Dissertation

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the Ruperto-Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
M.Sc. Nikola Enrico Krasowski
born in Bernkastel-Kues
Oral examination: 13.07.2016

Automated Segmentation for Connectomics Utilizing Higher-Order Biological Priors

Referees:	Prof. Dr. Fred A. Hamprecht
	Prof. Dr. Ekaterina A. Kostina

Zusammenfassung

Automatische Segmentierung für die Konnektomik unter Berücksichtigung komplexen biologischen Vorwissens

Diese Dissertation stellt neue Methoden für die automatische Segmentierung von Neuronen auf der Basis elektronenmikroskopischer Aufnahmen vor, die sich auf Techniken aus dem Bereich des maschinellen Lernens stützen. Es wird aufgezeigt, wie komplexe biologische Zusammenhänge berücksichtigt werden können.

Zum einen wird der V-Multicut eingeführt, der es ermöglicht topologische Bedingungen an die Detektion der neuronalen Membranen zu stellen. So können biologisch unplausible Membranverläufe korrigiert werden. Des Weiteren wird gezeigt wie, zusätzlich zur lokalen Membrandetektion und topologischen Kriterien, die Berücksichtigung komplexer biologischer Zusammenhänge von Vorteil ist. Für die Umsetzung dieser Aufgabe erweisen sich sowohl der Asymmetric Multiway Cut als auch der hier vorgestellte Semmantic Agglomerative Clustering Algorithmus als geeignet. Konkret wird die in Säugetieren beobachtbare örtliche Trennung von Axonen und Dendriten ausgenutzt, um eine signifikante Verbesserung der Segmentierungsqualität zu erwirken.

Weiterhin werden neue Ansätze präsentiert, welche die Skalierbarkeit der verwendeten Algorithmen verbessern. Zusammengefasst leistet diese Arbeit einen Beitrag zum Forschungsfeld Connectomics indem sie die automatische Segmentierung von Neuronen vorantreibt.

Abstract

Automated Segmentation for Connectomics Utilizing Higher-Order Biological Priors

This thesis presents novel methodological approaches for the automated segmentation of neurons from electron microscopic image volumes using machine learning techniques. New potentials for neural segmentation are revealed by incorporating (high-level) biological prior knowledge. This goes beyond the modeling of neural tissue which has been applied for the purpose of its segmentation, so far.

Firstly, the V-Multicut algorithm is introduced which enables the consideration of topological constraints for segmented membranes. In this way, biologically implausible appearances of membranes are corrected. Secondly, this thesis proves that, in addition to local evidence and topological requirements for the detection of neural membranes, the consideration of high-level biological prior knowledge is beneficial. For this task, both the recently proposed Asymmetric Multiway Cut and the introduced Semantic Agglomerative Clustering algorithm are implemented and quantitatively evaluated. To be precise, the spatial separation of dendrites and axons in mammals is exploited to significantly improve the segmentation quality.

Additionally, new ways to improve the scalability of the used algorithms are presented. All in all this thesis serves as another step towards fully automated segmentation of neurons and contributes to the field of connectomics.

Acknowledgments

I would like to thank Prof. Dr. Fred Hamprecht for his supervision of my research and this thesis. I greatly enjoyed the Image Analysis and Learning Group's friendly, supporting and sophisticated environment he has created.

My thanks also go to Prof. Dr. Ekaterina Kostina for taking the effort of being my second adviser and to PD Dr. Ullrich Köthe, who always takes the time for a discussion about any topic of image analysis. Dr. Thorben Kröger was helping me to acquire the necessary tools the trade needed in our research field. He was always open to discuss new ideas on how to improve the segmentation of neurons. As was Dr. Anna Kreshuk, who convinced me that the understanding of the biology of neurons can open new possibilities for their segmentation. For most of my time at the Heidelberg Collaboratory for Image Processing (HCI) I shared my office with Thorsten Beier. His dedicated spirit and his motivation were bolstering everyone in his vicinity. Although Dr. Bernhard Kausler left our group not long after I joined, I think that his professional spirit is still affecting the way I am approaching problems. I would like to thank Dr. Ferran Diego and Dr. Melih Kandemir, Carsten Haubold, Steffen Wolf, Nasim Rahaman, Philip Schmidt, Sven Peter, Elke Kirschbaum, Dr. Martin Schiegg, Dr. Christoph Straehle and Christoph Decker for their support and their collegiality. Being able to discuss matters of neuron segmentation with Constantin Pape, Timo Prange and Philip Schill was not only fun but also motivating. In addition I would like to thank Barbara Werner, Evelyn Wilhelm, Dominic Spangenberger and Dr. Nils Krah for supporting me in administrative matters. I thank Stuart Berg and all other developers for their tirelessly work on the ilastik software on which I relied on in basically all of my experiments. For the opportunity to work on their data, I would like to thank Graham Knott and his team.

I am grateful for the financial support I received from the German Research Foundation (DFG) within the program of the Research Training Group on Spatio/Temporal Graphical Models and Applications in Image Analysis (grant GRK 1653). I am happy to got the opportunity to attend interesting courses and interdisciplinary discussions within Research Training Group led by Prof. Dr. Christoph Schnörr. Being able to regularly exchange myself with Freddie Aström, Johannes Berger, Ecaterina Bodnariuc, Robert Dalitz, Tobias Dencker, Mattia Desana, Stefan Richter, Francesco Silvestri, Vera Trajkovska, Nikolai Ufer, Artjom Zern, Tabea Zuber enriched my doctoral studies.

Finally I thank my family Rosa, Roman, Meike, Lukas and Sabrina for their patience support and love.

Contents

Chapter 1

Introduction

1.1 Connectomics and its Computational Challenges

What underlies intelligent behavior, consciousness or memorization? These are questions that belonged in the realm of philosophy for a long time. Recent technological advances e.g. in imaging techniques lead to the hope that natural sciences could also have a word in the discussion. The complexity of the questions demands for an interdisciplinary approach. Among others, co-operation between the fields of neurobiology, chemistry, physics, cognitive science, engineering and computer science is pivotal. This thesis aims at contributing to the automation of data analysis for this field, specifically to the fully automated segmentation procedure, a necessity given the amounts of image data, showing neural cells, that requires processing.

Cognitive phenomena seem to origin from a variety of different mechanisms. The existence of a Society of Plant Neurobiology [1] and the findings in the research area of swarm intelligence as e.g. described by Garnier et al. [2] are hinting at a wide range of processes that underly the basic object of study: cognition. To make them more manageable, the following considerations will be restricted to intellectual abilities arising from distinct nervous cells as one finds in all major animal groups comprised in the clade of eumetazoans. The understanding of the function of such a nervous system is still a sophisticated task. Only recent developments in imaging techniques and advances in computational and algorithmic abilities have been fueling hopes that the goal of revealing some of the secrets behind cognition can be accomplished.

Nervous systems are networks of neurons. They are able to transmit and process information that are encoded in chemical and electrical signals. These signals can be passed from one cell to another at special structures, the synapses.

From this brief sketch of the basic building blocks of nervous systems one can already see that both the architecture of the neurons and the temporal pattern emerging in the network must be understood in order to explain the systems way of functioning. In other words on needs to know which neuron is connected via synapses to which other neuron and how certain signals propagate to the given network.

This is already where the first difficulties arise. Nowadays there are two major trends in the imaging community relating to the problem at hand. The technique of Diffusion MRI allows to capture the dynamics of a system due to its non-invasive nature. The current maximal achievable spatial resolution is around $1 \cdot 10^{-3}$m [3]. This resolution is not sufficient to capture the individual neurons. The diameter of human axon fibers can be as huge as $2.5 \cdot 10^{-5}$m but typically they will be thinner than 10^{-6}m [3]. Processes as small as $50 \cdot 10^{-9}$m can be encountered in the used image volumes (e.g. in the data described in appendix A.1.1). The spatial resolution of electron microscopes (EM) is good enough to capture even these thin processes. The big disadvantage here is that the sample of interest can either be scanned on its surface only via reflected electrons, or given an adequately thin probe imaged via transmitted electrons. This requirement on its own prevents an in vivo imaging. Therefore, one is in principle either able to see dynamics on a rough scale measuring superpositions of countless signals or to get a spatially well resoluted snapshot in time.

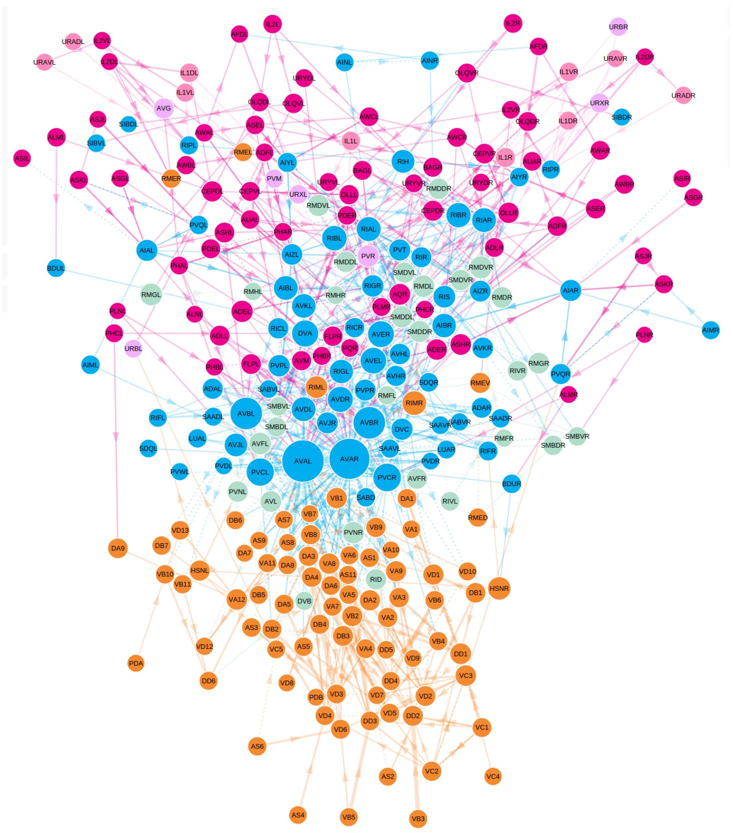


Figure 1.1: A connectome of C. elegans. Neurons (data described in [4]) are represented by nodes and their connections are depicted via directed edges between them. Since the neuronal structure of this animal is consistent over individuals the neurons can actually be named. The coloring is encoding the function of the respective neurons. The figure was created with the tool provided in [5]

Having access to the spatial structure, parts of the functional interactions may emerge by means of network analysis. The branch of research that has set itself the task of reconstructing the whole circuit diagram (connectome) that represents the individual neurons and their connections is called **connectomics**. [6] reviews the impact that the knowledge about the spatial structure of nervous systems has. According to the author, opinions on the topic vary between the assessment that a connectome is completely useless and the estimate that a connectome is the one most useful ingredient in understanding the brain. He summarizes the averaged opinion of the field like this: “A connectome is necessary, but not sufficient"[6].Tasks arising in the vicinity of connectomics can be assigned to one of three consecutive steps: i) acquisition of images showing all parts of interest, ii) construction of a connectome based on these images, iii) network analysis on the connectome.

This work aims at contributing to step ii) by providing novel methods for the segmentation of neurons from EM image volumes. A precise segmentation allows the reliable reconstruction of the numerous connections mediated by the strongly branching neurons. In particular we will see that so far proposed segmentation algorithms lack the ability to include higher-order biological priors in their segmentation decision. This is information that human experts would use when performing the task manually. The contribution of this thesis is the formalization of multiple of these priors and their inclusion into optimization problems that are solvable on reasonable problem sizes. We observe that due to

species	#neurons		#synapses	
C. elegans	302	[7]	7 500	[7]
common fruit fly	1.35×10^5	[9]	1×10^7	[10]
house mouse	7.089×10^7	[11]	5.5×10^{11}	[12]
human	8.6×10^{10}	[13]	1.5×10^{14}	[14]

Table 1.1: Comparison of estimated mean number of neurons and synapses in the central nervous system that need to be recovered in order to construct a complete connectome of the respective species.

the included biological priors the segmentation quality improves significantly. This can be seen as one step toward a fully automated segmentation procedure and therefore as one step towards the long term goal of a possible human connectome.

That the steps involved in the construction of a connection between synapses are highly demanding can be seen by a looking into the history of the field. The first comprehensive connectivity map of an organism was constructed 1986 by White et al. [7] for the nematode Caenorhabditis elegans. It took more than 20 years until the next complete connectome of C. elegans was put together by Varshney et al. [8]. In fact this model organism with its 302 neurons is the only organism whose complete connectome was successfully constructed up to this point in time.

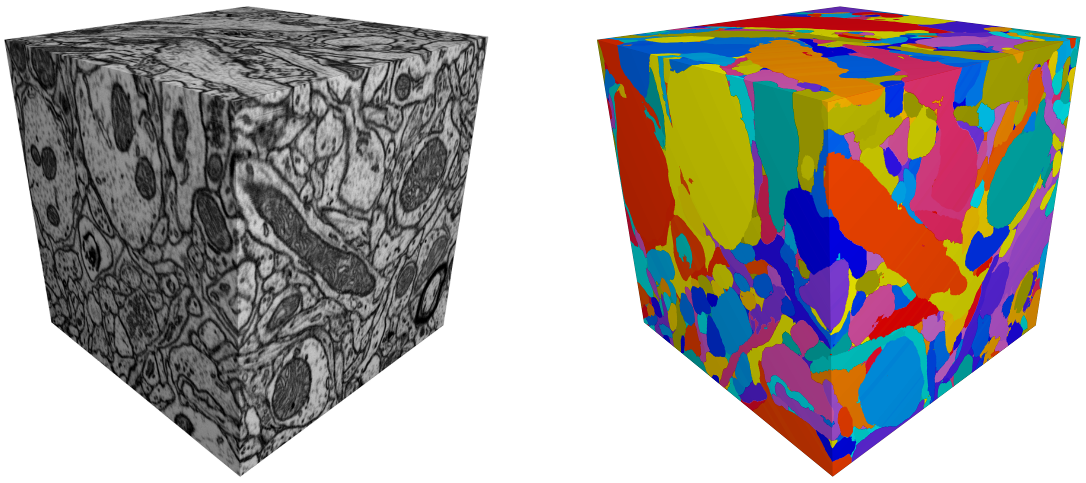
The reason why we still lack a connectome of a more complex organism becomes apparent when having a closer look at actual number of connections that need to be reconstructed. For C. elegans, even though the number of neurons is relatively low, the number of connections mediated via synapses amounts to roughly 7 500. This hints already towards the complexity that we nowadays face when aiming for connectomes of more complex model organism like the common fruit fly (Drosophila melanogaster) or the house mouse (Mus musculus). From Table 1.1, where the estimated numbers of neurons and synapses are presented for different species, one can get an idea of the increasing complexity of the task and how huge the project of creating a human connectome is.

All efforts to reconstruct a connectome rely on volumetric image data covering a sufficiently large section of the neuronal network in the required quality. As I will elucidate in section 1.2.1, only electron microscopy is able to fulfill the requirements for the spatial resolution and only advanced microtomes, automated slicing devices, allow the imaging of larger sections in reasonable time. The amount of data that these imaging facilities produce is immense. Lichtman et al. [15] estimate that the complete human cortex will take a zetabyte of storage space. Seeing this number makes it clear that automated methods will eventually not only be judged by their quality of segmentation but also by their scalability.

The following section will further explain the different angles from which automated image analysis could approach the problem. The ensuing section 1.3 will have its focus more on the specific approaches that have been proposed so far. Subsequently section 1.4 will give an overview on the contribution of this thesis to the field of automated neuron segmentation.

1.1.1 The Need for Automation

John White and his colleagues started in 1969 with the project of constructing the complete set of synaptic connections in the nervous system of C. elegans. As described in [11]



(a) EM image volume (b) respective segmentation

Figure 1.2: An exemplary isotropic EM image volume in (a), recorded by Knott et. al. [16], described in detail in the appendix A.1 and a respective color-coded segmentation (b) of the neurons.

it took them 15 years to complete this task. The reconstruction was done manually by tracing the neurons on printouts of the data with a thin pen. Nowadays, computers can support the manual tracing of neurons. Still the time for a manual reconstruction of a human connectome, as estimated by Plaza et al. [17], is around 1.4×10^{10} man-years. This illustrates how valuable a reliable algorithmic solution to the tracing problem is.

Besides the determination of the course of the neurons, the tracing human experts as well as any automated solution is required to detect synapses, the structures that form the connection between th neurons. In addition the determination of the respective synaptic partners and the excitatory or inhibitory nature of the synapses are essential [18]. It has been shown that the detection of synapses can reliably be done automatically by Kreshuk et al. [19], Becker et al. [20] and Fua and Knott [21]. Also for the problem of synaptic partner detection algoritmic solutions are made available by Kreshuk et al. [22].

The challenge that I will address in this work is the actual establishment of the connections between the synapses via the neurons. In Figure 1.1 the connections of two neurons via synapses are indicated via the directed edges in the graph. The task of tracing the neurons is then complementary with the task of identifying the synaptic partners in the direct adjacency of the synapses with each other.

Let us distinguish two different directions that an automation can take:

A1 An algorithm could trace the course of neurons and thereby construct a corresponding skeleton. This is the way humans approach the problem.

A2 Alternatively, an algorithm could detect all cell membranes. This corresponds to a complete segmentation of the individual neurons as is shown in Figure 1.2.

The course of a skeleton (as produced e.g. by human tracers) is not at all unique. The decision on the course of the skeleton depends not on local indicators in the data but on indirect information from the surrounding membranes. While for human annotators the consideration of membrane configuration in a huge field of view is an easy task, image analysis algorithms do utilize their strengths better when detecting the membranes directly.

Therefore, within this work we I concentrate on the second approach (A2). The detection of Membranes can be based on the local appearance of the respective parts of the data. Different convolution filters help in this binary classification problem. Rather than handcrafting the decision rules based on these filters, they are learned.

1.1.2 Machine Learning in Image Analysis

Machine Learning, also refereed to as pattern recognition is one subfield of artificial intelligence. According to Bishop [23] "[t]he field of pattern recognition is concerned with the automatic discovery of regularities to take actions such as classifying the data into different categories". A distinction between unsupervised and supervised learning is often made. In unsupervised learning only the data itself is available and patterns/clusterings must be detected solely on the distribution of the data-points in some feature space. In supervised learning in contrast, one additionally provides the desired output of the learning algorithm, the ground truth, on a special training set. The goal then is to learn rules from the training data that are general enough that one can apply them to unseen data. This methodology is useful in a huge range of fields, image analysis being one of them. Handcrafting rules for image analysis tasks is especially desperate since capturing all relevant configurations on a pixel grid of possibly multiple channels must be ensured. An exception are low level image analysis tasks like edge detectors that could be constructed learning free.

In this work I will make use of different classifiers like the random forest to compute certain local probabilities and I will utilize models expressible as probabilistic graphical models to find a global consensus of all local indications (e.g. in chapter 3) and to include biological prior information that helps in the segmentation task (as e.g. done in chapter 6).

In the field of segmentation for connectomics the vast majority of the recently presented algorithms (described in section 1.3) utilize aspects from machine learning to some extent.

1.2 Background

Being able to think about the problem of automatic segmentation for connectomics in the first place is only made possible due to advances in imaging techniques in recent years. To get an impression of the achievements of the imaging community an overview on commonly used methods will be given. Subsequently I will describe what one can actually see on the taken images. Understanding some aspects of the biology of the neurons will turn out to be crucial in the improvement of the current automated segmentation approaches.

1.2.1 Image Acquisition

All efforts on automated segmentation, as well as all other efforts in the direction of connectomics, rely on the fact that images covering the relevant parts of the neural tissues are available. This can not be taken for granted. As we will see in the following, the technical requirements on the resolution of the recordings are high and therefore the production of high quality image volumes is a delicate endeavor.

Since thin neural processes of a diameter of 50 nm do exist, the raw data, on which the connectome is based on, must necessarily have a resolution better than 50 nm. In addition, the complete connectome requires the imaging of the complete nervous system. Therefore the imaging techniques must be scalable. In the following, several imaging principles are presented and their applicability for the given task is discussed. We will see that only a few are able to match the minimal requirements.

Photon Microscopy

One choice of visualization principle is based on the measurements of photons. The concrete setup of the imaging device depends on the wavelength of the respective radiation.

Light Microscopy In the domain of (visible) light microscopy there are several techniques to beak the diffraction barrier associated with Ernst Abbe (1873). Without these approaches the resolution would not suffice:
For a light microscope with a typical a numerical aperture (NA) of 1.5 the maximally possible resolution d would be $d = \frac{\lambda}{\mathrm{NA}}$. With blue light of a wavelength λ of 400 nm this results in $d \approx 250$ nm.

So far, the mentioned surpassing of the diffraction limit is always restricted to special experimental conditions. As shown by Ausserré and Valignat [24] a technique called Sarfus for example enables the detection of particles with a diameter of 10 nm. This is only possible though if the particles are isolated. If the probe can be made photo-switchable florescent other approaches like [25, 26, 27] are feasible. They rely on a sequential activation of the fluorescence dyes. From the received signals they can recover super-resoluted images. This approach is able to image the sparsely occurring synapses quite well [28] but does still seem to fail in the task of imaging neural membranes reliably and in a sufficient resolution.

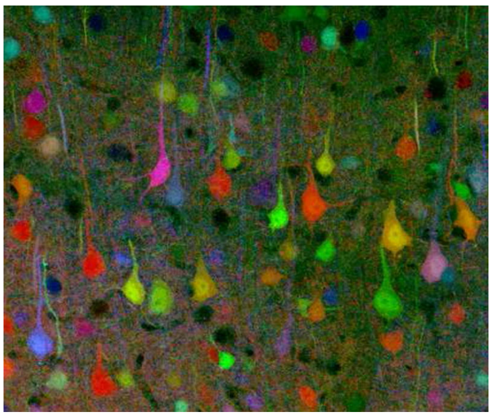


Figure 1.3: Examplary Brainbow image from [29]. Each neuron is colored individually.

An approach introduced by Livet et al. [30], providing optically pleasing images of neurons via light microscopy, is called Brainbow. Fluorescent proteins of different colors are expressed in the individual neurons in random fractions and result in an individual coloring of the cells as it can be seen in Figure 1.3. The coloring clearly gives a huge advantage in the segmentation progress. The cell bodies and the thick branches of the neurons can be easily distinguished but the fine processes can not be resolved. The value of this data for connectomics on the neuron level is therefore limited. This kind of imaging reflects nicely one of the major difficulties in segmenting neurons. In all recordings with sufficient resolution they are mostly locally indistinguishable from each other. For a long time the membrane evidence was the only indication used for segmenting individual neurons. In chapter 6 I will show how to partly evade the indistinguishably.

X-Ray Microscopy One way to increase the resolution is to increase the energy of the used photons. With common X-ray microscopes one can achieve resolution of several 10 nm [31]. In [32] the imaging of neurons with a 15 nm resolution is achieved. The imaging is made possible by a staining via the The Golgi-Cox method put forward by Ramon-Moliner [33]. This methods relies on the fact that one can apply a staining to only a small subset of all neurons. In the X-ray images one can observe the difference between the stained and the non stained neurons. While this method can be utilized to analyze the shape and topology of single neurons it is not suited for connectomics. Distinguishing only two kinds of neurons (stained and not stained) can not in any ratio of stained vs not stained be used for a dense segmentation into all individual neurons.

Electron Microscopy

With wavelengths of orders of magnitudes less than visible light, accelerated electrons are able to resolve finer structures. In order to adjust the naturally low electron optical contrast of biological tissue a staining, often times including heavy metals, is necessary. This staining can make a distinction of cell membranes, cytoplasm and some organelles possible but as it can be seen in Figure 1.4 it does not distinguish individual neurons. If an electron beam hits a stained tissue, a fraction of the electrons are cast backwards while some do penetrate the tissue[1]. This causes the division of the methodology of electron microscopy in transmission electron microscopy (TEM) and scanning electron microscopy (SEM). The "scanning" is due to the point-wise probing of the tissue on an regular grid. A similar scanning procedure as in SEM, where the backscattered electrons are captured and therefore the object may be massive, can be applied for TEM leading to the technique of scanning transmission electron Microscopy (STEM).

No matter which kind of microscope is chosen, one will end up with a two dimensional image of the probe. Obviously the reconstruction of a connectome can hardly be done by imaging the brains surface only. Also a single slice through the neural tissue is not sufficient. Thus, a consecutive slicing of the tissue is necessary. Therefore the volumetric resolution of the combined images is not only determined by the resolution in 2D of the microscope but also depends on the thickness of the slicing of the probe.

TEM/SSTEM The resolution of variants of the TEM can reach up to 0.05 nm [34]. This allows the imaging of single atoms. Lichtman et al. [15] consider the fact that the high lateral resolution is beneficial for identifying fine structural details in neural tissues but the fact that the slice of tissue has to be cut of in a way that preserves the structure restricts the resolution in cutting direction. In practice the cutting is done via a thin diamond blade. Both Silvestri et al. [35] and Wanner et al. [36] report a minimal slice thickness of $\sim$ 40 nm. If an automated mictorome is used to cut the slices which are then automatically transported to the imaging device, the term serial section transmission electron microscopy (ssTEM) has established itself [37].

SEM SEM in contrast scans the surface of a tissue. To be able to get a three dimensional impression of the complete tissue, this surface needs to be removed gradually. The big advantage over transmission based techniques is that there is no need for the removal

[1]There are multiple effects that can be responsible for electrons that are cast backwards. Besides a simple backscattering so called secondary electrons can escape the surface as a result of ionizing radiation of any kind that is caused by the primary electrons. The Auger effect is another possible source of returning electrons. The penetrating electrons, leaving the the tissue in the original beam direction, can both be scattered or transmitted unperturbed.

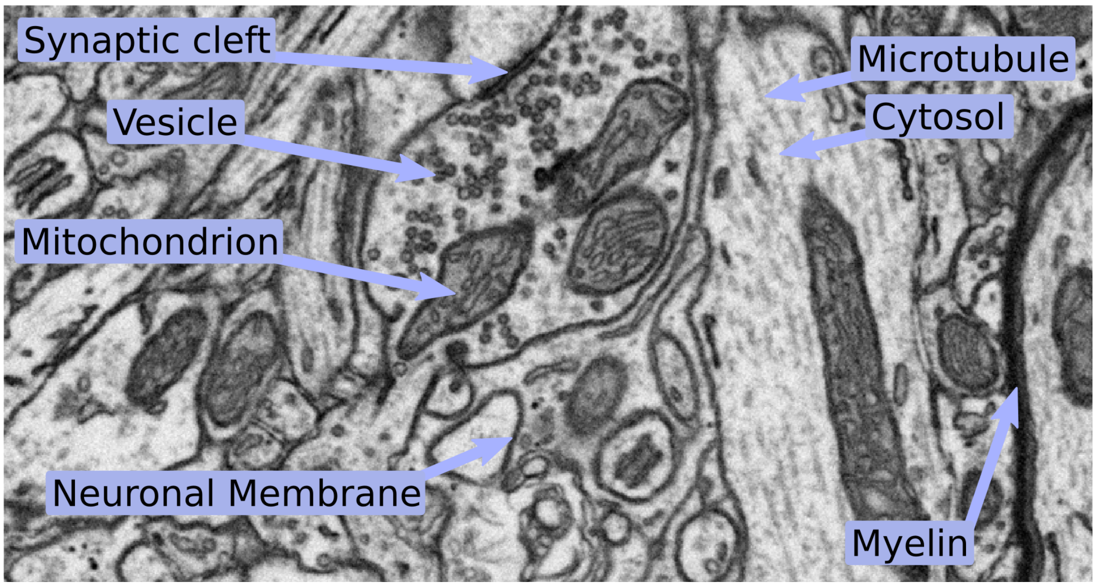


Figure 1.4: A section of an electron microscopic image of a somatosensory cortex of an adult mouse (data described in appendix A.1). All organelles that are of relevance within this thesis are visible.

procedure to preserve the erased layer. Besides the already mentioned diamond knives, focused ion beams can be utilized (FIB-SEM). Respective setups could generate volumetric images of an isotropic resolution of 5 nm as shown by Knott et al. [16]. Approaches utilizing SEM will generally suffer less from alignment problems between the individual images than TEM based methods. Goldstein et al. [38] state that resolutions around 1 nm can routinely be achieved via commercial instruments.

A big advantage of nondestructive methods is that the imaging process can be distributed to multiple microscopes and therefore the overall imaging time can be reduced [36]. This speed advantage is also owed to improved microtomes, tools able to automatically cut thin slices from a tissue as developed by Denk and Horstmann [39] and Hayworth et al. [40].

In principle one wants the resolution to be just as good such that automated reconstruction is possible. Increasing resolution over this point will unnecessarily lead to increased imaging and processing time in addition to a need for bigger storage space. Although the best trade-of resolution has to be determined yet, human tracing performance leads to the assumption that the complete exhausting of the maximal possible resolution will probably not be necessary. The concrete type of electron microscope that was used to produce the data, used within this thesis, and the respective resolution will be specified for the individual experiments separately.

1.2.2 Biology of Neurons

Figure 1.4 shows a small section of neural tissue imaged by an electron microscope. This image contains all of the cells' components that will play a role within this thesis. There are several prominent organelles appearing in most eukaryotic cells like Mitochondria, involved in the cells energy supply or the Microtubules that are part of the cytoskeleton and do facilitate the mechanical stability of the cells. The herefore mentioned internal substructures are embedded in the liquid parts of the cytosol which holds all kinds of dissolved materials and proteins. The cells cytoplasm, the cytosol and the organelles is confined by a selectively permeable membrane.

There are also characteristics only occurring in neural cells. At some places the neural membrane is additionally reinforced by tubes of myelin. This fatty substance reduces the electrical capacitivity of the neurons and therefore enables a fast signal transmission via saltatory conduction. The color of the myelin gives the white matter in the brain, where myelinated cells are frequent, its name. In contrary regions with low percentage of myelizations are called gray matter. The crucial places where signals can leap from one cell to another are called synapses. At their presynaptic side (in the “sending” neuron) oftentimes vesicles, holding neurotransmitters ready to be released into the synaptic cleft, are detectable.

Cells in general and neurons in particular are extremely complex structures consisting of many more building blocks. We restrict ourself in this brief outline to the ones that are visible in the used recordings from electron microscopes and relevant in the process of automated reconstruction of the connectome. Readers with deeper biological interests can refer to the textbook of Alberts et al. on cell biology [41] and the textbook of Levitan and Kaczmarek [42] with the focus on neural cells.

1.3 Related Work

Reconstruction of neural circuits from image data attained by electron microscopy is a necessary task in the process of constructing a connectome. Consequently the scientific interest in this area is huge. A big variety of approaches have been proposed to solve this problem. While some rely on human interaction, others try to fully automate the process. To be able to pigeonhole the contributions of this thesis I will give an overview over respective prominent and influential approaches of the different paradigms.

1.3.1 Manual Reconstruction

Lichtman et al. [15] state that humans are capable of producing segmentations that are good enough to reconstruct neural connections with a satisfying accuracy. The authors report more than 99.9% consensus in connectivity in cases where multiple tracers process the same data independently.

In the following I want to give an overview over some of the most popular approaches in this direction:

- In [43, 44] a software package named Knossos is introduced. Knossos allows the visualization of huge data volumes as well as the tracing of skeletons by a humans.
- Raveler [45] is another annotation tool developed by the Fly EM project team at the Janelia Research Campus. It allows its users the creation and modification of segmentations.
- A similar functionality offers the tool Mojo [46] and its web based relative Dojo [46].
- The Carving algorithm [47], made available within the ilastik software [48], approaches the problem from a slightly different angle by doing a seeded segmentation where the user provides the seeds in an interactive way.

- The Catmaid software introduced in [49] is able to trace neurons through huge image volumes cooperatively and in addition proivides the means for semantic segmentation.

For Plaza et al. [17] a combination of Raveler and Carving is the most efficient way to manually construct a dense segmentation of neurons. According to the authors a complete reconstruction of a human brain in this way would take around 14×10^9 man-years of manual annotations. Still overwhelmingly 4.5×10^6 years are estimated for a mouse brain and according to them the fruit-fly brain would take 4.7×10^3 years.

These numbers make it very clear that automation of the segmentation process is not only nice to have but an absolute necessity. This does by no means render all manual annotation tools superfluous. On the one hand even a reconstruction of some small subpart of a nervous system could reveal some of its functional principles. On the other hand supervised learning algorithms rely on ground truth. Manual annotation tools can be an important instrument in the creation of labeled data of a certain quality and quantity.

1.3.2 Semi Automated Methods

Methods that try to partly unburden human annotators are summarized in this section. Some of them could be described as proofreading tools. These are programs that can be used to manually correct an existing segmentation. If a tool is able to construct a segmentation from the scratch it will also be able to modify an existing one. Therefore all tools presented in section 1.3.1 creating segmentations are able to proofread an automatically produced segmentation [45, 46] and can therefore be listed here if used in such a context.

Chklovskii et al. [50] introduce a complete pipeline including 2D segmentation, linking in the third dimension and an subsequent proofreading step. Such an inbuilt proofreading possibility is also offered by the workflow presented by Jurrus et al. [51]. There, results of an automated membrane detection and linking of neurons over anisotropic slices can be corrected. In the more recent approach of Jones et al. [52] a merging hierarchy is automatically determined and users are asked to correct presented regions in order to improve segmentation quality. The idea to apply level set segmentation in a semi automatically way was introduced by Joeng et al. [53]. The presented tool NeuroTrace links two dimensional contours of neurons via a tracking approach in the third dimension.

For non-dense neuron data, (e.g. single prepared neurons via Neurobiotin or fluorescent dye from optical microscopy), an approach aiding the tracing was presented in [54]. The application to images from electron microscopy is questionable though. Already their visualization of the skeletons does rely on the sparse nature of the data.

The more parts of a complete segmentation pipleline can be automatized without loosing the necessary segmentation quality, the closer the community comes to the goal of a complete connectome of an animal more complex than the nematode Caenorhabditis elegans. Therefore, having a fully automated approach that needs human interaction at most in a training phase is most desirable.

1.3.3 Automated Segmentation

The work presented in this thesis falls into the category of automated segmentation. Therefore a closer look into the related fully automated segmentation approaches is appropriate.

Pixel Level

Many of the proposed fully automated segmentation algorithms start from pixel-wise membrane detection. On its own this turned out to be not sufficient at the current resolution and data quality (see section 2.1.4 for more details). Nevertheless a good detection accuracy on the pixel level is beneficial for all subsequent higher reasoning.

In recent years artificial neural networks got viral in the Computer Vision community. Their often times reported high performance became possible only lately due to advances in graphics cards, which they can exploit efficiently. In the filed of neuron segmentation already Jain et al. [55] report an increase in quality of membrane detection over approaches utilizing common Markov Random Field or Anisotropic diffusion. The applicability of this class of classifiers to membrane segmentation in two dimensions is also proven by Ciresan et al. [56]. The potential the increased memory of modern graphic cards is also impressively demonstrated by Huang and Jain [57]. The authors show how far one can get by increasing the size of the network and thereby the space of parameters that are available for fitting an meaningful mapping.

One interesting idea is pursued by Jurrus et al. [58] where multiple neural networks are stacked. Seyedhosseini et al. [59] continue their work in this spirit. They also combine multiple artificial neural networks in an hierarchical way whereby the images are down-sampled from stage to stage to learn more higher-order relations. Ronneberger et al. [60] provide the newest popular contribution that could be seen in relation to [58] and [59]. Here the down-sampling is included directly in the network. The attained higher-order information is up-sampled again with the help of lower level but better resoluted filter responses from earlier layers.

A commonality of all approaches including neural networks is the need for large amounts of training data which, in the case of neuron segmentation, is hard to come by. Since random forests can deal with much less training data as can be seen in [48] they are the weapon of choice in this thesis for the problem of the determination of initial pixel-wise membrane probability. The idea of stacking the classifiers applied in [58] can naturally also be applied for random forests where it improves results as well.

Superpixel Level

As I will elucidate further in section 2.4, going from pixels to the superpixel level is advantageous not only in terms of runtime but also in terms of quality. The specifics of the applied pipelines strongly depend on the kind of available image data. As mentioned in section 1.2.1, transmission electron microscopy depends on preserved thin slices of tissue. This restricts the resolution in slicing direction to ~ 40 nm. Since resolution in the imaging plane is higher this leads to anisotropic data. Oftentimes a stepwise approach is deployed – first doing a 2D segmentation followed by a linking in 3D.

For isotropic data as e.g. produced by a FIB-SEM approaches working intrinsically in 3D are more suitable.

Anisotropic Data A typical approach of 2D segmentation with subsequent linking in 3D is presented by Yang and Choe [61]. The graph cut algorithm (see chapter 6) is utilized for the segmentation part.

The idea of co-clustering is brought to neuron segmentation by Vitaladevuni and Basri [62]. The two dimensional segmentation of consecutive slices are coupled. The assumption is that the spatial proximity of the slices results in a similar appearance.

Weak membrane evidence within one slice can thereby be overcome by evidence from neighboring slices.

Uncertainty in the 2D segmentation due to insufficient image quality is tried to be overcome by Vazquez-Reina et al. [63] and Funke et al. in [64] by the consideration of multiple segmentation hypothesis. Only if a 2D hypothesis leads to a plausible 3D continuation it will finally be chosen.

The basic idea of allowing multiple hypotheses is also underlying the work of Kaynig et al. [65]. For initial 2D segmentation they aim to close holes in membranes via an anisotropic smoothing cast in a conditional random field. They combine their hypothesis into consistent 3D segmentations via segmentation fusion.

The trend to utilize as much information from neighboring slices as possible is also apparent in the work of Laptev et al. [66]. Their method of choice for doing this is the SIFT Flow algorithm. It basically matches densely sampled, pixel-wise SIFT features.

Looking closely to what causes an improvement in the segmentation quality over the years one finds that utilizing 3D information is definitely important. Therefore it might be assumed that a fully automated segmentation algorithm delivering sufficient quality will first be found for isotropic data where the 3D information is much more valuable and an intrinsic 3D representation is possible.

Isotropic Data For data with isotropic resolution, the advantage of reliable information from neighboring slices is given and most segmentation algorithms operate directly in 3D. The stepwise segmentation approach, first proposed by Andres et al. [67], is currently used by many other 3D segmentation pipelines. The initial supervoxel oversegmentation introduces structure and drastically reduces the problem size, thus allowing more advanced algorithms to be used. Jain et al. [68] use a reinforcement learning approach to merge the supervoxels. Similarly, Nunez-Iglesias et al. [69] use reinforcement learning within a learning-based iterative hierarchical clustering algorithm. While very fast, this greedy approach can make incorrigible wrong merge decisions.

Parag et al. [70] delay the merging decisions by several steps, until some of their consequences are visible and the merging can be reverted if necessary. Liu et al. [71] proposes to greedily solve the segmentation problem within the framework of a merge-tree, and recently the merge-tree approach has been further improved to the globally optimal level by Uzunbas et al. [72]. This method is still limited to the segmentations contained in the merge-tree.

Andres et al. [73] introduce an overall globally optimal segmentation (for a given supervoxel oversegmentation and supervoxel boundary evidence) that is found by the introduction of topological constraints on the supervoxel adjacency graph. This work builds in part on this set of constraints and on the Multicut algorithm [74] in general. But, unlike [73] and all other methods listed above, I am not basing the merge/split decisions purely on the local boundary evidence.

1.4 Contributions

Having an overview of the field of connectomics and an idea of where in this endeavor the contributions of this thesis have to be located, I want to give an overview over the main novelties presented in this thesis that aim to take forward automatic segmentation approaches for neurons from electron microscopic image volumes. While semi automated methods may still be superior in terms of segmentation quality, the vast amount of data requires full automation eventually (see chapter 1.3 for an overview of semi automated and fully automated methods).

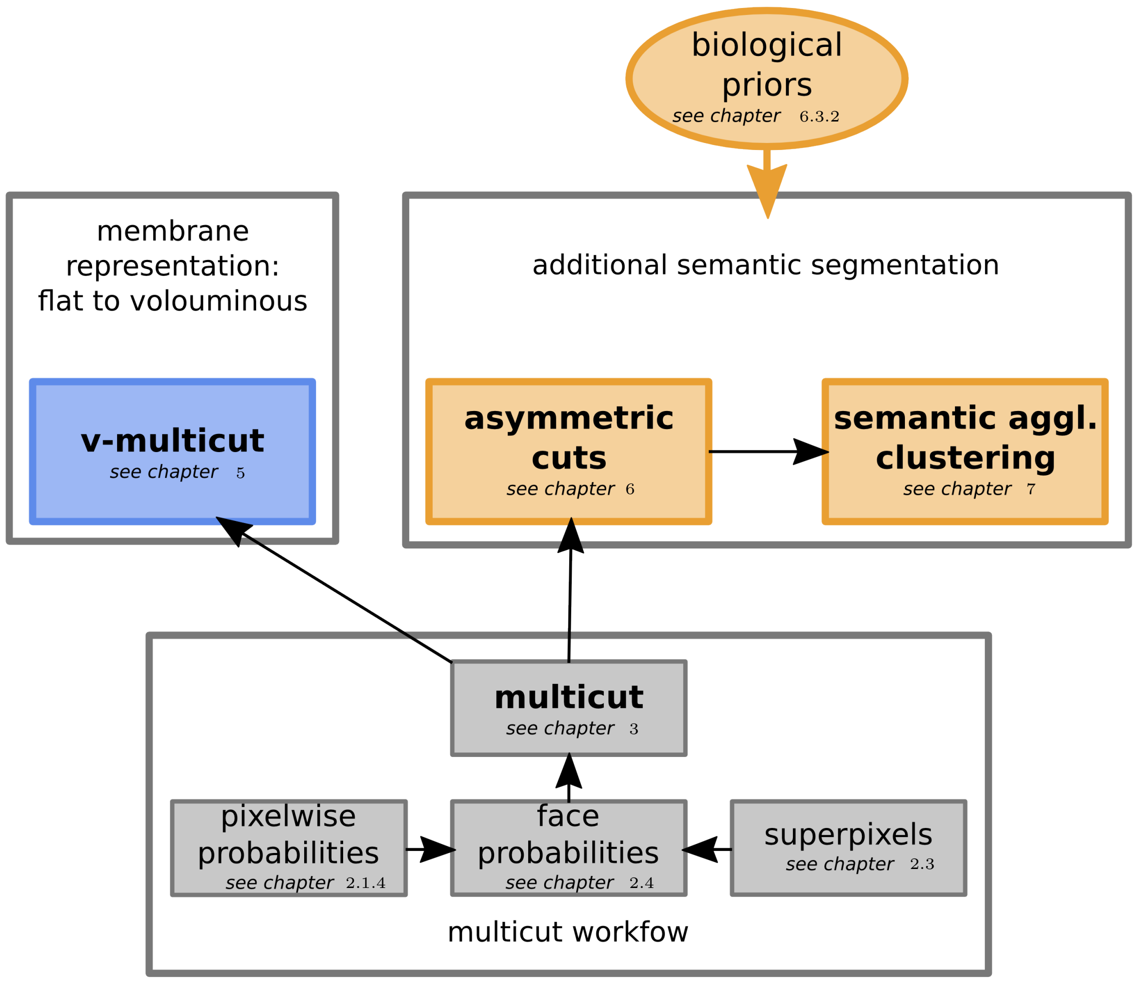


Figure 1.5: A sketch of the main contributions of this thesis in relation to an established workflow deploying the Multicut on pairwise merge probabilities (face probabilities) computed on the basis of superpixels and pixel-wise membrane probabilities. The V-Multicut is an extension of the principles behind the Multicut transfered to a volouminous representation. Both with the usage of the Asymmetric Cuts and the Semantic Agglomerative Clustering I show how higer order biological priors can be considered during the segmentation. The experiments including the Asymmetric Multiway Cuts allow for a direct comparison with the Multicut. A significant improvement can be observed when including the additional biological prior information.

