hexagon

Figure 4.16: Segmentation for the Cannon image with about 200 pixels per superpixel, where each superpixel is labeled with its corresponding area value.

Summary - To get a good overview of all the results for the BSD dataset they are summarized in Table 4.1, where green indicates good performance and desired parameters and red otherwise. The segmentation performance of a rectangular Grid and a hexagonal Grid are also added for reference. The columns presented are recall, standard deviation of the superpixel area, mean and standard deviation for similarity to a circle, the runtime in seconds per MP, if there is a parameter P to set (0 equals no parameter) and if the number of superpixels SP can directly be specified. The values are compared for a number of 100 pixels per superpixel. We can see that Achanta [1] lacks in terms of robustness to noise and also has a parameter that needs to be set. Veksler [48]



P/SP	recall	area (pixel)	similarity to circle		runtime			
100	normal	std	mean	std	s/MP	P	SP	3D
MonoSLIC	0.72	29	5.54	1.83	1.6	0	1	1
Achanta	0.73	26	5.88	1.96	1.6	1	1	1
Veksler	0.71	85	6.77	5.56	38.0	0	1	0
Mori	0.84	25	6.15	2.49	455.4	0	1	0
Engel	0.71	266	5.53	6.97	4.8	1	0	0
Felzenswalb	0.77	234	3.19	3.74	0.9	1	0	0
Hexagon	0.54	8	8.05	0.90	-	-	-	-
Grid	0.50	2	6.31	0.12	-	-	-	-

Table 4.1: Summary of the results on the BSD images for an over-segmentation into 100 pixels per superpixel (the reference Grid having a recall rate of 0.5), where green indicates desired results.

also is influenced by noise and creates superpixel that vary in size and regularity. The runtime discussed in Section 4.6 shows a performance of only $38s/MP$. The method of Mori [37] only lacks in terms of runtime, taking about $455s/MP$. Due to its memory use it only works on images with less than $0.2MP$ and does not work for volumes. The two in terms of number of number of superpixels unconstrained approaches of Engel [15] and Felzenswalb [20] create a segmentation with a high variety in superpixel size. The superpixels are also of irregular shape and a parameter needs to be set. *MonoSLIC* does not lack performance in any of the summarized results, although dropping in recall performance when increasing the number of pixels per superpixel (e.g. creating larger superpixels).

4.4.2 Annotated Medical Images - 2D VISCERAL

For the medical images, only the recall is presented in more detail. The results in terms of variation in superpixel size and regularity are presented in the summary.

Recall - The recall result for the 2D VISCERAL dataset in Figure 4.17 shows that Engel and Mori have the best performance of the methods tested followed by Felzenswalb. *MonoSLIC* has similar results as Felzenswalb for a low number of pixels per superpixel while its performance for a higher number is similar to Achanta and Veksler. In numbers for 120 pixels per superpixel (where the Hexagon has a recall of about 0.5) Mori achieves a recall rate of 0.84, followed by Felzenswalb 0.76 and *MonoSLIC* 0.73. Veksler has a recall rate of 0.67 and Achanta 0.66. Important to note is that for the method of Engel no calculations could be generated where there are less than 100 pixels per superpixel, therefore the value was interpolated to a recall of 0.82. For Mori the calculation of more than 500 pixels per superpixels was skipped due to the high runtime of the method.

Summary - The values measured are again compared in a summary similar to the BSD benchmark and shown in Table 4.2. The largest differences compared to the BSD benchmark are for the method of Achanta and Engel. While Engel has a very good recall performance 0.82 on the

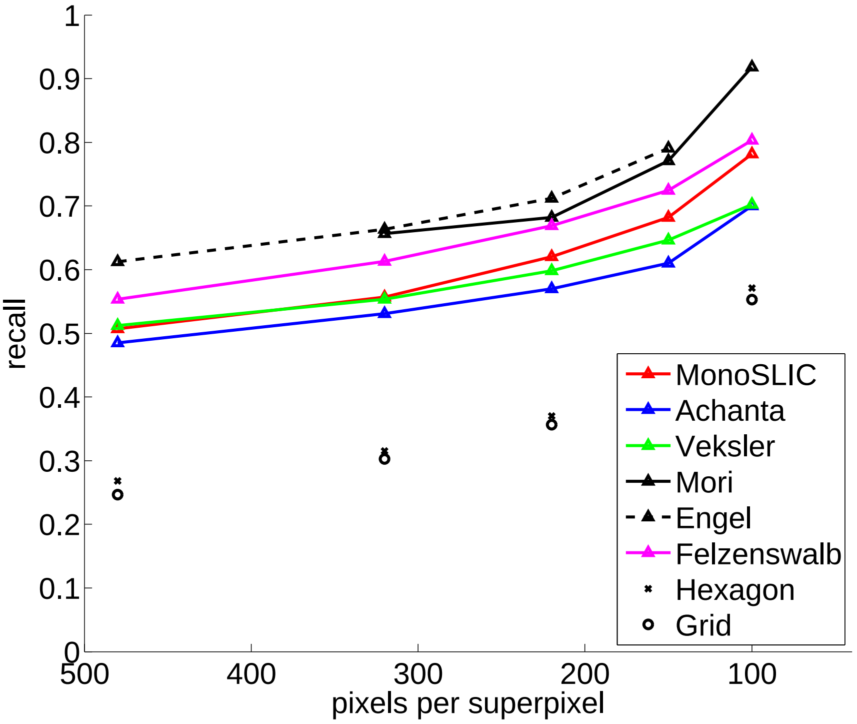
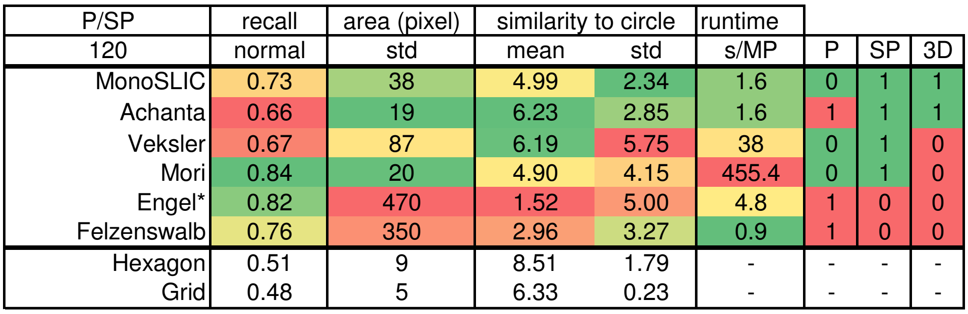


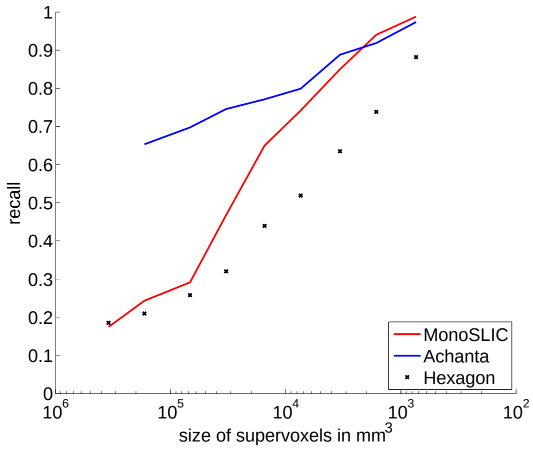
Figure 4.17: Recall for the 2D VISCERAL image dataset.



P/SP	recall	area (pixel)	similarity to circle		runtime			
120	normal	std	mean	std	s/MP	P	SP	3D
MonoSLIC	0.73	38	4.99	2.34	1.6	0	1	1
Achanta	0.66	19	6.23	2.85	1.6	1	1	1
Veksler	0.67	87	6.19	5.75	38	0	1	0
Mori	0.84	20	4.90	4.15	455.4	0	1	0
Engel*	0.82	470	1.52	5.00	4.8	1	0	0
Felzenswalb	0.76	350	2.96	3.27	0.9	1	0	0
Hexagon	0.51	9	8.51	1.79	-	-	-	-
Grid	0.48	5	6.33	0.23	-	-	-	-

Table 4.2: Summary of the 2D VISCERAL results for an over-segmentation of 120 pixels per superpixel, where green indicates desired results. For Engel* the values at 120 pixels per superpixel were interpolated.

medical images the performance of Achanta drops to 0.66. The regularity in terms of similarity to circle is biased by the black background caused by air in the medical images, where Achanta for example has very regular and smooth superpixels when compared to the BSD dataset.



(a)

Figure 4.18: Recall for the 3D Fetal dataset.

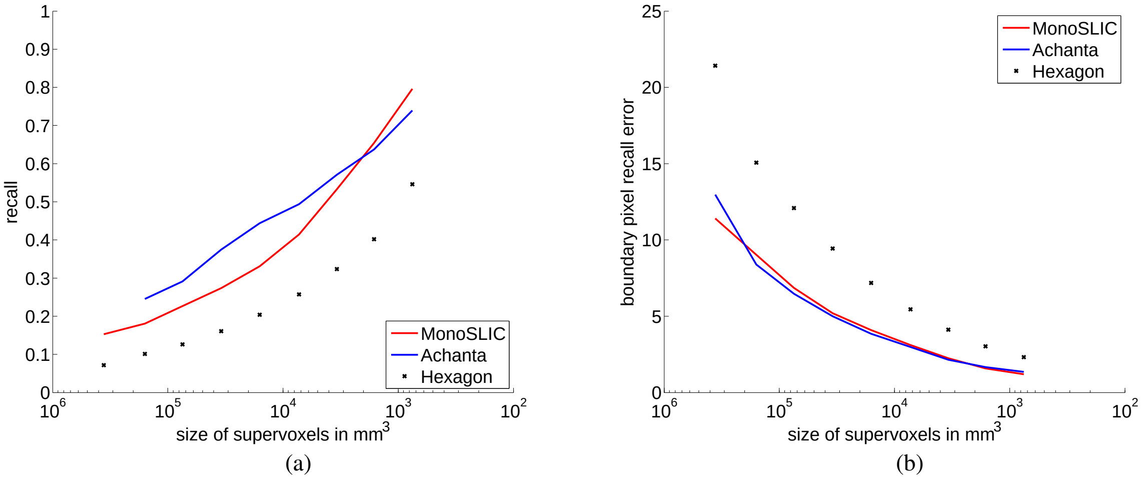
4.5 Results on 3D Volumes

In this section the performance results for the 3D annotated medical volumes are presented. The volumes represent real world recordings and the size of one pixel is known and given in mm^3. As the recordings within each dataset can differ in size the over-segmentations are calculated based on the size of the supervoxel instead of the supervoxel count. Therefore the results are presented for volume size and resolution independent supervoxel sizes of $3 \times 10^5 mm^3$ to $7 \times 10^2 mm^3$. Results are shown for the methods of Achanta [1] and *MonoSLIC*, as these are the only ones able to process 3D data. For reference the performance of the hexagonal grid is also calculated.

