Dissertation

submitted

to the

Combined Faculty 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

Diplom-Physiker Thorben Kröger

Born in Hannover

Oral examination: ___________________

Learning-based Segmentation for Connectomics

Advisor: Prof. Dr. Fred A. Hamprecht

Abstract

Recent advances in electron microscopy techniques make it possible to acquire high-resolution, isotropic volume images of neural circuitry. In *connectomics*, neuroscientists seek to obtain the circuit diagram involving *all* neurons and synapses in such a volume image. Mapping neuron connectivity requires tracing each and every neural process through terabytes of image data. Due to the size and complexity of these volume images, fully automated analysis methods are desperately needed. In this thesis, I consider automated, machine learning-based neurite segmentation approaches based on a *simultaneous* merge decision of adjacent supervoxels.

- Given a learned likelihood of merging adjacent supervoxels, Chapter 4 adapts a probabilistic graphical model which ensures that merge decisions are consistent and the surfaces of final segments are closed. This model can be posed as a multicut optimization problem and is solved with the cutting-plane method. In order to scale to large datasets, a fast search for (and good choice of) violated cycle constraints is crucial. Quantitative experiments show that the proposed closed-surface regularization significantly improves segmentation performance.

- In Chapter 5, I investigate whether the edge weights of the previous model can be chosen to minimize the loss with respect to *non-local* segmentation quality measures (e.g. Rand Index). Suitable $\boldsymbol{w}$ are obtained from a structured learning approach. In the Structured Support Vector Machine formulation, a novel fast enumeration scheme is used to find the most violated constraint. Quantitative experiments show that structured learning can improve upon unstructured methods. Furthermore, I introduce a new approximate, hierarchical and blockwise optimization approach for large-scale multicut segmentation. Using this method, high-quality approximate solutions for large problem instances are found quickly.

- Chapter 6 introduces another novel approximate scheme for multicut segmentation – *Cut, Glue & Cut* – which is based on the move-making paradigm. First, the graph is recursively partitioned into small regions (cut phase). Then, for any two adjacent regions, alternative cuts of these two regions define possible moves (glue & cut phase). The proposed algorithm finds segmentations that are – as measured by a loss function – as close to the ground-truth as the global optimum found by exact solvers, while being significantly faster than existing methods.

- In order to jointly label resulting segments as well as to label the boundaries between segments, Chapter 7 proposes the *Asymmetric Multi-way Cut* model, a variant of Multi-way Cut. In this new model, within-class cuts are allowed for some labels, while being forbidden for other labels. Qualitative experiments show when such a formulation can be beneficial. In particular, an application to joint neurite and cell organelle labeling in EM volume images is discussed.

- Custom software tools that can cope with the large data volumes common in the field of connectomics are a prerequisite for the implementation and evaluation of novel segmentation techniques. Chapter 3 presents version 1.0 of ILASTIK, a joint effort of multiple researchers. I have co-written its volume viewing component, VOLUMINA. ilastik provides an interactive pixel classification workflow on larger-than-RAM datasets as well as a semi-automated segmentation module useful for acquiring gold standard segmentations. Furthermore, I describe new software for dealing with hierarchies of cell complexes as well as for blockwise image processing operations on large datasets.

The different segmentation methods presented in this thesis provide a promising direction towards reaching the required reliability as well as the required data throughput necessary for connectomics applications.

Zusammenfassung

Neue Methoden der Elektronenmikroskopie ermöglichen es, hochauflösende Volumenbilder von neuronalem Gewebe aufzunehmen, die isotrope Auflösung haben. Im Forschungsfeld *Connectomics* versuchen Neurowissenschaftler, den Schaltplan *aller* Neuronen und Synapsen in einem solchen Volumenbild zu rekonstruieren. Um die neuronale Konnektivität kartieren zu können, müssen Neuronen durch Bilddaten in Terabyte-Größe verfolgt werden. Aufgrund der Größe und Komplexität der Daten werden hierfür dringend automatische Methoden benötigt. In dieser Arbeit beschäftige ich mich mit automatischen Segmentierungsverfahren, die auf maschinellem Lernen basieren. Dabei verfolge ich Ansätze, die auf dem *gleichzeitigen* Verschmelzen von benachbarten Supervoxeln aufbauen.

- Basierend auf gelernten Wahrscheinlichkeiten des Verschmelzens benachbarter Supervoxel wird in Kapitel 4 ein probabilistisches graphisches Modell adaptiert, welches konsistente Verschmelzungsentscheidungen sicherstellt und damit erzwingt, dass die Oberflächen der resultierenden Segmente geschlossen sind. Das Modell kann als Multicut-Optimierungsproblem formuliert werden und anschließend mit der Cutting-Plane-Methode gelöst werden. Für die erfolgreiche Optimierung großer Datensätze ist eine schnelle Suche (und eine gute Auswahl) von verletzten Kreis-Ungleichungen entscheidend. Quantitative Experimente zeigen, dass die vorgeschlagene Regularisierung die Qualität der Segmentierungsergebnisse wesentlich verbessert.

- In Kapitel 5 untersuche ich, ob die Kantengewichte $\boldsymbol{w}$ im vorher beschriebenen Modell derart gewählt werden können, dass der Loss bezüglich *nicht-lokaler* Maße für Segmentierungsqualität (z.B. des Rand Indexes) minimiert wird. Derartige $\boldsymbol{w}$ können mit Hilfe von Methoden des strukturierten Lernens gefunden werden. In einer Structured-Support-Vector-Machine-Formulierung wird eine neue Methode zum systematischen Aufzählen von Segmentierungen benutzt, um die am meisten verletzte Zwangsbedingung zu finden. Quantitative Experimente zeigen, dass mit strukturiertem Lernen Verbesserungen gegenüber unstrukturierten Verfahren erzielt werden können. Zudem stelle ich eine neue approximative Optimierungsstrategie für große Multicut-Probleme vor, die blockweise und hierarchisch arbeitet. Mit dieser Methode können Näherungslösungen hoher Qualität schnell gefunden werden.

- In Kapitel 6 wird ein weiteres approximatives Verfahren zur Optimierung von Multicut-Problemen – *Cut, Glue & Cut* – vorgestellt, das auf der Idee des Move-Making aufbaut. Hierbei wird zunächst der Graph rekursiv in kleine Regionen zerlegt (Cut-Phase). Dann wird für jedes Paar benachbarter Regionen geprüft, ob es bessere Cuts zwischen diesen Regionen gibt (Glue & Cut-Phase). Der vorgeschla-

gene Algorithmus findet Lösungen, die – gemessen an Loss-Funktionen – ebenso gut sind wie die global optimale Segmentierung. Diese empirisch guten Lösungen findet der neue Algorithmus wesentlich schneller als existierende approximative Methoden.

- In Kapitel 7 diskutiere ich eine neue Variante des Multi-way Cut, den *Asymmetric Multi-way Cut*. Ziel ist es, *gleichzeitig* sowohl die resultierenden Segmente als auch die Grenzen zwischen Segmenten zu labeln. In dem neuen Modell werden Cuts innerhalb bestimmter Klassen erlaubt, während sie für andere Klassen verboten sind. Erste qualitative Experimente zeigen, wann eine solche Formulierung vorteilhaft sein kann. Insbesondere diskutiere ich die Anwendung für das gleichzeitige Labeln von Neuronen und Zellorganellen in EM-Volumenbildern.

- Spezielle Softwarelösungen für den Umgang mit den großen Connectomics-Datenmengen sind Voraussetzung für die Implementierung und Evaluierung neuer Segmentierungsverfahren. In Kapitel 3 stelle ich die Version 1.0 des ILASTIK Softwarepakets – einem Gemeinschaftsprojekt – vor. Ich habe insbesondere die Anzeigekomponente für Volumenbilder, VOLUMINA, mitentwickelt. ilastik bietet eine interaktive Klassifikation von Pixeln in Datensätzen, die die Größe des Arbeitsspeichers weit überschreiten können. Außerdem bietet ilastik eine halbautomatische Methode zur Segmentierung, die für das Erstellen von dichten Referenzsegmentierungen benutzt wurde. Darüber hinaus beschreibe ich neue Software für den Umgang mit hierarchischen Zellkomplexen. Weitere Software stellt außerdem Bildverarbeitungsoperationen auf großen Datenmengen bereit.

Die verschiedenen Segmentierungsmethoden, die in dieser Arbeit diskutiert werden, stellen einen vielversprechenden Schritt in Richtung von Zuverlässigkeit und Datendurchsatz dar, wie sie für Connectomics Anwendungen gebraucht werden.

Acknowledgments

First, I want to thank Prof. Fred Hamprecht for supervising the research that has led to this thesis. Many years ago already, Prof. Hamprecht introduced me to the field of pattern recognition and its exciting application to the analysis of images from the life sciences. His research group – Multidimensional Image Processing – has always been a great combination of friendly, helpful, knowledgeable and fun people, stimulating scientific discussions, fruitful collaborations, and challenging problems on data from the forefront of current research.

Ullrich Köthe was always happy to share his expertise, discuss ideas and talk about programming. I am grateful to Björn Andres for introducing me to the problem of automated neurite segmentation and our excellent collaboration. For discussion and advice I want to thank Dr. Bogdan Savchynskyy, as well as my collaborator Jörg Kappes. Apart from being an enthusiastic C++ programmer, Thorsten Beier has always been around to discuss ideas about segmentation. I am happy that I could convince him about the usefulness of Python scripting and want to thank him for the enjoyable collaboration.

With my colleagues I have spent countless hours on pair programming, bug hunting and discussion while working on the ILASTIK project. With Bernhard Kausler I had great fun designing the VOLUMINA library. He has pushed me to write cleaner code. Together with Christoph Sommer, Christoph Straehle, Anna Kreshuk, Luca Fiaschi, Martin Schiegg and Stuart Berg we have worked towards a version 1.0 of ilastik.

I also wish to thank all my other colleagues for the excellent group atmosphere and discussions, in particular Xinghua Lou, Ferran Diego, Melih Kandemir, Burcin Erocal, Kemal Eren, Frank Herrmannsdörfer, Philipp Hanslovsky, Robert Walecki, Buote Xu, Ben Heuer, Chong Zhang, Niko Krasowski, Carsten Haubold as well as Barbara Werner.

Acquiring gold standard segmentations of EM data volumes can be a frustrating task, but was nevertheless carried out with perseverance by Oliver Petra and Kai Karius.

Furthermore, I want to thank Boray Tek for the good collaboration during his research stay in Heidelberg.

The amazing electron microscopy data of neural tissue I was able to work with has been key for my desire to work on automated image segmentation methods. I am very grateful to Prof. Winfried Denk and Kevin Briggman for providing the isotropic E1088 dataset. Shawn Mikula has answered many of my questions on neuroscience and electron microscopy and provided me with multiple great datasets from his cutting-edge research. I also want to thank him for the great collaboration on the soma detection problem. During her visit, Natalya Korogod showed me how difficult proper annotation of electron microscopy data really is, but still persisted in using the then crash-prone ILASTIK tools for annotation. I am indebted to Graham Knott for providing amazing FIBSEM volume images and inviting me to visit his laboratory in Lausanne as well as to everyone in the FlyEM team in Janelia Farm for showing me around during my visits and many interesting discussions.

Finally I would like to thank my family and all my friends for supporting me in my decision to pursue a PhD. Most of all, I am grateful to Christine for her understanding and love.

To Christine.

Contents

Chapter 1

The Case for Connectomics

In order to understand how the brain is able to perceive, think and remember, neuroscientists are acquiring huge volume images of neural circuitry. Using this data, they want to map the connectivity of the brain at a cellular level. The analysis of connectivity information, termed *connectomics*, promises to answer many questions about the functioning principles of the brain [135]. This chapter summarizes why neuroscientists believe that connectomics can answer those questions. Furthermore, it gives an overview of the technology used for volumetric image acquisition of neural tissue — serial sectioning electron microscopy. Finally we argue why fully automated analysis is necessary.

1.1 The Neuron Doctrine

Psychology describes mental functions and behavior. Here, the entire brain is treated as a black box: inputs are varied and outputs analyzed. However, little can be learned about the fundamental functioning principles of the brain.

Neuroscientists now want to look inside this black box. After having discovered and largely understood the *elementary functional units* of brain tissue, current research aims to discover how these rather simple elements are combined to give rise to complex behavior.

On the macroscale, neuroanatomists first used histological criteria to create maps of different brain areas, such as the one by Korbinian Broadman in 1909, in which the cerebral cortex is divided into 52 regions. The brains of persons with mental illnesses sometimes revealed damage to certain brain regions. This allowed to assign function to them, for example the "source of speech" to Broca's region. Today, *magnetic resonance imaging* (MRI) can be used for noninvasive measurement of structural connectivity (diffusion-weighted MRI) and functional connectivity (functional MRI) of the living brain with cubic millimeter resolution [43]. *Diffusion-weighted MRI* (dMRI) measures the diffusion direction of water in each volume element (voxel), which is strongly corre-

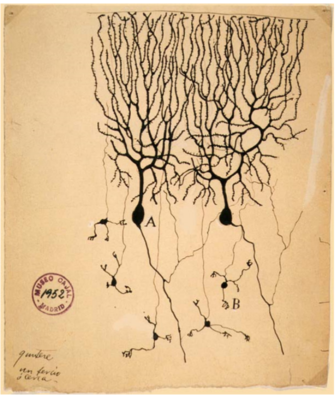


Figure 1.1: Ramón y Cajal's hand-drawn illustration [156] from 1899 of (A) Purkinje cells and (B) Granule cells from pigeon cerebellum. The Golgi staining made only a small fraction of the densely packed nerve cells visible, such that Cajal could use a low resolution light microscope to study the form of the stained cells.

lated with the directionality of axon bundles through white matter. However, the low resolution effectively averages the directionality of hundreds of thousands of axons (the main signaling units) in each voxel; for single axon resolution different imaging techniques are needed. *Functional MRI* (fMRI) is another widely used method for indirect measurements of neural activation over time, from which functional correlations between brain regions can be inferred.

On the microscale, a more thorough understanding of functionality critically relies on the ability to image at nanometer resolution. The work of Camillo Golgi and Santiago Ramón y Cajal (shared Nobel Prize in Medicine in 1906) is therefore considered as the foundation of modern neuroscience [46]. In 1873, Golgi had developed a *sparse* staining method for brain tissue, which was further perfected and extensively used by Cajal. Using a light microscope with apochromatic lenses, the stain revealed an intricately connected network of thin and branching processes. Figure 1.1 shows a drawing by Cajal made in 1899 of nerve cells in pigeon cerebellum. The Golgi method stains only a small fraction of cells, but if a cell *is* stained, it is stained in its entirety. Though most structures in neural tissue are smaller than visible wavelengths, the structure of the few stained cells can still be captured by light microscopy if they are contrasted against the unstained background.

Based on his observations, Cajal put forward the *neuron doctrine*, which states that the nervous system is made up of individual cells ("neurons"), as opposed to a diffuse network as suggested by Golgi and others. The neurons would transmit nerve impulses amongst each other at the points of contact. Otto Loewi and Henry Dale (Nobel Prize 1936) found evidence that information can be sent between neurons by chemical means: neurotransmitter molecules are released in one cell and sensed by another [135]. Contact sites between neurons where information can be transmitted are called *synapses*.

A *neuron* is a cell consisting of the cell body called *soma*, from which two types of processes, dendrites and axons (collectively called *neurites*), emanate. The soma typically gives rise to multiple *dendrites*, which branch in a tree-like fashion in the vicinity of the

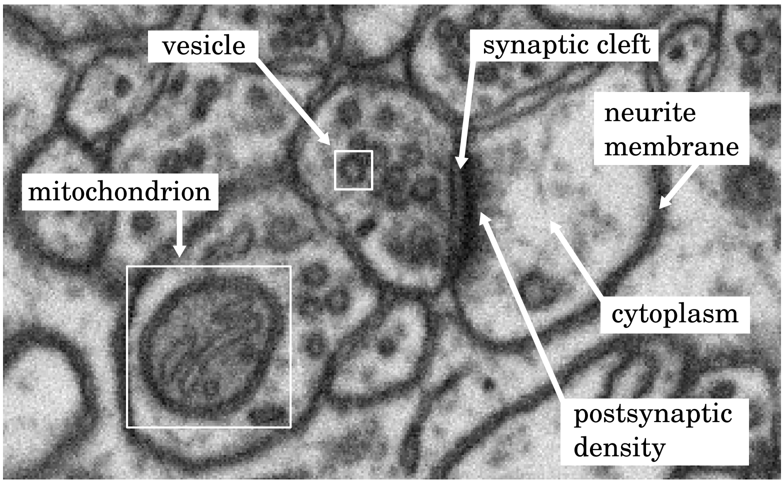


Figure 1.2: Appearance of cell organelles and neurite membranes in mouse brain tissue (conventional heavy metal staining, FIBSEM imaging at 5 nm resolution). Data courtesy of Graham Knott. See [58] for more details on synapse ultrastructure appearance in electron microscopy images.

soma. This neuron can receive information from multiple other neurons via synapses formed at the dendrite branches' ends (the part of a synapse which resides at the end of a dendrite branch is called the *postsynaptic terminal*). A single *axon*, a tubular process which can extend over large distances, also emerges from the soma. At its destination, the axon branches and forms *presynaptic terminals* (which are part of synapses relaying information to the dendritic branches of multiple other neurons). Neurons are the main signaling units of the brain: activated synapses cause graded electrical signals to travel from postsynaptic terminals towards the soma. If the sum of all converging graded potentials is above some threshold, an all-or-nothing electrical impulse is generated which travels along the axon (the neuron *spikes*). When reaching the presynaptic terminals, this impulse stimulates the emission of neurotransmitter molecules from their containers, the *vesicles.* These molecules drift through the synaptic cleft separating the pre- and postsynaptic sites and are sensed by receptor molecules, which make up the *postsynaptic density.* Synapses provide *directed* connections between one presynaptic cell and one or several postsynaptic cells [77, 135]. Importantly, there exist different types of synapses: firing *excitatory synapses* contribute fractional votes towards the spiking of the postsynaptic neuron, whereas *inhibitory synapses* contribute negative fractional votes. The existence of inhibitory pathways prevents an exponential cascade of spiking neurons.

In addition to vesicles, other cell organelles, such as the energy-producing *mitochondria*, can be found within neurons. Figure 1.2 labels a mitochondrion and a synapse's ultrastructure in an electron microscopy image.

A large number of *glial cells* surround the neurons. They provide several functions, such as structural support, insulation, debris and excess neurotransmitter removal as well as nutritive and developmental functions [77, Chapter 2].

Of course, the above is a much simplified description of neural wiring and signaling. The discussion has mainly focused on stereotypic connections in mammalian brains, whereas in the fly brain, for example, many neurites are short and do not spike [135]. Furthermore, there are also synapses which relay information by electrical impulses directly instead of using the diffusion of neurotransmitter molecules. Exceptions from the rule of axo-dendritic connections also occur.

However, it is believed that individual neurons may nonetheless be summarized with a relatively simple voting model and that complex function arises because of the *specific connections* (and their relative strengths) that neurons make or do not make amongst each other [31, 61, 135, 114], a theory discussed in more detail in Section 1.3.

Consequently, the next section focuses on imaging techniques that enable us to see individual neurons and their connections via synapses.

1.2 Imaging at Neuron Resolution

Most structures in *neuropil* [61] – brain tissue containing a highly packed and entangled mass of axons, dendrites, cell bodies, glial cells and blood vessels – are tiny: somas have diameters of 50 µm, but the branching dendrites and axons can have diameters as low as 30 nm (dendrites in the fly mushroom body) or 40 nm (dendritic spine necks in mouse hippocampus), cf. [61]. To study the structure and dense connectivity of neuropil, high resolution imaging techniques – together with effective staining procedures – are necessary.

1.2.1 Light Microscopy

Abbe's diffraction theory limits the resolution capability of an optical microscope, such that most structures in neuropil cannot be resolved, as they are smaller than the wavelength of visible light (from red to blue: 700 nm to 300 nm). However [139], super-resolution techniques are now becoming available. For example the technique of fluorescence tag photoswitching [131] can now achieve a localization accuracy of 10 nm by combining many images with stochastic sparse emission of visible light from marker molecules.

For volumetric imaging, Micheva and Smith [111] have introduced *array tomography*, which first cuts the acrylic resin embedded tissue into ultrathin (50 nm) sections, which are then transferred onto glass slides to form a matrix of consecutive sections. The entire array can be stained with fluorescent markers and imaged with a light microscope. It is possible to elute the staining and to restain with different markers multiple times, allowing to obtain a stack of fluorescent images per section. As a last step, the tissue slices can even be stained and prepared for electron microscopy imaging (Section 1.2.2).

Another approach using light microscopy is the *brainbow* imaging technique [102, 34], which may allow to distinguish and trace individual neurons even with light microscopy resolution. Whereas Golgi staining only allows to map a small fraction of neurons, Jeff Lichtman and colleagues have developed a way to randomly express different ratios of red, green and blue fluorescent proteins in each neuron. *All* neurons are labeled, but each one has a distinctive color, drawn from a large pool of available colors. This may allow to differentiate neighboring neurites even in low resolution images, where their colors appear mixed. However, at present, the limited amount of colors coupled with the complexity of neuropil at visible wavelength scales precludes the use of brainbow imaging for connectivity analysis in the brain (J. Lichtman, talk at Society for Neuroscience annual meeting, 2013). For a recent review of light microscopy techniques for brain circuit mapping, see [122].

1.2.2 Electron Microscopy

An electron microscope can achieve a much higher lateral resolution by using *electrons* for imaging, which have a shorter De-Broglie wavelength than visible light. The course of the charged electrons can be altered by applying electric and magnetic fields, such that building a “magnetic lens” is possible.

Imaging in Two Dimensions

In the first electron microscopes [88], electrons were sent through a thin section of the sample and detected on the other side as a two-dimensional image (*Transmission Electron Microscopy*, TEM). Regions of the sample containing atoms with large atomic numbers will absorb and scatter more electrons than less electron dense regions, thus creating contrast in the detected image.

Transmission Electron Microscopy requires very thin sample sections (fractions of a micrometer), such that electrons can penetrate the sample. In *Scanning Electron Microscopy* (SEM), a thin beam of electrons raster-scans the surface of the sample. The number of electrons from elastic or inelastic back-scattering are detected at each location, consecutively forming a grayscale image. SEM therefore allows to image the two-dimensional surface of a thick sample without having to first cut a thin section off.

Staining and Embedding

Staining protocols, already in use for decades, are based on the selective binding of heavy metals (such as osmium, uranium or lead) to membranes or protein aggregates [31, 48]. For block-face volume electron microscopy techniques (discussed below), it is necessary to stain the entire sample “en-bloc” before embedding it into plastic material

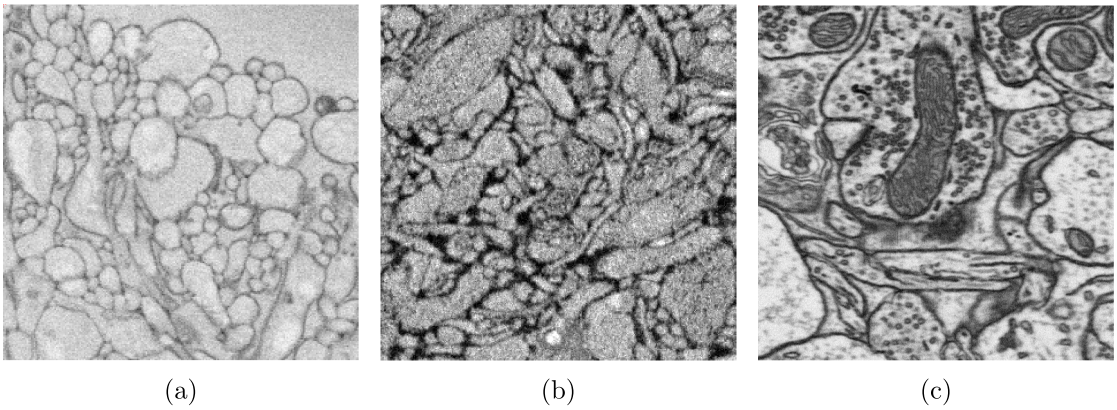


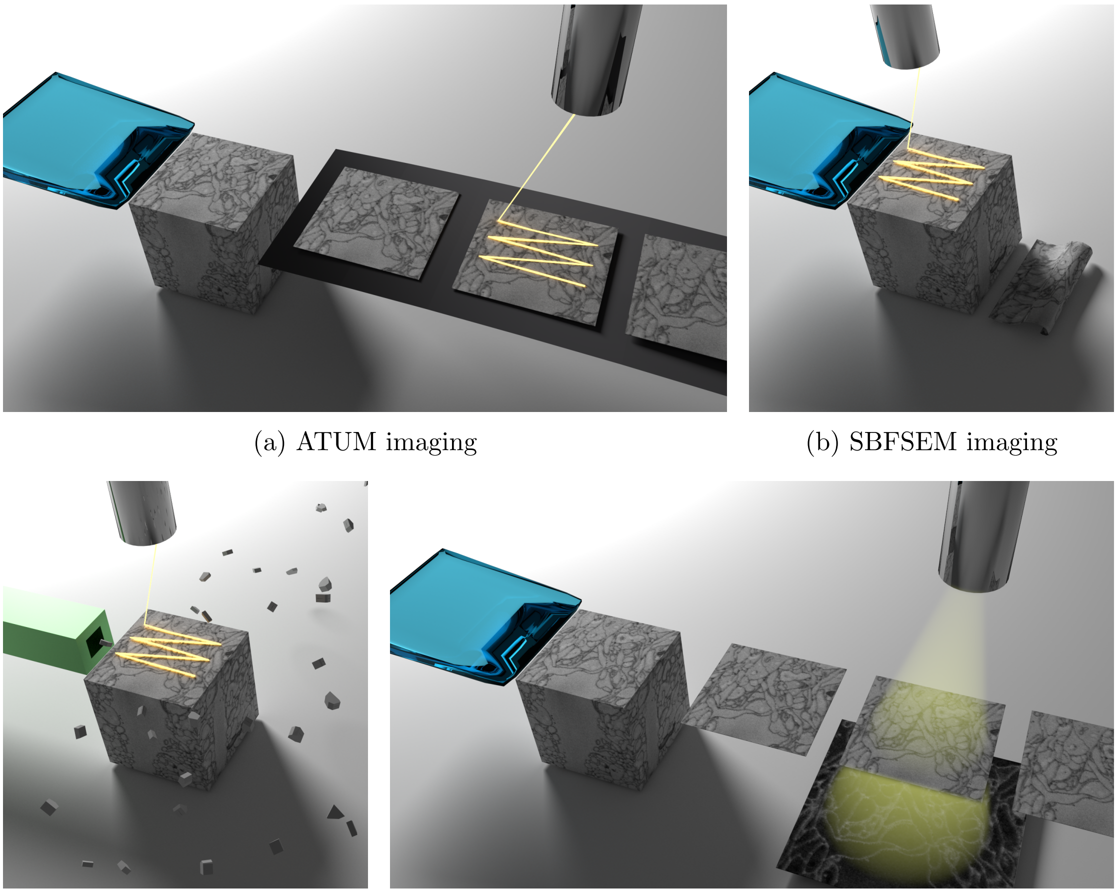
Figure 1.3: The choice of specimen, brain region, imaging technique and staining can produce EM images that look quite different.
(a) HRP staining (SBFSEM imaging; data courtesy of Kevin Briggman and Winfried Denk) contrasts the thin extra-cellular space against the intra-cellular space, but does not target intra-cellular organelles like mitochondria. (b) A different membrane-emphasizing staining protocol (SBFSEM imaging; preliminary work by Shawn Mikula towards whole-brain staining). (c) Conventional staining shows – apart from neurite membranes – also mitochondria, vesicles and synapses (FIBSEM imaging; data courtesy of Graham Knott).

for stabilization. In order to facilitate the tracing of neurites, heavy metal stains that emphasize the cell surface have been developed. For example, introducing horseradish peroxidase (HRP) into the living tissue triggers the production of a polymer with high osmium affinity [33, 63, 48]. However, this staining protocol de-emphasizes intracellular structures, such as vesicles or postsynaptic densities, which renders synapse detection much harder. Figure 1.3 visually compares different staining protocols.

Recently, Mikula et al. have demonstrated [112] that the whole mouse brain can be stained and embedded, such that myelinated axons can be traced with high accuracy. Axons that travel long distances from one brain region to another are electrically insulated from each other by myelin sheaths. Even though this method does not yet allow the tracing of all neurites, tracing just myelinated axons will enable researchers to map the *inter-areal connectome* or *projectome*. Current research indicates (S. Mikula, personal communication) that whole brain staining and imaging will enable the reliable tracing of all processes in the near future.

Volume Electron Microscopy

In order to be able to reconstruct the morphology and connectivity of a set of densely packed neurons, *three*-dimensional imaging techniques are needed. Current techniques all build on *serial sectioning*, in which thin slices are repeatedly sliced, scraped or milled



(a) ATUM imaging

(b) SBFSEM imaging

(c) FIBSEM imaging

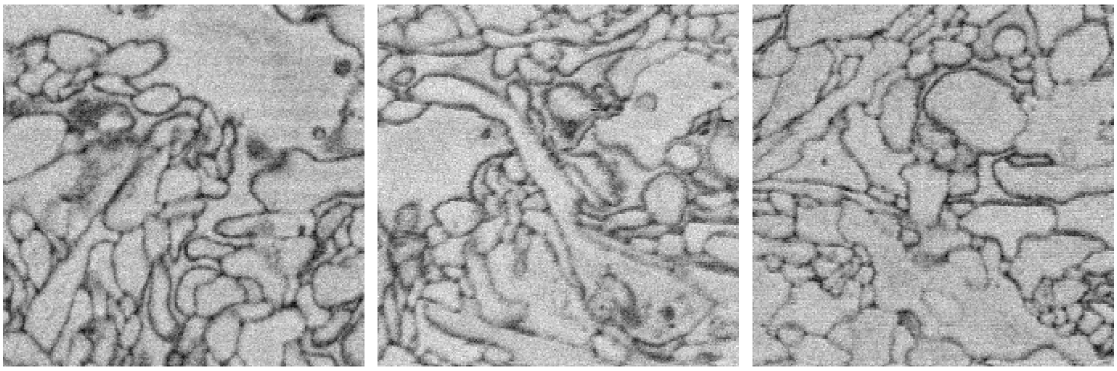
(d) ssTEM imaging

Figure 1.4: Illustration of volume electron microscopy techniques used for connectomics analysis. All methods have serial sectioning in common.

off the embedded tissue block's surface. Two dimensional electron microscopy is used to either image each of the cut slices or to image the block face exposed by scraping. The resulting images are then stacked along the z-axis to yield a three dimensional volume image, which can have a z-resolution of five to 50 nm, depending on the cutting method.

Individual techniques differ in the way how and when images are taken (transmission or scanning EM, before or after cutting) and in the way how slices are removed from the block (diamond knife or focused ion beam milling). Figure 1.4 illustrates these methods.

In *serial section Transmission Electron Microscopy* (ssTEM [154], Figure 1.4d) an ultramicrotome uses a diamond knife to cut a slice (approximately 50 nm thick) off the top of a block of epoxy resin embedded neural tissue. The slice is then transferred to a transmission electron microscope and imaged. In this manner, thousands of consecutive



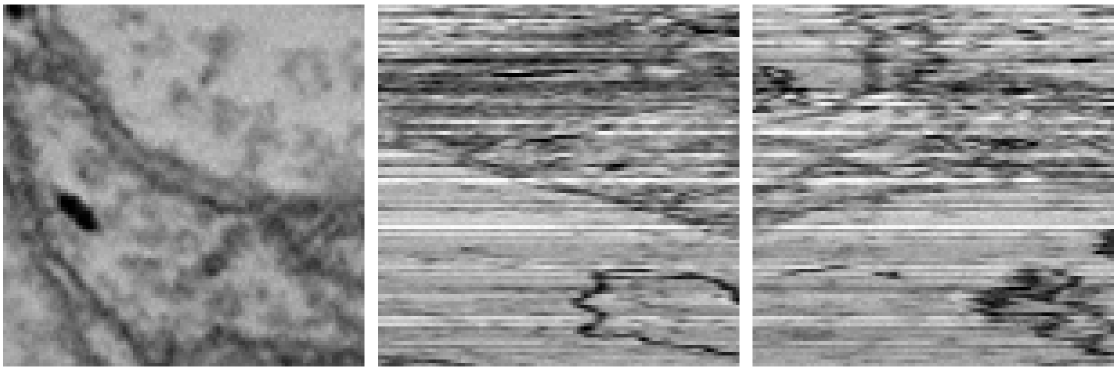
(a) x, y-view (b) y, z-view (c) x, z-view

Figure 1.5: The SBFSEM data from [63] is as good as isotropic: even though serial sectioning produced a z-stack of (x, y)-images (a), reslicing (b, c) produces images with what appears to be the same quality.

slices are imaged. An advantage of this method is the high throughput and high lateral resolution that can be obtained with high-energy electron beams. A disadvantage is that ssTEM involves transferring the slices from the ultramicrotome to the microscope. As the slices are very thin, distortions, folds, tears or loss may occur in the process, which at best complicate subsequent registration of the image stack [133], but may also lead to irrevocable information loss (missing slices, folds). Nonetheless, ssTEM has been used for the reconstruction of the C. Elegans nervous system [155] and more recently in a connectomics study of fly medulla (Takemura et al. [146], 1769 slices at 40 nm thickness) and of mouse primary visual cortex (Bock et al. [25], 1215 slices at 50 nm thickness imaged with a fast camera array).

The *Automated Tape-collecting Ultramicrotome* (ATUM, Hayworth et al. [60], cf. Figure 1.4a) makes the collection of slices more robust. A plastic tape automatically collects the slices one after another. The slices stick to the tape, which can be rolled up onto reels for storage. Because of the thick tape, ATUM uses scanning electron microscopy to capture the image. As the sliced brain tissue is physically stored on tape, individual section can be imaged again later, for example at higher resolution, when necessary [135].

Serial Block-Face Scanning Electron Microscopy invented by Denk and Horstmann in 2004 (SBFSEM or SBEM [49], Figure 1.4b) transfers the ultramicrotome into the low-vacuum chamber of a scanning electron microscope, which detects backscattered electrons. The block-face of the tissue sample is first imaged with SEM. Then, a diamond knife scrapes away the surface of the block. Because the image is acquired before cutting, the slice thickness can be smaller than in ssTEM, where slices have to be transferred for imaging. Repeated scraping and imaging yields a stack of images, which is easier to



(a) x, y-view (b) y, z-view (c) x, z-view

Figure 1.6: Part of the anisotropic dataset for which an automated 3D neurite segmentation algorithm was sought in an ISBI 2013 challenge. While membranes are easy to see in (a), note the low resolution, registration artifacts and varying contrast in (c) and (b).

align than ssTEM image stacks. The resulting volume image typically contains fewer artifacts. A recent connectomics study by Helmstaedter et al. [63] of the inner plexiform layer in the mouse retina used SBFSEM to acquire 3200 serial sections with 16.5 nm lateral resolution and 25 nm cutting thickness.

Focused Ion Beam Scanning Electron Microscopy (FIBSEM, Knott et al. [90], cf. Figure 1.4c) is similar to SBFSEM, but replaces the diamond knife with a focused ion beam which ablates the top few nanometers off the block face. Using this method, unprecedented isotropic resolution of 5 nm^3 could be achieved. However, the small field of view (smallest dimension at most 40 µm, [61]) currently precludes the analysis of many interesting neuronal circuits. Nonetheless, efforts are underway by a team of scientists at Janelia Farm Research Campus to analyze small circuits in fly medulla using a FIBSEM volume image.

Reviews of the different electron microscopy techniques can be found in [32, 31, 61]. Knott and Genoud argue [89] why electron microscopy remains a vital imaging technique to this day in various other areas of the biological sciences.

1.3 Connectivity Matters

In 1986, J.G. White et al. published the first (and until today only) complete synaptic wiring diagram of any organism [155]. They studied all 302 neurons of the hermaphrodite roundworm *Caenorhabditis Elegans* and how they are interconnected via synapses. In the 50 µm × 50 µm × 1000 µm [31] worm, the nervous system is spread throughout the body. White and colleagues used serial section Transmission Electron Microscopy to image the worm. Then, in a painstaking effort that lasted over a decade, neurons were identified and labeled on large printouts of each slice by hand and manually tracked across the preceding and following slices.