Figure 1.5 is meant to give an overview of the three major contributions of this thesis. In a way they can all be related to the previous state-of-the-art in fully automated neuron segmentation, which is given by a pipeline including the Multicut algorithm[2]. The algorithm will be explained in chapter 3 such that all implemented improvements

[2]This statement is based on the SNEMI3D challenge [75] organized by the Massachusetts Institute of Technology where the currently best ranking approach is a version of the Multicut as proposed by the Image Analysis and Learning (IAL) group, Heidelberg University.

and extensions can be understood. One improvement directly related to the Multicut is presented in section 3.3. It is shown how the optimization problem related to the Multicut can be decomposed into several small problems without loosing global optimality in practice. In the performed experiments this results in a significant speedup.

The well known Agglomerative Clustering procedure is presented in chapter 4. It is extended by a novel way of determining the right moment to stop the iterative clustering even if the weights, used for clustering, give no indication on how to stop. We will see in chapter 7 that only the availability of such a stopping criterion enables the utilization of semantic labels within the Agglomerative Clustering procedure.

In chapter 5 we will see how the idea of enforcing consistent (no open ends in membranes) solutions in the explicit flat representation, as it is utilized in the Multicut case, can be translated to a voluminous representation. While the finding of inconsistencies on lower dimensional objects is straight forward it is nontrivial to detect inconsistencies in the voluminous representation unambiguously. This problem is solved within this chapter by relating to the membranes' skeleton where topological constraints are more easily found.

Chapter 6 opposes the long held assumption, underlying all methods mentioned in section 1.3.3, that the cell membranes are the only usable indications for the task of segmenting neurons from EM images. It is shown how an extension of the Multicut, the Asymmetric Multiway Cut, can be utilized to incorporate information about general neuron structures. In particular the fact that pre- and post-synaptic parts in individual mammalians neurons are spatially separated is exploited.

The contribution of chapter 7 is to show how the well known Agglomerative Clustering algorithm can be extended to include biological prior knowledge as well. We will see that the greedy nature of the Semantic Agglomerative Clustering comes with an advantage in runtime and a disadvantage in segmentation quality in comparison to the globally optimal Asymmetric Multiway Cut. Finally a novel approach to handle Mitochondria-segments in a neuron segmentation result is presented in chapter 8.

It is not proven yet that automated segmentation algorithms can in principle achieve human segmentation quality. Increasing the quality automated methods to reduce the gap to human performance is the main focus of this thesis. Nevertheless one needs to keep the enormous amounts of data in mind that wait for a reliable automatic segmentation pipeline. Therefore I present experiments on the decomposition of the Multicut and the V-Multicut that show that a globally optimal solution can often times achieved by a optimization of several disjunct subproblems. The decomposition is naturally emerging from the structure of the constraints.

Chapter 2

Segmentation Essentials

As explained in chapter 1.1 one promising way to get the connections formed by the neurons is attaining a complete segmentation of the neurons. This thesis will follow this path exclusively. To base further considerations on solid ground, I will give an introduction to the foundations of segmentation algorithms in image analysis in the following.

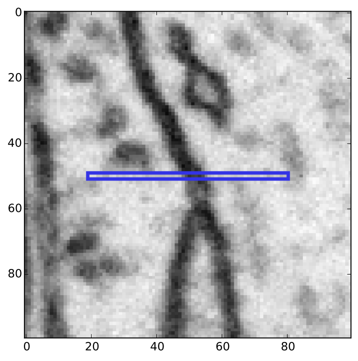
All image segmentation algorithms must utilize information about attraction/repulsions between pixels of one sort or another in order to produce meaningful results. For a big variety of problem instances there is a clear notion of edges. In this context edges correspond to pattern in an image that indicate a change of segment. There are other cases where getting a clear concept of localized edges is hard but a clear repulsion between non adjacent regions can be established for example due to differences in color or texture. The following sections will give a more detailed explanation for the different sources of attraction/repulsion between regions in an image and show the peculiarities of neuron segmentation.

2.1 Edges

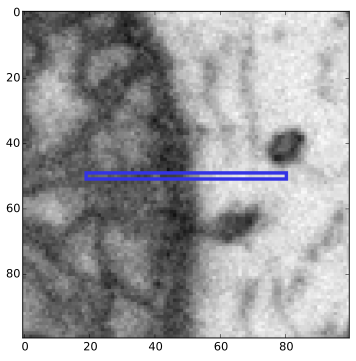
2.1.1 Ridges and Steps

In the image analysis literature one can find the distinction between **ridge edges** and **step edges** [76]. Occasionally one can also find the notion of ramp edges and roof edges. These are mere smoothed versions of the ridge edge and the step edge. In Figure 2.1 the idealized edge-types (green) as well as some respective exemplary one-dimensional extracts from actual EM data (blue) are compared.

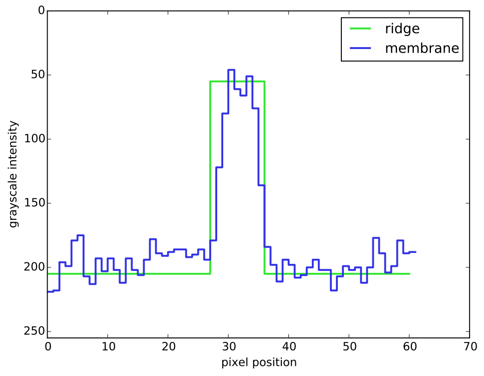
As it can be seen in Figure 1.4 individual neurons are separated by cell membranes that mostly appear as black lines (they are basically extended over two dimensional surfaces but in a sliced profile they appear as lines). In Figure 2.1 these membranes can be seen as ridge edges. One can see that step edges are also appearing in the data. Mitochondria, which are organelles of the cells (see chapter 1.2.2), should not be separate segments in a final neuron segmentation. We see that differentiating these edges already at this early stage is of importance for the results. Only cell membranes appearing as ridge edges should induce a repulsion. Even at this stage one can see that the mitochondria are a potential source of errors in the segmentation: On the one hand they have their own membrane. The transition in Figure 2.1 therefore is more like a superposition of a pure step- and a pure ridge edge. This can result in a confusion of mitochondria membranes and cell membranes. On the other hand the mitochondria can be positioned inside the neuron



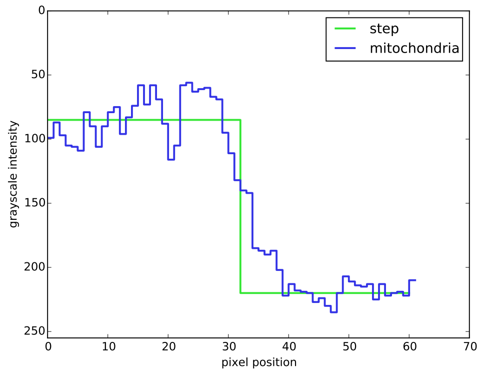
(a) cell membrane



(b) mitochondrion verge



(c) cell membrane - 1D profile



(d) mitochondrion - 1D profile

Figure 2.1: Two examples of different edges in EM images as described in appendix A.1 in the upper row. The lower row gives the respective one dimensional distribution of gray values within the blue frames in the upper images. The green curves are showpieces of the respective idealized edge classes. (The vertical axis is inverted for visual convenience)

in direct proximity of the membranes and thereby veil the characteristic appearance of the membranes, exacerbating their detection. One typical example is shown in Figure 2.2.

From this example one can get the impression that detecting the membranes purely based on basic filter operations like first or second derivative filters (as introduced in 2.1.3) will not work reliably. Applying machine learning algorithms that are able to consider a big field of view around the questionable membrane as well as having the possibility to do a higher-order reasoning based on the seen data turns out to be necessary. The mitochondria, for which I will introduce a possible solution in chapter 8, are only one of the difficulties of this kind of data that is tackled in this thesis. As can be seen in Figure 1.4, there are many other organelles and imaging artifacts that make the reliable detection of membranes hard.

2.1.2 Region Information

Imagine a chain of elements each only insignificantly different than their neighbors. Therefore the edge evidence between all elements is weak. Nevertheless both ends of the chain can look completely different. We see that there is more in image segmentation than edge information: Region Information. The segmentation of neurons is rather the exception

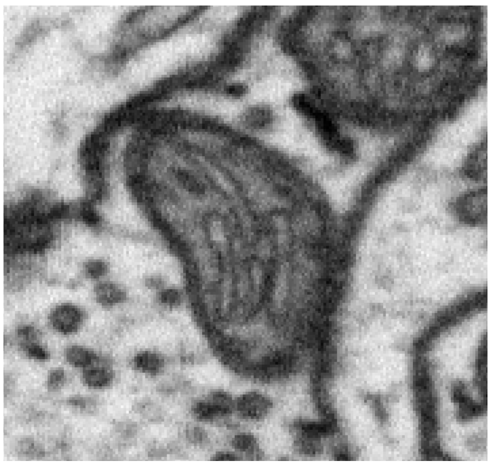


Figure 2.2: Mitochondria touching cell membranes. These are difficult cases for membrane detection because the ridge nature of the cell membranes is veiled by the step nature of the mitochondria.

than the rule in image analysis problems in the sense that the desired segments locally look mostly indistinguishable. While e.g. blue sky and green grass in natural images can easily be distinguished by color and texture even if the transition is out of focus and the precise localization of the transition is hard, not even experts can tell for most pairs of patches of neural tissue if they belong to the same cell or not.

A special kind of sparsely occurring region-wise information will be discussed in chapter 6. There, edges are neither blured due to effects related to the depth of field of the recording device nor extended gradients (edge that are defined on a bigger scale). The scenario is rather that the location of the segment transition based only on region-wise information is impossible to deduce, but the existence of a transition is evident. Two players of the same sports team that overlap on an image due to partial occlusion can be hard to separate since e.g. their shirts are alike. If both of their numbers are visible though it can be deduced that there must be a segment transition somewhere between the two detected numbers[1]. We will see in chapter 6 that the principle behind this admittedly hypothetical example can actually be applied to neuron segmentation. I will show how knowledge about biological functionality can complement edge indicators (local membrane evidence) in the segmentation process.

2.1.3 From Filter Responses to Learned Edge Indicators

What kind of information would one use to determine if one pixel is probably belonging to a cell membrane or not? First of all the gray-scale value originating directly from the electron microscopic imaging process (1.2.1). Since a thresholding of the data leads not to a sufficient segmentation quality (see section 2.1.4 for an related experiment), more information is needed. The only other source of information available is the gray-scale values of the other pixels, thereby the neighborhood of the pixel in question has to be considered. Low level image analysis convolution filters have proven to be an solid starting point [77] for a meaningful extraction of the pixels context. The discrete version of a convolution of an three dimensional image volume I with a discrete kernel k can be

[1]Note that in this scenario the two players could probably be separated when utilizing prior information about the shape of the persons that ought to be segmented. Since the plausible shapes of neurons are very diverse this approach is not easy applied there. But since humans are partly able to detect wrongly segmented neurons only by their shape this task (that is not explored further in this thesis) is not completely past any hope.

written as:

$$I'_{i_1 i_2 i_3} = \sum_{i'_1} \sum_{i'_2} \sum_{i'_3} k_{i'_1 i'_2 i'_3} I_{(i_1 - i'_1)(i_2 - i'_2)(i_3 - i'_3)} =: k * I \tag{2.1}$$

where i_a, i'_a are integers indexing a 3D image grid here. In praxis, the coefficients of the kernel for bigger distances $|i_a - i'_a|$ are vanishing. Therefore only a localized neighborhood of the pixel in question is given influence.

In the field of **scale-space theory** an image (in general any signal) will be represented on different scales by a family of smoothed continuous images $L(\mathbf{x}, t)$ with $\mathbf{x} \in \mathbb{R}^d$ as the continuous generalization of the coordinates and $t \in \mathbb{R}$ indicating the scale related to the smoothing. Lindeberg [78] describes L as the solution of the diffusion equation

$$\partial_t L = \frac{1}{2} \nabla^2 L \tag{2.2}$$

$$\text{with } L(\mathbf{x}, t = 0) = I(\mathbf{x}) \tag{2.3}$$

with the bed of nails like version of the the given d dimensional image

$$I(\mathbf{x}) = \sum_{i_1, \ldots, i_d} \delta\left(\mathbf{x} - (i_1, \ldots, i_d)^T\right) \cdot I_{i_1, \ldots, i_d} \tag{2.4}$$

that one can probe continuously as initial state. L needs to be constructed in a way that new structures are not hallucinated in when going from a low scale parameter t to a higher one. The solution of Eq. (2.2) is given by

$$L(\mathbf{x}, t) = \frac{1}{\sqrt{2\pi t}} \int \exp\left(-\frac{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}{2t}\right) I(\mathbf{y})\, d\mathbf{y}\,, \tag{2.5}$$

a multidimensional Gaussian kernel. For a (big enough) given scale parameter t one does now have a meaningful continuous representation of the image. On this base a series expansion on the basis of the derivatives of L can be created (jet representation [79]). In the scope of this thesis I will utilize the first two spatial derivatives of L as rotational invariant features characterizing the individual pixels. The *gradient magnitude*

$$|\nabla L(\mathbf{x}, t)| = \left.\sqrt{\sum_{i=1}^{d} \left(\partial_{x'_i} L(\mathbf{x}', t)\right)^2}\,\right|_{\mathbf{x}' = \mathbf{x}} \tag{2.6}$$

for the scale t gives an estimate for the slope of the steepest increase at coordinate $\mathbf{x}$. Therefore it is a detector for step edges of a scale of the order t. As a “blob detector”, a detector of spherical symmetrical objects, the *Laplacian operator* applied on L is suitable.

$$\nabla^2 L(\mathbf{x}, t) = \left.\sum_{i=1}^{d} \partial_{x'_i} \partial_{x'_i} L(\mathbf{x}', t)\right|_{\mathbf{x}' = \mathbf{x}} \tag{2.7}$$

The eigenvalues of the *Hessian Matrix* $H \in \mathbb{R}^{d \times d}$

$$H_{ij}(\mathbf{x}, t) = \left.\partial_{x'_i} \partial_{x'_j} L(\mathbf{x}', t)\right|_{\mathbf{x}' = \mathbf{x}} \tag{2.8}$$

describe the local curvature at the scale of t in the direction of the principal axis. Higher order derivatives have been found to be more vulnerable to noise. In addition the extraction of scalars as features is not trivial for higher dimensional tensors. Eq. (2.6) does characterize the slope on one scale t. Information about the coherence of the gradients on scale t on a specific bigger scale s is also be of interest. It is covered by the *structure tensor* S.

$$\left(S_w\left(\mathbf{x};t,s\right)\right)_{ij}=\int w\left(\mathbf{y};s\right)\left(\partial_{x_i}L\left(\mathbf{x}-\mathbf{y},t\right)\right)\left(\partial_{x_j}L\left(\mathbf{x}-\mathbf{y},t\right)\right)d\mathbf{y} \tag{2.9}$$

Lindeberg [78] reasons why not only the smoothing within L, but also the smoothing over the orientations done by the window function w should be of Gaussian nature. Taking the eigenvalues of S again provides the desired property of rotational invariance of the features. Note that rotational invariance is not generally desirable. Think about natural images where blue sky tends to be more present in the upper parts of the image while green grass will probably more often appear on the lower parts. This does not at all mean that rotational invariant filters are useless on these kind of images though. EM images of neural tissue do not have any intrinsic preferred direction. Therefore the restriction to rotational invariant features as presented in this section is justifiable. Evaluating the attained continuous functions

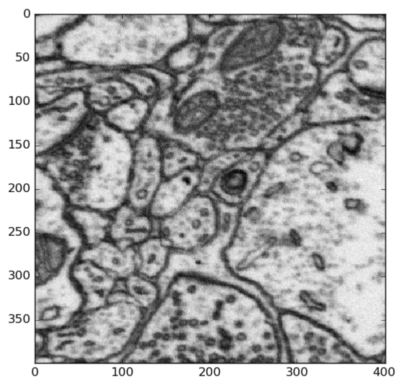
$$f_t\left(\mathbf{x}\right)\in\left\{L\left(\mathbf{x};t\right),\left|\nabla L\left(\mathbf{x};t\right)\right|,\triangle L\left(\mathbf{x};t\right),H\left(\mathbf{x};t\right),S_w\left(\mathbf{x};t,s\right)\right\} \tag{2.10}$$

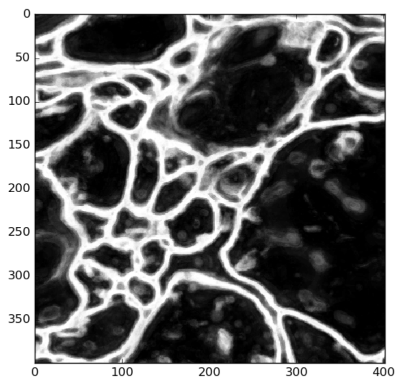
at the centers of the respective pixel squares (see Eq. (2.12)) one reattains a discretized image: $\left(f_t\right)_{ij}=f_t\left(i,j\right)$. All presented features are relying on derivatives of L and therefore derivatives of a Gaussian g convolved with the image I:

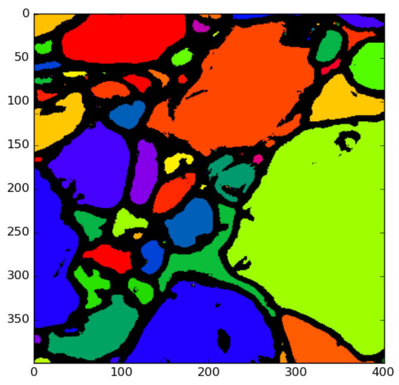
$$\begin{aligned}\partial_{x_i}L\left(\mathbf{x},t\right)&=\partial_{x_i}\left(\frac{1}{\sqrt{2\pi t}}\int\exp\left(-\frac{\left(\mathbf{x}-\mathbf{y}\right)^T\left(\mathbf{x}-\mathbf{y}\right)}{2t}\right)I\left(\mathbf{y}\right)d\mathbf{y}\right)\\&=\frac{1}{\sqrt{2\pi t}}\int\left(\partial_{x_i}\exp\left(-\frac{\left(\mathbf{x}-\mathbf{y}\right)^T\left(\mathbf{x}-\mathbf{y}\right)}{2t}\right)\right)I\left(\mathbf{y}\right)d\mathbf{y}\,.\end{aligned} \tag{2.11}$$

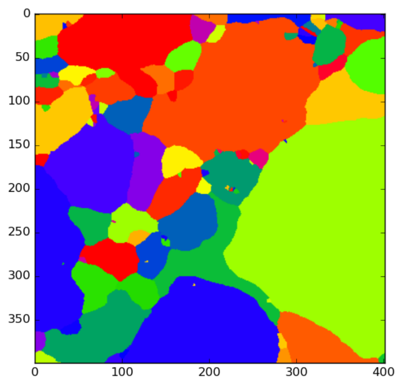
In short $\partial\left(g*I\right)=\left(\partial g\right)*I$ means that all described pixel-wise features can be computed in practice by a convolution of the image with an altered kernel. For actual computations we draw on the open source library for computer vision presented by Köthe et al. [80] called vigra. It is based on `C++`. Most of the functionality is wrapped to `python` making experimentation easy. Taken the filter responses of 2.1.3 on different scales as features one is able to train a classifier for the task at hand: pixel-wise membrane detection. To be able to use the advantages of supervised learning algorithms ground truth labels have to be provided.

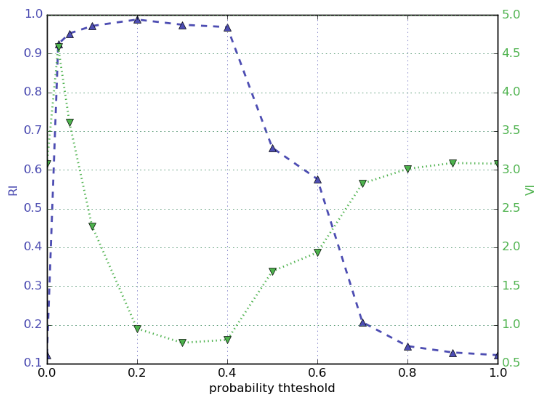
In practice the ilastik software [48] provides an interface for interactive labeling and direct assessment of the resulting classification quality. The core classification algorithm is a random forest as introduced by Liaw and Wiener [81]. It is used in ilastik due to its fast training time (parallelizable since ensemble method) as well as due to its relative robustness with respect to its hyper parameters. One exemplary pair of raw data and respective membrane pseudo probability produced this way is shown in Figure 2.3 (a) and (b).


(a) raw data


(b) membrane probability


(c) threshold and connected components


(d) watershed



(e) segmentation quality vs. threshold

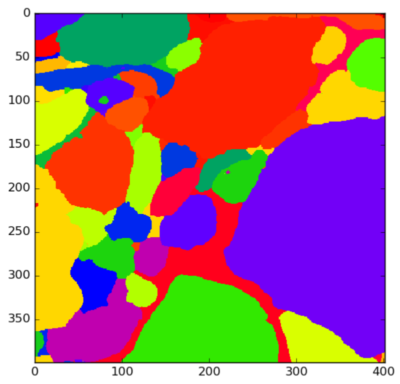

(f) ground truth

Figure 2.3: The figures visualize the steps in a naive segmentation procedure of the neurons shown in (a) described in appendix A.1.3, based on pixel-wise membrane probabilities in (b). As it can be seen in (e), different thresholds for a membrane classification lead to different connected components that do lead to different segmentation quality. (c) shows the classified membrane pixels (in black are the thresholded probabilities at 0.3) and the resulting connected components (color-coded). Given the nature of the ground truth in (f) the results in (c) need to be cleansed of the membrane pixels (as shown in (d)). This is done by a watershed seeded on the already existing segments grown on the probability map itself.

2.1.4 The Need to Go Beyond Pixel Level

The pixel-wise membrane detectors, that are available so far, can only be an interim step on the way to a segmentation because of the sensitivity of the connected components of neurons to single pixel decisions. To make this point clear, Figure 2.3 presents a naive segmentation pipeline using only the pixel-wise membrane-probabilities (learned interactively with ilastik [48] by a random forest using filter responses as introduced in 2.1.3 as features): The membrane probabilities are thresholded and connected components are determined. In order to compare the resulting segmentation to the given ground truth the neurons are grown in to the membrane class via a watershed (see section 2.3) on the probability itself. The best segmentation that can be produced this way is achieved at a threshold of 0.2 when relying on the Rand index (RI) as quality measure (RI = 0.9879. The best variation of information (VI) of VI = 0.7692 can be achieved by applying the threshold of 0.3 (a detailed description of the used quality measures can be found in section 2.6). Note that the absolute numbers are not trivially comparable between different problem instances. Using them to compare approaches on the exact same dataset is much more illuminating.

The fact that the best segmentation is found for a threshold below 0.5 can be explained as follows: Like in general two class-classification problems one can divide possible errors into false positives (pixels that should not be membrane but are classified as such) and false negatives (pixels that should be membrane but are missed). Due to the structure of the neural membranes, false negatives on the pixel level are more crucial than false positives. A few false negative pixels can easily lead to a perforation of a membrane and therefore to a false merge of neurons. A complete mistakenly separated part of one neuron via false positives is less probable. An accidental tight 2D surface requires more errors in the classification, given a sufficient diameter of the processes, than an accidental hole.

The quality (details on the uses measures in chapter 2.6) achievable in this way is not sufficient for an extraction of an meaningful connectome. Still it is remarkable how relatively good the results of this naive approach are and how hard it is to close the gap to perfection.

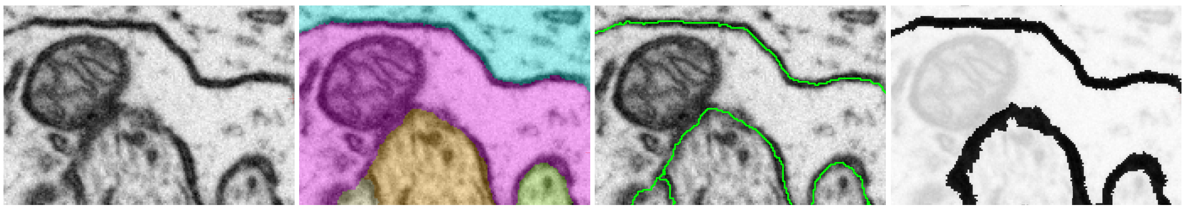
We will show in the later chapters , 4, 5, 6 and 7 how to go beyond the the localized pixel-wise information. All these approaches will rely on superpixels. The following sections will lay the foundation to the understanding of superpixles and their neighboring relations.

2.1.5 Representation of Edges

Within this thesis the representation, in which the desired segmentation is encoded varies. We will utilize two different approaches to which we will refer to as **flat representation** and **voluminous representation**. As their names suggest they differ in the dimensionality of the membranes as they appear in the segmentation.

The most common representation for segmentations is the one shown in 2.4 (b), the implicit flat representation in which each pixel is assigned to a segment id/color. Note that the particular ids are irrelevant. For neighboring pixels only the information of a label transition is crucial. We will see that for some optimization problems the ambiguity of the particular labeling is problematic for actual computations (see chapter 3).

In these cases it will be more useful to explicitly encode the one relevant information: label transition yes or no? It is convenient to introduce interpixel facets as explicit objects that separate the segments. In section 2.2.1 I will describe how these facets can be



(a) raw data (b) id pixel labeling; implicit flat repres. (c) interpixel edges; explicit flat repres. (d) bin. pixel labeling; voluminous repres.

Figure 2.4: Possible representations of a segmentation of neurons in the raw image (a) described in more detail in the appendix A.1: (b) shows a segmentation represented by neuron ids that are assigned to the individual pixels (color-coded). This representation is closely related to (c) where an inter-pixel boundary marks the transition of neurons. (d) provides an alternative approach: A binary classification of the pixels in membrane and no membrane.

administered in practice. In this representation the introduction of constraints enforcing closed surface is feasible (e.g. in chapter 3 and 6). The explicit and the implicit flat representation are different description of the same situations – they are transformable into one another.

In contrast to this, the voluminous representation as depicted in 2.4 (d) is fundamentally different. There exist particular pixels representing the cell membrane. This representation is adapted by all approaches that work directly on a pixel level (as in section 1.3.3). As already mentioned in section 1.3.3, many approaches with impressive pixel-wise accuracy have been proposed. Eventually this is no immediate guarantor for a good resulting segmentation. Single pixel decisions can alter the connectivity of segments and therefore lead e.g. to undesired connections in the connectome (see section 2.1.4).

In chapter 5 I will propose a novel method, the V-Multicut, relying on the voluminous representation. I will overcome the shortcomings that normally come with this representation by introducing closeness inducing constraints. These are inspired by the ones imposed in chapter 3, where the original Multicut algorithm is explained. We will see to what extent the benefits of both representations can be combined.

2.2 Formalizing Segmentation

The basic object in all image analysis tasks is the image or image volume I. We assume it to be a set of N measured intensities y_a where $I = \{y_a; a = 1, 2, \ldots, N\}$ and $y_a \in \mathbb{R}^c$. The image can contain c channel dimensions. It will simplify future definitions to have an explicit name to call the pixel p_a holding the attribute y_a. In all cases discussed within this thesis the pixels will be arranged on a regular grid in d dimensions (primitive cubic system). Therefore the pixel positions can alternatively be indexed by d indices[2]: $p_{i_1 i_2,\ldots,i_d}$ with $i_b \in [1, 2, \ldots, N_b]$. An exact definition of a pixel can be given in the spirit

[2]The mapping between the flat indices p_a and the ones respecting a spatial embedding of the measurents $p_{i_1 i_2,\ldots,i_d}$ can be constructed as follows: $a = \sum_{b=1}^{d} \left(i_b \cdot \left(\prod_{\beta<b} N_\beta \right) \right)$.

of Peters [82] in an appropriately normalized space $\mathbb{R}^d$:

$$p_{i_1 i_2,\ldots,i_d} = \{(x_1, x_2, \ldots, x_d);\ i_b - 0.5 \leq x_b < i_b + 0.5\ \forall x_b \in \mathbb{R}\}\ , \tag{2.12}$$

as a d dimensional cube. Since within this work I will segment 2D and 3D images only, we will restrict ourselves to $d = 2$ and $d = 3$ from now on. The indices/the coordinates of the pixels will play an important role in considerations about digital topology in section 2.2.1. Therefore an explicit symbol is given to the tuple of coordinates for pixel p: $c^{\mathrm{2D}}_{p_{ij}} = (i, j)$ or $c^{\mathrm{3D}}_{p_{ijk}} = (i, j, k)$.

The goal of all segmentation algorithms is the assignment of a segmentation id $S : I \to \mathbb{N}$ to all pixels in the image (volume).

$$S\,(p) \in \{1, \ldots, N\} \tag{2.13}$$

Here N is the total number of segments if all integers in the range $[1, N]$ are used. Such cases are called dense labellings. We explicitly require the mapping S to be designed in a way that the resulting segments, $I \supset s_i = \{p_i;\ S\,(p_i) = i\}$ are spatially connected ($\succ\!\!\!\longrightarrow\!\!\!\prec$)[3]:

$$p_a \in s, p_b \in s \Leftrightarrow S\,(p_a) = S\,(p_b) \overset{!}{\Leftrightarrow} p_a \succ\!\!\!\longrightarrow\!\!\!\prec p_b \tag{2.14}$$

2.2.1 Digital Topology

Given a segmentation $S\,(p)$ by means of a labeled image/volume, it is beneficial to define a construct responsible for the bookkeeping of neighborhood relations. Being able to easily access e.g. all neighboring segments of some particular segment will simplify the theoretical explanations of the algorithms presented in chapters 3, 4, 6 and 7 as well as the actual implementation of these algorithms. In addition the skeletonization algorithm presented in chapter 5 relies on the concepts presented within this section.

For a d dimensional problem there are d natural ways to introduce the neighborhood $N\,(p) \subset P$ of a pixel $p \in P$.

For 2D we can define

$$p_b \in N(p_a) \Leftrightarrow p_a \in N(p_b) \Leftrightarrow \|c^{\mathrm{2D}}_{p_a} - c^{\mathrm{2D}}_{p_b}\| \begin{cases} = 1 & \text{as 4-neighborhood} \\ \leq \sqrt{2} & \text{as 8-neighborhood} \end{cases} \quad \forall a \neq b \tag{2.15}$$

In 3D we can distinguish

$$p_b \in N(p_a) \Leftrightarrow p_a \in N(p_b) \Leftrightarrow \|c^{\mathrm{3D}}_{p_a} - c^{\mathrm{3D}}_{p_b}\| \begin{cases} = 1 & \text{as 6-neighborhood} \\ \leq \sqrt{2} & \text{as 18-neighborhood} \\ \leq \sqrt{3} & \text{as 26-neighborhood} \end{cases} \quad \forall a \neq b \tag{2.16}$$

The 18-neighborhood is not used commonly in practice. The used restriction of $a \neq b$ means that we are defining a pixel to not be in its own neighborhood. Because the frequent usage of these neighborhood relations we will define an adjacency relation ($\succ\!\!\prec$):

$$p_a \succ\!\!\prec p_b \Leftrightarrow p_a \in N_{4/6}(p_b)\ . \tag{2.17}$$

[3]The concept of connectedness is straight-forward and intuitive. For a strict definition see Eq. (2.34).

with respect to the smallest of the available neighborhoods, the 4-neighborhood N_4 and the 6-neighborhood N_6. This neighborhood relations can be easily be generalized to segments. For them, it is not intuitive to speak of 4-/8-/6-/18-/26-neighborhood any more. For 2D we define:

$$N^{2D}_{\text{line}}(s_i) \ni s_j \Leftrightarrow \exists p_1 \in s_i,\, p_2 \in s_j \text{ s.t. } N_4(p_1) \ni p_2 \tag{2.18}$$
$$N^{2D}_{\text{vertex}}(s_i) \ni s_j \Leftrightarrow \exists p_1 \in s_i,\, p_2 \in s_j \text{ s.t. } N_8(p_1) \ni p_2 \tag{2.19}$$

And analogous in 3D the definitions are:

$$N^{3D}_{\text{face}}(s_i) \ni s_j \Leftrightarrow \exists p_1 \in s_i,\, p_2 \in s_j \text{ s.t. } N_6(p_1) \ni p_2 \tag{2.20}$$
$$N^{3D}_{\text{line}}(s_i) \ni s_j \Leftrightarrow \exists p_1 \in s_i,\, p_2 \in s_j \text{ s.t. } N_{18}(p_1) \ni p_2 \tag{2.21}$$
$$N^{3D}_{\text{vertex}}(s_i) \ni s_j \Leftrightarrow \exists p_1 \in s_i,\, p_2 \in s_j \text{ s.t. } N_{26}(p_1) \ni p_2 \tag{2.22}$$

It is important to think about the different neighborhood systems when talking about connectedness. Think about a binary labeling of an image where a closed path in the foreground region passes through a connected background region. We see that using the 8-neighborhood in both classes can lead to unintuitive configurations. This phenomenon is known as a connectivity paradox or as a topological paradox in the literature (e.g. described by Kong and Rosenfeld [83]). In chapter 5 where we aim to construct constraints based on the connectedness of the different classes this insight is vital. Also in the chapters 3, 4, 6 and 7 it is important to have a clear understanding of adjacency of superpixels.

Within this work I will utilize two kinds of data structures responsible for the bookkeeping of all neighborhood relations. Both will introduce explicit objects for the contact area of neighboring superpixels. This is beneficial since they allow an easy, lower dimensional representation of membranes in the case of neuron segmentation (a perspective that is e.g. take in sections 3.1.2 and 6.2.2 and visualized in Figure 2.4 (c)). The first is a classical region adjacency graph utilized in a large variety of applications. Nunez-Iglesias et al. [84] and Trémeau and Colantoni [85] e.g. draw on this data structure. The second is the abstract cell complex originated in digital topology mentioned in the early work of Listing [86] and summarized by Klette and Rosenfeld [87]. It is for example deployed by Andres et al. [73].

Region Adjacency Graph

A region adjacency graph is a common undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ whose nodes $\mathcal{V}$ correspond to localized regions in an image(volume) I defined via a segmentation S. In our case each node $n_i \in \mathcal{V}$ corresponds to one segment s_i. The edges are determined by the neighborhood relations of the segments. If two regions/segments s_i and s_j are adjacent

$$s_i \rightarrowtail\!\!\!\!\leftarrowtail s_j \Leftrightarrow \exists p_a \in s_i \text{ and } p_b \in s_j \text{ s.t. } p_a \rightarrowtail\!\!\!\!\leftarrowtail p_b\,, \tag{2.23}$$

there is one respective edge $e \in \mathcal{E}$, otherwise not.

$$\mathcal{V} = \{1, \ldots, N\} \tag{2.24}$$
$$\mathcal{E} = \{e = (S(s_i), S(s_j))\,;\ \forall s_i, s_j \text{ if } s_i \rightarrowtail\!\!\!\!\leftarrowtail s_j \text{ and } i < j\} \tag{2.25}$$

The (arbitrary) restriction $i < j$ makes sure that edges are unique.

Abstract Cell Complex

The abstract cell complex as motivated and described by Köthe et al. [88] and Kovalevski [89] is an abstract set. Each element/cell in this set has a nonnegative integer dimension d assigned to it. For practical purposes we can restrict the abstract cell complex to the dimensionality of three. This means that cells of dimension 0,1,2 and 3 can occur. In the following they are refered to as pointels, edgels, surfels and voxels.

The introduction of a topological grid simplifies the definition of the relation between the cells since there they can be uniquely located. Given an original image volume with pixels p and coordinates $c_p = (i, j, k)$ with $i \in \{1, \ldots, N_x\}$, $j \in \{1, \ldots, N_y\}$, $k \in \{1, \ldots, N_z\}$ an associated topological grid is be of shape $(2N_x - 1,\ 2N_y - 1,\ 2N_z - 1)$. In between each original pixel a new position has been created to explicitly describe e.g. the boundaries separating two objects. The pixels in the topological grid are refereed to as $\tilde{p}_{mno}$ with the topological coordinates $\tilde{c}_{\tilde{p}} = (m, n, o)$. The dimensionality of a topological cell associated with a topological pixel in the topological grid $\dim(\tilde{p}_{mno})$ is directly coupled with its coordinates. More specifically with the number of odd coordinates determined via the modulo operator %:

$$\dim(\tilde{p}) = (1+m)\%2 + (1+n)\%2 + (1+o)\%2\,. \tag{2.26}$$

Whenever relevant the dimensionality of a topological pixel is denoted in the superscript: $\dim\left(\tilde{p}^d_{mno}\right) = d$. A bounding relation between two $\tilde{p}$ of different dimension can be established:

$$\tilde{p}^d_{mno} \succ \tilde{p}^{d-1}_{qrs} \Leftrightarrow \|\tilde{c}^d_{mno} - \tilde{c}^{d-1}_{qrs}\| = 1 \text{ for } 0 < d \leq 3\,. \tag{2.27}$$

The bounding relation can be generalized:

$$\tilde{p}^{d_1}_{mno} \succ \tilde{p}^{d_2}_{qrs} \Leftrightarrow \exists \left\{\tilde{p}^{d_1-1}, \tilde{p}^{d_1-2}, \ldots, \tilde{p}^{d_2+1}\right\} \text{ s.t. } \tilde{p}^{d_1}_{mno} \succ \tilde{p}^{d_1-1} \succ \ldots \succ \tilde{p}^{d_2}_{qrs} \text{ for } d_2 < d_1 \leq 3\,. \tag{2.28}$$

These bounding relations are important also because the adjacency relations between cells of certain dimension d can be easily established by them:

$$\tilde{p}^d_i \rightarrowtail\!\!\!\!\leftarrowtail \tilde{p}^d_j \Leftrightarrow \exists \tilde{p}^{d-1} \text{ s.t. } \tilde{p}^d_i \succ p^{d-1} \text{ and } p^{d-1} \prec \tilde{p}^d_j\,. \tag{2.29}$$

This relation together with the placement of the dimensionalities on the topological grid by Eq. (2.26) results in an asymmetric neighborhood system for the surfels and the edgels. This is not unintuitive since in 3D surfaces and lines, geometrical objects the respective cells can be associated with, do have a preferred direction. The explicit adjacency relations using only the topological coordinates look like this:

$$\tilde{p}^3_i \rightarrowtail\!\!\!\!\leftarrowtail \tilde{p}^3_j \Leftrightarrow \|\tilde{c}_{p^3_i} - \tilde{c}_{p^3_j}\| = 2 \tag{2.30}$$

$$\begin{aligned}\tilde{p}^2_i \rightarrowtail\!\!\!\!\leftarrowtail \tilde{p}^2_j \Leftrightarrow \Big(&\|\tilde{c}_{p^2_i} - \tilde{c}_{p^2_j}\| = \sqrt{2} \\ \text{or}\quad &\|\tilde{c}_{p^2_i} - \tilde{c}_{p^2_j}\| = 2 \text{ and } \tilde{c}_{p^2_i}\%2 = \tilde{c}_{p^2_j}\%2\Big)\end{aligned} \tag{2.31}$$

$$\begin{aligned}\tilde{p}^1_i \rightarrowtail\!\!\!\!\leftarrowtail \tilde{p}^1_j \Leftrightarrow \Big(&\|\tilde{c}_{p^1_i} - \tilde{c}_{p^1_j}\| = \sqrt{2} \\ \text{or}\quad &\|\tilde{c}_{p^1_i} - \tilde{c}_{p^1_j}\| = 2 \text{ and } \tilde{c}_{p^1_i}\%2 = \tilde{c}_{p^1_j}\%2\Big)\end{aligned} \tag{2.32}$$

$$\tilde{p}_i^0 \rightarrowtail\!\!\!\prec \tilde{p}_j^0 \Leftrightarrow 1 = 0 \tag{2.33}$$

where the modulo operation % acts element-wise. With this definition of the topological grid at hand it is possible to define the cell complex representation of a labeled volume S. Here we are interested in connected components of these voxels, surfels, edgels and pointels. The connected components are defined via a path connection $\succ\!\!\!-\!\!\!\prec$:

$$\begin{aligned} p_1^d \succ\!\!\!-\!\!\!\prec p_n^d \Leftrightarrow \exists \left\{\tilde{p}_1^d, \tilde{p}_2^d, \ldots, \tilde{p}_n^d\right\} &\text{ with } \tilde{p}_1^d \succ\!\!\prec \tilde{p}_2^d \succ\!\!\prec \ldots \succ\!\!\prec \tilde{p}_n^d \\ &\text{and } \widetilde{\mathrm{ID}}\left(\tilde{p}_1^d\right) = \widetilde{\mathrm{ID}}\left(\tilde{p}_2^d\right) = \cdots = \widetilde{\mathrm{ID}}\left(\tilde{p}_n^d\right) \end{aligned} \tag{2.34}$$

The function $\widetilde{\mathrm{ID}}$ is introduced here only for the purpose of a cleaner notation.

$$\widetilde{\mathrm{ID}}\left(\tilde{p}^3\right) = \begin{cases} \text{not existant} & \text{for 2D} \\ S\left(p^3\right) & \text{for 3D} \end{cases} \tag{2.35}$$

$$\widetilde{\mathrm{ID}}\left(\tilde{p}^2\right) = \begin{cases} S\left(p^2\right) & \text{for 2D} \\ \left\{\widetilde{\mathrm{ID}}\left(\tilde{p}_1^3\right), \widetilde{\mathrm{ID}}\left(\tilde{p}_2^3\right)\right\} \text{ with } \tilde{p}_1^3 \prec \tilde{p}^2 \text{ , } \tilde{p}_2^3 \prec \tilde{p}^2 & \text{for 3D} \end{cases} \tag{2.36}$$

$$\widetilde{\mathrm{ID}}\left(\tilde{p}^1\right) = \left\{\widetilde{\mathrm{ID}}\left(\tilde{p}_1^2\right), \ldots, \widetilde{\mathrm{ID}}\left(\tilde{p}_k^2\right)\right\} \text{ with } \tilde{p}_i^2 \succ \tilde{p}^1 \text{ for } i \in \{1, \ldots, k\} \text{ and } k \in \{3, 4\} \tag{2.37}$$

$$\widetilde{\mathrm{ID}}\left(\tilde{p}^0\right) = \tilde{c}\left(\tilde{p}^0\right) \tag{2.38}$$

Path connections of topological voxels $\tilde{p}^3$ are mediated via other topological voxels whose respective pixles p^3 have the same segmentation id. $S\left(p^3\right)$ surfels/edgels are connected if they are in their neighborhood and if they bound the same segments/surfaces. No pointel is connected with an other pointel. The relation of the connected components can be expressed by assigning a label $\mathrm{T}(\tilde{p})$ to all cells associated with the topological pixels $\tilde{p}$:

$$T\left(\tilde{p}_i^d\right) = T\left(\tilde{p}_j^d\right) \Leftrightarrow p_i^d \succ\!\!\!-\!\!\!\prec p_j^d \text{ and } \widetilde{\mathrm{Active}}\left(\tilde{p_i}^d\right) \text{ and } \widetilde{\mathrm{Active}}\left(\tilde{p_j}^d\right) . \tag{2.39}$$

A connected component (defined by common label T) of voxels is called supervoxel. We will refer to connected components of surfels as faces. To avoid confusion with the adjacency graph nomenclature the connected components of edgels are referred to as lines. We will refer to connected components of surfels/edgels only as faces and lines if they are actually seperating two supervoxels/faces:

$$\widetilde{\mathrm{Active}}\left(\tilde{p}^3\right) = \begin{cases} \text{not existant} & \text{for 2D} \\ \text{True} & \text{for 3D} \end{cases} \tag{2.40}$$

$$\widetilde{\mathrm{Active}}\left(\tilde{p}^2\right) = \begin{cases} \text{True} & \text{for 2D} \\ \widetilde{\mathrm{ID}}\left(\tilde{p}_1^3\right) \overset{?}{\neq} \widetilde{\mathrm{ID}}\left(\tilde{p}_2^3\right) \text{ with } \tilde{p}_1^3 \succ \tilde{p}^2 \text{ , } \tilde{p}_2^3 \succ \tilde{p}^2 & \text{for 3D} \end{cases} \tag{2.41}$$

$$\widetilde{\mathrm{Active}}\left(\tilde{p}^1\right) = \widetilde{\mathrm{ID}}\left(\tilde{p}_i^2\right) \overset{?}{\neq} \widetilde{\mathrm{ID}}\left(\tilde{p}_{j\neq i}^2\right) \ \forall \tilde{p}_i^2 \succ \tilde{p}^1 \text{ , } \tilde{p}_{j\neq i}^2 \succ \tilde{p}^1 \tag{2.42}$$

$$\widetilde{\mathrm{Active}}\left(\tilde{p}^0\right) = \widetilde{\mathrm{ID}}\left(\tilde{p}_i^1\right) \overset{?}{\neq} \widetilde{\mathrm{ID}}\left(\tilde{p}_{j\neq i}^1\right) \ \forall \tilde{p}_i^1 \succ \tilde{p}^0 \text{ , } \tilde{p}_{j\neq i}^1 \succ \tilde{p}^0 . \tag{2.43}$$

The mapping $T : \mathbb{N}^D \to \mathbb{N}$ can be seen analogous to the mapping S defining the segmentation. It makes the definition of the concrete objects $t^d = \{\tilde{p}_i^d | T\left(\tilde{p}_i^d\right) = t\}$ possible. All relations between these objects can be related to the respective objects on the level of the topological pixels $\tilde{p}$:

$$t_1^{d_1} \sim t_2^{d_2} \Leftrightarrow \exists \tilde{p}_1^{d_1} \in t_1^{d_1}, \tilde{p}_2^{d_2} \in t_1^{d_2} \text{ s.t. } \tilde{p}_1^{d_1} \sim \tilde{p}_2^{d_2} \text{ with } \sim \in \{\succ, \prec, \succ\!\prec, \succ\!\!-\!\!\!-\!\!\prec\}\,.$$

For pixels of the original (not the topological) grid one can use the correspondence:

$$p_{abc} \succ\!\!-\!\!\!-\!\!\prec p_{ijk} \Leftrightarrow \tilde{p}^3_{(2a)(2b)(2c)} \succ\!\!-\!\!\!-\!\!\prec \tilde{p}^3_{(2i)(2j)(2k)} \tag{2.44}$$

$$p_{abc} \succ\!\prec p_{ijk} \Leftrightarrow \tilde{p}^3_{(2a)(2b)(2c)} \succ\!\prec \tilde{p}^3_{(2i)(2j)(2k)} \tag{2.45}$$

The major difference between the edges from section 2.2.1 and the faces defined here is the following: There can be multiple faces between two segments but there can be only one edge. The reason we are not limiting ourselves here to the simpler region adjacency graph from section 2.2.1 is twofold. On the one hand this formalism makes the explanation of concepts from chapter 5 easier. On the other hand the computation of merge probabilities can be done more differentiated. Later in section 2.4 I will go into more detail about the classification of faces. In the made experiments the overhead of the abstract cell complex representation is valuable in cases where the uncertainty of the classifier for the faces is inaccurate. This can happen for very small faces where the statistics are insufficient. In these cases it can be beneficial to compose the edge-weights from the face-weights. It seems to be more natural to classify spatially separated objects individually.