4.5.1 3D Fetal

In this section the recall and precision performance of Achanta and *MonoSLIC* are compared on the 3D Fetal dataset.

Recall - The results for the specific annotated structures of the 3D Fetal dataset are shown in Figure 4.18. Comparing the two algorithms in terms of recall, their results correlate similar to the 2D results presented in Section 4.4, where Achanta performs better when using larger supervoxels (low number of supervoxels) and *MonoSLIC* otherwise. For the reference point, where the Hexagon reaches a recall of 0.5, Achanta reaches a value of 0.79 and *MonoSLIC* 0.74 (supervoxel size of $4 \times 10^3 mm^3$. For this dataset the point where *MonoSLIC* performs better than the approach of Achanta is shifted to the far right at $2 \times 10^3 mm^3$. Due to the volume properties of relative small image information and annotations compared to the volume size, the recall rate for *MonoSLIC* is similar to the Hexagon recall for supervoxels larger than $7 \times 10^5 mm^3$. That is because the monogenic response is dominated by the significant black texture caused by the air in the volumes.



(a) (b)

Figure 4.19: Recall (a) and recall boundary pixel pixel error (b) for the 3D Aorta dataset.

Precision - The precision rate is in the range of $0.001 - 0.01$ which is even lower as for the 2D examples, as the relation of annotated pixels to total number of pixels further decreases. This also makes the precision over recall curve obsolete and therefore the results containing the precision rate are omitted for the 3D datasets, as they contain no relevant information.

4.5.2 3D Aorta

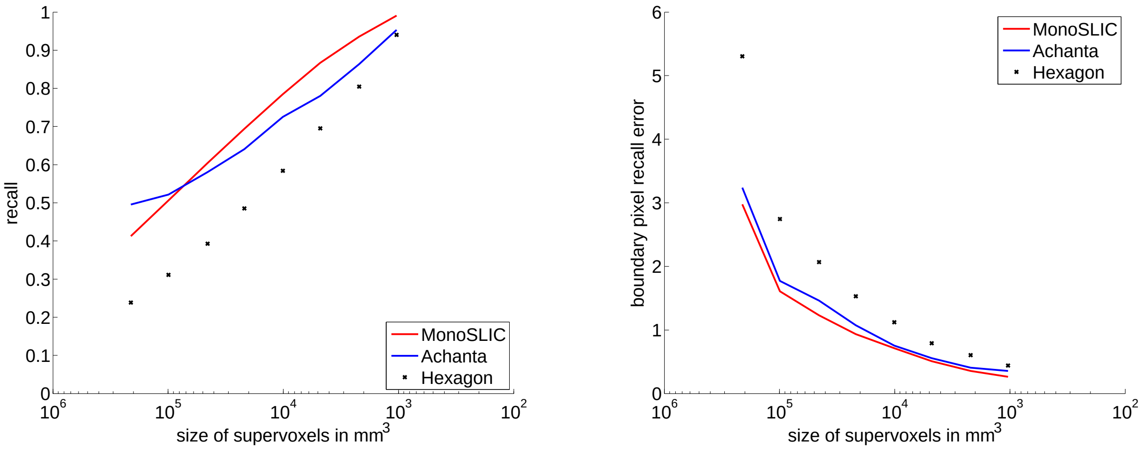
In this section the recall rate of Achanta and *MonoSLIC* are compared on the 3D Aorta dataset. The precision results are omitted as they contain no meaningfulness information.

Recall - The high resolution volumes of the Aorta dataset allow finer annotation, therefore the recall rates in Figure 4.19 (a) for this dataset and the chosen supervoxel sizes, are lower when compared to the other 3D dataset. In numbers for this dataset the recall for $10^3 mm^3$ supervoxels is 0.72 for *MonoSLIC*, while it is close to 1 for the Fetal and 3D VISCERAL datasets. For the reference recall rate of the Hexagon of 0.5 the method of Achanta has a lower recall rate of 0.68 compared to 0.72 for *MonoSLIC*. The recall rate for *MonoSLIC* increases faster than the one from Achanta and surpasses it at $2 * 10^3 mm^3$ and a recall rate of about 0.63, showing again the same properties observed in the previous results. This is also reflected in the the boundary pixel recall error Figure 4.19 (b).

4.5.3 3D VISCERAL

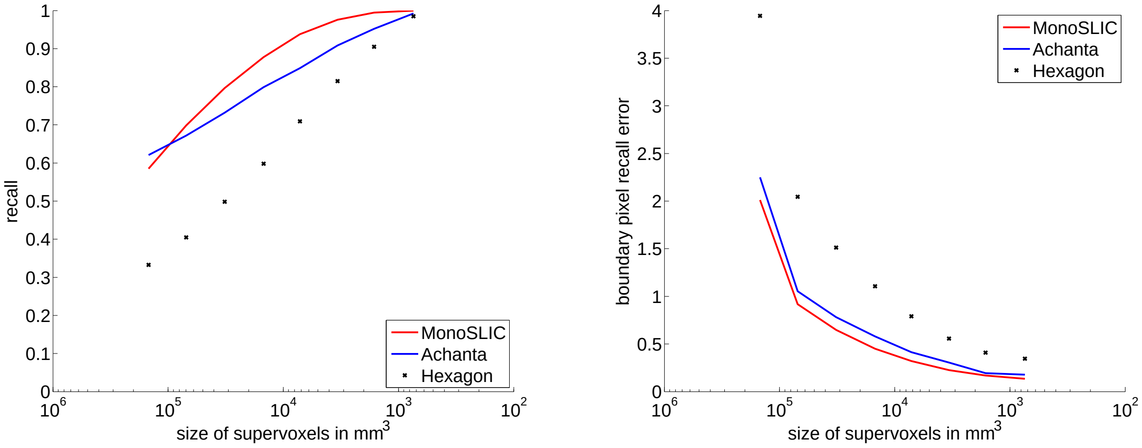
The last dataset of this thesis is divided into MR and CT and the combination of both. The recall and the regularity results are presented with a summary concluding this section.

Recall - For the recall the MR results are presented in Figure 4.20 and the CT results in Figure 4.21. The two show similar recall (a) characteristics, which are also observed in the pre-



(a) (b)

Figure 4.20: Recall (a) and boundary pixel pixel error (b) for the 3D VISCERAL MR dataset.



(a) (b)

Figure 4.21: Recall (a) and boundary pixel pixel error (b) for the 3D VISCERAL CT dataset.

vious results. For MR the recall rate at the reference Hexagon value of 0.5 (supervoxel size of $2 \times 10^4 mm^3$) is 0.68 for *MonoSLIC* and 0.62 for Achanta. For CT it is at $3 \times 10^4 mm^3$ supervoxel size with 0.79 for *MonoSLIC* and 0.72 for Achanta. Due to the wider variety of annotated structures the point where *MonoSLIC* outperforms Achanta is now reached earlier, at a supervoxel size of $10^5 mm^3$ and a recall rate of 0.55 for the MR and 0.65 for the CT dataset. The pixel recall error (b) decreases from an average of 3 pixels to less 0.2 pixels and for both datasets *MonoSLIC* has a lower error than Achanta.

The MR and CT recall curves are now combined in Figure 4.22. For the reference recall (a) value of 0.5 for the Hexagon ($10^4 mm^3$ supervoxel size), *MonoSLIC* has a recall of 0.82 and