In 2005, Olaf Sporns coined the term "connectome" for the set of data describing "a comprehensive structural description of the network of elements and connections forming the [human] brain" [142]. The scientific field is known as "*connectomics*", similar to how *genomics* studies the genome. Neurobiologists believe that the knowledge of the wiring diagram will prove to be invaluable to "reverse engineer" the brain.

Winfried Denk draws an analogy to another computational device [48]: an electronic circuit consisting of binary logic gates (AND, OR, NOT). By applying all possible inputs and observing the outputs (*functional* measurements), it is in theory possible to obtain a complete description of the circuit's behavior. However, the amount of measurements required is exponential in the number of inputs. This is in contrast to the compact description of the circuit in a hardware description language. With knowledge of the circuit diagram, however, we can predict each output (given the functional description of the used logic gates). Importantly, meaningful connectivity diagrams are *sparse*, and can therefore be described much more compactly than the exponentially large value table of input-output relations. In the brain, the circuit elements are neurons and synapses (likely all stereotypical instances from a small set of types available), which can be well understood individually.

Various review articles discuss the promise of "connectomics" for the field of structural neurobiology at the macroscale (brain regions and pathways, [142, 43]) and the microscale (single neurons and synapses, [114, 48, 61]). Sebastian Seung, a prominent researcher in the field, has also written an accessible book [135] on connectomics.

1.3.1 Requirements for Cellular-Level Connectomics

Given the volume of tissue containing the circuit to be studied, obtaining the circuit diagram requires the reconstruction of most neuronal wires within. The only viable option for *dense* reconstruction is using high-throughput volume electron microscopy for imaging (Section 1.2.2) with dense staining, high lateral resolution, very thin cutting, and highly reliable, preferably automated data acquisition.

The electron-dense staining targets each neuron's membranes (as well as cell organelles' membranes) indiscriminately, making all of them visible at the same time. In order to reliably detect synapses, vesicles and the postsynaptic density need to be stained as well. Because neuronal processes do not necessarily have a preferred direction, the lowest resolution dimension should account for the smallest occurring neurite diameters [61]. In the volume image, the smallest tubular cross-sections should have diameters of a few voxels. Traceability not only depends on achieving the required minimal resolution, but also on the reliability of the staining (no artifacts, homogeneous throughout the volume) and the ability to collect a large number of slices without artifacts (no slice loss, no folds).

Helmstaedter and Briggman give examples [61, 31] of minimal circuit volumes considered meaningful: mouse retina ($40\,\mu m^3$), mouse olfactory bulb ($300\,\mu m^3$) and mouse cortical column ($400\,\mu m \times 400\,\mu m \times 1000\,\mu m$). Assuming an isotropic resolution of $20\,nm^3$, and 8 bits per voxel, this would amount to data sizes of 20 GB for the retina circuit, 10 TB for the olfactory bulb circuit and 60 TB for the cortical column circuit. With SBF-SEM, the acquisition of the cortical column dataset would take in the order of one year, but significant speed-ups are already on the horizon with the introduction of multi-beam scanning electron microscopes.

1.3.2 The Need for Automation

Even though acquisition times of one year may sound long, image acquisition is fast compared to *data analysis*. The accurate neurite reconstruction from the acquired volume images is much more time consuming and labour intensive.

One approach is to manually trace the contour of each neuron in each slice and then to link these contours across consecutive slices (*contouring*). The resulting 3D reconstruction of the local neurite morphology is invaluable in the study of synaptic ultrastructure, cf. [113, 58]. For connectivity studies, volumetric reconstructions are not required; it is sufficient to trace only the center lines (*skeleton tracing*), which results in a speed up of approx. 50 times compared to contouring [61].

Branching neurites must be traced with high accuracy, because any mistraced branch causes a large number of synapses to be assigned to the wrong neurons. Because humans often make attention-related mistakes, or disagree about the interpretation of the raw data at difficult locations, the skeleton of each neuron is often traced redundantly by many tracers [62, 146, 31, 63] after which a consensus skeleton is constructed [62].

The manual reconstruction of the C. Elegans connectome from paper printouts of ssTEM images needed a few thousand hours analysis time spread over the course of 12 years [155, 61]. A computer-assisted (but still essentially manual) workflow streamlines skeleton annotation, such that analysis times on the same data could be reduced ten-fold [157]. A recent connectomics study of mouse retina by Helmstaedter et al. [63] needed a combined annotation time of 30 000 hours.

Clearly, automated methods are necessary: the volume of neural tissue that can be imaged using serial EM methods is rapidly increasing, while humans are slow in tracing and make attention-related errors. The time of expert annotators is also too valuable to be spent on dull tasks such as neurite tracing and synapse identification.

This is why this thesis focuses on *fully automated methods* for neurite segmentation in electron microscopy volume images of neural tissue.

Chapter 2

Related Work

In this chapter, we give an overview of existing approaches to cope with the huge volumetric datasets obtained for connectomics studies.

Section 2.1 summarizes various existing *software packages* for visualizing and exploring connectomics datasets. For each software, we describe its approach to handling the massive dataset sizes and highlight the different methods offered for manual or semi-automated annotation. This summary will be helpful when we describe our own software, ILASTIK, in Section 3.1. Methods for *semi-automated segmentation* of neurites are further discussed in Section 2.2.

Ultimately, interactive annotation should only be needed for teaching a computer what to do. Once properly parameterized (or *trained*), a computer algorithm would then be able to analyze vast amounts of data at little cost. Section 2.3 highlights automated methods for detecting and segmenting *neuron ultrastructure*, in particular mitochondria and synapses. Importantly, Section 2.4 focuses on related work in *fully automated neurite segmentation*, the main topic of this thesis.

The question of how to quantitatively evaluate a neurite segmentation algorithm is the topic of Section 2.5. The introduced *segmentation quality measures* are used later for evaluating the proposed multicut segmentation algorithm in Chapters 4, 5 and 6 and are used as a loss function for learning in Chapter 5.

tool	display	tracing/annotation	RAM limited?	Open Source
Reconstruct [52]	2D+z	2D contour	no	yes[a]
TrakEM2 [37]	2D+z	2D contour, skeletons	no	yes[b]
Elegance [157]	2D+z	skeletons	no	yes[c]
CATMAID [132]	2D+z	2D contour, skeletons	no	yes[d]
Raveler [120]	2D+z	proofreading (split/merge)	no	yes[e]
KNOSSOS [62]	3D orthogonal	skeletons	no	yes[f]
itk-snap [160]	3D orthogonal	3D active contours	yes	yes[g]
ilastik [140]	3D orthogonal	3D seeded region growing	no	yes[h]
Ssecrett [70]	arbitrary	skeletons	no	no
NeuroTrace [71, 70]	arbitrary	2D active contour + tracking	no	no
ConnectomeExplorer [22, 21]	arbitrary	n/a	no	no

[a] https://github.com/meawoppl/reconstruct-1101
[b] http://repo.or.cz/w/trakem2.git
[c] https://github.com/Emmonslab/Elegance
[d] https://github.com/acardona/CATMAID
[e] https://openwiki.janelia.org/wiki/display/flyem/Raveler
[f] http://code.google.com/p/knossos-skeletonizer
[g] http://sourceforge.net/projects/itk-snap
[h] https://github.com/ilastik

Table 2.1: Overview of various tools for visualization, manual tracing and semi-automated tracing of electron microscopy volume images of neural tissue.

2.1 Visualization and Manual Analysis

Various software tools have been written for visualization and to support human tracers in their analysis of connectomics datasets. As we introduce ILASTIK and its viewer component, VOLUMINA, in Sections 3.1 and 3.2, we give a short overview of related work here.

Some applications (Reconstruct, TrakEM2, Elegance, CATMAID) are designed for anisotropic data, for which a preferred "depth" direction exists. The data is treated as a set of two-dimensional images stacked in z-direction. Annotation is performed on each (x, y)-slice individually and tracked across z. The user interacts with the data volume by scrolling through the image stack.

Other applications (KNOSSOS, ilastik) are designed with isotropic data in mind. To interact with the data, they usually offer three orthogonal slice views, (x, y), (x, z) and (y, z). Annotation can either be performed in 2D, but on any of the views, or in 3D, for example by constructing a sphere centered on one of the slice views. Applications such as Ssecrett and ConnectomeExplorer even allow to display arbitrary 2D slicing views through the volumetric dataset by resampling the data.

Software for Viewing and Analysing Volume Images

- Reconstruct [52] focuses on dense reconstruction in anisotropic volume data. To segment an object, the user traces the contour of the object on each slice. The program then links all contours together and calculates a surface representation of the object. Contouring can be semi-automated by using a 2D region growing algorithm (with a user-defined stopping criterion) to generate contours. Reconstruct has been used for example in [113, 58]; however, the last release (version 1.1) was in 2007. Reconstruct is written in C++ targeting the Win32 API.

- Ssecrett or the Serial Section Reconstruction and Tracing Tool [70] is a volumetric dataset viewer (allowing arbitrary slicing) and a manual skeleton tracing tool with a client-server architecture to support massive datasets.
  The companion module NeuroTrace [71, 70] implements a 2D segmentation plus tracking approach for the reconstruction of tubular structures. After manual initialization, two nested level sets converge under the influence of various internal and external forces to the inner and outer wall of the cell membrane, yielding a segmentation of the two-dimensional cross-section. The algorithm then attempts to follow the centerline of the tubular structure by tracking the cross section across the depth dimension.

- ConnectomeExplorer [22, 21], developed by the same research group after their previous tools Ssecrett and NeuroTrace, focuses on the interactive exploration of volume segmentation and annotation data. A client-server architecture allows for arbitrary slicing and large-scale on-demand volume rendering. The program as well as the source code for Ssecrett, NeuroTrace and ConnectomeExplorer are not publicly available, however.

- TrakEM2 [37] has been used for several connectomics reconstructions, for example of anisotropic serial section EM data from drosophila central nervous system by Cardona et al. [36] and mouse primary visual cortex by Bock et al. [25]. TrakEM2 is part of the Fiji[1] suite of image processing components and written in Java. During pre-processing of image stacks, it computes tiled mipmaps of each slice. Similar to Reconstruct, it offers various tools to semi-automatically label an area based on user clicks.

- Catmaid or the Collaborative Annotation Toolkit for Massive Amounts of Image Data [132] offers similar functionality to TrakEM2, but consists of a web front end (HTML + JavaScript) and server back end (built with Django), such that collaborative annotation over the internet is easily possible.

[1] `http://fiji.sc`

- Elegance [157] is a tool for skeleton reconstruction of anisotropic datasets and has been used to re-trace the data of the first published complete connectome [155]. Unlike its competitors, Elegance shows three consecutive sections side-by-side for context. Elegance is written in Java.

- Raveler [120] is a manual annotation tool developed by the FlyEM team at Janelia Farm. Raveler is written in C++/Python, using the Tkinter[2] toolkit. It has been released as open source software at the end of 2013.

- Knossos [62] is designed for redundant skeleton tracing in isotropic datasets. The data can be viewed and navigated using three orthogonal slice views; data is loaded on-demand from pre-generated 3D mipmaps stored on disk. Skeletons are traced manually, but a streamlined workflow (for example revisiting marked branching points) accelerates manual tracing. KNOSSOS has been used in recent connectomics studies by Helmstaedter et al. [62, 63]. Knossos is written in C/C++; a switch to the popular Qt[3] user interface toolkit is currently in development.

- itk-snap [160] offers orthogonal slice views for navigation through volumetric datasets that fit into RAM. It implements 3D active contour algorithms for interactive segmentation using C++ and the VTK and ITK toolkits.

- ilastik, the interactive learning and segmentation toolkit (Sommer et al. [140] and Sections 3.1, 3.2 and 3.3) enables interactive machine learning on large volumetric datasets, which are displayed using its orthogonal slicing viewer, volumina. ilastik performs computations on-demand on small chunks of data at a time, with volumina visualizing the resulting image blocks as soon as their computation is finished. ilastik's voxel classification workflow allows to interactively train and apply a voxel classifier on huge datasets.
  In addition, the Carving module provides semi-automated segmentation based on sparse user scribbles [144, 143].

To summarize, a plethora of software applications for working with EM volume images exist. Allmost all applications allow to stream parts of the dataset from disk for viewing regions of interest. While tools for manual and semi-automatic annotation are common, few programs offer automated processing of large datsets.

Each software has its own list of strengths and weaknesses, and most are tailored to the specific use cases of the research group that led the software development. I have contributed to the development of ilastik, being mainly responsible for the volumina viewer and the Carving user interface; for details, see Sections 3.1 and 3.3.

[2] http://docs.python.org/2/library/tkinter.html

[3] http://qt-project.org

2.2 Semi-Automated Segmentation

Fully manual annotation of neurites is too time-consuming [61]; Section 1.3.2 has argued why automation is needed. Ideally, a fully automated algorithm would trace all neurites without intervention by the domain expert to extremely high accuracy. As the problem (the focus of this thesis) is very difficult, it is worthwhile to consider *semi-automated segmentation methods*. These methods solve the easy parts of the problem automatically, but require help from the domain expert for the harder parts.
Semi-automated segmentation algorithms can be classified as

- *proof-reading* of dense automated segmentations [84] by manually correcting split and merge errors or by multiple-choice [158],
- utilizing *sparse annotations of membrane pixels* in order to constrain the segmentation of a 2D image slice [72]. Here, annotations are refined until *all* segments in the slice are correctly segmented,
- methods in which *one* neurite is being worked on at a time. One interaction mode is to let the annotator give *sparse scribbles indicating foreground* (the neurite of interest) and *background* (everything else). The constrained 2D or 3D segmentation is displayed, and further scribbles are given to correct any remaining errors [127, 158, 144, 143].

For obtaining correct results, semi-automated segmentation methods should

- minimize the number and complexity of manual annotations,
- provide interactive update speeds,
- guide the annotator to locations of uncertain segmentation.

All competitive *fully* automatic neurite segmentation approaches today rely on *machine learning*. The algorithms discussed in this section therefore have other uses: to quickly create precise and sufficiently large *training* and *validation sets*.

The ILASTIK software (Section 3.1) includes a module for semi-automated segmentation similar to Reconstruct, based on a region-growing algorithm (termed *Carving*) by Christoph Straehle et al. [144, 143]. Carving has been used for reconstruction of neurons in a recent study of correlative two-photon and FIBSEM microscopy [105]. Datasets can be larger than RAM if pre-processed beforehand (construction of a supervoxel-adjacency graph). In the re-implementation of ilastik, I have been responsible – apart from other tasks – for the user interface implementation and interaction design of the carving interface, which is described in more detail in Section 3.3. This tool has been used to acquire gold standard segmentations of small volumes of neural tissues, which have been useful for the quantitative evaluation of the proposed fully automated algorithms (Chapter 4 and Chapter 5).

2.3 Detection of Mitochondria and Synapses

In order to reconstruct the wiring diagram, tracing all branching neurites within the volume image ("segmentation", reviewed in Section 2.4) is not enough. Additionally, one needs to find all the synapses in the volume. It may also be useful to understand more about neurite ultrastructure, for example by an automated semantic labeling of voxels (classes vesicle, mitochondrion, neurite membrane, postsynaptic density, etc.; see Figure 1.2).

Synapses. Various automated synapse detection methods have been proposed [65, 115, 97, 18, 19]. Anna Kreshuk et al. propose a multi-step procedure [97]: a trained classifier predicts for each voxel whether it belongs to a synapse based on local image features. The probability volume is thresholded to yield connected components, which are filtered by size. All remaining components are reported as detected synapses. For training, the ilastik software (Chapter 3) was used. This method is able to find synapses in high-resolution isotropic FIBSEM images with high reliability. Carlos Becker et al. [18, 19] argue that context cues (synapses have nearby vesicles on the presynaptic side) should help to improve classification accuracy. Their method uses a boosting classifier to select relevant features from a large pool of context cues. After voxel classification, the method proceeds as in [97] to detect synapses. It is worth noting that none of these methods make use of neurite segmentation results, but rather treat the detection of synapses as decoupled from the reconstruction of neural wiring.

Mitochondria. Automated mitochondrion detection and segmentation has also been studied [104, 103, 136]. The distribution of mitochondria within neural tissue is an interesting biological question in its own right. However, even those studying *neurite* segmentation need to be concerned with mitochondria, because when in close proximity to neurite membranes they tend to confuse automated segmentation algorithms.

Aurélien Lucchi et al. [104, 103] first divide the volume image into supervoxels using the SLIC algorithm [1], and then treat mitochondrion segmentation as a binary labeling problem on the supervoxel-adjacency-graph. For each supervoxel x_i and all pairs of adjacent supervoxels x_{ij}, a number of textural and shape features are computed. Supervised learning (in this case a SVM [23] classifier) is used to derive the weights which are combined into a submodular "graph cut" energy function [93].

Mojtaba Seyedhosseini et al. [136] use algebraic curve fitting on 2D image patches to extract textural and shape features. A Random Forest classifier [30] then predicts for each patch whether its center lies within a mitochondrion or not.

This previous work does not consider neurite segmentation and cell organelle detection as a joint problem, but considers the detection problem independently. In Chapter 7, we propose a joint segmentation and detection approach.

2.4 Fully Automated Segmentation

In Sections 2.1 and 2.2 we have reviewed methods for semi-automated tracing of individual neurons. Because the annotator has to re-initialize these methods for each neuron individually, they do not scale to connectomics-size datasets. Instead, a truly automated approach should trace *all* neurons without intervention by the user. Once parameterized correctly, such an algorithm can be applied on massive datasets.

Various review articles and commentaries [38, 68, 61, 35] highlight the importance of automated analysis. In particular, Moritz Helmstaedter writes in the 2013 special issue "Focus on Mapping the Brain" of Nature Methods [63]:

> "Analysis of electron microscopy data is thus meeting a substantial challenge: humans can solve difficult locations but are slow and make attention-based errors, whereas machines are efficient at solving easy locations but fail when neurites become small and their packing is dense. [. . .] The current accuracy of automated classifiers is at least three orders of magnitude less than what is needed for reconstruction of entire neurons."

Chapter 1 has described the brain as an intricate machine composed of virtually identical parts. The fact that there are only few neuron types, and that these types can only be distinguished by their overall morphology (as opposed to being distinguishable by local texture), makes the connectomics segmentation problem very challenging.

Telling two adjacent neurons apart is rendered so difficult because of the constraints of tissue preparation (staining) and electron microscopy imaging. For example, in Brainbow imaging, each neuron expresses a random fluorescent color. However, in electron microscopy, the heavy metal stain targets all membranes of the tissue volume indiscriminately. While this allows full reconstruction (unlike Golgi staining), the drawback is that neuropil appears as a tangled mass of identically looking neurites.

One neuron *can* however be distinguished from another because they are separated by a stained *membrane*. This is why all segmentation approaches revolve around membrane detection. If voxels belonging to neurite membranes could be labeled with 100% accuracy, the neurite segmentation problem would be solved.

In 2007, Viren Jain et al. discussed two simple baseline image processing approaches for neurite segmentation [67, Section 4]. Their dataset, similar to the one shown in Figure 1.3a, contrasted the bright intra-cellular space against the dark extra-cellular space and therefore lent itself to binary labeling. First, they tried thresholding the raw data (cf. Figure 2.6b). However, they found that the voxel-wise accuracy – even on the training set – rendered this approach useless. Next, in an attempt to de-noise the data, an edge-preserving anisotropic smoothing filter was applied to the raw data first, and after that thresholding was performed. This did not improve performance significantly.

Obviously, the noisy raw image data (Poisson noise, staining imperfections, imaging artifacts, registration problems, low resolution, small structures) requires far more complex models for automated analysis. All methods today use *machine learning* to find good parameterizations of such complex models with a supervised learning setup.

2.4.1 Categorization of Existing Approaches

Current approaches for tackling the neurite segmentation problem can be categorized as follows:

(1) Voxel classification. A classifier is trained that can predict, for each pixel or voxel in the image, if it belongs to a neurite membrane or not. This decision is based on local image features.

(1a) The classifier uses *designed* local image features. For example, the software package ilastik (Chapter 3, [140]) can be used to interactively train a Random Forest classifier, which uses a set of convolution-based features at various scales. Features explicitly designed for the neurite segmentation problem have also been proposed [99]. After prediction, thresholding the probability map and connected component analysis can be used to obtain a segmentation. However, voxel classification is usually just a pre-processing step in a segmentation pipeline [7, 10].

(1b) Voxel classification using *learned* local image features, and convolutional neural networks [149, 66, 68, 150, 64, 40, 108, 74, 75]. As in (1a) this may only be a pre-processing step; see [119, 69].

(1c) Mojtaba Seyedhosseini et al. study how to incorporate multi-scale context [137] and develop a hierarchical model for voxel classification [138].

(2) Pixel-level regularization. Because pixel classification is noisy, regularization is desirable. Verena Kaynig et al. propose a graph cut regularization method over all pixels in a 2D image [82, 84]. Their method is designed to close holes in the segmentation of membranes.

(3) 2D segmentation and linking. For anisotropic image stacks with a preferred z-direction a common approach is to first propose one or multiple segmentations for each 2D slice independently and then find a linking of the segments across all slices to form neurites [83, 55, 151, 84, 74]. To take context into account, one may formulate the segmentation of two consecutive slices as a co-clustering problem [153].

(4) Supervoxel-based algorithms. The volume image is first decomposed into *supervoxels*. Supervoxels are small connected components of voxels that adhere to the local edge structure present in the data. Supervoxel algorithms have to balance two opposing requirements: on the one hand, supervoxels should be as

large as possible while on the other hand, they should never straddle more than one semantic object (e.g. a neurite). All algorithms proceed to build the supervoxel adjacency graph RAG $= (\mathcal{V}, \mathcal{E})$. The segmentation problem is then to find the sequence of merge operations on RAG that yields the desired segmentation into neurites.

(4a) In *Agglomerative Clustering*, supervoxels are merged sequentially. The merge order is determined by a greedy algorithm [69, 119, 76, 101, 26].

(4b) In this thesis, we consider a *simultaneous decision* of all pairs of adjacent supervoxels whether they should be merged together or not [7, 10, 98, 20].

In the following, these approaches are discussed in more detail. Voxel classification using Random Forest classifiers (1a) and Neural Networks (1b) is the subject of Section 2.4.2. Pixel-level regularization (2) is discussed in Section 2.4.3. For anisotropic datasets, 2D segmentation and linking (3) is the method of choice (Section 2.4.4). Finally, we discuss supervoxel-based algorithms (4) in Section 2.4.5.

2.4.2 Voxel Classification

Supervised machine learning methods [23, 59] can be used to obtain a decision function which assigns a class label to every voxel based on local image features. More precisely, let $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$ be the set of N training samples. Each sample consists of a *feature vector*, $\boldsymbol{x} \in \mathbb{R}^m$, and a *target label y*. In *classification*, y can attain values from a discrete set $\mathcal{L}$, such as $y \in \{\text{membrane, non-membrane}\}$. The aim of supervised machine learning is to use the training set to find a mapping

$$f : \mathbb{R}^m \to \mathcal{L} \ , \tag{2.1}$$

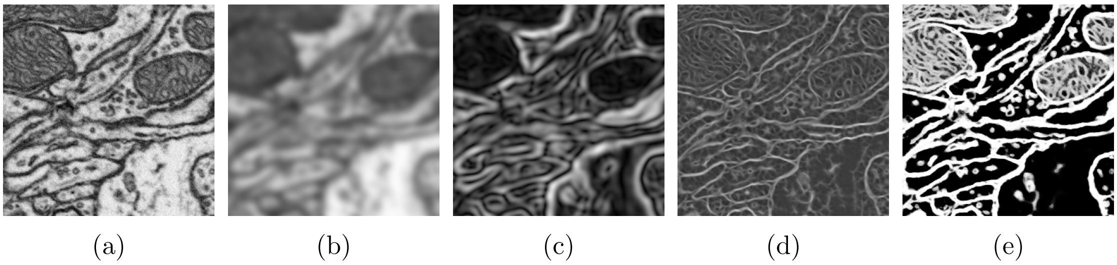
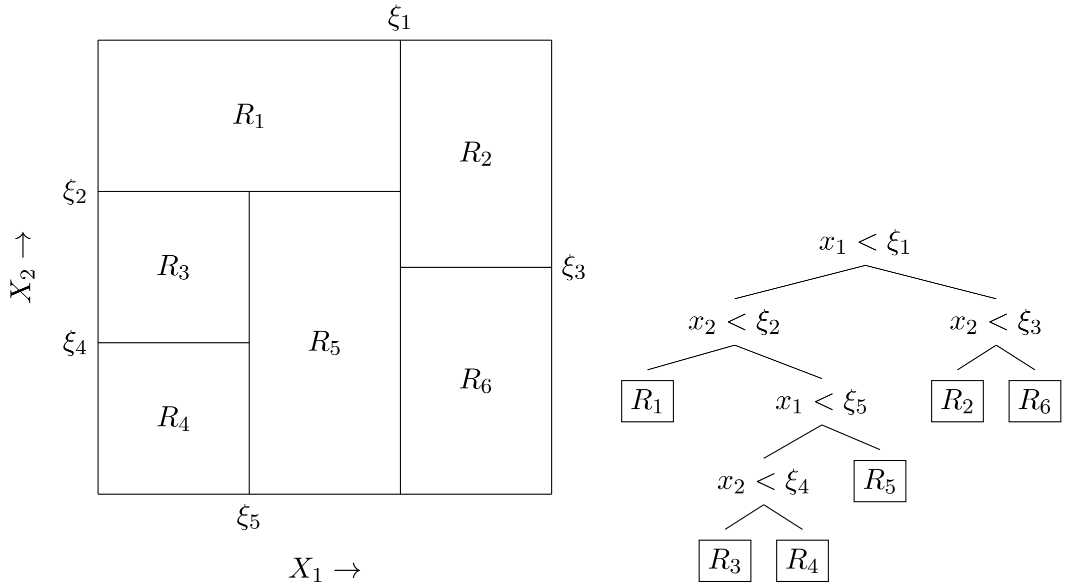


Figure 2.1: From the raw data (a), voxel features include the smoothed raw data (b), the gradient magnitude (c) and the greatest eigenvalue of the Hessian Matrix (d). Using the ILASTIK software, a Random Forest classifier was trained, which yielded the output (e). Data courtesy of Graham Knott.



(a)

(b)

Figure 2.2: Illustration of the idea behind Classification and Regression Trees, inspired by [59], Chapter 9. A partitioning of the (two-dimensional) feature space (a) induces a binary decision tree (b).

such that f is likely to perform well on previously unseen test samples. The mapping f is called a *classifier*. In the simplest case, performance is measured by the *misclassification rate*, i.e. the fraction of training samples that are assigned a wrong label. To avoid overfitting one typically tunes a regularization hyper-parameter which limits model complexity.

Figure 2.1 illustrates voxel classification: given the raw data, a set of different voxel features is computed, which are stacked to make up the feature vector $\boldsymbol{x}$ for each voxel. Based on these features, a trained classifier outputs a probability for each voxel to belong to the "membrane" class on the test data.

Voxel Classification with Random Forests

A *Random Forest* classifier [30, 59, 44] is an ensemble of decision trees. To classify a sample $\boldsymbol{x}$, each decision tree casts a vote for a specific class label $l \in \mathcal{L} = \{1, \ldots, |\mathcal{L}|\}$. The majority vote of all trees in the ensemble decides the output of the classifier f for sample $\boldsymbol{x}$.

Figure 2.2b illustrates a binary *decision tree*, consisting of nodes and edges. *Leaf nodes* have no children, while *split nodes* have exactly two child nodes. Each split node is associated with a test function to be applied to data. Each leaf node stores a class label. Given a sample $\boldsymbol{x} \in \mathbb{R}^m$ from the test set, the decision tree applies a sequence of test functions in order to predict the label of $\boldsymbol{x}$. First, $\boldsymbol{x}$ is passed to the root nodes' test function. If it yields "true", the datum $\boldsymbol{x}$ is passed to the left child, if it yields "false", $\boldsymbol{x}$ is passed to the right child. This process is continued until a leaf node is reached. The stored label determines the classification decision of the tree for the given sample.

In Random Forests, each test function is usually chosen to divide the feature space $\mathbb{R}^m$ into two halves with an axis-aligned split: feature $\boldsymbol{x}_k$, $1 \leq k \leq m$, is tested for $\boldsymbol{x}_k < \xi_k$ with an abitrary threshold $\xi_k \in \mathbb{R}$. A decision tree thus partitions the feature space into a set of regions, which are each assigned a label via the leaf nodes. This is illustrated in Figure 2.2.

The ensemble of decision trees is constructed in a greedy fashion from the training data. In order to avoid overfitting to the training data,

- each tree is constructed independently,
- each tree sees a different training set, generated with bagging,
- the optimal decision function for each split is found only from a random subset of all possible decision functions,
- the output of all trees in the ensemble is averaged to reduce bias.

To train a single decision tree, a new training set $\bar{\mathcal{S}}$ with $|\bar{\mathcal{S}}| = |\mathcal{S}|$ is first generated by sampling uniformly with replacement from the original training set $\mathcal{S}$. This procedure is called *bagging*. Starting at the root node, the decision tree is then constructed greedily. At each node τ, the algorithm considers only that subset $\bar{\mathcal{S}}_\tau \subset \bar{\mathcal{S}}$ of the training samples for which the previously generated decision functions on the path from τ to the root node apply. To find an appropriate decision rule $\boldsymbol{x}_k < \xi_k$ for node τ, a subset of all features $F \subset \{1, \ldots, m\}$ is considered (usually, one chooses $|F| = \sqrt{m}$). Then, an exhaustive search over all features $k \in F$, and over all possible splits ξ_k is performed, in order to find those parameters which maximize the decrease in Gini impurity [30] or entropy [44].

Let $n := |\bar{\mathcal{S}}_\tau|$ and n_j denote the number of samples in $\bar{\mathcal{S}}_\tau$ labeled $j \in \mathcal{L}$. Then the *gini impurity* $i(\tau)$ is defined as

$$i(\tau) = 1 - \sum_j \left(\frac{n_j}{n}\right)^2 \quad . \tag{2.2}$$

Based on the current choice of decision rule $\boldsymbol{x}_k < \xi_k$, the samples $\bar{\mathcal{S}}_\tau$ will be split into the sets $\bar{\mathcal{S}}_{\tau,\text{left}}$ and $\bar{\mathcal{S}}_{\tau,\text{right}}$, which have cardinality n_l and n_r, respectively. The decrease Δi in gini impurity is then calculated as

$$\Delta i(\tau) = i(\tau) - \frac{n_l}{n} \cdot i(n_l) - \frac{n_r}{n} \cdot i(n_r) \quad . \tag{2.3}$$

Node splitting stops when $n = 1$. Random Forests have the reputation to "do remarkably well, with very little tuning required" [59, Chapter 15], which is why they are the classifier of choice in the ILASTIK software, which we describe in more detail in Section 3.1.

In the proposed neurite segmentation algorithm (Chapter 4), we make use of Random Forest classifiers both for obtaining supervoxels from a voxel membrane probability map (voxel classification) and for separating true from false boundaries.

Voxel Classification with Neural Networks

In 2007, Viren Jain et al. proposed to use convolutional neural networks to tackle the neurite segmentation problem [67].

A feed-forward neural network is a universal function approximator [45]. It can be represented as a directed network, see Figure 2.3. In *convolutional neural networks* (CNNs), each arrow represents a convolution with a filter mask (fixed size) with weights to be parameterized. For example in [67], such a filter would be of size 5 × 5 × 5 voxels, representing 125 free parameters. Each *hidden node* (blue) represents a non-linear function applied to the biased sum of the node's input. Often this function is chosen to be a sigmoid $f : \mathbb{R} \to [0,1]$, $f(x) = 1/(1+e^{-x})$, see Figure 2.5. The nodes are organized in *hidden layers* k which consist of different *feature maps* indexed with a. In the architecture of [67], the output of a node I_a^k depends on all nodes in the previous layer:

$$I_a^k = f\left(\sum_b w_{ab}^k \otimes I_b^{k-1} - \theta_a^k\right) \quad , \tag{2.4}$$

where "$\otimes$" denotes convolution of the filter mask w_{ab}^k with the input image I_b^{k-1} and θ_a^k is a node-specific bias parameter. The function represented by such a CNN has many parameters: in Figure 2.3, there are 85 connections (5 each from the input and to the output, plus $3 \cdot 5^2$ for connections between the hidden layers). Assuming each connection represents a $5 \times 5 \times 5$ filter mask, and each hidden node contributes a bias, this sums to 10645 free parameters in total.

In a CNN architecture, each additional hidden layer increases the context that each voxel can use to compute its output value. For filters with fixed size F^3 and K layers, the context is $(F-1) \cdot K + 1$. In the example above, this is a context of 25^3 voxels.

For learning, one first has to choose a *loss function* Δ that defines a distance measure between the output image $\hat{X}$ produced by the neural network and the true solution X. A simple loss function considers each voxel x separately using a per-voxel loss $l(x,\hat{x})$ and then sums the result:

$$\Delta(X,\hat{X}) = \sum l(x,\hat{x}) \quad . \tag{2.5}$$

Examples (Figure 2.4) for the per-voxel loss are quadratic (Eq. 2.6), square-square

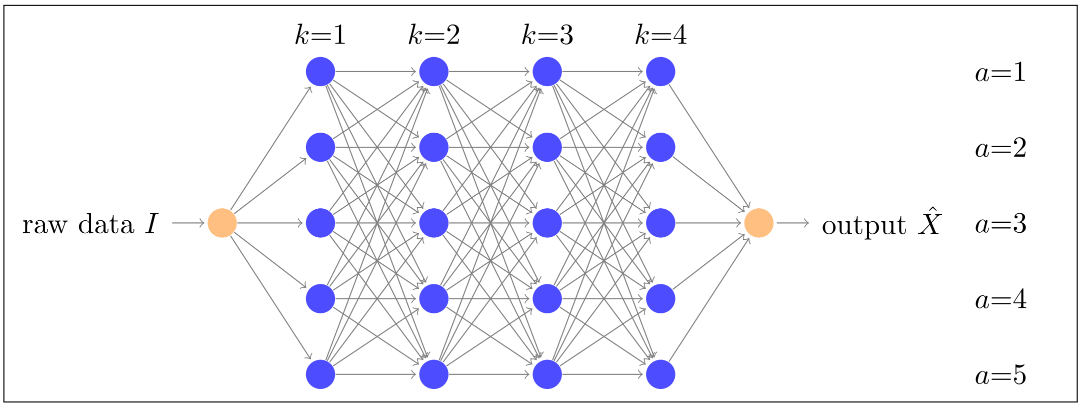


Figure 2.3: Setup of a convolutional network as used in [67, 149]. This network has four hidden layers (columns, $k = 1, \ldots, 4$) and five feature maps (rows, $a = 1, \ldots, 5$) at each depth layer.

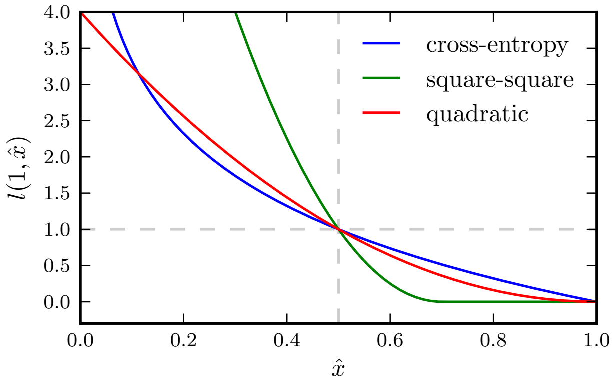


Figure 2.4: Different loss functions $l(x, \hat{x})$ for voxel classification and $x = 1$. The functions were rescaled to pass through the point $(0.5, 1)$. For square-square loss, $m = 0.3$ as in [149].

(Eq. 2.7, used in [149]) and the cross-entropy loss (Eq. 2.8, used in [67]).

$$l(x, \hat{x}) = \frac{1}{2}(x - \hat{x})^2 \ , \tag{2.6}$$

$$l(x, \hat{x}) = x \cdot \max(0, 1 - \hat{x} - m)^2 + (1 - x) \cdot \max(0, \hat{x} - m)^2 \ , \tag{2.7}$$

$$l(x, \hat{x}) = x \ln \hat{x} + (1 - x) \ln(1 - \hat{x}) \ . \tag{2.8}$$

The predicted image $\hat{X} = \hat{X}(I, \boldsymbol{w})$ is the complicated function represented by the neural network which depends on the input image I and the parameters $\boldsymbol{w}$. However, because $f(\cdot)$ is chosen as a sigmoid function, $\hat{X}(I, \boldsymbol{w})$ is a smooth and differentiable function. For training, the aim is to find the parameters $\boldsymbol{w}$ which minimize the loss $\Delta(X, \hat{X}(I, \boldsymbol{w}))$. Minimization can be performed by gradient descent algorithms. Training a feed-forward neural network using gradient descent is known as *backpropagation* from the work of Rumelhart et al. [130].