Eventually it makes sense to end up with a representation of the region adjacency graph if one aims for an implicitly flat representation. Multiple edges between two nodes are restricted to the same state anyways. Section 2.4.3 discusses different possible mappings from face-wise predictions to edge-wise predictions.

2.3 Superpixels

Oftentimes starting a segmentation procedure from the pixel level is hard. The runtime of advanced algorithms becomes intractable and the features one can use to classify possible merges are weak as can be seen in section 2.4. Some algorithms, like the Multicut, even rely completely on an initial oversegmentation and would even with good features and independent of runtime produce non-sensible results on am initial pixel level (see section 3.2.4). A stepwise approach in the context of neuron segmentation has been proven to be beneficial by Andres et al. [67].

In the following we will treat superpixels as a general segmentation with some additional demands $\{SP(p)\} \subset \{S(p)\}$. Given a ground truth segmentation $S_{\text{gt}}(p_i) \to \{1, \ldots, N_{\text{gt}}\}$ the desired properties of superpixels $sp_a = \{p_i;\ SP\,(p_i) = a\}$ are that they do not lead to undersegmentation:

$$SP\,(p_i) = SP\,(p_j) \overset{!}{\Rightarrow} S_{\text{gt}}(p_i) = S_{\text{gt}}(p_j)\,. \tag{2.46}$$

An oversegmentation:

$$SP\,(p_i) \neq SP\,(p_j) \overset{!}{\not\Rightarrow} S_{\text{gt}}(p_i) \neq S_{\text{gt}}(p_j) \tag{2.47}$$

is uncritical. Therefore one can refer to superpixel segmentation as oversegmentation. Since taking the original pixels as superpixles one will ensure Eq. (2.46) but this is not the only applicable quality criterion. Good superpixels distinguish themselves not only by the fact that Eq. (2.46) holds at most parts but also by their size (see Figure 2.5).

Since in most cases the distinction between 2D and 3D is not necessary, the term superpixels will encompass 2D superpixels as well as 3D supervoxels as the term pixels is used to speak of 2D pixels as well as of 3D voxels. At places where the distinction is necessary I will point this out explicitly.

There is no one best superpixel algorithm for all applications. For natural images where step edges in the color space are most common the so called SLIC superpixels are performing well. Achanta et al. [90] demonstrate their generally solid performance. The principle idea behind this algorithm is a k-means clustering in the combined space of coordinates and color. Since cytoplasm in EM-recordings mostly looks alike independent of the neuron it belongs to, a clustering according to color similarity, as performed by SLIC, is not expedient. Therefore, for my experiments I draw upon the Watershed algorithm. The watershed algorithm relies on two inputs: a set of seed-pixels $\{\text{seed}_s\}$ and a scalar map the size of the original image, the "height-map" $H(p) \in \mathbb{R}$.

The watershed $\text{WS}(p)$ is a way to assign each pixel p to one of the seeds. The mapping is described by:

$$\begin{aligned} \text{WS}(p) =& \text{argmin}_s \left(\text{min}_{\text{Path} \in \mathcal{P}_s} \left(\text{max}_{p_{i_a} \in \text{Path}} \left(H\left(p_{i_a}\right)\right)\right)\right) \\ \mathcal{P}_s :=& \left\{\left(p_{i_1}, p_{i_2}, \ldots, p_{i_n}\right) \mid p \asymp p_{i_1} \asymp p_{i_2} \asymp \ldots \asymp p_{i_n} \asymp \text{seed}_s\right\}. \end{aligned} \tag{2.48}$$

The pixel p gets assigned to the natural number s corresponding to the seed which is reachable over a path with the least maximal hight on it. The seed that is assigned to a pixel p is the one of all seeds that is first reached in a gedankenexperiment by water raining on p if H gives the height of a landscape.

From Eq. (2.46) follows that at least one seed needs to be placed within one of the segments of a desired segmentation in order to avoid undersegmentation - a wrongfully merging of segments. The resulting requirement on the height map H is that a ridge exists where a segment transition is desired. A learned edge indicator as shown in Figure 2.3 (b) would suffice.

2.4 Face-wise Predictions

Given some meaningful oversegmentation, e.g. $WS(p)$ from Eq. (2.48) the next step is to merge some of the superpixels in order to end up with the desired segmentation. While the superpixel procedure was solely relying on pixel-wise information the oversegmentation gives the problem a structure that can be deployed. The decision whether two neighboring superpixels do belong to the same segment in the final segmentation can be based e.g. on the shape of the superpixels, their touching surface and statistics of pixel-wise informations on the surface as well as on the region themselves.

At this stage the decision on whether to use the region adjacency graph based edges (section 2.2.1) or the cell complex based faces (section 2.2.1) becomes relevant. The feature accumulation and estimation of local merge probability will be effected by this decision. Since all results originating from a region adjacency graph representation can be replicated by a cell complex representation, the latter one is chosen for convenience.

2.4.1 Feature Accumulation

The eventual goal is the estimation of a local merge probability of pairs of neighboring superpixels based on local evidence. This estimation is done by a random forest classifier based on extracted features. As a basis for feature extraction procedures the raw data as well as pixel-wise features (as described in section 2.1.3) $L(\mathbf{x};t)$, $|\nabla L(\mathbf{x};t)|$, $\triangle L(\mathbf{x};t)$, $H(\mathbf{x};t)$, $S_w(\mathbf{x};t,s)$, for different smoothing parameters t and s, can be utilized. One can use statistics either along the pixels that are bounded by the touching face or one can use statistics on the both superpixles.

Face-Statistics

Statistics on the facelets of which single faces are composed of are not possible since no data is available. Remember that the raw data provides information only for the voxles. One facelet bounds two voxles: $\tilde{p}^2 \prec \tilde{p}^3_1$, $\tilde{p}^2 \prec \tilde{p}^3_2$. The statistics for the face can therefore be computed on the set of all voxels that are bounded by some facelet of the face.

In the presented experiments I will use the mean, median, variance, quantiles, kurtosis and skewness on the grayvalues of the respective voxels as well as the total number of faceltes.

Region-Statistics

Statistics can also be computed on the gray-values of the different superpixels. The same set of statistics used before are computed here as well. In addition the radii of the segments and histograms on the gray-values are computed. Two feature descriptors f_u, f_v computed this way need to be mapped to one feature descriptor of the respective face. This is done by the following three operations:

$$f_{\text{face}(u,v)} = \begin{pmatrix} \min(f_u, f_v) \\ \max(f_u, f_v) \\ |f_u - f_v| \end{pmatrix}. \tag{2.49}$$

The three operations are meant element-wise. Min and max are reasonable since they introduce an ordering that would otherwise be arbitrary. The absolute difference of the values is redundant and therefore any learning algorithm should be able to learn the relation in principle. Since looking at the difference of the descriptors is one obvious way to reason about the merge-probability of two segments it is justifiable to add this feature. The limited number of training examples do not need to be used to learn this dependency but can be used to learn less obvious dependencies.

2.4.2 The Benefit of Huge Superpixels

If the underlying oversegmentaion is too fine, the extreme case would be the pixelgrid, the expressiveness of the accumulated features is strongly limited. Therefore, in practice, a reasonably coarse oversegmentation is a necessary prerequisite.

The following experiment on the data described in appendix A.1.3 illustrates the benefits of big superpixels. The watershed algorithm is used to construct an oversegmentation. The used underlying height-map H is constructed with the help of a pixel-wise membrane prediction of a random forest based on features as e.g. shown in Figure 2.3 (b). Specifically the height map H_{g2}, a slightly smoothed version (Gaussian smoothing with a sigma of 2) of the pixel-wise membrane prediction, is taken. The smoothing leads to less wiggly faces, especially on the membranes.

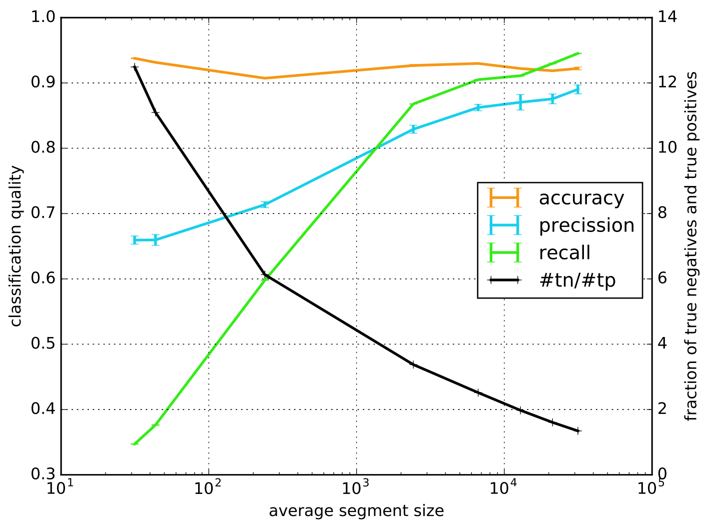


Figure 2.5: Expressive power of features computed for superpixels of different average size measured by random forrest classification quality measures

For this experiments the number of segments in the oversegmentation must be varied. The seeds are therefore identified with the local minima of of a smoothed version (Gaussian smoothing with a variable sigma) of $H + \frac{1}{3}N$. Here N is white noise of the same range as H. The variation of the smoothing parameter allows the adjustment of the number of resulting seeds and therefore resulting segments. In Figure 2.5 the horizontal axis represents the average segment size resulting from the varying number of segments. The quality measures are determined by ten-fold cross validation. Even though the number of training examples decreases for bigger average segment sizes both the recall and the precision profit from big super-pixels. Interestingly the accuracy stays roughly constant.

The reason is hidden in the structure of the neural membranes. All membrane faces are basically distributed on a two dimensional subspace. The non-membane faces are located within the neuron bodies, therefore living in the full three dimensional space. By consistent increase of the number of segments the proportion of true non-membrane to true membrane faces increases. Assuming that adding faces in the center of neurons is equivalent with adding “trivial” decisions, the number of true negative predictions rises. This affects the accuracy but not precision and recall (see section 2.6.1).

The overall message is that the features on which the face classification is based on get better with increasing superpixel size. This does not mean that one should use the biggest superpixels possible. Due to the simplicity of the method there will be violations of Eq. (2.46) – there will be undersegmentation. In most possible subsequent processing steps this can not be corrected.

2.4.3 From Face Predictions to Edge-weights

In the previous section we learned of a way to predict the class affiliation of individual faces/edges. Since an ensemble method (the random forest) is used to condense the accumulated features, one is left with some kind of a pseudo probability p_{ij}. In following chapters I will propose different methods how to cluster the superpixels based on some weight w_{ij}. The question arises how both values should be related. One way is to identify them one to one:

$$w_{ij} = p_{ij} \,. \tag{2.50}$$

One alternative is to utilize the Gibbs measure explained in more detail in Eq. (2.59). The reasoning is the following: If we are formulating the clustering as an energy minimization process in the form of a Markov random field it is natural to interpret w_{ij} as little contributions to the energy. Following this argumentation, taking the probabilities for both events split (p_{ij}) and merge ($1-p_{ij}$) and following the idea to gauge the energies in a convenient way from Eqs. (3.5) and (3.6), the weights/energies would look like this:

$$w_{ij} = E_{ij} = -\log\left(\frac{p_{ij}}{1-p_{ij}}\right), \tag{2.51}$$

where weights bigger zero correspond to an attraction and negative weights correspond to a repulsion of the segments i and j. A minimization of the sum of energies does then correspond to a maximization of the product of the probabilities. The desired solution is the maximum a posteriori (MAP) estimation.

Eventually only experimental results will decide for the the direct approach from Eq. (2.50) and the evolved way from Eq. (2.51). Therefore the following experiment was carried out: The best superpixels from section 2.4.2 were taken. Edgeweights were computed as described and evaluated in 2.4.2 and depicted in Figure 2.5. Then both alternative methods from Eqs. (2.50) and (2.51) are applied to attain two set of edge-weights. Based on each of them the Multicut algorithm is applied to perform the final clustering (see chapter 3 for details). In addition to the mapping from face-wise-probabilities to face-wise weights, the mapping to edge-weights must be done (as justified also in 2.2.1). This is the place where different mappings from face to edge can be compared. We restrict ourself to the following possibilities:

$$w^e_{\text{min}} = \min_{f\in e}\left(w^f\right) \tag{2.52}$$

$$w^e_{\text{max}} = \max_{f\in e}\left(w^f\right) \tag{2.53}$$

$$w^e_{\text{sum}} = \sum_{f\in e} w^f \tag{2.54}$$

$$w^e_{\text{mean}} = \frac{\sum_{f\in e} w^f}{\sum_{f\in e} 1} \tag{2.55}$$

$$w^e_{\text{wmean}} = \frac{\sum_{f\in e} w^f \|f\|}{\sum_{f\in e}\|f\|} \tag{2.56}$$

$$w^e_{\text{swmean}} = \frac{\sum_{f\in e} w^f \sqrt{\|f\|}}{\sum_{f\in e}\sqrt{\|f\|}}. \tag{2.57}$$

Here $f \in e$ states that the face f is a part of the edge e: $f \in e = (i,j) \Leftrightarrow f \prec i,\ f \prec j$. $\|f\|$ denotes the number of facelets, the face f consists of. The quality of the segmentation is measured by the Rand index and the variation of information (see section 2.6). The results of this comparison can be seen in Table 2.1. Almost exclusively the Gibbs weights from Eq. (2.51) exceed the direct ones from Eq. (2.50). In this experiment, taking the maximal value of the membrane probabilities of the composing faces for the edges is the best strategy for mapping face-weight to edge-weights. The fact that the mean is worse than the weighted mean supports the statement made in 2.2.1 that small faces' predictions are unreliable and supports the usage of the abstract cell complex over the region adjacency graph where such a differentiated prediction is not possible.

RI	min	mean	sum	swmean	wmean	max
direct (2.50)	0.98462	0.98661	0.99085	0.99302	0.99307	**0.99425**
Gibbs (2.51)	0.98491	0.98592	0.99221	0.99310	0.99324	**0.99460**

VI	min	mean	sum	swmean	wmean	max
direct(2.50)	0.75315	0.63917	0.56433	0.42600	0.41431	**0.38714**
Gibbs (2.51)	0.73628	0.63928	0.49505	0.40824	0.40515	**0.36415**

Table 2.1: This table illustrates the dependence of the final segmentation quality on both the mapping from face-probabilities to face-weights (direct vs. Gibbs) as well as the dependence on the mapping from face- to edge-weights (min vs. mean vs. sum vs. swmean vs. wmean vs. max). The comparison was done on an image volume described in appendix A.1.3.

2.5 Graphical Models

Markov Random Fields As a preparation for the following chapters it is helpful to introduce Markov random fields (MRF). Both the Multicut and the Asymmetric Multiway Cut can be formulated within this framework.

MRFs are a way to depict conditional (in-)dependences of variables x_i described with the joint probability distribution $p\left(x_1, x_2, \ldots, x_N\right)$. The conditional dependencies are implied via edges in a graph G whose nodes correspond to the random variables. The goal is to factorize the joint probability in the fashion of Markov networks:

$$p\left(x_1, x_2, \ldots, x_N\right) = \frac{\prod_c \phi_c\left(\mathcal{X}_c\right)}{\sum_{x_1,x_2,\ldots,x_N} \prod_c \phi\left(\mathcal{X}_c\right)}\,, \tag{2.58}$$

where ϕ_c are the potentials, non negative functions, corresponding to the c'th maximal clique $\mathcal{X}_c$ of G. A clique $\mathcal{X}_i$ is a subset of the random variables that are all connected to each other. It becomes a maximal clique if it is assured that there is no other clique $\mathcal{X}_j$ with $\mathcal{X}_j \supset \mathcal{X}_i$. MRFs can be defined by a set of conditional distributions $p\left(x_i|N\left(x_i\right)\right)$ where $N\left(x_i\right)$ is the set of random variables in the neighborhood of x_i established by edges in the G. In other words there exists an edge in the graph between the nodes representing x_i and x_j if $p\left(x_i, x_j|x_{\setminus\{i,j\}}\right) \neq p\left(x_i|x_{\setminus\{i,j\}}\right) p\left(x_j|x_{\setminus\{i,j\}}\right)$.

According to the **Hammersley Clifford Theorem** a set of local conditional distributions $p\left(x_i|N\left(x_i\right)\right)$ can only form a consistent joint distribution $p\left(x_1,x_2,\ldots,x_N\right)$ if and only if that joint distribution can be written as a Gibbs random field:

$$p\left(x_1, x_2, \ldots, x_N\right) = \frac{\exp\left(-\beta \sum_c E_c\left(\mathcal{X}_c\right)\right)}{\sum_{x_1,x_2,\ldots,x_N} \exp\left(-\beta \sum_c E_c\left(\mathcal{X}_c\right)\right)}\,, \tag{2.59}$$

with a set of real valued functions E_c. In Eq. (2.58) this does translate into positive potentials ϕ. In the field of statistical physics Eq. (2.59) relates to the canonical ensemble, taking into account all possible states of a mechanical system.

Factor Graphs Eq. (2.58) and therefore also Eq. (2.59) are in the form of

$$f\left(x_1, x_2, \ldots, x_N\right) = \prod_i \psi_i\left(\mathcal{X}_i\right)\ ; \tag{2.60}$$

which makes it possible to write them as a factor graph. A factor graph consists of nodes representing the random variables x_i as well as of nodes representing the factors $\psi_i(\mathcal{X}_i)$. Graphically the second type is often times distinguished by rectangles. For each factor $\psi_i(\mathcal{X}_i)$ edges to all variables $x_j \in \mathcal{X}_i$ are introduced. This illustration is e.g. utilized in Figures 3.1 and 6.2, where the respective models, the Multicut and the Asymmetric Multiway Cut, are depicted.

For a more detailed introduction to MRFs and Factor Graphs the interested reader is referred to the textbook of Barber [91].

2.6 Quality Measures

It is of importance that the usability of all made efforts for the final goal, the reconstruction of the connectome, can be judged. Besides the visual inspection of the results, sensible quantitative quality measures are needed. Judging the quality of a pixel-wise/face-wise membrane prediction is a common task in classifications and is described in section 2.6.1. These measures are not reliably correlated with error measures on the connection network. They are not completely unrelated though. A perfect pixel-wise probability will necessarily translate in a perfect connectome.

Error measures on the level of segmentations, as described in section 2.6.2, will give a more reliable estimate of the resulting connectivity quality. The wrongfully merge of two neurons could be caused by a single unfortunate pixel-decision, merely reflected in the respective error measure. The error measures working on segmentation level will in contrast be severely effected as any sensible error measures on the connectome level would be.

2.6.1 Error Measures for Classification

If dealing with a classification problem with n classes the construction of a so called confusion matrix $M \in \mathbb{N}^{n\times n}$ is advantageous. M_{ij} reflects the number of times the true class is i and the given classifier returns class j. A perfect classifiers confusion matrix therefore would contain exclusively zero valued off-diagonal elements.

Precision and Recall are commonly used to describe aspects of the classification qualify for class i:

$$\text{Precision}_i := \frac{M_{ii}}{\sum_j M_{ji}} \tag{2.61}$$

$$\text{Recall}_i := \frac{M_{ii}}{\sum_j M_{ij}}\,. \tag{2.62}$$

The precision is the proportion of rightly classified elements of class i vs all elements classified as i. The recall is the proportion of rightly classified elements of class i vs all elements that are truly of class i.

While precision and recall are defined per class, the accuracy gives an overall measure for the classification quality:

$$\text{Accuracy} := \frac{\sum_i M_{ii}}{\sum_{ij} M_{ij}}\,. \tag{2.63}$$

In the special case of a binary classification, where one class can be identified as the class of interest that the classifier is looking for, often times the precision and the recall are stated for the classification without specifying this class as a shorthand notation. (E.g. the precision of a classifier detecting a disease would refer to $\text{Precision}_{\text{desease}}$ and not to $\text{Precision}_{\text{no desease}}$). One example for the usage of these quality measures for classification within this work is the classification of superpixel faces as shown in Figure 2.5.

2.6.2 Error Measures for Segmentation

When comparing two segmentations the actual segmentation ids are irrelevant. A direct translation of the error measures of 2.6.1 is therefore not possible. The relevant information of any segmentation/partition is the pairwise affiliation. Either two elements belong to the same segment or not. Therefore one way to calculate similarity measures for segmentations will be based on pairs of elements (Rand index). Another way is utilizing the concept of mutual information (variation of information). If one of the compared segmentation is the ground truth the similarity measure becomes a quality measure for the other.

Rand Index

The Rand index RI is the segmentation equivalent of the accuracy based on pairs of elements instead of elements themselves. Given the elements p_i and their cluster affiliation/segmentation id $S_1(p_i)$ from the first segmentation or $S_2(p_i)$ from the second one the Rand index can be computed as:

$$\mathrm{RI} = \frac{\sum_i \sum_{j\neq i} \delta\left(\delta\left(S_1(p_i), S_1(p_j)\right), \delta\left(S_2(p_i), S_2(p_j)\right)\right)}{\sum_i \sum_{j\neq i} 1} \tag{2.64}$$

with δ being defined as

$$\delta(a,b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ otherwise}. \end{cases} \tag{2.65}$$

If both segmentations S_1 and S_2 do agree that two elements i and j should be separated or merged (have different/identical ids) the Rand index profits. The Rand index ranges between zero and one. Higher Rand indices do usually correspond also visually to better correspondence of the pair of segmentations. This error measure is only reliable if the segmentations are somehow similar and the segment size is not too small in comparison to the pixel-size.

Variation of Information

The variation of information is based on reflections on the information content I, sometimes sloppily referred to as surprisal, of the clusterings. In general the accumulated information content of two independent events x_1 and x_2 with the respective probabilities $P(x_1)$ and $P(x_2)$ should be identical to the information content of the joint event $x_1 \cap x_2$

(x_1 and x_2 are both happening). This is reflected in Eq. (2.66):

$$I\left(x_1 \cap x_2\right) \overset{!}{=} I\left(x_1\right) + I\left(x_2\right) \;,. \tag{2.66}$$

Since the probability for the joint event $x_1 \cap x_2$ is the product of the probabilities of the single events $P(x_1 \cap x_2) = P(x_1)P(x_2)$ one can conclude that $I\left(P(x_1)P(x_2)\right) \overset{!}{=} I\left(P(x_1)\right) + I\left(P(x_2)\right)$. The log of the probability in Eq. (2.67) satisfies the requirement from Eq. (2.66).

$$I\left(x_i\right) \sim \log_b\left(P\left(x_i\right)\right) \tag{2.67}$$

$$I\left(x_i\right) := -\log_b\left(P\left(x_i\right)\right) \tag{2.68}$$

The sure event with $P = 1$ does not hold any information (is no surprise). The more uncertain an event is the bigger is its information content. Since one wants $I \geq 0$ there occurs a minus in the definition of Eq. (2.68).

The expected information content is called entropy H:

$$H(X) = \sum_i P(x_i)I(x_i) = \sum_i P(x_i)\log_b\left(P(x_i)\right) \tag{2.69}$$

$$H(X,Y) = \sum_{ij} P(x_i,y_j)I(x_i,y_j) = \sum_{ij} P(x_i,y_j)\log_b\left(P(x_i,y_j)\right) \tag{2.70}$$

For all actual computations I will use exclusively the logarithm to the basis $b = e$. Therefore the unit of the entropy will be nat. The variation of information can now be defined as the sum of the difference of the joint entropy of both random variables X and Y and the respective solitary entropies:

$$\mathrm{VI}\left(X,Y\right) := \left(H\left(X,Y\right) - H\left(X\right)\right) + \left(H\left(X,Y\right) - H\left(Y\right)\right) \;. \tag{2.71}$$

The mutual information MI of two variables X and Y is

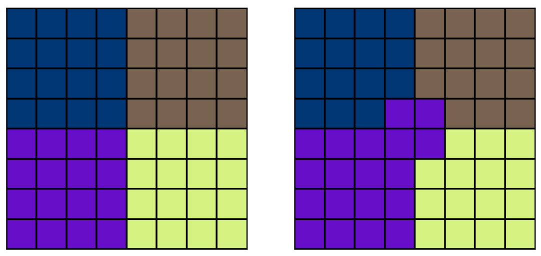
$$MI\left(X,Y\right) := H\left(X\right) + H\left(Y\right) - H\left(X,Y\right) \;. \tag{2.72}$$

Hereby theVI can also be written as

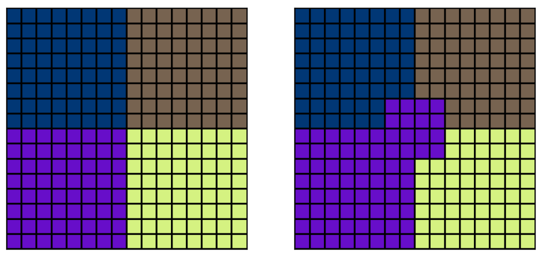
$$\mathrm{VI}\left(X,Y\right) = \left(H\left(X\right) - MI\left(X,Y\right)\right) + \left(H\left(Y\right) - MI\left(X,Y\right)\right) \;. \tag{2.73}$$

The mutual information can be interpreted as the amount of information that one can get from the state of one random variable X about the other variable Y – The more one has to change one to end up with the other, the more diverse they are and the bigger the VI gets. A VI of 0 means perfect correspondence between the states of the random variables. The bigger the value the bigger their deviation.

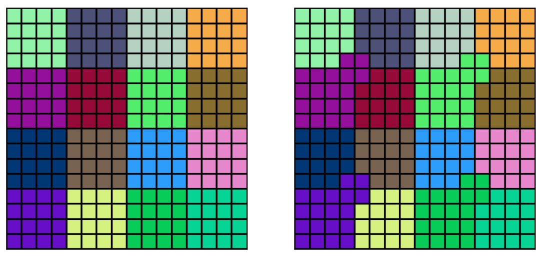
In the case of segmentations, the discrete random variables X and Y correspond to segmentations S_1 and S_2 and the probability of a certain segmentation id x_i or y_i is given by their relative commonness $P(x_i) = \frac{\sum_{p \in I} \delta(S(p) - x_i)}{\sum_{p \in I} 1}$ where the sums are over all pixels in the segmented image.



(a) RI: 0.95238095238095
VI: 0.35632688540993



(b) RI: 0.98823529411764
VI: 0.35632688540993



(c) RI: 0.95294117647058
VI: 0.35632688540993

Figure 2.6: Three exemplary pairs of simple 2D segmentations for the illustration of the different behavior of both used measures. The colorful digits indicate a coincidence with (a).

RI vs. VI

To get a feeling for both used similarity measures I provide three pairs of segmentation images in Figure 2.6 as well as the respective similarity measures. The starting point of this experiment are two 8×8 images shown in 2.6 (a). In (b) the resolution of the same segmentation has doubled. While the variation of information is unimpressed, the Rand index significantly improves. This experiment gives an impression of the measures' behavior in case the segments get bigger. Figure 2.6 (c) is pretending that the object size in (a) is fixed but there are now more of the same size. The variation of information is again completely invariant under this transformation. The Rand index reacts way less extreme than in (b) but still a noticeably increase can be reported.

Both measures cover different aspects in a comparison and therefore the computation ob both of them is valuable. Notice the peculiarity of the Rand index that for an increasing field of view the measure will increase.

2.7 Partitioning vs. Semantic Segmentation

In section 2.3 it is mentioned how relatively simple algorithms can be utilized to do easy merge decisions on the pixel level to end up with an oversegmentation. In 2.4 it is shown how this oversegmentation can lift the computation of merge probabilities to a more sophisticated level – If one knows about a set of pixels that will definitely belong to one neuron the computation of all kind of statistics on the data of the pixels in the set is possible. All following chapters start from this basis.

The presented approaches for the segmentation of neurons so far did build on the detection of the membranes. They are the strongest and most obvious indication available.

Therefore the respective literature (see in chapter 1.3) relies on the membrane information only. In the next three chapters 3, 4 and 5 I work on the segmentation problem following this pure partitioning paradigm.

In the subsequent two chapters 6 and 7 I explore what kind of additional information about the neurons architecture can be utilized in the segmentation procedure. In a flat representation, as introduced in section 2.1.5, the membrane evidence translates into edge-weights of a graph as in sections 2.2.1 and 2.4. All additionally introduced evidence can be seen as characteristics of the nodes. On a first glance, neurons within EM images look locally indistinguishable (see Figure 1.4). For incorporating any node-wise information into the segmentation procedure, one needs to make sure that missing data will be treated properly. More specifically we are facing three facets of sparsity:

1. Within one neuron distinctive node-wise information is sparse.
2. If restricted to a small field of view neurons may not be completely within it[4]. In combination with point 1., one concludes that some individual neurons within the problem do not have distinctive node information at all.
3. Distinguishing individual neurons automatically by their local texture and appearance is not possible from the given images. Therefore in addition to point 1. and 2. there is sparsity in the possibility to distinguish neurons. I will show that even if detecting a fingerprint of single neurons is not possible it is possible to distinguish classes of neurons and deduce constraints based on this.

We will refer to the problem class, matching these requirements, as **partially seeded segmentation**. “Seeded” refers to point 1. where node-wise information per neuron is only available at some places from which it can spread. “Partially” refers to point 2. and point 3.

In the following I will describe two algorithms that are able to include sparse, possibly semantic, unary information: The Asymmetric Multiway Cut, which could be seen as the natural generalization of the Multicut for incorporating unaries and a variant of the Agglomerative Clustering I will refer to as Semantic Agglomerative Clustering.

We will introduce two principally different ways of sparse node-wise information. In chapter 6 I will make use of the relation of synaptic partners that can be encoded within two semantic classes. This means that the Asymmetric Multiway Cut as a globally optimal algorithm is still applicable.

In chapter 7 I will show that the Semantic Agglomerative Clustering is able to greedily resemble the objective of the Asymmetric Multiway Cut from chapter 6 albeit not being able to reach segmentation quality as its globally optimal competitor. This algorithm can show its strengths in cases where additional human input is available. In section 7.2 I show how Semantic Agglomerative Clustering can handle a huge number of semantic classes, the individual neuron identities seeded from man-made neuron skeletons. The runtime of the globally optimal Asymmetric Multiway Cut prevents its usage on this problem.

[4]In fact in all our experiments the scale of the field of view is far to small to cover one neuron completely. Every neuron will leave the block of interest at at least one position.

Chapter 3

Multicut

The Multicut[1] algorithm is designed for clustering nodes in an undirected weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ ($\mathcal{V}$ is the set of all nodes and $\mathcal{E}$ the set of all edges). The edge-weights can both be attractive and repulsive. Besides this, its ability to handle more than two segments differentiates it from the Graph-cut algorithm advanced by Greig et al. [96] and Boykov and Kolmogorov [97] (a more detailed comparison can be found in section 6.2.3).

The problem setup is as follows: Each node itself has no affinity to any special label in the final segmentation/cluster (there are no unaries). Only label transitions are favored/punished. This describes a well known class of problems and has been discussed extensively by Kappes et al. [74, 92] and Bansal et al. [95]. Since it can be seen as an origin of several parts of this work I will introduce the algorithm in the following. For further information the reader is referred to [92].

3.1 Methods

3.1.1 Node Formulation

The most straightforward way to formalize the Multicut problem is in the implicit flat edge representation (described in section 2.1.5) by introducing partition ids $l^p = \left(l^p_1, \dots, l^p_{\|\mathcal{V}\|}\right)$, one for each node i in the graph. Since one has to be able to depict all possible partitions there must be as many ids as there are nodes. The scenario of every node belonging to its own cluster is a valid partitioning.

$$l^p_i \in \{1, \dots, \|\mathcal{V}\|\} \tag{3.1}$$

The superscript p here indicates that the label is a pure partition label meaning that, as usual for the implicit flat representation, the actual id of a node is meaningless – All permutations of node ids are equivalent. That there is no affinity of some nodes to certain ids is reflected in the objective function in Eq. (3.2) adopting a factor graph representation of the problem (see section 2.5). No unary potentials are present. The only available information lies in the pairwise factors encoding attraction/repulsion of

[1]Depending on whether the background of the authors lies more in discrete optimization or in machine learning the exact same problem is called Multicut [74, 92] or correlation clustering [93, 94, 95]. To avoid confusion I will refer to the problem only as Multicut in the following.

neighboring nodes. They contribute in the objective function via the energies E_{ij}:

$$\underset{l^p}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{E}} E_{ij}\left(l_i^p, l_j^p\right) . \tag{3.2}$$

In a general factor graph the pairwise factor $E_{ij}\left(l_i^p, l_j^p\right)$ can attain $\|\mathcal{V}\| \times \|\mathcal{V}\|$ different states. These are numbered in Eq. (3.3) by upper indices while the lower index tuple (i, j) is supposed to remind us that there is one individual energetic contribution per edge.

$$E_{ij}\left(l_k^p, l_l^p\right) = \begin{pmatrix} e_{(i,j)}^{11} & e_{(i,j)}^{12} & \cdots & e_{(i,j)}^{1\|\mathcal{V}\|} \\ e_{(i,j)}^{21} & e_{(i,j)}^{22} & & \\ \vdots & & \ddots & \\ e_{(i,j)}^{\|\mathcal{V}\|1} & & & e_{(i,j)}^{\|\mathcal{V}\|\|\mathcal{V}\|} \end{pmatrix}_{kl} \tag{3.3}$$

$$\overset{*}{=} \begin{pmatrix} a_{(i,j)} & b_{(i,j)} & \cdots & b_{(i,j)} \\ b_{(i,j)} & a_{(i,j)} & & \\ \vdots & & \ddots & \\ b_{(i,j)} & & & a_{(i,j)} \end{pmatrix}_{kl} = \begin{cases} a_{(i,j)} & \text{if } l_k^p = l_l^p \\ b_{(i,j)} & \text{if } l_k^p \neq l_l^p \end{cases} \tag{3.4}$$

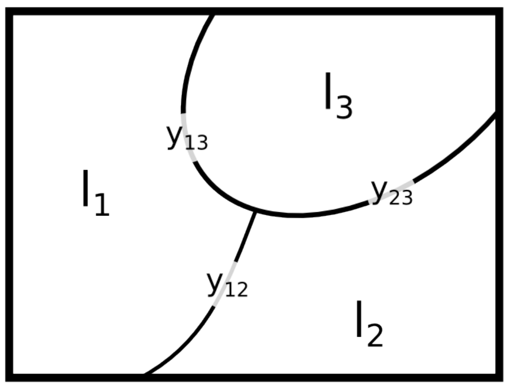
Due to the restriction that we are dealing with partitions the factor matrix simplifies. If $E_{ij}\left(l_k^p, l_l^p\right) \neq E_{ij}\left(l_k^p, l_m^p\right)$ for $k \neq l$, $k \neq m$ and $j \neq k$ would hold then the nodes would not be energetically indifferent to the actual node label. Therefore $\overset{*}{=}$ in Eq. (3.4) holds. The argmin of the objective function in Eq. (3.2) is invariant under a global shift

$$\sum_{(i,j)\in\mathcal{E}} E_{ij}\left(l_i^p, l_j^p\right) \to \left(\sum_{(i,j)\in\mathcal{E}} E_{ij}\left(l_i^p, l_j^p\right)\right) + \alpha = \sum_{(i,j)\in\mathcal{E}} \left(E_{ij}\left(l_i^p, l_j^p\right) + \alpha_{(i,j)}\right) , \tag{3.5}$$

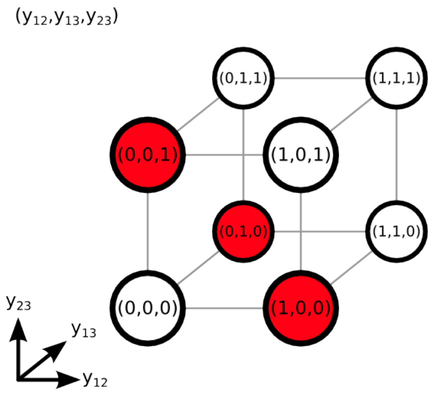
with α being a scalar. It is now easy to see that alpha can be expanded in $\alpha = \sum_{(i,j)\in\mathcal{E}} \alpha_{(i,j)}$ and that these can be commuted in the respective summands of the objective. This gives us the possibility to encode each factor as it is shown in Eq. (3.4) by just one number $c_{(i,j)}$:

$$E_{ij}\left(l_i^p, l_j^p\right) = \begin{cases} 0 & \text{if } l_i^p = l_j^p \\ c_{(i,j)} & \text{if } l_i^p \neq l_j^p \end{cases} , \tag{3.6}$$

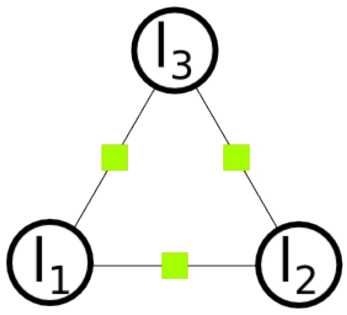
with $c_{(i,j)} = b_{(i,j)} - a_{(i,j)}$. Node i and j are now attracted if $c_{(i,j)} > 0$ and repulsed if $c_{(i,j)} < 0$. Section 2.4.3 discusses how to relate $c_{(i,j)}$ with a learned face-wise membrane probability. Due to the fact that all permutations of the labels are energetically equivalent there are (at least) $n!$ equally good global optimal solutions to Eq. (3.2) where n is the number of used partition labels in the end. This makes the optimization of the problem in this formulation difficult. The energetic degeneracy has been resolved by rewriting the problem in terms of a binary labeling of the edges instead of the partition labeling of the nodes.



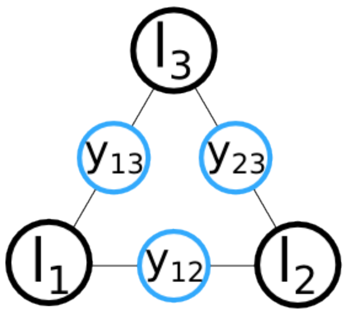
(a) exemplary segmentation



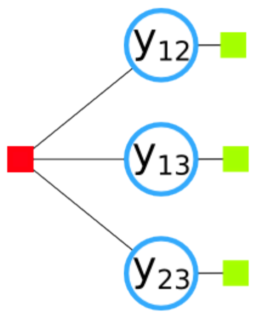
(b) forbidden edge configurations (red)



(c) node representation



(d) node and edge variables



(e) edge representation

Figure 3.1: Illustration of the two equivalent formulations of the Multicut. Given an exemplary segmentation (a) where there is one label l_i per region (represented by ○ in (c) and (d)) and one binary indicator per face (indicated by ○ in (d) and (e)). In (c) the Multicut in the node representation is visualized. (e) shows the respective edge representation. The pairwise factors in (c) 🟩 translate to unaries in (e). The price one has to pay is the necessity of one global factor 🟥 assuring the consistancy of the edgelabels. In (b) the invalid configurations, prevented by 🟥 are marked in red. The relationship between the l_i and the y_{ij} is made explicit in (d).

3.1.2 Edge Formulation

In the case of Eq. (3.6) only the information $l_i^p \overset{?}{=} l_j^p$ is energetically relevant. This binary information can be encoded in the binary state of the edge $(i,j) \in \mathcal{E}$:

$$y_{ij} := \begin{cases} 0 & \text{if } l_i^p = l_j^p \\ 1 & \text{if } l_i^p \neq l_j^p \end{cases} \tag{3.7}$$

An edge (i,j) with $y_{ij} = 1$ is called a cut edge. The intrinsic degeneracy existent in Eq. (3.2) is avoided. Note that there can still be some degeneracy depending on the used weights c_{ij} from Eq. (3.6). This remaining degeneracy, which is present in the objective function in Eq. (3.2) in addition to the one caused by permutation invariance, is not artificial but inherent to the respective problem instance. There can be different partitions of the graph that are energetically equal.

The space of possible binary edge labels covers all possible partitions. This can be seen since each partition given by a node labeling can be transfered to an edge labeling. But the space of binary edge labels includes also configurations that are not transferable to a consistent labeling of the nodes in the sense of Eq. (3.1). Figure 3.2 gives an example. Nodes that are supposed to have a different partition label due to a separating edge between them can be connected by a chain of other edges that form a connection and therefore imply an corresponding partition label; a contradiction. Therefore the price to pay for the unambiguous representation in Eq. (3.7) is that one has to explicitly constrain the solutions to be within the subspace of consistent solutions. As it can be seen in Figure 3.1 (b) for a three dimensional example, all solutions lie on the edges of a hypercube. If we now want to restrict the solution-space we can do this by introducing linear constraints.

These can be represented as hyperfaces in the solution-space dividing the space into an accessible and a forbidden halfspace. For example a constraint involving all variables is cutting away one of the corners of the hypercube defined by the possible solutions. As I will show in chapter 5 not all of the necessary constraints are of this kind. If a constraint is only concerning a subset of all variables $\mathbf{y}_c \subset \mathbf{y}$ the respective hyperplane will be parallel to all variables $\mathbf{y} \backslash \mathbf{y}_c$. If a hyperplane representing a constraint is parallel to a certain dimension representing a variable, the state of this variable does not has any affect on violation of the constraint. This can be exploited to decompose the problem in subproblems that can be solved independently of each other.

A class of constraints that is sufficient to completely define the valid subspace of the solution space, the Multicut polytope are the cycle constraints. Both Kappes et al. [92] and Kim et al. [93] are using them exactly as described here. Andres et al. [98] are not considering the clustering of general graphs but is looking closely on the implications of the cycle constraints in image segmentation. Others, like Nowozin and Jegelka [99], use the constraints to tighten the Multicut polytope relaxation for an linear program.