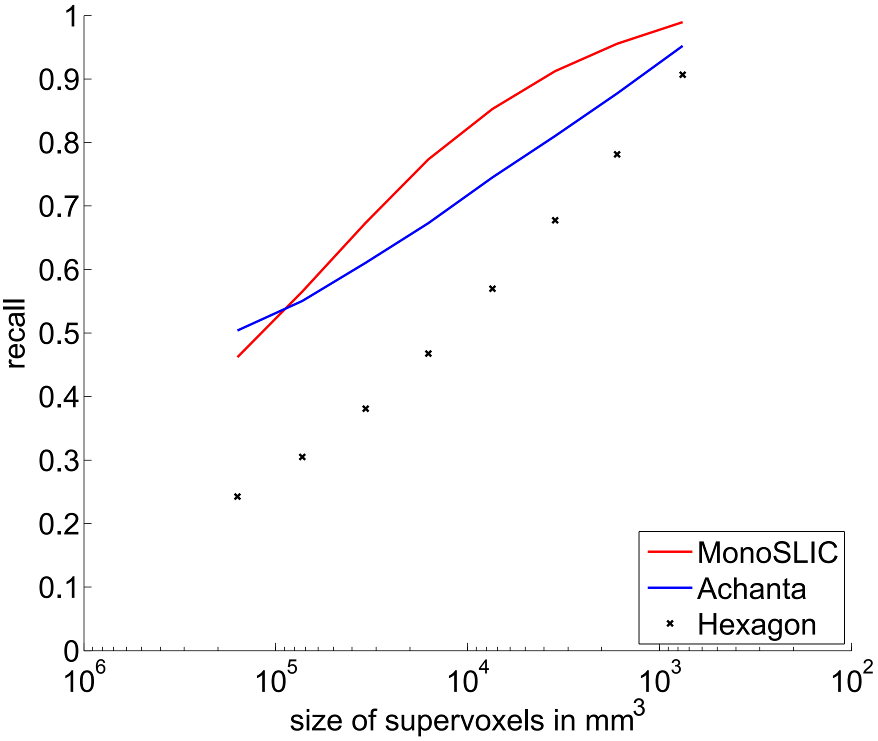


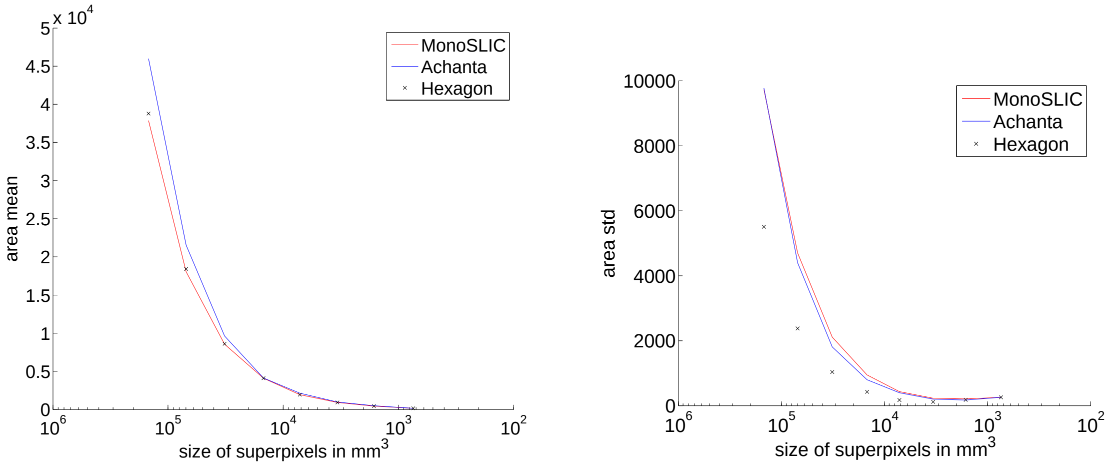
Figure 4.22: Recall rate for the combined CT and MR 3D VISCERAL dataset.

Achanta 0.71. In terms of boundary recall error (b) *MonoSLIC* performs better than Achanta.

Regularity - For the 3D VISCERAL CT dataset also the average and variation of the supervoxel area are presented in Figure 4.23. In (a) Achanta has a slightly higher average supervoxel area for lower number of supervoxels than the Hexagon and *MonoSLIC*. This is an indicator that the implementation of Achanta creates slightly less supervoxels than the set parameter. The standard variation of the area (b) is slightly lower for Achanta than for *MonoSLIC*, which is the result of *MonoSLIC* allowing larger supervoxels and therefore higher variation than Achanta 3.3.

Due to the bias of the black background, causing a regular segmentation of Achanta, which was shown in Section 4.4, the similarity to a sphere value is omitted.

Summary - The performance of the algorithms is summarized in Table 4.3 similar to the summary in Section 4.4.1, where again green indicates good performance and desired parameters and red otherwise. *MonoSLIC* has a higher recall rate, a lower runtime and does not need parameter tuning compared than Achanta [1]. Because *MonoSLIC* allows larger supervoxels the variation in area is slightly lower for Achanta.



(a) (b)

Figure 4.23: Mean (a) and standard deviation (b) of the size of the superpixel on the 3D VISCERAL CT dataset.

mm³	recall	area (pixel)	runtime		
4*10^3		std	s/MP	P	SP
MonoSLIC	🟩 0.82	🟥 994.31	🟩 0.7	🟩 0	🟩 1
Achanta	🟥 0.71	🟩 843.59	🟥 2.1	🟥 1	🟩 1
Hexagon	0.50	442.15	-	-	-

Table 4.3: Summary of the results for an over-segmentation into 4 ccm sized superpixels, where green indicates desired results.

4.6 Runtimes

Before comparing the run-times we have to first think of what exactly is compared. The methods are not implemented in the same programming language and by different people. The method of Veksler for example is only implemented to show the proof of the method and could benefit from a more efficient implementation. Therefore the run-times do not directly relate to the complexity presented in Section 2.6 of each method, but for practical purpose the exact algorithms run-times of how they can currently be used is important. The results calculated on 12 Core Xeon with 74 GB RAM. All algorithms are implemented as single core except for parts of the Mori implementation and the calculation of the Monogenic Signal before clustering on single core again.

The absolute run-times are shown in Figure 4.24 on logarithmic time scale. They are calculated for two different image and volume sizes and the volume run-times are only calculated for algorithms that can process 3D data. The superpixel count for the images was set to 800, for the small and big volumes $7,500$ and $60,000$ respectively. Felzenswalb has the lowest runtime of all algorithms for small images, while Mori is the slowest on all images. With the *MonoSLIC*

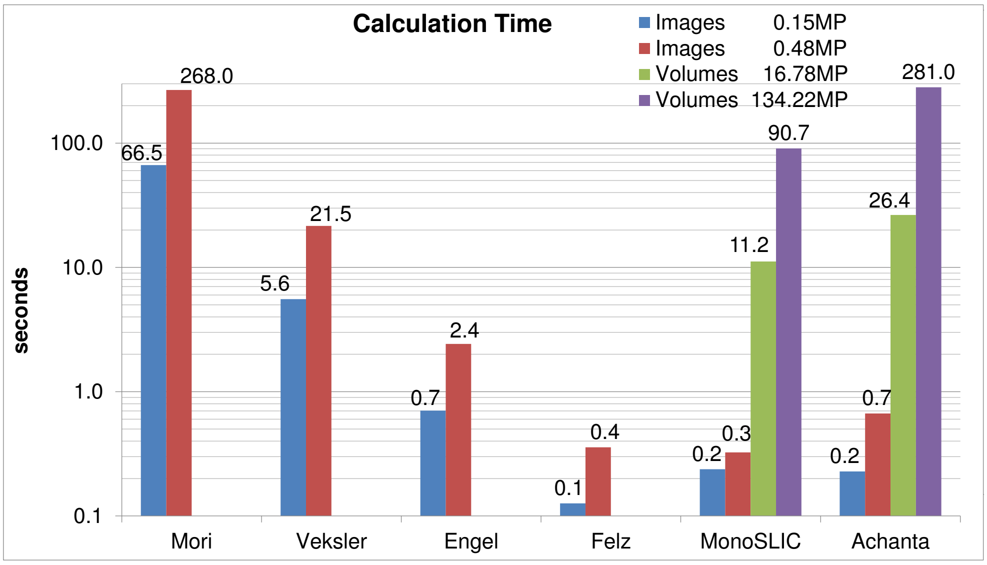


Figure 4.24: Run-times in seconds (logarithmic scale) for all algorithms on 2D images and 3D volumes.

algorithm it is possible to over-segment a volume of size 512^3 in 90 seconds, where Achanta takes 281 seconds.

Another representation of the same data can be viewed in Figure 4.25 where the normalized runtime per MP is shown again on logarithmic time scale. For medium sized images and any volumes *MonoSLIC* is the fastest algorithm with a calculation time of $0.7s/MP$.

4.7 Summary

The methods were compared in terms of recall, precision, influence of noise, regularity and runtime.

Recall - In Table 4.4 all the recall rate values from the 2D and 3D datasets are combined. The average overall recall represents the quality of object boundary detection by the algorithms. With a recall value of 0.83 Mori has the highest recall rate followed by the in terms of superpixel size unconstrained approaches of Felzenswalb 0.78 and Engel 0.76. The two algorithms that are available for 3D have an average recall of 0.74 for *MonoSLIC* and 0.70 for Achanta. With 0.67 Veksler has the lowest recall.

Precision - Due to the design of over-segmentation methods the precision is low when compared to the achieved recall rates. For the examined superpixel sizes of 500 to 100 pixels per superpixel all methods have a precision value of < 0.18. in terms of precision over recall the unconstrained approach of Felzenswalb shows the best performance, achieving (precision,recall)