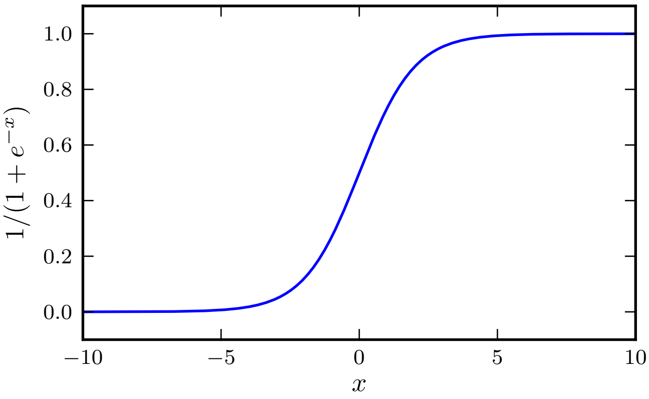
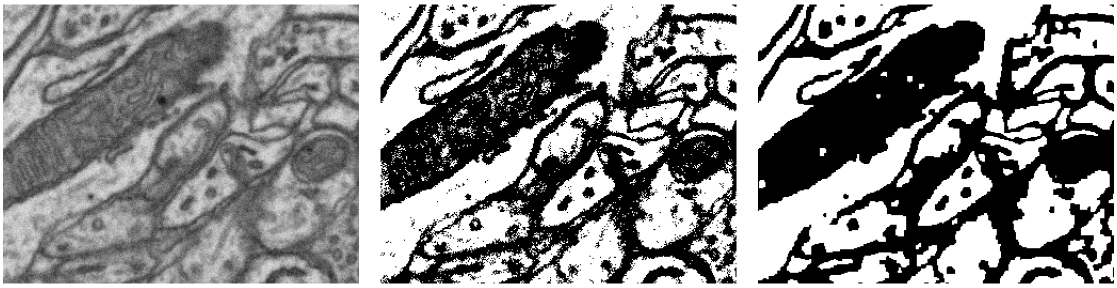


Figure 2.5: The sigmoid function $f : \mathbb{R} \to [0, 1]$ maps any real value to the range between zero and one. It is defined as $f(x) = 1/(1 + e^{-x})$.

Srinivas Turaga et al. [149] argue that a different loss function is better suited to the segmentation problem when the post-processing consists of thresholding $\hat{X}$ and computing the connected component labeling. As defined above, the loss function Δ just sums up the individual contributions l of each voxel to the loss. However, the accuracy of voxel classification must be weighted depending on the context: for some pixels, a wrong decision will not change the segmentation much; for others, this may have dramatic consequences. This concept, also crucial to our work, is discussed in more detail in Section 4.3.2.

The authors of [149] advocate using the *Rand Error* [124], a distance measure between two segmentations that is discussed in detail in Section 2.5.1. Briefly, each possible pair of pixels contributes to the loss if the segmentations disagree on whether this pair should be connected or not. The segmentation is formulated using the voxel-adjacency graph (called *affinity graph*). The CNN produces a real-valued affinity for each edge (the larger the affinity the stronger the belief that the voxels should be connected). Thresholding the affinity graph and connected component analysis yields a segmentation. In their stochastic gradient descent learning, the aim is to minimize the Rand Error on the training set. A stochastic gradient update step involves picking any pair of voxels i and j. Among all possible paths $i \to j$, they find the path which maximizes the minimal affinity along that path. The derivative of the loss l of that *maximin edge* (with respect to the parameters) is used for the gradient update.

The work of Turaga et al. has been influential for this thesis. The observation that some edges are more important than others when applying thresholding and connected component analysis has led to the multicut formulation of neurite segmentation (Chapter 4). By now, the use of the Rand Error (as well as the Variation of Information, see Section 2.5.2) has become a standard measure in image segmentation [13, 126] as well as neurite segmentation [119, 84, 149, 101, 10, 98, 40]. In Chapter 5, we investigate the use of Rand Error and Variation of Information for structured learning of multicut segmentation.



(a) raw data (b) $\lambda = 0$ (thresholding) (c) $\lambda = 0.1$

Figure 2.6: Graph cut segmentation on FIBSEM data. The raw image data D, normalized to range $[0, 1]$, was used to define the unary potentials $E_i(0) = D_i$, $E_i(1) = 1 - D_i$ and the pairwise potentials $E(0, 0) = E(1, 1) = 0$, $E(0, 1) = E(1, 0) = \lambda$.
Panel (b) amounts to thresholding. In (c), there is much less noise visible. However, some membranes have gaping holes (especially visible in the lower left). Data courtesy of Graham Knott.

Other neural network architectures, incorporating hierarchical sparse pixel neighborhood sampling [74, 76] or deep neural networks [64, 40, 108] (which include convolutional networks as sub-components) have recently been shown to yield impressive results for neurite segmentation. However, the segmentation quality is still not good enough for connectomics scale segmentation. Our segmentation methods (Chapters 4 to 7), can profit from improved voxel classification methods, as they all start from a supervoxel representation. In general, the better the classification performance on the voxel level, the better the supervoxel segmentation derived from it.

2.4.3 Pixel-Level Regularization

Markov Random Field (MRF) and Conditional Random Field (CRF) methods [24] have been enormously successful for many labeling problems in computer vision [128, 29]. Briefly, in a typical discrete CRF model of image segmentation, each pixel can attain one out of a set of labels (e.g. foreground, background). Each pixel quantifies its (noisy) desire to be assigned each label. To introduce regularization, one further specifies the *joint* preferences of pairs of pixels for all possible joint labelings. The problem of finding likely image labelings can then be approached as an inference problem on multidimensional probability distributions.

The popularity of CRF models is partly due to the *graph cut algorithm*, which can compute the most likely binary labeling (MAP estimate) in polynomial time for a certain subset of models [93].

More formally, such models are formulated using the pixel (or superpixel) adjacency graph RAG $= (\mathcal{V}, \mathcal{E})$. A binary labeling of the nodes $\mathcal{V}$ (pixels, superpixels) is denoted

$\boldsymbol{y} \in \mathcal{Y} = \{0,1\}^{|\mathcal{V}|}$. Any function $E_i : \{0,1\} \to \mathbb{R}$ gives the local compatibility of each label with the image evidence (the higher the value, the less likely the state is). Similarly, $E_{ij} : \{0,1\}^2 \to \mathbb{R}$ gives the compatibility of the labeling of pairs $(i,j) \in \mathcal{E}$ of adjacent pixels (superpixels) with the image evidence. One then defines an *energy function*

$$E(\boldsymbol{y}) = \sum_{i \in \mathcal{V}} E_i(y_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j) \ . \tag{2.9}$$

Alternatively, one can formulate the probability distribution over all possible labelings

$$P(Y = \boldsymbol{y}) = \frac{1}{Z} \exp\{-E(\boldsymbol{y})\} \ , \tag{2.10}$$

where Z is a normalization constant called the partition function. The MAP state can be found by maximizing $P(\boldsymbol{y})$ over $\boldsymbol{y} \in \mathcal{Y}$, or, alternatively by minimizing $E(\boldsymbol{y})$.

In “graph cut models” of binary image segmentation, the regularization is chosen to discourage label transitions. In the MAP labeling, this has the effect that small holes are filled and small speckles are eliminated.

To apply these methods to neurite segmentation, one could set up a binary labeling problem (neurite membrane versus everything else) over the pixels, and hope that the regularization terms will serve to close those small holes in the membranes which can have catastrophic consequences in connected component labeling on the probability map (see Chapter 4). However, this naïve approach does not work. Graph cut segmentation suffers from “shrinking bias” [152, 82] because it regularizes boundary length: thin, elongated structures (such as membranes!) are cut off (cf. Figure 2.6).

Verena Kaynig et al. have designed the terms E_{ij} specifically to overcome this problem by incorporating the orientedness in each pixel [82, 84]. However, in practice only small holes can be closed using this method.

Closing small gaps in membranes may also be achieved by other means; for example, Cory Jones et al. propose a method using partial differential equations [73].

2.4.4 2D Segmentation and Linking

For anisotropic data (e.g. obtained with ssTEM imaging), many researchers have approached the segmentation problem from a tracking perspective.

First, the cross sections of neurites are segmented individually on each slice. Here, automated segmentation algorithms can benefit from the high lateral resolution. However, membrane appearance varies with the angle between membrane normal and image plane (Figure 2.7): for small angles, the membrane appears thin and crisp; for larger angles the membrane appears faint and broadly smoothed out. This appearance variability in thick-slice data complicates 2D segmentation; the context of adjacent slices may be necessary for the correct interpretation [35].

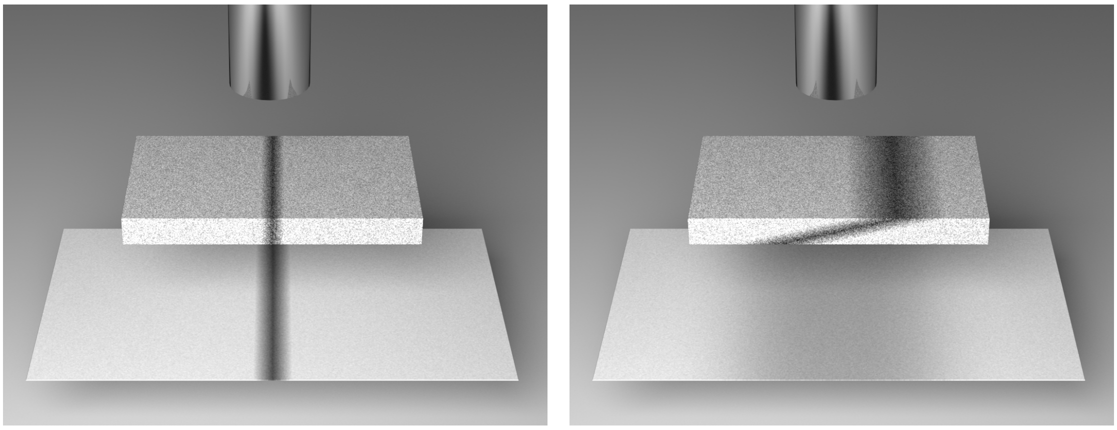


Figure 2.7: In thick-slice data, membranes appear different depending on their relative orientation with respect to the slice. *Left:* The membrane runs perpendicular to the cutting plane, resulting in a crisp image. *Right:* An oblique membrane produces a broad and fuzzy image.

Second, the individual segmentations are linked together by tracking the previously segmented 2D contours across slices. In this *tracking-by-association approach*, segments in adjacent slices may be associated by location and similarity of shape and internal texture. Figure 2.8 compares this approach to segmentation in isotropic volume data.

In the linking step, the algorithm needs to find a plausible tracking of *all* 2D segments across the image stack. This is further complicated by the fact the neurites may start, branch and end, which requires (apart from the common one-to-one matchings) also one-to-many (split), many-to-one (merge), none-to-one (appearance) and one-to-none (disappearance) assignments of 2D segments across adjacent z-slices. For non-branching neurons (traversing through the whole image stack) Elizabeth Jurrus et al. propose a greedy shortest-path algorithm [76] working on the hypothesis graph of possible region linkages. The current state-of-the-art is to propose many segmentation hypotheses for each slice [151, 55, 84] and then to find an optimal and consistent set of assignments.

In isotropic volume data, there is no preferred slicing direction. This makes it hard to apply the tracking approach on these datasets (of course, one could break the symmetry by choosing an arbitrary directionality). In this thesis, I work with isotropic data, for which true 3D segmentation approaches are developed.

2.4.5 Supervoxel-based Algorithms

Many algorithms begin by partitioning each image slice into *superpixels*, or, for isotropic data, partitioning the volume image into *supervoxels*. Superpixels "group pixels into perceptually meaningful atomic regions which can be used to replace the rigid structure

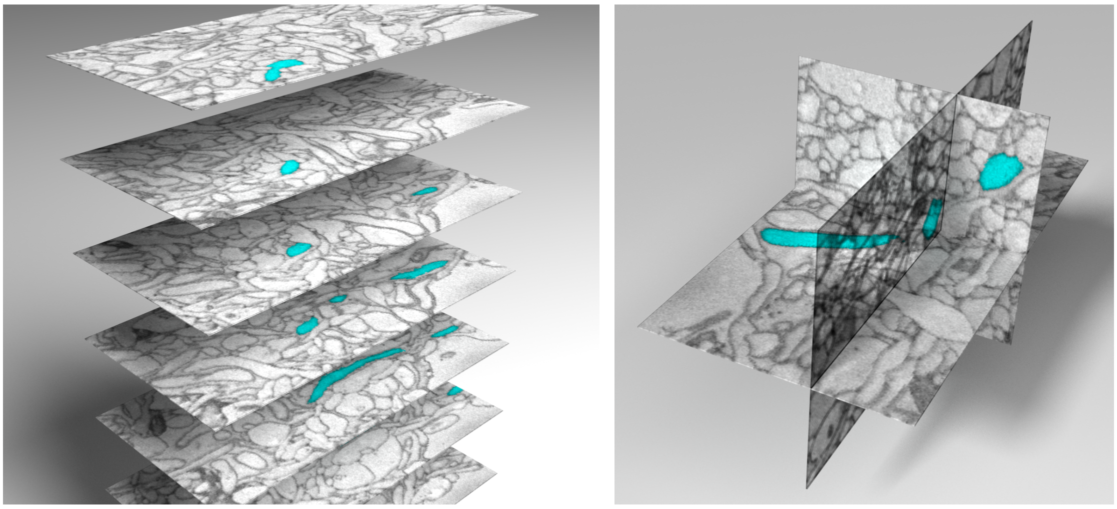
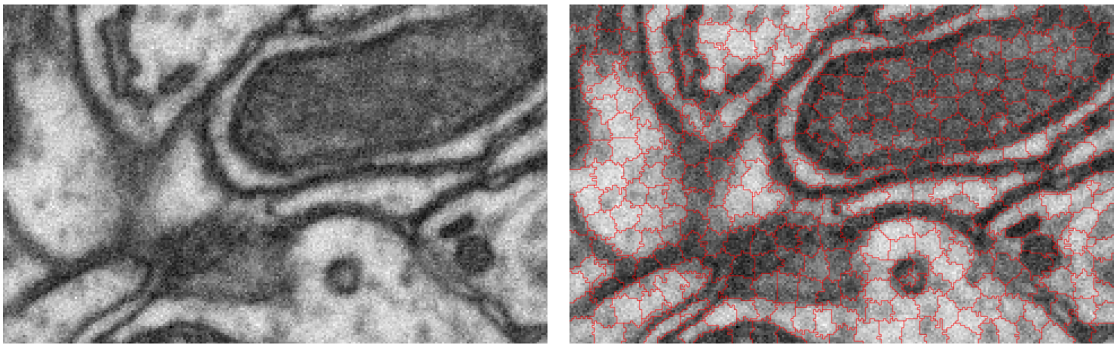


Figure 2.8: *Left*: Illustration of the 2D segmentation and linking approach. One neurite is tracked across z-slices. Note that contours can merge and split. *Right*: In isotropic volume data there is no preferred direction. Conceptually, 2D contours may be found on any arbitrary slicing plane through the data (shown: three perpendicular slicing planes).



(a) raw data

(b) SLIC superpixels

Figure 2.9: Using the SLIC algorithm [1] to generate superpixels on a slice of FIBSEM raw data. Notice that superpixels lie *on* membranes or *within* cytoplasm. Data courtesy of Graham Knott.

of the pixel grid. They capture image redundancy, provide a convenient primitive from which to compute image features, and greatly reduce the complexity of subsequent image processing tasks.” [1].

Superpixels have been used – to name only a few examples – for unsupervised partitioning [5, 159, 87], and semantic labeling [54] of photographs. Using superpixels has the following advantages and disadvantages:

(+) They drastically reduce the size of the graph RAG $= (\mathcal{V}, \mathcal{E})$ by transitioning from the pixel-adjacency graph to the *superpixel-adjacency* graph. For example, the graph cut approach of V. Kaynig et al. described in Section 2.4.3 could also have been executed on the superpixel graph for faster inference.

(+) Features computed from irregular superpixels are more expressive than features computed from a rigid pixel neighborhood.

(−) The above only holds if no superpixel ever straddles more than one atomic region (for example, in the context of Figure 2.9, it should never encompass both membrane pixels and non-membrane pixels; in Figure 2.10, it should never straddle two distinct intra-cellular regions). In practice, this is impossible to achieve and a compromise between superpixel size and the frequency of such errors needs to be made.

In the following, we describe two superpixel algorithms, SLIC and watershed, and use them to highlight important choices in the definition of what constituates a “meaningful atomic region”.

SLIC algorithm

The *simple linear iterative clustering* or SLIC algorithm [1] is a variant of k-means clustering for fast computation of superpixels which has been used in the work of Aurélien Lucchi et al. on mitochondrion segmentation [104, 103]. Figure 2.9 shows example output on EM data.

As input, the user sets a number k of desired superpixels. Then, cluster centers C_i are distributed equally on the images using a rigid grid with spacing S. Each center is then moved to the location of minimal gradient magnitude within a 3×3 neighborhood. Next, each pixel is assigned to the nearest cluster center within a spatial distance of $2S$. Here, “nearest cluster” is measured using a distance measure D which takes both spatial distance and color similarity into account. Finally, the cluster centers are updated as the mean of all assigned pixels (according to distance D). Assignment and center updates are repeated multiple times.

The above algorithm will find compact superpixels of similar-colored pixels. In Figure 2.9, this results in two types of superpixels: (1) dark superpixels which lie *on* a membrane, and (2) light superpixels which lie in the cytoplasm.

Watershed algorithm

The watershed algorithm [42] is a popular choice in neurite segmentation pipelines: it is used to generate 2D segments from neural network output in the ssTEM pipeline of E. Jurrus et al. [76] and to generate 3D supervoxels (for subsequent hierarchical clustering) in the pipeline of the FlyEM project at Janelia Farm [119, 146]. Supervoxels based on the watershed algorithm are also the starting point in the agglomerative clustering algorithm of Viren Jain et al. [69] and are used in the semi-automated "Carving" neurite segmentation algorithm [144], Section 3.3. In this thesis, we make use of the watershed algorithm to obtain supervoxels in all our algorithms described in Chapters 4 to 7.

The watershed transform of an image can be understood by thinking of the image as a topographic surface, where the gray scale value of pixel (x, y) gives the height at that point. Define a set of cluster centers C_i (called *seeds*; for now chosen as the local minima of the image). From these centers, water starts to pour into the surrounding basins, such that the water level rises uniformly across the topographic surface. Whenever water from two neighboring basins would flow together, an inter-pixel barrier – called a *watershed* – is constructed. All watersheds together define the boundaries between the superpixels. Note that each seed creates one superpixel.

Instead of choosing all local minima as seeds, the number of created superpixels can be substantially reduced by choosing the seeds in a different way (the algorithm is then called *seeded watershed* or *seeded region growing*). The algorithm is straightforward to apply to 3D data, too.

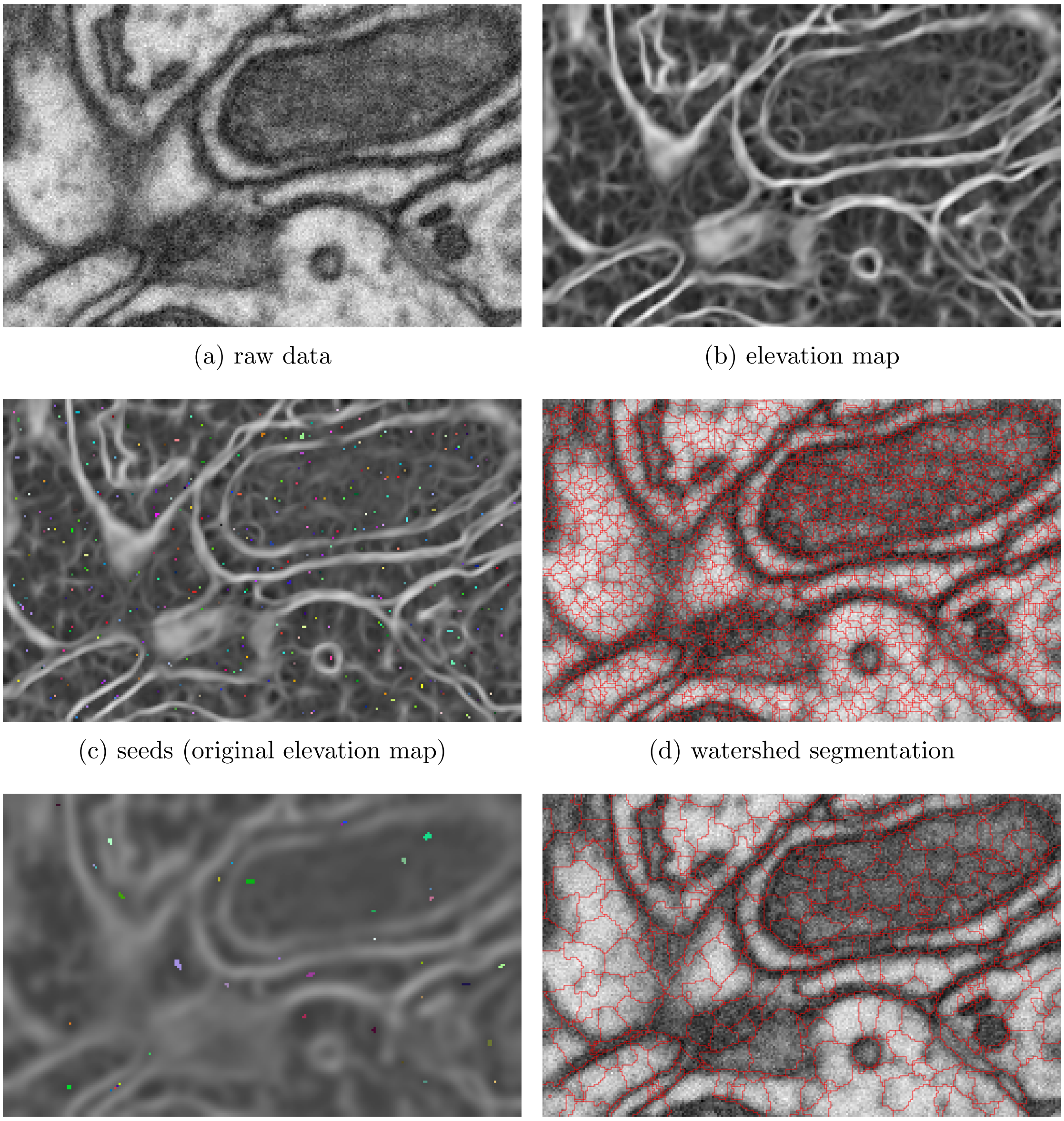
For neurite segmentation, we follow [8, 7] and compute a supervoxel segmentation by seeded region growing on a volume image in which the value of each voxel indicates the probability of it being part of a membrane. Such an image is also called an *elevation map*, which can be obtained by using ILASTIK to train a membrane versus non-membrane voxel classifier or simply by computing the largest eigenvalue of the Hessian matrix on the raw data. Seeds are obtained as local minima of a *smoothed* version of the elevation map. The amount of smoothing controls the trade-off between fewer but larger supervoxels and undesirable under-segmentation already at the supervoxel level. Figure 2.10 shows the watershed algorithm applied to EM data.

Discussion: Modelling Choices and Supervoxels

Above, we have illustrated two different choices for defining supervoxels as a starting point for neurite segmentation pipelines:

(i) *region-based:* supervoxels encompass either only membrane voxels, only intra-cellular voxels or only extra-cellular voxels (Figures 2.9 and 2.11b),

(ii) *boundary-based:* supervoxels touch on the centerline of a membrane (Figures 2.10 and 2.11a).



(a) raw data

(b) elevation map

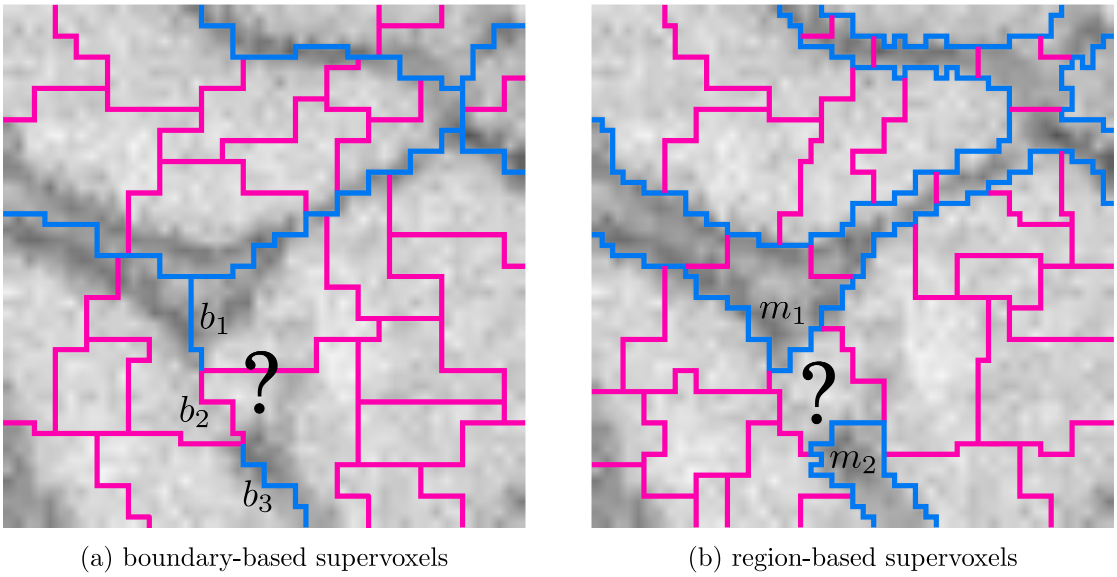
(c) seeds (original elevation map)

(d) watershed segmentation

(e) seeds (smoothed elevation map)

(f) watershed segmentation

Figure 2.10: Given the raw data (a), an elevation map is obtained by computing the largest eigenvalue of the 3D Hessian matrix (b). When choosing the local minima of this elevation map as seeds (c), the resulting watershed supervoxels are numerous and tiny (d). By choosing only local minima of the *smoothed* elevation map as seeds (e), the number of supervoxels is reduced and each defines a more useful atomic region (f).



(a) boundary-based supervoxels

(b) region-based supervoxels

Figure 2.11: How to recognize "holes" in the current membrane detection depends on the choice of supervoxels. In (a), blue lines indicate the belief that the adjacent supervoxels are separated by a membrane, whereas red lines indicate the opposite. In (b), regions believed to be a membrane are outlined in blue, while all other boundaries are marked in red. In both panels, the "?" indicates that there might be a "hole" in the membrane.

First, choice (i) seems best suited for high-resolution data in which membranes are at least a few pixels thick. At places where membrane appearance is very faint however, it can be difficult to obtain supervoxels adhering to the extent of the membrane. In contrast, choice (ii) usually finds a membrane boundary even if image evidence is faint.

Second, choice (i) does not differentiate between 3-dimensional entities (intra- and extra-cellular space) and 2-dimensional entities (thin membranes). In contrast, choice (i) models the interior of cells as 3-dimensional regions composed of supervoxels, but models membranes as 2-dimensional surfaces between regions.

In order to close "holes" in the detected membranes (staining imperfections, noise, faulty classification etc.), it will be necessary to connect the detected membrane fragments. For boundary-based supervoxels, holes occur where boundaries classified as membrane continue as boundaries classified as non-membrane (Figure 2.11a: b_1 and b_3 are classified as membrane, while b_2 is classified as non-membrane). However, for region-based supervoxels, detecting holes is more difficult. Should membrane regions m_1 and m_2 be connected in Figure 2.11b? This depends on whether we interpret m_1, m_2 merely as "bumps" of the membrane supervoxel or rather as an indication of boundary fragments.

In Chapters 4, 5, 6 and 7, we describe algorithms which all build upon boundary-based supervoxels in order to deal with the problem of holes in membranes.

2.5 Performance Evaluation

The performance of any algorithm should be evaluated with the application in mind. For the problem of segmenting electron microscopy volume images for connectomics, the aim is to trace neural wiring over long distances without error. Helmstaedter et al. have introduced the RESCOP measure, which compares the proposed skeleton tracing of a neurite with multiple tracings of the same neurite, each done by a different person [62]. However, as a high-level error measure it requires working on very large datasets and acquiring large amounts of consensus tracings. As current best automatic segmentation methods still make mistakes on a more local level, we instead turn to error measures popular in the image segmentation community.

2.5.1 Rand Index

The *Rand Index*, published in 1971 by William Rand [124], is a measure of the similarity of two different clusterings of a given dataset. The data to cluster is $D = (d_1, \ldots, d_N)$. A *partition* or *clustering* S of D into K clusters is a set of nonempty sets:

$$\begin{aligned} S = \{s_1, \ldots, s_K\} \text{ with}& \\ &\forall i : s_i \subseteq D \\ &\forall i \neq j : s_i \cap s_j = \emptyset \\ &\bigcup_{i=1}^{K} s_i = D \ . \end{aligned} \tag{2.11}$$

Given two data elements $d_i, d_j \in D$, the function $\delta(S_i, S_j)$ indicates whether they are within the same cluster in S:

$$\delta(S_i, S_j) = \begin{cases} 1 & \text{if } \exists k : d_i \in s_k \wedge d_j \in s_k \\ 0 & \text{else} \end{cases} \ . \tag{2.12}$$

$C(S, d)$ returns the index of the cluster in S that contains the element d:

$$C(S, d) = i \,|\, d \in s_i \ . \tag{2.13}$$

For two given partitionings S^1 and S^2, the Rand Index RI is a measure of their agreements

$$\mathrm{RE} = \binom{N}{2}^{-1} \cdot \sum_{i<j} \left| \delta(s_i^1, s_j^1) - \delta(s_i^2, s_j^2) \right| \ , \tag{2.14}$$

$$\mathrm{RI} = 1 - \mathrm{RE} \ . \tag{2.15}$$

The Rand Error (RE) measures the *disagreement* between S^1 and S^2 instead.

		Contribution to RI	
$\delta(s^1_i, s^1_j)$	$\delta(s^2_i, s^2_j)$	summand	comment
0	1	0	S^1, S^2 *dis*agree on whether to cluster d_i
1	0	0	and d_j together or not
0	0	1	S^1, S^2 *agree* that both elements d_i, d_j are not clustered together
1	1	1	S^1, S^2 *agree* that both elements d_i, d_j are clustered together

Table 2.2: Contributions ("summand") to the Rand Index (2.15) for a single pair of elements d_i and d_j.

In (2.15), the sum runs over all *pairs* of elements. If both are contained within the same cluster in one clustering and contained in different clusters in the other clustering, this will contribute the count "1" to the sum. For all other cases (both elements of the pair are contained within the same cluster for the one clustering as well as the other; both elements of the pair are contained within different clusters for the one clustering as well as the other), the contribution is zero (also compare Table 2.2). Finally, dividing by the number of possible pairs normalizes RI and RE to the range $[0, 1]$.

2.5.2 Variation of Information

Recently a distance measure proposed by Marina Meilă [110] – the *Variation of Information* (VI) – has become increasingly popular for comparing segmentations. VI uses information theory to relate how much information there is in each of the two clusterings with the amount of information one clustering has about the other. The probability $P(k) : [1, \ldots, K] \to [0, 1]$ of picking any datum in cluster s_k (assuming each $d_1, \ldots, d_N$ has equal probability of being picked) is

$$P(k) = \frac{|s_k|}{N} \quad . \tag{2.16}$$

The *entropy* of this random variable (and the clustering S) is

$$H(S) = -\sum_{k-1}^{K} P(k) \log P(k) \quad . \tag{2.17}$$

Furthermore, the probability $P(k, k') : [1, \ldots, K] \times [1, \ldots, K'] \to [0, 1]$ of picking a datum belonging to cluster s_k in S^1 and $s_{k'}$ in S^2 is

$$P(k, k') = \frac{|s_k \bigcap s_{k'}|}{N} \quad . \tag{2.18}$$

The *mutual information* of the clusterings S^1 and S^2 measures how much the entropy of one clustering is reduced if the second clustering is known:

$$I(S^1, S^2) = \sum_{k=1}^{K} \sum_{k=1}^{K'} P(k, k') \log \frac{P(k, k')}{P(k)P(k)} \quad . \tag{2.19}$$

The *Variation of Information* is then defined as

$$\begin{aligned} \mathrm{VI}(S^1, S^2) &= \big(H(S^1) - I(S^1, S^2)\big) + \big(H(S^2) - I(S^1, S^2)\big) \\ &= H(S^1) + H(S^2) - 2I(S^1, S^2) \quad . \end{aligned} \tag{2.20}$$

VI satisfies the properties of a metric.

2.5.3 Computing RE and VI

Both Rand Error and Variation of Information belong to the class of clustering measures that can be computed efficiently from the *contingency table*:

$S^1 \backslash S^2$	s_1^2	s_2^2	$\dots$	s_s^2	Sums
s_1^1	n_{11}	n_{12}	$\dots$	n_{1s}	a_1
s_2^1	n_{21}	n_{22}	$\dots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
s_r^1	n_{r1}	n_{r2}	$\dots$	n_{rs}	a_r
Sums	b_1	b_2	$\dots$	b_s	N

For a gold standard segmentation S^1 and proposed segmentation S^2 of a volume with N voxels, the contingency table contains the overlap count of segment s_i^1 and segment s_j^2, $n_{ij} = |s_i^1 \cap s_j^2|$ (such that $N = \sum_{ij} n_{ij}$), and the sizes of segments s_i^1 and s_j^2, $a_i = \sum_j n_{ij}$, and $b_j = \sum_i n_{ij}$.

Variation of Information. With $p_i = a_i/N$, $q_j = b_j/N$ and $p_{ij} = n_{ij}/N$, the Variation of Information (2.20) between S^1 and S^2 can be rewritten as expressions involving the contingency table only:

$$H_0 = -\sum_i p_i \log p_i, \quad H_1 = -\sum_j q_j \log q_j \quad , \tag{2.21a}$$

$$I = \sum_{i,j} p_{ij} \cdot \log \frac{p_{ij}}{p_i \cdot q_j} \quad , \tag{2.21b}$$

$$\mathrm{VI} = H_0 + H_1 - 2I \quad . \tag{2.21c}$$

Rand Error. The Rand Error is based on the counts of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) of *pixel pair labelings* in S^1 and S^2. If two pixels x, y lie within the same segment, they are connected ($x \leftrightarrow y$), otherwise disconnected ($x \not\leftrightarrow y$).

- TP: $x \leftrightarrow y$ in S^1 and S^2,
- TN: $x \not\leftrightarrow y$ in S^1 and S^2,
- FP: $x \not\leftrightarrow y$ in S^1 and $x \leftrightarrow y$ in S^2,
- FN: $x \leftrightarrow y$ in S^1 and $x \not\leftrightarrow y$ in S^2.

These counts are easily computed from the contingency table:

$$\mathrm{FP} = \sum_j \binom{b_j}{2} - \mathrm{TP}, \qquad \mathrm{FN} = \sum_i \binom{a_i}{2} - \mathrm{TP}, \tag{2.22a}$$

$$\mathrm{TP} = \sum_{i,j} \binom{n_{ij}}{2}, \qquad \mathrm{TN} = \binom{N}{2} - \mathrm{TP} - \mathrm{FP} - \mathrm{FN}. \tag{2.22b}$$

With these counts, the *Rand Error* (2.15) can be rewritten in terms of the contingency table

$$\mathrm{RE} = 1 - \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \ . \tag{2.23}$$

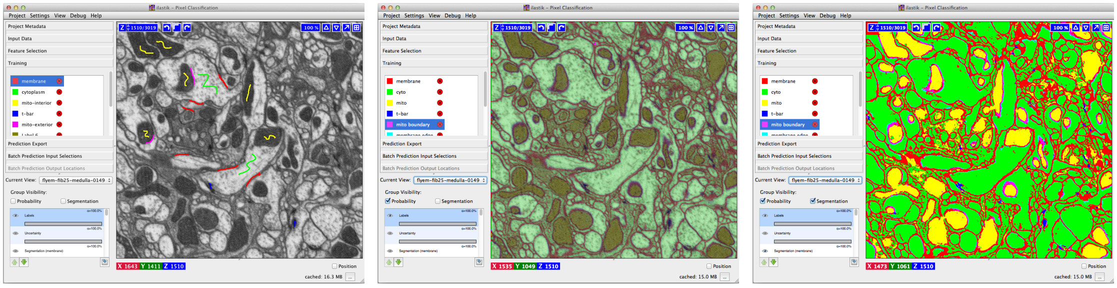
Chapter 3

Software

The research presented in this thesis necessitated writing or extending various software tools and libraries. Section 3.1 introduces the open source ILASTIK software, a joint effort of multiple researchers. I have contributed to various aspects of the software, focusing mainly on the volume viewer component (VOLUMINA, Section 3.2) and the user interaction in ilastik's semi-automated segmentation module (CARVING, Section 3.3). Not only has ilastik enabled me to run voxel classification training and prediction on large volume images (a first step in the proposed neurite segmentation pipeline), but the development of volumina has provided me with a stand-alone viewer component which I have used frequently during my research. For example, volumina could be easily extended for *interactive boundary learning* (Section 3.7) central to our segmentation approach in Chapter 4.