The actual constraints are easy to understand: Within all edges of all cyclic paths the situation of one single cut edge is forbidden. More formal:

$$\sum_{(i,j)\in C} \mathbf{y}_{ij} \neq 1 \quad ; \quad \forall C \in \text{Cycle}(u,v) \subseteq \mathcal{E}\,. \tag{3.8}$$

The slight reformulation in terms of paths, that are basically cycles with one missing edge, facilitates the formulation of the constraints as inequalities.

$$\text{Path}(u, v) := \text{Cycle}(u, v) \setminus (u, v) \tag{3.9}$$

This allows a formulation of the given problem as an integer linear program.

$$\min_{\mathbf{y}} \left(\sum_i w_i y_i \right) \tag{3.10}$$

$$\text{s.t.} \quad \sum_{(i,j) \in P} \mathbf{y}_{ij} \geq \mathbf{y}_{uv} \;\; \forall (u, v) \in \mathcal{E};\; \forall P \in \text{Path}(u, v) \subseteq \mathcal{E} \tag{3.11}$$

Now all kinds of solvers and techniques from this well examined problem class can be applied. The number of all possible paths P for all neighboring nodes u and v is growing fast with the number of edges in the graph. Enforcing all of them would not be sensible.

Depending on the actual graph structure and the actual weights not all possible constraints are needed. The problem of finding actually violated and therefore relevant constraints is called the separation problem. Iteratively finding violated constraints, adding them to the originally plain objective in Eq. (3.10) and resolving it will necessarily result in a state where none of the possible constraints from Eq. (3.11) is violated. This state is a globally optimal consistent solution. This approach is called a cutting planes approach since more and more parts of the originally untouched hypercube of viable solutions are cut away by hyperplanes representing linear inequalities.

3.2 Practical Implementation Details

3.2.1 Facet Defining Constraints

As explained by Haas and Hoffmann [100], on a chordless path P_c in a graph $\mathcal{G}$ none of the nodes within the path are connected via an edge without the path. With the definition of a vertex-induced subgraph being a subset of all vertices $\mathcal{V}_s$ together with the subset of edges $\mathcal{E}_s$ whose endpoints are both in the subset of vertices: $((u, v) \in \mathcal{E}_s) \Leftrightarrow (u \in \mathcal{V}_s \text{ and } v \in \mathcal{V}_s)$, a chordless path can also be described as follows: If the vertex-induced subgraph of the set of nodes in a path P is the path itself, P is chordless.

Chopra and Rao [101] show that a chordless path corresponds to a facet defining constraint. The hyperplanes corresponding to a facet defining constraint do coincide with facets of the respective polytope. This means that during the cutting planes steps only chordless paths must be considered/only facet defining constraints must be added to the set of violated constraints. This insight will lead to a stronger localization of the constraints which lead to a more pronounced decoupling as described in section 3.3.

3.2.2 LP Relaxation

Solving a general integer linear program is NP-hard. If the problem at hand has sufficiently well behaved weights w_i only a small subset of all possible constraints are needed and problems of relevant size are still tractable. This is often times the case when applying the Multicut on segmentation tasks in image analysis (e.g. by Andres et al. [98, 102]).

There the structure given by the neighborhood relations of the used superpixles and the favorable weights are advantageous.

Nevertheless one might want to relax the problem in the sense that $y_{ij} = \{0,1\} \rightarrow y_{ij} = [0,1]$. The resulting problem is solvable in polynomial time and therefore will circumvent the troubles that come with the worst case complexity of the integer linear program (LP). If one does not rely on the exact globally optimal solution one could solve the relaxed linear program and round the solutions in the end (the rounding, a projection to the Multicut polytope to ensure consistency is itself a nontrivial step. Further details on this topic can be found in the work of Kappes et al. [92]). An alternative usecase for the relaxation is as provider of a warm start for a subsequent integer linear program.

The total runtime of the Multicut algorithm is determined by the optimization step as well as the seperation problem where the violated constraints are found. Note that the shortest path algorithm on a graph with binary edges is of kinder complexity than the shortest path on a graph with real-valued edge-weights w_i as described in Eq. (3.10). Therefore, especially for small problem sizes it is hard to predict whether the overall runtime of a LP-relaxation of the Multicut or its integer formulation will require more computational time. If one calls on the LP-relaxation of the Multicut one can think of adding additional types of constraints like the odd-wheel constraints as discribed by Kappes et al. [92] to tighten the relaxation.

3.2.3 Optimization

The actual optimization problem is to solve the occurring integer linear programs with the so far found violated constraints. These minimization steps in the iterative procedure are best left for a specialized commercial library like CPLEX [103]. They utilize a lot of heuristics that can make a huge difference when working on discrete combinatorial optimization problems.

3.2.4 The Need for Superpixels – MC for Closing Gaps

If one gives spatial structure to the graph by localizing the nodes in an image (as done e.g. by Andres et al. [98, 102]) the clustering of the graph corresponds to a segmentation of the image. The naive localization by assigning one node per pixel is not expedient though in the most cases. It is true that the Multicut enforces closed surfaces and therefore is suited to close a corrupted membrane[2]. The membrane is extended in the image domain. Therefore it must be assured that the final segmentation does not contain the membrane as an segment of its own. We need to make use of superpixels that are not containing membranes but do have membranes only at their boundary. In such a setup the closeness constraints for the segmentation actually translates to closeness constraints of the objects bounded by the membranes.
Figure 3.2 shows a disrupted membrane on the pixel level. Applying the Multicut algorithm on the pixel level can result in a consistent segmentation that still represents a disrupted membrane. If one restricts the possible edges by using appropriate superpixels a situation representing a discontinuous membrane is not possible. Either the Multicut

[2]In this paragraph we will talk about edges in the image, represented by pixels and edges in the graph. To avoid confusion and with the application of the segmentation of neurons in mind we will call the first type membranes and the second type edges in the following.

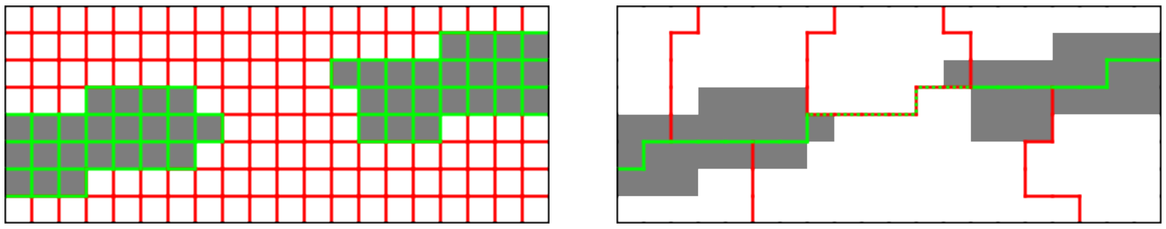


Figure 3.2: Gray indicates a membrane. Both images show an interrupted boundary of an object. The left image shows a pixel oversegmentation while the right image already has some pixels clustered together in a superpixel oversegmentation. A green line locally indicates a transition between segments and a red line indicates a shared segment id. Only in the right case the Multicut has a change of closing the hole in the membrane by activation of the striped line.

will close the hole in the membrane or it will completely get rid of the open ends at all (depending on the weights).

The said is not true for the naive application of the Multicut algorithm on the pixel level. There are different ideas how to do this: Keuper et al. [104] use long-range potentials to make sure that regions with boundary evidence between are separated even if this boundary evidence is slightly disrupted. They call this approach the Lifted Multicut.

A different direction is explored in chapter 3.3. The introduction of constraints as shortest paths is possible in the original framework since the topology of the objects representing edges is clearly defined. One can easily talk about open ends and one can decide on which side of the membrane the superpixels are located. I will show how one can introduce a notion of topology not on an interpixel level but on actually space-filling representations of membranes.

3.3 Dynamical Decomposition of Multicuts

In the following section I will introduce a way to exploit the structure of the optimization problem, specified by the localization of the nodes in the image domain, to decompose it. Global optimal results can be guaranteed while the optimization time is significantly reduced.

3.3.1 Methods

The Multicut algorithm provides a globally optimal solution with respect to the edgeweights w_i from Eq. (3.10). This means that a change in the unary potential of any variable can influence the optimal state of any other variable in the problem. In Figure 3.1 this is depicted by the fact that the factor representing the Multicut-constraints is connected with all variables y_{ij} in the problem. I will show in the following that this global nature is not fully exploited in cases of 3D neuron segmentation tasks. Therefore a decomposition into independently solvable parts is possible in practice without giving up global optimality. The global factor can then be replaced with a set of localized ones of a much lower order. As I will show, this decomposition is done dynamically during the cutting planes approach.

In the beginning of this chapter 3 it is already stated that the Multicut problem can be optimized utilizing a cutting planes approach where iteratively only the violated facet defining cycle constraints are added. In the very first iteration, where no constraints have been added yet, each variable is solved completely independent of the others. From Eq. (3.10) it becomes apparent that a mere thresholding will suffice in the unconstrained setting:

$$y_{ij} \in \{0,1\};\ w_{ij} \in \mathbb{R};\ \Theta(x) := \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{3.12}$$

$$\min_{\mathbf{y}} \left(\sum_{(i,j)\in\mathcal{E}} w_{ij} y_{ij} \right) = \sum_{(i,j)\in\mathcal{E}} \min_{y_{ij}} (w_{ij} y_{ij}) \Rightarrow \hat{y}_{ij} = \Theta(w_{ij}) \tag{3.13}$$

Here $\hat{y}$ denotes the optimal solution. Note that the behavior of the Heaviside like function $\Theta(x)$ at $x = 0$ is not relevant in this case and can therefore be arbitrarily be declared to be either one or zero.

The iterations of the cutting planes approach are numbered with t and the initial thresholding step is defined to be $t = 0$. In each subsequent iteration t we find violated constraints that we add to a set of so far found violated constraints $\mathcal{C}^t_{\text{vsf}}$. Since none of the found constraints is removed (this could lead to endless circles) $\mathcal{C}^t_{\text{vsf}} \subset \mathcal{C}^{t+1}_{\text{vsf}}$ holds.

$$C_m := \sum_{(i,j)\in P_m} \mathbf{y}_{ij} \overset{!}{\geq} \mathbf{y}_{uv} \ ;\ \ (u,v) \in \mathcal{E},\ P_m \in \text{Path}(u,v) \subseteq \mathcal{E} \tag{3.14}$$

is one constraint as used in Eq. (3.11). The set of contained edges in C_m is defined as

$$\mathcal{C}^{\mathcal{E}}_m := \{(i,j);\ (i,j) \in P_m\}\ . \tag{3.15}$$

The integer linear program that needs to be solved in an iteration t reads then:

$$\min_{\mathbf{y}} \left(\sum_i w_i y_i \right) \quad \text{s.t.}\ \ C_m \ \ \forall C_m \in \mathcal{C}^t_{\text{vsf}}\,. \tag{3.16}$$

Let us approach the nature of the decoupling of this problem in several steps:

1. First, we want to examine the case of solving the problem with only one constraint present ($\|\mathcal{C}_{\text{vsf}}\| = 1$). All variables y_{ij} with $(i,j) \in \mathcal{C}^{\mathcal{E}}_m$ can not be solved independently any more. The state of one of them can be altered if another one of them changes its state. Every y_{ij} with $(i,j) \notin \mathcal{C}^{\mathcal{E}}_m$ can still be solved by thresholding only.

2. Now we will examine what happens in the presence of multiple constraints because most of the time one cycle constraint will not be sufficient to cover all inconsistent solutions that are energetically better than the energetically best solution of the fully constrained problem.
Through the introduction of multiple constraints, variables y_{ij} and y_{kl}, for which no single constraint C_m can be found such that $(i,j) \in \mathcal{C}^{\mathcal{E}}_m$ and $(k,l) \in \mathcal{C}^{\mathcal{E}}_m$, can be coupled (can not be solved independently any more as in iteration $t = 0$). If constraints share variables, all variables in the union of these constraints (all connected

components $\mathfrak{C}$s) must be considered together.

$$\text{connected}(C_m, C_n) = \text{connected}(C_n, C_m) := \begin{cases} \text{False} & \text{if} \quad \mathcal{C}_m^{\mathcal{E}} \cap \mathcal{C}_n^{\mathcal{E}} = \varnothing \\ \text{True} & \text{otherwise} \end{cases} \tag{3.17}$$

$$\mathfrak{C}_a^{\mathcal{E}} \subset \mathcal{E}$$

$$\begin{aligned} & (i,j) \in \mathfrak{C}_a^{\mathcal{E}};\ (k,l) \in \mathfrak{C}_a^{\mathcal{E}} \\ \Leftrightarrow\ & \exists \{C_k\}_{k=1}^{N} \subset C_{\text{vsf}} \\ & \text{s.t. } (i,j) \in \mathcal{C}_1^{\mathcal{E}},\ (k,l) \in \mathcal{C}_N^{\mathcal{E}} \\ & \text{and connected}(C_a, C_{a+1})\ \forall a \in \{1, 2, \ldots, N-1\} \end{aligned}$$

Such a strict definition of $\mathfrak{C}$ is bulky but the idea is simple: Constraints connect each other via shared edges. $\mathfrak{C}_a^{\mathcal{E}}$ will be the set of all edges that are covered by any constraint in $\mathfrak{C}_a$.

3. As a next step we will emphasize that not all of the variables associated in edge-sets $\mathfrak{C}^{\mathcal{E}}$s do need to be recomputed in each iteration t. If one defines edges that are not part of any constraint as not being their own $\mathfrak{C}$, in the zeroth iteration (thresholding phase without any constraints) the number of connected components is zero: $\#(\mathfrak{C})_0 = 0$. In the next iteration there might occur $\#(\mathfrak{C})_1$ connected components. This is the maximal number of $\mathfrak{C}$s one will find during the whole optimization process: $\#(\mathfrak{C})_t \geq \#(\mathfrak{C})_{t+1}\ \forall t \geq 1$. Every constraint that is introduced in a later iteration $t > 1$ must share at least one variable with a constraint that has already been present in iteration $t-1$. Otherwise it could have been already be found in iteration $t = 1$. This is the case because we can only expect a change in the intermediate solution $\mathbf{y}_t$ to the thresholding solution $\mathbf{y}_0$ at places where constraints are present. The reduction $\#(\mathfrak{C})$ follows from a merging of existing $\mathfrak{C}$s e.g. by the introduction of a new constraint that covers edges in both $\mathfrak{C}$s.
The interesting part for the optimization is now that, if a $\mathfrak{C}$ is solved in t and for $t+1$ no new constraints are found that do overlap with at least one edge in this $\mathfrak{C}$, the sub-solution for the edges in $\mathfrak{C}$ will not change – we already found the optimal solution given this set of constraints. Therefore it does not need to be recomputed.

$$\begin{aligned} & \min_{\mathbf{y}} \left(\sum_i w_i y_i \right) \text{ s.t. } C_m\ \forall C_m \in \mathcal{C}_{\text{vsf}} \qquad (3.18) \\ = & \sum_a \left(\min_{\mathbf{y}} \sum_{(i,j) \in \mathfrak{C}_a^{\mathcal{E}}} w_{ij} y_{ij} \right) + \left(\min_{\mathbf{y}} \sum_{(i,j) \notin \mathfrak{C}_a^{\mathcal{E}} \forall a} w_{ij} y_{ij} \right) \text{ s.t. } C_m\ \forall C_m \in \mathcal{C}_{\text{vsf}} \end{aligned}$$

The intermediate optimal state for the edge (i,j) at the iteration t shall be called $\hat{y}_{ij}^t$. It can now be computed according to

$$\hat{y}_{ij}^{t} = \begin{cases} \left(\underset{\mathbf{y}_a}{\operatorname{argmin}} \left(\sum_{(i,j)\in\mathbb{C}_a^{\mathcal{E}}} w_{ij} y_{ij}\right)\right)_{ij} \text{ s.t. } C_m \ \forall C_m \in \mathbb{C}_a^t & \text{if } (i,j) \in \mathbb{C}_a^t \\ & \text{and } \mathbb{C}_a^t \neq \mathbb{C}_a^{t-1} \\ \left(\underset{\mathbf{y}_a}{\operatorname{argmin}} \left(\sum_{(i,j)\in\mathbb{C}_a^{\mathcal{E}}} w_{ij} y_{ij}\right)\right)_{ij} \text{ s.t. } C_m \ \forall C_m \in \mathbb{C}_a^{t-1} & \text{if } (i,j) \in \mathbb{C}_a^t \\ & \text{and } \mathbb{C}_a^t = \mathbb{C}_a^{t-1} \\ \Theta\left(w_{ij}\right) & \text{if } (i,j) \notin \mathbb{C}_a \ \forall a \ . \end{cases} \tag{3.19}$$

If no additional violated constraints can be found in iteration $\hat{t}$ then $\hat{y}_{ij}^{\hat{t}} = \hat{y}_{ij}^{\hat{t}+1}$ and $\hat{y}_{ij}^{\hat{t}}$ is the globally optimal solution leading to a closed surface segmentation. Note that only for the first of the three possibilities for the computation of $\hat{y}_{ij}^{t}$ it is necessary to actually solve an integer linear problem. The second line utilizes the solution already found in the previous iteration and the third one copies the stage zero solution, the mere thresholding.

To summarize: I presented a scheme how to dynamically divide the Multicut problem into several small subproblems without loosing global optimality. One can see that each step of the cutting plane iterations compulsory ends up with the same intermediate solution no matter if the problem was solved globally or for each connected component independently and then composed of the sub-solutions (if there exists an unique optimal solution). Note that the search for constraints, via e.g. a shortest path algorithm, still has to be performed on the whole problem.

The Hyperplane Perspective

Since the decoupling is constructed solely on the affiliation of constraints to variables we can find a geometrical interpretation of the decoupling in the space of all solutions mentioned in chapter 3. There the constraints correspond to hyperplanes, each defining a invalid half space. Let H_m be the hyperplane associated with constraint C_m.

$$(i,j) \notin \mathcal{C}_m^{\mathcal{E}} \Rightarrow H_m \parallel y_{ij} \tag{3.20}$$

$$(i,j) \in \mathcal{C}_m^{\mathcal{E}} \Rightarrow H_m \nparallel y_{ij} \tag{3.21}$$

$H_m \parallel y_{ij}$ means that the hyperplane is parallel to the dimension of the solution-space to which y_{ij} assigned. The definition in Eq. (3.17) of two constraints being connected means geometrically that the two constraints have no dimensions in common, in which they are not parallel:

$$\text{connected}\left(C_m, C_n\right) = \text{True} \ \Leftrightarrow \ \exists y_{ij} \text{ with } H_m \nparallel y_{ij} \text{ and } H_n \nparallel y_{ij} \tag{3.22}$$

An simplified visualization can be found in Figure 3.3, showing different sets of constraints in a two dimensional subspace of a Multicut problem.

3.3.2 Experiments

For solving Multicut problems to global optimality I rely on the opengm library presented by Andres et al. [105]. This library automatically translates the problem from a formulation as a graphical model (see 3.1.1) into an integer linear program that will

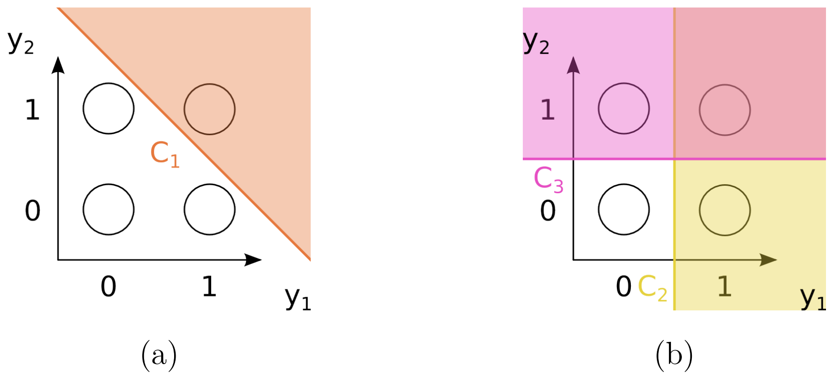


Figure 3.3: Exemplary visualization of constraints in a 2D subspace in the solution space. Each circle symbolizes a possible solution. Only circles not lying within a colored (constrained) half-space can correspond to closed surface solutions. $C_1 \nparallel y_1$, $C_1 \nparallel y_2$; $C_2 \nparallel y_1$, $C_2 \parallel y_2$; $C_3 \parallel y_1$, $C_3 \nparallel y_2$. The constraints C_2 and C_3 in (b) do not form one connected component on their own. C_1 in (a) on the other hand couples the variables y_1 and y_2.

be solved via the commercial solver CPLEX [103]. Since the actual source code of the solver is not openly availible, theoretical predictions on the benefit of the decomposition of the Multicut are hard. We therefore rely on timing experiments on a typical superpixel oversegmentation on the dataset described in appendix A.1.3. The results can be seen in Figure 3.4. It is notable that in all cases where there is an unique globally optimal solution with respect to the current set of constraints, both tested approaches lead to completely identical results $\hat{\mathbf{y}}^t$ in all intermediate stages as well as in the final iteration.

The presented results are reported for a serial computation of all connected components for constraints in the decomposed approach for comparison. Since all connected components are independent of each other, a parallelization of the decomposed approach is trivially possible and may, depending on the used hardware, lead to further improvements in runtime.

3.3.3 Conclusion

There exist fast approximate solvers for the Multicut problem like the ones presented by Beier et al. [106, 107]. Nevertheless in cases where global optimality must be assured or in cases where even small improvements in the resulting segmentation quality counts[3], adopting the presented decomposition is worthwhile.

The advantages in runtime of a blocked problem have already been reported by Kröger [109]. There one has to give up any guarantees for global optimality. In a sense the decomposition scheme proposed in this section can be seen as a continuation in the spirit of [109]. The advantage is that the blocking is not done on the basis of geometrical considerations but the particular problem itself specifies the blocking.

This chapter provides the means to scale up the multucut algorithm. Assuming that there is a maximal size the constraint-clusters will not exceed in practice and that there is a maximal time that could be spend for such a maximal cluster this would transform the NP-hard Multicut problem into a problem scaling linearly with the problem size.

[3]Note that the globally optimal solution is not necessarily the best one according to global error measures as e.g. presented in section 2.6 since we are not optimizing for it but for a simpler, hopefully related energy. One designs the energy function in a way that desired results are probable. As shown e.g. by Batra et al. [108] modes that are not globally optimal might also be of interest.

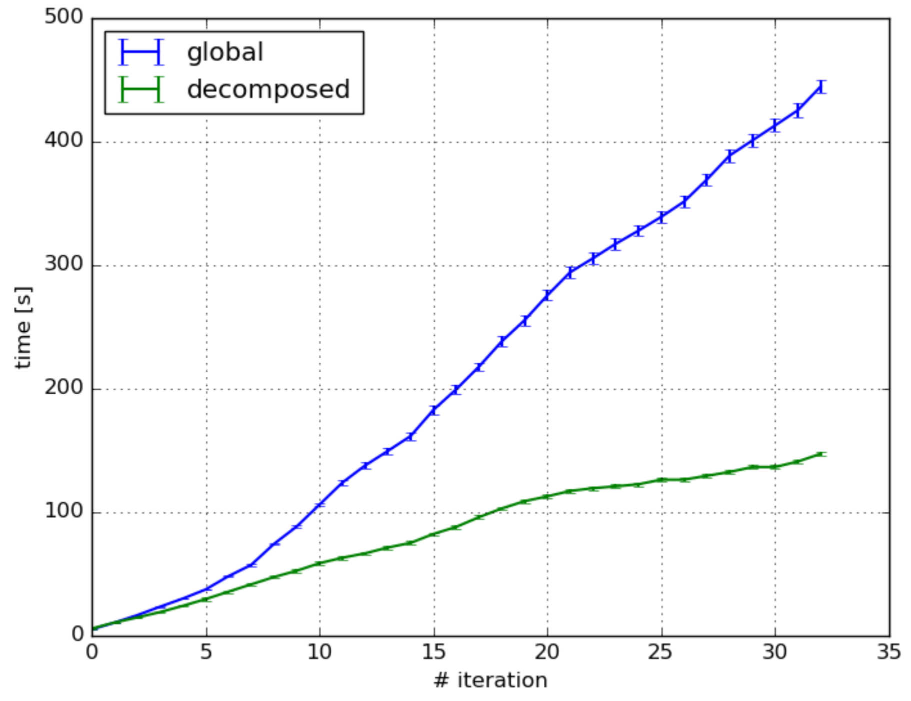


Figure 3.4: Comparison of the pure accumulated ILP-runtime (the costs for the search of the path constraints are not included since they are identical in both approaches) per iteration of the problem. After 32 iterations the globally optimal solution is found by both the global approach from section 3.1.2 and by the decomposed procedure of solution proposed in this chapter. The final mean runtime of the global approach is 444.7 s whereas the decomposed approach takes 147.3 s which makes up for a factor of 3 on this typical problem for neuron segmentation described in appendix A.1.3.

The existence of a maximal size of constraint-clusters for neural data independent of the problem size is questionable though. Therefore future work should focus on the further analysis of the behavior (e.g. the size distribution) of the constraint areas for increasing problem sizes.

Chapter 4

Agglomerative Clustering

Agglomerative Clustering can in some ways be seen as the greedy cousin of the Multicut for image segmentation. While the Multicut aims to find globally optimal solutions given the edge-weights (w_i from Eq. (3.10)), Agglomerative Clustering is a stepwise approach. To base the decision about the inclusion of semantic labels in the hierarchical clustering in chapter 7 (to e.g. emulate the Asymmetric Cuts from chapter 6) on solid ground, the procedure is described in the following section. A fast initial entry to the algorithm is granted by the description via Markov Decision Processes as it was done by Jain et al. [110]. In addition, I will propose a novel, natural stopping criterion for clusterings according to non probabilistic and potentially dynamically changing sequencing-weights.

4.1 Agglomerative Clustering as Markov Decision Processes

A stochastic Markov decision progress as described by Bellman [111] is a tuple (S, A, $P(\cdot,\cdot)$, $R(\cdot,\cdot)$, γ) consisting of a finite set of states S, a finite set of interactions A, transition probabilities P, immediate reward R and a discount factor γ. An action $a_t \in A$ applied in time step t to a particular state $s_t \in S$ will result in a certain other state $s_{t+1} \in S$ in the subsequent time step with the probability $P_{a_t}(s_t, s_{t+1})$. $R_{a_t}(s_t, s_{t+1})$ will be the expected immediate reward for the action a_t.

The goal is now to find a deterministic decision policy $\pi(s_t) = a_t$ that maximizes the expected discounted sum of rewards:

$$\pi(s_t) := \underset{a}{\operatorname{argmax}} \left\{ \sum_{s_{t+1}} P_a(s_t, s_{t+1}) \left(R_{a_t}(s_t, s_{t+1}) + \gamma V(s_{t+1}) \right) \right\} \tag{4.1}$$

$$V(s_t) := \sum_{s_{t+1}} P_a(s_t, s_{t+1}) \left(R_{a_t}(s_t, s_{t+1}) + \gamma V(s_{t+1}) \right) . \tag{4.2}$$

The bigger the so called discount factor $\gamma \in [0, 1]$, the bigger the influence of future actions on the current decision a_t.

As it is shown by Jain et al. [110] one can formulate Agglomerative Clustering as as Markov decision progress. The states s_t at a certain time step t correspond to the segmentation produced by the iterative merging of the initial segments of the oversegmentation up to t. We do not deal with any uncertainties concerning the merge operations $P_{a_t}(s_t, s_{t+1}) \in \{0, 1\}$. Therefore we deal with a deterministic Markov decision process.

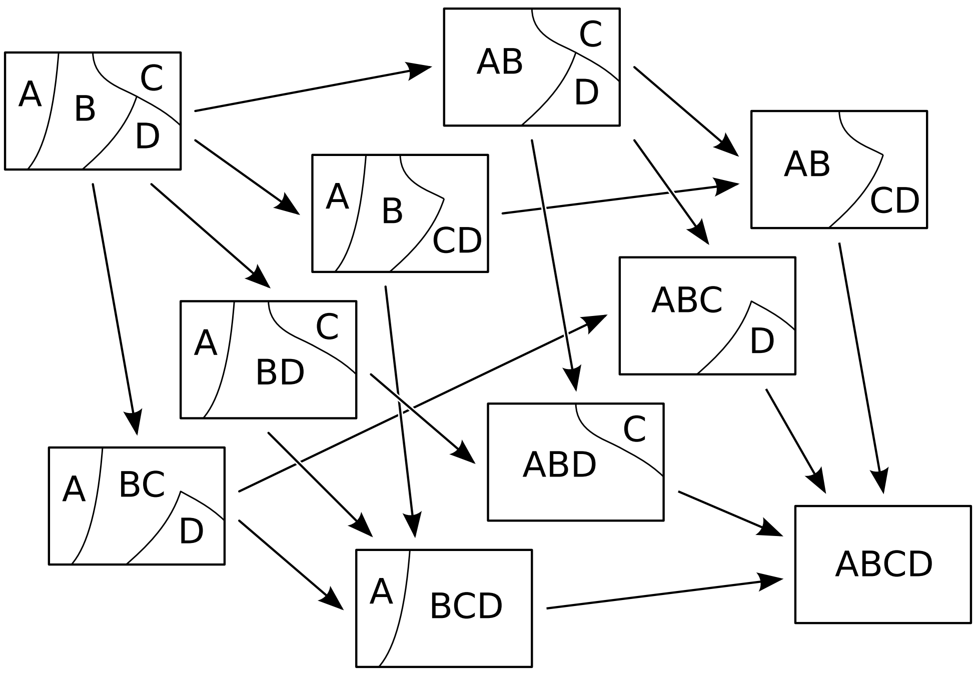


Figure 4.1: All allowed ways of iteratively clustering the four initial segments A, B, C and D in the given exemplary segmentation image illustrated via a Hasse diagram (The fineness of the partitions defines a partial order).

In the straightforward implementation of the Agglomerative Clustering the actions (the merge decisions) do not consider future actions. In Eq. (4.1) the restriction of $\gamma = 0$ ensures that only the summand concerning the current state is considered. The optimal action value function is then equivalent with the immediate reward.

The states S, the merge operations A and the discount factor are now specified. Also the transition probabilities are known.

$$P = \begin{cases} 1 & \text{if merged segments are neighbours} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

This is not to be confused with the merge probabilities that can contribute to the weights w_i. Eq. (4.3) states that if the algorithms decide for one possible action it will definitely succeed.

The most desirable reward is directly related to the error measure (Er) used to judge the resulting segmentation s_t in the end given a ground truth s_{gt} e.g. $R_a\left(s_t, s_{t+1}\right) = \text{Er}\left(s_t, s_{\text{gt}}\right) - \text{Er}\left(s_{t+1}, s_{\text{gt}}\right)$. Here it is assumed that a lower value in Er is better. The ground truth is not available in practice. Therefore we rely on local merge probabilities. The policy π is then: merge the pair of neighboring segments with the highest merge probability. Figure 4.1 illustrates the different ways an agglomeration can take in a simple example, depending on the respective weights. New pairs of segments emerge through the merge process. The simplest way to attain their weight is to average the weight of the composing edges.

There are several possibilities to improve the explained agglomeration procedure. One can adjust the merge policy π. Prior knowledge on e.g. the size distribution of the final segments can enter here. Any improvement on the local merge probabilities will enter in π. Nunez-Iglesias et al. [84] propose a framework to re-learn a classifier for edge-wise merge probabilities and thereby increase the quality of a re-prediction of bigger intermediate edges. I will compare this approach with the Multicut as well as with Asymmetric Cuts in section 6.4.

Another possibility to improve on the vanilla Agglomerative Clustering is to abolish the restriction of $\gamma = 0$. Parag et al. [70] for example take possible future merge decisions into account when reasoning on the present actions. This way, some wrong merge decisions can be foreseen and avoided.

Edgeweights vs. Nodeweights

For actual segmentation procedures via Agglomerative Clustering it is beneficial to utilize superpixels as initial oversegmentation (for reasons stated by Parag et al. [70]). The nodes $\mathcal{V}$ in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ represent the superpixel regions. Each merge action of two neighboring nodes a and b changes the graph in the following way:

$$\begin{aligned} \text{merge}(a,b) : \mathcal{V} &\to (\mathcal{V} \setminus \{a,b\}) \cup c \text{ with } c := \{a\} \cup \{b\} && (4.4) \\ \mathcal{E} &\to \mathcal{E} \setminus (((i,j)\, \forall j \in n_h(i))\, \forall i \in \{a,b\}) && (4.5) \\ &\quad \cup (c,i)\, \forall i \in (n_h(a) \cup n_h(b) \setminus \{a,b\}) \\ &\quad \text{with } c := \{a\} \cup \{b\} \end{aligned}$$

Mostly n_h is defined via the 4- or the 6- neighborhood (see section 2.2.1).
As stated before in section 4.1, at any point in the iteration the edge with the highest merge weight w_{ij} is merged next.

Although the edges $\mathcal{E}$ cover all pairs of neighboring nodes, there is a fundamental difference between the computation of weights w_i on the nodes with a subsequent mapping to the edges $w_{ij} = d(w_i, w_j)$ due to some metric d and a direct assignment of edge-weights w_{ij}: The direct assignment is not restricted by the triangle inequality. Edge indicators based on ridges, where both sides of the ridge are equivalent are better assigned directly. Step edges based on a change in color/intensity of an image can be described via the metric approach.

4.2 Using Intermediate Stages

The disadvantages in terms of final segmentation quality of an iterative approach in comparison to the globally optimal Multicut (see chapter 3) are apparent. Parag et al. [70] show that taking some possible future merges into account is beneficial for the final segmentation. The Multicut is able to take all possible merge-decisions into account at the same time.

There are also some aspects in which the Agglomerative Clustering is in a better position. As seen in section 2.4, the classification of bigger faces tends to be easier than the classification of smaller faces. In intermediate steps the Agglomerative Clustering is producing bigger sized edges and thereby has the possibility to increase the intermediate edge-weight quality while the Multicut has to rely on the initial weights.

As I will show in 6.4, this advantage does not outweigh its shortcomings in terms of quality. The big advantage of this algorithm lies in its computational speed. While in the worst case the Multicut is NP-hard, there exist algorithms that are applicable for the Agglomerative Clustering problem of complexity $\mathcal{O}(n^2)$ like the one presented by Sibson [112].

4.2.1 Dynamically Alter Merge Priority - Size Regularization

The Multicut can not consider any intermediate states. It finds the globally optimal solution given some initial edge-weights. As described, Agglomerative Clustering is a greedy strategy. In general this means that segmentation results do not as good as the ones produced by the Multicut algorithm (see Figure 4.2). The fact that intermediate states exist enables the usage of these stages. The weights of newly emerging edges can be updated more evolved (as for example done by Nunez-Iglesias et al. [84]) or other properties of an intermediate solution can be considered.

The question is what kind of information can be utilized that is not be utilized in the first place. In the case of neuron segmentation, where the shape of the cells strongly varies, not much can be said in advance. In the spirit of the method revisited by Murtagh et. al. [113] the sizes of the segments are considered here in the following way:

$$w_{ij} \to w_{ij} \frac{\left(\frac{1}{\mathrm{size}_i^\theta} + \frac{1}{\mathrm{size}_j^\theta}\right)}{2}\,, \tag{4.6}$$

with a real, non-negative parameter θ. The bigger θ, the bigger the influence of the sizes become in comparison to the original weights. For $\theta = 0$ the weights stay unchanged. Applying this regularization basically encourages small segments to merge. Eventually this leads to a more balanced size distribution of the resulting segments. Eq. (4.6) encodes a size prior for the segments which is biologically sensible – Neurons within the field of view will be of a certain minimum size.

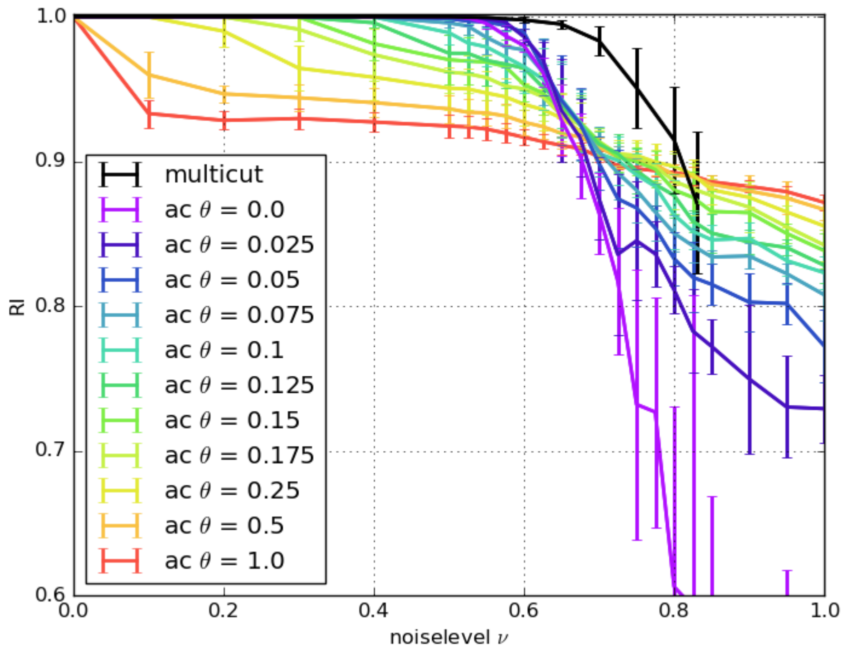
This is a heuristic that eventually is only backed by experimental success. The success of both the Multicut and the Agglomerative Clustering depends strongly on the quality of th edge-weights w_{ij}. In cases where thresholding leads to the correct result, the Multicut algorithm, as introduced in chapter 3, will not find any violated constraints in the very first iteration and therefore terminate immediately. Also the straightforward Agglomerative Clustering algorithm leads to perfect results for thresholdable weights.

The behavior of both algorithms is interesting for increasingly bad edgeweights. The following experiment is supposed to give an impression of the capability of both algorithms for a variety of edgeweights of different quality on a typical neighborhood structure as it appears in the case of neuron segmentation. On a block of data for which a ground truth segmentation is available (appendix A.1.3), typical initial superpixles (see section 2.3) are computed. To be able to theoretically get a perfect segmentation measure, the ground truth is projected to the superpixels by majority vote. This means that also for the edges y_{ij} the ground truth solution $\hat{y}_{ij}$ is known. To get an objective, adjustable measure of face quality, we use a linear combination of white noise $n_{ij} \in [0, 1]$ and the ground truth $\hat{y}_{ij} \in \{0, 1\}$ as artificial edge probabilities:

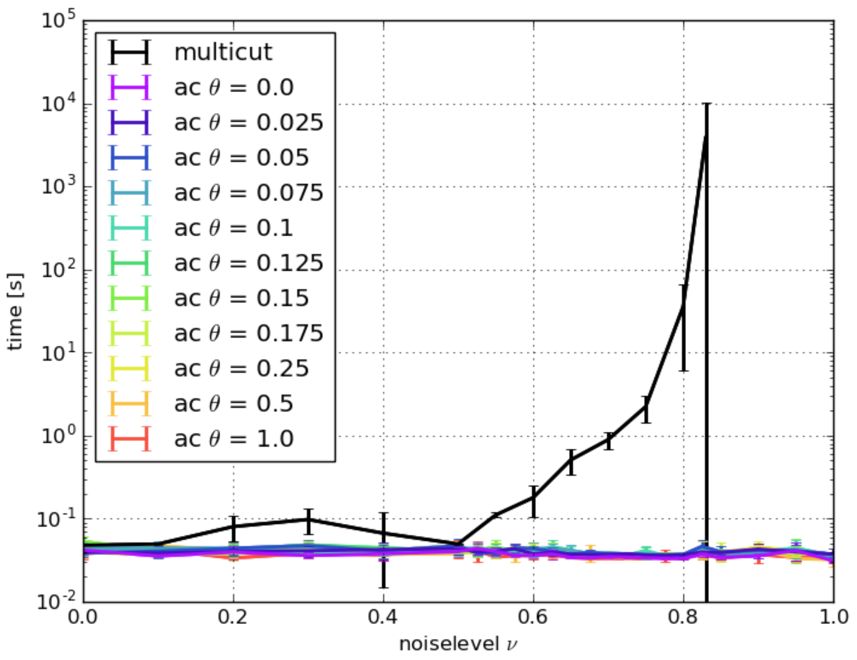
$$p_{ij}^\nu = \nu n_{ij} + (1 - \nu)\hat{y}_{ij}\,,$$

where $1 - p_{ij}^\nu$ is the merge probability for segments i and j.

The result of the comparison between the Multicut and the Agglomerative Clustering based on these artificial edge-weights is condensed in Figure 4.2 (a). The Agglomerative Clustering procedure is stopped as soon as there are as may segments as there are in the ground truth. It becomes obvious that the Multicut (black) is way less error-prone in case of bad edge-weights than all variants of Agglomerative Clustering that were tested (in color). Up to a noiselevel $\nu = 0.5$, thresholding of the edge-weights leads to a perfect



(a) segmentation quality



(b) runntime comparison

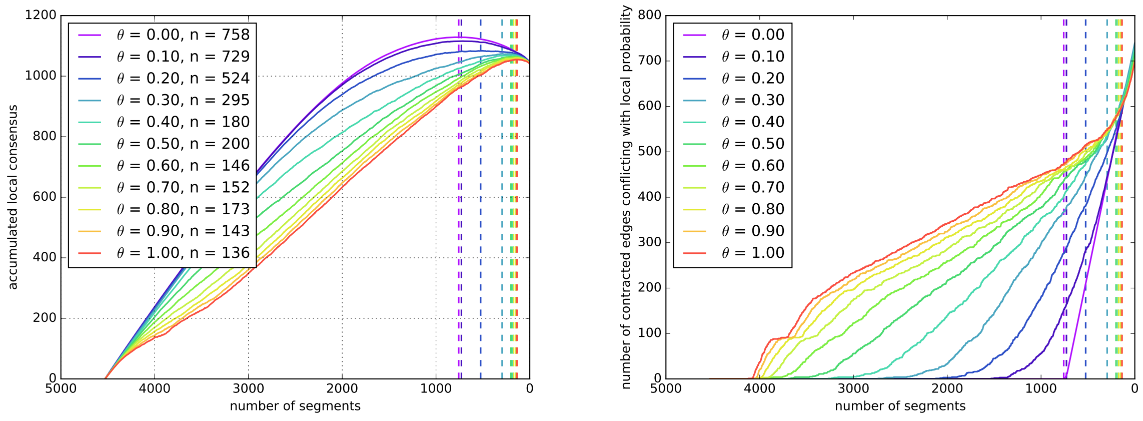
Figure 4.2: (a) shows a comparison of the error-proneness of the Multicut algorithm and the Agglomerative Clustering algorithm (stopped as soon as the number of segments apparent in the ground truth are reached) for multiple size regularizations θ. The used error measure is the Rand index. (b) reflects the actual runtime.

solution since the noise is normalized. The vanilla Agglomerative clustering with $\theta = 0$ and the Multicut both achieve a perfect score in this regime. The bigger the size regularization parameter θ, the more the qualitatively good edge-weights get polluted with the less precise idea of an uniform size distribution. This leads to an early deviation from the prefect score for all size regularized Agglomerative Clustering algorithms ($\theta > 0$).

Looking at the right side of the graph in Figure 4.2 (a), where $\nu > 0.5$ the quality of the edge-weiths get worse. Now the information about an uniform size distribution becomes more valuable in comparison. In the extreme case where the edge-weights are free of sensible information, it is beneficial to rely on the sizes and nothing else. It is interesting to observe the algorithms behavior in the intermediate region, where the edge-weights are not nonsensical but can be helped by a modest size regularization.

There are no measurements in Figure 4.2 (a) for the Multicut at high noise levels. The reason is simple: We run against the wall of NP-hardness – computation time explodes. This is demonstrated in Figure 4.2 (b). While all variants of Agglomerative Clustering are completely indifferent to the edge-weight quality, the Multicut has to pay a high price for its performance (as seen in Figure 4.2 (a)).

4.2.2 How to Stop?



(a) local consensus (b) local "error"

Figure 4.3: This image shows a possibility to naturally stop Agglomerative Clustering with altered edge indicators for a typical small neural segmentation problem. (a) shows for different parameters θ the course of the accumulated consensus value of each merge with decreasing number of segments. The number of segments in the groudtruth image is 244. (b) shows the number of merges that are done against local probability. The bigger θ the earlier in the merging procedure the clustering merges nodes that locally should stay seperated.