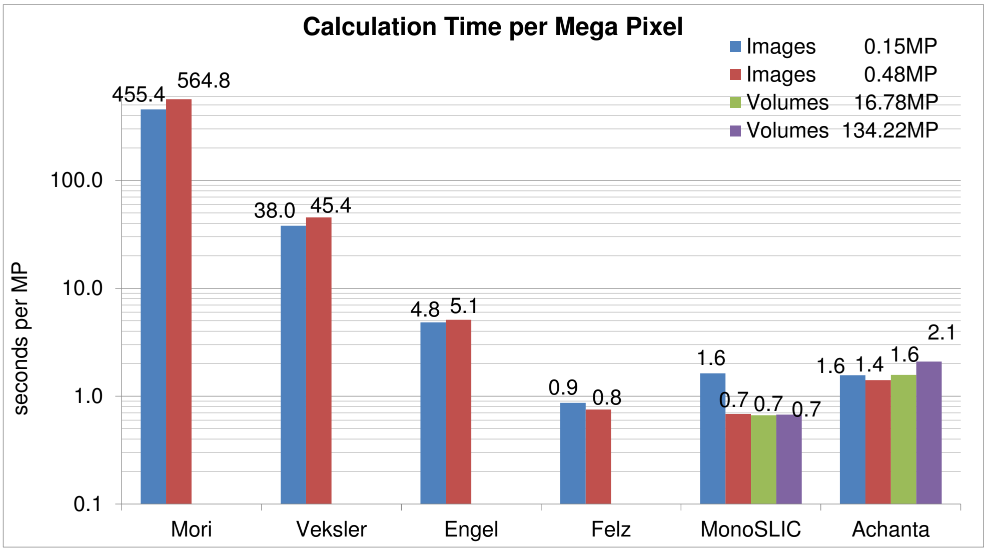
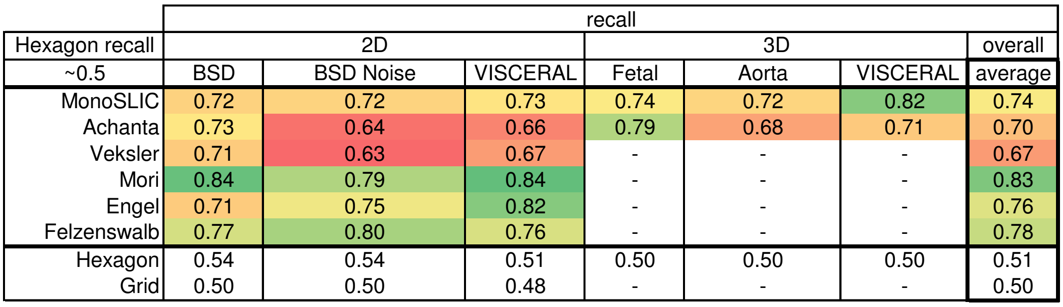


Figure 4.25: Run-times in seconds (logarithmic scale) normalized by Mega Pixel for all algorithms for 2D images and 3D volumes.



	recall						
Hexagon recall	2D			3D			overall
~0.5	BSD	BSD Noise	VISCERAL	Fetal	Aorta	VISCERAL	average
MonoSLIC	0.72	0.72	0.73	0.74	0.72	0.82	0.74
Achanta	0.73	0.64	0.66	0.79	0.68	0.71	0.70
Veksler	0.71	0.63	0.67	-	-	-	0.67
Mori	0.84	0.79	0.84	-	-	-	0.83
Engel	0.71	0.75	0.82	-	-	-	0.76
Felzenswalb	0.77	0.80	0.76	-	-	-	0.78
Hexagon	0.54	0.54	0.51	0.50	0.50	0.50	0.51
Grid	0.50	0.50	0.48	-	-	-	0.50

Table 4.4: Collection of the recall values from each dataset and the average recall.

pairs from $(0.4, 0, 45)$ to $(0.1, 0.77)$, followed by Veksler, Engel, Achanta, Mori and *MonoSLIC*.

Influence of Noise - In terms of influence by noise Figure 4.12 (b) shows that the method of *MonoSLIC* is least affected by noise, with almost the exact same performance 98% as on image without added noise. The other methods only perform at about $80 - 90\%$ of their original performance. Felzenswalb drops to 60% for large superpixels but also, together with Engel, has a recall increase for 100 pixels per superpixel with 105%.

Regularity - The variation in area of the superpixel size indicates how regular the size of superpixels is for the over-segmentation and is summarized in Table 4.5. The approaches of Felzenswalb and Engel do not restrict the size of one superpixel, which is also reflected in the

	Regularity - area std		
Hexagon recall	2D		3D
~0.5	BSD	VISCERAL	VISCERAL
MonoSLIC	29.10	37.91	994.31
Achanta	25.69	19.26	843.59
Veksler	85.38	86.67	-
Mori	25.29	20.37	-
Engel	266.11	469.95	-
Felzenswalb	234.42	350.00	-
Hexagon	7.77	9.34	442.15
Grid	2.26	4.81	-

Table 4.5: Variation in area measured by the standard deviation for the BSD and the VISCERAL datasets.

	Regularity					
	similarity to circle					
Hexagon recall	mean			std		
~0.5	2D BSD	2D VISCERAL	average	2D BSD	2D VISCERAL	average
MonoSLIC	5.54	4.99	5.26	1.83	2.34	2.08
Achanta	5.88	6.23	6.06	1.96	2.85	2.40
Veksler	6.77	6.19	6.48	5.56	5.75	5.66
Mori	6.15	4.90	5.53	2.49	4.15	3.32
Engel	5.53	1.52	3.52	6.97	5.00	5.98
Felzenswalb	3.19	2.96	3.07	3.74	3.27	3.50
Hexagon	8.05	8.51	8	0.90	1.79	1.34
Grid	6.31	6.33	6	0.12	0.23	0.17

Table 4.6: Collection of the recall values from each dataset and the total average recall.

high variation of area, followed by Veksler. The methods of *MonoSLIC*, Mori and Achanta have similar low variation in superpixel area.

Another measurement for regularity is the similarity of the superpixel shape to a circle. This was measured for the 2D datasets and the results are summarized in Table 4.6. The highest average similarity to a circle is achieved by Veksler 6.48 closely followed by Achanta, Mori and *MonoSLIC*. Again Engel 3.52 and Felzenswalb 3.07 have a lower regularity value. The standard deviation indicates how diverse the shape of the superpixels are. This time *MonoSLIC* has the lowest variation with a standard deviation of 2.08 closely followed by Achanta. Mori and Felzenswalb have a variation of about 3.40 and Veksler and Engel have the highest with about 5.8.

Runtimes - For images/volumes with size of at least $0.48MP$ *MonoSLIC* has the fastest runtime as shown in Table 4.4 with $0.7s/MP$. For smaller volumes Felzenswalb is fastest with $0.9s/MP$. Achanta is the third fastest method with about $1.7s/MP$ followed by Engel $5.0s/MP$, Veksler $42.0s/MP$ and Mori $510.1s/MP$

CHAPTER 5

Discussion

In this chapter the results of algorithm performance on the datasets are discussed, as well as the special properties and limitations of the *MonoSLIC* approach. The chapter concludes with suggestions for future work including improvements for the *MonoSLIC* algorithm and the evaluation measurements.

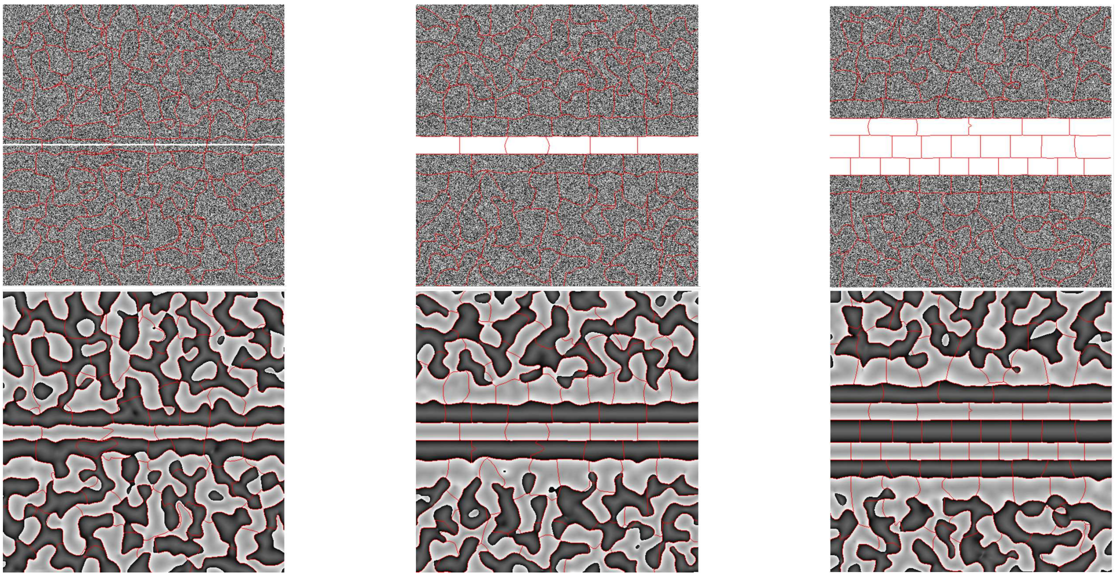
5.1 Monogenic SLIC Properties

In order to create a method where no parameter tuning is necessary the two parameters of the monogenic signal have to be fixed. Parameter σ regulates the ratio of the standard deviation of the Gaussian describing the log Gabor filters transfer function in the frequency domain. The value is set to the default value of $\sigma = 0.65$ suggested by Manohar[1] in the Matlab File Exchange[2]. The second, wavelength, defines the scale of the filter. This is linked to the number of superpixel set by the user. A larger number of superpixel causes a smaller wavelength and therefore smaller texture differences are detected. On the other hand a smaller number of superpixel has an increased filter wavelength of the monogenic signal. An important property of using only the phase angle is that it detects changes in texture. If a structure is smaller than the wavelength of the filter, it will be dominated by other structures and will not be detected. A structure itself will be reflected in the monogenic phase response, but the type of the response depends on the used filter wavelength. This is simulated by creating random values in the range of $0 - 1$ and a horizontal line with value 1 as shown in Figure 5.1. In the upper row the thickness of the line increases, from smaller than the wavelength of the filter in (a), to approximately the same size as the filter in (b), and finally in (c) a line that is thicker than the filter wavelength. The lower row visualizes the corresponding modified monogenic phase response. In Panel (a) the line is not directly recognized by the filter. It still is the dominating local structure and is roughly captured