Section 3.4 briefly reviews region adjacency graphs, topological grids and cell complexes, in order to describe my solution to query a *hierarchy of cell complexes* in Section 3.5. This software was needed to implement the approximate hierarchical and blockwise multicut segmentation scheme in Chapter 5. The relation between cell complexes and region adjacency graphs is explored in Section 3.6.

Finally, Section 3.8 introduces the BLOCKEDARRAY library, which provides a chunked, in-memory compressed array class and implements some blockwise operations (such as connected component labeling) on large volume datasets. The library was used in my joint work with Boray Tek on soma detection [148].



(a) (b) (c)

Figure 3.1: Usage of ILASTIK by our collaborators, the FlyEM team at Janelia Farm. (a) Using a brush tool, various structures visible in the EM image – such as membranes, mitochondria, synapses and cytoplasm – have been sparsely annotated at the pixel level. Each class is associated with a color. (b) ILASTIK predicts the likelihood of assigning each of the chosen classes to each pixel in the image. Here, color opacity indicates these probabilities. (c) A classification of all pixels is obtained by choosing the most likely class label for each pixel. This *interactive* workflow is implemented both for 2D images as well as larger-than-RAM volume images (of which these screenshots show only a subset).

3.1 ilastik

ILASTIK is a set of open-source frameworks and graphical user applications for interactive machine learning on multi-dimensional images. The project was initiated by Christoph Sommer and Fred Hamprecht, and a first version was published in 2011 [140].

The software package is best known for its *pixel classification* module. Here, the user first defines a set of classes he wishes to distinguish at the voxel level (such as "membrane" versus "non-membrane" for binary classification, see Section 2.4). After choosing a set of image features likely to be relevant for the classification problem at hand, the user gives sparse training labels at the voxel level using a brush tool. A Random Forest classifier [30] is trained and the current prediction on the entire dataset is displayed to the user as soon as possible. This enables a feed-back loop: after examining the current classification, erroneous voxel predictions can be corrected by giving more training samples. By focusing on the difficult samples, this labeling strategy achieves good accuracy faster than indiscriminately labeling all pixels [84].

Pixel classification is a first step in the proposed neurite segmentation pipeline (Chapter 4; Figure 3.1). Unfortunately, the first version of ilastik (v0.5) was limited by the computer's main memory: raw data, all voxel features (single precision float) and classification results had to fit into RAM, severely limiting the applicability to connectomics-size datasets. Furthermore, the monolithic architecture made it hard to extend ilastik for use cases beyond pixel classification.

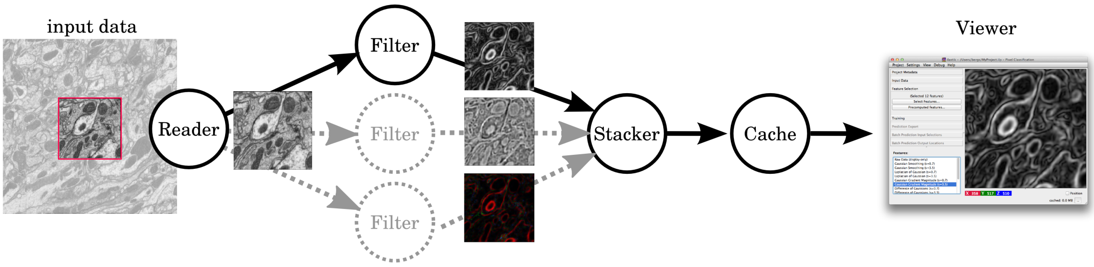


Figure 3.2: Data flow in ilastik's pixel classification workflow. Note that only the region marked with a red rectangle is read from the input data file in order to show the filter result in the viewer.

This motivated us to re-implement ilastik as a set of loosely coupled components based on a lazy computation paradigm (version 1.0, manuscript in preparation).

Computational Backend. The LAZYFLOW library implements a directed acyclic data flow graph (development led by Christoph Straehle). Nodes represent operations (an *operator*) on the data associated with the incoming edges. The result of the computation is sent along the outgoing edges to further operators downstream. Usually, operators do not have state; however, an important subclass are *caches* which serve previously computed results (as long as they are still valid). When requesting the result on only a small subset of the data, lazyflow takes care to only schedule the operations necessary to give the correct result, but no more. This ensures fast response time for interactive speed. Lazyflow also schedules tasks in parallel, if possible. Figure 3.2 illustrates the data flow in pixel classification schematically.

Viewer Frontend. The VOLUMINA library (development led by Bernhard Kausler and myself) is a slicing viewer for large volumetric datasets which supports pixel-level annotations. Its main strength is the tile-based display of orthogonal slice views. Data is only requested asynchronously when needed. Volumina can display a stack of image layers (blended with different opacities) and caches the resulting composite image tiles.

GUI library. Finally, the ILASTIK project (development led by Stuart Berg) ties together the lazyflow and volumina libraries. Graphical user interfaces for new image processing and machine learning tasks can be quickly created from existing building blocks. Each application consists of a sequence of steps (for example loading data, selecting features, labeling and prediction for the pixel classification workflow). Across all steps, the underlying lazyflow graph ensures that as little computation as possible is performed in order to satisfy a user request (such as viewing a new region of interest or an update to the set of labels).

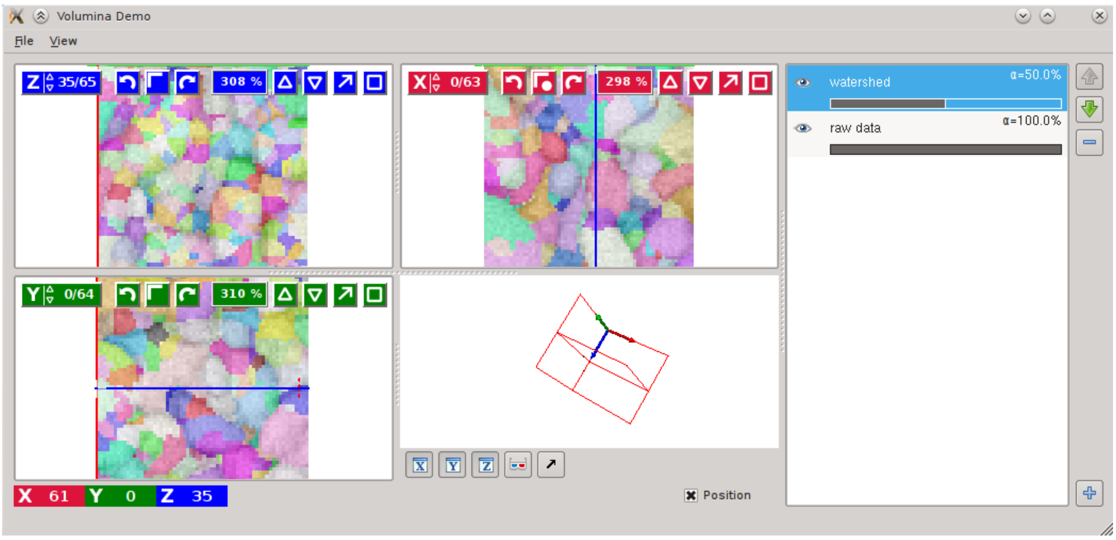


Figure 3.3: The VOLUMINA viewer shows a 3D dataset with a watershed segmentation overlay. The code to bring up this view is printed in Listing 3.1.

ilastik is mostly written in PYTHON, with the exception of some computationally expensive operations, which are implemented in C++. For array processing, the NUMPY and VIGRA[1] libraries are used on the Python and C++ sides, respectively. The user interface builds upon the QT toolkit (using the PYQT bindings). All source code can be found on `https://github.com/ilastik`.

3.2 volumina – a Volumetric Dataset Viewer and Editor

While Section 2.1 has given an overview of existing viewers for large datasets in connectomics research, none of them fit our requirements at the time:

- support for datasets larger than main memory
  (limit data requests to current viewport)
- can work with images (2D), volume images (3D) and time sequences of image volumes (4D), each possibly with multiple channels
  (ilastik is also used for 3D+t tracking, [81])
- orthogonal slice views and navigation
- allow the user to draw on top of any slice view with a pixel brush
  (for creating training labels in ilastik)
- interface with Python/Numpy for integration into ilastik

[1] `https://github.com/ukoethe/vigra`

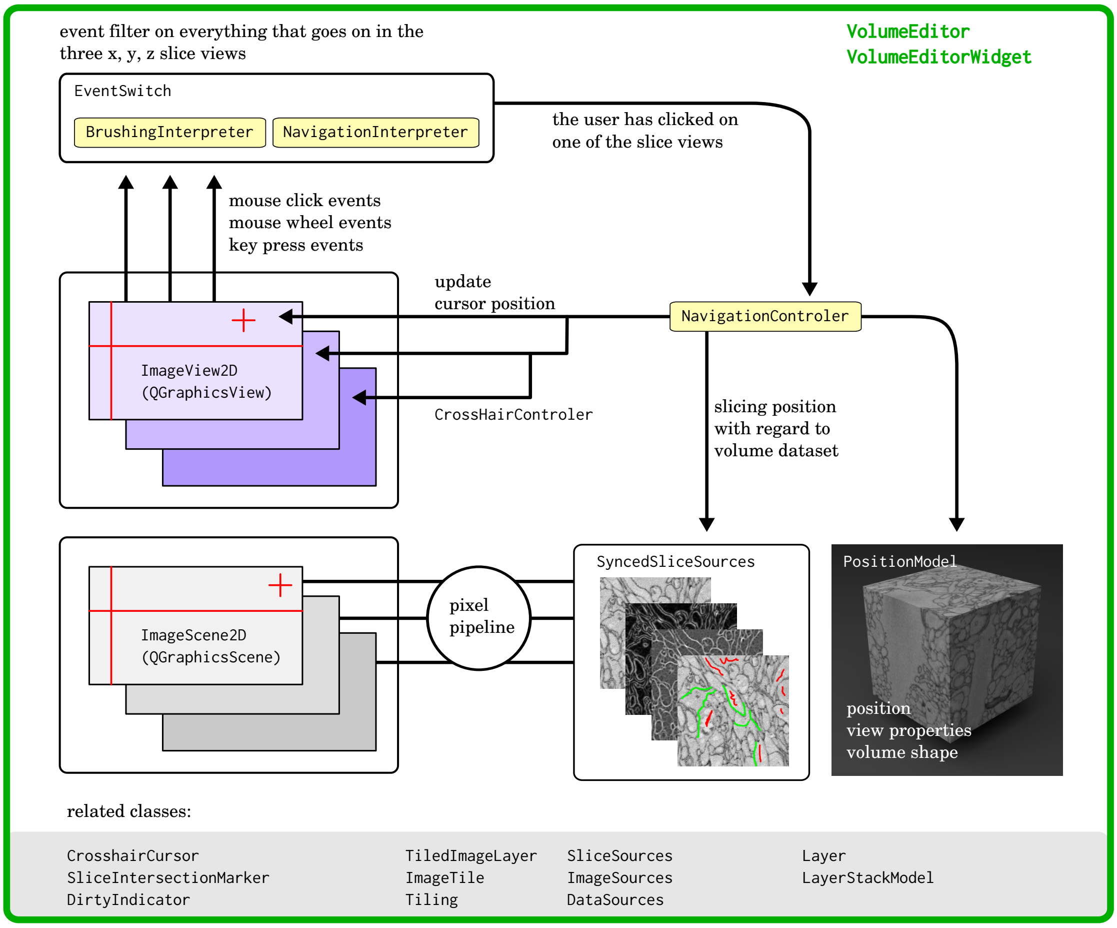


Figure 3.4: State updates and data flow after the user has initiated a change of position by clicking on one of the orthogonal slice views in volumina.

- ability to display multiple volumes as a stack of layers (similar to the GIMP or Photoshop software), and allow grayscale, indexed color table, and RGBA display (in order to overlay various results on top of the raw data)
- provide an asynchronous interface for data access (with responsive interfacing to the LAZYFLOW library)
- stand-alone component that is easily extensible.

To address these requirements, volumina is written in PYTHON and PYQT using the Model-View-Controller design pattern where possible in order to promote decoupling of components using QT's signals and slots mechanism. Figure 3.4 illustrates state updates and data flow after the user has initiated a change of position by clicking on one of the orthogonal slice views.

Briefly, each slice view is an instance of ImageView2D (a subclass of QGraphicsView) which visualizes the scene model in the associated ImageScene2D instance (a subclass of QGraphicsScene). In the scene, each layer is composed of non-overlapping fixed-size image tiles (class ImageTile, grouped into a Tiling). A limitation of this approach is that a zoomed-out view of large datasets will request too many image tiles at the original resolution — a pyramid scheme is not yet implemented.

The synchronization of views to show a certain (x, y, z)-position is mediated by the PositionModel, which is observed by the slice views. User interactions on the slice views (double clicking to jump to a position, mouse wheel scrolling or keyboard shortcuts) are interpreted as navigational commands using the NavigationControler and then forwarded to the position model.

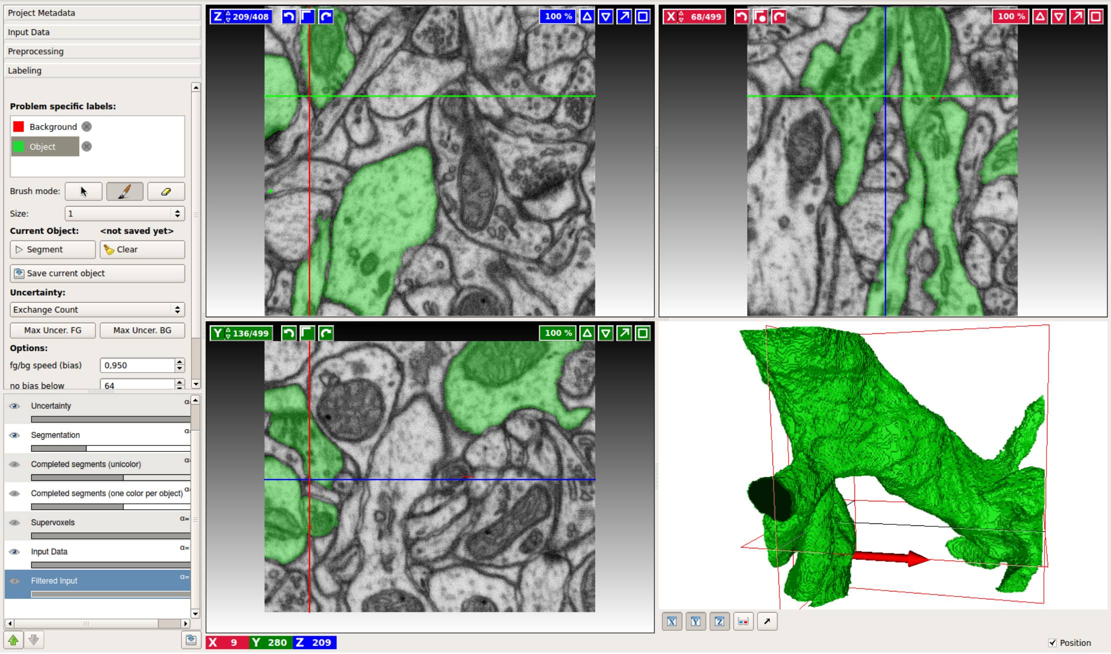
On jumping to a new region of interest, each viewport checks if the needed image tiles are still valid and in the cache of composited tiles. If yes, they are blitted on the screen. If no, an asynchronous update operation is triggered. For each visible layer, the raw data is first requested and then transformed into an image using the *pixelpipeline* submodule. Finally, all layers' tiles are composited together and made available for blitting.

The LayerStack stores an ordered list of Layer objects, which have the *visible* and *opacity* properties needed to compute the alpha-blended composite image of the layer stack. Different representations of the data (grayscale, indexed color table, RGBA) are chosen by using different Layer subclasses such as GrayscaleLayer, ColortableLayer and RGBALayer.

More details on the architecture can be found in Bernhard Kausler's thesis [81]. The code is open source and available at https://github.com/ilastik/volumina. Listing 3.1 demonstrates the use of volumina's high-level API in order to bring up the window shown in Figure 3.3.

```python
import sys, h5py; from numpy import float32, uint8
from vigra.filters import hessianOfGaussianEigenvalues,gaussianSmoothing
from vigra.analysis import watersheds
from vigra.analysis import labelVolumeWithBackground,extendedLocalMinima3D
from PyQt4.QtCore import QTimer; from PyQt4.QtGui import QApplication
app = QApplication(sys.argv)

from volumina.api import Viewer
v = Viewer()
v.title = "Volumina Demo"
v.showMaximized()

f = h5py.File("data.h5"); data = f["sbfsem"].value; f.close()

v.addGrayscaleLayer(data, name="raw data")

ev    = hessianOfGaussianEigenvalues(data.astype(float32), 1.0)[:,:,:,0]
ev    = gaussianSmoothing(ev, 1.0)
seeds = labelVolumeWithBackground(extendedLocalMinima3D(ev).astype(uint8))
seg   = watersheds(ev, seeds=seeds)[0]

wsL         = v.addRandomColorsLayer(seg, name="watershed")
wsL.visible = True
wsL.opacity = 0.5

v.editor.posModel.slicingPos = (10,20,30)

def cycleZ():
    Z = v.editor.posModel.volumeExtent(2)
    z = v.editor.posModel.slicingPos[2]
    if z<Z-1:
        return z+1
    else:
        return 0

t = QTimer()
t.setInterval(200)
def jump():
    v.editor.posModel.slicingPos = (0,0,cycleZ())
t.timeout.connect(jump)
t.start()
app.exec_()
```

Listing 3.1: Example usage of VOLUMINA's high-level API. A 3D data volume is displayed as a grayscale layer. On this data, a seeded watershed segmentation is computed and visualized as a layer with 50% opacity overlaying the raw data. Each segment is assigned a random color. Finally, the position model is leveraged to program a continuous cycling through the z-dimension.



Size: 1

Current Object: <not saved yet>

Exchange Count

fg/bg speed (bias) 0.950

no bias below 64

Figure 3.5: Screenshot of the *Carving workflow* in ILASTIK version 1.0. Using sparse foreground/background annotations, the user can quickly extract volumetric objects from 3D datasets. After refining the segmentation by adding more brush strokes, the result is saved to disk.

3.3 Creating Gold Standard Segmentations with Carving

Christoph Straehle's *Carving* algorithm [144, 143] has proven to be a useful tool for creating dense gold standard segmentations to be used either as training or validation volumes (for Carving also see Section 2.2).

For the work presented in Chapter 4, I have modified the Carving in ilastik v0.5 so as to enable a more streamlined workflow for *dense* volume annotation. The original interface (as implemented by C. Straehle) allowed the user to segment multiple objects by using differently colored foreground labels and one background label. However, while segmenting a new object the user might be biased by the previously segmented objects in his placement of labels. Furthermore, the Carving algorithm depends on *all current* labels, making the segmentation of a single neurite depend on the entire segmentation history.

In order to circumvent these problems, we propose to segment neurites *independently* as a binary segmentation problem. The user only has to consult the previous segmentation when choosing a new, yet unlabeled neurite. The implementation allows to save and load the individual segmentations (by name or by context menu on any slice view), making it easy to correct mistakes if found later. Labeling errors can often be found by inspecting the overlap of independently segmented neurites.

After a first implementation in ilastik v0.5, Christoph Straehle, Stuart Berg and myself have reimplemented this approach in ilastik v1.0. Figure 3.5 shows a screenshot of the new version.

3.4 Representing Topology and Geometry of a Supervoxel Volume

In order to describe my solution for computing boundary features (Section 3.4.3) and for hierarchical and blockwise supervoxel adjacency graph construction (Section 3.5 and Chapter 5), I have to introduce some key concepts first.

Consider a supervoxel segmentation algorithm that produces a label volume as output. Formally, the *label volume* σ maps from the finite voxel grid $G = \{0, \dots, n_1 - 1\} \times \{0, \dots, n_2 - 1\} \times \{0, \dots, n_3 - 1\}$ to the n_{SV} supervoxels or *segments*:

$$\begin{aligned} &\sigma : G \to \{1, \dots, n_{\mathrm{SV}}\} \quad \text{such that} \\ &\qquad \forall v_1, v_2 \in G : \quad v_1 \leftrightarrow v_2 \Leftrightarrow \sigma(v_1) = \sigma(v_2), \end{aligned} \tag{3.1}$$

where "$\leftrightarrow$" denotes path connectivity under the 6-neighborhood.

Implicit versus explicit representations. Obviously, the label volume σ describes the *geometry* of each segment. A more explicit representation for the segment labeled l is its constituent set of coordinates $CC_l = \{v \in G \mid \sigma(v) = l\}$. Furthermore, consider two adjacent segments with labels l_1 and l_2:

$$\begin{aligned} &l_1, l_2 \in \{1, \dots, n_{\mathrm{SV}}\} \textit{ adjacent} \\ &\qquad \Leftrightarrow \exists\ (v_1, v_2) \in G : \sigma(v_1) = l_1 \wedge \sigma(v_2) = l_2 \wedge (v_1, v_2) \text{ are 6-neighbors.} \end{aligned} \tag{3.2}$$

The list of adjacent segments is only implicitly encoded in σ. Another interesting entity are the shared surfaces between segments l_1 and l_2. To represent these explicitly, one needs to find all adjacent pairs of pixels which are labeled l_1 and l_2, respectively.

Often, it is worthwhile to construct *explicit* representations of the above entities (geometry) and their relationships (topology) amongst each other. This way, expensive search operations through the volume are only executed once (at the expense of storage space). In addition, the explicit representation of these entities makes them *addressable* with dense indices. For example, this makes attaching a feature value to a shared surface easy. Using the relationship of entities, questions such as "which two segments have the given shared surface in common?" can be easily answered.

There exist various ways to represent the topology and geometry inherent in σ more explicitly. Figure 3.6 illustrates two choices with an example label image.

3.4.1 Region Adjacency Graph

A common choice is the *region adjacency graph* RAG $= (\mathcal{V}, \mathcal{E})$ where the segments form the set of nodes $\mathcal{V}$ and an edge $e \in \mathcal{E}$ is created between each pair of adjacent segments. Formally,

$$\begin{aligned} \mathcal{V} &= \{1, \dots, n_{\mathrm{SV}}\} \ , \\ \mathcal{E} &= \{e = (\sigma(v_1), \sigma(v_2)) \in V \times V \mid \sigma(v_1) \neq \sigma(v_2) \wedge \|v_1 - v_2\| = 1\} \ . \end{aligned} \tag{3.3}$$

For the optimization algorithm in Chapter 4, the simple RAG data structure is sufficient. However, for the enumeration of segmentations (Chapter 5) or for boundary feature extraction (Section 3.4.3) we have found it convenient to use a *Cell Complex representation* of σ. It can be easily reduced to a RAG when desired (Section 3.6).

3.4.2 Cell Complex Representation

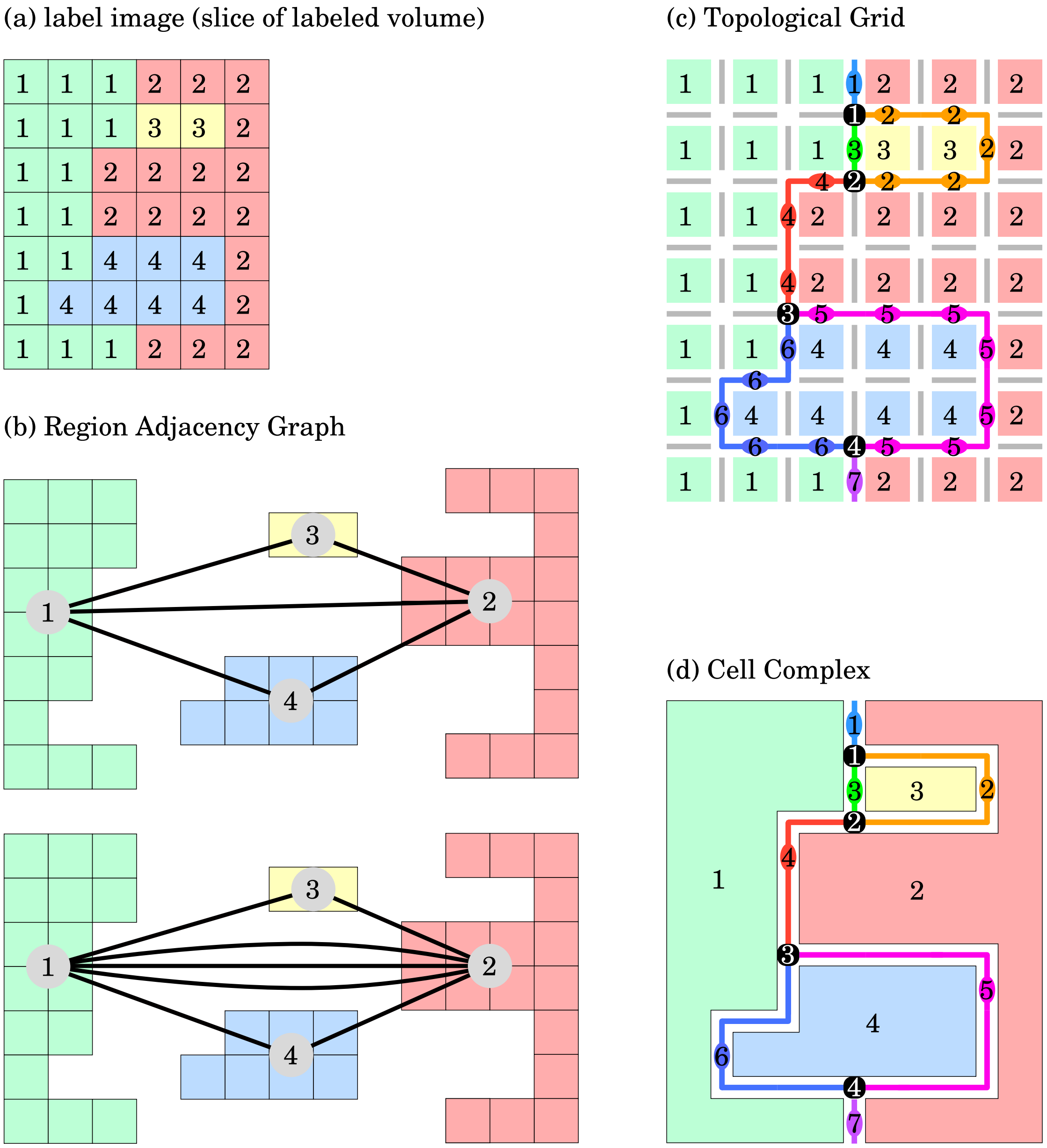
The boundaries between supervoxels are only represented *implicitly* in the label image σ as inter-voxel surfaces where two adjacent supervoxels meet. The *topological grid* T [96] does not only represent *voxels*, the unit cubes or *3-cells* of the volume image. It also represents the following inter-voxel entities (Figure 3.7a-c):

- *surfels* (*2-cells*), the unit squares in between two voxels,
- *edgels* (*1-cells*), the unit lines in between two surfels,
- *pointels* (*0-cells*), the points between two edgels.

Let $T = \{0, \dots, 2n_1 - 2\} \times \{0, \dots, 2n_2 - 2\} \times \{0, \dots, 2n_3 - 2\}$. This enlarged grid allows to represent surfels, edgels and pointels *explicitly*:

- voxel $v \in G$ corresponds to position $2 \cdot v \in T$ which is a coordinate with all-even entries,
- the surfels of voxel v are found at positions $\{v + \delta \mid \delta \in \{(1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), (0,0,-1)\}\ \}$, which are coordinates with two even entries and one odd entry,
- the linels of voxel v are found at positions $\{v + \delta \mid \delta \in \{(1,1,0), (1,0,1), (0,1,1), (-1,-1,0), (-1,0,-1), (0,-1,-1),$ $(1,-1,0), (1,0,-1), (0,1,-1), (-1,1,0), (-1,0,1), (0,-1,1)\},$ which are coordinates with one even entry and two odd entries,
- the pointels of voxel v are found at positions $\{v + \delta \mid \delta \in \{(1,1,1), (-1,1,1), (1,-1,1), (1,1,-1), (-1,-1,1),$ $(1,-1,-1), (-1,1,-1), (-1,-1,-1)\}\ \},$ which are coordinates with all-odd entries.

Next, we review some definitions [6, 96].



(a) label image (slice of labeled volume)

1	1	1	2	2	2
1	1	1	3	3	2
1	1	2	2	2	2
1	1	2	2	2	2
1	1	4	4	4	2
1	4	4	4	4	2
1	1	1	2	2	2

(b) Region Adjacency Graph

(c) Topological Grid

(d) Cell Complex

Figure 3.6: Different ways to represent the topology and geometry inherent in a label image. (a) the original label image. (b) a RAG representation as a graph (above) and multi-graph (below). Alternatively, the relation among segments in (a) may be represented by a cell complex (c,d). For details see text.

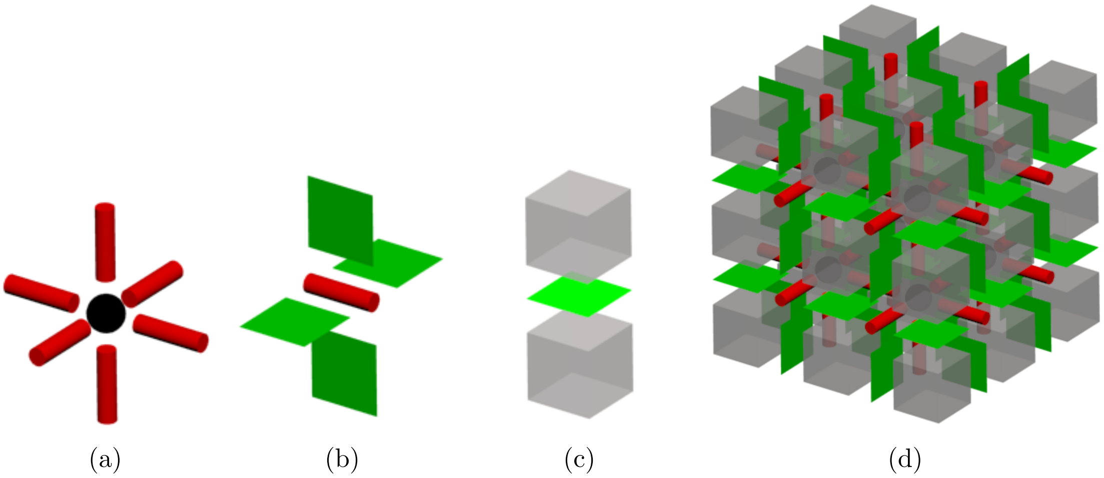


Figure 3.7: The *topological grid* representation of label volume. (a) The Γ-neighborhood of 0-cells, (b) of 1-cells and (c) of 2-cells. Each 0-cell bounds six 1-cells, each 1-cell bounds four 2-cells , and each 2-cell bounds two 3-cells. Panel (d) shows the topological grid T for a volume image of 3 × 3 × 3 voxels.

Definition (Γ-neighborhood)
The Γ-neighborhood *of a j-cell $t \in T$ (with $0 \leq j < 3$) is a mapping $\Gamma : T \rightarrow \mathcal{P}(T)$, such that $\Gamma(t)$ consists of all 6-neighbors of t that are $(j+1)$-cells.*

Definition (bounded by, bounded, adjacent)
*The Γ-neighborhood describes the relation of j-cells: A 0-cell bounds six 1-cells, which in turn each bound four 2-cells, which each bound two 3-cells (Figure 3.7). The "*bounded*" relation is a partial order relating j-cells and $(j+1) - cells$. By transitivity, it induces an order amongst j-cells of any dimensionality. The inverse relation, "*bounded by*", states that a 3-cell is bounded by six 2-cells. Each 2-cell is bounded by four 1-cells, and each 1-cell is bounded by two 0-cells. Finally, a j-cell t_1 with $j \geq 1$ is* adjacent *to j-cell t_2 iff there exists a $(j-1)$-cell $t \in T$ such that both t_1 and t_2 are Γ-neighbors of t.*

In order to define more complex entities than the atomic j-cells, we can define a labeling on the topological grid T:

$$\tau : T \rightarrow \mathbb{N}_0 \quad . \tag{3.4}$$

Two adjacent j-cells are called *connected* if they are both Γ-neighbors of a $(j-1)$-cell t for which $\tau(t) = 0$. A cell t is called *inactive* if $\tau(t) = 0$, otherwise it is called *active*.

Using this connectivity, one can obtain a labeling τ from σ in which the surfaces between adjacent segments, the curves in which these surfaces meet and the points where such curves meet are represented as *connected components*. This is depicted in

Figure 3.6c, in which we keep the notation for 3D volumes in order to avoid confusion. Any 3-cell is always active. The label for each voxel $v \in G$ is simply transferred as $\tau(2 \cdot v) := \sigma(v)$. However, 1-cells and 2-cells are either labeled as inactive by setting $\tau(v) = 0$, or as active by setting $\tau(v) \in \mathbb{N}$. In the figure, inactive 2-cells are marked in gray, and inactive 1-cells are invisible. The active 2-cells form a *maximal set of connected surfels* (a 2D surface in 3D space) such that each surfel lies between voxels belonging to the same pair of supervoxels. Similarly, 1-cells are active when three or more such faces meet. Active 1-cells form a *maximal set of connected linels* (1D curve in 3D space) such that each linel lies between surfels belonging to the same pair of surfaces. The relation of points, curves, faces and segments is described with a *cellular complex*:

Definition (cellular complex, [96])
A cellular complex $\mathcal{C} = (C, \prec, dim)$ *consists of a set of elements* $C = \bigcup C_i$ *which are related amongst each other via the* bounding relation *"$\prec$". The binary relation $\prec$ is antisymmetric, irreflexive and transitive. The function* $dim : E \to \mathbb{N}^0$ *assigns each element a dimensionality such that* $dim(e) < dim(e')$ *for all* (e, e') *with* $e \prec e'$.

For a 3-dimensional cell complex, the set C consists of four distinct sets with dimension $\dim \in \{0, 1, 2, 3\}$:

$$C = C_0 \cap C_1 \cap C_2 \cap C_3, \tag{3.5}$$

where C_3 are the *segments*, C_2 the *faces* between segments, C_1 the *curves* between faces and C_0 the *points* between curves. Because each $CC \in C_j$ is a connected component of elementary elements of like dimension j (voxels, surfels, edgels, linels or pointels), CC is called a *j-component* (see Figure 3.8).

This is illustrated in Figure 3.6d: here, C consists of (i) 3-components or segments $C_3 = \{1, 2, 3, 4\}$ which appear as areas, (ii) 2-components or faces $C_2 = \{1, 2, 3, 4, 5, 6, 7\}$ which appear as lines, (iii) 1-components or curves $C_1 = \{1, 2, 3, 4\}$ which appear as points. There are no 0-components in this slice. The j-components $CC \in C_j$ are addressed using their dimension $j \in \{0, 1, 2, 3\}$ and an integer label $l \in \{1, \dots, |C_j|\}$.

The "bounded", "bounded by" and "adjacent" relations are defined for j-components by using the definition for j-cells. For example, a 2-component $CC \in C_2$ may bound exactly two segments. For all 2-cells $c \in CC$, the set of labels in the Γ-neighborhood, $\{\tau(t) | \tau(t) > 0 \wedge t \in \Gamma(c)\}$, is the same.