In the previous experiment from section 4.2.1 we run the Agglomerative Clustering with dynamically altering weights because we used the number of ground truth segments as stopping criterion. Only as long as the edge-weights correspond to the local merge probability a stopping criterion can come naturally: Cluster as long as there are edges left with a merge probability of

$$p_{\text{merge}} > 0.5 \; . \tag{4.7}$$

Edge probabilities that do not have a natural threshold can be stopped by determining the final number of segments. Since this number is generally not known in advance, for the case of neuron segmentation, working with these kind of edge-weight is usually seen as not convenient.

To enable the usage of non trivially thresholdable weights for new data (where we do not already have the ground truth available) I introduce a differentiation between the sequencing-weights w^s and the probabilistic weights w^p. The sequencing-weights are the ones for which the clustering is performed and the probabilistic ones do represent local merge probabilities. The modified weights in Eq. (4.6) holding nonlocal information (current segment sizes in this case) are used as sequencing-weights. Staying with this example the criterion from Eq. (4.7) is not sensible any more. The sizes can encourage an edge with a locally low merge probability to contract. Stopping at this early point prevents future merges to suit local probability.

To introduce a criterion that is able to recover the ability to naturally stop the clustering with non-probabilistic weights I will define a consensus value for each clustering step:

$$\text{consensus}_i = w_i^p - 0.5\,, \tag{4.8}$$

where w_i^p is the probability for the ith merge decision initiated due to the weight w_i^s. It is an indicator of how well the merge decision is backed up by local probability. The iteration i at which the accumulated consensus ($\sum_{j=0}^{i} \text{consensus}_i$) takes its maximal value can be taken as the point at which the local probabilities are most satisfied and therefore suited as a stopping criterion. Figure 4.3 (a) shows the accumulated consensus value for different θs in Eq. (4.6) ($\theta = 0$ means not altering the weights at all). A maximum is identifiable in all cases.

The proposed method is not only valid for the specific altering of the cluster weights in Eq. (4.6). It is applicable for arbitrary sequencing-weights as long as the knowledge of probabilistic weights or similarly weights for which a natural stopping criterion can be defined are available. This encompasses also the Semantic Agglomerative Clustering, which will be introduced in chapter 7. In section 9.2 an experiment comparing the resulting segmentation of an Agglomerative Clustering stopped according to this criterion and a stopping at the number of ground truth elements is performed. The automatic stopping does only slightly fall behind the ground truth stopping.

4.2.3 Conclusion

Being able to stop the agglomeration procedure naturally even for dynamically changing and not naturally thresholdable sequencing weights w_i^s is important. It enables both the usage of dynamical regularizes like the size regularizer presented in Eq. (4.6) and the usage of semantic labels as introduced in chapter 7. The presented stopping procedure is based on the introduced consensus measure from Eq. (4.8). To maximize the performance, a test of alternative criterion could be an option. It is for example not clear why to prefer a linear weighting of the difference as done in Eq. (4.8) over e.g. a squared version.

Chapter 5

V-Multicut

The segmentation algorithms presented in chapter 3 and 4 relied on the flat representation of edges. In chapter 3 we saw how this representation enables the introduction of an abstract cell complex 2.2.1 based on the topological grid. That structure allowed the introduction of constraints that enforce closed surfaces as it was e.g. done by Andres et al. [98]. In the following we will see how similar constraints can be formulated for the voluminous representation.

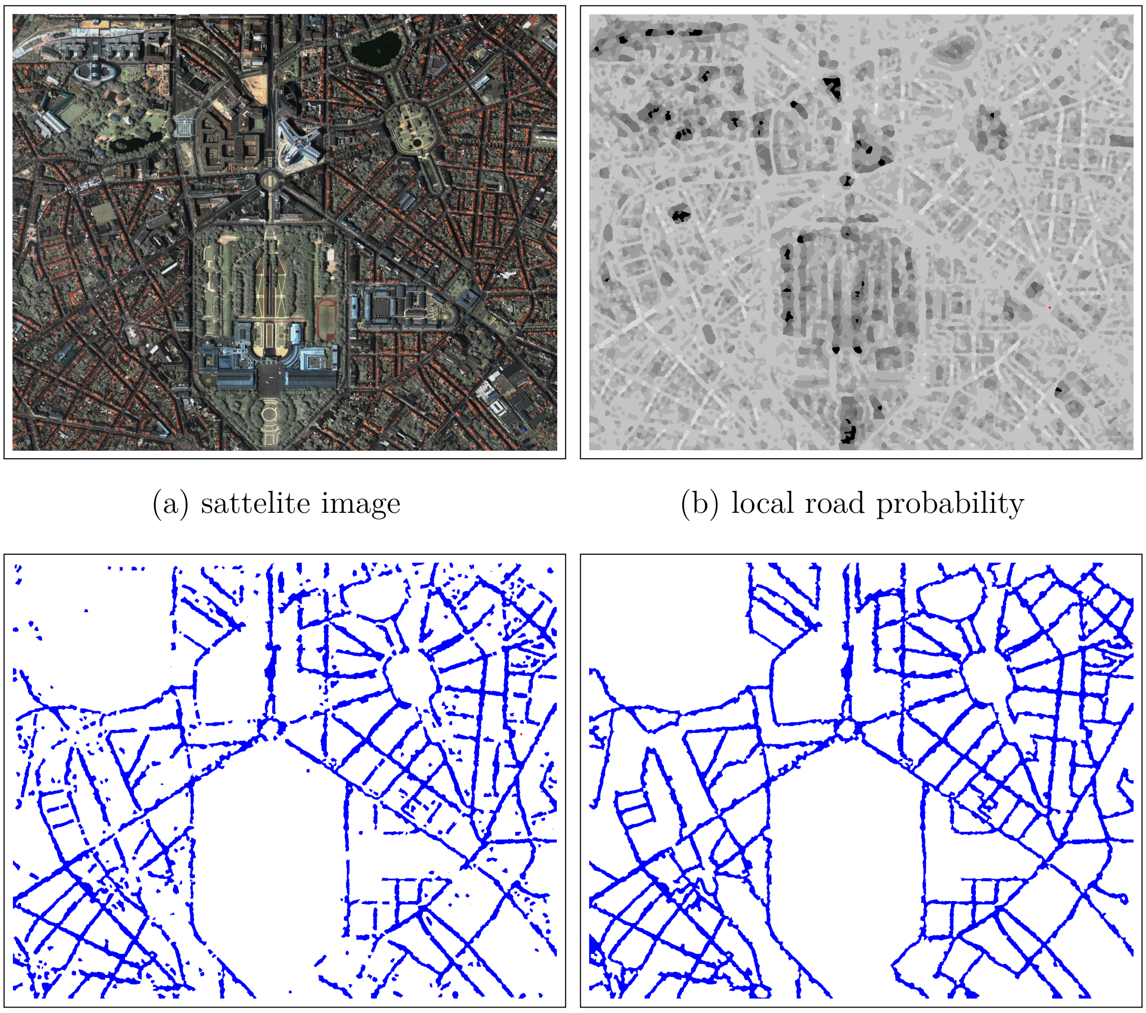
The thicker the edges/membranes in the image get the more stilted the flat representation gets compared to the volouminous representation (from section 2.1.5). Another benefit of the voluminous representation described i the following: We already saw that it is beneficial to introduce superpixels in a segmentation pipeline as an initial oversegmentation in cases where a critical boundary needs to be closed via the segmentation algorithm (one exemplary case is shown in Figure 3.2). Algorithms utilizing the flat representation rely on a face on top of the weak membrane. If the superpixels are already merged over the weak edge the situation of the Multicut or Agglomerative Clustering is desperate. This situation has no equivalent in the voluminous representation. There will always be a superpixel available to close the gap. The worst case is that the superpixels is bulky and does not represent the shape of the membrane precisely.

What hinders people from working on the voluminous representation is the difficulty of introducing gap closing constraints or other consistency constraints in the first place. Even the unambigous detection and classification of the gaps is not trivial. These are exactly the problems I will tackle within this chapter.

We will show how topological information based on abstract cell complexes can in general be used to constrain the shape of objects and present the segmentation of neural membranes from 2D images. To show that the V-Multuicut is not tailored to the segmentation of neurons but rather a general way to introduce topological constraints on binary classifications I present its performance on task of classifying a street network and the task of cell wall detection.

Since an unambiguously anchorage of a cell complex is possible on an object's skeleton, we develop a novel, deterministic and local skeletonization algorithm that allows to trace back the topological information from the skeleton to the original object and therefore allows the introduction of topological constraints on the object level.

The major difference to the cell complex as introduced in 2.2.1 is that there, each cell has its fixed place rooted in the topological grid. Since here only the original pixels (or equivalently superpixels) are utilized, the cell complex is dynamically constructed dependent on the current labeling of the pixels. A d dimensional pixel can represent



(a) sattelite image

(b) local road probability

(c) thresholding solution

(d) constrained solution

Figure 5.1: Segmentation of a street network: (a) satellite image (Google Imagery © 2014 DigitalGlobe, described in more detail in appendix A.3). To its right a superpixel-wise road probability is shown (b). A thresholded probability is shown in (c) and the segmentation utilizing the topological prior knowledge is shown on the lower right image in (d).

lower dimensional cells in the cell complex.

Figure 5.1 gives an impression of the impact of the topology driven constraints based on the cell complex structure on the example of a road network 5.1 (a). In (b) we can see a superpixel-wise local road probability computed by a random forest based on local appearance. The difference between (c) and (d) is due to the constraints formalized on the basis of a skeleton that express that the road network is not supposed to have any one way streets. This is surely an assumption that will not hold for general road networks but the given example shows that the Multicut like constraints on the voluminous representation, the V-Multicut, are potentially applicable beyond the segmentation of neurons.

One way to look at the problem at hand is to see it as a binary classification problem of pixels/superpixles only. Since one of the classes is interpreted as membrane, this binary classification problem translates into a partition problem. The shift in perspective is comparable to the shift in perspective when going from the edge representation to the node representation in the formulation of the Multicut in chapter 3. It justifies the classification of the V-Multicut as a partition algorithm.

5.1 Related Work

The two related directions of research are the incorporation of prior topological information like number of connected components or holes in a binary region labeling and the usage of abstract cell complexes for unsupervised image segmentation. This work can is localized between these two fields since I bring topological information, similar to the one used in the Multicut, to the object level. An early representative of the first direction is given by Zeng et al. [114]. The authors include topological priors in a graph based min-cut/max-flow algorithm. They are for example able to solve a foreground background segmentation with preset number of foreground objects approximately. A related problem is approached by Nowozin et al. [115] who show how to enforce one single connected foreground region within the framework of MAP-MRF LP relaxations. These problems are NP-hard and exact solutions can only be found on small problems. Recently there are attempts to include topological information in a more tractable way. Chen [116] changes the unary potentials of a random field iteratively for the final segmentation to fulfill certain topological requirements and Stühmer et al. [117] show that, in the case of a tree shaped foreground object, the connectivity constraint can be enforced via local constraints only and therefore the optimization can be done more efficient than in models with higher-order potentials covering global interactions. The second line of research, that is related to the ideas presented here, is the one on Multicuts relying on the flat representation of edges. Since this part of the literature was discussed in detail before, the reader is referred to the respective chapters 1.3 and 3.

5.2 Methods

This chapter provides all means to unambiguously detect and prevent all kind of topological priors on any segmented object in 2D. We will use this methodology to close holes in the segmentation of neural membranes and other network-like structures.

The popper formulation of the ideas outlined in section 5.2.1 base on several different concepts. The following sections will introduce everything necessary to formulate topological constraints on the object level. The very basic adjacency and neighborhood relations utilized within this chapter are specified in the following section 5.2.2. On this basis a skeletonization algorithm that meet all our requirements is constructed in section 5.2.3. Section 5.2.5 finally shows how localized constraints on the object level can be constructed from a skeleton in a way that they will stay valid, independent of all future label configurations.

5.2.1 General Problem Formulation

The problem we aim to solve can be expressed in the following way:

$$\mathrm{argmin}_{\mathbf{y}} \sum_i w_i y_i^T \text{ s.t. } C(\phi(\mathbf{y})) = \text{True} \tag{5.1}$$

We aim at finding a binary labeling $\mathbf{y} \in \{-1, 1\}^N \triangleq \{b, f\}^N$ of corresponding N superpixels as either foreground f or background b according to the weights $\mathbf{w}$. The foreground class f will be the one that is interpreted as a network like structure later on (road, membrane, cell-wall). $C(\phi)$ is some requirement on the skeleton $\phi(\mathbf{y})$ of the labeled foreground region. The constraints C are imposed on the skeleton level because there an unambiguous representation of an abstract cell complex can be found. This means that the constraints can be formulated in a convenient and reproducible way. Graphically speaking in Figure 5.3 defining what is a bump in the foreground and what is a real open end is hard. On the skeleton level this is trivial. In the following I will show how to impose constraints on the object level to enforce the constraints $C(\phi(\mathbf{y}))$ on the skeleton level. The objective function is linear in $\mathbf{y}$ and we will see later, that the constraints can be formulated linearly as well. Therefore existing solvers for integer linear programs can be used. In the presented examples, the goal is to find the energetic best labeling, that has a skeleton without dangling edges/open ends.

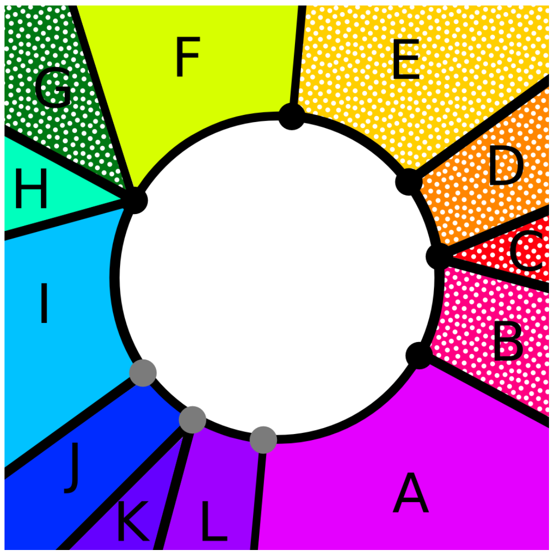
In an arbitrary background/foreground segmented image, the identification of junction like regions or endpoints is a haphazard task. However on a skeleton these informations are easily and unambiguously detectable. A skeleton endpoint has exactly one neighbor, a junction is characterized by having more than two[1]. Note that with this insight only computational limitations prevent us from using this for real world applications. One could e.g. use a cutting planes approach as follows: After an initial binary segmentation is performed by thresholding on the base of some local foreground indicators, a skeletonization allows the test on whatever topological constraint that one would like to impose. If the labeling leads to a skeleton that is not satisfying all demands, a global constraint forbidding this configuration can be imposed. This can be repeated until the resulting skeleton fulfills all requirements. This just looks in a brute force way through all possible solutions in an energetic order for the energetically best solution that fulfills the constraints on the resulting skeleton. In practice this is inefficient since we are not giving the algorithm any hints on where to correct the solution.

This leads to the next thought: Is a localization of this information possible or in other words, can one find a subset of all superpixels in the segmentation that lead to a characteristic element (e.g. an endpoint) in the skeleton, independent of all other elements in the complement of our subset. This depends on the problem instance and the skeletonization algorithm. The skeletonization must be deterministic and should use only local information. A non-deterministic skeletonization does not guarantee a consistent classification of the given labeling in successive steps in a cutting planes approach. An algorithm using global information for skeletonization works in principle (as explained above) but in practice it prevents the localization of the elements that cause a particular configuration in the skeleton. In the following I will introduce an appropriate skeletonization algorithm and a way to localize the cause on the object level of different configurations in the skeleton.

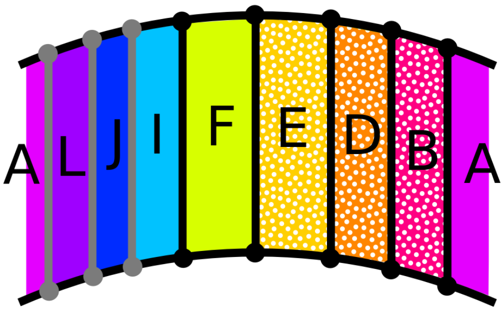
5.2.2 Neighborhood Relations

Since we make use of the neighborhood structure of different superpixels in the following (e.g. in the determination of simplicity and in the classification of skeleton elements), it is necessary to specify the used adjacency relations. In classical digital topology on a pixel level one needs to work with different (the lowest and the highest) neighborhoods for the two different classes to avoid a connectivity paradox (also called topological paradox) as

[1]For a rigorous definition see section 5.2.4.



(a)



(b)

Figure 5.2: Visualization of connectivity for a local simplicity criterion: The white element in the center of (a) is the one for which simplicity is tested. The colored regions are its neighbors. The white dots in segment B, C, D, E and G mark the elements belonging to the background class. The others, A, F, H, I, J, K and L are of the foreground (membrane) class. In image (b) we see the configuration as seen from a position inside the white center element. The elements shown here correspond to the foreground neighborhood N_f of the central segment. While background segments in the right figure are connected via all vertical lines, foreground elements are only connected via the gray ones. These are the ones where no background element is connected to the respective vertex. This is because for the foreground we do only take elements in N_f into account and for the background we count in the richer neighborhood N_b. On this example there are two connected components of foreground composed by F and I, J, L, A. The two connected components of background consist of G and E, D, C, B.

it is described in more detail in section 2.2.1.

We make the design choice that the foreground class in our two dimensional problems is supposed to be line connected and the background class is vertex connected (as defined in section 2.2.1).

$$N_f = N^{2D}_{\text{line}} \tag{5.2}$$

$$N_b = N^{2D}_{\text{vertex}} \tag{5.3}$$

An example is shown in Figure 5.2. In the left image A, B, D, E, F, I, J and L are in N_f of the white center element and C, G, H and K are the additional elements in N_b.

5.2.3 Skeletonization

The presented skeletonization algorithm works iteratively on a binary labeled set of superpixels. At the beginning one needs to assign an **order** to all segments. This order can be chosen randomly but once set, it must not change again.

Boundary

Once the order is set, one determines the border of the foreground object that has to be skeletonized: If an element s_i is labeled as foreground and at least one element in $N_b(s_i)$ is labeled as background then s_i is part of the boundary of the object.

Simplicity

The next step is to check all boundary elements in the preset order for simplicity. A simple element is one that, if its label is flipped from foreground to background, does not change the topology of the foreground object: The number of connected components of foreground and background shall not change. Again theoretically one could stop here and do a global connected components for both possible labellings of the questionable segment and compare them. This will prevent a localization of the constraints though.

In the following I will show that the global connectivity test can in 2D be substituted by an equivalent purely local criterion. This does not only affect the actual runtime of the simplicity checks but also assures that a localization of the constraints in section 5.2.5 is possible since the skeletonization is not depending on global information.

The core idea is visualized in Figure 5.2. As justified in section 5.2.2, we are working with different neighborhoods for foreground and background. It is enough to count the number of connected components in the foreground neighborhood of the questionable element $N_f(s_i)$ which is everything that shares a line with segment s_i. For considerations about connectivity between them, the richer neighborhood of the central element $N_b(s_i)$ needs to be considered.

For the simplicity consideration only connectivity on the surface of the central elements needs to be checked. Concretely the vertices bounding the central elements can be seen as the relevant connectors if one distinguishes them into two classes. The ones that bound background segments and the ones that do not (in Figure 5.2 distinguished by the shade of the vertex). Background segments are connected via all vertices, foreground segments only via the ones that do not bound background. For example in Figure 5.2 the foreground elements F and H are not connected because the background element G blocks the respective vertex. The two foreground elements J and L are connected because K, which is not in $N_f(s_i)$, is “mediating” the connection.

Alternatively to the graphically formulation with the two kinds of vertices one can write this in a more formal way utilizing the language established in section 2.2.1. Let us first define all vertices that are adjacent to a center element s_i as $v(s_i)$:

$$\tilde{p}^0 \in v(s_i) \Leftrightarrow \exists \tilde{p}^2 \text{ with } \tilde{p}^0 \prec \tilde{p}^2,\ \tilde{p}^2 \in s_i,\ \widetilde{\text{Active}}\left(\tilde{p}^0\right) \tag{5.4}$$

Let us further define all inter-superpixel lines L_{id} that touch a vertex in $v(s_i)$ and do not touch s_i itself as $l(s_i)$. Again formulated with the help of the notation from section 2.2.1:

$$\begin{aligned} L_{\text{id}} &= \{\tilde{p}_i^1 \,|\widetilde{\text{ID}}\left(\tilde{p}_i^1\right) = \text{id and } \widetilde{\text{Active}}\left(\tilde{p}_i^1\right)\} && (5.5)\\ L_{\text{id}} \in l(s_i) &\Leftrightarrow \exists \tilde{p}^0, \tilde{p}^1, \tilde{p}^2 \text{ with } \tilde{p}^1 \in L_{\text{id}},\ \tilde{p}^0 \prec \tilde{p}^1,\ \tilde{p}^0 \prec \tilde{p}^2, \tilde{p}^2 \in s_i && (5.6)\\ &\qquad \text{and } \widetilde{\text{Active}}\left(\tilde{p}^0\right) \text{ and not } \tilde{p}^1 \prec \tilde{p}^2 && (5.7)\end{aligned}$$

Vertices and lines are now the two entities that can establish connections in the neighborhood of s_i. We will now call $cc_q(E_1)$ the connected components within the set of segments

$E_1 = s_{i_1}, \ldots, s_{i_n}$ that are established by a set of lines and vertices q. For the following considerations, only the number of existing connected components is relevant. In our notation $\#_{E_2}cc_q(E_1)$ is the number of connected components in E_1 restricted to the subset $E_2 \subseteq E_1$ of elements. This means that for establishing connections all elements in E_1 can be used but for the counting in the end only elements in E_2 play a role. The condition for simplicity for element s_i is then

$$n_f(s_i) \overset{!}{=} n_b(s_i) \overset{!}{=} 1\,, \tag{5.8}$$

where

$$n_f(s_i) := \#_{N_f(s_i)}cc_{l(s_i)}\left(\mathrm{F}\left(N_b(s_i)\right)\right)\,, \tag{5.9}$$

$$n_b(s_i) := \#_{N_b(s_i)}cc_{v(s_i)}\left(\mathrm{B}\left(N_b(s_i)\right)\right)\,. \tag{5.10}$$

F and B are functions filtering out the elements that are either labeled as background or foreground. This means that if $s_i \in F(A) \Rightarrow s_i = f$. The restriction of counting to the foreground neighborhood N_f in Eq. (5.9) is important since it prevents for example foreground elements that have only one common vertex with s_i from contributing. A restriction to N_f for looking for connected components does not work since such vertex connected elements can “mediate”connections of the foreground class.

If an element is found to be simple according to Eq. (5.8), its label will change to background. This is respected by further simplicity checks of neighbors. If all elements in the boundary have been processed, the next layer of boundary is determined and the simplicity check starts anew. We are counting the number of times the boundary is recomputed and will refer to it as the **boundary level** (starting from one). The boundary level of a node is the one in which it is declared to be simple.

Preservation - Beyond Topology

Skeletonization by pure simplicity checks will result in skeletons which are topologically equivalent to the original objects. They will not necessarily represent the shape of their object though. One hole free object, for example, will always collapse to a single skeleton element. Since we want to be able to draw conclusions about the original objects’ shape from the skeleton, we need our skeletonization to reflect the appearance of an object as close as possible. Traditional skeletonization algorithms that rely on thinning preserve some points like local maxima of an eccentricity transform from the thinning. This turned out to work quite well in terms of representing shape. For this special use case, where a localization of the skeletonization is necessary for a further localization of the constraints, such an approach is unsuitable. The eccentricity transform introduces global dependencies – the exact location of a local maximum and therefore the location of a potential endpoint in the skeleton then depends on the whole object. This makes the desired localization of the problematic regions in a reproducible way impossible.

To resolve this problem one can rely on the following idea: Once a segment is tested for simplicity and found to be essential for the topology, it is protected from label switches in future iterations. This procedure, in combination with the random preset ordering of the elements, leads mostly to acceptable results. The issue is that the results strongly depend on the initial randomly chosen ordering of the elements. What becomes a large branch in a skeleton for one specific order could be completely ignored for another order. To resolve this ambiguity, we run the skeletonization twice. In the second run the order of elements is reversed. Every element that is preserved in the first run is also preserved in the second run. If the order is such that in the first run a branch is completely ignored, reversing the order will lead to a complete conservation of the branch. Note that this approach does not completely eliminate the dependency of the result on the order. One example of such an intrinsic ambiguity is an object (surrounded by background) consisting on three segments where each segment is connected (N_f) to both others. By changing the order each of them can become the resulting skeleton element.

5.2.4 About Islands, Endpoints and Junctions

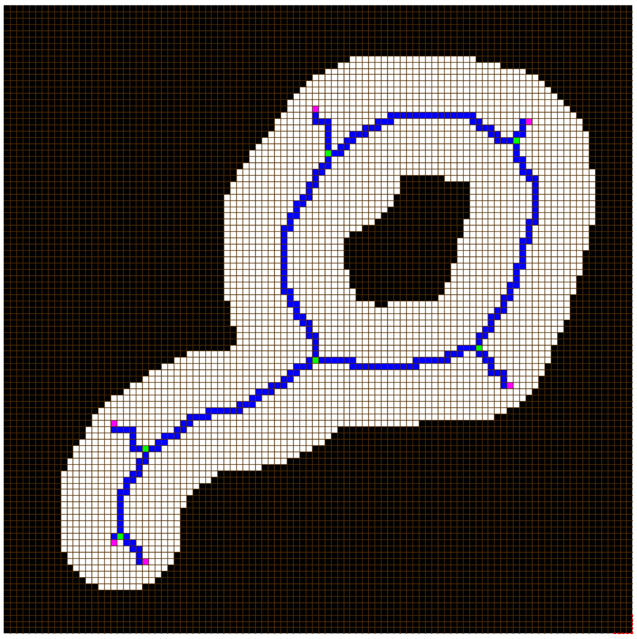


Figure 5.3: This figure shows a resulting skeleton (blue) of the white foreground object in the special case of an underlying pixel grid. The cyan pixels are skeleton elements classified as junctions, the magenta ones are endpoints.

n_b \ n_f	0	1	2	>2
0	-	l/j^*	-	-
1	$\boldsymbol{i}$	$\boldsymbol{e}/j^*$	-	-
2	-	-	$\boldsymbol{l}/j^*$	-
>2	-	-	-	$\boldsymbol{j}$

Table 5.1: Knowing the numbers n_f and n_b one can classify the skeleton elements. i stands for island. These are skeletal elements without any skeletal neighbors. e describes endpoints. These are the class of skeletal elements we want to prevent in our experiments. Junctions are marked by j. The distinction is necessary because of the ambiguities marked with '*'. The bold letters mark the cases that occur on a regular pixel grid.

A classification of elements in a 1D skeleton of a 2D object can on a pixel level be done by counting skeleton neighbors. This classification is essential since the different classes represent cells (of the abstract cell complex) of different dimensions. We distinguish between four different classes. Besides line elements, there are islands, endpoints and junctions. Skeleton elements classified as islands, endpoints and junctions represent zero cells whereas all other skeleton elements represent one cells. Both the bounding relation $\prec$ and the adjacency relation $\rightarrowtail\!\!\!\!\leftarrowtail$ of cells are now equivalent with the foreground neighborhood relation N_f of the segments which are associated with the cells. The concrete classification of the skeletal elements can be done with the help of Eqs. 5.9 and 5.10 and the Table 5.1. n_f and n_b are computed on the skeletonized foreground objects in the image. As shown in Table 5.1, there is an ambiguity for $(n_f, n_b) = (1, 1)$ and $(n_f, n_b) = (2, 2)$. This happens when more than one element is needed to make up a junction. These cases are distinguishable by the fact, that they are touching a vertex, that is surrounded by more than two skeleton elements (see Figure 5.2 for an example). This effect can not occur on the pixel level since it requires an irregular shape of the segments. The case$(n_f, n_b) = (1, 0)$ only occurs for completely surrounded elements. Their class has to be determined by the surrounding neighbors. Figure 5.3 gives an example of a classified skeleton.

5.2.5 From Skeletons to Constraints

Skeletonization as Mapping

Now that both skeletonization procedure and a possibility to detect all desired properties based on the abstract cell complex that was build upon the skeleton are given, one can think further. We want to go beyond imposing global constraints. Therefore we need a possibility to detect which elements in the original labeling cause the critical parts of the skeleton (the ones that we want to interdict). This is realized by interpreting the presented skeletonization ϕ as a mapping $\phi(s_i) = \{s_{i_1}, \ldots, s_{i_n}\}$. This can be done in a simple fashion: Each time an element is found to be simple, its ID, as given by the preset order (see beginning of section 5.2.3), is passed to its neighbors N_f. To prevent an order dependent global ID spread, an ID is only passed once per boundary level. In addition a simple element passes not only its own ID, but also the IDs that it received before. Note that this way no ID can be lost, meaning that every ID belonging to a foreground element in the beginning will be stored within at least one element of the skeleton in the end. An inverse mapping $\phi^{-1}(s)$ can now trivially be constructed. Given a skeleton element s, $\phi^{-1}(s)$ maps to all elements whose ID is stored in s. Note also that both ϕ and ϕ^{-1} are no bijections. To make the notation clearer lets define $\phi\left(\{s_{i_1}, \ldots, s_{i_n}\}\right) \coloneqq \phi\left(s_{i_1}\right) \cup \cdots \cup \phi\left(s_{i_n}\right)$.

Independent Domain

General Concept Imagine the following case: We have a preliminary labeling of our problem that leads to a skeleton with one endpoint. Assume further that our prior to the segmentation is that the foreground object does not lead to a skeleton with endpoints[2]. Is it now possible to exclude elements from the consideration of how to change the labeling such that it affects the skeleton in the desired way? To be able to answer this type of question, I introduce the concept of the independent domain: We call a subset D of the set of all segments in the problem S an independent domain of a skeleton point s_i if we can assure that each possible label change in $S \backslash D$ will change neither the fact that $s_i \in S$ is an element of the skeleton point nor its classification (according to section 5.2.4). The independent domain of a set of elements can be defined as the union of the independent domains of the elements in the input set. To counteract the occurrence of an endpoint one needs only to enforce a change of label within the independent domain of the endpoint. We want this domain to be as small as possible. The global problem would be a trivial independent domain. This would not help.

Concrete Construction One part in determining a small independent domain $D\left(s_i\right)$ of a skeleton element s_i is to look at $\phi(s_i)$. We know that this configuration has been

[2]Note that these constraints are not completely equivalent to the Multicut constraints since they allow curls (dangling lines closing on themselves).When taking the intersuperpixel line labels into account to deduce a region labeling this is utterly necessary. In the case of binary region- labeling this necessity is not given. In this sense the Multicut constraints are more strict .

mapped to s. Therefore we know that $\phi^{-1}(s_i) \overset{!}{\subset} D(s_i)$. To ensure that each following run will end up having the same skeleton element s, we need to ensure two things: First, all elements labeled as *background* before skeletonization and touching $\phi^{-1}(s_i)$ (via N_f) must be part of D as well. They being background in the first place started the skeletonization as we observed it and only for this specific skeletonization we can predict the outcome (because we ran it already). The second thing is that each element in $\phi^{-1}(s_i)$ should not be declared to be simple in a boundary level smaller than its original one. This can be ensured by including all elements in D that are reachable by a breadth-first search on the foreground starting from each of the elements s_i in $\phi^{-1}(s_i)$ with a depth of the boundary level of s_i, as introduced in section 5.2.3. For elements of $\phi^{-1}(s_i)$ in a certain depth we must ensure that it is not tested in an smaller boundary level. Otherwise this mismatch propagates and influences the skeletal character of s_i itself.

To summarize: We want to predict the outcome of the skeletonization of a certain subset of elements. This allows us to prevent unwanted configurations of skeletons to appear by constraining the problem in a respective way. Predicting the result of the skeletonization in general is tricky. Therefore we perform the skeletonization and find the regions that cause problems. If we can choose the region in a way that makes the decision on the skeleton independent of all elements that are not in this region, we can make a general rule (a constraint) out of it – We know it caused problems once and thus we know that it will do the same in each following run if the labeling within the region does not change.

5.2.6 Constraints

Given the independent domains (Ds) belonging to the critical regions of a skeleton of a labeling, the formulation of the constraints is straight forward. Our constraints forbid a labeling of the Ds, that is identical as it was before. Solving the problem with this constraint may resolve the problem immediately but can also cause new ones. New occurring problems can be of cause be treated in a similar way. Once a constraint is imposed, it shall not be removed to ensure the convergence to a consistent solution if possible. If $F_D^{t_D}$ and $B_D^{t_D}$ are all foreground and background elements in the independent domain D with the labels they have in the iteration t_D when the independent domain is found then the linear constraint is:

$$\sum_{f|s_f \in F_D^t} y_f - \sum_{b|s_b \in B_D^t} y_b \overset{!}{<} \sum_{f|s_f \in F_D^{t_D}} y_f - \sum_{b|s_b \in B_D^{t_D}} y_b = \text{const.} \tag{5.11}$$

In future iterations t the labeling of all elements in $F_D^{t_D}$ and $B_D^{t_D}$ must at least differ by one. Note that the solution computed by adding constraints for each found D is globally optimal. This means that it is identical to the solution gained in the “naive” way of imposing overall constraints whenever the skeleton has undesired properties. Because we constructed the Ds in a way that elements outside them do not change the classification of the respective skeleton point, only a label- change inside this regions can resolve problems. Label changes outside do not decrease the energy and do not change the problematic regions. A further reduction of the constraint regions to e.g. only the surface of an object may seem to be sensible but the guarantee to find the globally optimal labeling (labeling that provides the biggest energetic contribution and at the same time fulfills the requirements to its skeleton) is lost.

5.3 Decomposition

In the V-Multicut framework a decomposition of the problem is done in a similar fashion as in section 3.3: At the very beginning we see our problem as a graphical model containing only unaries. In following iterations we add constraints that can also be understood as higher-order factors connecting the nodes in the respective Ds. After the first iteration the problem can be reduced to just resolving the now connected problems within the Ds (see Figure 5.6 for visualization). We know that outside the constraint regions nothing will change because these elements still are influenced only by their unaries. The different Ds are completely decoupled problems. When adding additional constraints in the following iterations their Ds might overlap with the existing ones. The problem now decouples into all connected components of Ds.

5.4 Experiments

5.4.1 Three Applications

The procedure explained above allows to demand a big variety of requirements from the skeleton. In the following I will concentrate on forbidding dangling edges and islands in the skeleton since this has the potential to improve the segmentation of neurons. To test the robustness of the presented algorithm and to evaluate its potential I choose to try it for the segmentation of a road network from a satellite image (see Figure 5.1) and on a light microscopy image of plant cells and their cell walls in addition to the segmentation of neural membranes.

The starting point of all three applications is a superpixel oversegmentation. Therefore the superpixel construction is outlined in the following. The superpixels are created via watershed on the gradient image of a probability map. The pixel-wise probabilities are learned by ilastik [48]. Ideally we want to have superpixels on foreground and on background but no superpixels containing both classes. The superpixel are the elements s_i on which the binary classification, leading to a segmentation, is performed. The unary potentials w_i for the segments s_i are for this proof of concept created via averaging of the respective pixel probabilities.

For the approach of forbidding endpoints, the oversegmentation is essential. Operating on the pixel level, one could resolve constraints easily by relabeling one single pixel in the independent domain of an endpoint to background. This creates a loop. When operating on a reasonable oversegmentation it makes such loops more expensive. For it to work on the pixel level additional constraints must be introduced. One could forbid lollipop like configurations (curls) in the same fashion as it is done in the Multicut framework (see chapter 3).

Segmenting Road Networks from satellite images is a hard task if one aspires to get a plausible (connected) road segmentation. For example roofs of buildings are sometimes hard to distinguish locally from roads when. Via the constraints, the nonlocal context of the labeled regions is taken into account. Note that the assumption of absence of open ends in the skeleton/no impasses does not hold on general road maps. One

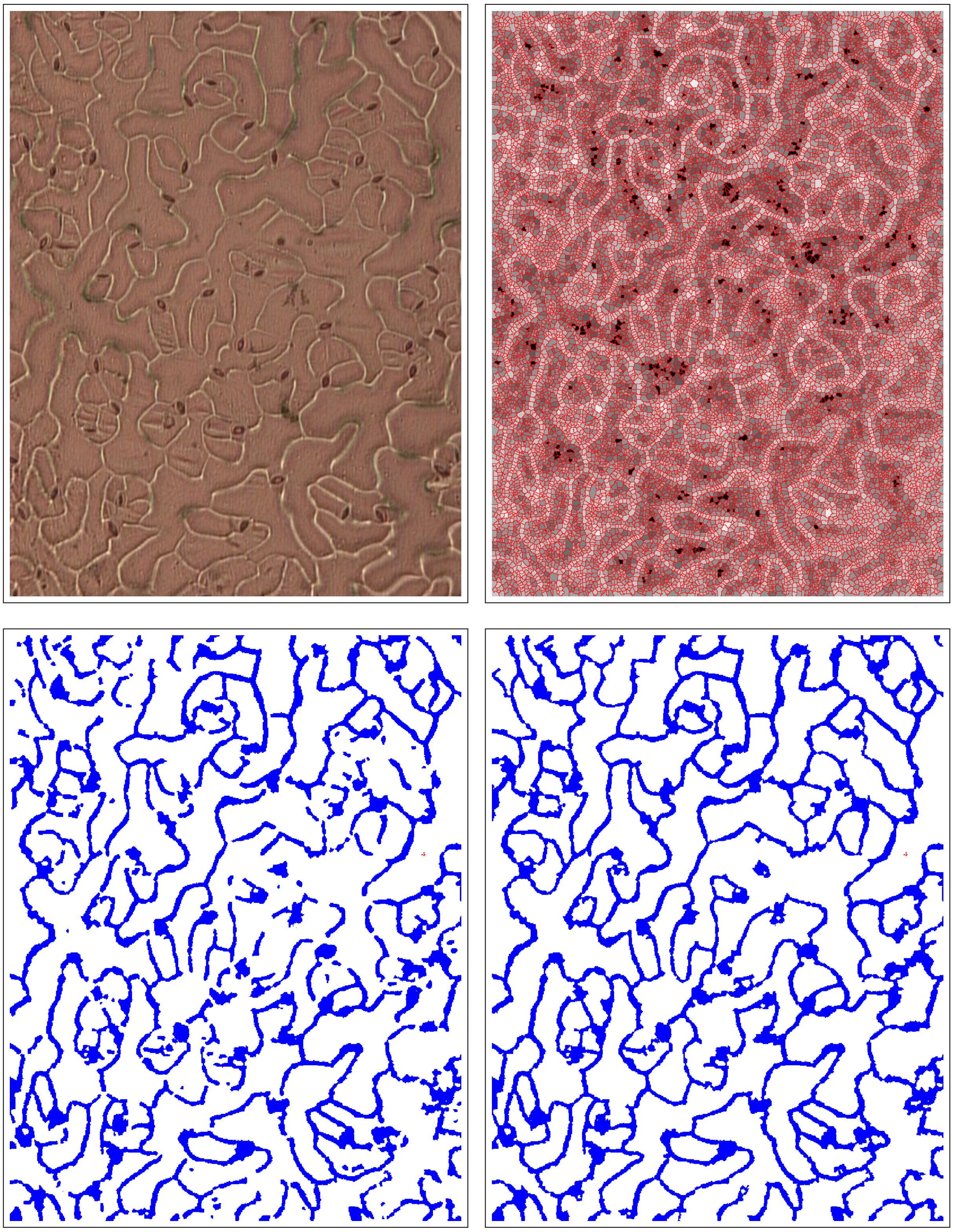


Figure 5.4: Segmentation of cell walls from light microscopic images (see appendix A.4 for more details). The top left image shows an image from light microscopy of plant cells. Since the cells are locally indistinguishable the segmentation of the cells can only be achieved indirectly by a segmentation of the bright cell walls. To the top right a wall probability on the superpixel level (gray-scale) and the explicit boundary of the superpixels used (red) are shown. The bottom left and right figures show the result of a mere local decision and a segmentation considering the proposed constraints.

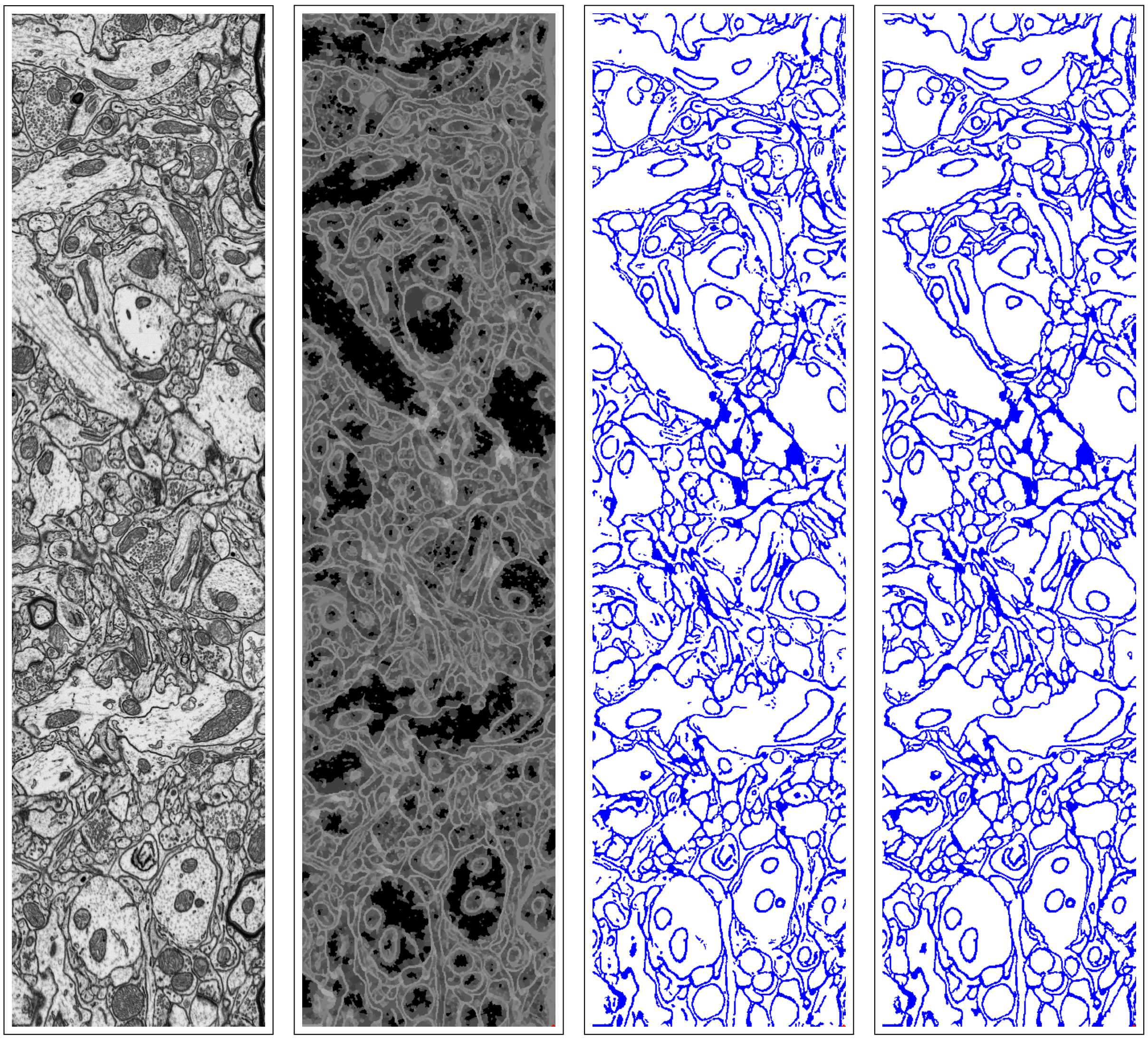


Figure 5.5: Segmentation of neural membranes: From left to right the images show the raw data originating from an electron microscope (unwrapped surface of 3D volume described in appendix A.1), the superpixel-wise membrane probabilities, the thresholding result and the constrained segmentation.

example of such a satellite image is shown in Figure 5.1 (a) together with the probability map for streets mapped to the superpixel level in Figure 5.1 (b). In Figure 5.1 (c) and (d), a direct comparison of the results of thresholding and the proposed method on this data is shown. One can see that gaps are either closed or the dangling edges are removed completely. Occasionally a dangling edge is transformed into a curl. This undermines the gap closing properties of the algorithm and indicates that for those places more complex constraints or better unary information is necessary.