---

[1] 38844-gabor-image-features/content/Gabor_Image_Features/monofilt.m

[2] http://www.mathworks.com/matlabcentral/fileexchange/



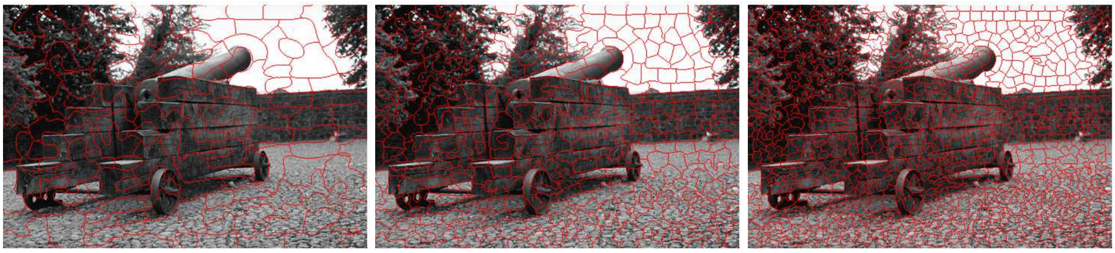
(a) (b) (c)

Figure 5.1: Original image in the upper row and corresponding monogenic phase response in the lower row for increasing edge thickness, where (a) edge smaller, (b) edge about equal and (c) edge thicker than the filter wavelength.

by the larger filter which creates an enveloping boundary that, because it is too large, cannot adhere to the edge boundary. In (b) and (c) the filter is able to directly detect the structure and the segmentation matches the boundaries of the edge. The problem of the wavelength being larger than the object or structure that should be detected is the cause for the bad boundary recall results occurring with low number of superpixels as seen in the results. An example for this is shown on the Cannon image of the BSD dataset in Figure 5.2 for a variation of number of superpixels. With increasing number of superpixels the filter wavelength decreases and therefore also the size of detectable objects. Assuming that the structure that should be detected can be captured by at least one superpixel ensures that the wavelength is small enough and the problem described does not occur. In Figure 5.2 (a) the general structure of trees, sky, wall and the ground are already roughly captured, while the Cannon and its wheels are started being segmented in (b). In (c) the finest structures like the wheels and the rest of the Cannon are detected.

5.2 Monogenic SLIC Performance Scaling

The most time consuming parts of the *MonoSLIC* algorithm are the calculation of the monogenic signal, the k-means clustering and the label cleanup post processing. The analysis is split into 2D (0.26 pixels) and 3D (16.78 MP) and for each the runtime values are calculated for $10,000$ to 10 pixels per superpixel. The tests are run on a 4-core Intel Xeon i5 with 8GB of RAM. Because the monogenic signal calculation is partially multi-threaded its single core time is approximated



(a) (b) (c)

Figure 5.2: Cannon image and overlaid segmentation of the MonoSLIC algorithm for an average of (a) 1000, (b) 400 and (c) 175 pixels per superpixels.

$sr = 1000$	**k-means**	**monogenic**	**label cleanup**	other
2D	24%	59%	3%	14%
3D	43%	43%	5%	9%

Table 5.1: Number of calculation steps needed for each of the k-means algorithms.

by dividing the original monogenic runtime with factor 3, or in other words we assume a speedup of $0.75 \times Core$.

The monogenic signal performance is similar for 2D and 3D resulting in a combined average single core runtime of 0.51 s/MP with a standard deviation of 0.03 s/MP. The runtime per MegaPixel is independent of the size of the image or the wavelength parameter (number of superpixels).

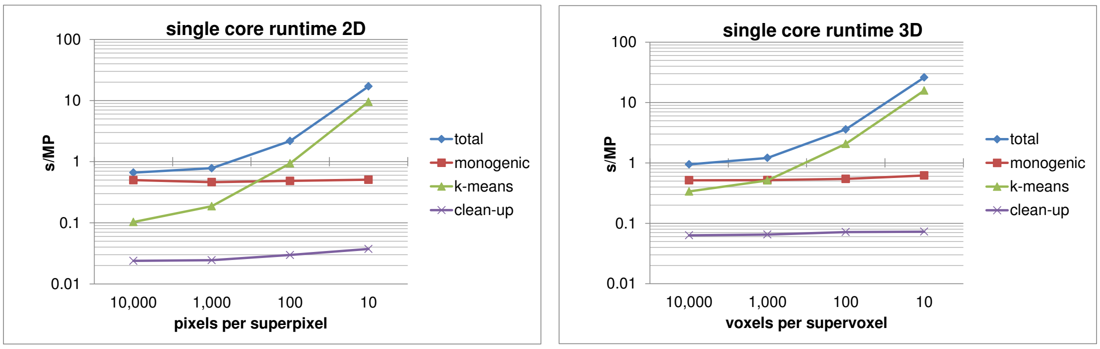
Performance of the k-means algorithm cannot be similarly generalized. When the number of superpixels SP is larger than a fraction *sr* (size reduction) of the number of pixels P

$$SP > \frac{P}{sr} \tag{5.1}$$

its runtime passes the runtime of the monogenic and the experiments showed this at a value of $sr \approx 100$. In Figure 5.3 (a) for the 2D case in numbers this would be at a value of $\frac{0.26 \times 10^6}{100} = 260$ pixels per superpixel. More exactly the min-max runtime is $0.10 - 9.54$ s/MP for 2D and $0.28 - 15.78$ s/MP for 3D.

Cleaning up the labels created by the k-means algorithm is the final process in *MonoSLIC* algorithm. Its runtime, similar to the monogenic signal, is constant in terms of number of superpixels. The average and standard deviation (std) of the increased runtime from 2D with $0.03 + -0.006$ s/MP to $0.07 + -0.004$ s/MP in 3D. Compared to the other parts this still only is a fraction of the total runtime. In numbers for a dimensional reduction of $1,000$ in 2D it is 3% and for 3D 5%.

Because the k-means part is the only inconsistent part in terms of scaling it causes a runtime percentage variation. For an value of $sr = 1000$ the runtimes are summarized in % in the Table 5.1



(a) (b)

Figure 5.3: Single core runtime in seconds per MP in logarithmic scale relative to the number of pixels per superpixel (increasing number of superpixel) for the three most time consuming parts of the MonoSLIC algorithm. Panel (a) shows the values for a 0.26 MP 2D image and (b) for a 16.78 MP 3D volume.

The performance can be summarized in Figure 5.3 where the single core run-times for each part are visualized in logarithmic scale in relation to the dimensional reduction applied. Panel (a) shows the results for the 2D case while (b) for 3D. The increase of the k-means runtime is unexpected, as its complexity is of $O(N)$ as shown in Section 3.4 would mean that it is constant in terms of number of pixels and number of cluster centers (equal to number of superpixels. This is a hint to a bug in the k-means implementation which we could not yet find. For the practical use this is not a problem because if $sr < 100$ the recall rate does not change significantly for any of the datasets presented in this thesis and one could use a regular grid for approximation. When looking at the highest count of superpixels used, the BSD dataset with an image size of 0.15 MP and a maximum over-segmentation into $1,500$ SP the corresponding value of $sr = 100$. For the 3D VISCERALCT dataset a segmentation into $1cm^3$ is equal to reducing a 43 MP volume to $450,000$ SP, again resulting in about $sr = 100$. Therefore the expected worst case single core runtime in 2D is 2.1 s/MP and for 3D 3.5 s/MP.

Using a multi core machine lowers the computation time for the monogenic signal by 75% per core. The approximated results are shown in Figure 5.4 where the runtime is calculated for a 12 core machine calculating superpixels on 3D volumes. The total runtime is decreased and the computation time of the monogenic is similar to the label clean-up method.

5.3 Recall, Noise and Regularity

The performance of *MonoSLIC* in terms of recall, robustness to noise and the regularity are discussed in this section.

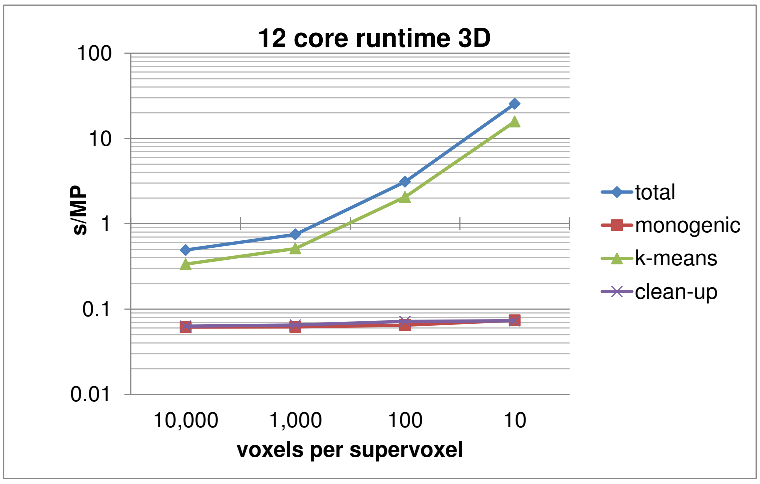
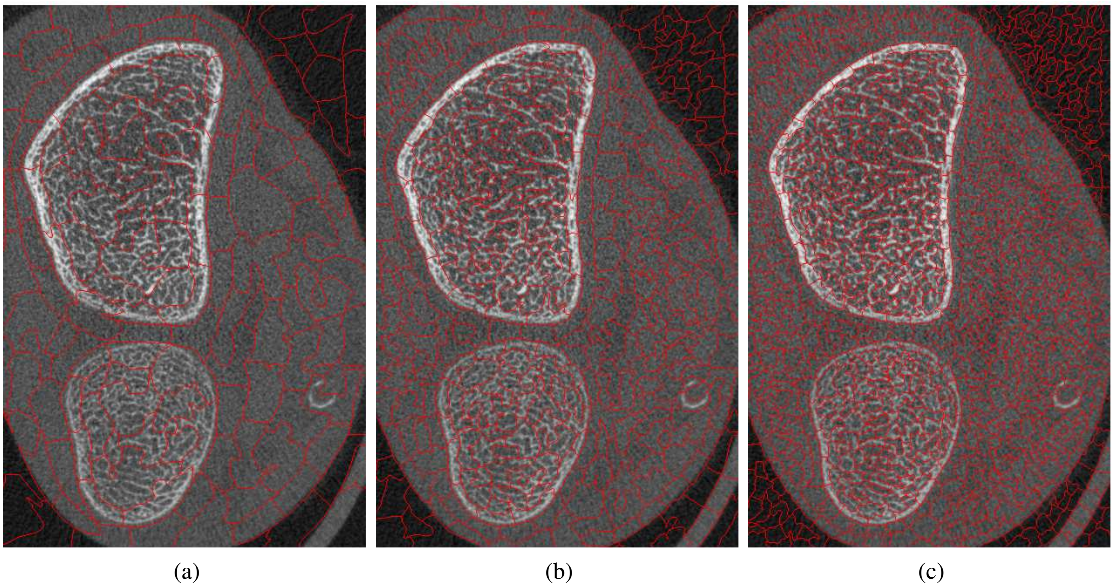


Figure 5.4: Multi core runtime in seconds per Mega Pixel in logarithmic scale relative to the number pixels per superpixel (increasing number of superpixel) for the three most time consuming parts of the monoSLIC algorithm on a volume with 16.78 MP.