In Figure 3.6, the bounding relation $\prec$ states, for example, that (i) face 1 bounds segments $\{1, 2\}$, and (ii) curve 1 bounds faces $\{1, 2, 3\}$. In practice, this information is stored using matrices (in which "0" denotes inactive neighbors):

$$B_j : \{1, \dots, |C_j|\} \to \{0, \dots, |C_{j+1}|\}^{6-2j} \ . \tag{3.6}$$

The CGP software package [6] first constructs τ by blockwise connected component labeling, which scales to large data sets. Starting from τ, the bounding relation of all entities is constructed. Additionally, the constituent set of j-cells making up each

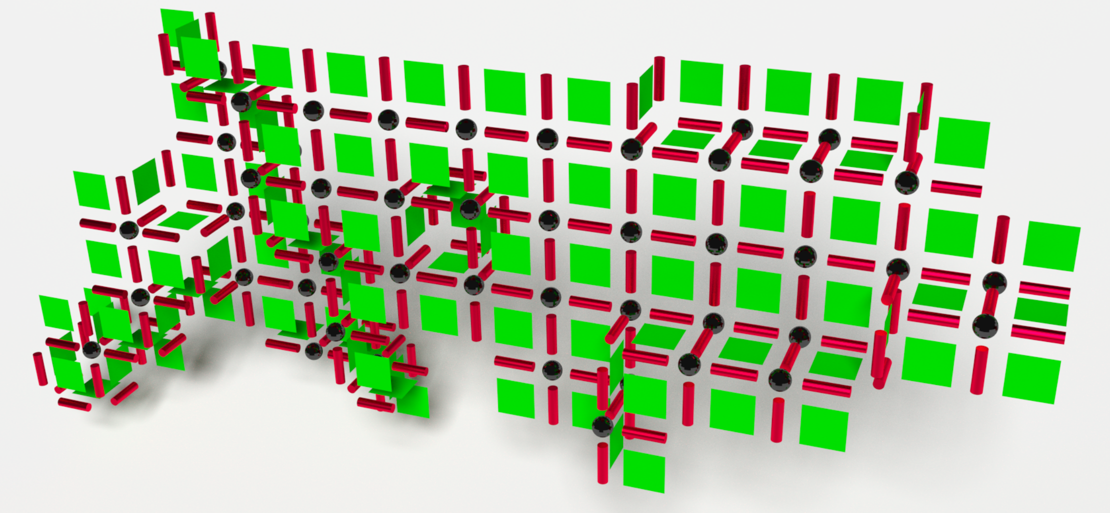


Figure 3.8: Example of a 2-component, a connected component of 2-cells. Internal, inactive 1-cells and 0-cells are shown in red and black.

j-component is written to disk. Together, this information defines the cell complex representation of σ.

3.4.3 Computing Boundary Features

In our application domain – the segmentation of neurites in EM volume images – we need to decide whether a 2-component (also called a *boundary* between two supervoxels) represents a true membrane or is only due to noise. In order to let a machine learning classifier decide, a feature vector $\mathbf{f}_{CC} \in \mathbb{R}^m$ describing j-component $CC \in C_2$ is needed. As features in $\mathbf{f}_{CC}$, we consider – amongst others – statistics of the set of voxel values accumulated along the boundary (cf. Section 4.8.2).

In order to collect these voxel values, the stored list of topological coordinates making up the 2-component is leveraged. Given face $CC \in C_2$, one first obtains the set of coordinates

$$V = \{\Gamma(c)/2 \mid c \in CC\} \ , \tag{3.7}$$

which are all voxel coordinates in G adjacent to any surfel making up CC. Next, we use these coordinates to index a voxel feature map $f : G \to \mathbb{R}$, such as the gradient magnitude computed on the raw data. The set of voxel features along the surface,

$$F = \{f(v) \mid v \in V\}, \tag{3.8}$$

is then used to compute various statistical features, such as the mean, standard deviation and quantiles.

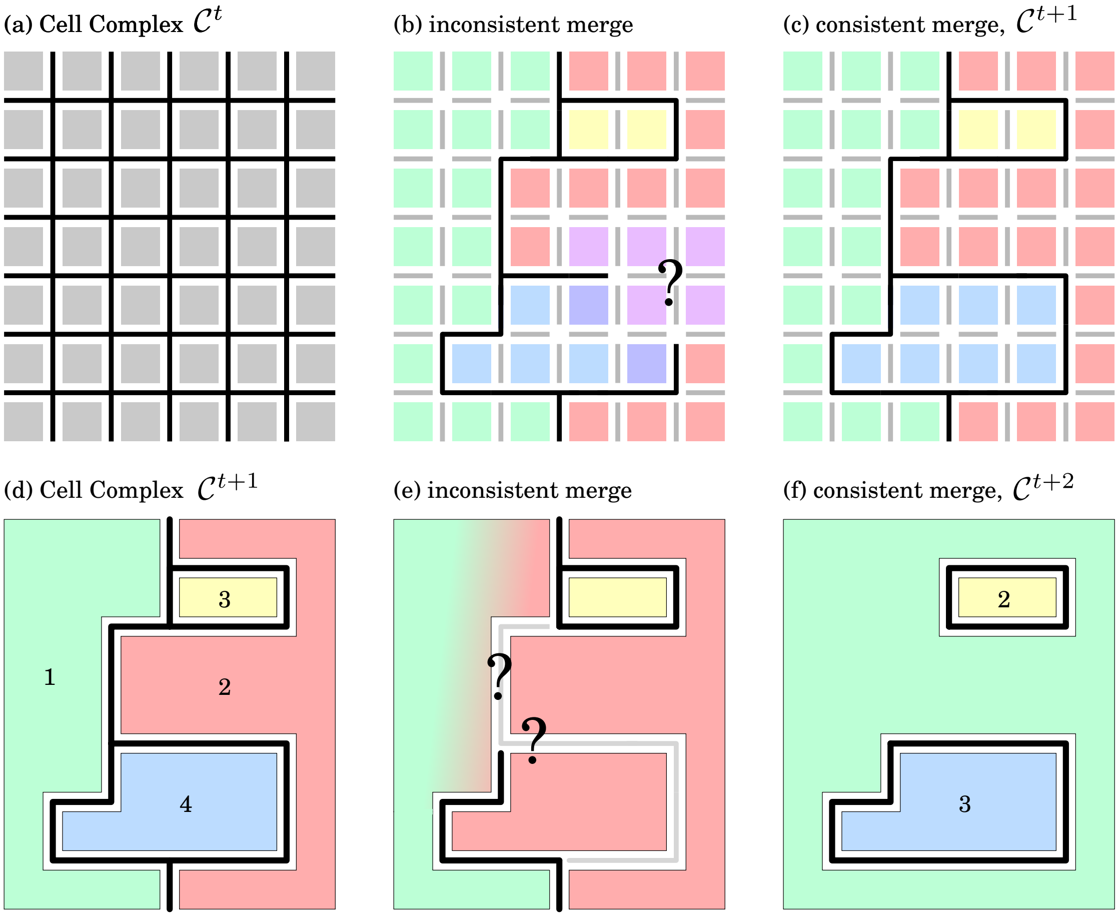

(a) Cell Complex $\mathcal{C}^t$
(b) inconsistent merge
?
(c) consistent merge, $\mathcal{C}^{t+1}$
(d) Cell Complex $\mathcal{C}^{t+1}$
1
2
3
4
(e) inconsistent merge
?
?
(f) consistent merge, $\mathcal{C}^{t+2}$
2
3

Figure 3.9: Given a cell complex $\mathcal{C}$, a binary labeling $\boldsymbol{y}$ for each face $CC \in C_2$ is *consistent* if there are no dangling faces.

3.5 Managing Cell Complex Hierarchies

Starting from a cell complex representation of some data's supervoxel representation (Section 3.4), we propose in this thesis to make a *simultaneous* decision whether to merge each two adjacent supervoxels or not. The result of this merge operation can be either described by a new region adjacency graph in which some edges have been contracted, or, alternatively, as a new cell complex in which some pairs of adjacent 1-components, 2-components or 3-components have been merged. This section describes how this new cell complex is constructed. The merge operation can be applied successively, creating a hierarchy of cell complexes.

3.5.1 Merge Operations in a Cell Complex

Let $\mathcal{C}^t$ be a given cell complex representation of any label volume. Furthermore, let a binary vector $\boldsymbol{y} \in \{0,1\}^{|C_2|}$ encode a simultaneous merge decision (zero means “merge” and 1 means “do not merge”) for each face $CC \subset \mathcal{C}_2^t$. Then, by applying the merge$(\cdot,\cdot)$ function,

$$\mathcal{C}^{t+1} = \text{merge}(\mathcal{C}^t, \boldsymbol{y}), \tag{3.9}$$

one obtains a new cell complex $\mathcal{C}^{t+1}$. We require that $\boldsymbol{y}$ is consistent, which is explained in Figure 3.9 and more formally in Chapter 4. For convenient notation, we refer to the segments, faces, lines and points in $\mathcal{C}^t$ in this context as j-cells, which are the constituent elements of j-components in the merged cell complex $\mathcal{C}^{t+1}$.

Constructing $\mathcal{C}^{t+1}$

Algorithm 3.1 describes the merge operation. Briefly, 2-cells that should be merged are marked inactive and 2-cells that should *not* be merged are marked active. The next step marks first 1-cells, then 0-cells active or inactive. A j-cell is marked as *active* only when at least three of its Γ-neighbors are marked active. An active 1-cell signals a junction of three or four surfaces, and an active 0-cell signals a junction of three or more curves. Next, the algorithm constructs the new cell complex by running a connected component labeling of 3-cells, 2-cells and 1-cells. Two j-cells are only connected if there exists an *active* $(j-1)$-cell which bounds them. By keeping track of the label mappings (relabeling function $r(\cdot)$ and its inverse, r^{-1}), it is then easy to construct the new boundings B_j^{t+1} from the old boundings B_j^t.

3.5.2 The `CellComplexMerger` Class

A `CellComplexMerger` instance `ccm1` represents a cell complex $\mathcal{C}^1$. For example, it implements functions `bounds`, `boundedBy`, and `adjacent` to query the bounding relation between j-cells. This interface replicates the interface of the *GeometryReader* class that I have written for the CGP package, as released in [6].

A new `CellComplexMerger` instance, `ccm2`, can then be constructed to represent the result $\mathcal{C}^2$ of a merge$(\mathcal{C}^1, m)$ call. In this case, `ccm2` stores a reference to the previous cell complex, `ccm1`.

The class offers multiple functions to translate between the j-cells in $\mathcal{C}^1$ to the j-components in $\mathcal{C}^2$ and vice versa. The *project forward* functions are called from `ccm2` and map information in terms of $\mathcal{C}^1$ into the domain of $\mathcal{C}^2$. Similarly, the *project back* functions are called from `ccm2` and map information in terms of $\mathcal{C}^2$ into the domain of $\mathcal{C}^1$. For example, given a set of j-components in $\mathcal{C}^2$, the `ccm2.projectLabelSetBack` function returns the union of the j-components in terms of the C_j^1 labels of j-cells. The

Algorithm 3.1: merge pairs of adjacent supervoxels and construct a new cell complex representation

```
Input  : Cell Complex $\mathcal{C}^t$, specified via
             the sets of $j$-cells $C_0^t, C_1^t, C_2^t, C_3^t$ and
             the boundings relations $B_0^{t+1}$, $B_1^{t+1}$, $B_2^{t+1}$
         merge decision $\boldsymbol{y} \in \{0,1\}^{|C_2^t|}$ for all faces of $\mathcal{C}^t$
Output: new cell complex $\mathcal{C}^{t+1}$
mark all $0, 1, 2$-cells in $\mathcal{C}^t$ as inactive and all 3-cells as active.
for $c \in C_2^t$ do
    mark $c$ active if $\boldsymbol{y}_c = 1$, else inactive
for $j \leftarrow 1$ to $0$ do
    for $c \in C_j^t$ do
        mark $c$ active if more than two $(j+1)$-components in the $\Gamma$-neighborhood
        are active
for $j \leftarrow 3$ to $0$ do
    $|C_j^{t+1}| \leftarrow 0$
    $r_j \leftarrow$ a mapping $C_j^t \rightarrow C_j^{t+1}$          // store relabeling from old to new
    for $c \in C_j^t$ do
        if c is inactive then
            $r_j(c) \leftarrow 0$                                   // relabel to special '0' value
        else if c does not yet belong to a connected component then
            $CC \leftarrow$ ConnectedComponent($j$,$c$)
            $|C_j^{t+1}| \leftarrow |C_j^{t+1}| + 1$                // unique label for CC
            for $\tilde{c} \in CC$ do
                $r_j(\tilde{c}) = |C_j^{t+1}|$                      // relabel from old to new
    Construct $r_j^{-1}$ // reverse relabeling from new to old
for $j \leftarrow 3$ to $0$ do
    for $CC \in C_j^{t+1}$ do
        take any $c \in r_j^{-1}(CC)$
        for $k \leftarrow 1$ to $6 - 2k$ do
            $B_j^{t+1}[CC, k] \leftarrow r_j(B^t[c, k])$            // relabel boundings
```

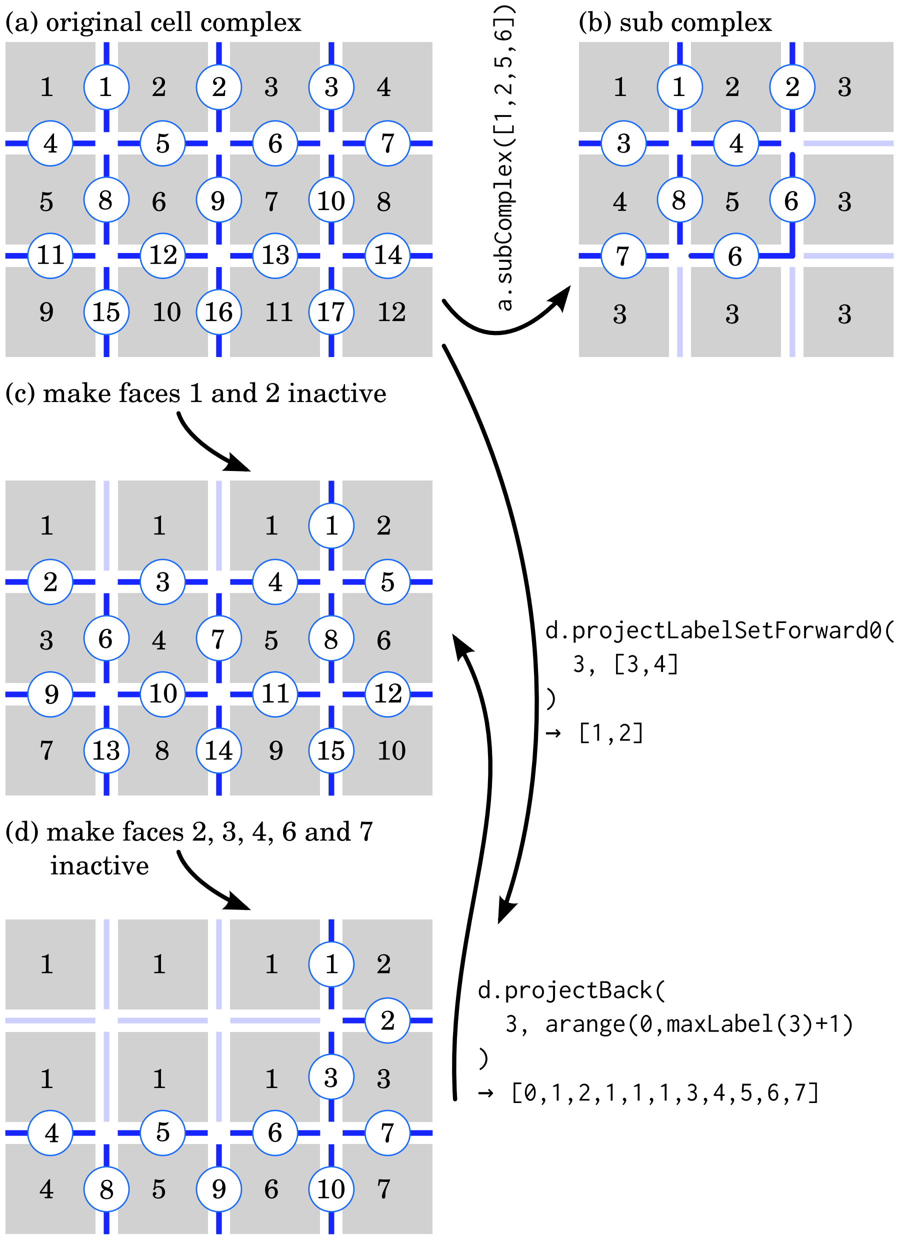
```
import numpy, sys; from numpy import *; U=uint32
from cgp import CellComplexMerger

#boundings to set-up panel (a)
bd0 = zeros((0,0), U) #no 0-sets for 2D example
bd1 = asarray([[1, 4, 5, 8], [2, 5, 6, 9], [ 3, 6, 7,10],
               [8,11,12,15], [9,12,13,16], [10,13,14,17]], U)
bd2 = asarray([[1,2], [2,3], [3, 4], [1, 5], [2, 6], [3, 7], [4, 8],
               [5, 6], [6,7],[7, 8], [5, 9], [6,10], [7,11], [8,12],
               [9,10], [10,11],[11,12]], U)

a = CellComplexMerger(bd0, bd1, bd2)
a.setObjectName("panel (a)")

b = a.subComplex(asarray([1,2,5,6], U), verbose=False)
b.setObjectName("panel (b)")

m = ones(bd2.shape[0], U)
m[1-1] = 0; m[2-1] = 0 #convert to 0-based indices
c = CellComplexMerger(a, m, verbose=False)
c.setObjectName("panel (c)")

m = ones(c.maxLabel(2), U)
m[2-1:5-1] = 0; m[6-1:8-1] = 0 #0-based indices!
d = CellComplexMerger(c, m, verbose=False)
d.setObjectName("panel (d)")

assert ( d.projectLabelSetForward0(3, asarray([3,4],U)) == [1,2] ).all()
assert ( d.projectBack(3, arange(0,d.maxLabel(3)+1,dtype=U))
           == [0,1,2,1,1,1,3,4,5,6,7] ).all()
```

Listing 3.2: Example usage of the `CellComplexMerger` class from Python. The code reproduces the operations illustrated in Figure 3.10.

`ccm2.projectLabelSetForward` functions takes a set of j-cells of $\mathcal{C}^1$ as input. It then finds those j-components in $\mathcal{C}^2$ which contain any of the j-cells.

A sequence of successive merge operations creates a linked list $\mathcal{C}^4 \leftarrow \mathcal{C}^3 \leftarrow \mathcal{C}^2 \leftarrow \mathcal{C}^1$ of `CellComplexMerger` instances. This list can be serialized to disk as an HDF5 file. On de-serializing any merger, all preceding mergers are also loaded.

The code is written in C++ and exported to Python via BOOST::PYTHON. Figure 3.10 and Listing 3.2 – together with Figure 3.10 – illustrate the usage of the `CellComplexMerger` class.



```
a.subComplex([1,2,5,6])
```

```
d.projectLabelSetForward0(
  3, [3,4]
)
→ [1,2]
```

```
d.projectBack(
  3, arange(0,maxLabel(3)+1)
)
→ [0,1,2,1,1,1,3,4,5,6,7]
```

Figure 3.10: Illustration of a few operations on a chain of `CellComplexMerger` objects. For details see text and Listing 3.2.

3.6 From the Cell Complex to the RAG Representation

Consider Figure 3.6d. Regions 1 and 2 share three distinct surfaces labeled 1, 4 and 7. This could be represented as a region adjacency *multi*graph (panel b, bottom). However, it is more convenient to construct a simple region adjacency graph (panel a, top). This is achieved with the following algorithm for cell complex $\mathcal{C}$:

- Create an empty map M from ordered pairs of labels to a set of labels.
- Each surface $c \in C_2$ bounds exactly two segments $(s_1, s_2) \in C_3$ with $s_1 < s_2$. Add c to the set $M[(s_1, s_2)]$.
- Construct RAG $= (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = C_3$ and $\mathcal{E} = \{(s_1, s_2) \in \mathcal{V} \times \mathcal{V} \mid M[(s_1, s_2)] \neq \emptyset\}$.

3.7 Interactive Classification of Supervoxel Boundaries

The application presented in this section allows to train a binary Random Forest classifier to distinguish true from false boundaries based on pre-computed boundary features $\mathbf{f}_i$. Here, we follow the successful ideas from the pixel classification module of ilastik, but extend the *interactive* labeling approach to boundaries between supervoxels. We base our application on the VOLUMINA component. It requires some pre-processing of the over-segmentation in order to quickly display the boundaries on each slice view in volumina.

Pre-processing the Over-Segmentation

Given an over-segmentation σ represented as a label volume (with dense labels starting from one), we use the CGP library [6] to obtain a topological grid representation τ in which the 3D segments, 2D faces between adjacent segments, 1D lines between adjacent faces and 0D points between adjacent lines are represented as connected components (see Section 3.4).

In a slice view through the volume image, the faces appear as inter-pixel paths between the cross-sections of two adjacent segments. However, a single face may appear as a set of disconnected paths (Figure 3.12b).

We pre-compute an *explicit* representation of the faces for each orthogonal slicing through the dataset. For each slice of σ (origin $\mathbf{p}$, normal $\mathbf{n}$), the corresponding slice τ_{sl} (origin $2 \cdot \mathbf{p}$, normal $\mathbf{n}$) from the topological grid τ is loaded. An entry $(i, j) \in \tau_{\text{sl}}$ with $\text{mod}(i, 2) + \text{mod}(j, 2) = 1$ represents an edgel. All such edgels which are part of a face (2-component) intersecting the slice are grouped according to the face label using a map data structure. All lines for a given face label $f \in C_2$ are then sorted to form a set of maximally connected paths representing f on the slice. These paths are serialized as illustrated in Figure 3.12. Finally, the serialized representations of all faces intersecting the slice are concatenated and stored using the HDF5 file format shown in Figure 3.12c.

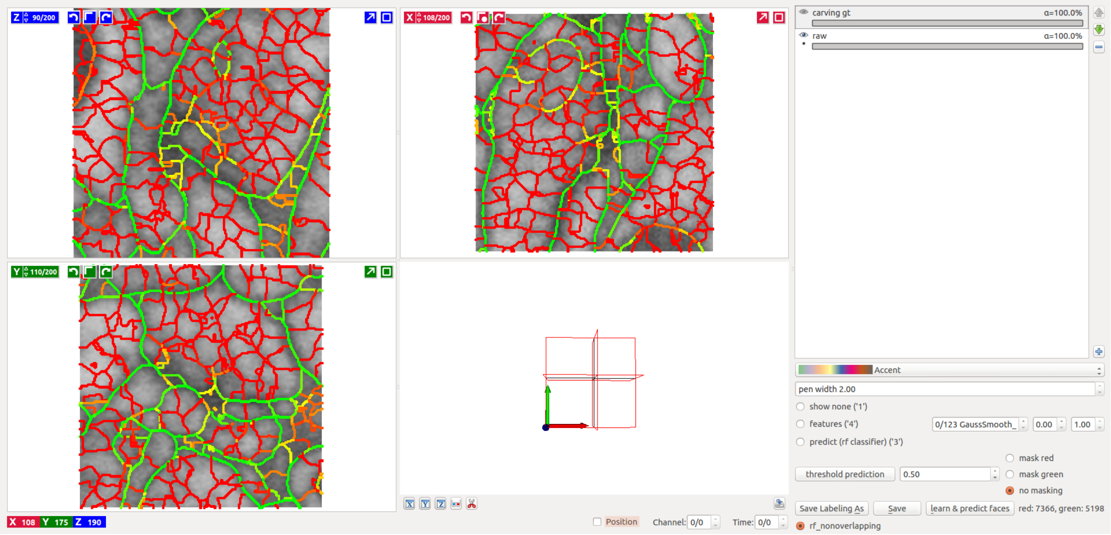


Figure 3.11: Screenshot of the FaceLabeler tool for interactive boundary learning based on pre-computed face features $\mathbf{f}_i$ and a Random Forest classifier.

Drawing Boundaries

We utilize this pre-processing for fast rendering in volumina. The BoundariesLayer class is derived from QGraphicsItem and takes a serialized slice as input. All boundaries can then be drawn quickly using successive calls to *drawPolyline* for each path. When additionally passing a real value for every face $f \in C_2$ (such as the value of a face feature), it can be visualized using a choice of color tables. Furthermore, BoundariesLayer implements user interaction. On mouse click, the nearest boundary is found and highlighted. If desired, the data entry for the clicked face $f \in C_2$ is toggled between zero and one, such that binary labeling can be implemented. The class is implemented in C++ and Qt and exposed to Python and PyQt using the sip[2] generator.

Interactive Boundary Learning

Using the BoundariesLayer component and building on top of the volumina viewer, we have implemented a GUI tool (class FaceLabeler) for learning a boundary classifier as shown in Figure 3.11. It offers the following features:

- display of the different pre-computed face features using a choice of color maps
- the creation of binary face labels via clicking on boundaries and the ability to save and load labels

[2] http://www.riverbankcomputing.co.uk/software/sip/intro

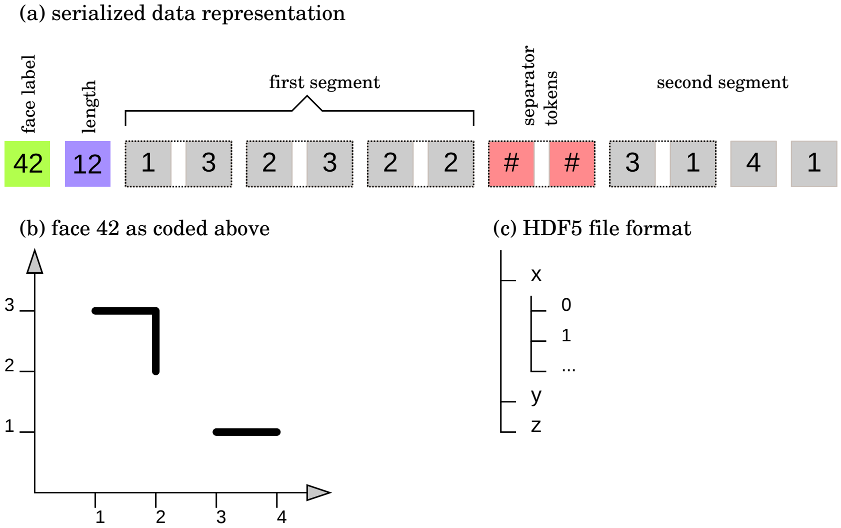


Figure 3.12: Data layout (a) for the serialized representation of the inter-voxel surface "42" as it appears on the slice shown in (b). Note that – while connected in 3D – the surface appears as two broken line segments within the slice. (c) The HDF5 file format for the entire volume image.

- training a Random Forest classifier at the click of a button; current prediction results are subsequently displayed using a colormap
- further analysis of the quality of the current prediction by thresholding in order to find misclassified samples (gaps in boundaries and spurious line segments)
- display multicut segmentation results (Chapter 4) as a separate layer.

In Chapter 4, we have utilized this application to quickly learn boundary classifiers for SBFSEM and FIBSEM datasets based on hundreds of boundary annotations. By inspecting the current result, and by analyzing gaps and spurious line segments in the thresholding view, it is easy to spot misclassified edges. These edges are iteratively added to the training set, thereby quickly improving the classifier.

3.8 Blockwise Array Processing

I have written the BLOCKEDARRAY library for two reasons: (i) to provide a clean and fast implementation of ilastik's cache of image tiles or volume blocks with optional in-memory compression, and (ii) to enable the computation of connected components on label volumes larger than main memory as needed for my joint work with Boray Tek on soma detection [148].

The library is written in C++, using VIGRA for multi-dimensional arrays and is exposed to Python with BOOST::PYTHON. It is available – together with extensive unit tests – as open source from `https://github.com/thorbenk/blockedarray`.

Briefly, with the `CompressedArray` class it provides a multi-dimensional array that can optionally be compressed *in-memory* using the fast SNAPPY[3] compression library. Especially for binary volumes or connected component labelings, compression can achieve a memory footprint that is an order of magnitude smaller.

The `Array` class represents a (potentially) large array whose data is stored in blocks internally. This has several advantages: (i) it allows for arbitrary large image volumes, of which only a small amount of data is loaded into RAM, possibly at various positions in the volume, (ii) it allows for in-memory compression: blocks that are currently not needed can be stored in RAM in compressed form, without incurring the hit of disk I/O, and (iii) using blocks instead of linear storage may alleviate cache-misses in the CPU caused by large strides when accessing certain array views.

The *Array* class also optionally keeps track of non-zero coordinates as well as the minimum/maximum value seen so far, and can save data version annotation ("dirtiness") down to the level of single slices within blocks. These features are a requirement for integration into lazyflow (Section 3.1).

As a foundation for blockwise processing, the interfaces `Source` and `Sink` define block-level read and write operations. All these classes have been used to implement some basic algorithms (such as thresholding) in a blockwise fashion, streaming from one file to another.

The blockwise connected component algorithm (illustrated in Figure 3.13) works by first running connected components independently on overlapping blocks. After labeling each block, it is stored compressed in memory. As a next step, each pair of overlapping blocks is de-compressed and a global union-find data structure updated to merge corresponding regions. Finally, each block is relabeled according to the union-find data structure and the result is written to disk.

In the work [148], this implementation achieved a compression factor of 1/20, enabling us to run our experiments on a standard desktop computer.

[3] `http://code.google.com/p/snappy`

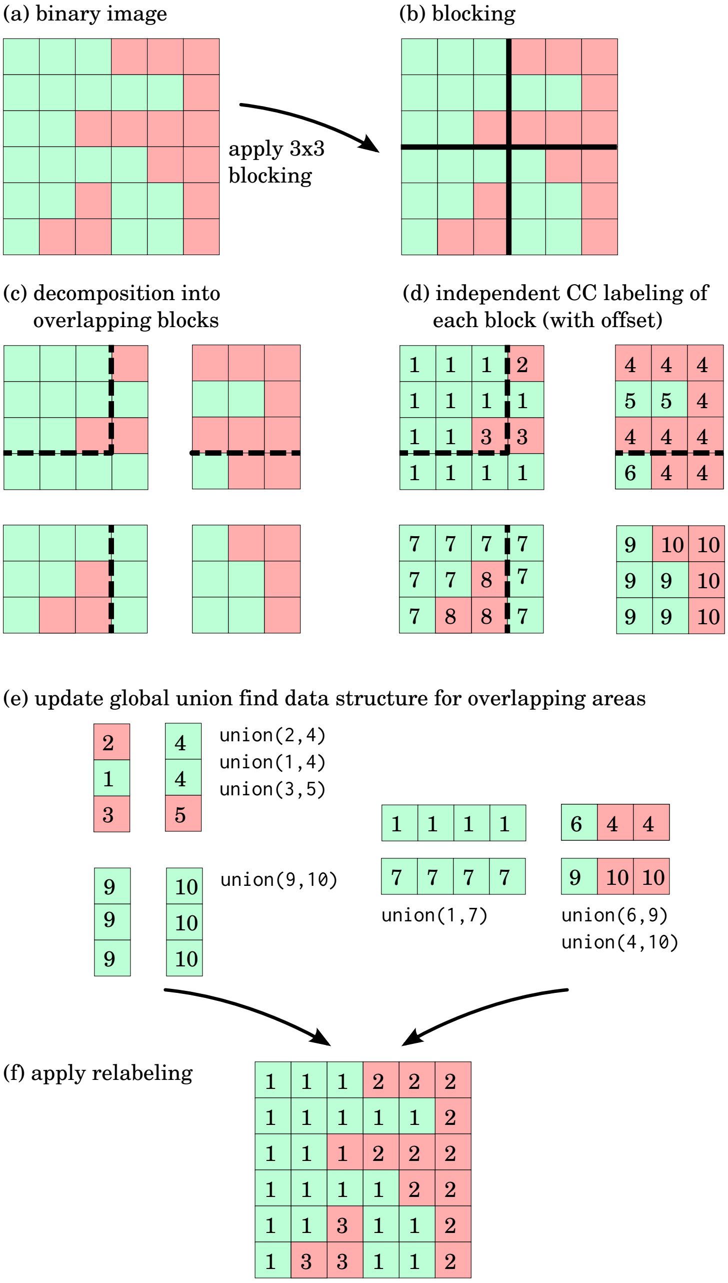

(a) binary image
(b) blocking
apply 3x3 blocking
(c) decomposition into overlapping blocks
(d) independent CC labeling of each block (with offset)
(e) update global union find data structure for overlapping areas
union(2,4)
union(1,4)
union(3,5)
union(9,10)
union(1,7)
union(6,9)
union(4,10)
(f) apply relabeling

Figure 3.13: Illustration of blockwise connected component labeling in the BLOCKEDARRAY library. By using blockwise processing and in-memory compression, datasets larger than RAM can be processed. For details see text.

3.9 Summary

In this chapter, I have presented various software solutions related to machine learning and image processing on large connectomics datasets. This software needed to be written for the implementation and evaluation of novel segmentation techniques, which are presented later in this thesis.

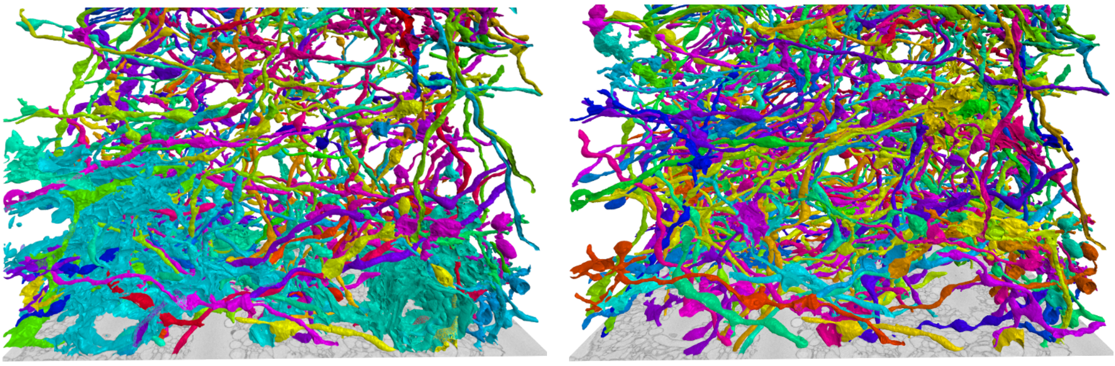
- I have introduced version 1.0 of the open source ILASTIK software framework, a toolkit and set of GUI programs for interactive machine learning on large volumetric image data, leveraging a lazy computation paradigm.
- In the context of this thesis, ilastik has been used both for interactive training of a voxel classifier (PIXEL CLASSIFICATION workflow, used in [10, 148]) and for acquiring gold standard segmentations (CARVING workflow, used in [10, 98, 20]).
- In particular, I have contributed to ilastik's VOLUMINA component (Section 3.2), a Python library for viewing and annotating large volumetric datasets. Volumina can handle larger-than-RAM datasets by a tile-based streaming approach and can display various data sources on top of the raw data.
- For interactive learning of the boundary classifier in [10], I have developed the FACELABELER tool (Section 3.7) which builds upon volumina.
- A cell complex representation of a label volume is beneficial when relating segments, surfaces between segments and other topological and geometric entities derived from the volume image. It can be easily converted to a Region Adjacency Graph representation if necessary. In Section 3.5, I have described an approach for dealing with hierarchies of cell complexes. This approach has been implemented as a C++ class called CELLCOMPLEXMERGER, which I haved used in [98], see Chapter 5.
- Finally, a new open source C++ library for blockwise array operations, called BLOCKEDARRAY, has been introduced in Section 3.8. This library was written both for the needs of ilastik's cache backend as well as for running connected component analysis on larger-than-RAM volume images as part of the soma detection pipeline in [148].

Chapter 4

Closed-surface Segmentation for Connectomics

In this chapter we address the problem of partitioning a volume image into a previously unknown number of segments, based on a likelihood of merging adjacent supervoxels. Towards this goal, we adapt a higher-order probabilistic graphical model that makes the duality between supervoxels and their joint faces explicit and ensures that merging decisions are consistent and surfaces of final segments are closed. First, we propose a practical cutting-plane approach to solve the MAP inference problem to global optimality despite its NP-hardness. Second, we apply this approach to challenging large-scale 3D segmentation problems for neural circuit reconstruction (connectomics), demonstrating the advantage of this higher-order model over independent decisions and finite-order approximations.

This chapter is an extended version of publication [10], also see page 145.