Segmenting Cell Walls is a problem that is closely related to the detection of neural membranes. As it can be seen in Figure 5.4 the interdiction of open ends in the cell walls skeletons are visually benefiting the segmentation. Both false positives are revised and false negatives are recovered to produce a topologically consistent segmentation.

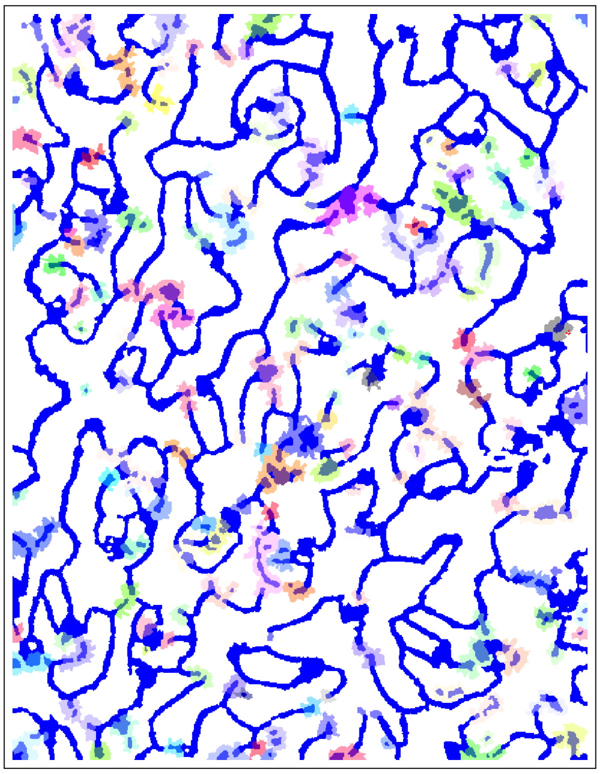
Segmenting Neural Membranes , being the primary goal of this thesis, is of special interest. The net-like structure arises from a cut through the densely packed neurons which are in principle (strongly) deformed spheres. Therefore no open ends are possible. Figure 5.5 gives an impression of the capabilities of the imposed constraints. The resulting segmentation definitely looks cleaner and some apparent holes in membranes have been closed satisfactorily. Some lollipop like structures have formed. At these places the desired gap closing behavior was prevented. Detection of these structures and their prohibition is possible on basis of the classified skeleton.

5.4.2 Decomposition

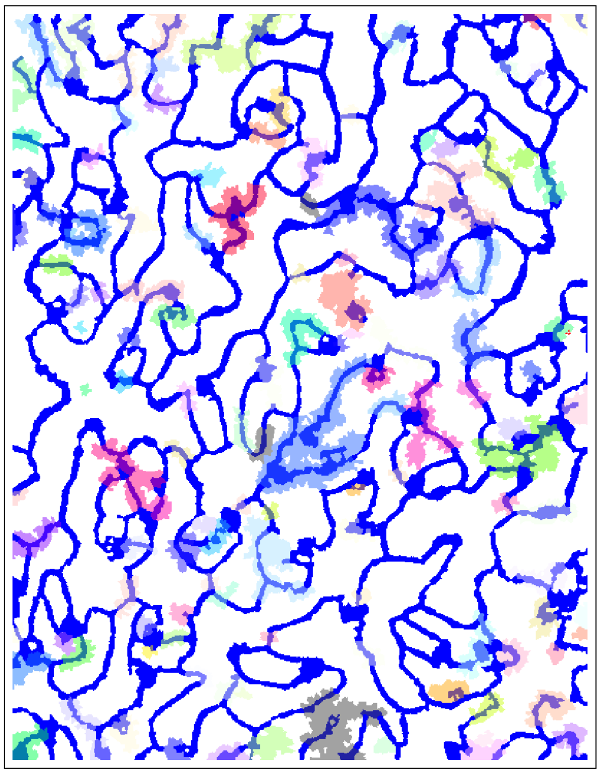
In Figure 5.7 one can see the benefit of the proposed decomposition in terms of runtime. There it gets visible that, in some cases an increased problem size leads to a smaller number of needed iterations. This happens when, for larger images, information that is outside the field of view in smaller snippets help the algorithm to converge. The variance of the runtime between the three experiments can be explained with a differing quality of the respective unaries. Figure 5.6 illustrates the connected components of constraints that can be solved independently for the first and the final iteration in the cutting planes approach.

5.5 Conclusion

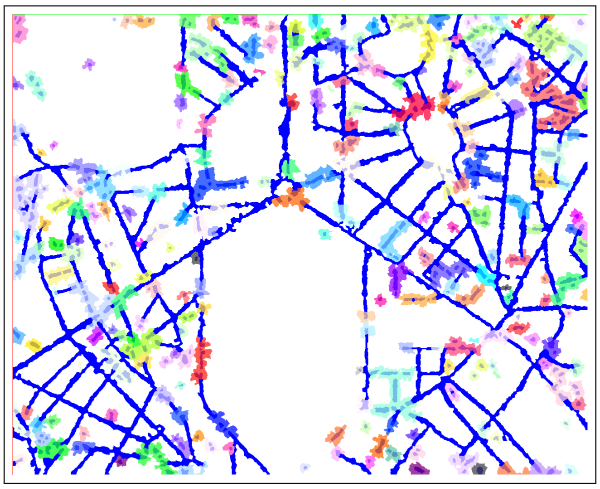
In this chapter we saw how in general topological information based on abstract cell complexes can be considered in a binary image labeling task. I demonstrate the usage of this method for the task of forbidding dangling edges in the segmentation of network like structures on 2D images. It is notable that this method does not suffer from shrinking bias since I do not draw on pairwise regularizers that try to minimize the length of object boundaries. One could try to include pairwise regularizes that try to promote a gap closing based on local orientation estimations. Note that this makes it impossible to decompose the problem in the presented way. Another way to extend the field of potential application of the algorithm is the inclusion of different constraints. Multicut like constraints, forbidding also lollipop like structures, can be realized by taking independent domains of whole paths into account. Allowing only spherical objects for example can be realized by allowing only islands in skeletons. In addition one could think of constraints



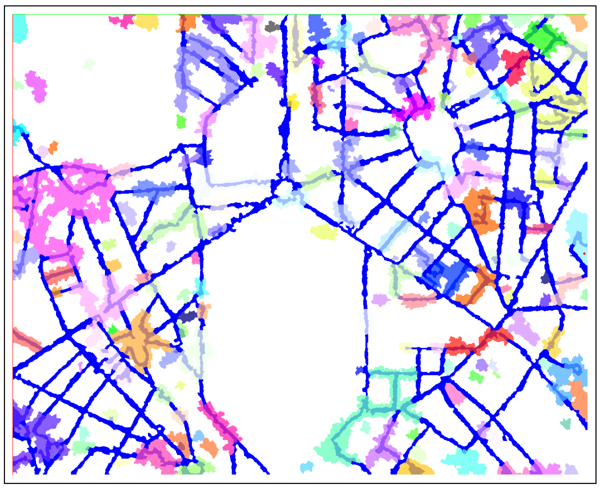
(a) plant cellwalls initial



(b) plant cellwalls final

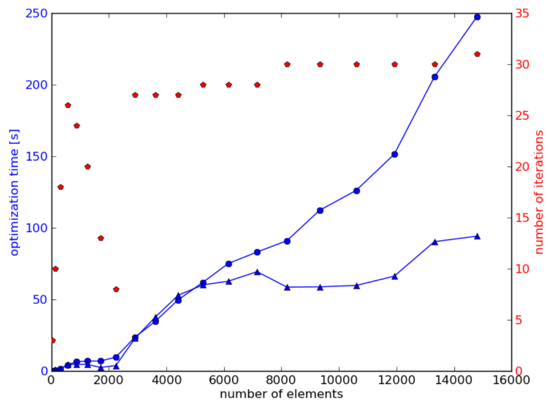


(c) road network initial

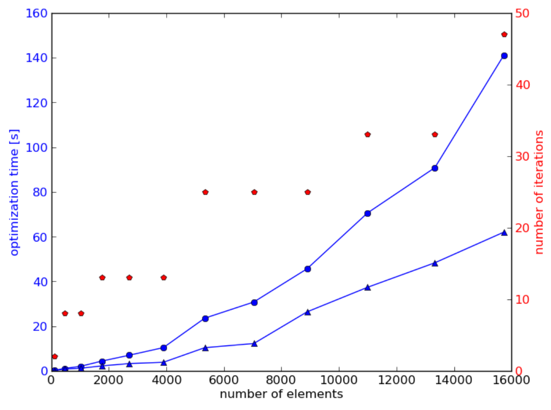


(d) road network final

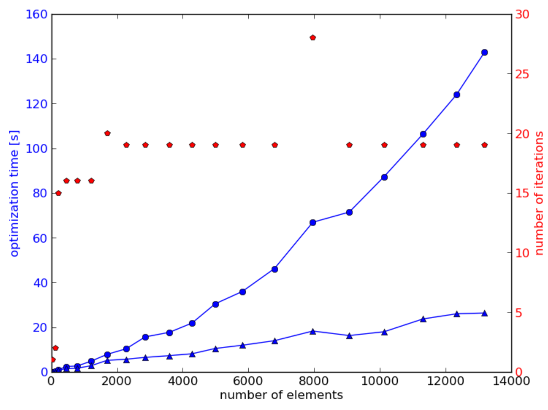
Figure 5.6: Visualization of decomposed regions in the problem of segmenting cell walls (a) and (b) and the segmentation of a road network (c) and (d). (a) and (c) show the thresholded solutions in blue together with all connected components of constraints found on this thresholded solution color-coded. (b) and (d) show the same color-coded regions for the last iteration of the cutting planes iterations where all constraints necessary for the optimal solution (blue) are visualized.



(a) road network



(b) plant cell walls



(c) neural cell membrane

Figure 5.7: For the three given applications, the road network (a), the cell-walls of plants (b) and the main application, the segmentation of neurons (c) the runtime (blue) of the global approach (circles) and the decomposed approach (triangles) is compared for different problem sizes. Also the respective total number of iterations (red) is reported.

coupling several independent domains and therefore enforce a skeleton with a fixed number of junctions and endpoints (for example for the segmentation of a starfish).

This work can also be expendable to higher dimensions. To apply the proposed skeletonization algorithm in 3D one has to adapt the localized simplicity criterion. The presented equality of the localized simplicity criterion and the topological properties of the skeletonized object do only hold for 2D. If this succeeds it will be possible to correct the segmentation of membranes and lines embedded in 3D.

Section 5.4.2 shows that the V-Multicut decomposes in a similar way as the Multicut does (shown in section 3.3). Following the arguments from section 3.3 one can parallelize the optimization. The skeletonization procedure as presented in section 5.2.3 is still a global procedure. For large scale problems one could try to decompose the skeletonization procedure by choosing the skeletonization order explicitly. Remember that that was done randomly so far. A path consisting of the first elements in the ordering, separating the image, neighbored by a path consisting of the elements with the latest elements in the ordering, separates the problem into two independent ones after the first line is processed. This could be done iteratively. How much this scheme can reduce the skeletonization time is an open question left for future investigations.

Chapter 6

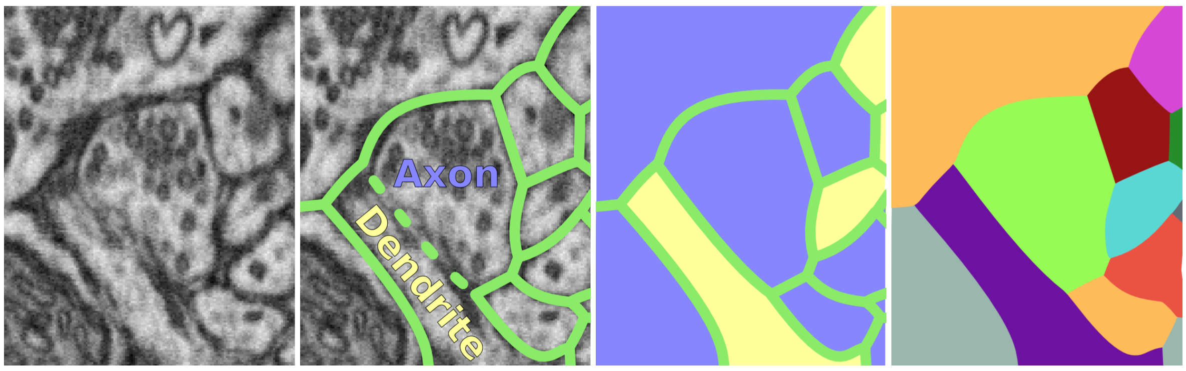
Asymmetric Multiway Cut

In this chapter I present a novel approach to the problem of neuron segmentation in image volumes acquired by electron microscopy. While existing methods rely solely on the detection of neuron membranes, I augment the membrane information by detecting markers for higher-level biological priors. These markers are then incorporated directly into a globally optimal segmentation procedure. The problem is formulated in the recently proposed framework of the Asymmetric Multiway Cut algorithm family. Within this class of problems, it is possible to simultaneously achieve an optimal *partitioning* and *semantic labeling* of a graph, representing the supervoxels of the image volume. In particular, we introduce "axon" and "dendrite" as semantic labels for the supervoxels and show how to extract the local evidence to support them. My experiments demonstrate that the introduction of semantic classes significantly improves the solution of the partitioning problem, i.e. the segmentation of neurons in the volume.

As described in section 1.2, the main difficulty for the automated segmentation lies in the fact that heavy metal staining, used for EM imaging, does not make the individual neural cells locally distinguishable from each other by color or texture. The algorithms therefore resort to separating the neurons by detecting the cell membranes. This approach presupposes an extremely high accuracy in the detection of membranes since a tiny local mistake in their segmentation can cut a neuron in two or merge two neurons together – a global error, which can substantially influence the reconstructed neural circuit.

Some of these errors can be pinpointed by analyzing the resulting circuit, where, for example, a segment which consists of two falsely merged neurons will exhibit contradictory biological properties of its constituents. However, to the best of my knowledge, none of the existing approaches for automated segmentation can incorporate non-local information of this kind directly into the segmentation procedure. Such a combination of high-level biological priors with local membrane evidence is the main target of this contribution.

In more detail, the priors taken into account reflect the probability of a supervoxel to belong to an axon or a dendrite. While for most supervoxels in the image volume these priors are uninformative, some contain axon/dendrite indicators, which can be found automatically. These indicators include, for example, vesicle clouds for axons. As suggested by Krasowski et. al. [118, 119], Asymmetric Multiway Cut (AMWC) – a recently proposed algorithm for simultaneous graph partitioning and semantic labeling [120] – can take such information into account and can ensure that no supervoxel containing dendrite evidence is found in the same reconstructed neuron as a supervoxel that must, according to its appearance, belong to an axon. In case the image volume includes neuron somas, they can be introduced as an additional class, since AMWC can handle multiple semantic



(a) raw data (b) problematic boundary (c) semantic labeling complements partitioning (d) the resulting segmentation profits

Figure 6.1: To the left an extract of the raw data including hardly visible cell membranes is shown (a). The second image highlights a possible wrong merge decision with a dotted lines while easily detectable membranes are marked by continuous lines (b). In addition semantic affiliation is indicated. Image (c) sketches how a semantic labeling of all neurons (visualized by color of regions) can help the partition process. The rightmost image (d) finally shows a resulting segmentation that profited from the semantic labeling in the intermediate stage.

classes. AMWC uses two complementary sources of features: those describing the affinity of touching segments, as well as those that capture distinct segment characteristics pointing to its semantic class.

The remainder of this chapter is organized as follows: section 6.1 gives an overview of related work beyond the one presented on general neuron segmentation in chapter 1.3. In the following section 6.2, the core algorithm of this chapter – the Asymmetric Multiway Cut (AMWC) – is explained in detail. Section 6.3 shows the variant of AMWC best suited for the inclusion of sparse biological priors and specifies the complete pipeline from the raw data to the final segmentation. Section 6.4 contains the experimental results as well as a comparison of performance with established algorithms. I compare explicitly against the learned agglomeration approach of Nunez-Iglesias et al. [69] and Multicut-based segmentation as proposed by Andres et al. [73] on an isotropic dataset produced via a FIB/SEM microscope by Knott et al. [16]. Finally all findings and their implications are discussed in section 6.5.

6.1 Related Work

Andres et al. [73] find an overall globally optimal segmentation, given supervoxel oversegmentation and supervoxel boundary evidence by introducing topological constraints on the supervoxel adjacency graph. My work builds on this set of constraints and on the Multicut algorithm as described by Kappes et al. [74] in general. Therefore all related work mentioned in chapter 1.3 is relevant for this work as well. But, unlike Andres et al. [73] and all other methods listed in chapter 1.3, the merge/split decisions are not purely based on the local boundary evidence. The main novelty of this contribution is a principled way to consider non-local prior information.

Computationally, I rely on the Asymmetric Multiway Cut algorithm, recently introduced by Kröger er al. [120]. This algorithm extends the Multicut of [73] by a simultaneous semantic node labeling.

The presented work relies on synapse detections, which provide important clues about the biological priors I want to include. Kreshuk et al. [19], Becker et al. [20] and Fua and Knott [21] all present reliable approaches to perform this task.

6.2 Methods

The key part of the presented segmentation pipeline is the Asymmetric Multiway Cut (AMWC) segmentation algorithm. AMWC was very recently presented in a conference submission by Kröger et al. [120], and a detailed examination of the algorithm is still lacking. I therefore offer an extensive explanation of the AMWC algorithm in the following section 6.2. In 6.3 it is shown how to set the stage for AMWC to tackle the EM neuron segmentation problem and how exactly one can include the biological prior information to improve the quality of the segmentation.

The Asymmetric Cuts or Asymmetric Multiway Cut algorithm is a generic algorithm for joint *semantic labeling* and *partitioning* of a graph. In image analysis the graph is often given by an oversegmentation of an image. In this case, the nodes correspond to the superpixels of the oversegmentaion. The partitioning is performed according to pairwise affinities of nodes. Simultaneously, a semantic labeling of the nodes based on local information is made consistent with respect to the partition.

For the neuron segmentation problem, the motivation for the inclusion of semantic labeling is, that in difficult cases, where the edge evidence given by the membrane detection is inconclusive, semantic labels provide additional indications that support merge decisions of neighboring nodes (see Figure 6.1).

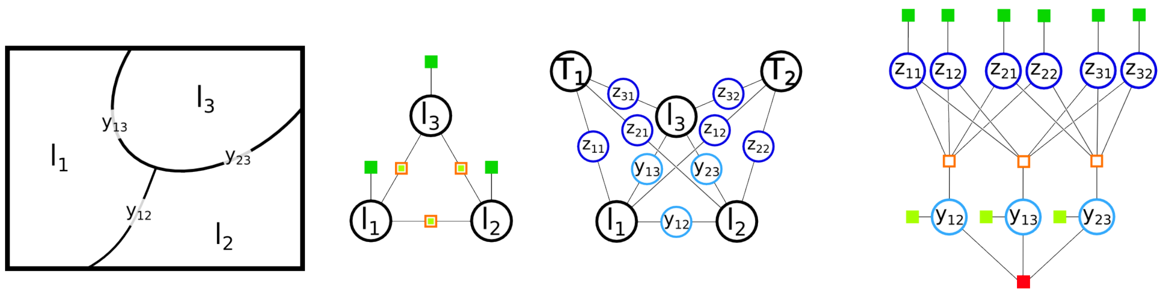
The work of Kröger et al. [120], where the AMWC problem was originally introduced, shows that the partitioning and the labeling do benefit from each other – solving the semantic segmentation problem first and following up with the partitioning problem or vice versa is not equivalent to the joint solution.

Analogous to the Multicut, the AMWC can be explained with respect to the regions in the oversegmentation (node representation) or the edges between them (edge representation). The first view is presented by a pairwise graphical model (an example is given in Figure 6.2 (b)). Note, that the consistency between the labeling and the partitioning has to be ensured by the pairwise factors (see Eq. 6.5). Also note that the pairwise affinities can be both repulsive and attractive. By going to the edge domain, where the random variables are not the original nodes, but the edges between them, we attain a reduced (binary) label-space for all random variables that is more optimization friendly and a more explicit formulation of the constraints.

The following chapter will explain both formulations and their relationship. Figure 6.2 – a simple example rooted in image analysis – will illustrate the descriptions. In principle, AMWC can be applied to fully connected graphs, but for the sake of clarity and with our applications in mind a sparsely connected example is chosen.

6.2.1 Node Representation

The AMWC problem can be formulated as a node labeling problem for a graph $G = G(\mathcal{V}, \mathcal{E})$, where each node i has a partition label l_i^p as well as a semantic label $l_i^s \in T$.



(a) exemplary segmentation (b) node representation (c) node and edge variables (d) edge representation

Figure 6.2: Two equivalent formulations of the AMWC optimization problem on the basis of an exemplary oversegmentation of an image (a) into segments l_1, l_2 and l_3, separated by edges y_{12}, y_{13} and y_{23}. Assume that two semantic labels are introduced for the regions. The **node representation**, acting on variables l_1, l_2, l_3 (○), can be found in (b). The **edge representation**, acting on variables y_{12}, y_{13}, y_{23} (○), can be found in (d). (c) demonstrates the relationship between the variables of both models, which can be made explicit by introduction of additional terminal nodes T_1 and T_2 (○) for the two semantic labels. These terminal nodes are connected to the original nodes l_1, l_2 and l_3 by terminal edges z_{11}, ..., z_{32} (○). In both (b) and (d), the prior information on the edge strength is shown by 🟩, while □-markers represent factors which ensure consistency between labeling and partitioning (Eq. (6.5) in the node representation and Eqs. (6.8) to (6.12) in the edge representation). Note that these factors are pairwise in (b), where they connect two region nodes, and of higher order in (d) where they connect an internal edge to $2 \times |T|$ terminal edges. Prior information on the semantic label affiliation of the regions is encoded in factors marked by 🟩. In the edge representation, another higher-order factor 🟥 has to be added to ensure that the binary decisions on individual edges correspond to a valid partitioning (Eq. (3.11)).

Here T is the set of all semantic classes. One can distinguish between two kinds of semantic classes: T_c and T_n with $T = T_c \cup T_n$ and $T_c \cap T_n = \emptyset$. We say we only allow "cuts within" in semantic classes $l^s \in T_c$. This means that after joint labeling and partitioning, adjacent segments may both be of class $l^s \in T_c$ but a segment of class $l^s \in T_n$ can only have semantically different neighbors. In the node formulation of AMWC, each supervoxel i is associated with a label

$$l_i = \left(l_i^p, l_i^s\right), \, l_i^p \in [1, \cdots, \|\mathcal{V}\|], \tag{6.1}$$

$$l_i^s \in [1, \cdots, \|T_n\| + \|T_c\|] \tag{6.2}$$

where l_i^p is an integer that indicates the affiliation of the node i in the partitioning problem and l_i^s denotes its semantic class which can be one of the $\|T_c\|$ classes with "cuts within" or one of the $\|T_n\|$ classes without. For the partitioning part of the problem, the number of final clusters is not known beforehand. To account for all possible partitions, one needs as many different partition labels as there are nodes: $\|\mathcal{V}\|$.

To enforce the desired behavior, I introduce pairwise factors between all neighboring nodes, which represent the inclination of a pair of neighbor nodes to end up in the same cluster and thus to have the same partition label l_p. In addition to this, the pairwise factors enforce consistency between the semantic labels and the partition labels - e.g. if the semantic labels of an adjoining pair of nodes differ, the partition labels should differ as well. The semantic labels, in their turn, are influenced by unary factors that encode the local affinity of nodes to the semantic classes (see Figure 6.2 (b)). Minimizing the following energy function results in the MAP state of the graphical model:

$$\underset{l}{\operatorname{argmin}} \left(w \sum_{i \in \mathcal{V}} E_i(l_i^s) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(l_i, l_j) \right). \tag{6.3}$$

Note that the unary terms depend only on the semantic labels.

$$E_i(l_i^s) = -\log \left(\frac{p_i^s}{1 - p_i^s} \right) \tag{6.4}$$

Here p_i^s is the prior probability that node i belongs to the semantic class $s \in T$. With p_{ij} being the probability that node i and node j are in different partitions, the pairwise potentials are constructed in the following way:

$$E_{ij}\left(l_i, l_j\right) = \begin{cases} 0 & \text{if } l_i^p = l_j^p, l_i^s = l_j^s \\ -\log\left(\frac{p_{ij}}{1-p_{ij}}\right) & \text{if } l_i^p \neq l_j^p \\ \infty & \text{if } l_i^p = l_j^p, l_i^s \neq l_j^s \end{cases} \tag{6.5}$$

Solving the partitioning- and the semantic segmentation problem simultaneously to global optimality is hard. One reason is that the energy function is highly degenerate, as it is invariant under all permutations of the partition labels. Another reason is the non-submodular nature of the pairwise factors. On the one hand they may have positive as well as negative values and on the other hand they implicitly encode consistency constraints between the partition labels and the semantic labels (encoded in Figure 6.2 (b) by orange frames). Nevertheless, Kröger et al. [120] show that globally optimal solutions can be found for problems of relevant size by rewriting them as a cut problem in the edge domain.

6.2.2 Edge Representation

While the formulation presented in section 6.2.1 is intuitive and close to the formulation of common probabilistic graphical models, only the equivalent formulation as a cut problem can actually be solved in practice. We will see how it can be written as an integer linear program, to which a whole variety of solvers can be applied. In the general case, the problem is NP-hard. Since we apply the algorithm not on arbitrary graphs, but on graphs that originate from volumetric images, the graph architecture is highly structured. Kappes et al. [74] show that for those cases only a small subset of the cycle constraints is really needed to find the globally optimal solution. A cutting-planes approach, where violated constraints are added iteratively, therefore stands to reason.

The cut formulation in the edge domain relies on binary random variables, which can be associated with edges in the adjacency graph, and binary variables, which indicate the affiliation of the nodes to the so called terminal nodes, representing semantic classes. The price one has to pay for this clear representation is that not every possible edge labeling corresponds to a consistent node partitioning. Besides, the uniqueness of the affiliations to terminal nodes is not given naturally. These shortcomings must be addressed by the introduction of explicit constraints. One big advantage of this representation is that there is no intrinsic degeneracy of the objective.

More formally, the graph G from section 6.2.1 is now extended by terminal nodes T and terminal edges $\mathcal{E}_T$. Together with the internal nodes $\mathcal{V}$ and the internal edges $\mathcal{E}$ they describe all nodes $\mathcal{V}'$ and all edges $\mathcal{E}'$ in the augmented graph $G'(\mathcal{V}', \mathcal{E}')$. Figure 6.2 (c) shows the terminal nodes and edges introduced for the graph in Figure 6.2 (a). All terminal nodes are connected to all internal nodes in $\mathcal{V}$ via terminal edges $\mathcal{E}_T$.

$$\mathcal{V}' = \mathcal{V} \cup T, \mathcal{E}' = \mathcal{E} \cup \mathcal{E}_T, \text{where } \mathcal{E}_T = (i,t) \mid i \in \mathcal{V}, t \in T \tag{6.6}$$

We introduce binary variables $y_{ij} \in \{0,1\}$ and $z_{it} \in \{0,1\}$ for all edges in $\mathcal{E}$ and $\mathcal{E}_\mathcal{T}$. The optimization problem then reads:

$$\underset{\mathbf{y}' \in \{0,1\}^{|\mathcal{E}'|}}{\operatorname{argmin}} \left(\kappa \sum_{(i,t)\in\mathcal{E}_T} v_{it}(1 - z_{it}) + \sum_{(i,j)\in\mathcal{E}} w_{ij} y_{ij} \right) \text{s.t. } \mathbf{y}' \in \text{AMWC}_{G'}. \tag{6.7}$$

where $\mathbf{y}' = (\mathbf{y}, \mathbf{z}) \in \mathcal{E}'$ and $\text{AMWC}_{G'}$ is the Asymmetric Multiway Cut polytope. Our convention is that $y_{ij} = 1$ stands for separated nodes i and j (the respective edge is cut) and $y_{ij} = 0$ stands for merged nodes i and j. $z_{it} = 0$ means that the internal node i is associated with the semantic class represented by T_t. $AMWC_{G'}$ is defined by the following linear constraints:

$$\sum_{(i,j)\in P} \mathbf{y}_{ij} \geq \mathbf{y}_{uv} \;\; \forall (u,v) \in \mathcal{E};\; \forall P \in \text{Path}(u,v) \subseteq \mathcal{E} \tag{6.8}$$

$$\sum_{t\in T} \mathbf{z}_{it} = |T| - 1 \;\; \forall i \in \mathcal{V}; \tag{6.9}$$

$$\mathbf{y}_{uv} \geq \mathbf{z}_{vt} - \mathbf{z}_{ut} \;\; \forall (u,v) \in \mathcal{E};\; t \in T \tag{6.10}$$

$$\mathbf{y}_{uv} \geq \mathbf{z}_{ut} - \mathbf{z}_{vt} \;\; \forall (u,v) \in \mathcal{E};\; t \in T \tag{6.11}$$

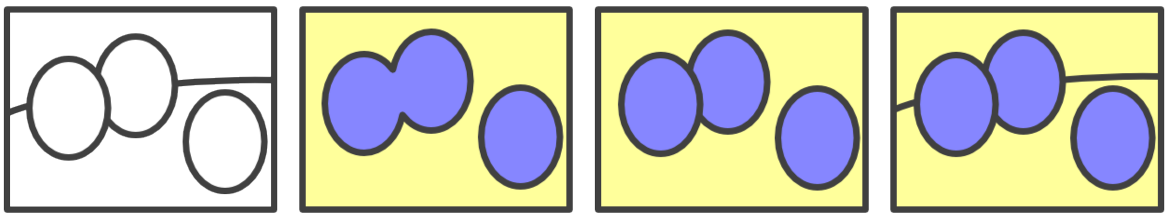
$$\mathbf{y}_{uv} \leq \mathbf{z}_{ut} + \mathbf{z}_{vt} \;\; \forall (u,v) \in \mathcal{E};\; t \in T_n \tag{6.12}$$

The cycle inequalities (6.8), also used in Eq. (3.11) and described in detail by Kappes et al. [74], ensure the consistency of the partitioning. Eq. (6.9) ensures the unique affiliation of each node with a terminal node. The consistency between semantic labeling and partition labeling is ensured by Eqs. (6.10) and (6.11). They state that whenever adjacent nodes are assigned to two different terminal nodes, there is an internal edge separating them and therefore they must belong to different partition elements. The constraints in Eq. (6.12) ensure that "cuts within" are not allowed for all semantic classes $t \in T_n$.

The weights v_{it} in Eq. (6.7) are equivalent to the unary energies in Eq. (6.5). Also the weights w_{ij}, which determine attraction or repulsion between the internal nodes i and j, are related to the energies in the node representation:

$$v_{it} = E_i(t); \;\; w_{ij} = -\log\left(\frac{p_{ij}}{1 - p_{ij}}\right).$$

While the weights w_{ij} are purely encoding pairwise affinities, the pairwise energies E_{ij} in the node representation also have to ensure the consistency between partitioning and labeling. Figure 6.2 (b) shows the visual fragmentation of the pairwise factors, while Figure 6.2 (d) shows how it can be done by higher-order AMWC factors.



(a) MC partitioning | (b) GC/MWC sem. segmentation | (c) AMWC sem. seg. + partitioning | (d) SC sem. seg. + partitioning

Figure 6.3: Possible results of the variants of the **A**symmetric **M**ulti**w**ay **C**ut algorithm. **(a)** partitioning Multicut algorithm. The image is divided into several segments with no semantic labels. **(b)** Pott's model, solved by **G**raphcut or **M**ulti**w**ay **C**ut. Blue and yellow indicate the semantic affiliation of the segments. It is not possible to have a subdivision of connected components of one semantic class. Unlike Graphcut, the Multiway Cut is able to handle multiple semantic classes and non-submodular potentials. **(c)** AMWC with "cuts within" only allowed in the foreground (blue) class. **(d)** **S**ymmetric **C**uts, with "cuts within" in all classes, used for neural segmentation later on.

6.2.3 Relation to Other Models

The AMWC framework of joint labeling and partitioning generalizes some well known algorithms.

For $\|T_c\| = 1$ and $\|T_n\| = 0$ we fall back to the case of the **Multicut** for unsupervised partitioning/correlation clustering problems (see chapter 3 and [74]). Here we omit all unary potentials/all semantic labels in the node formulation or equivalently we omit the constraints from Eqs. (6.10) to (6.12) in the edge formulation. The omission of all terminal edges $\mathcal{E}_T$ considerably reduces the size of the problem. For pure partitioning/correlation clustering problems this model provides a globally optimal solution. Figure 6.3 (a) shows an exemplary Multicut partitioning.

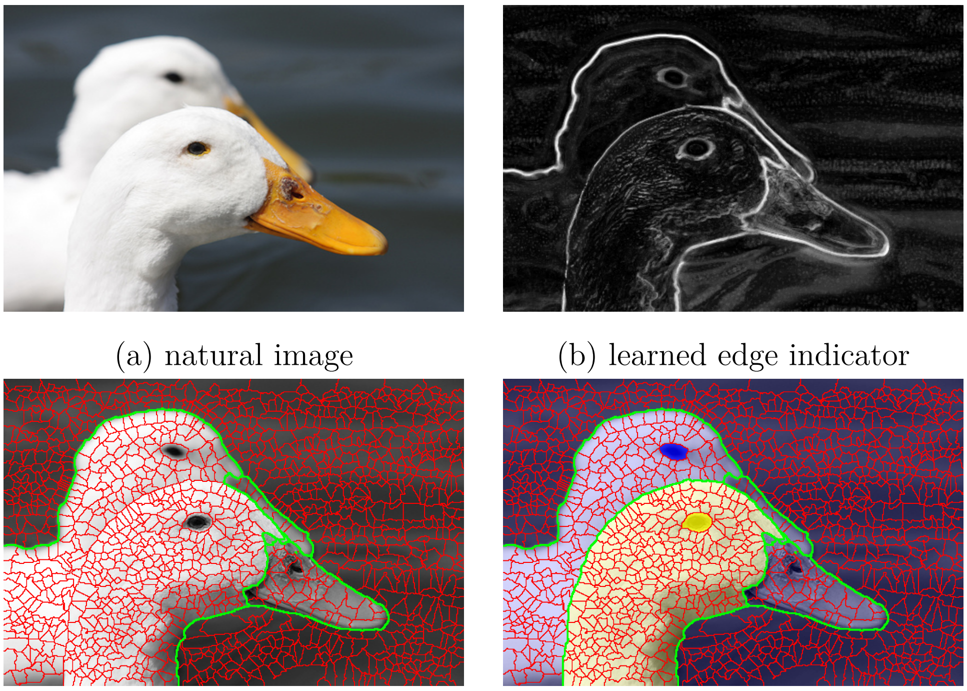
The **Multiway Cut** (as introduced by Boykov et al. [121]) is the special case of $\|T_c\| = 0$. All semantic classes are forbidden to have "cuts within". An exemplary problem from image analysis is an image labeling problem, where a single object is segmented from the background or where no instances of semantic classes are touching. While Figure 6.3 (c) and (d) could not be produced by MWC, (b) is a possible output.

If $\|T_c\| = 0$ and $\|T_n\| = 2$ and in addition there are no attractive pairwise potentials, this Pott's model can be solved by the popular **Graphcut** algorithm. Since in Figure 6.3 (b) only two semantic classes are used and no weights are specified, it could as well be produced by Graphcut.

If $\|T_c\| \neq 0$ and $\|T_n\| \neq 0$, only some semantic classes are allowed to have "cuts within", while the background does not have inner boundaries. This case is studied by Kröger et al. [120] and illustrated in Figure 6.3 (c).

The special case of $\|T_n\| = 0$ – the antagonist of the Multiway Cut – has not been discussed in detail before. Here all semantic classes are allowed to have "cuts within", so we will refer to it as "Symmetric Cuts". This variant is needed for segmenting images with no specific background class, such as neural EM volumes. A possible output of Symmetric Cuts is shown in Figure 6.3 (d).

In the classic setup of semantic segmentation, establishing the semantic class of every



(a) natural image (b) learned edge indicator

(c) Multicut result (d) AMWC result

Figure 6.4: An illustration of the active principle of the AMWC as an algorithm for partially seeded segmentation (seeded from the eyes). The original image (a) is provided by Batra et al. [122]. (b) shows a pixel-wise, learned edge indicator. (c) shows the segmentation that the Multicut is able to produce bases on the edgeweights only. In (d) we see that adding “semantic” information in two pixels only leads to a separation of the geese.

pixel is the overall objective of the algorithm. For AMWC this use case has been explored by Kröger et al. [120], with simultaneous neuron and mitochondria segmentation serving as a motivating example. The unaries v_{it} of Eq. (6.7) can then simply be derived from an output of a semantic classifier.

Complementary to the work of Kröger et al. [120] and more suiting to our immediate use case of neuron segmentation, I would like to explore the use of semantic affiliation of segments as a means to an end, as additional information for solving the partitioning problem. This information can be very sparse, with most nodes exhibiting no affinity to either semantic class. In a way, this problem is similar to seeded segmentation, where the information is propagated from the seeds to the segment boundaries. Our prior information is, however, much weaker: seeds of the same semantic class are present in multiple objects, which need to be separated in the final segmentation. Besides, many objects do not have seeds at all. Another similar setup is a partitioning Multicut with additional long range repulsive edges between some non-neighbor segments as introduced by Andres et al. [123]. This model is equivalent to AMWC, if not more than one seed is present for each semantic class, or if the costs of switching the semantic label of a node are infinite (hard constraints). Such constraints are easy to introduce if, for example, an expert user specifies that two nodes definitely belong to different objects in the final segmentation (in 3D, this task is very non-trivial). Our goal, however, is to derive the sparse semantic “seeds” and the corresponding costs of switching the node labels directly from the data. This limits the necessary user involvement to training a semantic classifier on a subset of data.

The predictions of this classifier do not need to represent the semantic class affinity directly. For example, in the neural segmentation use case, a classifier for vesicle clouds would indirectly predict that superpixels with a high response likely belong to an axon. The superpixels with a low response remain unassigned – as many axon superpixels are not covered by vesicle clouds, negative predictions of the classifier add no class affinity information. In the following we will refer to such predictions as "proxy probabilities". Consequently, we define p_i^s of Eq. (6.4) as

$$p_i^s = \max\left(0.5, \tilde{p}_i^s\right),$$

where $\tilde{p}_i^s$ is the proxy probability. With the unaries defined this way, the sparse semantic class information can propagate through the nodes with no class affinity (as no cost is payed to switch their semantic label to either class) and influence the cut decisions for an uncertain edge far away from the original "seeds". This property is especially important in our 3D application.

6.3 AMWC Workflow for Neuron Segmentation

This section describes the setup of the AMWC algorithm for the task of neural segmentation: the creation of supervoxels for the graph, the estimation of supervoxel edge affinities and the definition of semantic classes and their proxies.

6.3.1 Supervoxel Generation

The oversegmentation of the image volume into supervoxels is based on a pixel-wise membrane probability map. This probability map is denoised by the non-local means algorithm described by Budas et al. [124]. Local minima on the smoothed map serve as seeds for a watershed algorithm. The supervoxels, resulting from the watershed algorithm, are termed w-supervoxels. For the test block, I constructed 111 164 w-superpixels in total.

The w-supervoxels are coarse enough to compute the first set of pairwise features and to train a random forest classifier for the respective affinities. The GALA hierarchical clustering algorithm as introduced by Nunez-Iglesias et al. [69] does this in a convenient fashion and merges the supervoxel pairs iteratively. The authors recompute the probabilities for all newly emerging pairs. Since this greedy strategy cannot provide any guarantees, that the solution will be close to the globally optimal one, we do not want to use it to attain the final segmentation (cluster until there are no pairs of neighboring supervoxels with a positive affinity left). Instead, we only apply it for easy merges, where the algorithm can utilize its advantage of adapting edge-weights, and stop at a conservative threshold to avoid undersegmentation. After running GALA on our test block, there are 11 672 supervoxels left. In the following, we will refer to them as g-supervoxels.

The early stopping at a conservative threshold has multiple advantages: if we want to follow up with a globally optimal procedure such as AMWC we profit from a smaller adjacency graph in terms of runtime. Also, features accumulated on supervoxel surfaces become more expressive the larger the supervoxels get as I demonstrated already in section 2.4.2.

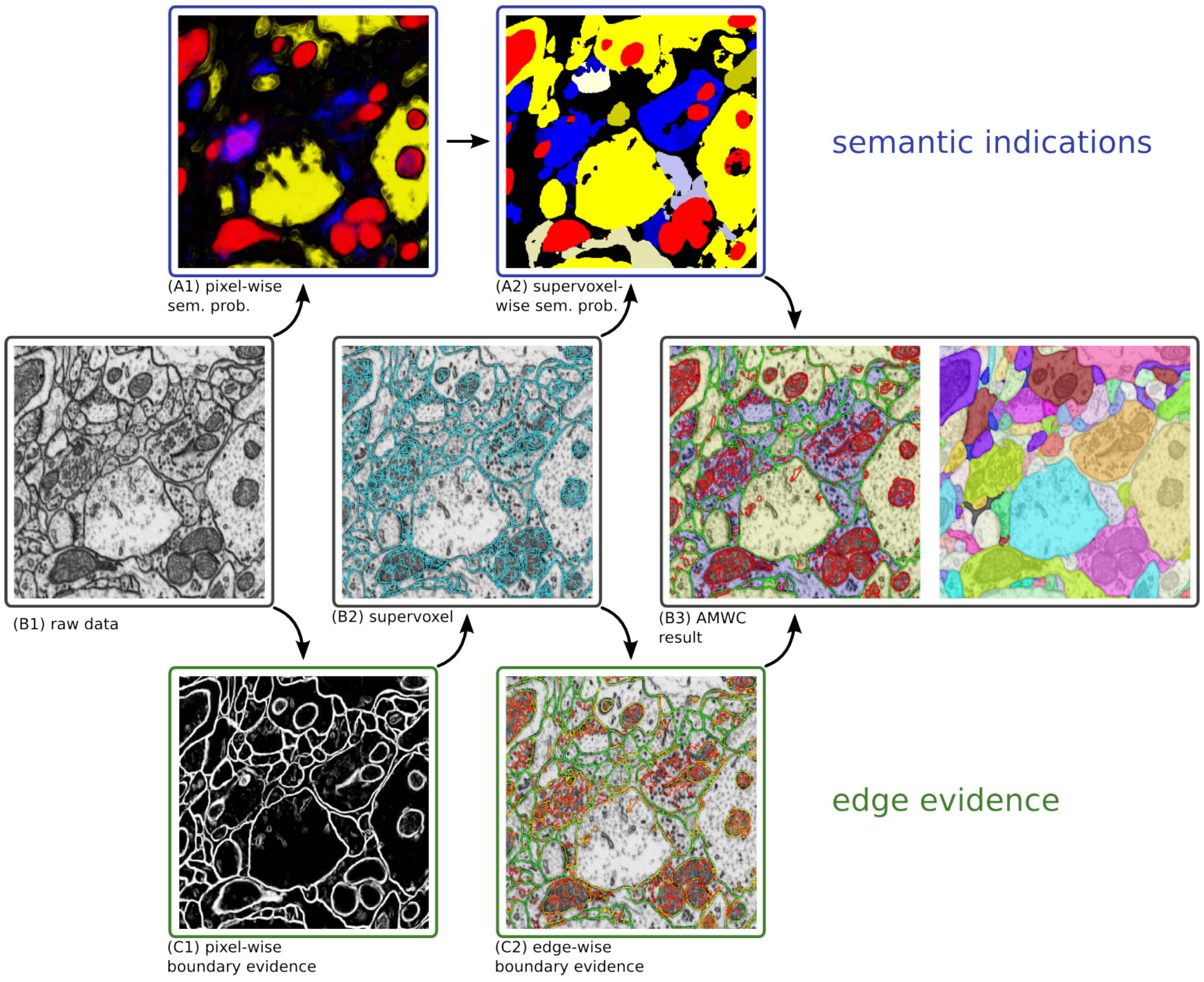

semantic indications
(A1) pixel-wise sem. prob.
(A2) supervoxel-wise sem. prob.
(B1) raw data
(B2) supervoxel
(B3) AMWC result
(C1) pixel-wise boundary evidence
(C2) edge-wise boundary evidence
edge evidence

Figure 6.5: The flow diagram for the presented segmentation pipeline, with consecutive steps going from left to right. The starting point (B1) in the central row to the left is the raw EM volume described in appendix A.1. The intermediate image volumes (A1), (C1) and (C2) rely on predictions of a random forest classifier: (A1) and (C1) show the voxel-wise semantic and membrane probability. (C2) shows the pairwise split probability projected on the touching faces of the g-supervoxel oversegmentation (see 6.3.2 for details). The edge color represents how likely the edge is to remain in the final segmentation, with green indicating the most likely and red indicating the most unlikely edges. (B2) shows the GALA supervoxels as described in 6.3.1. The big GALA supervoxels allow for expressive features and therefore good classification of the faces. The mapping from the pixel-wise semantic probabilities to the supervoxel level is described in 6.3.2. The core AMWC algorithm is described in full detail in 6.2. Its result, a jointly optimized semantic labeling and a partitioning is represented in (B3). For reference, (B3) also shows the final segmentation, where each object is assigned a different color.