Recall An important value for over-segmentation algorithms is the recall performance, because if a boundary is not captured by the algorithm this information is lost for all following programs that solely use the superpixel segmentation as input. As discussed in Section 5.1 at higher wavelengths (larger superpixels) the monogenic phase is not able to capture the local structures defining objects much smaller than the wavelength. This is the reason for the bad performance for example in Figure 4.10 (a) when the number of superpixels is lower than 150 pixels per superpixel. But as soon as the wavelength is smaller than the objects that should be detected the performance is equal or better than the state of the art methods. Because the wavelength corresponds to the average superpixel size, one can also say that the average superpixel size has to be smaller than the smallest object that should be detected. Another medical image example is shown in Figure 5.5, where the forearm is scanned using a high resolution peripheral quantitative computed tomography (HR-pQCT) acquired in [47]. In Panel (a) the general structure of the bone and forearm is detected, although the segmentation is not precisely at the corresponding boundary. With decreasing wavelength (b) more detail is captured and the small circular structure in the bottom right is segmented. The precision of the boundary further increases with smaller superpixel sizes can be seen in Figure 5.5(c).

A visual comparison in terms of recall rate is presented for the CT and MR VISCERAL dataset in Figures 5.6 and 5.7. The segmentation of the algorithm is shown in red, the annotated ground truth in green and blue corresponds to a matching segmentation of the algorithm to the ground truth. Each figure is divided by a white line, where the top shows the results of Achanta and the bottom of *MonoSLIC*. In Figure 5.6 the VISCERAL CT example is presented. If the object boundary is clearly visible (high contrast) it is segmented by Achanta as seen at for the lung and the silhouette of the body, while the opposite is true in the abdomen region where



(a) (b) (c)

Figure 5.5: Forearm HR-pQCT (source [47]) example with overlaid segmentation of the MonoSLIC algorithm for (a) 1850, (b) 700 and (c) 300 pixels per superpixel.

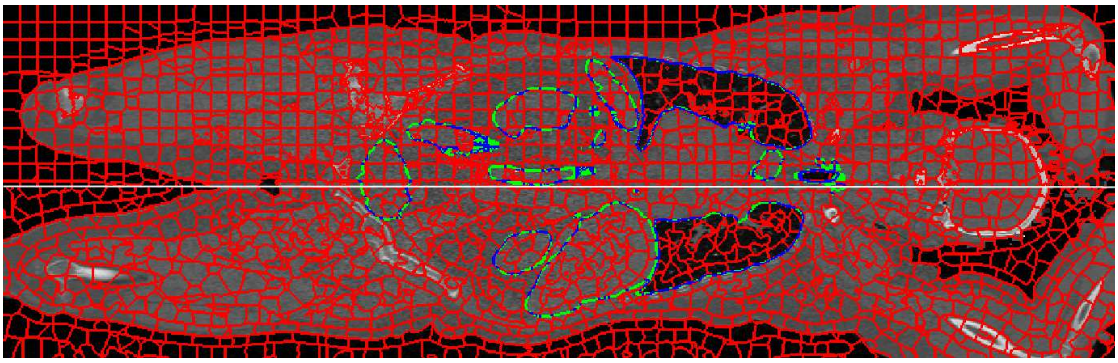


Figure 5.6: Comparison of the Achanta and MonoSLIC over-segmentation on an VISCERAL CT [27] example with a supervoxel size of $7ccm$.

Achanta is not able to capture the structure. *MonoSLIC* on the other hand reacts to all the structures, which is the reason for the better recall performance. Looking at the VISCERAL MR dataset one can see the same effect as shown in Figure 5.7. This is again the reason for the better recall rate of the *MonoSLIC* approach, although due to the bad resolution and the resulting thick annotations the recall rate difference is not as big as the visual difference.

Noise Another important aspect of the monogenic phase is the stability to noise, as shown in Figure 4.12. The methods of Achanta and *MonoSLIC* are now compared using different noise

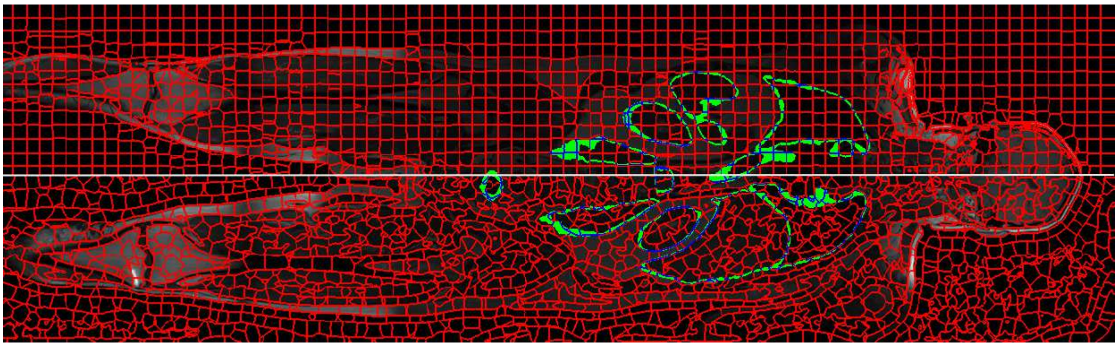
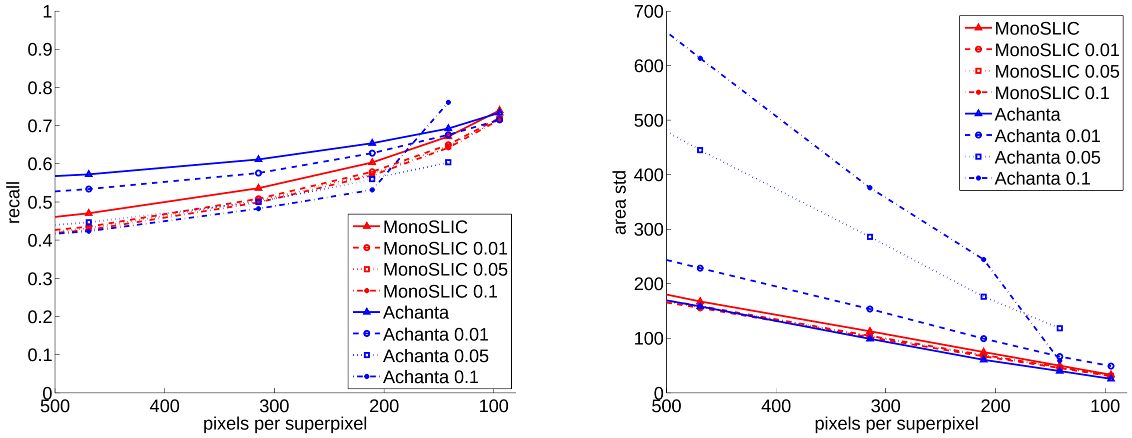


Figure 5.7: Comparison of the Achanta and MonoSLIC over-segmentation on an VISCERAL MR [27] example with a supervoxel size of $7ccm$.



(a) (b)

Figure 5.8: Recall curve (a) for Achanta and MonoSLIC on the original and noisy BSD images. In (b) the standard deviation of area is shown for the same image set.

variations (0.01, 0.05 and 0.1) on the images. In Figure 5.8 (a) it can be observed that the red curves of the monogenic only change when noise is introduced for the first time with a variation of 0.01, but barely changes for higher variations of 0.05 and 0.1. The recall of Achanta on the other hand gradually decreases with higher noise, even causing problems to the final post-processing step resulting in a limited number of superpixels. This is the reason why there is no data to plot for 100 pixels per superpixel and noise variation of 0.05 and 0.1 for Achanta. The high recall rate for a variation of 0.1 at 140 pixels per superpixel is an outlier. Only a few images were segmented with small superpixels and those seem to have a very high recall rate. In (b) the standard deviation of the superpixel area is shown, signalizing that for Achanta the variation in superpixel size is also increases with the added noise, while there is no change for *MonoSLIC*.

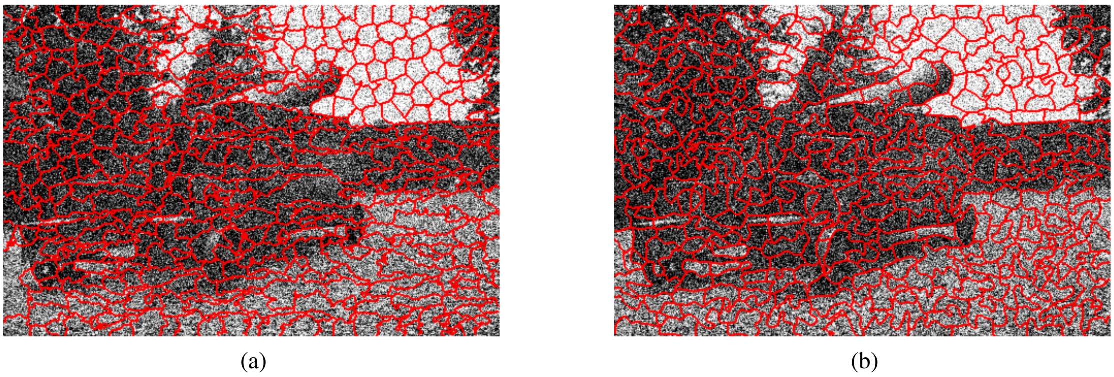

(a) (b)

Figure 5.9: The Cannon image with Gaussian noise variation of 0.1 segmented into about 400 pixels per superpixels using (a) Achanta and (b) MonoSLIC.