(a) without multicut constraints

(b) with multicut constraints

Figure 4.1: Segmenting volume images based on a likelihood of merging adjacent supervoxels is difficult if merging decisions are made independently. (a) Segmentation errors remain a problem even if the model is biased optimally with $\beta = 0.8$ in (4.9). Note the under-segmentation at the bottom and missing segments at the top. (b) Multicut constraints alleviate this problem and allow for an unbiased, parameter-free model.

4.1 Introduction

We study the problem of simultaneously merging pairs of adjacent supervoxels with the main application being neurite segmentation. Given a volume image, the number of segments contained within are previously unknown. However, the likelihood of merging adjacent supervoxels can be computed.

We choose a graphical model approach in which binary variables are associated with the joint faces of supervoxels, indicating for each face whether the two adjacent supervoxels should belong to the same segment (0) or not (1). Models of low order can lead to inconsistencies where a face is labeled as 1 even though there exists a path from one of the adjacent segments to the other along which all faces are labeled as 0. As a result, the union of all faces labeled as 1 need not form closed surfaces. Such inconsistencies can be excluded by a higher-order conditional random field (CRF) [5] that constrains the binary labelings to the multicut polytope [39], thus ensuring closed surfaces. While the number of multicut constraints can be exponential [2], constraints that are violated by a given labeling can be found in quadratic time [57]. The MAP inference problem can therefore be addressed by the cutting-plane method, i.e. by solving a sequence of relaxed problems to global optimality until no more constraints are violated [57].

Here, we show that the optimization scheme described in [5] is unsuitable for large 3D segmentations where the supervoxel adjacency graph is denser and non-planar. We therefore extend the cutting-plane approach by adding only constraints which are facet-defining by a property of the multicut polytope (Section 4.4.3), a double-ended, parallel search to find violated constraints (Section 4.4.2) and a problem-specific warm-start heuristic (Section 4.4.4). This approach can scale to volume images that consist of 10^9 voxels and are initially segmented into 10^6 supervoxels (Section 4.5).

The fact that exact MAP inference remains tractable at this scale is important for the reconstruction of neural circuits from electron microscopic volume images (Figure 4.1 and Chapter 1) where multicut constraints substantially improve the quality of segmentations across different imaging techniques (Section 4.5).

For this quantitative analysis, we manually segmented 10^9 voxels of two different datasets acquired at different laboratories using different imaging techniques and assess the performance of three models:

- the simplest, *local model* uses learned unary conditional probabilities that state, independently, if each face should be *on* (establish part of an object boundary) or *off* (merge adjacent supervoxels). Optimization of this model almost always results in inconsistent labelings (Figure 4.6b, 4.11b).
- The intermediate, *finite-order model* guarantees consistency across a small local horizon, leading to better results.
- The best results are obtained with the *fully constrained model* which admits only labelings that are globally consistent. With the cutting-plane approach proposed here, the full model is often faster to optimize than the finite-order approximation. Figures 4.6, 4.11 and 4.7, 4.13 summarize why the fully constrained method is the one we recommend.

4.2 Related Work

The problem we address is known as the multicut problem [41] in combinatorial optimization and as correlation clustering [16, 47] in statistics, and it is a special case of the partition problem [39]. Both problems are NP-hard [16, 56]. Instances of the multicut problem have been solved by tightening an outer approximation of the multicut polytope [141] via cutting planes [17, 57]. In computer vision, this technique has been applied in [79, 86, 116] where the relaxed LP is solved first, as well as in [5] where the integrality constraints are kept throughout the cutting-plane loop. Cutting planes have also been used to enforce connectivity in foreground vs. background image segmentation [152, 117, 100]. Here, we build on the probabilistic formulation in [5] but without the likelihood terms w.r.t. geometry that were shown to have a negligible effect on segmentations of photographs.

We concentrate on exact MAP inference for which we propose a cutting-plane approach that is efficient for large-scale 3D segmentation. In particular, we discuss the efficient search for violated constraints that are facet-defining and the parallelization of this search. We measure how the optimization runtime scales with the size of the image and the number of variables, respectively, with and without improvements.

Results were shown in [5] for photographs in the Berkeley Segmentation Dataset [107] which consist each of 10^5 pixels that were initially partitioned into 10^4 superpixels, leading to optimization problems with at most 10^4 variables. Here, we segment 3D images of up to 10^9 voxels which are initially partitioned into 10^5 supervoxels, leading to 100 times larger optimization problems with 10^6 variables.

4.3 Probabilistic Model

In order to make the duality between supervoxels and their joint faces explicit, we build a cell complex $\mathcal{C} = (C, \prec, \dim)$ representation of the supervoxel segmentation by connected component labeling of the finest possible grid-cell topology (for details, see Section 3.4). This yields disjoint sets of supervoxels C_3, joint-faces between supervoxels C_2, curves C_1 and points C_0. The function $\dim : C \to \mathbb{N}$ maps each cell to its dimension. For any cells $c, c' \in C = \bigcup_{j=0}^{3} C_j$, the relation $c \prec c'$ indicates that c bounds c', which implies $\dim(c) < \dim(c')$. Compared to a region adjacency graph (Section 3.4.1), this representation has the advantage that multiple disconnected joint-faces separating the same pair of supervoxels can be treated independently for feature extraction and classification.

4.3.1 Modeling Decisions

We model the posterior probability of a joint labeling $\boldsymbol{y} \in \{0,1\}^{|C_2|}$ of all faces, given

- features $\mathbf{f}_c \in \mathbb{R}^m$ of each face $c \in C_2$,
  the rows of the feature matrix $\mathbf{F} \in \mathbb{R}^{m \times |C_2|}$
- the bounding relation between faces and supervoxels,
  encoded in a topology matrix $\mathbf{T} \in \{0,1\}^{|C_2| \times |C_3|}$ in which $T_{cc'} = 1$ iff $c \prec c'$.

We make the following independence assumptions for all pairs of faces $c' \neq c'$:

$$\mathbf{F} \perp\!\!\!\perp \mathbf{T} \tag{a}$$

$$\mathbf{F} \perp\!\!\!\perp \mathbf{T} \mid \boldsymbol{y} \tag{b}$$

$$\mathbf{f}_c \perp\!\!\!\perp \mathbf{f}_{c'} \tag{c}$$

$$\mathbf{f}_c \perp\!\!\!\perp \mathbf{f}_{c'} \mid \boldsymbol{y} \tag{d}$$

$$y_c \perp\!\!\!\perp y_{c'} \mid \mathbf{f}_c \tag{e}$$

$$y_c \perp\!\!\!\perp y_{c'} \tag{f}$$

$$y_c \perp\!\!\!\perp \mathbf{f}_{c'} \quad . \tag{g}$$

Note, however, that the entries in $\boldsymbol{y}$ become dependent as soon as the topology is known; that is $y_c \not\perp\!\!\!\perp y_{c'} \mid \mathbf{T}$.

From these conditional independence assumptions follows together with Baye's rule (BR)

$$
\begin{aligned}
p(\boldsymbol{y}|\mathbf{F},\mathbf{T}) &\overset{\text{BR}}{=} \frac{p(\mathbf{F},\mathbf{T}|\boldsymbol{y})\; p(\boldsymbol{y})}{p(\mathbf{F},\mathbf{T})} && \text{(4.1a)}\\
&\overset{(a,b)}{=} \frac{p(\mathbf{F}|\boldsymbol{y})\; p(\mathbf{T}|\boldsymbol{y})\; p(\boldsymbol{y})}{p(\mathbf{F})\; p(\mathbf{T})} && \text{(4.1b)}\\
&= \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \cdot \frac{p(\mathbf{F}|\boldsymbol{y})}{p(\mathbf{F})} \cdot p(\boldsymbol{y}) && \text{(4.1c)}\\
&\overset{(c,d)}{=} \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \prod_{c\in C_2} \left\{ \frac{p\left(\mathbf{f}_c|\boldsymbol{y}\right)}{p\left(\mathbf{f}_c\right)} \right\} \cdot p(\boldsymbol{y}) && \text{(4.1d)}\\
&\overset{\text{BR}}{=} \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \prod_{c\in C_2} \left\{ \frac{p\left(\boldsymbol{y}|\mathbf{f}_c\right)}{p(\boldsymbol{y})} \right\} \cdot p(\boldsymbol{y}) && \text{(4.1e)}\\
&\overset{(e,f)}{=} \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \prod_{c\in C_2} \left\{ \left\{ \prod_{c'\in C_2} \frac{p\left(y_{c'}|\mathbf{f}_c\right)}{p(y_{c'})} \right\} \cdot p(\boldsymbol{y}_c) \right\} && \text{(4.1f)}\\
&= \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \prod_{c\in C_2} p(y_c|\mathbf{f}_c) \prod_{c\neq c'} \frac{p(y_{c'}|\mathbf{f}_c)}{p(y_c)} && \text{(4.1g)}\\
&\overset{(g)}{=} \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \prod_{c\in C_2} p(y_c|\mathbf{f}_c) \underbrace{\prod_{c\neq c'} \frac{p(y_{c'})}{p(y_c)}}_{=1} && \text{(4.1h)}\\
&\overset{\text{BR}}{=} \frac{p(\mathbf{T}|\boldsymbol{y})}{p(\mathbf{T})} \prod_{c\in C_2} \frac{p(\mathbf{f}_c|y_c)\; p(y_c)}{p(\mathbf{f}_c)} && \text{(4.1i)}\\
&\propto p(\mathbf{T}|\boldsymbol{y}) \prod_{c\in C_2} p(\mathbf{f}_c|y_c)\; p(y_c)\;\; . && \text{(4.1j)}
\end{aligned}
$$

The prior $p(y_c)$ is assumed to be identical for all faces and is specified with a single design parameter $\beta \in (0,1)$ as

$$
p(y_c) = \begin{cases} \beta & \text{if } y_c = 1 \\ 1-\beta & \text{if } y_c = 0 \end{cases} \;. \tag{4.2}
$$

4.3.2 Consistency

The likelihood $p(\mathbf{T}|\boldsymbol{y})$ is the instrument that is used to enforce consistency: while uninformative for all consistent labelings, it assigns zero probability to all inconsistent labelings

$$
p(\mathbf{T}|\boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{y} \text{ consistent} \\ 0 & \text{if } \boldsymbol{y} \text{ inconsistent} \end{cases} \;. \tag{4.3}
$$

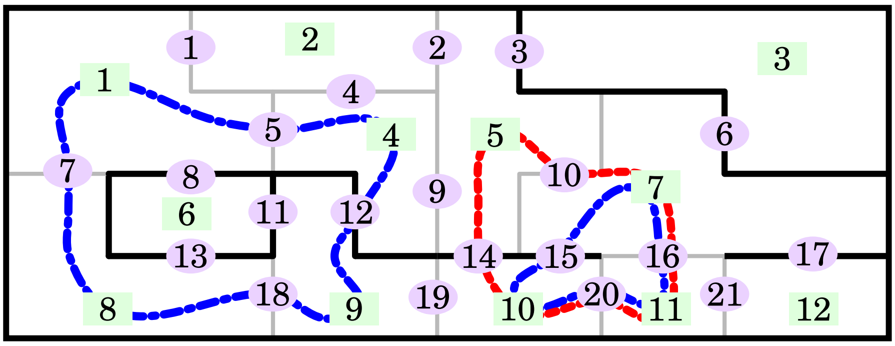


Figure 4.2: Depicted is one slice of a supervoxel segmentation. Supervoxels (square, green nodes) are bounded by joint faces (round, violet nodes) which are labeled as 0 (boundary is gray) or 1 (boundary is black). The simple circles drawn in blue are chordless; the red circle is chordal.

In Section 3.5, we have already described the consistency problem informally using the example in Figure 3.9. A more formal definition is

Definition (Consistent face labeling)
Let $\mathcal{C}$ describe the cell complex representation of a label volume. Vector $\boldsymbol{y} \in \{0,1\}^{|C_2|}$ attaches a binary decision to every face $c \in C_2$, which bounds segments $(s_1, s_2) \in C_3$. Setting $y_c = 1$ expresses the desire that s_1 and s_2 should be separated. Conversely, $y_c = 0$ signifies that s_1 and s_2 should be merged.

Now consider the connected component labeling of the segments C_3 according to $\boldsymbol{y}$, in which any adjacent segments separated by a face labeled "0" are merged.

If $y_c = 1$ but s_1 and s_2 belong to the same connected component, we say that the labeling of c is inconsistent*: locally, s_1 and s_2 should be separated, but in the connected component labeling, they are merged transitively.*

The labeling $\boldsymbol{y}$ is said to be inconsistent *as soon as any face $c \in C_2$ is inconsistent.*

A simple cycle of n segments, denoted cy $\in$ SC(n), is a sequence cy $= \{c_1, \ldots, c_{2n+1}\}$ of n pairwise distinct segments (the nodes of a graph) via n pairwise distinct faces (the edges of a graph), for which all of the following (Equations 4.4a-4.4d) is true:

$$c_1 = c_{2n+1} \tag{4.4a}$$

$$\forall j \in \{1, \ldots n\} \quad : \quad c_{2j-1} \in C_3 \wedge c_{2j} \in C_2 \tag{4.4b}$$

$$\forall j \in \{1, \ldots n\} \quad : \quad c_{2j} \prec c_{2j-1} \wedge c_{2j} \prec c_{2j+1} \tag{4.4c}$$

$$\forall j, k \in \{1, \ldots n\} \quad : \quad j = k \vee (c_{2j} \neq c_{2k} \wedge c_{2j-1} \neq c_{2k-1}) \ . \tag{4.4d}$$

For example, in Figure 4.2, the cycle $cy = \{11, 16, 7, 15, 10, 20, 11\}$ describes a cycle formed by segments 11, 7 and 10 together with the faces 16, 15 and 20. The set $SC(n)$ collects all possible such cycles of n segments.

A cycle $cy = \{s_1, t_1, \dots, s_n, t_n, s_1\} \in \mathrm{SC}(n)$ describes an *inconsistent* face t_1 if t_1 is one but all $t_i \neq t_1$ are zero. Alternatively, face t_1 is *consistent* if the following inequality holds, which ensures that along the cycle either none or more than one face is labeled as one:

$$t_1 \textit{ consistent } \Leftrightarrow y_{t_1} \leq \sum_{j=2}^{n} y_{t_j} \ . \qquad \text{(cycle inequality, 4.5)}$$

The set of all labelings $\boldsymbol{y} \in \{0,1\}^{|C_2|}$ which are *consistent* with respect to all cycles in $SC(n)$ is termed

$$\mathrm{MC}(n) = \left\{ \boldsymbol{y} \in \{0,1\}^{|C_2|} \;\middle|\; \forall (s_1, t_1, \dots s_n, t_n, s_1) \in \mathrm{SC}(n) : \ y_{t_1} \leq \sum_{j=2}^{n} y_{t_j} \right\} \ , \qquad (4.6)$$

and, for arbitrary n,

$$\mathrm{MC} = \bigcap_{j=1}^{|C_2|} \mathrm{MC}(j) \ . \qquad (4.7)$$

Consistent labelings $\boldsymbol{y} \in \mathrm{MC}$ are labelings inside the *multicut polytope*, the convex hull of MC, by Lemma 2.2 in [39].

4.3.3 Considered Models

The likelihood $p(\mathbf{f}_c|y_c)$ is learned by means of a Random Forest (cf. Section 2.4.2). More precisely, we learn $\hat{p}(y_c|\mathbf{f}_c)$ from class-balanced training data, i.e. with $\hat{p}(y_c) = 0.5$, and assume $p(\mathbf{f}_c|y_c) = \hat{p}(\mathbf{f}_c|y_c)$. Therefore, $p(\mathbf{f}_c|y_c) \propto \hat{p}(y_c|\mathbf{f}_c)\ \hat{p}(\mathbf{f}_c)$ and thus it follows from (4.1j) that

$$p(\boldsymbol{y}|\mathbf{F}, \mathbf{T}) \propto p(\mathbf{T}|\boldsymbol{y}) \prod_{c \in C_2} \hat{p}(y_c|\mathbf{f}_c)\ p(y_c) \ . \qquad \text{(global model, 4.8)}$$

For comparison, we consider two simpler models, a *local model* in which $p(\mathbf{T}|\boldsymbol{y})$ is uniform and thus faces are labeled independently,

$$p'(\boldsymbol{y}|\mathbf{F}, \mathbf{T}) \propto \prod_{c \in C_2} \hat{p}(y_c|\mathbf{f}_c)\ p(y_c) \ , \qquad \text{(local model, 4.9)}$$

and a *finite-order approximation* of (4.8) in which not all multicut constraints need to be fulfilled but only those that correspond to cycles up to length 4. With $p''(\mathbf{T}|\boldsymbol{y}) = \text{const.}$ if $\boldsymbol{y} \in \bigcap_{j=1}^{4} \mathrm{MC}(j)$ and $p''(\mathbf{T}|\boldsymbol{y}) = 0$, otherwise,

$$p''(\boldsymbol{y}|\mathbf{F}, \mathbf{T}) \propto \ p''(\mathbf{T}|\boldsymbol{y}) \prod_{c \in C_2} \hat{p}(y_c|\mathbf{f}_c)\ p(y_c) \ . \qquad \text{(finite order model, 4.10)}$$

4.4 MAP Inference

Instead of maximizing (4.8) to find the most likely solution, we minimize the negative log likelihood. For $p(\mathbf{T}|\boldsymbol{y}) = 1$ we have

$$\begin{aligned}
p(\boldsymbol{y}|\mathbf{F}, \mathbf{T}) &\propto \prod_{c \in C_2} \hat{p}(y_c|\mathbf{f}_c)\, p(y_c) \qquad &(4.11\text{a})\\
&= \prod_{c \in C_2} \hat{p}(y_c = 0|\mathbf{f}_c)^{1-y_c}\, \hat{p}(y_c = 1|\mathbf{f}_c)^{y_c}\, p(y_c = 0)^{1-y_c}\, p(y_c = 1)^{y_c} \qquad &(4.11\text{b})\\
-\log p(\boldsymbol{y}|\mathbf{F}, \mathbf{T}) &= \sum_{c \in C_2} -y_c \log \hat{p}(y_c = 1|\mathbf{f}_c) - (1 - y_c) \log p(y_c = 0|\mathbf{f}_c) \qquad &(4.11\text{c})\\
&\qquad - y_c \log \beta - (1 - y_c) \log(1 - \beta)\\
&\propto \sum_{c \in C_2} y_c \underbrace{\left[\log \frac{p(y_c = 0|\mathbf{f}_c)}{p(y_c = 1|\mathbf{f}_c)} + \log \frac{1 - \beta}{\beta}\right]}_{w_c} \qquad &(4.11\text{d})\\
&= \boldsymbol{w}^T \boldsymbol{y} \ . \qquad &(4.11\text{e})
\end{aligned}$$

4.4.1 Integer Linear Programming Problem

Consequently, the MAP solution of (4.8) is found by the following *integer linear program* (ILP)

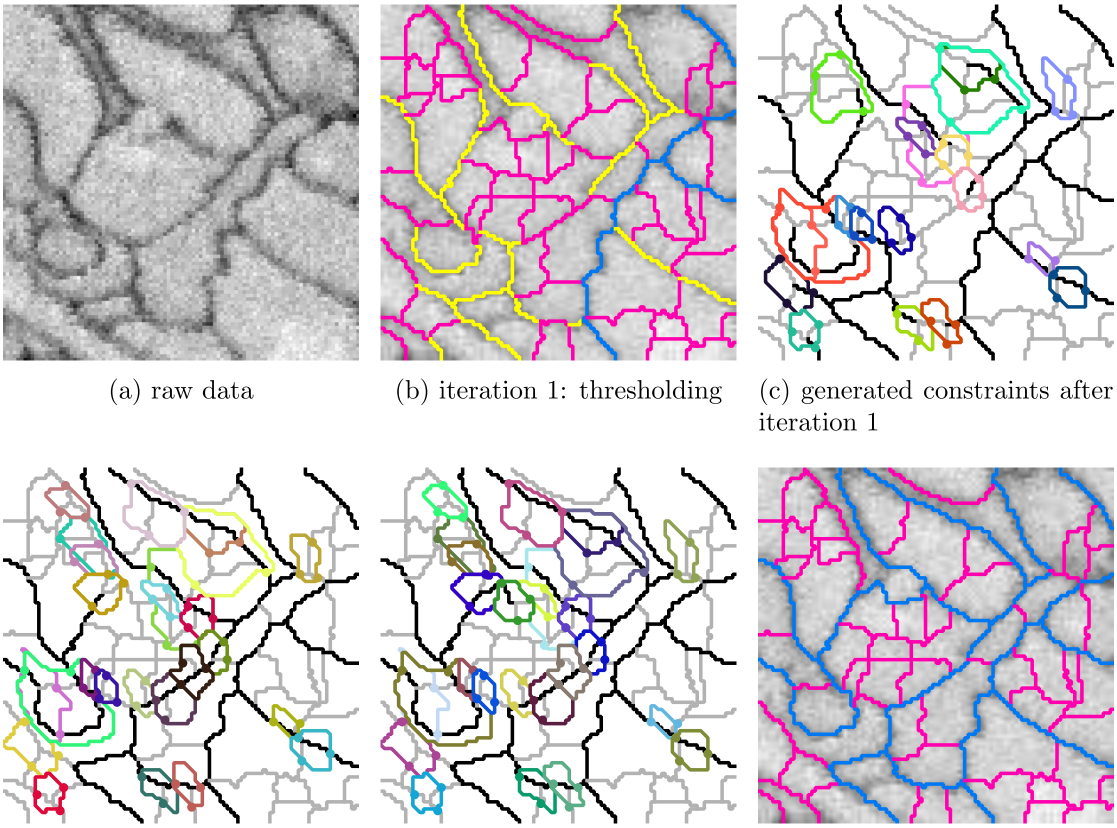
$$\boxed{\begin{array}{c} \min\limits_{\boldsymbol{y} \in \{0,1\}^{|C_2|}} \ \boldsymbol{w}^{\mathrm{T}} \boldsymbol{y} \\ \text{subject to} \ \ \boldsymbol{y} \in \mathrm{MC} \end{array}} \tag{4.12a}$$

with $\boldsymbol{w} \in \mathbb{R}^{|C_2|}$ such that $\forall j \in \{1, \dots, |C_2|\}$

$$w_c = \log \frac{p(y_c = 0|\mathbf{f}_c)}{p(y_c = 1|\mathbf{f}_c)} + \log \frac{1 - \beta}{\beta} \ . \tag{4.12b}$$

Note that the constraint $\boldsymbol{y} \in \mathrm{MC}$ ensures that all inconsistent solutions have zero probability, as stated by the chosen prior $p(\mathbf{T}|\boldsymbol{y})$, see (4.3).

We start by solving the (trivial) ILP without multicut constraints. The optimal solution is simply obtained by thresholding at zero, such that only faces with $w_c < 0$ contribute to the objective function $\boldsymbol{w}^T \boldsymbol{y}$. We then search for constraints that are violated by the solution, add these to the constraint pool and re-solve the constrained ILP using the branch-and-cut algorithm of a state-of-the-art solver. This procedure is repeated until no more multicut constraints are violated and thus the original problem (4.12a) has been solved to optimality. Figure 4.3 shows the iterations of this cutting plane procedure for a 2D segmentation example.



(a) raw data

(b) iteration 1: thresholding

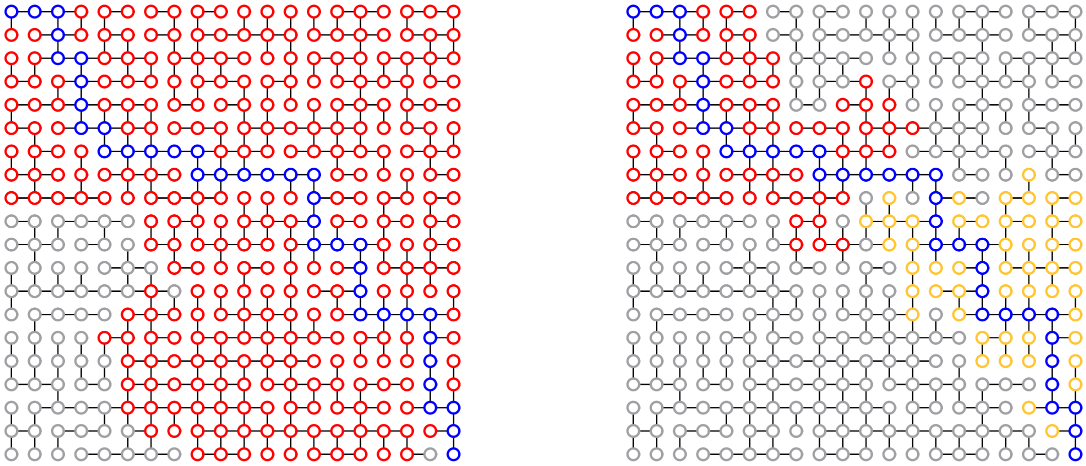
(c) generated constraints after iteration 1

(d) generated constraints after iteration 3

(e) generated constraints after iteration 4

(f) final result, iteration 5

Figure 4.3: Running the cutting plane algorithm on a 2D example problem. Using the raw data (a), superpixels were computed (b) and simple weights $\boldsymbol{w}$ for the faces were chosen. The panel shows the thresholding of these weights, such that positive w_c are red, and negative w_c are either yellow (inconsistent) or blue. Panels (c-f) show some cutting plane iterations. Here, the current solution $\boldsymbol{y}$ is shown in black ($y_c = 1$) and gray ($y_c = 0$). All constraints that have been generated up to this point are visualized. Each constraint is shown as a cycle with a random color, connecting faces (filled dot) and passing through the bounded segments. Panel (f) shows the final result, where no multicut constraints are violated anymore.



(a) single-ended path search

(b) double-ended path search

Figure 4.4: The number of nodes that need to be visited (all colored nodes) to find the shortest path between two given nodes (here: top left and bottom right node of the graph) can be reduced by starting a breadth-first search not just from the start node (a), but also from the target node (b).

4.4.2 Search for Violated Constraints

If any multicut constraints are violated by a given labeling $\boldsymbol{y}$, then at least one can be found by considering all faces $c \in C_2$ with $y_c = 1$ and looking for a shortest cycle $(s_1, t_1 = c, s_2, t_2, \ldots, s_n, t_n, s_1) \in \mathrm{SC}(n)$ with $n \in \mathbb{N}$ along which all other faces $t_2, \ldots, t_n$ are labeled as 0 [5, 57]. Shortest cycles correspond to constraints with minimal numbers of variables.

Double-ended search. Although a breadth-first search for such a cycle can be carried out in time $O(|C_2| + |C_3|)$, starting from either s_1 or s_2, the absolute runtime to perform this task for all relevant faces can be comparable to that of solving the ILP (Section 4.5). We therefore propose to grow two search trees, rooted at s_1 and s_2, simultaneously (illustrated in Figure 4.4). This saves runtime because both trees are only half as deep as a single one would be, at the point when a shortest cycle is found.

Parallelization. Shortest paths need to be found for many faces, from one adjacent supervoxel to the other, and yet not so many that it would be profitable to solve the All Pairs Shortest Path problem. We therefore propose to solve Single Pair Shortest Path problems in parallel with a space complexity that is linear in the number of threads. In practice, we use OpenMP for this embarrassingly parallel task.

4.4.3 Chordality Check

Not all inequalities in (4.7) define a facet of the multicut polytope due to the following

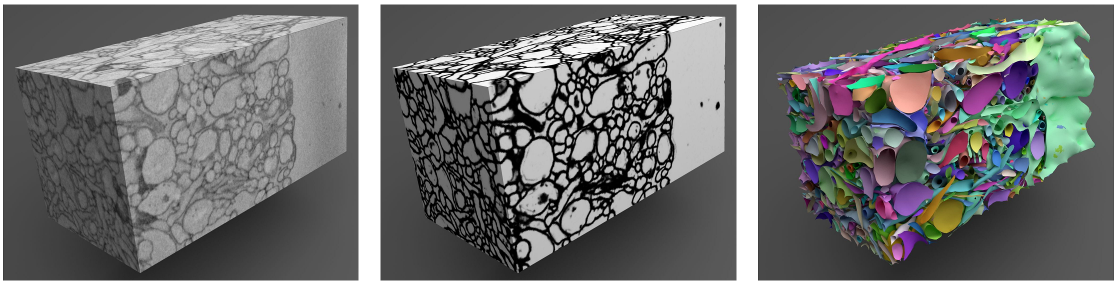
Theorem (facet-defining constraints [39])
Given a simple cycle $(c_j) = (s_1, t_1, \ldots s_n, t_n, s_1) \in \mathrm{SC}(n)$, the inequality $y_{t_1} \leq \sum_{j=2}^{n} y_{t_j}$ is facet-defining if and only if (c_j) is chordless.

A path is *chordless* (opposite: *chordal*) if each node is connected only to its successor and predecessor. Here, the path of segments via joint faces is chordless if each segment is connected by a face only to its successor and predecessor. Figure 4.2 illustrates both chordal and chordless cycles. For example the path $11, 16, 7, 15, 10, 20, 11$ is chordless. However, the path $11, 16, 7, 10, 5, 14, 10, 20, 11$ is chordal, because segments 7 and 10 are connected via face 15 in addition to their respective predecessors/successors.

We exploit this algorithmically by adding violated inequalities only if these correspond to chordless cycles.

4.4.4 Warm Start Heuristic

When violated constraints are added, the solution of the relaxed problem becomes infeasible and thus the upper bound on the global minimum is lost. However, the structure of the problem allows us to find a new feasible solution efficiently: a given labeling $\boldsymbol{y} \in \{0, 1\}^{|C_2|}$ is mapped to a labeling on the multicut polytope by labeling all variables of all violated inequalities as 0. Since violated inequalities have already been found, this heuristic does not change the runtime complexity of the optimization scheme overall.



(a) raw data (b) probability map (c) gold standard seg.

Figure 4.5: SBFSEM dataset (a) together with the voxel probability map (b) learned with ILASTIK. The gold standard segmentation (c) encompasses a volume of $400 \times 200 \times 200$ voxels.

4.5 Applications

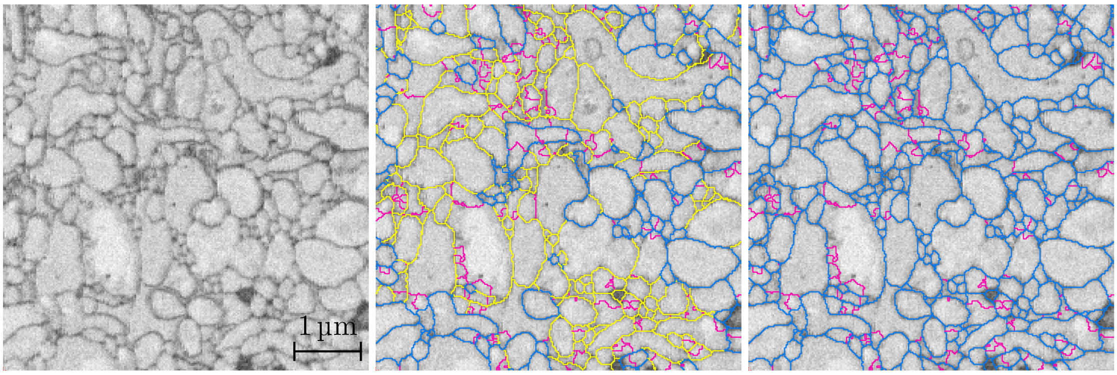
4.5.1 SBFSEM Volume Image

A volume image referred to as E1088 in [62] was acquired using serial block-face electron microscopy (SBFSEM) [49] and shows a section of rabbit retina at the almost isotropic resolution of $22 \times 22 \times 30\,\mathrm{nm}^3$. A small subset is shown in Figure 4.6a. The bright intra-cellular space that makes up more than 90% of the volume contrasts the stained extra-cellular space that forms thin membranous faces. This staining [33] simplifies the automated segmentation because no intra-cellular structures such as mitochondria or vesicles are visible (Figure 4.11a shows a different staining). A supervoxel segmentation is obtained as described in Section 4.8.3.

Features $\mathbf{f}_c$ of each face $c \in C_2$ described in Section 4.8.2 include statistics of voxel features over c as well as characteristics of the two supervoxels that are bounded by c. In order to learn $p(y_c|\mathbf{f}_c)$ by means of a Random Forest, 437 faces per class were labeled interactively in a subset of 250^3 voxels in about one hour, starting with obvious cases and continuing where the predictions needed improvement.

This online learning workflow was implemented as a custom extension of ILASTIK, and is described in detail in Section 3.7. The user can mark faces as “on”, “off” or “unlabeled” via a single mouse click and inspect intermediate predictions by viewing faces colored according to $\hat{p}(y_c|\mathbf{f}_c)$.

Qualitative results on independent test data are shown in Figure 4.6a-c. The global maximum of the local model (4.9) is inconsistent, i.e. not all surfaces are closed. A consistent labeling with closed surfaces and thus a segmentation is obtained by merging supervoxels transitively, i.e. whenever there exists a path from one supervoxel to the other along which all faces are labeled as 0 (cf. Section 4.4.4), regardless of how many faces between these supervoxels are labeled as 1. This mapping to the multicut polytope



(a) SBFSEM raw data (b) w/o constraints, $\beta = 0.5$ (c) with constraints, $\beta = 0.5$

Figure 4.6: Segmentations of the SBFSEM dataset (242^2 voxels). Faces are colored to show if the algorithm decides that these are part of a segment boundary (blue) or not (magenta). Importantly, yellow faces are decided to be part of a segment boundary, but are ignored because the bounded supervoxels are merged elsewhere.

is biased towards under-segmentation (Figure 4.6b). In contrast, the global maximum of the fully constrained model (4.8) is consistent and directly yields a segmentation with closed surfaces (Figure 4.6c). Note also the quality of the initial supervoxel segmentation from which only 19.3% of all faces are removed in the found global optimum.

Quantitatively, the effect of introducing multicut constraints is shown in Figure 4.7 where maxima of the global model (4.8), the local model (4.9) and the finite-order approximation (4.10) are compared to a man-made segmentation of $400 \times 200 \times 200$ voxels (Section 4.8.1) in terms of the Variation of Information (VI) [110] and Rand Index (RI) [124]. These segmentation quality measures are discussed in detail in Section 2.5.

The overall best segmentation is obtained from (4.8), i.e. with all multicut constraints, and without an artificial bias $\beta \neq 0.5$; note that the bias term in (4.12b) vanishes for $\beta = 0.5$. Without any multicut constraints, the best segmentation, obtained for $\beta = 0.8$, is worse in terms of both VI and RI. Segmentations of intermediate quality are obtained from the finite-order approximation (4.10).

Run times for optimizing (4.8) are shown in Figure 4.8 and Table 4.1 for our C++ implementation. The ILP (4.12a) is solved with IBM ILOG CPLEX and Gurobi alternatively, with the duality gap set to 0 in order to obtain globally optimal solutions. Global optima of (4.8), for a block of 800^3 voxels with $1.6 \cdot 10^6$ faces (i.e. variables y_c) are found in less than 13 minutes (Table 4.1), about 22 times as fast as with the optimization scheme in [5].

Maximizing (4.8) can be faster than maximizing the finite-order approximation (4.10) that does not guarantee closed surfaces and yields worse segmentations empirically (run times not shown). We therefore recommend to use *all* multicut constraints.