6.3.2 Probabilities

Both the GALA algorithm and the subsequent AMWC base their merge/split decisions on the pairwise affinities of supervoxels. I compute these via a random forest classifier, which determines whether a touching face of two supervoxels is representing a membrane or not. The classification is based on statistics of pixel feature values in the direct proximity of the faces and on sorted statistics of the whole volumes of the two supervoxel neighbors.

The pixel-wise features are composed of the following filter responses: Gaussian smoothing, Gaussian gradient magnitude, the Hessian of Gaussian eigenvalues and Laplacian of Gaussian as explained in section 2.1.3. They are applied on scales 1.6, 4.2 and 8.3. The following statistics are computed along the faces (all voxels in neuron i/j, which have a voxel in neuron j/i in their 6-neighborhood): mean, variance, quantiles (25%, 75%), kurtosis median, skewness and size of the faces. Besides those, histogram of grayvalues, voxel count, kurtosis, maximum, minimum, quantiles (10%, 25%, 50%, 75%, 90%), radii of supervoxel, skewness, sum and variance are computed on each of the volumes of two adjacent supervoxels. All feature pairs can be translated to features of the touching faces by applying: min(. , .), max(. , .) and ‖difference(. , .)‖. This avoids an ambiguity in ordering of the plain feature pairs of the two touching supervoxels.

In mammalian cortex, a clear separation exists between the axon and the dendrite of a neuron (see Figure 6.6). Therefore, and since most current high-resolution image volumes do not encompass the soma, axonic and dendritic neurites inside the volume must be disjoint[1]. Consequently, if for every supervoxel in the volume we knew whether it belongs to an axon or a dendrite, edges between axons and dendrites would be preserved regardless of the strength of the edge indicator. For most supervoxels this information is, however, not available and their axonic/dendritic affiliation is not clear even for expert neuroscientists. Nevertheless, sparse affiliation indicators do exist and the AMWC framework allows us to benefit even from a small number of supervoxels, where these indicators can be found. The particular indicators we use are listed below:

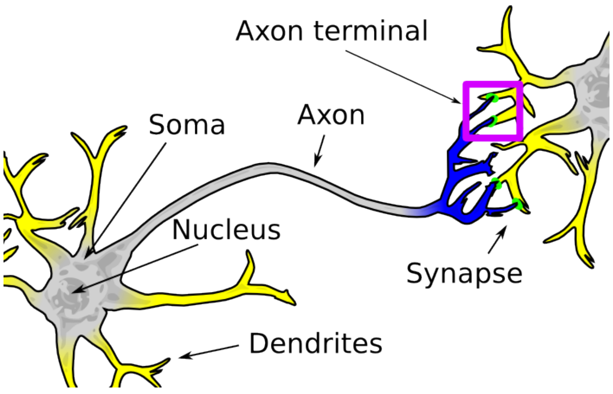


Figure 6.6: Two mammalian neurons and their synaptic connections (green). The pink rectangle sketches qualitatively our field of view. The yellow/blue parts are the two semantic classes used to distinguish neurons. (Figure based on [125])

- **Synapses**. Most chemical synapses in the mammalian cortex are between an axon and a dendrite or dendritic spine. Spines are small protrusions from the dendrite and receive the majority of excitatory contacts. Given that synapses in FIB/SEM data can be found automatically along with their direction ([19, 20]), we can assign supervoxels, touching the synapse on the presynaptic side to axons and on the post-synaptic side to dendrites.

[1]Note that this does not restrict this model from scaling up to bigger volumes. If somas are present in the data, they just have to be included in the modeling as a separate semantic class.

- **Vesicle clouds**. These are mostly found in axons, as they contain neurotransmitters for synaptic transmission. We train a pixel classifier to detect vesicles and discard detections that are not part of a cloud.

- **Dendritic cytoplasm**. Dendritic shafts can become significantly thicker than axons. We train another pixel classifier for the cytoplasm of the dendritic shafts and discard all small connected components of the resulting segmentation.

The mapping of proxy probabilities from pixels to g-supervoxels is performed in two steps. In the first stage we take the mean pixel value for each w-supervoxel. In the second step we take the maximum of all the values from the w-supervoxels that compose the respective g-supervoxel. The mean operation results in lower vulnerability to noise and the maximum operation expands the volume for which we have the class information available.

Along with axonic and dendritic priors described above, we find it beneficial to separate the mitochondria in a semantic class of their own. Remember the confusion caused by mitochindria already described in section 2.1 and visible in Figure 2.2. In addition, false merging errors often occur at locations, where mitochondria are situated very close to the neuron membrane (examples for such problematic situations can be found in Figure 6.9, second and third row). Introduction of the “mitochondrion” semantic class helps the algorithm to resolve the cases, where a mitochondrion membrane is not distinguishable from the cell membrane. The prior for mitochondria is also derived from the pixel-wise mitochondria probability map, but, since mitochondria are directly observable, a simple averaging over the g-supervoxels is sufficient. Mitochondria are merged back with their respective neurons by a post-processing step, described in section 8.

Obviously, in a volume of neural tissue an axon can touch another axon and a dendrite can touch another dendrite. Therefore, I choose the Symmetric Cut variant of the AMWC as introduced before in section 6.2.3, where “cuts within” are allowed for all semantic classes.

6.3.3 Optimization

We use the open source library OpenGM [105] together with the commercial optimization software package CPLEX from IBM for the actual inference. OpenGM allows for the simple formulation of the problem as a graphical model and performs the transformation into an explicit integer linear program formulation automatically. Technical details of the optimization procedure can be found in the work of Kappes et al. [92].

6.3.4 Post-Processing

We aim for a segmentation of neurons. We use Mitochondria as an additional semantic class and therefore cause their separation in the partitioning. The additional Mitochondria segments will not harm any connectomes that are derived from the segmentation. In order to do this a Mitochondrion would have to completely plug a thin neural process. It is harmful though when trying to determine the segmentation quality with respect to a neuron segmentation ground truth with the help of the Rand index and the variation of information described in section 2.6. Therefore I introduce a way to merge the Mitochondria segments in the respective neuron in chapter 8. The quality measures reported in Table 6.1 are all relating to the post processed segmentations.

	w-GALA	g-MC	g-AMWC	GT
RI	0.9907	0.9956	0.9965	1.0
RI_w	0.9918	0.9967	0.9977	0.9989
RI_g	0.9920	0.9972	0.9981	0.9987
VI	0.6971	0.5745	0.5421	0.0
VI_w	0.4693	0.3494	0.3105	0.2177
VI_g	0.4282	0.2874	0.2490	0.2702

Table 6.1: Segmentation quality of the GALA algorithm with watershed supervoxels (w-GALA), the Multicut algorithm with GALA supervoxels (g-MC) and the Asymmetric Multiway Cut algorithm with GALA supervoxels (g-AMWC), as measured by the Rand index (RI, higher is better) and variation of information (VI, lower is better), against pixel-wise ground truth. RI_w and VI_w show the comparison with the ground truth projection on w-supervoxels, RI_g and VI_g – on g-supervoxels. To illustrate the ratio of supervoxel errors to segmentation algorithm errors, the last column (GT) compares pixel-wise ground truth with its projections on w- and g-supervoxels.

6.4 Experiments

The presented workflow has been tested on a FIB/SEM dataset from adult mouse somatosensory cortex with approximately isotropic resolution of $5 \times 5 \times 6$ nm (for more details see appendix A.1). The dense volumetric ground truth segmentation of the dataset was created by an expert neuroscientist using the carving algorithm from ilastik [47, 48] and the connectome annotation tool Mojo 2.0 [65] mentioned in section 1.3.

The first block of $900 \times 900 \times 200$ px is used to train the random forests. The pixel-wise classifiers for the semantic classes and for the membrane/non-membrane predictions are trained on sparse user labels in ilastik [48]. The random forests that predict the merge probability of neighboring supervoxels are trained using a projection of the dense, pixel-wise ground truth on the respective supervoxels. The projection is done in a way that the projected ground truth respects the supervoxel boundaries, by an argmax operation on all voxel-wise neuron ids within the voxel composing a supervoxel. All the tests have been performed on a dataset of $700 \times 700 \times 700$ px, adjacent to the training set, but not overlapping with it.

Our experiments aim to establish if the globally optimal approach of the Multicut algorithm is beneficial despite the increased CPU-load and if, in their turn, the semantic classes, introduced for the AMWC algorithm, carry enough additional information to improve on the standard partitioning Multicut. For the first point, we compare to the popular GALA hierarchical clustering algorithm. Table 6.1 summarizes the results of our three experiments:

1. GALA segmentation on w-supervoxels (w-GALA)

2. MC on g-supervoxels (g-MC)

3. AMWC on g-supervoxels (g-AMWC)

It is infeasible to run MC/AMWC on small w-supervoxels Running GALA on g-supevoxels is equivalent to a single run from w-supervoxels to a higher threshold.

6.5 Discussion

The comparison of GALA with both MC and AMWC in Table 6.1 demonstrates the advantage of global optimal merge decisions. In the commonly used quality measures for segmentation, Rand index and variation of information (see section 2.6), Multicut clearly outperforms the greedy GALA algorithm on our test data.

As for the additional semantic classes, the comparison between Multicut and AMWC, using the same supervoxels and the same merge probabilities, shows that the semantic information is indeed complementary to our boundary evidence and brings a significant benefit, even when the priors are sparse. Figure 6.9 illustrates this conclusion by four difficult examples, correctly solved by AMWC, but not by the Multicut.

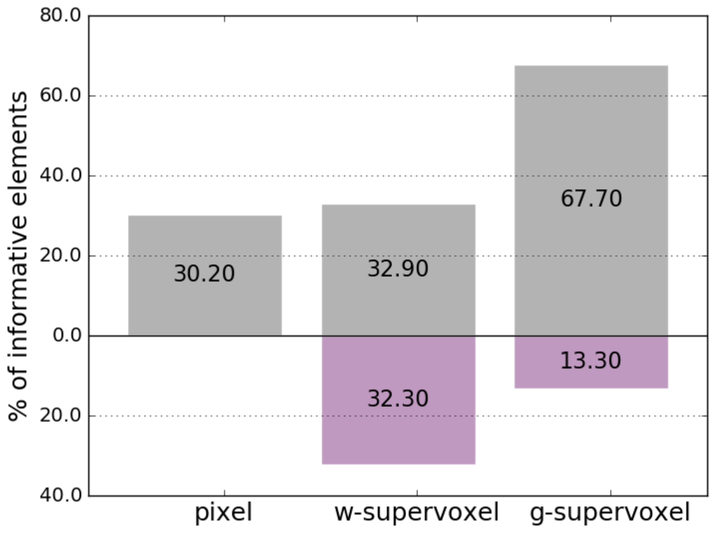
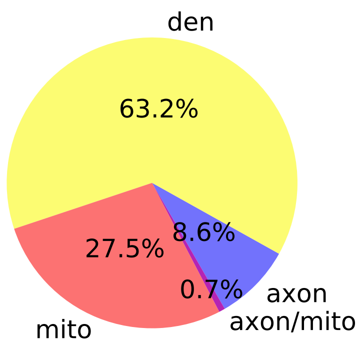


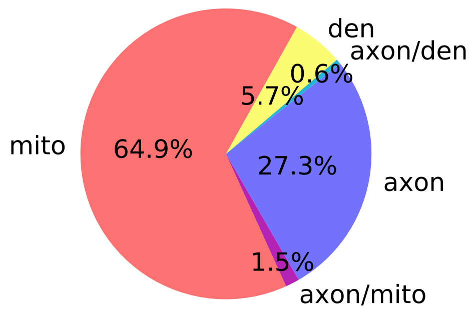
Figure 6.7: Percentage of elements with prior knowledge ($p > 0.5$). The grey bars show the *volumetric* coverage in the testblock (how many pixels have prior information): original pixel-wise priors (left), projected on w-supervoxels (center), projected on g-supervoxels (right). The purple bars show the *node-wise* coverage in percent (supervoxels with prior information): for w-supervoxels (center) and for g-supervoxels (right).

The RI_w, VI_w, RI_g and VI_g rows of Table 6.1 compare all three algorithms with the ground truth projection on w- and g-supervoxels. Since both the segmentation and the ground truth respect the boundaries given by the supervoxels, this comparison minimizes the effect of insignificant boundary shifts and shows more of the serious, topology-changing errors. For the AMWC algorithm, such errors are roughly of the same magnitude as inaccuracies of the boundaries.

Now that the benefit of the introduction of semantic class priors is clear, let us take a more detailed look at their distribution in the data. As described in 6.3.2, for “axon” and “dendrite” classes the priors are sparse and indirect: the absence of vesicles, for example, says nothing about the affinity of a supervoxel to the axonic or dendritic class. Figure 6.7 demonstrates, just how sparse our prior knowledge is on the pixel level and how the volume coverage of the priors increases as we propagate them to w- and g-supervoxels. It is interesting to note, that the volumetric information gain, which comes from the max-projection of the priors from w- to g-supervoxels, is not reflected on the node level



(a) volume-wise



(b) node-wise

Figure 6.8: Class-wise composition of elements with prior knowledge from Figure 6.7: pixel-wise (left), supervoxel-wise (right, for g-supervoxels used in AMWC). Elements with conflicting information are marked as overlap between the main classes in red, blue and yellow. For instance, in (b) 29.4% of all informative nodes (g-supervoxel) vote for the axon class (blue) and 27.3% do so without local contradiction.

(compare the grey/purple bars in Figure 6.7). In fact, we observe a percental decrease of semantically conclusive supervoxels after GALA clustering.

Seeking to interpret this decrease, I studied the class-wise composition of the priors (Figure 6.8). The “dendrite” class contains most of the informative pixels (Figure 6.8 (a)). However, most of the easy merge decisions, which are delegated to GALA, are made in the “empty” dendritic shafts, and since the dendrite supervoxels are relatively large, their absolute number (Figure 6.8 (b)) is small.

Figure 6.8 also shows that co-occurence of priors is not very common. In the rare cases where it happens it does not necessarily harm the segmentation. If all priors had maximal amplitude, all assignments to semantic classes would be of the same cost and we would fall back to a prior-less situation of the Multicut.

It is worth mentioning that the runtime of the AMWC as used here excels the one of the Multicut considerably. While the solution of the Multicut problem on a 700^3 pixel plock (described in appendix A.1.2), given the g-superpixels, took 262 s, the Asymmetric Multiway Cut took 4898 s[2].

The fact that AMWC can make use of sparse priors, where most of the data exhibits no information on class affinity, suggests that the algorithm could be applied to other problems, where the individuals to be segmented differ only by a small part or by a manually introduced seed. Another example from the bioimaging domain could be the segmentation of cells with different nuclei appearance or, for real world images, segmentation of airplanes with different tail logos or birds with different beaks.

6.6 Conclusion

We introduced a principled approach to incorporate high-level biological priors into the Multicut neuron segmentation procedure of Andres et al. [73]. While the Multicut algorithm is already superior to greedy approaches due to its globally optimal nature, we show that it can be further improved by considering sparse global priors of neuron type in

[2]On an Intel Xeon E5-2650 v3

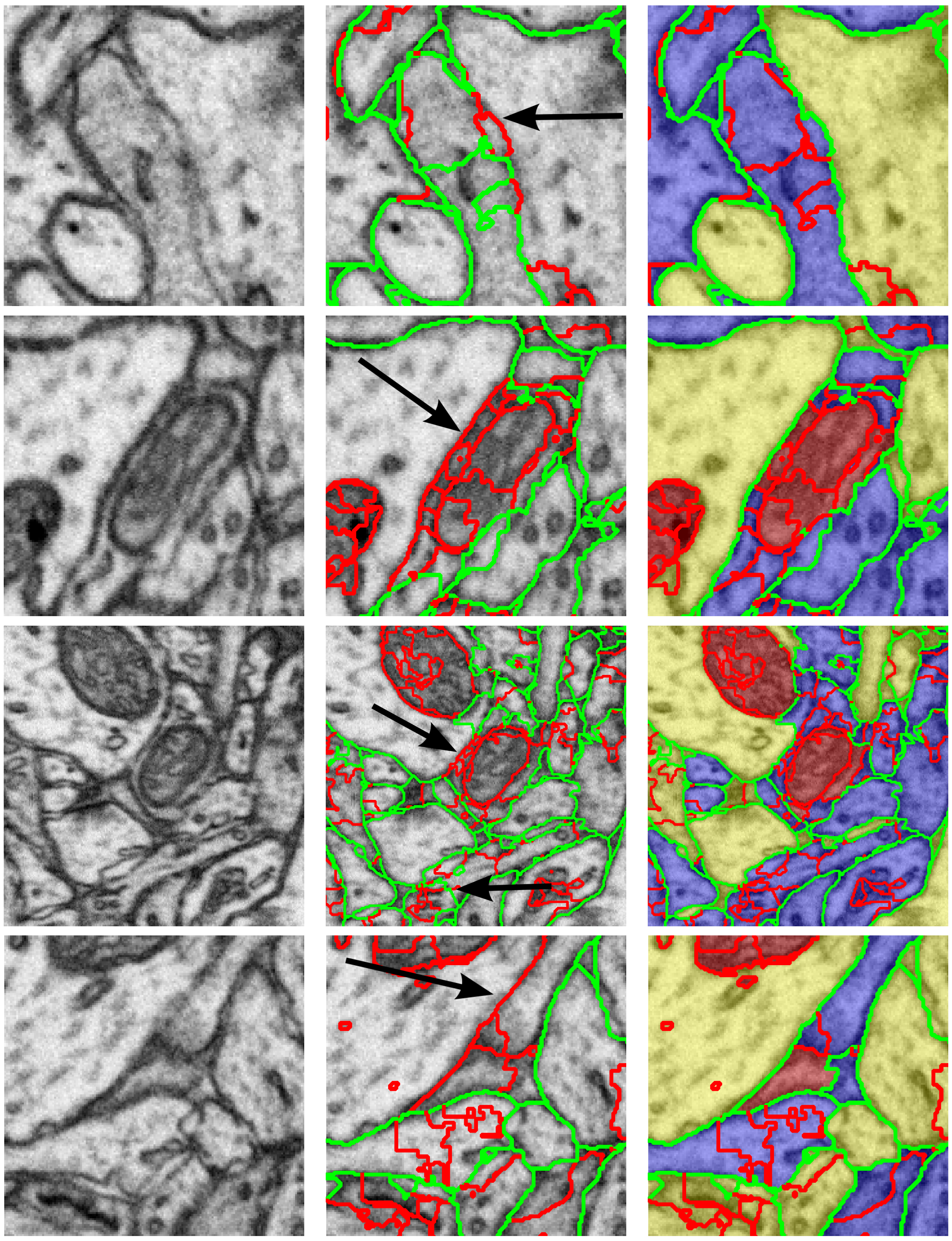


Figure 6.9: Left: difficult locations in the raw data. Center: Multicut fails to correctly close the edge, pointed out by the arrows. Right: Asymmetric Multiway Cut overcomes this problem. The red supervoxel edges are merged in the final segmentation, while the green edges remain “on”. In the AMWC result colors of the final segments denote their semantic class membership: blue for axon, yellow for dendrite, red for mitochondria.

addition to local boundary evidence. The formulation of the partitioning problem in the edge domain allows us to find – by a cutting-planes optimization procedure – a globally optimal segmentation with respect to both the semantic and the transition probabilities. The respective increase of the computation time is counteracted by introduction of very large supervoxels, which also serve to propagate the sparse voxel-wise prior information to larger parts of the image volume.

Segmentation of neurons with semantic differentiation between axons and dendrites is only the first of many possible areas of AMWC application. In the future an extension to other biological classes or other proxies for existing semantic classes is possible. This is especially important to enable the use of the algorithm for non-mammalian neural tissue without strong axon/dendrite differentiation.

The AMWC algorithm itself can potentially be generalized further. Eqs. (6.8) to (6.12) are ensuring the consistency of semantic labels and partition labels. Concretely they do enforce a partition transition if a semantic transition is apparent. What would happen if these constraints were omitted for some semantic class transitions? In Eqs. (6.8) to (6.12) one would have to introduce two different indices for the terminal nodes and a matrix describing the relations between the classes must be introduced. This way one could introduce an other level of asymmetry. In the case of neuron segmentation situations like this could be modeled: Axons and dendrites are not allowed to merge but both axon and dendrite are allowed to merge with a class representing the cell body. With this approach also other interesting relationships could be modeled. By introduction of semantic classes without unary evidence but with clear rules on their merging behavior one could enforce a certain minimal graph distance between two other semantic class detections.

Finally the decomposition of the Asymmetric Multiway Cut can be examined in the future. The same scheme that was introduced for the decomposition of the Multicut problem can be applied here since the additional constraints enforcing the consistency of semantic- and partition label are purely local as well. This can be seen in Figure 6.2 (d). This can help to scale up the approach to bigger image volumes.

Chapter 7

Semantic Agglomerative Clustering

7.1 Methods

As seen in chapter 4, Agglomerative Clustering can be done according to edge-weights, according to node-weights (together with a metric) or according to both, in a weighted way. Since the metric is free to choose, all kind of settings with dense unary information can be handeled. Whenever the unaries of two neighboring nodes are expressive, this assists the clustering. Simulating the setting of the Asymmetric Cuts from section 6.3, where only a subset of the nodes has unary information, with a vanilla Agglomerative Clustering algorithm is not possible though. In the Asymmetric Cuts setting the requirement that all nodes eventually need a semantic label leads to the spreading of the semantic classes from the respective seeds of semantic information. Therefore also two neighboring nodes that initially both do not have semantic information can be aided in their merge decision.

We will present an addition to the standard Agglomerative Clustering that enables its employment also in situations with sparse unary information like the semantic differentiation explained in section 6.3.2: Label Propagation. While the AMWC can handle probabilistic affiliations, the Semantic Agglomerative Clustering must decide for a semantic class immediately and can not wait until the end of the clustering procedure to ponder possibilities globally. This means that an initial decision on the class label must be taken. The assumption from chapter 6 was that superpixels within the same partition will necessarily belong to the same semantic class. The proposition is now to modify the sequencing-weights (as introduced in section 4.2.2) in the following way:

1. If two neighboring nodes i and j have both no semantic label, leave the sequencing-weights for edge (i, j) untouched.

2. In the case where only one of the neighboring nodes i and j hold a semantic class it is possible but not necessary to adjust the edge-weight. There are situations where a growing behavior from the seeds is desirable. For these cases a respective modification of the edge-weights can be done here.

3. For the third possible case of both neighboring nodes i and j, holding a semantic label, the enduing edge modification depends on the equality of both labels:

   different In case both labels differ, modify the edge-weights such that a merge will not be performed.

identical If both semantic labels are identical, the question is whether for this semantic class cuts within should be allowed (see also section 6.2.1 where the cuts within are introduced).

(a) If cuts within are allowed, leave the edge-weights untouched.

(b) If not, change the sequencing-weights in a way that a merge is inevitable.

Since we do expect parts of neurons within a given field of view that do not have skeletal information we do not prioritize merge decisions with labeled nodes (point 2. in the following). The propagation of the label information that was implicitly done in chapter 6 due to the globally optimal nature of the algorithm is here now done explicitly:

1. If neither node i nor node j hold a semantic label, the node that results from a merge does not hold a semantic label either.

2. If either i or j holds a semantic label, the merged node consisting of i and j holds this label as well.

3. The only case in which two nodes both holding a semantic label can merge is when their semantic label is identical. Their union leads to a node of the same label.

In contrast to the Asymmetric Multiway Cut, not all elements must end up with a semantic label. Only the ones connected with a seed will be labeled. Unlabeled regions fall back to the non-semantic case of Agglomerative Clustering.

7.2 Experiment: Including Skeleton Information

The following experiment evaluated the inclusion of sparse region-wise information in the clustering procedure.

The data (described in detail int eh appendix A.2) is originating from a ssTEM aperture (see section 1.2.1). Therefore the resolution is anisotropic (3.8 nm in the imaging plane and a slice thickness of 50.0 nm). The merge decisions between superpixels of different slices are delicate and therefore we will not leave them to the initial watershed constructing the oversegmentation (see section 2.3). Therefore all used supervoxels are flat – All pixels within have the same z-coordinate. In Figure 7.1 (b) the superpixel faces parallel to the imaging plane, as can be seen in a region-wise shading while the orthogonal ones are lines. The color is encoding the merge probability. All presented algorithms so far worked on general superpixels. Therefore it is reasonable to assume that they work on flat superpixels as well.

This data set differentiates itself form the data used e.g. in chapter 6 not only by its resolution but also by the availability of ground truth. While for the isotropic data used for the Asymmetric Multiway Cut (described in appendix A.1) dense segmentation ground truth is available, here one has can only rely on skeletons. This means that a very small subset of pixels is annotated, with individual neuron ids, by a human expert. Images in the first row of Figure 7.3 color the superpixels which contain skeleton pixels. Even on a relatively small problem instance of $500 \times 500 \times 6$ pixels with 7338 watershed superpixels (as e.g. shown in Figure 7.2) neither the multi-cut nor the Asymmetric Multiway Cut terminate within two days. The underlying edge-weights have been trained by sparse man-made labels (659 out of 51519 possible faces have been labeled).

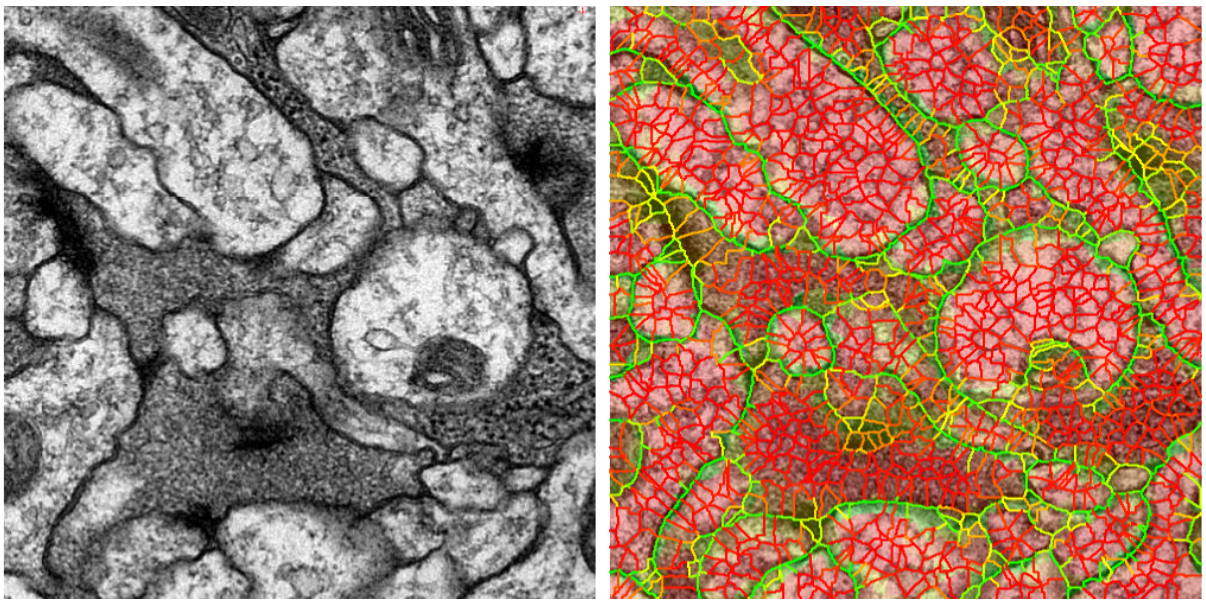

(a) (b)

Figure 7.1: An subset of the raw data (describes in appendix A.2) in the imaging plane (a) and the interactively learned face probabilities in the image plane (lines) as well as between the slices (regions) in (b).

Visually the probabilities of the faces parallel to the imaging plane are inferior to the orthogonal ones. Since no segmentation ground truth is available, a quantitative evaluation is restricted to the pixels for which the skeleton information is accessible. Consequently the evaluation of the error metrics Rand index $\mathrm{RI}_{\mathrm{sk}}$ and variation of information $\mathrm{VI}_{\mathrm{sk}}$ is based on the skeleton pixels only. This is mainly a measure for the topology of the neurons and is indifferent w.r.t deviations at the boundary of the neurons.

The first step is to compare vanilla Agglomerative Clustering for pure 2D segmentation and 3D segmentation in the following way:

1. Do Agglomerative Clustering on the faces in the imaging plane according to the shown probabilities according to Figure 7.1 (b) for each slice individually. For evaluation compare the slices individually ($\mathrm{RI}_{\mathrm{sk}}^{2D}$, $\mathrm{VI}_{\mathrm{sk}}^{2D}$).

2. Do a full 3D Agglomerative Clustering allowing both merges in the imaging plane as well as merges between the slices. For the evaluation also examine only 2D slices individually ($\mathrm{RI}_{\mathrm{sk}}^{2D}$, $\mathrm{VI}_{\mathrm{sk}}^{2D}$).

The quantitative evaluation is presented in Table 7.1. It basically shows that the 2D segmentation is easier than the additional 3D segmentation.

	$\mathrm{RI}_{\mathrm{sk}}^{2D} \uparrow$	$\mathrm{VI}_{\mathrm{sk}}^{2D} \downarrow$
1. ac^{2D}	0.92647 ± 0.01822	0.94000 ± 0.22248
2. ac^{3D}	0.79509 ± 0.04606	1.65246 ± 0.19758

Table 7.1: Comparison of a pure two dimensional Agglomerative Clustering ac^{2D} and the 3D variant.

It is concluded that the quality of the predictions of the faces parallel to the imaging plane is worse than the quality of the orthogonal faces. This is to be expected due to

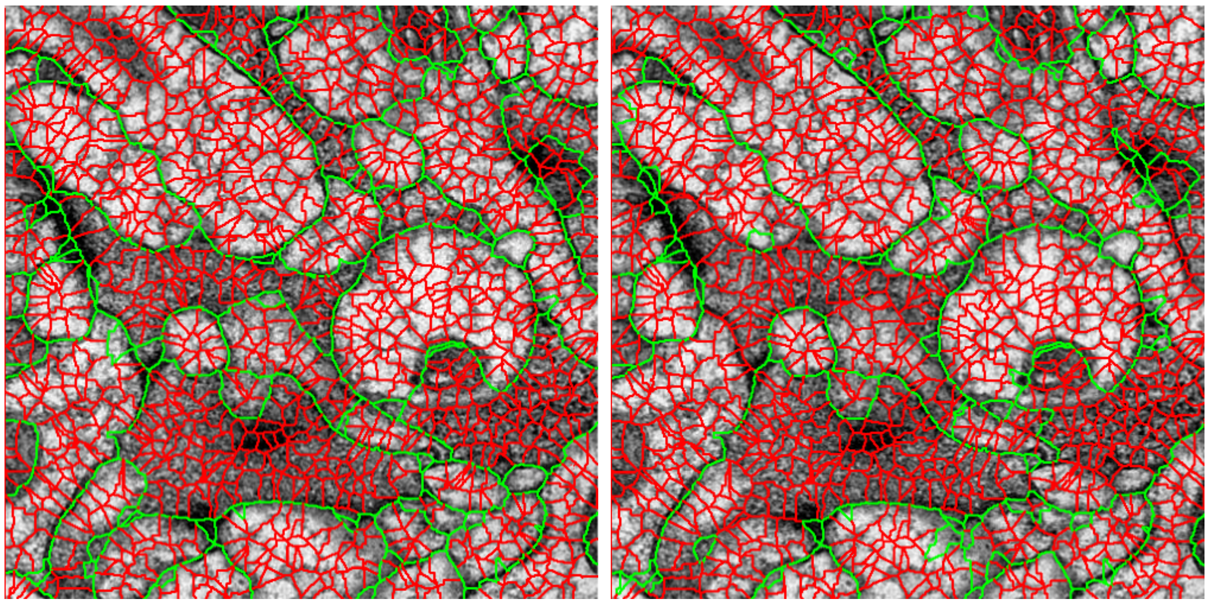

(a) 2D Agglomerative Clustering (b) 3D Agglomerative Clustering

Figure 7.2: A visual comparison of standard Agglomerative Clustering without allowed mergers in the slicing direction (a) and the case where merge decisions in the third dimension are treated equally (b) according to the learned merging probability as shown in Figure 7.1 (b). The underlying data is described in appendix A.2.

the differences in the image resolution. Visually one can see problems in 2D arising from taking the third dimension into account in Figure 7.2. Insufficient quality in the flat faces oftentimes leads to false merges. Furthermore false splits can be observed.

The next question is how the consideration of the skeletal information, as presented in section 7.1, is aiding the segmentation. Since skeleton information is induced in the clustering process, evaluating on it will not be informative. Evaluating the gained precision in the exact course of the cell boundaries by adding the skeletal information must be left solely to visual inspection. One way to evaluate how much potential for corrections based on the skeletal information there is, is to test the vanilla 3D semantic segmentation ac^{3D} on the full 3D skeleton ($\mathrm{RI}^{3D}_{\mathrm{sk}}$ and $\mathrm{VI}^{3D}_{\mathrm{sk}}$). The results of this experiment are condensed in Table 7.2. Note that the quality measures are applied on the resulting spatial partitioning, not on the resulting semantic labeling. If only spatially connected segments do have the same label and if the skeleton pixels are not connected, this can lead to disconnected partitions of the same semantic class. Note also that connectedness of semantic classes can not be enforced by the presented procedure. If one compares not the resulting partitioning but the resulting semantic labeling, prefect scores both of the Rand index $\mathrm{RI}^{3D}_{\mathrm{sk}}$ and the variation of information restricted to skeleton pixels in 3D $\mathrm{VI}^{3D}_{\mathrm{sk}}$ would be assured.

	$\mathrm{RI}^{3D}_{\mathrm{sk}} \uparrow$	$\mathrm{VI}^{3D}_{\mathrm{sk}} \downarrow$
ac^{3D}	0.82741	1.85323
sac^{3D}	0.97279	0.41727
$\mathrm{sac}^{3D}_{\mathrm{dt}}$	0.97689	0.35039

Table 7.2: Comparison of the Agglomerative Clustering without any semantic information in 3D ac^{3D} and two variants of the Semantic Agglomerative Clustering sac^{3D} and $\mathrm{sac}^{3D}_{\mathrm{dt}}$ also natively in 3D.

One can see in Table 7.2 as well as in Figure 7.3 how the clustering procedure translates the sparse skeleton information into a more plausible dense segmentation. In section 7.1, where the basic principles of brining semantic labels to Agglomerative Clustering are presented, the potential of favoring the merging with already labeled segments is mentioned. One argument in favor of this is, that the skeleton lines most often lay in the middle of the neurons where most merge decisions are easy. It can be argued that merging the easy decisions in the neurons' centers first is beneficial for the critical decisions on the neurons' boundaries. Due to previous merge actions, when it eventually comes to the merge decisions close to the cell boundaries, the faces in question are bigger. Therefore the weights are more expressive (see chapter 2.4.2). I already argued why prioritizing semantic merges is problematic for partially seeded segmentation.

Nevertheless the desired behavior can be mimicked. Section 2.1 shows how to end up with a pixel-wise membrane probability. Therefore one has an admittedly imperfect membrane estimation. It is now possible to binarize this by thresholding at 0.5 and calculating a distance transform on it. These pixel-wise distance values can be accumulated along the faces and their mean value can be determined. Now one has an estimate on how close to the cell membrane a particular face is. The new clustering weight will now be the mean of the normalized distance and the locally estimated merge probability – probable merges far from any membrane evidence will be performed first. In Table 7.2 and Figure 7.3 this approach is abbreviated with $\mathrm{sac}^{3D}_{\mathrm{dt}}$. sac^{3D} denotes the approach where the cluster probabilities are only modified via a small size regularization of $\theta = 0.01$ (as explained in 4.2.1). The prioritizing of faces far from membrane evidence is beneficial for the segmentation quality.

7.3 Conclusion

This chapter introduces a way how to include sparse semantic classes, related to the Asymmetric Multiway Cut from chapter 6, into the Agglomerative Clustering procedure. It is possible to introduce a large amount of semantic classes without any measurable effects on the runtime (see section 9.2). This enables the usage of Agglomerative Clustering when working with already skeletonized neural data as done in the previous section 7.2. In addition the fast procedure allows for an interactive approach where e.g. human annotators put seeds in single neurons and correct their segmentations where necessary.

The inclusion of the semantic information is only possible because of the new stopping criterion introduced in section 4.2.2. As mentioned before the metric on how to include the semantic information can freely be chosen. So far, I handle only binary decisions for class affiliations. The rules for propagating this information in case of sparsity are intuitive (section 7.1). In principle, it is possible to influence the merge decision based on probabilities instead of constraining it. Therefore one has to determine how different probabilities from different segments are merged into a new segment. To avoid handcrafting rules one could relearn the semantic affiliations of segments in the same way as the GALA algorithm relearns the edgewise merge probabilities (presented in [84]). This way one could emulate the behavior of the Asymmetric Multiway Cut, that is able to also consider semantic probabilities, more closely.

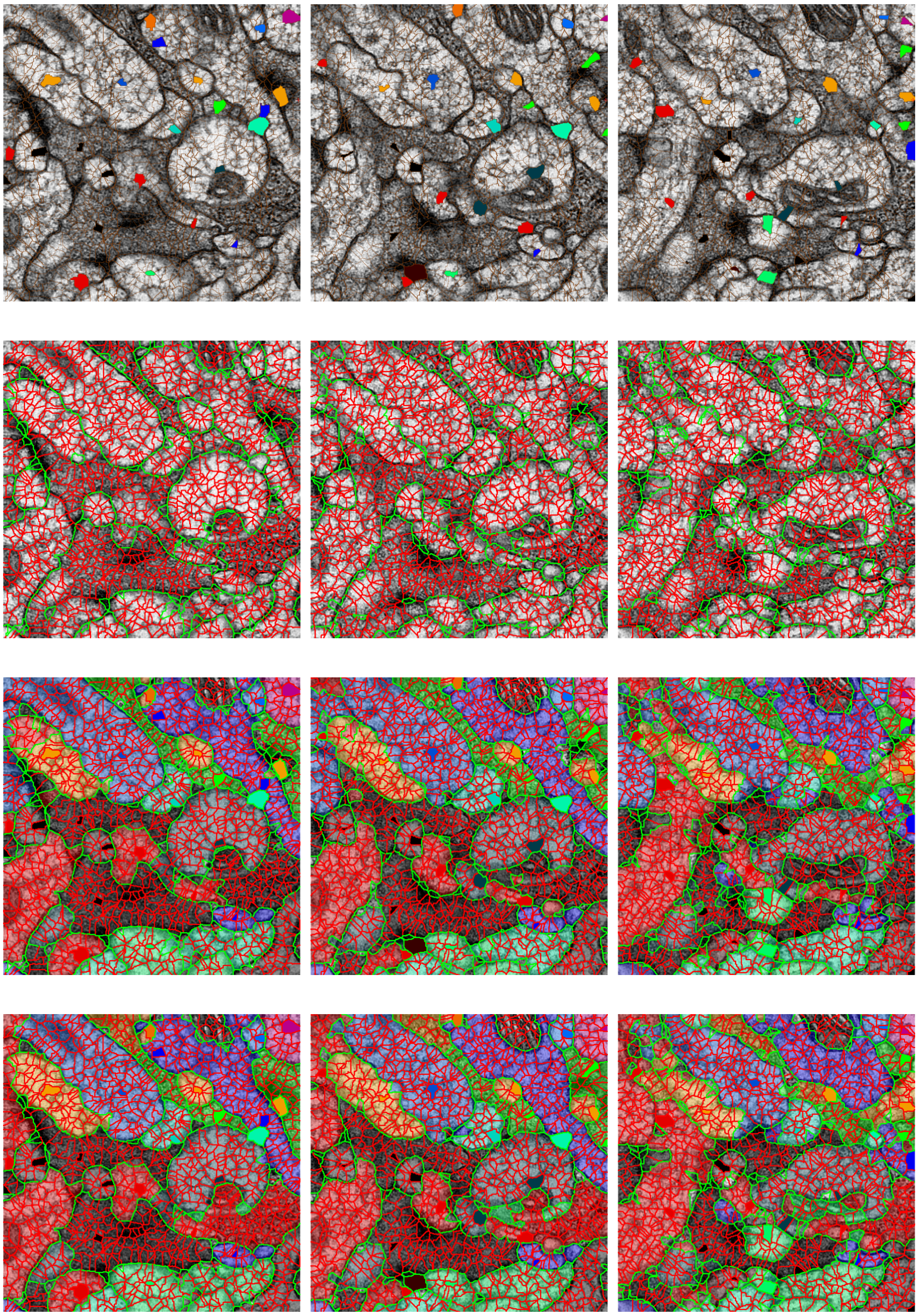


Figure 7.3: The three columns show consecutive slices of the data. Row one shows the raw data (as described in appendix A.2) in the imaging plane as well as color-coded the superpixles that contain a skeleton element. The second one depicts the results of a standard Agglomerative Clustering in 3D. The last two rows show two different approaches of Semantic Agglomerative Clustering without cuts within (sac^{3D} and $\mathrm{sac}^{3D}_{\mathrm{dt}}$).

Chapter 8

Post-Pocessing

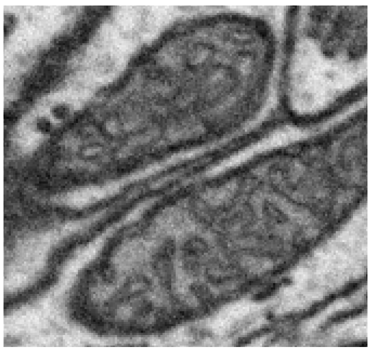
We already saw in section 2.4 and Figure 2.5 that the classification of superpixel faces get better the bigger the superpixels get. If not an oversegmentation is given but a segmentation that claims to represent the actual neurons, the amount of prior knowledge that can be applied is much bigger. Solely by the shape or the topology of segments humans can detect errors in the segmentation. To add such prior knowledge in the initial segmentation procedure is not trivial at all. The shape of an object is a nonlocal property. Therefore all naive approaches rely on higher-order factors rendering the relevant problem sizes unsolvable.

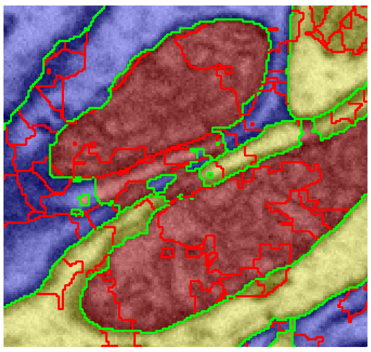
While this knowledge could not be incorporated in the segmentation step, one can usie it in a post-processing step. This means that any given segmentation proposal can be checked for implausible configuration and corrected if necessary. Below I will introduce two rules to detect critical segments in a segmentation. In principle it all boils down to the same general idea: Segments representing true neurons must be connected to the block boundaries since the extent of currently processable blocks is way smaller than the neurons extent. The detection of the critical places is only the first step though. For bigger parts of inter-cellular space that has been collected within one segment, an automated correction is challenging. Thus in this work, two special cases are chosen which enable an automated correction:

1. **Completely enclosed segments**, which are segments that have only one neighbor and do not touch the block-borders, can be directly merged with the one respective neighbor. The reasoning is that completely enclosed neurons make biologically no sense and do not contribute to the connectome.