Regularity This can also be observed in the Cannon example segmentations for both algorithms in Figure 5.9 on the extreme case of noise variation 0.1 where the segmentation is widened by 1 pixel for increased visibility. In the center and the bottom left of Achanta (a) large superpixel can be observed. Using only pixel wise information it is difficult to segment either trees or the wheels of the cannon. Furthermore the over-fitting causes less regular superpixels where the boundaries become rougher and visually less pleasing by breaking the law of continuity and smoothness mentioned in Section 2.1. Changing the parameter of Achanta would create smoother boundaries at the cost of segmentation performance. The monogenic on the other hand, does not require any parameter to be set. It analyses the local neighborhood of the pixel and therefore is less influenced by noise. The trees, wheels and shadows are again segmented similar to the results of absent noise.

5.4 Future Work

Future work can include improving the k-means implementation, using a different segmentation technique or features. There are also other evaluation measurements that can be used.

Improving k-means Addressing the poor scaling of the k-means performance the algorithm and implementation could be improved, as theoretically the runtime should be independent of the number of cluster center used. It is also implemented as single core and could be sped up by redesigning it for using multiple cores.

Different Segmentation Method Investigating the potential of the monogenic phase in terms of recall a different but simpler segmentation method is used that does not create superpixels as defined in Section 2.2, but just an edge-detection like segmentation. It creates a boundary when there is a change of local phase from -0.5 to 0.5. The results on the BSD dataset are shown in Figure 5.10 where the MonogenicRaw information is added as red dots. This shows

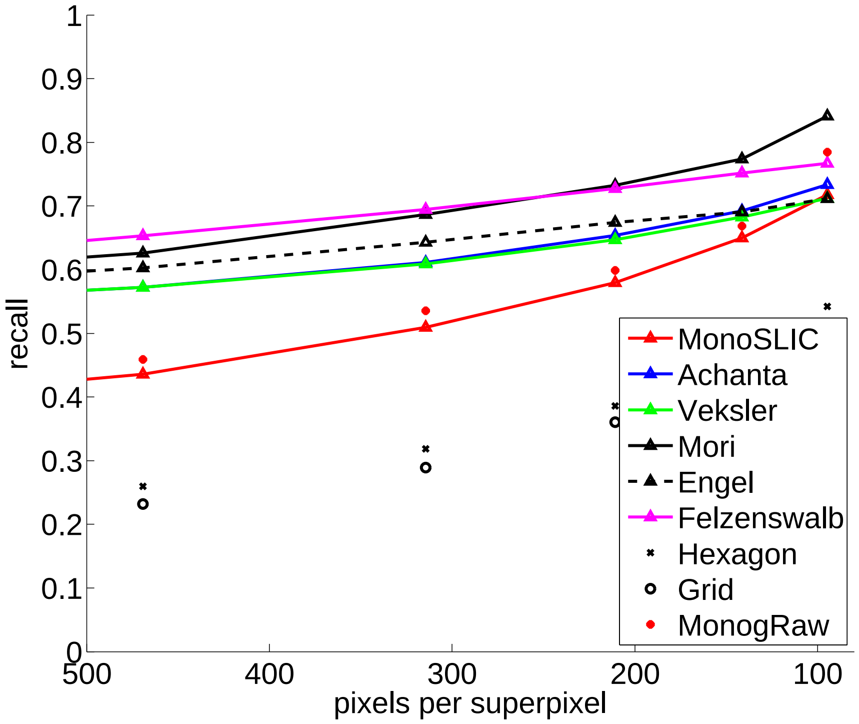


Figure 5.10: Recall rates on the BSD dataset including the MonogenicRaw information.

the potential of using a different method than k-means for creating the superpixel segmentation. An idea would be to find the minimum of the sign of the monogenic phase and move regularly placed seed-points to their location. In a final iteration each pixel is assigned to the nearest seed point with the same monogenic sign value.

Additional Features The recall rate could also be improved with minimal additional computational effort using a combination of features of the monogenic phase, the monogenic amplitude and the original gray-values. One could either change the current k-means input based on any combination of those features or add them as another dimension to the clustering space. But one has to keep in mind that adding features that are not brightness/contrast invariant would cause the loss of an important property of the presented method.

Another option would be to calculate the monogenic for different scales and combine the results to one final, precise feature. Especially for low number of superpixels (when the superpixel area starts matching the area of the object or structure) the recall performance has improvement potential. The downside would be the increased runtime due to calculation of the monogenic response for multiple scales.

Evaluation Measurement In terms of evaluation Schick et al. 2012 [45] suggest to use a measurement named under-segmentation error which was first presented by Levinshtein et al 2009 [28].

It describes how much superpixels within a segmented structure violate the annotation boundary, where less violation (called *bleeding out*) means more precise segmentation.

5.5 Summary

In this section the properties of the *MonoSLIC* method was discussed. It detects structures and edges that have a size that is at least as large as the wavelength of the filter, but performance drops when it gets smaller than the filter wavelength. This is reflected in the recall rates measured in the 2D and 3D datasets.

The feature extraction (monogenic), segmentation (k-means) and labeling cleanup implementations were examined in terms of runtime, showing that for 2D images the feature calculation takes 59% of the total runtime while taking 43% in 3D. It was also shown that the runtime of k-means increases when the superpixel size gets smaller, expecting worse case runtimes of $2.1s/MP$ for 2D and $3.5s/MP$ for 3D instead of the optimal $0.7s/MP$, which might be caused by a bug in the implementation.

The evaluation measures discussed include recall, noise and regularity. The recall rate is at least equal to state of the art methods when the average size of a superpixel is at least the size of the smallest structure that has to be detected. The visual examples for MR and CT volumes show how well *MonoSLIC* performs visually when there is low brightness and contrast. The robustness to noise is also shown, as the recall rate is less affected than any state of the art methods with reaching 98% of the original performance, independent of the pixels per superpixel. Also the regularity of *MonoSLIC* is not affected by the Gaussian noise, keeping a consistent (in terms of average superpixel size) and regular over-segmentation.

Future work can include the improvement of the k-means algorithm to remove the increase in runtime at smaller superpixel sizes. Also replacing the k-means segmentation method can improve the recall performance of the algorithm, as the edge information detected by the monogenic phase is not completely captured by it. Another way of improving the segmentation performance could be by using additional features that are extracted by the monogenic signal.

Finally another evaluation measurement is suggested that measures the under-segmentation error.

CHAPTER 6

Conclusion

The aim of this thesis is to analyze state of the art over-segmentation methods and create an efficient, fast and robust method that matches the analyzed methods in terms of performance and can be used for large 3D medical volumes.

The result from the analysis on the BSD dataset is that the algorithm of Mori extracts the most information of the image, including texture and gray-values, before running a costly segmentation method. It has the best recall performance with 84% for 100 pixels per superpixels, but the computation time of $564.8s/MP$ and the memory cost $O(N^2)$ is the highest of the compared methods. As a result larger images or even volumes would not compute. The fastest performing method Felzenswalb, with 0.8 s/MP gives no control over the number of desired superpixels and therefore also has no superpixel size or regularity constraints. It also has a parameter that needs to be tuned depending on the image type. The approach of Engel has the same properties as the former, but in the current implementation has a slower runtime of $5.1s/MP$. From Veksler the proposed *Constant Intensity* version was analyzed. It allows the choice of number of superpixels and has a mediate recall performance of 71% with a compared high runtime of $45.4s/MP$ for the tested implementation. The final tested method of Achanta has the third best recall rate after Mori and Felzenswalb with 73%. While it is faster than Mori with the second fastest runtime of 1.6 s/MP when compared to Felzenswalb it also allows control over the number of superpixels, although still a parameter for superpixel regularity has to be chosen before segmentation. From the analyzed methods only for Achanta code was available that could be compiled to perform 3D over-segmentation. The goal now was to create a method that is at least as fast and accurate as the approach of Achanta, without the necessity of an extra parameter to be set. In contrast to the pixel intensity based approach of Achanta the new method should also be resilient to noise as it could occur on OCT recordings and should work for either MR or CT recordings, meaning contrast and brightness independent.

The presented method *MonoSLIC* full-fills the previous demands by combining the contrast-brightness invariant and noise robust monogenic phase with the fast k-means clustering method for segmentation. The wavelength of the texture based monogenic method is directly linked to the number of superpixels, with the idea in mind that the smallest texture the algorithm can detect should match the average size of the superpixels. Therefore no extra parameter apart of the number of superpixels has to be set. The result is a fast $0.7s/MP$ and exact, over-segmentation method. Its memory consumption is limited to 6 times the number of voxels by the monogenic phase calculation, which means that a volume of size $260MP$ would need $12.6GB$ of RAM to compute. The current k-means implementation performs best when the number of superpixels S is less than 1% of the number of pixels P of the volume, otherwise the runtime increases to up to $12s/MP$ for $S = 10\% * P$. Important to note is that if the structure that should be detected is larger than the wavelength, the algorithm will not detect it. In numbers for the BSD dataset the segmentation performance drops to 43% for 500 pixels per superpixel compared to an average of 60% for the other algorithms. Beside these constraints the algorithm is able to over-segment a 260 MP volume into $60,000$ supervoxels within 180 seconds on 12 Core Xeon CPU. The recall rate for VISCERAL, the largest 3D annotation dataset available, is 82% for supervoxel of size $4cm^3$ compared to 71% for Achanta and 50% for a reference hexagon grid, with a speed of 0.7 s/MP compared to 2.1 s/MP of Achanta.