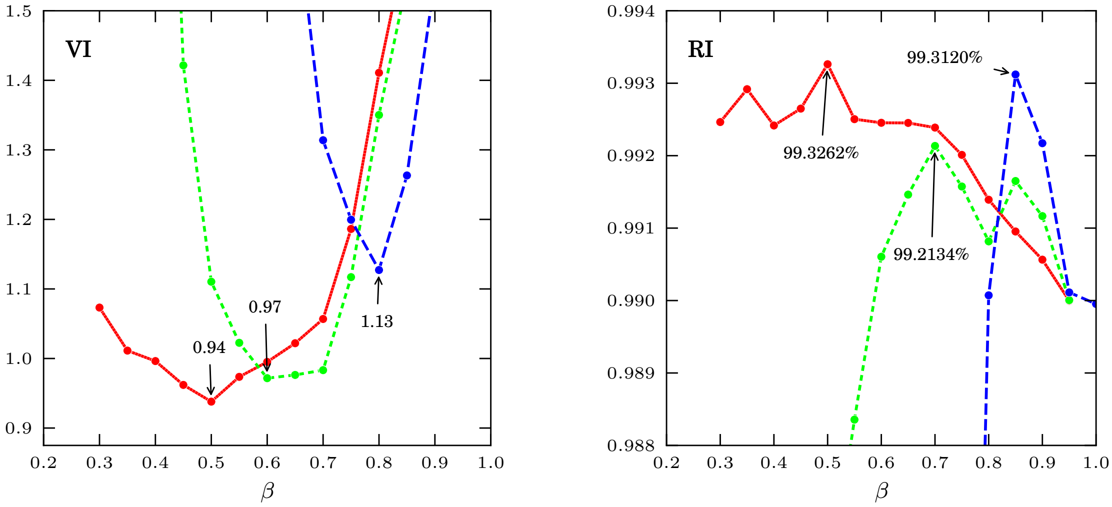
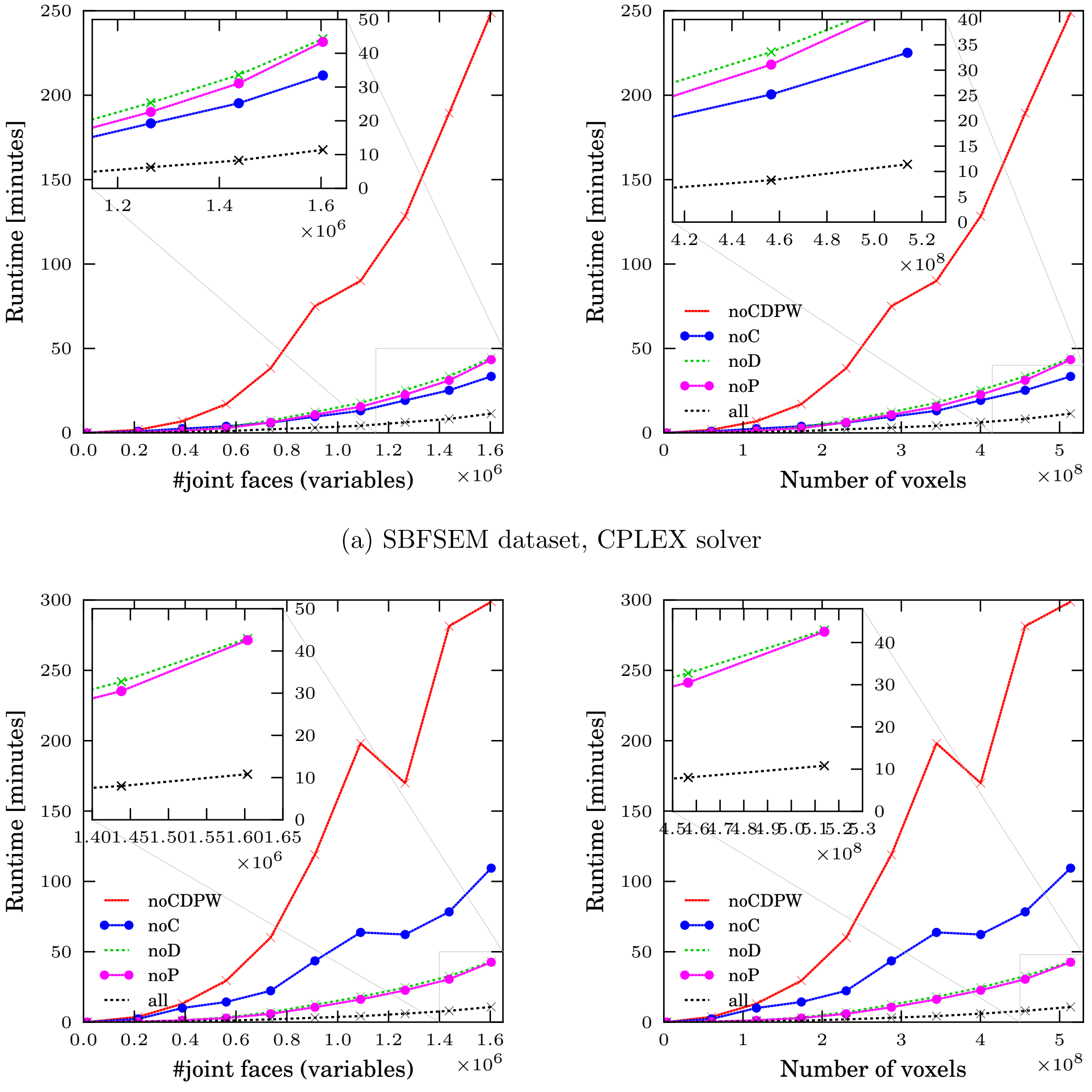


Figure 4.7: Variation of information (VI) and Rand Index (RI) wrt. the gold standard segmentation (400 × 200 × 200 voxels) of the SBFSEM dataset. For various priors β, shown are the optima of *red:* fully constrained model (4.8), *green:* finite-order approximation (4.10), *blue:* local model (4.9).

CPLEX	$\sqrt[3]{\lvert C_3\rvert}$	$\lvert C_2\rvert$	all	noW	noD	noC	noP	noCDPW
	150	13 987	0.01	**0.01**	0.01	0.02	0.01	0.01
	391	215 331	0.18	**0.17**	0.30	0.93	0.33	1.82
	488	388 849	0.57	**0.54**	1.34	2.47	1.26	6.86
	557	561 431	**1.04**	1.07	3.40	3.90	2.82	17.04
	612	736 037	**2.11**	2.23	7.32	6.03	6.24	38.27
	659	910 296	**3.07**	3.52	12.35	9.61	10.75	75.02
	700	1 089 973	**4.21**	4.80	17.79	13.08	15.50	90.07
	736	1 265 219	**6.26**	7.05	25.35	19.25	22.63	128.35
	769	1 438 276	**8.28**	8.94	33.59	25.21	31.10	189.41
	800	1 603 683	**11.41**	12.20	44.27	33.43	43.37	248.74
Gurobi	$\sqrt[3]{\lvert C_3\rvert}$	$\lvert C_2\rvert$	all	noW	noD	noC	noP	noCDPW
	150	13 987	0.01	0.01	0.01	0.02	**0.01**	0.01
	391	215 331	0.22	**0.21**	0.35	2.29	0.36	3.73
	488	388 849	**0.52**	0.67	1.31	10.02	1.27	13.06
	557	561 431	**1.08**	1.29	3.32	14.31	2.87	29.47
	612	736 037	**1.94**	2.31	7.01	22.25	5.92	60.09
	659	910 296	**3.11**	3.99	12.41	43.52	10.56	118.95
	700	1 089 973	**4.27**	5.30	18.02	63.75	16.16	198.14
	736	1 265 219	**6.02**	6.71	24.60	62.27	22.61	169.88
	769	1 438 276	**8.01**	9.27	32.72	78.36	30.49	281.43
	800	1 603 683	**10.81**	11.86	42.93	109.43	42.56	298.81

Table 4.1: Runtimes for the SFBSEM dataset. All times are given in minutes.



(a) SBFSEM dataset, CPLEX solver

(b) SBFSEM dataset, Gurobi solver

Figure 4.8: Wall clock run times for optimizing (4.8) with $\beta = 0.5$ (on an 8-core Intel i7 at 2.8 GHz), with and without improvements to the optimization scheme in Andres et al. [5], for increasing problem size (number of joint faces of supervoxels, left, and number of voxels, right), for datasets ranging in size from 150^3 through 800^3 voxels. *all*: proposed optimization scheme, *noC*: no chordality check, *noD*: no double-ended search, *noP*: no parallel search, *noCDPW*: no improvements.

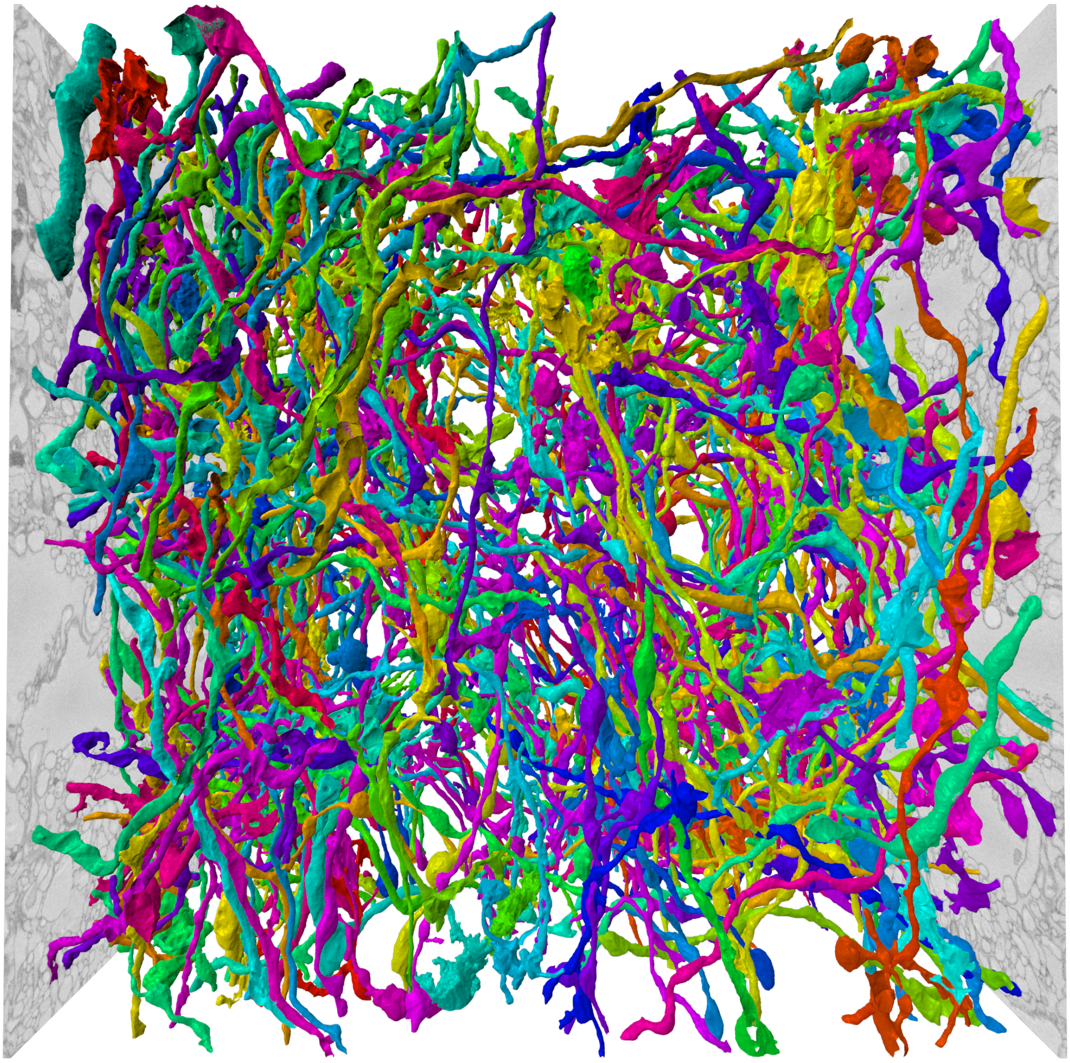
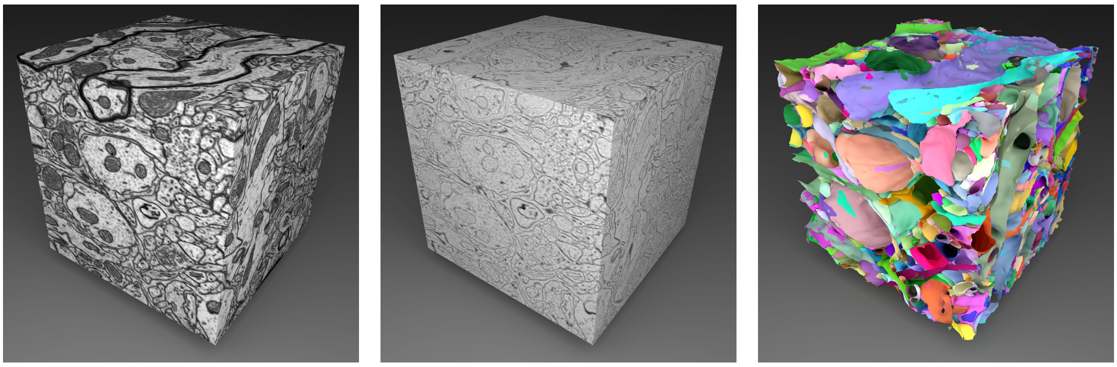


Figure 4.9: The largest SBFSEM volume image we have segmented with multicut constraints. Shown are 600 segments within a dataset of 1400 × 1400 × 600 voxels size.



(a) raw data (b) probability map (c) gold standard seg.

Figure 4.10: FIBSEM dataset (a) and the elevation map (b) used for obtaining the initial oversegmentation (largest value of Hessian eigenvalues). The gold standard segmentation (c) encompasses a volume of 900^3 voxels.

4.5.2 FIBSEM Volume Image

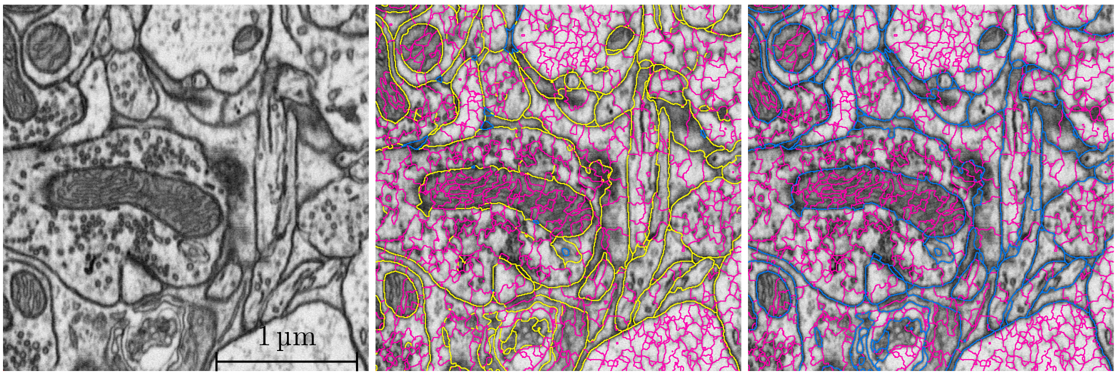
A volume image acquired with a focused ion beam serial-section electron microscope (FIBSEM) [90] shows a section of adult mouse somatosensory cortex at the almost isotropic resolution of $5 \times 5 \times 6\,\text{nm}^3$. A small subset is shown in Figure 4.11a. Intra- and extra-cellular space are indistinguishable by brightness and texture. Not only cell membranes but also intra-cellular structures such as mitochondria and vesicles are visible due to a different staining. The resolution is four times as high as that of the SBFSEM image. However, intra-cellular structures have membranous surfaces themselves and thus make the problem of segmenting entire cells more difficult. A supervoxel segmentation is obtained as described in Section 4.8.3.

We use the same features as for the SBFSEM dataset but adjusted in scale. Our labeling strategy was to annotate membranes of both cells and mitochondria[1] as $y_c = 1$.

Qualitative results are shown in Figure 4.11a-c for a slice of 512^2 voxels. The MAP labeling of the local model (4.9) is inconsistent and almost all faces are removed by transitivity (note the lack of blue faces in Figure 4.11b). In contrast, the global maximum of the fully constrained model (4.8) is consistent (Figure 4.11c). In contrast to the SBFSEM dataset in which only 19.3% of the faces are removed, 80.7% of all faces are removed here, due to the inferior supervoxel segmentation.

Quantitative results are shown in Figure 4.13. VI and RI are w.r.t. a complete segmentation of 900^3 voxels carried out by a neurobiologist (Section 4.8.1). Similarly as for the SBFSEM volume segmentation, the best segmentations are obtained from the full

[1] It is still possible to obtain the geometry of cells because mitochondria can be detected reliably [103], even via their mean gray-value once a segmentation is available. Segments which are classified as mitochondria are disregarded when computing VI [110] and RI [124].



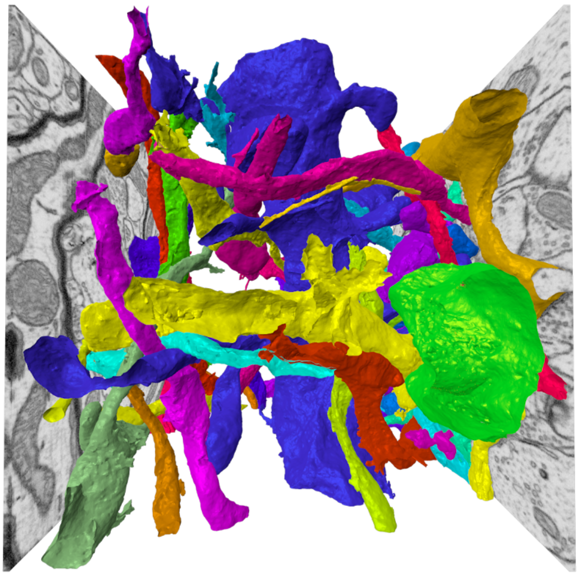
(a) FIBSEM raw data (b) w/o constraints, $\beta = 0.5$ (c) with constraints, $\beta = 0.5$

Figure 4.11: Segmentations of the FIBSEM dataset (512^2 voxels). Faces are colored to show if the algorithm decides that these are part of a segment boundary (blue) or not (magenta). Importantly, yellow faces are decided to be part of a segment boundary, but are ignored because the bounded supervoxels are merged elsewhere.

model (4.8). Exact MAP inference for the finite-order approximation (4.10) becomes intractable for most β.

Run times are reported in Figure 4.14 and Table 4.2 for problem instances from blocks between 150^3 and 800^3 voxels. Unlike for the SBFSEM dataset (Figure 4.8), the overall speedup is dominated by the chordality check.

Figure 4.12: The largest FIBSEM volume image we have segmented with multicut constraints. Shown are 50 objects in a volume of 900 × 900 × 900 voxels size. Note that – due to the high resolution – there are considerable less, but larger objects in the volume compared to the SBFSEM image (Figure 4.9).



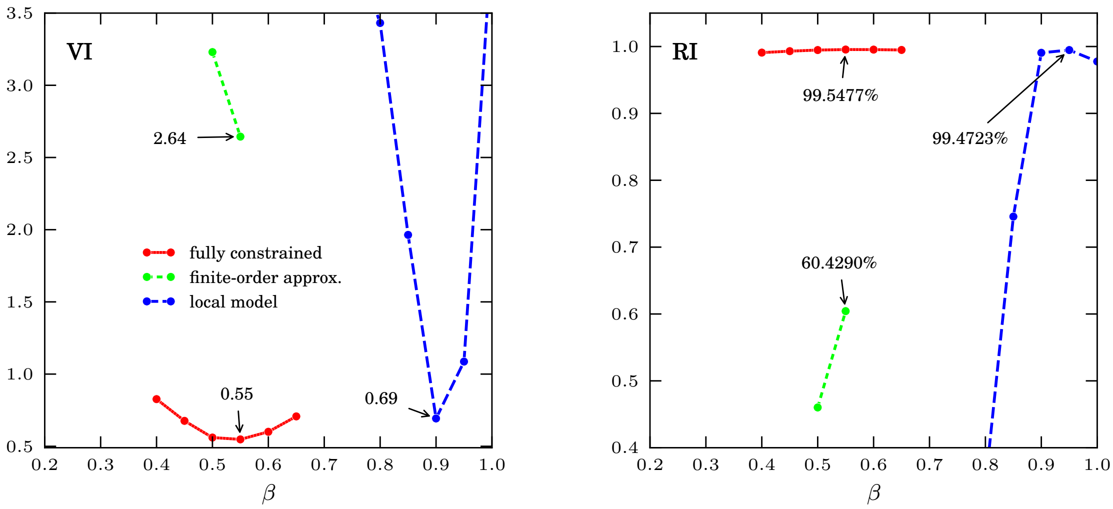
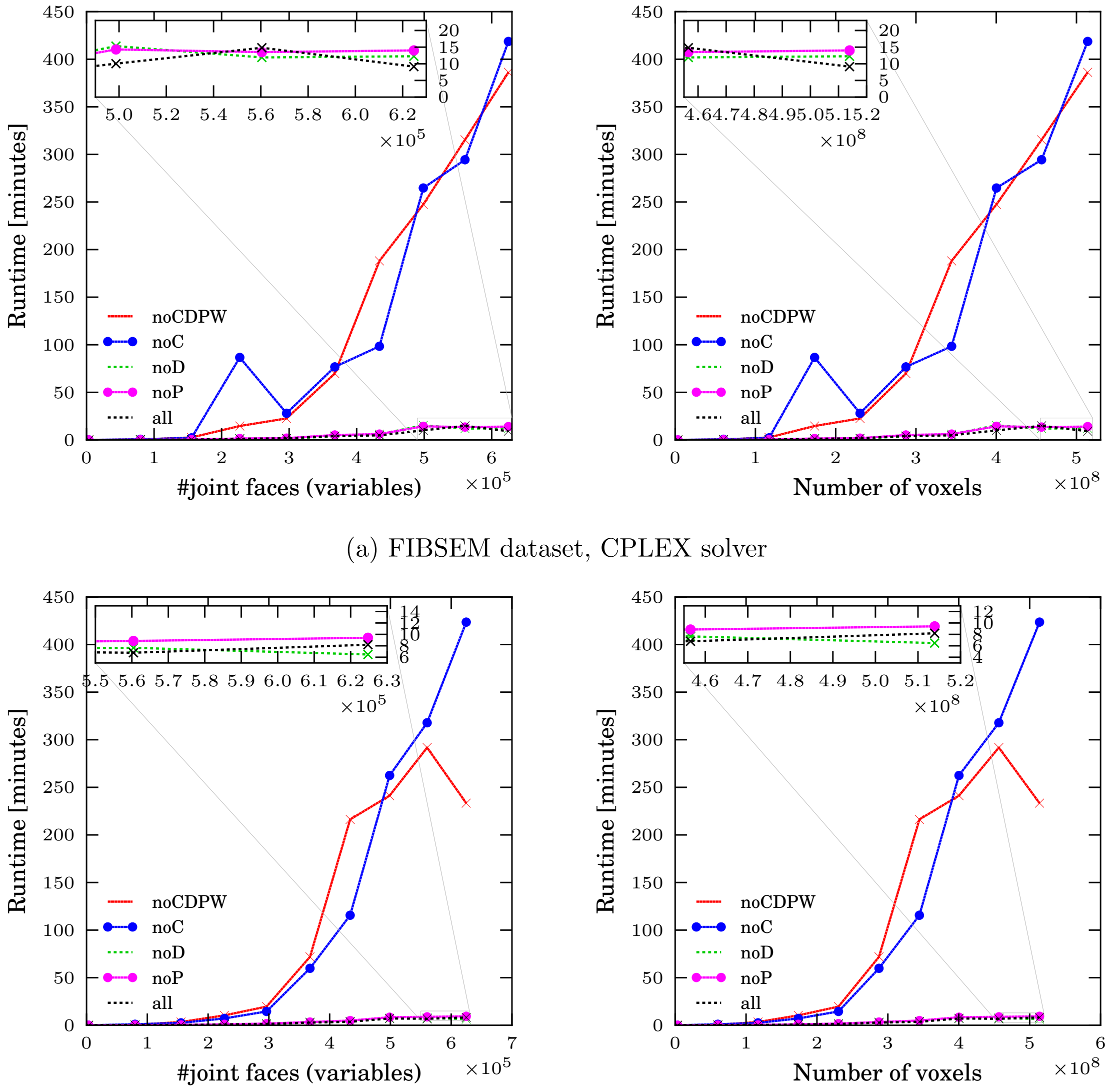


Figure 4.13: VI and RI compared to the gold standard segmentation (900 × 900 × 900 voxels) of the FIBSEM dataset. Exact optimization of the finite-order approximation becomes intractable for most β on the FIBSEM data. In contrast, optima of the full model are found in less than 13 minutes. (Figure 4.14).

CPLEX	$\sqrt[3]{\lvert C_3 \rvert}$	$\lvert C_2 \rvert$	all	noW	noD	noC	noP	noCDPW
	150	3416	0.02	0.02	0.02	0.02	0.03	**0.00**
	391	79 769	**0.24**	0.28	0.24	0.69	0.25	0.34
	488	155 439	0.31	**0.24**	0.34	2.13	0.38	2.61
	557	226 890	**1.14**	1.53	1.40	86.65	1.51	14.63
	612	296 171	1.73	2.21	**1.53**	28.01	1.87	22.58
	659	367 649	4.07	3.79	**3.67**	76.72	5.18	69.79
	700	433 743	**4.80**	7.63	6.03	98.39	6.13	188.12
	736	498 705	**10.04**	12.37	15.31	264.69	14.31	247.46
	769	560 433	14.82	13.50	**11.91**	294.41	13.51	315.37
	800	624 747	**9.15**	11.15	12.26	418.66	14.01	386.24
Gurobi	$\sqrt[3]{\lvert C_3 \rvert}$	$\lvert C_2 \rvert$	all	noW	noD	noC	noP	noCDPW
	150	3416	0.01	0.02	0.01	0.01	0.01	**0.0**
	391	79 769	0.24	0.26	0.25	1.02	**0.23**	0.45
	488	155 439	0.29	**0.27**	0.28	2.53	0.39	3.42
	557	226 890	**0.82**	0.98	0.87	7.17	1.00	10.39
	612	296 171	**1.29**	1.97	1.65	14.50	1.75	19.62
	659	367 649	**2.82**	3.09	3.28	59.82	3.40	71.90
	700	433 743	**3.63**	5.06	4.07	115.59	4.95	216.28
	736	498 705	7.13	**6.34**	7.40	262.46	8.44	241.28
	769	560 433	**6.79**	7.27	7.65	317.84	8.85	291.75

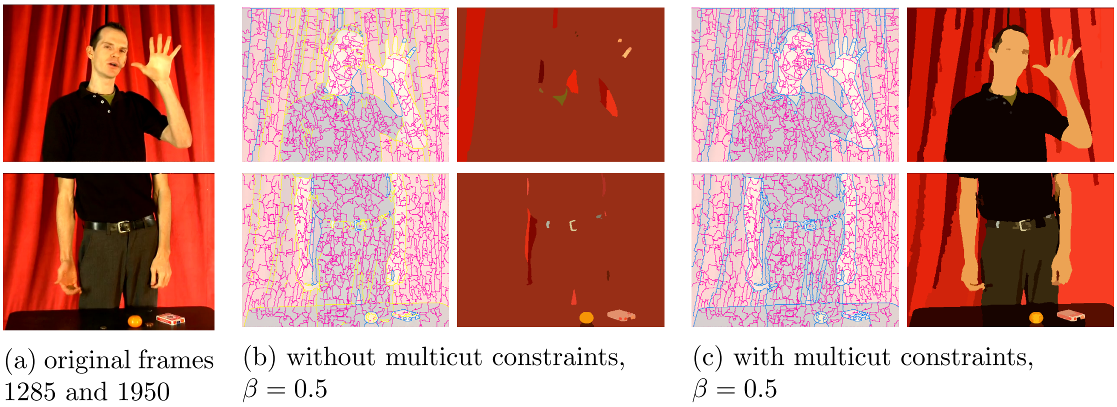
Table 4.2: Runtimes for the FIBSEM dataset. All times are given in minutes.



(a) FIBSEM dataset, CPLEX solver

(b) FIBSEM dataset, Gurobi solver

Figure 4.14: Wall clock run times for optimizing the fully constrained model with $\beta = 0.5$ (on an 8-core Intel i7 at 2.8 GHz), with and without improvements to the optimization scheme in Andres et al. [5], for increasing problem size (number of joint faces of supervoxels, left, and number of voxels, right), for datasets ranging in size from 150^3 through 800^3 voxels. *all*: proposed optimization scheme, *noC*: no chordality check, *noD*: no double-ended search, *noP*: no parallel search, *noCDPW*: no improvements.


(a) original frames 1285 and 1950

(b) without multicut constraints, $\beta = 0.5$

(c) with multicut constraints, $\beta = 0.5$

Figure 4.15: Color video segmentation. Supervoxel faces are colored blue if $y_c = 1$, magenta if $y_c = 0$ and yellow if the decision is inconsistent, i.e. $y_c = 1$ but the adjacent supervoxels are merged. Segments are filled with their mean color.

4.5.3 Video Segmentation

As a proof-of-concept, we applied the same model to bottom-up video segmentation, treating the video[2] as a three-dimensional (x,y,t)-volume. Obtaining a supervoxel segmentation that strikes a balance between negligible under-segmentation and a small number of excessive supervoxels has proven difficult. We settled for a marker-based watershed transformation of the color gradient magnitude in the Lab color space. A 486×360 video with 1000 frames (Figure 4.15a) is thus partitioned into $|C_3| = 22\,056$ supervoxels with $|C_2| = 256\,734$ joint faces.

As features $\mathbf{f}_c$, we use (i) the mean (over the face c) of the 2D patch features in [5] which are computed per frame, and (ii) the absolute distance of color histograms of the bounded supervoxels. A training set of 153 labels per class was acquired using the same tool and protocol as for the SBFSEM dataset. Only faces intersecting frame 1015 were labeled.

Qualitative results are shown in Figure 4.15. While the finite-order model is intractable for this problem, global optimization of the fully constrained model (4.8) with $\beta = 0.5$ takes 131 seconds.

[2] Frames 1000–2000 of the video at `youtube.com/watch?v=YN0I-TZFn58`

4.6 Discussion

In a fully connected graph with $n \in \mathbb{N}$ nodes, the $\binom{n}{3}$ cycle constraints of order three imply all higher-order cycle constraints. However, in the sparse graphs that we consider, the higher-order constraints need to be dealt with explicitly. An example is depicted in Figure 4.2 where the constraint that corresponds to the blue cycle on the left has order four, excluding from the feasible set a locally closed loop that is globally inconsistent. Our experiments have shown that including these higher-order constraints is essential to achieve the best performance with regard to ground truth.

When inconsistent labelings are permitted, unlike in the fully constrained model (4.8), and mapped to the multicut polytope as described in Section 4.4.4, the risk of false mergers is higher in 3D than in 2D because there are on average more and shorter paths from one supervoxel to another along which faces can be incorrectly labeled as 0. We therefore expect multicut constraints to be more important in 3D than in 2D.

Solving (4.12a) as proposed here requires the solution of problems of an NP-hard class. Whether or not this is tractable in practice depends on the quality of the predictions $\hat{p}(y_c|\mathbf{f}_c)$. The learning of this function is therefore especially important.

The warm start heuristic described in Section 4.4.4 is biased maximally towards undersegmentation. Finding smarter heuristics is an interesting problem for future research.

The reconstruction of neural circuits such as a neocortical column or the central nervous system of *Drosophila melanogaster* will eventually require the segmentation of volume images of 10^{12} voxels. The result that 10^{9} voxels can be segmented by optimizing a non-submodular higher-order multicut objective exactly on a single computer in 13 minutes is encouraging.

4.7 Towards Exploiting Biological Prior Knowledge

The multicut segmentation model described above draws upon *local* image evidence to enforce *global* consistency. It enforces the biological prior that membranes form closed surfaces. Any perforated membrane visible in the raw data is assumed to be due to staining artifacts. In this section, we discuss the possibility of incorporating additional biological priors within the multicut segmentation framework.

Bounded number of neurites. The number of neurites in a given volume image G is bounded. Each neurite which can be found in G has to originate from a soma (cell body) either inside or outside the imaged tissue block. In typical datasets of interest, somata occur rarely, and can even be found automatically [148]. Therefore, an upper bound on the number $N < N_{\max}$ of neurites in G can be found by counting the number of neurite cross sections on the six bounding faces ∂G of the cuboidal dataset and adding the number of somata found.

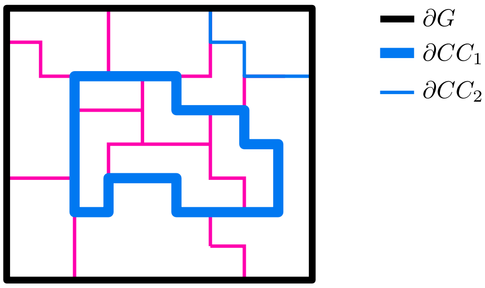


Figure 4.16: Notation for the model in which neurites are constrained to touch the border. Here, the labeling $\boldsymbol{y}$ induces three connected components of superpixels, CC_1 (does not touch the border ∂G of the image), CC_2 (touches ∂G), and CC_3.

Imposing the constraint $N < N_{\max}$ is difficult, because it only applies to segments corresponding to *neurites*. However, many segments are due to cell organelles, glial cells, extra-cellular space (depending on sample preparation) or imaging artifacts. In order to apply the above constraint, it would be necessary to first classify segments into neurite and non-neurite. Furthermore, implementing the constraint in the multicut segmentation framework is very difficult. This is because the number of segments is only encoded *implicitly* through the boundary indicator variables $\boldsymbol{y}$. We therefore turn to a relaxed variant of this constraint.

Neurites are connected to ∂G. From above it follows that no neurite can be wholly contained within G (unless somata exist) — each neurite has to reach the border somewhere. Let B be the set of all supervoxels that touch the border ∂G of the dataset G. A face labeling $\boldsymbol{y} \in \mathrm{MC}$ induces a labeling of the superpixels C_3 as connected components $CC(\boldsymbol{y}) = \{CC_1, \ldots, CC_N\}$. Then, the problem can be formulated as

$$\begin{aligned} &\min_{y} \left\{\boldsymbol{w}^T \boldsymbol{y}\right\} \quad \text{subject to} \\ &\boldsymbol{y} \in \mathrm{MC} \ , \\ &CC_i \cap B \neq \emptyset \ \ \forall CC_i \in CC(\boldsymbol{y}) \text{ with } CC_i \text{ neurite } \ . \end{aligned} \tag{4.13}$$

Problem (4.13) can be solved with the cutting plane method (see Figure 4.16 for notation). First, the ILP (4.8) is solved to obtain $\boldsymbol{y}$. Each connected component in the solution $CC(\boldsymbol{y})$ is classified to be either neurite or non-neurite. Then, for each neurite segment CC_i which does not touch the border, the following constraint is added to the problem:

$$\sum_{c \in \partial CC_i} y_c \neq |\partial CC_i| \ , \tag{4.14}$$

where ∂CC_i is the set of surfaces $c \in C_2$ which separate the connected component CC_i from the surrounding segments. Intuitively, the constraint enforces that the segment has to be merged with at least one other adjacent segment by creating a “hole” in the previously closed surface around CC_i. Next, this constrained problem is solved to yield a new labeling $\boldsymbol{y}$. The process is iterated until in the final solution, all induced segments either touch the border of the data cuboid or are classified as non-neurites.

I have implemented this algorithm, using the CPLEX integer linear programming solver and I have made the following observations:

- Even for small volume images (200^3 voxels, ca. 5000 supervoxels), optimization of (4.13) using cutting planes can take a considerable amount of time (tens of minutes). This is to be expected: each cutting plane iteration produces constraints which are not very informative. One solution is excluded, but many nearby solutions (slight variations in the extent of objects by adding or removing supervoxels) exist, which will have to be excluded in later iterations.
- The solution does not consistently improve in terms of Rand Index or Variation of Information. One reason could be that the model leaves too much freedom in how to re-assign supervoxels in order to fulfill the constraints: it does not consider cues such as "good continuation" based on the geometry of segments.

In summary, (4.13) contains too little non-local information (other than hard constraints) to yield useful results.

4.8 Detailed Methods

In this section, we give more details on the acquisition of the test set, the choice of both voxel and face features as well as details on the choice of supervoxels.

4.8.1 Acquisition of Gold Standard Segmentations

We manually segmented subsets of the SBFSEM and the FIBSEM dataset (Figures 4.5 and 4.10) using the interactive method "Carving" (Straehle et al. [144], see Section 3.3 for details on the integration into the ILASTIK software framework). Each object was segmented independently. Some independently segmented objects needed correction because there was overlap. This asserts a consistent ground truth and shows that the segmentation problem is non-trivial, even for a human. For the SBFSEM dataset, 528 objects (90.8% of $400 \times 200 \times 200$ voxels) were segmented by one expert in two weeks. For the FIBSEM dataset, 514 objects (96.8% of a cubic block of 900^3 voxels) were segmented by a neurobiologist in three weeks.

4.8.2 Voxel- and Boundary Features

From every joint face of adjacent supervoxels, 31 features are extracted (Table 4.4). One of these features is the response of a Random Forest that discriminates between membranes on the one hand and intra-/extra-cellular tissue on the other hand, based on 28 rotation-invariant non-linear features of local neighborhoods of 11^3 voxels (Table 4.3).