2. **Mitochondria**, if ended up in a separate segment must be merged into the respective neuron. This relies on two things: First the mitochondrial nature of the segments must be determined (if it is not automatically done by the algorithm as shown in chapters 6 and 7 a separate classification of the segments is necessary). Secondly, the respective neuron must be determined. This is not always trivial . Oftentimes mitochondria touch cell membranes and in mean cases as shown in Figure 8.1 two mitochondria seem to to be in touch over cell membranes. This can lead to segments containing multiple mitochondria of different neurons. I propose the following procedure based on the approximately spherical shape of the metochondria:

   (a) After detecting a mitochondrion as visualized in Figure 8.1 (b) a shortest path between all pairs of superpixels within the mitochondrion segment is computed.


(a) two mitochondria of differnt neurons


(b) partition and semantic labeling as determined by amwc

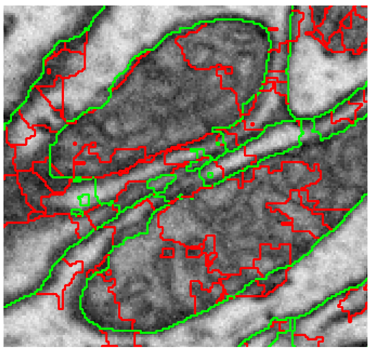
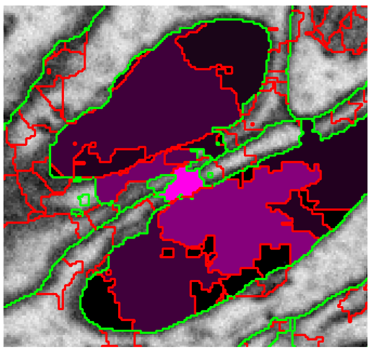
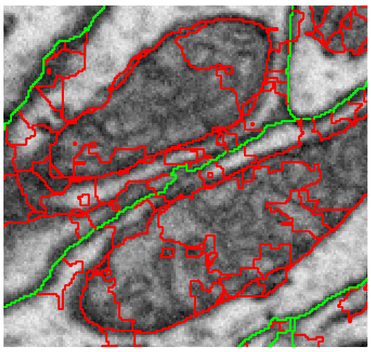

(c) pure partition extracted from (b)

Figure 8.1: Exemplary merging of mitochondria segment in the respective neurons. (a) shows the raw data (described in appendix A.1). (b) and (c) show aspects of a proposed segmentation that is supposed to be corrected. (d) indicates the measure used to detect the critical superpixel and (e) shows the correctly merged segments.


(d) pathcount color-coded in shades of pink


(e) automatically repaired segmentation

Each segment keeps track on how often a shortest path passes through. This measure is visualized in Figure 8.1 (d). Local maxima in this measure encode possible bottlenecks (superpixels connecting two Mitochondria).

(b) Now the local maxima are excluded from the segment and the remaining connected components are determined.

(c) The remaining connected components are all iteratively merged into the one neighboring neuron with which it has the biggest common surface.

(d) Finally the remaining (possibly) connecting segments are merged according to the same rule.

Both actions will be beneficial for the segmentation quality (comparing against a dense segmentation ground truth). Besides the experiments done in chapter 6 the comparison experiment described in section 9.2 can give an idea of the value of the presented procedures for segmentation quality.

Nevertheless they corrections do not change the connectivity implied by the segmentation and therefore do not change the resulting connectome. For algorithms respecting a semantic Mitochondria class like the AMWC from chapter 6 and the Semantic Agglomerative Clustering presented in 7 applying this postprocessing is necessary to judge the segmentation quality when comparing against e.g. the Multicut that does not aim at separating Mitochondria.

Chapter 9

General Discussion and Conclusion

This thesis introduces novel ideas to improve automated segmentation procedures for connectomics from electron microscopic image volumes. In the following section 9.1 I will condense the made contributions and summarize the algorithmic solutions. Since the prerequisites of most of these algorithms are identical we have the opportunity to compare them directly. This comparison is conducted in section 9.2. On this basis, a final conclusion and an outlook are offered in section 9.3.

9.1 Summary

Section 1.4 outline the three major contributions of this thesis: The V-Multicut, the possibility to introduce biological prior information via the Asymmetric Multiway Cut and the Semantic Agglomerative Clustering. In the following recap I elucidate these contributions as well some minor but important ideas. To get a quick overview on the basic segmentation paradigm that underlies the different algorithms I will distinguish pure partition algorithms and algorithms that are jointly solving a partitioning and a semantic labeling problem. Since membrane detection is the backbone of all approaches, also the classification of the algorithms by the utilized membrane representation is illuminating.

I will start with a recap of the partitioning Multicut problem as e.g. described by Kappes et. al. [74] and presented in chapter 3. Multicut-based algorithms do currently rank highest on public challenges for neuron segmentation (like the SNEMI3D challenge [75] for anisotropic data). The explicit flat representation of the neural membranes allows for an unambiguous classification of topological implausible membrane configurations. They are corrected via a cutting planes approach. The Multicut can be seen as the origin of several contributions of this thesis.

One contribution inspired by it is the V-Multicut. The basic question is the following: How can one impose Multicut-like constraints when working not with a flat, but an voluminous membrane representation? It turns out that the unambiguous detection of topologically implausible membrane configurations is not trivial. I show in chapter 5 how Multicut-like gap closing behavior can be imposed while simultaneously the benefits of the voluminous membrane representation are obtained. This is done by relating not to the detected membrane itself but to its skeleton, attained via a novel skeletonization approach.

Starting again from the Multicut problem, progress can also be achieved by staying within the flat representation but leaving the domain of pure partitioning. The recently proposed Asymmetric Multiway Cut is a generalization of the Multicut that is able to include sparse semantic information. I show in chapter 6 how this algorithm can utilize the spatial separation of axons and dendrites in mammals to significantly improve upon segmentations attainable via Multicut.

In addition to the improvements in segmentation quality, I provide a dynamical decomposition scheme of both the original Multicut and the V-Multicut that provides a significant speedup in all tested problem instances in sections 3.3 and 5.4.2.

So far, the proposed methods are providing globally optimal segmentations with respect to the provided weights. In section 4.2.1 is is shown that for problem instances where local membrane evidence is not reliable, globally optimal methods can be unfeasible. The presented alternative is Agglomerative Clustering, a well known greedy partitioning procedure working on the flat representation. In section 4.2.2 I introduce a novel stopping criterion that is not requiring probabilistic membrane detection. What seems like a minor modification turns out to be one of the keys to enable the introduction of sparse semantic information to Agglomerative Clustering. The resulting algorithm, that is described in chapter 7, is called Semantic Agglomerative Clustering. This algorithm is able to imitate the Asymmetric Multiway Cut. In section 7.2 an application is presented, for which globally optimal methods do not terminate on reasonable timescales. There, Semantic Agglomerative Clustering can demonstrate its strengths: Its runntime is indifferent to the quality of the used weights/the membrane detector (see Figure 4.2).

The eventual quality of all utilized segmentation algorithms does always depend on the initial oversegmentation (see sections 2.3 and 2.4.2) as well as on the quality of the used weights. In the following section 9.2 I will use the fact that all presented segmentation procedures, but the V-Multicut, rely on the flat representation, to perform a direct quantitative comparison.

9.2 Comparison Experiment

Since presented algorithms aim to achieve the same goal, the segmentation of neurons, it is of interest to directly compare both their segmentation quality and their runtime. To do so one needs to keep in mind that the eventual segmentation quality depends not only on the final segmentation procedure (Multicut, Asymmetric Multiway Cut, Semantic Agglomerative Clustering) but also on all previous steps like the supervoxel generation and the learned affinities between them. An overview on the necessary prerequisites of the presented algorithms was already given in Figure 1.5. For all but chapter 5, I relied on the flat representation of membranes (see section 2.1.5), where the membranes are supposed to coincide with the supervoxel boundaries. This means that the Multicut (MC), the Agglomerative Clustering (AC), the Asymmetric Multiway Cut (AMWC) and the Semantic Agglomerative Clustering (SAC) can be performed on identical supervoxels and identical face-weights. This makes a direct comparison feasible.

This section reports on the results of this comparison. They form the basis for the final discussion of the findings of this thesis. The V-Multicut is excluded from this direct comparison. Not only is the concept so far only realized for 2D, but also does the utilized voluminous representation dictate supervoxels of a different nature (see section 5.4.1).

In cases where such semantic information is available for the superpixels, even if it is only available sparsely, both the Asymmetric Multiway Cut from chapter 6 and the Semantic Agglomerative Clustering, introduced in chapter 7, are able to include it in the partitioning process. Also here the stopping criterion from section 4.2.2, determining the best possible solution based on probabilistic weights, is applied.

In addition to these algorithms I introduce several post-processing procedures. Depending on the availability of semantic information different post-processing steps are possible. For example, the merging of mitochondria, introduced in chapter 8, requires their detection in the first place. In case of the Asymmetric Multiway Cut and Semantic Agglomerative Clustering this post-processing enables a meaningful comparison to the Multicut and the vanilla Agglomerative Clustering.

Given all these algorithms it is now interesting to see them performing on the exact same superpixels, the same edge-weights and the same node-wise information (as far as they can utilize it). The face-wise membrane predictions are computed by a random forest as explained in section 2.4 and are then mapped to the respective edges in the region adjacency graph by the max operation as suggested by the experiments in section 2.4.3. The superpixel-wise sparse affiliation to the semantic classes (axon, dendrite and mitochondrion) are also learned by a random forest as presented in section 6.3. As reasoned in section 7.1, the Semantic Agglomerative Clustering does not handle probabilistic semantic affiliations. The propagation of already made class decisions (hard labels) can be done conclusively. For this experiment I chose to transform the individual class probabilities into hard labels in the following way: If the probability of some class c is higher than 0.75 and no other class has a probability higher than 0.5 than the hard decision for class c is done right away. This concrete class can now be propagated as explained in section 7.1.

Table 9.1 condenses the result of this comparison experiment. The quality measures, the Rand index and the variation of information (see section 2.6) are evaluated on a pixel-wise, dense ground truth that does not necessarily need to respect the superpixel boundaries. In addition the runtime for the respective inference and the post-processing is listed.

Generally, it can be concluded that adding the semantic information benefits the segmentation quality. The post-processed Asymmetric Multiway Cut (VI = 0.50139) improves significantly over the post-processed Multicut (VI = 0.54526). Even stronger improvements are observed when going from AC to SAC. Since there initial state is worse, there are more places that semantic information can help.

Note that in the AMWC as well as in the SAC variants, without post-processing, the mitochondria are in their separate partition element. Therefore, the respective algorithms have an disadvantage with respect to the ones without semantic information (MC and AC) when compared against neuron segmentation ground truth. The not post-processed Asymmetric Multiway Cut compared to the Multicut suffers more from the additional mitochondria segments than it benefits through the prevention of false merges due to semantic information. These two effects basically cancel out for the equivalent comparison of Agglomerative Clustering. This means the absolute value of the error measure attained by missegmenting all mitochondria is as big as the gain that is attainable via semantic priors.

In terms of stopping criteria for the Agglomerative Clustering one can say that the automatic stopping in $\mathrm{AC}^{\mathrm{auto}}$, $\mathrm{AC}^{\mathrm{auto}}_{\mathrm{post}}$, $\mathrm{SAC}^{\mathrm{auto}}$ and $\mathrm{SAC}^{\mathrm{auto}}_{\mathrm{post}}$ leads to slightly worse segmentation quality than the one attained by the injection of ground truth information in $\mathrm{AC}^{\mathrm{gt}}$, $\mathrm{AC}^{\mathrm{gt}}_{\mathrm{post}}$, $\mathrm{SAC}^{\mathrm{gt}}$ and $\mathrm{SAC}^{\mathrm{gt}}_{\mathrm{post}}$. This justifies the stopping criterion introduced in section 4.2.2 experimentally.

Time-wise it is apparent that the improvements of segmentation quality, gained by the globally optimal approaches MC and AMWC has a price. While one run of Agglomerative Clustering takes around one second, the Asymmetric Multiway Cut takes over three minutes. Note that here the global, not decomposed approaches are considered. The actual

		RI	VI	t	t_{post}	t^{CPU}	t^{CPU}_{post}
🟥	MC	0.98947	0.55027	38.1 ± 5.5	-	76.4 ± 1.0	-
🟥	MC_{post}	0.98969	0.54526	38.1 ± 5.5	6.0 ± 0.5	76.4 ± 1.0	6.0 ± 0.45
🟥	AC^{gt}	0.97438	1.02761	0.8 ± 0.1	-	0.8 ± 0.1	-
🟥	AC^{gt}_{post}	0.98115	0.96607	0.8 ± 0.1	3.6 ± 0.6	0.8 ± 0.1	3.67 ± 0.6
🟥	AC^{auto}	0.97398	1.07441	1.6 ± 0.1	-	1.6 ± 0.1	-
🟥	AC^{auto}_{post}	0.98074	1.01289	1.6 ± 0.1	3.8 ± 0.6	1.6 ± 0.1	3.8 ± 0.6
🟦	AMWC	0.98141	0.69487	186 ± 7	-	718 ± 11	-
🟦	$AMWC_{post}$	0.99401	0.50139	186 ± 7	114 ± 21	718 ± 11	121 ± 32
🟦	SAC^{gt}	0.97496	1.00294	0.8 ± 0.1	-	0.8 ± 0.1	-
🟦	SAC^{gt}_{post}	0.98324	0.85082	0.8 ± 0.1	977 ± 33	0.8 ± 0.1	983 ± 33
🟦	SAC^{auto}	0.97432	1.05884	1.4 ± 0.3	-	1.6 ± 0.1	-
🟦	SAC^{auto}_{post}	0.98497	0.86939	1.4 ± 0.3	369 ± 27	1.6 ± 0.1	378 ± 33

Table 9.1: This table shows a comparison of several algorithms utilized within this thesis. As quality measures the Rand index (RI, higher is better) and the variation of information (VI, lower is better) are used. In addition, the respective absolute inference time t, the respective absolute post-processing time t_{post} and the CPU[1]time t^{CPU} as well as the CPU time for the post-processing t^{CPU}_{post} are reported. The rows where post-processing is applied are shaded in gray. The very left column visually indicates the use of semantic information. While pink rows do rely on pairwise affiliations only, the blue ones utilize semantics in addition. The algorithms tested are the Multicut (MC), the Agglomerative Clustering stopping when the number of segments in the ground truth is reached (AC^{gt}) as well as the Agglomerative Clustering stopped naturally by the criterion introduced in section 4.2.2 (AC^{auto}). In addition the Asymmetric Multiway Cut (AMWC) and the Semantic Agglomerative Clustering (SAC) in both stopping variants are tested.

advantage of the Agglomerative Clustering based algorithms is the scalability as discussed in chapter 4. The automatic stopping of the Agglomerative Clustering is twice as slow as the one stopped using ground truth information. This reflects its actual implementation. One complete run of the Agglomerative Clustering is performed until only one segment is left to determine the best possible stopping point according to the introduced criterion. In a second round the algorithm is terminated at this best point.

As it can be seen in the runtime of the post-processing steps in Table 9.1, the implementation of the merging procedure of the mitochindria, relevant for $AMWC_{post}$, SAC^{gt}_{post} and SAC^{auto}_{post}, is slow. The routine of computing the shortest path for all pairs of superpixles within one connected component of mitochondrion gets out of hand for bigger mitochondria segments very fast. Therefore it can be seen as a proof of concept only. In any upscaled pipeline one could, for example, try to subsample the pairs within the connected mitocondria components for the determination of the connecting elements (see chapter 8). The following section will interpret the presented results, look into implications of this work and conclude this thesis.

[1]Intel Xeon E5-2650 v3

9.3 Conclusion and Outlook

The three major contribution of this work are: The introduction of Multicut-like constraints in a volouminous representation via the V-Multicut, the consideration of biological priors via the Asymmetric Multiway Cut and the consideration of biological priors via the Semantic Agglomerative Clustering (introduced respectively in chapters 5, 6 and 7).

While the key idea of the V-Multicut, to bring prior information about the topology of cell membranes to the level of voluminous membrane segmentations, is appealing theoretically, the algorithm (as formulated in section 5.2.1) is so far restricted to 2D. This prevents it currently from being widely applicable for neural segmentation. To elevate the V-Multicut also to 3D, two things have to be adapted. The simplicity criterion, determining the relevance of a given superpixel for the topology, must be modified. It is an open research question how this can be done in a localized way. This is necessary to allow the localization of respective constraints. Secondly, the criteria on which the skeleton elements are to be classified must be adapted. Eventually it would be most desirable to generalize the formulation to be valid irrespective of the dimension of the problem. An orthogonal direction to evolve the presented algorithm is the introduction of new constraints. Constraining not only open ends, but demanding a truly closed surface for the membranes, as in the Multicut case, could resolve more problems in the segmentation of the neural membranes.

The conducted comparison experiment in section 9.2 is suitable as a basis for concluding thoughts on both the Asymmetric Multiway Cut and the Semantic Agglomerative Clustering. As one can also observe in the comparison between the Multicut algorithm and the vanilla Agglomerative Clustering, the increased runtime that global optimality demands pays off in terms of quality. A similar trend is apparent in the comparison of the Asymmetric Multiway Cut and the Semantic Agglomerative Clustering in section 9.2. The conducted experiments prove that the inclusion of the sparse biological priors supports the (otherwise membrane driven) segmentation of neurons: The Asymmetric Multiway Cut exceeds the Multicut and the Semantic Agglomerative Clustering exceeds the Agglomerative Clustering quality-wise. Further biological inquiries may further increase the number of usable semantic classes. It might, for example, be possible to distinguish astrocytes, a type of support cells for the neurons, and introduce them as an additional semantic class in the presented frameworks to further improve the segmentation.

One next important milestone for the field of segmentation for connectomics is to show that automated methods can match human performance. Of the presented approaches, the Asymmetric Multiway Cut is the most promising candidate for this task. It allows for the inclusion of higher-level biological prior information via the sparse semantic labels. The utilized information is also used by human experts when performing manual tracing. I show that the spatial separation between axons and dendrites in mammals can be exploited in this way. In a direct comparison in chapter 6 and section 9.2 this inclusion leads to a significant improvement over the results produced by the Multicut algorithm, which can not take these clues into account. Besides the Assymmetric Multiway Cut, other ways to include semantic information are conceivable. For example the mentioned, recently proposed Lifted Multicut is theoretically able to model semantic repulsions via long-range potentials. Whether the pipeline first achieving human performance will be based on the AMWC or not is hard to predict. The achieved improvements within this work suggest that the presented semantic information will most certainly be utilized. There are image volumes for which the globally optimal methods, as described in chapters 3 and 6, embedded in the currently used pipelines are not suitable because of insufficient data quality. One example is shown in chapter 7. The resolution of the data makes a local membrane classification challenging. Globally optimal methods rely on a certain edge quality. Here the variants of the Agglomerative Clustering as presented in chapters 4 and 7 are applicable since their runtime is indifferent to the hardness of the problem at hand. I show how within this framework, available semantic information can be included (chapter 7). This approach reveals the algorithm's potential for a wider range of applications. Not only could skeletons be inflated to a full neuron segmentation but also interactive settings, where users place seeds for respective neurons at corrupted places are possible due to the advantages in runtime. If in the future reliable microtubules detectors exist, these detected structures could replace the handmade skeletons as seeds in a Semantic Agglomerative Clustering workflow to end up with a fully automated approach.

I hope that the demonstrated improvements will lead other researchers to consider more high-level biological information and finally serve as another step to fully automated connectome reconstruction.

References

[1] "The society of plant neurobiology." http://ds9.botanik.uni-bonn.de/zellbio/AG-Baluska-Volkmann/spn/index.php. Accessed: 2016-04-17.

[2] S. Garnier, J. Gautrais, and G. Theraulaz, "The biological principles of swarm intelligence," *Swarm Intelligence*, vol. 1, no. 1, pp. 3–31, 2007.

[3] D. Le Bihan, "Looking into the functional architecture of the brain with diffusion MRI," *Nature Reviews Neuroscience*, vol. 4, no. 6, 2003.

[4] B. L. Chen, D. H. Hall, and D. B. Chklovskii, "Wiring optimization can relate neuronal structure and function," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 12, pp. 4723–4728, 2006.

[5] T. Bührmann and E. Izquierdo, "Worm circuitry explorer." http://elegans.herokuapp.com/. Accessed: 2015-12-08.

[6] F. Jabr, "The connectome debate: Is mapping the mind of a worm worth it?." http://www.scientificamerican.com/article/c-elegans-connectome/, 2012. Accessed: 2016-04-24.

[7] J. White, E. Southgate, J. Thomson, and S. Brenner, "The structure of the nervous system of the nematode caenorhabditis elegans: the mind of a worm," *Philosophical Transactions of the Royal Society of London B*, vol. 314, pp. 1–340, 1986.

[8] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, D. B. Chklovskii, *et al.*, "Structural properties of the caenorhabditis elegans neuronal network," *PLoS Computational Biology*, vol. 7, no. 2, p. e1001066, 2011.

[9] A. P. Alivisatos, M. Chun, G. M. Church, R. J. Greenspan, M. L. Roukes, and R. Yuste, "The brain activity map project and the challenge of functional connectomics," *Neuron*, vol. 74, no. 6, pp. 970–974, 2012.

[10] C. H. Lee, S. J. Blackband, and P. Fernandez-Funez, "Visualization of synaptic domains in the drosophila brain by magnetic resonance microscopy at 10 micron isotropic resolution," *Scientific Reports*, vol. 5, 2015.

[11] S. Herculano-Houzel, B. Mota, and R. Lent, "Cellular scaling rules for rodent brains," *Neuron*, vol. 74, no. 6, pp. 970–974, 2012.

[12] A. Schüz and G. Palm, "Density of neurons and synapses in the cerebral cortex of the mouse," *Journal of Comparative Neurology*, vol. 286, no. 4, pp. 442–455, 1989.

[13] S. Herculano-Houzel, "The human brain in numbers: a linearly scaled-up primate brain," *Frontiers in Human Neuroscience*, vol. 3, 2009.

[14] B. Pakkenberg, D. Pelvig, L. Marner, M. J. Bundgaard, H. J. G. Gundersen, J. R. Nyengaard, and L. Regeur, "Aging and the human neocortex," *Experimental Gerontology*, vol. 38, no. 1, pp. 95–99, 2003.

[15] J. W. Lichtman, H. Pfister, and N. Shavit, "The big data challenges of connectomics," *Nature Neuroscience*, vol. 17, no. 11, pp. 1448–1454, 2014.

[16] G. Knott, H. Marchman, D. Wall, and B. Lich, "Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling," *The Journal of Neuroscience*, 2008.

[17] S. M. Plaza, L. K. Scheffer, and D. B. Chklovskii, "Toward large-scale connectome reconstructions," *Current Opinion in Neurobiology*, vol. 25, pp. 201–210, 2014.

[18] S. Seung, *Connectome: How the brain's wiring makes us who we are.* Houghton Mifflin Harcourt, 2012.

[19] A. Kreshuk, C. Straehle, C. Sommer, U. Koethe, *et al.*, "Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images.," *PLoS ONE*, 2011.

[20] C. Becker, K. Ali, G. Knott, and P. Fua, "Learning context cues for synapse segmentation," *IEEE Transactions on Medical Imaging*, 2013.

[21] P. Fua and G. W. Knott, "Modeling brain circuitry over a wide range of scales," *Frontiers in Neuroanatomy*, vol. 9, 2015.

[22] A. Kreshuk, J. Funke, A. Cardona, and F. A. Hamprecht, "Who is talking to whom: Synaptic partner detection in anisotropic volumes of insect brain," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pp. 661–668, Springer, 2015.

[23] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[24] D. Ausserré and M.-P. Valignat, "Wide-field optical imaging of surface nanostructures," *Nano Letters*, vol. 6, no. 7, pp. 1384–1388, 2006.

[25] M. J. Rust, M. Bates, and X. Zhuang, "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)," *Nature Methods*, vol. 3, no. 10, pp. 793–796, 2006.

[26] E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz, and H. F. Hess, "Imaging intracellular fluorescent proteins at nanometer resolution," *Science*, vol. 313, no. 5793, pp. 1642–1645, 2006.

[27] S. T. Hess, T. P. Girirajan, and M. D. Mason, "Ultra-high resolution imaging by fluorescence photoactivation localization microscopy," *Biophysical Journal*, vol. 91, no. 11, pp. 4258–4272, 2006.

[28] A. Dani, B. Huang, J. Bergan, C. Dulac, and X. Zhuang, “Superresolution imaging of chemical synapses in the brain,” *Neuron*, vol. 68, no. 5, pp. 843–856, 2010.

[29] "Wikimedia Commons", “Brainbow.” `https://upload.wikimedia.org/wikipedia/commons/8/8c/Brainbow_(Smith_2007).jpg`, 2008. Accessed: 2015-12-14.

[30] J. Livet, T. A. Weissman, H. Kang, R. W. Draft, J. Lu, R. A. Bennis, J. R. Sanes, and J. W. Lichtman, “Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system,” *Nature*, vol. 450, no. 7166, pp. 56–62, 2007.

[31] A. Schropp, R. Hoppe, J. Patommel, D. Samberg, F. Seiboth, S. Stephan, G. Wellenreuther, G. Falkenberg, and C. Schroer, “Hard x-ray scanning microscopy with coherent radiation: Beyond the resolution of conventional x-ray microscopes,” *Applied Physics Letters*, vol. 100, no. 25, p. 253112, 2012.

[32] H. Mizutani, Y. Takeda, A. Momose, A. Takeuchi, and T. Takagi, “X-ray microscopy for neural circuit reconstruction,” in *Journal of Physics: Conference Series*, vol. 186, p. 012092, IOP Publishing, 2009.

[33] E. Ramon-Moliner, “The golgi-cox technique,” in *Contemporary Research Methods in Neuroanatomy*, pp. 32–55, Springer, 1970.

[34] R. Erni, M. D. Rossell, C. Kisielowski, and U. Dahmen, “Atomic-resolution imaging with a sub-50-pm electron probe,” *Physical Review Letters*, vol. 102, no. 9, p. 096101, 2009.

[35] L. Silvestri, L. Sacconi, and F. S. Pavone, “The connectomics challenge,” *Functional Neurology*, vol. 28, no. 3, pp. 167–173, 2013.

[36] A. Wanner, M. Kirschmann, and C. Genoud, “Challenges of microtome-based serial block-face scanning electron microscopy in neuroscience,” *Journal of microscopy*, vol. 259, no. 2, pp. 137–142, 2015.

[37] R. W. Ware and V. LoPresti, “Three-dimensional reconstruction from serial sections,” *International Review of Cytology*, vol. 40, pp. 325–440, 1975.

[38] J. Goldstein, D. E. Newbury, P. Echlin, D. C. Joy, A. D. Romig Jr, C. E. Lyman, C. Fiori, and E. Lifshin, *Scanning electron microscopy and X-ray microanalysis: a text for biologists, materials scientists, and geologists.* Springer Science & Business Media, 2012.

[39] W. Denk and H. Horstmann, “Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure,” *PLOS Biology*, vol. 2, no. 11, p. e329, 2004.

[40] K. J. Hayworth, J. L. Morgan, R. Schalek, D. R. Berger, D. G. Hildebrand, and J. W. Lichtman, “Imaging ATUM ultrathin section libraries with wafermapper: a multi-scale approach to EM reconstruction of neural circuits,” *Frontiers in Neural Circuits*, 2014.

[41] B. Alberts, D. Bray, K. Hopkin, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential cell biology.* Garland Science, 2013.

[42] I. B. Levitan, L. K. Kaczmarek, *et al.*, *The neuron: cell and molecular biology.* Oxford University Press, 2015.

[43] M. Helmstaedter, K. L. Briggman, and W. Denk, "High-accuracy neurite reconstruction for high-throughput neuroanatomy," *Nature Neuroscience*, vol. 14, no. 8, pp. 1081–1088, 2011.

[44] J. Kornfeld, F. Svara, M.-T. Nguyen, N. Pfeiler, M. Pronkin, O. Shatz, S. Spaar, and S. Alex, "Knossos." `https://github.com/knossos-project/knossos`, 2016. Accessed: 2016-04-17.

[45] Fly EM. project team, "Raveler." `https://openwiki.janelia.org/wiki/display/flyem/Raveler`, 2016. Accessed: 2016-04-24.

[46] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. W. Lichtman, and H. Pfister, "Design and evaluation of interactive proofreading tools for connectomics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2466–2475, 2014.

[47] C. Straehle, U. Koethe, G. Knott, and F. Hamprecht, "Carving: scalable interactive segmentation of neural volume electron microscopy images.," *Proceedings of MICCAI*, 2011.

[48] C. Sommer, C. Straehle, U. Koethe, and F. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," *IEEE International Symposium on Biomedical Imaging*, 2011.

[49] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák, "Catmaid: collaborative annotation toolkit for massive amounts of image data," *Bioinformatics*, vol. 25, no. 15, pp. 1984–1986, 2009.

[50] D. B. Chklovskii, S. Vitaladevuni, and L. K. Scheffer, "Semi-automated reconstruction of neural circuits using electron microscopy," *Current Opinion in Neurobiology*, vol. 20, no. 5, pp. 667–675, 2010.

[51] E. Jurrus, S. Watanabe, R. J. Giuly, A. R. Paiva, M. H. Ellisman, E. M. Jorgensen, and T. Tasdizen, "Semi-automated neuron boundary detection and nonbranching process segmentation in electron microscopy images," *Neuroinformatics*, vol. 11, no. 1, pp. 5–29, 2013.

[52] C. Jones, T. Liu, N. W. Cohan, M. Ellisman, and T. Tasdizen, "Efficient semi-automatic 3D segmentation for neuron tracing in electron microscopy images," *Journal of Neuroscience Methods*, vol. 246, pp. 13–21, 2015.

[53] W.-K. Jeong, J. Beyer, M. Hadwiger, A. Vazquez, H. Pfister, and R. T. Whitaker, "Scalable and interactive segmentation and visualization of neural processes in EM datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1505–1514, 2009.

[54] D. R. Myatt, T. Hadlington, G. A. Ascoli, and S. J. Nasuto, "Neuromantic: from semi-manual to semi-automatic reconstruction of neuron morphology," *Frontiers in Neuroinformatics*, vol. 6, no. 4, pp. 1–14, 2012.

[55] V. Jain, J. F. Murray, F. Roth, S. Turaga, V. Zhigulin, K. L. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung, “Supervised learning of image restoration with convolutional networks,” in *IEEE 11th International Conference on Computer Vision*, pp. 1–8.

[56] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in *Advances in Neural Information Processing Systems*, pp. 2843–2851, 2012.

[57] G. Huang and V. Jain, “Deep and wide multiscale recursive networks for robust image labeling,” *Proceedings of International Conference on Learning Representations*, 2014.

[58] E. Jurrus, A. R. Paiva, S. Watanabe, J. R. Anderson, B. W. Jones, R. T. Whitaker, E. M. Jorgensen, R. E. Marc, and T. Tasdizen, “Detection of neuron membranes in electron microscopy images using a serial neural network architecture,” *Medical Image Analysis*, vol. 14, no. 6, pp. 770–783, 2010.

[59] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen, “Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks,” *Proceedings of International Conference on Computer Vision*, 2013.

[60] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pp. 234–241, Springer, 2015.

[61] H.-F. Yang and Y. Choe, “Cell tracking and segmentation in electron microscopy images using graph cuts,” in *IEEE International Symposium on Biomedical Imaging*, pp. 306–309, IEEE, 2009.

[62] S. N. Vitaladevuni and R. Basri, “Co-clustering of image segments using convex optimization applied to em neuronal reconstruction,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2203–2210, IEEE, 2010.

[63] A. Vazquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister, “Segmentation fusion for connectomics,” *Proceedings of International Conference on Computer Vision*, 2011.

[64] J. Funke, J. Martel, S. Gerhard, B. Andres, *et al.*, “Candidate sampling for neuron reconstruction from anisotropic electron microscopy volumes,” in *Proceedings of MICCAI*, 2014.

[65] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister, “Large-scale automatic reconstruction of neuronal processes from electron microscopy images,” *Medical Image Analysis*, vol. 22, no. 1, pp. 77–88, 2015.

[66] D. Laptev, A. Vezhnevets, S. Dwivedi, and J. M. Buhmann, “Anisotropic ssTEM image segmentation using dense correspondence across sections,” in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012*, pp. 323–330, Springer, 2012.

[67] B. Andres, U. Köthe, M. Helmstaedter, W. Denk, and F. A. Hamprecht, “Segmentation of SBFSEM volume data of neural tissue by hierarchical classification,” in *Pattern Recognition*, pp. 142–152, Springer, 2008.

[68] V. Jain, S. Turaga, K. Briggman, M. Helmstaedter, *et al.*, “Learning to agglomerate superpixel hierarchies,” in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2011.

[69] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. Chklovskii, “Machine learning of hierarchical clustering to segment 2D and 3D images,” *PLoS ONE*, 2013.

[70] T. Parag, A. Chakraborty, S. Plaza, and L. Scheffer, “A context-aware delayed agglomeration framework for electron microscopy segmentation,” *PLoS ONE*, vol. 10, no. 5, 2015.

[71] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen, “A modular hierarchical approach to 3D electron microscopy image segmentation,” *Journal of Neuroscience Methods*, vol. 226, no. 0, pp. 88 – 102, 2014.

[72] M. G. Uzunbas, C. Chen, and M. Dimitris, “An efficient conditional random field approach for automatic and interactive neuron segmentation,” *Medical Image Analysis*, 2015.

[73] B. Andres, T. Kroeger, K. Briggman, W. Denk, *et al.*, “Globally optimal closed-surface segmentation for connectomics,” *Proceedings of ECCV*, 2012.

[74] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schn, “Globally optimal image partitioning by multicuts,” in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 31–44, Springer, 2011.

[75] I. Arganda-Carreras, S. Seung, A. Vishwanathan, and D. Berger, “SNEMI3D: 3D Segmentation of neurites in EM images .” `http://brainiac2.mit.edu/SNEMI3D/leaders-board`, 2016. Accessed 2016-02-20.

[76] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[77] B. Jähne, *Digitale bildverarbeitung: und bildgewinnung.* Springer-Verlag, 2012.

[78] T. Lindeberg, *Scale-space theory in computer vision*, vol. 256. Springer Science & Business Media, 2013.

[79] J. J. Koenderink and A. J. van Doorn, “Representation of local geometry in the visual system,” *Biological Cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[80] U. Köthe *et al.*, “Vigra-vision with generic algorithms,” *Cognitive Systems Group, University of Hamburg, Germany*, 2008.

[81] A. Liaw and M. Wiener, “Classification and regression by randomforest,” *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[82] J. F. Peters, “Topology of digital images,” *Visual Pattern Discovery in Proximity Spaces*, vol. 63, 2014.

[83] T. Y. Kong and A. Rosenfeld, “Digital topology: Introduction and survey,” *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 3, pp. 357–393, 1989.

[84] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz, “Graph-based active learning of agglomeration (GALA): a python library to segment 2D and 3D neuroimages,” *Frontiers in Neuroinformatics*, vol. 8, 2014.

[85] A. Trémeau and P. Colantoni, “Regions adjacency graph applied to color image segmentation,” *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 735–744, 2000.

[86] J. B. Listing, *Der census räumlicher complexe: oder verallgemeinerung des euler’schen satzes von den polyedern*, vol. 10. Dieterich, 1862.

[87] R. Klette and A. Rosenfeld, *Digital geometry: Geometric methods for digital picture analysis.* Elsevier, 2004.

[88] U. Köthe, B. Andres, T. Kröger, and F. Hamprecht, “Geometric analysis of 3D electron microscopy data,” in *Applications of Discrete Geometry and Mathematical Morphology*, pp. 93–108, Springer, 2012.

[89] V. Kovalevsky, “Algorithms in digital geometry based on cellular topology,” in *Combinatorial Image Analysis*, pp. 366–393, Springer, 2004.

[90] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, “Slic superpixels compared to state-of-the-art superpixel methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[91] D. Barber, *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[92] J. H. Kappes, M. Speth, G. Reinelt, and C. Schnörr, “Higher-order segmentation via multicuts,” *arXiv preprint arXiv:1305.6387*, 2013.

[93] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo, “Higher-order correlation clustering for image segmentation,” in *Advances in Neural Information Processing Systems*, pp. 1530–1538, 2011.

[94] S. Bagon and M. Galun, “Large scale correlation clustering optimization,” *arXiv preprint arXiv:1112.2903*, 2011.

[95] N. Bansal, A. Blum, and S. Chawla, “Correlation clustering,” *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, 2004.

[96] D. Greig, B. Porteous, and A. H. Seheult, “Exact maximum a posteriori estimation for binary images,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 271–279, 1989.

[97] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[98] B. Andres, J. H. Kappes, T. Beier, U. Köthe, F. Hamprecht, *et al.*, "Probabilistic image segmentation with closedness constraints," in *IEEE International Conference on Computer Vision, pages=2611–2618, year=2011, organization=IEEE.*

[99] S. Nowozin and S. Jegelka, "Solution stability in linear programming relaxations: Graph partitioning and unsupervised learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 769–776, ACM, 2009.

[100] R. Haas and M. Hoffmann, "Chordless paths through three vertices," *Theoretical Computer Science*, vol. 351, no. 3, pp. 360–371, 2006.

[101] S. Chopra and M. R. Rao, "The partition problem," *Mathematical Programming*, vol. 59, no. 1-3, pp. 87–115, 1993.

[102] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. , U. Koethe, and F. A. Hamprecht, "Globally optimal closed-surface segmentation for connectomics," in *Computer Vision–ECCV 2012*, pp. 778–791, Springer, 2012.

[103] "IBM ILOG CPLEX Optimization Studio." `http://www-03.ibm.com/software/products/de/ibmilogcpleoptistud`. Accessed: 2016-03-07.

[104] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres, "Efficient decomposition of image and mesh graphs by lifted multicuts," *arXiv preprint arXiv:1505.06973*, 2015.

[105] B. Andres, T. Beier, and J. H. Kappes, "OpenGM: A C++ Library for Discrete Graphical Models," *ArXiv e-prints*, 2012. Projectpage: `http://hciweb2.iwr.uni-heidelberg.de/opengm/` . Accessed: 2016-04-24.

[106] T. Beier, T. Kroeger, J. H. Kappes, U. Kothe, and F. A. Hamprecht, "Cut, glue, & cut: A fast, approximate solver for multicut partitioning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 73–80, IEEE, 2014.

[107] T. Beier, F. A. Hamprecht, and J. H. Kappes, "Fusion moves for correlation clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3507–3516, 2015.

[108] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich, "Diverse m-best solutions in markov random fields," in *Computer Vision–ECCV 2012*, pp. 1–16, Springer, 2012.

[109] T. Kröger, *Learning-based segmentation for connectomics.* PhD thesis, Heidelberg University, 2014.

[110] V. Jain, S. C. Turaga, K. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung, "Learning to agglomerate superpixel hierarchies," in *Advances in Neural Information Processing Systems*, pp. 648–656, 2011.

[111] R. Bellman, "A markovian decision process," tech. rep.

[112] R. Sibson, "Slink: an optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, pp. 30–34, 1973.

[113] F. Murtagh and P. Legendre, “Ward’s hierarchical clustering method: clustering criterion and agglomerative algorithm,” *arXiv preprint arXiv:1111.6285*, 2011.

[114] Y. Zeng, D. Samaras, W. Chen, and Q. Peng, “Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in n–d images,” *Computer Vision and Image Understanding*, vol. 112, no. 1, pp. 81–90, 2008.

[115] S. Nowozin and C. H. Lampert, “Global connectivity potentials for random field models,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 818–825, IEEE, 2009.

[116] C. Chen, D. Freedman, and C. H. Lampert, “Enforcing topological constraints in random field image segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2089–2096, IEEE, 2011.

[117] J. Stühmer, P. Schroder, and D. Cremers, “Tree shape priors with connectivity constraints using convex relaxation on general graphs,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2336–2343, 2013.

[118] N. Krasowski, T. Beier, G. Knott, U. Koethe, F. Hamprecht, and A. Kreshuk, “Neuron segmentation with high-level biological priors,” *IEEE International Symposium on Biomedical Imaging*, 2015.

[119] N. Krasowski, T. Beier, G. Knott, U. Koethe, F. Hamprecht, and A. Kreshuk, “Improving 3D EM data segmentation by joint optimization over boundary evidence and biological priors,” *submitted to: IEEE Transactions on Medical Imaging*, 2016.

[120] T. Kroeger, J. Kappes, T. Beier, U. Koethe, and F. Hamprecht, “Joint supervised-unsupervised segmentation,” *Proceedings of GCPR*, 2014.

[121] Y. Boykov, O. Veksler, and R. Zabih, “Markov random fields with efficient approximations,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 648–655, IEEE, 1998.

[122] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen, “icoseg: Interactive co-segmentation with intelligent scribble guidance,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3169–3176, IEEE, 2010.

[123] B. Andres, J. Yarkony, B. Manjunath, S. Kirchhoff, E. Turetken, C. C. Fowlkes, and H. Pfister, “Segmenting planar superpixel adjacency graphs wrt non-planar superpixel affinity graphs,” in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 266–279, Springer, 2013.

[124] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[125] "Wikimedia Commons", “Multipolar neuron.” `https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png`, 2013.

[126] M. E. Berck, A. Khandelwal, L. Claus, L. H. Nuñez, G. Si, C. J. Tabone, F. Li, J. W. Truman, R. D. Fetter, M. Louis, A. D. T. Samuel, and A. Cardona, “The wiring diagram of a glomerular olfactory system,” *bioRxiv*, p. 037721, 2016.

Appendix A

Data Specifications

Within this work different images and image volumes are used to demonstrate and test the presented algorithms in practice. In this Appendix these different data sets are described in more detail.

A.1 FIBSEM - Mouse

This data shows parts of the somatosensory cortex of an adult mouse (female, C57BL/6; 10 weeks of age). The details on how it was taken via serial section scanning electron microscopy is described by Graham Knott et al. in [16] from the École polytechnique fédérale de Lausanne, Switzerland. The resolution is about 4 nm in the imaging plane and 5 nm in the slicing direction. It can be considered isotropic for the purposes of this thesis. There are three volumes in particular that were used within this thesis:

A.1.1 Volume 1

A volume holding $898 \times 899 \times 198$ pixels used exclusively for training of both the respective pixel-wise probabilities (membrane and semantic) as well as the face-wise merge probabilities.

A.1.2 Volume 2

A $700 \times 700 \times 700$ pixel volume used for testing e.g. in chapter 6.

A.1.3 Volume 3

This testing block holds $400 \times 401 \times 402$ pixels. It is a subset ob block 2.

All three blocks are carefully chosen in order to avoid myelin sheaths which, if not treated explicitly, can mess up the pixel-wise membrane probabilities and the superpixels. What distinguishes these three datasets is the fact that dense segmentation ground truth produced by human experts via ilastik/carving [48, 47] and mojo [46] is available. This does not only allow for a tangible training set but also for a reliable way to measure the performance of the presented algorithms.

A.2 ssTEM - Fruit Fly

This data set, taken by Berck et al. [126] in the Lab of Albert Cardona from the Janelia Research Campus in Ashburn, Virginia, shows parts of an antennal lobe of a common fruit fly. The specific section that is used in the chapter presenting the Semantic Agglomerative Clustering in 7 encompasses $500 \times 500 \times 6$ pixels.

For the subset that is relevant for this thesis a skeleton is available for every neuron.

A.3 Satellite Image

The used satellite image of Brussels used in chapter 5 was taken from Google Maps (Map data ©2014 Google Imagery ©2014 , Aerodata International Surveys, Cnes/Spot Image, DigitalGlobe, Landsat) on the 12.8.2014.

A.4 Plant Cell Walls

The image used in chapter 5 was taken by Maryline Lièvre from the Labotatoire d'Ecophysiologie des Plantes Sous Stress Environnementaux Équipe SPIC : Stress Environnementaux et Processus Intégrés and Léo Guignard from the INRIA project-team Virtual Plants, both located at the University in Montpelier, France.