Future research can include a review on the monogenic phase as only input for the k-means algorithm, because combining it with additional information like monogenic amplitude or the original intensity values could improve the overall recall performance of the algorithm. Another point for improvement is the increased k-means runtime for a higher number of superpixels. As the monogenic phase already contains compact information for segmentation a different, faster method could be used to create the final segmentation and capture the full potential of the monogenic phase. An interesting evaluation value could be added to the existing ones, called under-segmentation error that reflects how much superpixel area of superpixels representing an annotated structure, is outside of the annotated structure, relative to the area of the annotation.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels. Technical Report 11, EPFL, November 2010.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE PAMI*, 34(11):2274–82, November 2012.

[3] Y. Adini, D. Sagi, and M. Tsodyks. Context-enabled learning in the human visual system. *Nature*, 415(6873):790–793, February 2002.

[4] B. Andres, U. Köthe, M. Helmstaedter, W. Denk, and F. Hamprecht. Segmentation of SBFSEM Volume Data of Neural Tissue by Hierarchical Classification. In *Proceedings of the 30th DAGM Symposium on Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, chapter 15, pages 142–152. Springer Berlin Heidelberg, 2008.

[5] S. Beucher. Watersheds of functions and picture segmentation. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, volume 7, pages 1928–1931, May 1982.

[6] S. Beucher. The watershed transformation applied to image segmentation. In *Conference on Signal and Image Processing in Microscopy and Microanalysis*, pages 299–314, 1991.

[7] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*, September 1979.

[8] S. Bhat, I. V. Larina, K. V. Larin, M. E. Dickinson, and M. Liebling. Multiple-Cardiac-Cycle Noise Reduction in Dynamic Optical Coherence Tomography of the Embryonic Heart and Vasculature. *Optics Letters*, 34(23):3704–3706, 2009.

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

[10] J. Canny. A Computational Approach to Edge Detection. *IEEE PAMI*, 8(6):679–698, November 1986.

[11] L. Cohen. Time-frequency distributions-a review. *Proceedings of the IEEE*, 77(7):941–981, 1989.

[12] T. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[13] F. Ding, W. K. Leow, and S. Venkatesh. Removal of Sbdominal Wall for 3D Visualization and Segmentation of Organs in CT Volume. In *16th ICIP*, pages 3377–3380. IEEE, November 2009.

[14] J.S. Duncan and N. Ayache. Medical Image Analysis: Progress over Two Decades and the Challenges Ahead. *IEEE PAMI*, 22(1):85–106, January 2000.

[15] D. Engel and C. Curio. Scale-Invariant Medial Features based on Gradient Vector Flow Fields. In *19th ICPR*, pages 1–4. IEEE, December 2008.

[16] P. Felkel, M. Bruckschwaiger, and R. Wegenkittl. Implementation and complexity of the watershed-from-markers algorithm computed as a minimal cost forest. *CoRR*, cs.DS/0206009, 2002.

[17] M. Felsberg. Low-level image processing with the structure multivector. Technical Report 0202, Kiel, Germany, March 2002.

[18] M. Felsberg and G. Sommer. A new extension of linear signal processing for estimating local properties and detecting features. In *22. DAGM Symposium Mustererkennung*, pages 195–202. Springer-Verlag, 2000.

[19] M. Felsberg and G. Sommer. The Monogenic Signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, 2001.

[20] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[21] B. Fulkerson, A. Vedaldi, and S. Soatto. Class Segmentation and Object Localization with Superpixel Neighborhoods. In *Proceedings of ICCV*, pages 670–677, 2009.

[22] W. S. Geisler. Visual Perception and the Statistical Properties of Natural Scenes. *Annual Review of Psychology*, 59(1):167–92, January 2008.

[23] S. Haas, R. Donner, A. Burner, M. Holzer, and G. Langs. Superpixel-Based Interest Points for Effective Bags of Visual Words Medical Image Retrieval. In *Proceedings of the 2nd MICCAI International Conference on Medical Content-Based Retrieval for Clinical Decision Support*, MCBR-CDS'11, pages 58–68, Berlin, Heidelberg, 2012. Springer-Verlag.

[24] D. Hoiem, A. A. Efros, and M. Hebert. Closing the Loop in Scene Interpretation. In *CVPR*, pages 1–8. IEEE, June 2008.

[25] D. Huang, E. A. Swanson, J. S. Lin, W. G. Stinson, W. Chang, M. R. Hee, T. Flotte, K. Gregory, and C. A. Puliafito. Optical coherence tomography. *Science*, 254(5035):1178–81, 1991.

[26] R. Kikinis and S. Pieper. 3D Slicer as a Tool for Interactive Brain Tumor Segmentation. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2011, pages 6982–6986, January 2011.

[27] G. Langs, H. Müller, B. H. Menze, and A. Hanbury. VISCERAL: Towards Large Data in Medical Imaging - Challenges and Directions. In *Medical Content-Based Retrieval for Clinical Decision Support*, pages 92–98, Nice, France, 2013.

[28] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. TurboPixels: fast superpixels using geometric flows. *PAMI*, 31(12):2290–2297, 2009.

[29] Z. Li, X. Wu, and S. Chang. Segmentation using Superpixels: A Bipartite Graph Partitioning Approach. In *PAMI*, pages 789–796. IEEE, 2012.

[30] H. Liu, Y. Qu, Y. Wu, and H. Wang. Class-Specified Segmentation with Multi-Scale Superpixels. In Jong-Il Park and Junmo Kim, editors, *Computer Vision - ACCV 2012 Workshops*, volume 7728 of *Lecture Notes in Computer Science*, pages 158–169. Springer Berlin Heidelberg, 2013.

[31] E.P. Lyvers and O.R. Mitchell. Precision Edge Contrast and Orientation Estimation. *IEEE PAMI*, 10(6):927–937, 1988.

[32] J. Malik. Normalized Cuts and Image Segmentation. *IEEE PAMI*, 22(8):888–905, 2000.

[33] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.

[34] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proceedings of the 8th ICCV*, volume 2, pages 416–423, 2001.

[35] F. Meyer. Topographic Distance and Watershed Lines. *Signal Processing*, 38(1):113–125, July 1994.

[36] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel Lattices. In *CVPR*, pages 1–8. IEEE, June 2008.

[37] G. Mori. Guiding model search using segmentation. In *10th ICCV*, volume 2, pages 1417–1423. IEEE, 2005.

[38] M. Nilchian and M. Unser. Differential phase-contrast x-ray computed tomography: From model discretization to image reconstruction. In *9th IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 90–93, May 2012.

[39] S. M. Pizer, G. Siddiqi, K.and Székely, J. N. Damon, and S. W. Zucker. Multiscale medial loci and their properties. *Int. J. Comput. Vision*, 55(2-3):155–179, November 2003.

[40] J. M. S. Prewitt. Object Enhancement and Extraction. *Picture Processing and Psychopictorics*, pages 75–149, 1970.

[41] T. Randen and J.H. Husoy. Filtering for Texture Classification: A Comparative Study. *IEEE Transactions on PAMI*, 21(4):291–310, April 1999.

[42] X. Ren and J. Malik. Learning a Classification Model for Segmentation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 1, pages 10–17. IEEE, 2003.

[43] A. Rosenfeld. The Max Roberts Operator is a Hueckel-Type Edge Detector. *PAMI*, 3(1):101–103, January 1981.

[44] F. Rousseau, E. Oubel, J. Pontabry, M. Schweitzer, C. Studholme, M. Koob, and J. Dietemann. BTK: An Open-Source Toolkit for Fetal Brain MR Image Processing. *Computer Methods and Programs in Biomedicine*, 109(1):65–73, 2013.

[45] A. Schick, M. Fischer, and R. Stiefelhagen. Measuring and evaluating the compactness of superpixels. In *21st International Conference on Pattern Recognition (ICPR)*, pages 930–934, 2012.

[46] E. Schwartz, J. Holfeld, M. Czerny, and G. Langs. Visualizing Changes in Vessel Wall Dynamics due to Stent-Grafting in the Aortic Arch. In *9th ISBI*, pages 788–791. IEEE, May 2012.

[47] A. Valentinitsch, J. M. Patsch, A. J. Burghardt, T. M. Link, S. Majumdar, L. Fischer, C. Schueller-Weidekamm, H. Resch, F. Kainberger, and G. Langs. Computational identification and quantification of trabecular microarchitecture classes by 3-d texture analysis-based clustering. *Bone*, 54(1):133–140, 2012.

[48] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and Supervoxels in an Energy Optimization Framework. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, pages 211–224, 2010.

[49] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations. *IEEE Transactions on PAMI*, 13(6):583–598, June 1991.

[50] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[51] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt II. *Psychologische Forschung*, 4(1):301–350, 1923.

[52] G.A. Wright. Magnetic resonance imaging. *Signal Processing Magazine, IEEE*, 14(1):56–66, Jan 1997.

[53] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *Trans. Img. Proc.*, 7(3):359–369, March 1998.

[54] L. Yi-Wen. Hilbert transform and applications. In *Fourier Transform Applications*, pages 291–300. InTech, 2012.

[55] S. Yu. Computational models of perceptual organization. Technical Report CMU-RI-TR-03-14, Pittsburgh, PA, May 2003.

[56] D. Ziou and S. Tabbone. Edge Detection Techniques - An Overview. *International Journal of Pattern Recogintion and Image Analysis*, 8:537–559, 1998.