Index	Feature
1	Volume image
2	Bilateral filter
	$w_{\sigma_s}(r) = \frac{1}{\sigma_s(2\pi)^{3/2}} \exp\left(-\frac{r^2}{2\sigma_s^2}\right)$, $w_{\sigma_v}(v) = \frac{1}{1+\frac{v^2}{\sigma_v^2}}$
3–4	Gradient magnitude
5–16	Structure Tensor eigenvalues
17–28	Hessian matrix eigenvalues

Table 4.3: Features of voxel neighborhoods

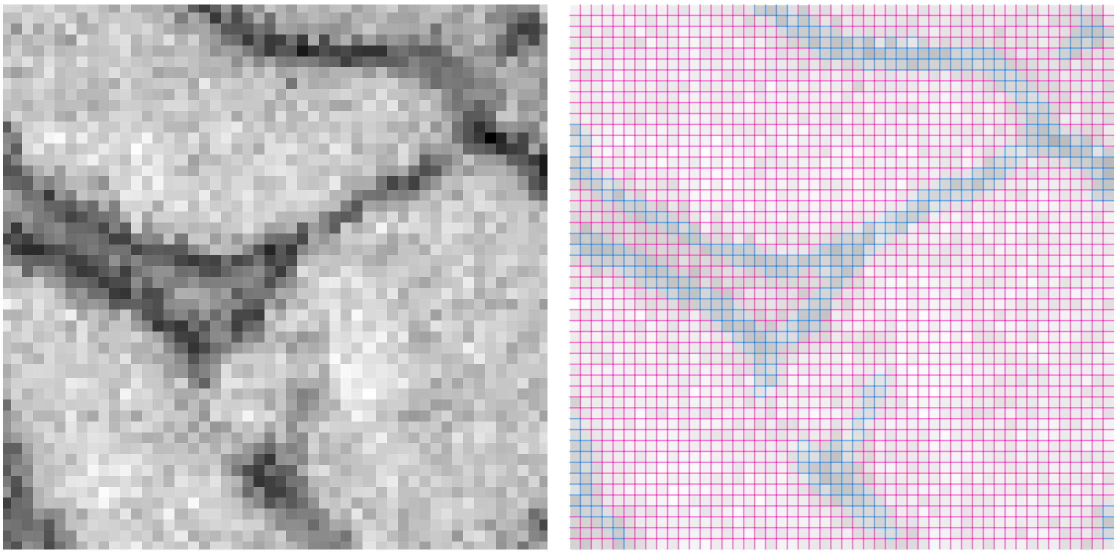
Index	Feature	Details
1	Size of the face	
2–3	Sizes v_1 and v_2 of	$(v_1 + v_2)^{1/3}$
	adjacent supervoxels	$\|v_1 - v_2\|^{1/3}$
4–10	Bilateral filter b	*
11–17	Gradient magnitude	*
18–24	Hessian matrix of	max. eigenvalue*
25–31	Voxel classifier	*

Table 4.4: Features of supervoxel faces. The statistics (*) include the min, max, mean, median, standard deviation, 0.25- and 0.75-quantile over all voxels adjacent to the face.

4.8.3 Supervoxel Segmentation

Are supervoxels necessary? First, we should pose the following question: why start from supervoxels and not simply voxels? Apart from the argument that supervoxels reduce problem size, there is a more fundamental reason: multicut segmentation loses its advantage of being able to close holes in a voxel-based setting. This is demonstrated in Figure 4.17. Boundaries (edgels in this settings) which lie on or near membranes in the image all obtain similar (negative) weights w_c via the Random Forest classifier. This is because (i) membranes are a few voxels in width and (ii) boundary features are computed from the voxel neighborhood and cannot be accurate to sub-voxel precision. Now, however, it is almost always energetically cheaper to remove inconsistencies by forming compact groups of membrane edgels, rather than closing "holes" in the membrane over long distances. In a sense, we have described the multicut equivalent of shrinking bias.

Supervoxels of sufficient size are therefore a pre-requisite for the segmentation approach described in this chapter.



(a) raw data (b) multicut segmentation

Figure 4.17: Formulating multicut segmentation for data (a) on the pixel level does not make sense. Because boundaries cannot be given a pixel-precise location (they have widths of multiple pixels), many edgels on and near the boundaries are given negative weights w_c. In the multicut segmentation result (b), this leads to over-segmentation into single pixels along the visible membranes. Even more importantly, multicut segmentation loses the advantage of being able to close membrane "holes": It is almost always energetically cheaper to form compact clusters of membrane edgels. This is the multicut equivalent of graph cut's shrinking bias.

Details on supervoxel generation. For the experiments in Section 4.5, we computed supervoxels by marker-based watersheds (see Section 2.4.5). To obtain an elevation map and markers for the SBFSEM dataset, we train a Random Forest classifier to distinguish between two classes of voxels, *extra-cellular space* and *intra-cellular space*, based on rotation invariant features of local neighborhoods (Table 4.3 and Section 4.8.2). As training data, 1600 voxels per class were labeled interactively in two subsets of 150^3 voxels which has taken three hours using ILASTIK [140]. Predicted probabilities are used as elevation levels. Connected components of at least three voxels classified as intra-cellular space are used as markers. For the FIBSEM dataset, the elevation level is defined as the largest eigenvalue of the Hessian matrix at scale $\sigma = 1.6$. Markers are taken to be maximal plateaus of the raw data that consist of at least two voxels.

4.9 Conclusion

In this chapter we have addressed the problem of segmenting volume images based on a learned likelihood of merging adjacent supervoxels. To solve this problem, we have adapted a probabilistic model that enforces consistent decisions via multicut constraints to 3D cell topologies and suggested a fast scheme for exact MAP inference. The resulting 22-fold speedup has allowed us to systematically study the positive effect of multicut constraints in large-scale 3D segmentation problems for neural circuit reconstruction. The best segmentations have been obtained for an unbiased parameter-free model with multicut constraints.

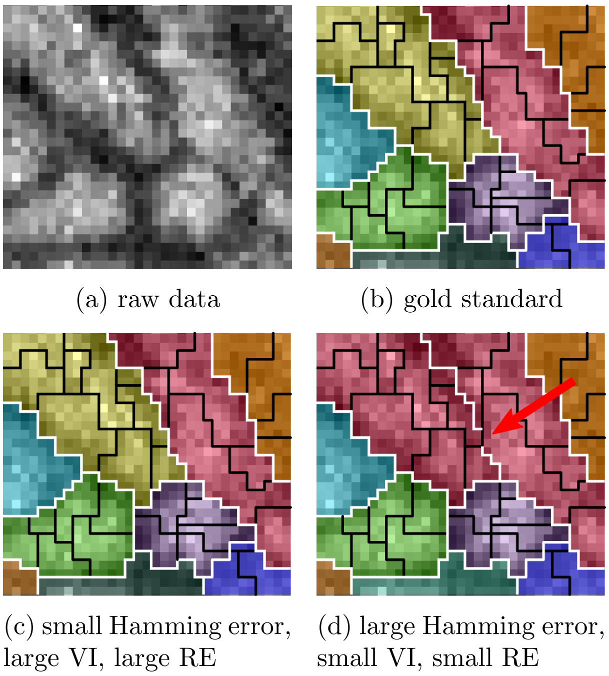
Here, we have used a Random Forest classifier for learning $p(y_c|\mathbf{f}_c)$. This probability was then used to derive the weights w_c in the multicut model (4.12a). However, an open question is whether the weights w_c can be chosen directly such as to yield better segmentations as measured by Rand Index or Variation of Information. This is the subject of Chapter 5.

Chapter 5

Learning to Segment with a Global Loss Function

Segmentation schemes such as hierarchical region merging or correllation clustering rely on edge weights between adjacent (super-)voxels. The quality of these edge weights directly affects the quality of the resulting segmentations. *Unstructured* learning methods seek to minimize the classification error on *individual* edges. This ignores that a few local mistakes (tiny boundary gaps) can cause catastrophic global segmentation errors. Boundary evidence learning should therefore optimize *structured* quality criteria such as Rand Error or Variation of Information. We present the first structured learning scheme using a structured loss function; and we introduce a new hierarchical scheme that allows to approximately solve the NP hard prediction problem even for huge volume images. The value of these contributions is demonstrated on two challenging neural circuit reconstruction problems in serial sectioning electron microscopic images with billions of voxels. Our contributions lead to a partitioning quality that improves over the current state of the art.

This chapter is an extended version of publication [98], also see page 145.


(a) raw data (b) gold standard

(c) small Hamming error, large VI, large RE (d) large Hamming error, small VI, small RE

Figure 5.1: Segmentation quality is commonly [10, 13] measured by Rand Error [124] and Variation of Information [110] which capture (in contrast to the Hamming error HE) not only local segmentation quality, but also some global structural correctness. (b) Gold standard for supervoxel segmentation: white edges are correct, black edges incorrect. (c) A single missed edge (arrow) has the catastrophic global consequence of merging the two adjacent regions (resulting in the red segment). (d) The slightly displaced boundary (arrow) produces a large HE, but small RE or VI.

5.1 Introduction

Connectomics requires extremely accurate circuit reconstruction because minor local mistakes can lead to catastrophic global connectivity errors (Chapter 1). Automatic methods that achieve the required accuracy level and scale to huge datasets are still an open problem (Chapter 2). When segmentation is based on electron microscopy volume images, one must exclusively rely on boundary evidence, because the desired regions (neurons) cannot be differentiated on the basis of appearance features.

For such partitioning problems, correlation clustering [16, 53], or multicut segmentation [39], is a powerful paradigm [153, 87, 5, 79, 15, 159, 10] as we have demonstrated in Chapter 4. An image is represented as a weighted region adjacency graph of (super)-voxels. Positive edge weights indicate that the incident regions should be merged; negative weights indicate that they should be kept separate. An optimal segmentation makes binary decisions for each edge so as to minimize the total cut weight, subject to the constraint of producing a topologically consistent solution [39].

The quality of the resulting segmentation depends critically on the edge weights, which are some function of features computed from the raw data. We focus on learning such weights using a cutting-planes approach. In each iteration, a structured loss is used to compare the segmentations obtained from the current weights to the gold standard.

Ideally, [149, 69], the loss function takes the entire segmentation into account (Figure 5.1). Unfortunately, such loss functions do not decompose over the binary decisions for individual edges, prohibiting efficient inference. This is why previous work has resorted to merely counting the number of deviating edge decisions between gold standard and current prediction [87]. Figure 5.10 shows that better results can be achieved when a *structured* loss function is used during training. Various attempts have been made to train struct-SVMs with more complex loss functions, but these approaches are tailored to specific applications [147], or use approximation techniques that do not apply to RE and VI [125].

Our first contribution (Section 5.2) is to allow arbitrary structured loss functions, such as Rand Error (RE) or Variation of Information (VI) during struct-SVM training on moderately-sized neighborhoods. This is made possible by a non-redundant and efficient exhaustive enumeration of segmentations. Our second contribution is a hierarchical, blockwise scheme for the structured prediction on large volumes. It produces segmentations that are empirically close to optimal (Section 5.3). Experiments on two different electron microscopy volume images of neural tissue show that the proposed method can improve upon unstructured learning using SVM or Random Forest classifiers (Section 5.4).

5.2 Structured Learning for Segmentation

As in Chapter 4 we work in a dual representation which specifies a segmentation in terms of binary labels $\boldsymbol{y}$ pertaining to boundaries between supervoxels. However, not every candidate configuration $\boldsymbol{y} \in \{0,1\}^{|C_2|}$ (where $|C_2|$ is the number of boundaries) represents a valid segmentation: if $y_i = 1$ (boundary is correct) but the adjacent supervoxels belong to the same region (i.e. there exists a path between these supervoxels along which all y_k are labeled $y_k = 0$), $\boldsymbol{y}$ is *inconsistent*, (Figures 5.1c, 5.2b and Section 4.3.2). All consistent $\boldsymbol{y}$ form the set of *multicuts* MC [39]. Region labels are easily determined by connected components. The alternative approach to assign region labels directly (primal representation) leads to a much larger search space, see Table 5.1.

We employ a structured risk minimization formulation in order to learn suitable edge weights from training data. Structural risk minimization aims at finding a regularized predictor that minimizes the empirical loss [118]. Training samples $\boldsymbol{x}^n$ are connected components of surfaces with the gold standard labeling $\boldsymbol{y}^n$. Sample $\boldsymbol{x}$ and any labeling $\boldsymbol{y}$

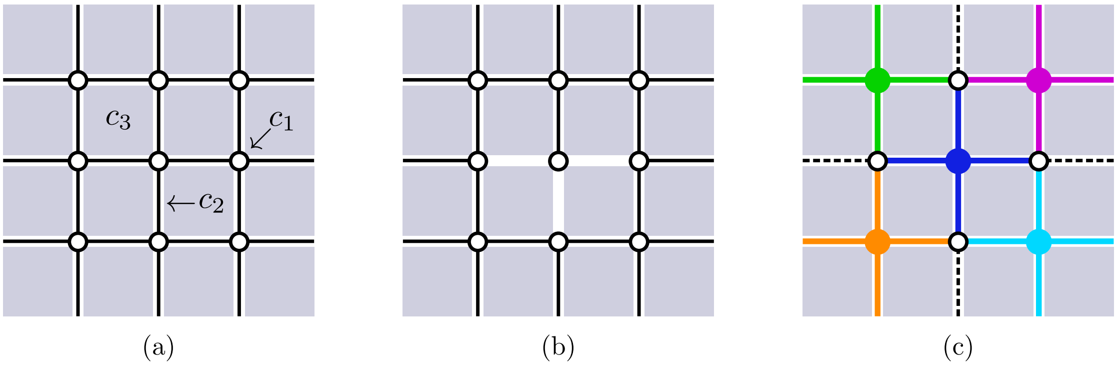


Figure 5.2: (a) 2D slice through a 3D cell complex representation. (b) "Dangling" surfaces can be detected locally. (c) More efficient enumeration uses the colored lines and their bounded surfaces.

are described by a joint feature vector $\phi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^M$. The optimal multicut segmentation $\boldsymbol{y}^*$ according to a structured SVM model [118] is then given by

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathrm{MC}}{\operatorname{argmin}} \langle \mathbf{m}^*, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle \quad , \tag{5.1}$$

$$\mathbf{m}^* = \underset{\mathbf{m} \in \mathbb{R}^D}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\mathbf{m}\|^2 + \frac{\lambda}{N}\, l(\mathbf{m}) \right\} \quad , \tag{5.2}$$

$$l(\mathbf{m}) = \sum_{n=1}^{N} \max_{\boldsymbol{y} \in \mathrm{MC}} \left\{ \Delta(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{m}^T \phi(\boldsymbol{x}^n, \boldsymbol{y}^n) + \mathbf{m}^T \phi(\boldsymbol{x}^n, \boldsymbol{y}) \right\} \quad . \tag{5.3}$$

The model is trained by finding optimal weights $\mathbf{m}^*$. The *loss function* $\Delta(\boldsymbol{y}^n, \boldsymbol{y})$ measures the deviation of $\boldsymbol{y}$ from the gold standard (e.g. by using RE or VI). Hyperparameter λ trades off the regularization and data terms.

How should $\phi(\boldsymbol{x}, \boldsymbol{y})$ be chosen? The objective is linear in the edge weights $\boldsymbol{w}$ subject to exponentially many constraints restricting $\boldsymbol{y}$ to multicuts (cf. Equation 4.12a).

$$\boxed{\begin{aligned} &\min_{\boldsymbol{y} \in \{0,1\}^{|C_2|}} \quad \boldsymbol{w}^{\mathrm{T}} \boldsymbol{y} \\ &\text{subject to} \quad \boldsymbol{y} \in \mathrm{MC} \end{aligned}} \tag{5.4}$$

We seek to optimize the edge weights $\boldsymbol{w}$ indirectly by choosing optimal feature weights $\mathbf{m}$ in the ansatz $w_i = \langle \mathbf{m}, \boldsymbol{\alpha}^{(i)} \rangle$, where $\boldsymbol{\alpha}^{(i)}$ is a suitable feature vector associated with each edge. Note that $\boldsymbol{w}$ is learned from entire configurations, whereas existing methods [5, 10, 159] learn a probabilistic model $p(y_i | \boldsymbol{\alpha}^{(i)})$ for *individual* edges and then define

$$w_i = \log p(y_i = 0 | \boldsymbol{\alpha}^{(i)}) \ / \ \log p(y_i = 1 | \boldsymbol{\alpha}^{(i)}). \tag{5.5}$$

This yields after an exchange of summation order

$$\min_{\boldsymbol{y}\in\text{MC}} \sum_{i=1}^{|C_2|} \left\langle \mathbf{m}, \boldsymbol{\alpha}^{(i)} \right\rangle y_i \quad = \quad \min_{\boldsymbol{y}\in\text{MC}} \sum_{i=1}^{|C_2|} \sum_{j=1}^{M} m_j \alpha_j^{(i)} y_i \tag{5.6a}$$

$$= \quad \min_{\boldsymbol{y}\in\text{MC}} \sum_{j=1}^{M} m_j \sum_{i=1}^{|C_2|} \alpha_j^{(i)} y_i \tag{5.6b}$$

$$\stackrel{\text{def}}{=} \quad \min_{y\in\text{MC}} \left\langle \mathbf{m}, \phi \right\rangle \ . \tag{5.6c}$$

The joint feature vector $\phi(\boldsymbol{x}, \boldsymbol{y})$ has length M, the number of features for each surface:

$$\phi(\boldsymbol{x}, \boldsymbol{y})^T = \left(\sum_{i=1}^{|C_2|} \alpha_1^{(i)} \cdot y_i, \cdots, \sum_{i=1}^{|C_2|} \alpha_M^{(i)} \cdot y_i \right) . \tag{5.7}$$

The key operation for determining optimal weights in (5.3) is the maximization over all feasible segmentations in $l(\mathbf{m})$, i.e. the identification of the *most violated constraint.* In this chapter we investigate how this can be done by *exhaustive search* on a subset of the data as large as possible. We explain our approach to the efficient enumeration of segmentations by means of 2D grids, but the findings likewise apply to 3D supervoxels, which are used in the experiments.

Table 5.1 shows the ratio between the number of true segmentations S and the number of possible configurations for different grid sizes: SP are the number of candidates that would have to be enumerated using the primal representation, SD for the dual representation and SDI for an improved dual enumeration described below. Apparently, S/SDI achieves the best ratio, i.e. the least work is wasted for solutions that are ultimately rejected.

In the dual representation, when exhaustively enumerating all binary vectors $\boldsymbol{y}$, an inconsistent configuration ($\boldsymbol{y} \notin$ MC) can be identified by connected component labeling, which is expensive. Fortunately, many inconsistent configurations (those which contain one or more “dangling” surfaces, Figure 5.2b) can be identified more simply.

Efficient and unambigous enumeration is greatly facilitated by a cell complex data structure (Chapter 3.4.2). It represents entities of different dimensionality simultaneously along with their bounding relations: supervoxels $c_3 \in C_3$ are bounded by joint-faces between pairs of adjacent supervoxels $c_2 \in C_2$ which in turn are bounded by joint-lines $c_1 \in C_1$ between adjacent faces. Our 2D illustrations should be understood as slices through 3D data: surfaces appear as lines and joint-lines between surfaces appear as points (Figure 5.2a). A *line* $c_1 \in C_1$ is formed where multiple faces $c_2 \in C_2$ meet. Using the Γ-neighborhood, this is written as $c_2 \in \Gamma(c_1)$. For example, in Figure 5.2, ‘green line’ $\in \Gamma$(‘green dot’).

Table 5.1: For a $n \times m$ pixel patch, the ratio between the number of feasible segmentations S and the number of candidate configurations (SP, SD or SDI) depends on the enumeration technique.

$n \times m$	$2^{\lvert C_2 \rvert}$	S/SP	S/SD	S/SDI
2×2	16	0.1875	0.7500	1.0000
2×3	128	0.0723	0.5781	0.7708
3×3	4’096	0.0219	0.3501	0.6224
4×3	131’072	0.0066	0.2119	0.5024
4×4	16’777’216	0.0016	0.1008	0.4249
5×4	2’147’483’648	0.0004	0.0480	0.2695

For efficient enumeration of segmentations, we first find a preferably maximal set of lines C_1', such that no Γ-neighborhoods overlap:

$$\max_{C_1' \in \mathcal{P}(C_1)} \left\{ \, |C_1'| \, \right\} \quad \text{such that} \quad \bigcap_{c \in C_1'} \Gamma(c) = \emptyset \ . \tag{5.8}$$

In Figure 5.2c, such a maximal set C_1' consists of all the colored dots. In order for $\boldsymbol{y} \in \mathrm{MC}$ to hold, a necessary condition is that all surfaces $c_2 \in \Gamma(c_1')$ must be assigned a locally consistent labeling. From the cycle inequalities (4.5) follows:

$$\begin{aligned} \forall c \in C_2 : \quad & \left\{ c_1^2, c_2^2, c_3^2, c_4^2 \right\} := \Gamma(c) \text{ with} \\ & (y_{c_1^2}, y_{c_2^2}, y_{c_3^2}, y_{c_4^2}) \notin \left\{ (1,0,0,0), \ \ (0,1,0,0), \ \ (0,0,1,0), \ \ (0,0,0,1) \right\} \ . \end{aligned} \tag{5.9}$$

The configurations in which only one face in $\Gamma(c)$ is assigned $y_i = 1$ are already *locally inconsistent* (a “dangling” surface, Figure 5.2b). Consequently, any $\boldsymbol{y}$ in which these configurations occur for one or more of the sets $\Gamma(c_1')$ with $c_1' \in C_1'$ can be excluded from the enumeration.

We term this algorithm *improved dual enumeration* or *DI*. After having excluded many locally inconsistent segmentations, we only need the expensive connected components check on a tiny fraction of the actual number of segmentations. This way we can handle larger subsets.

In a supervoxel segmentation, due to the voxel grid topology, either three or four surfaces meet to form a line c_1. We first find an approximately maximal set C_1' by random sampling. Then *candidate* configurations $\boldsymbol{y} \in \{0,1\}^{|C_2|}$ that are locally consistent are enumerated. For each locally consistent candidate, a connected component labeling is then performed in order to check if it is also globally consistent.

In our C++ implementation, each segmentation $\boldsymbol{y}$ with $|C_2| < 32$ is stored efficiently as a 4-byte integer. The enumeration via C_1' is implemented via fast bitwise operations.

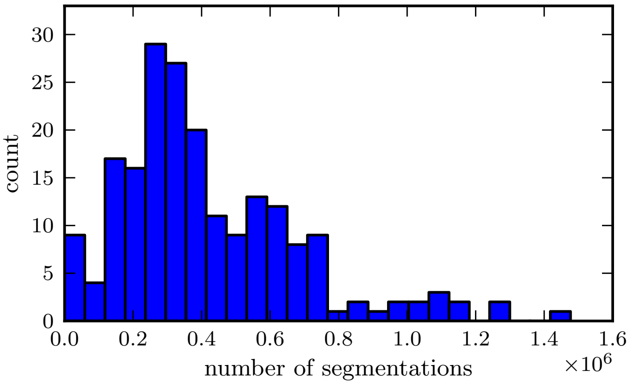


Figure 5.3: Histogram of the number of (unique) segmentations for the 200 training samples from the mouse dataset. These sizes are still sufficiently small for storage and exhaustive search.

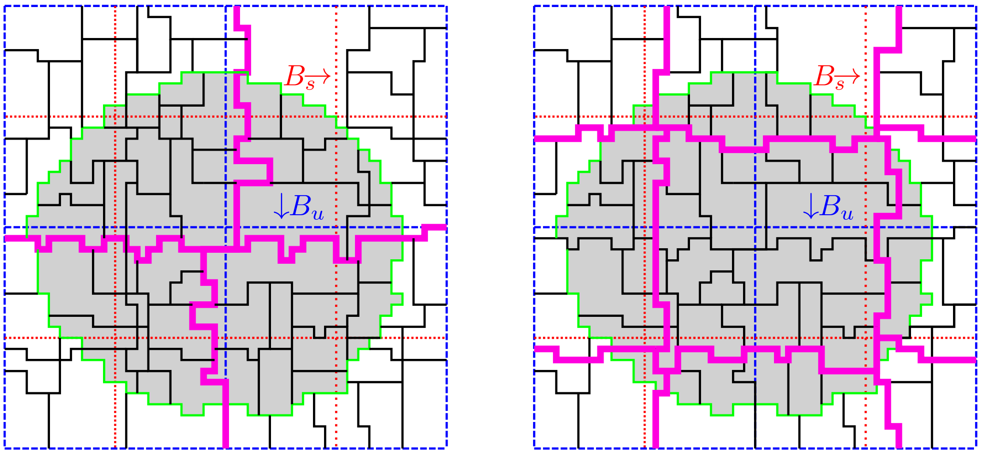
5.3 Structured Prediction

As the structured prediction problem (5.4) is NP-hard [16], in practice a solution cannot be found if the number of variables is too large or the weights make for a "difficult" problem (Figure 5.5 left). Given weights $\boldsymbol{w}$ obtained with structured learning, the global optimum of the multicut objective is found using the integer linear programming approach developed in Chapter 4. In our experiments, only problems with about 10^5 variables could be optimized in reasonable time, while we would like to run structured prediction on problems which are several orders of magnitude larger (note that the datasets used in this chapter seem to be more difficult than those in Chapter 4). We therefore propose a hierarchical blockwise optimization scheme.

Blocks. Given an oversegmentation $\mathcal{C}$ and weights $\boldsymbol{w}$ we divide the problem into *subproblems* via blocks:

- $\mathcal{B}_u$ (*unshifted blocking*) is a partitioning of the volume into rectangular blocks with shape $\mathbf{L} = (L_1, L_2, L_3)$,
- $\mathcal{B}_s$ (*shifted blocking*) is a blocking which is shifted by $\mathbf{L}/2$.

For the chosen blocking $\mathcal{B}$, each supervoxel is uniquely assigned to the block of smallest scan-order index $b \in \mathcal{B}$ with which it intersects (creating a supervoxelized blocking of the volume, Figure 5.4 right). Let $C_3(b)$ be the set of supervoxels assigned to block b and $C_2(b)$ the set of all surfaces which bound at least one of these supervoxels. $\partial C_2(b) \subset C_2(b)$ forms b's surface.



(a) (b)

Figure 5.4: Dividing a supervoxel segmentation into blocks.
Panel (a) shows $\mathcal{B}_u$, panel (b) shows $\mathcal{B}_s$. Dashed blue lines indicate the unshifted blocking $\mathcal{B}_u$ of the pixels, dashed red lines the shifted blocking $\mathcal{B}_s$. The set ∂C_2 of surfaces separating adjacent blocks is shown with bold magenta lines.

Blockwise Optimization. For hierarchy level 1, an initial block shape $\mathbf{L}$ is chosen. We start with an unshifted blocking $\mathcal{B}_u$.

For each block b, the optimization problem (5.4) is solved, subject to the additional constraints that surfaces $c_2 \notin C_2(b)$ are assigned zero and $c_2 \in \partial C_2(b)$ are assigned one. Effectively, this reduces the problem size to $|C_2(b)|$ variables. Results from all blocks are combined via binary OR to yield a result $\boldsymbol{y}(\mathcal{B}_u)$. As each subproblem yields a consistent solution, and all surfaces separating the blocks have $y_i = 1$, the entire state $\boldsymbol{y}(\mathcal{B}_u)$ is consistent. The procedure is repeated with a *shifted blocking* $\mathcal{B}_s$, yielding $\boldsymbol{y}(\mathcal{B}_s)$.

A vector $\boldsymbol{y}^{(1)}$ for the first hierarchy level is obtained by binary OR of $\boldsymbol{y}(\mathcal{B}_u)$ and $\boldsymbol{y}(\mathcal{B}_s)$. As an intersection of two segmentations, $\boldsymbol{y}^{(1)} \in \mathrm{MC}$. Combining $\mathcal{B}_u$ and $\mathcal{B}_s$ considerably reduces boundary artifacts, see Figures 5.5 and 5.6.

Hierarchical Optimization. In a hard decision, all variables $y_i = 0$ are removed from the problem. We then obtain a new cell complex $\mathcal{C}'$ (hierarchy level 2) by a connected component labeling of the 1,2, and 3-cells in $\mathcal{C}$, and a bijection $M(\mathcal{C}) \to \mathcal{C}'$ mapping between the entities of level 1 and 2. $\mathcal{C}'$ consists of fewer, but bigger lines, surfaces and segments. New weights $\boldsymbol{w}(c')$ for $c' \in \mathcal{C}'$ are computed by $w_{c'} = \sum_{c \in M^{-1}(c')} w_c$. Finally, the block size is increased, and the above scheme is applied to $\mathcal{C}'$. In this way, a hierarchy of N levels is created. The final optimization uses no blocking. This algorithm can be parallelized and performs well empirically (Section 5.4).

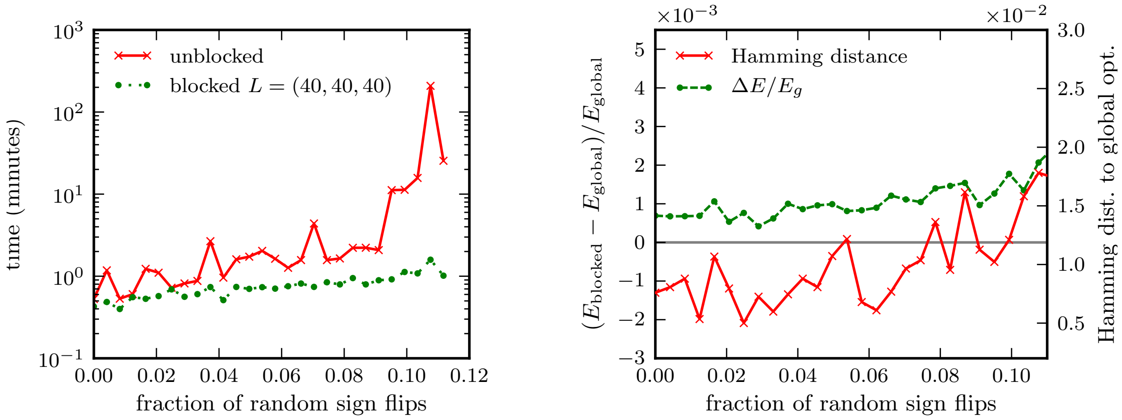
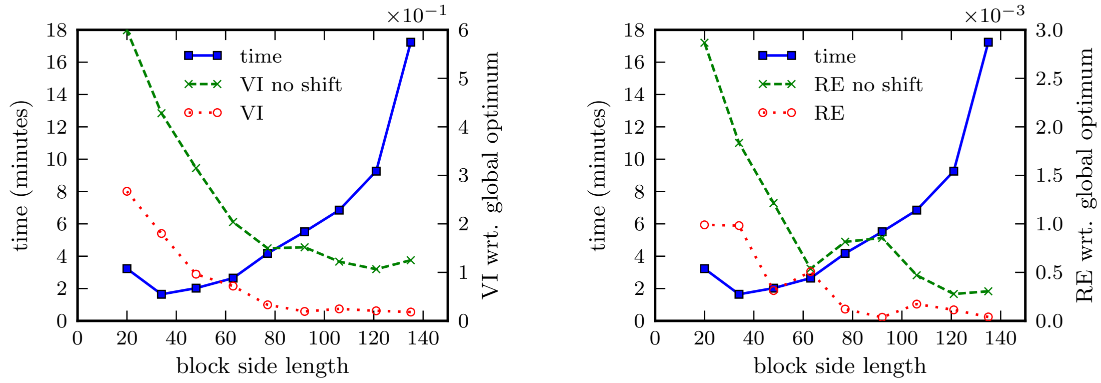


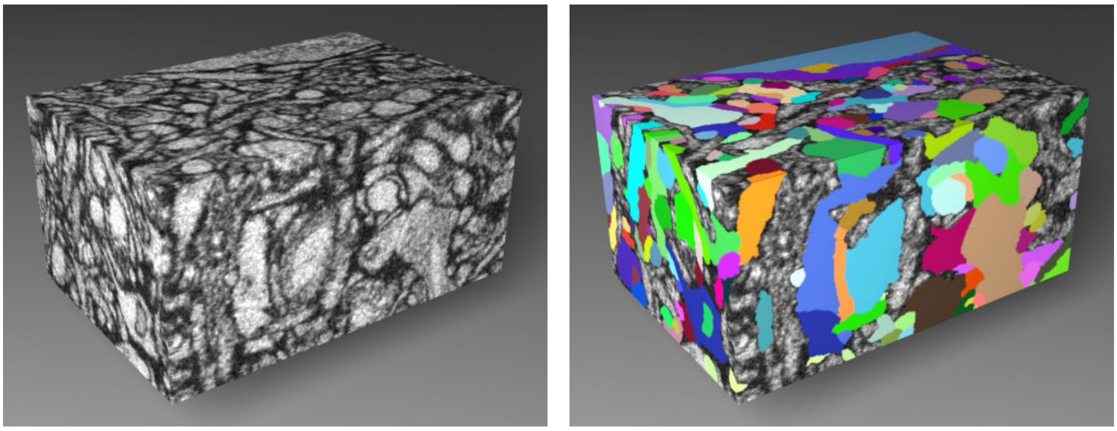
Figure 5.5: *Left:* Increasing sign noise on the weights $\boldsymbol{w}$ simulates "difficult" weights. With increasing noise, the runtime for obtaining a globally optimal solution explodes. However, if the hierarchical blocked algorithm is used, overall runtime is substantially reduced. *Right:* the quality of the approximation degrades very slowly, as measured by the energy gap and Hamming distance relative to the optimal solution.



(a) Variation of Information

(b) Rand Error

Figure 5.6: Effect of the block side length on the quality of the segmentation. For each parameter, blockwise multicut optimization was run either with no block shift (green curve) or with block shift (red curve). Both in terms of VI and RE, using the shifted blocks additionally improves performance, as does a larger block size.



(a) raw data

(b) gold standard segmentation

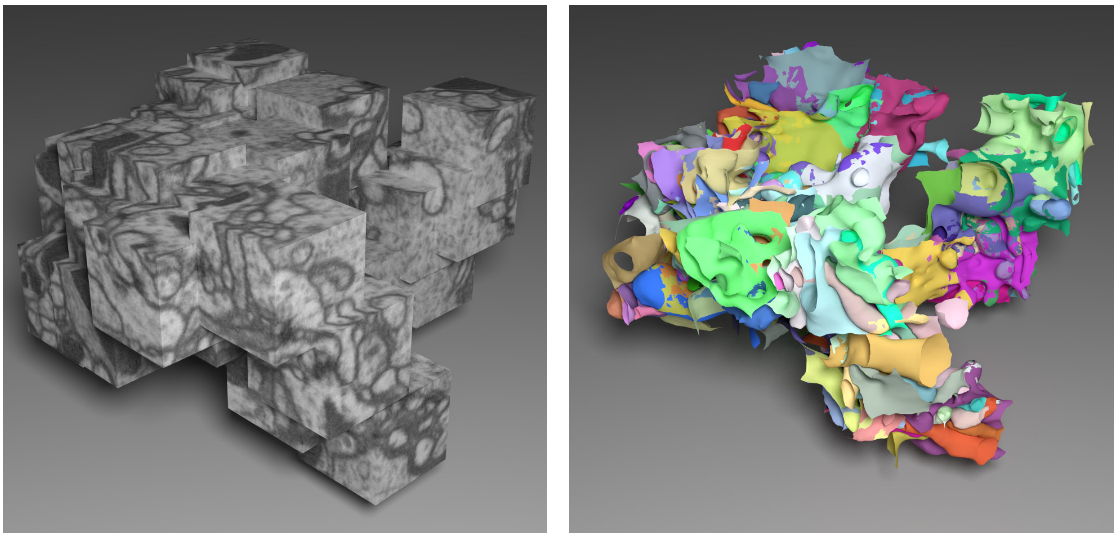
Figure 5.7: For the *mouse dataset*, a block of 200 × 300 × 150 voxels (a) was partially annotated using CARVING to obtain a gold standard segmentation (b).

5.4 Experiments

The first dataset (Figures 5.9, top left and 5.7), shows part of adult mouse cerebral cortex at 20 nm^3 voxel size (SBFSEM imaging). The sample was prepared to preserve extracellular space and to suppress intracellular organelle contrast. A 200 × 300 × 150 subset was segmented into 185 segments as gold standard. The second dataset (Figures 5.9, bottom left and 5.8), shows a part of Drosophila medulla (voxel size 10 nm^3, FIBSEM imaging). Here, organelles such as mitochondria are also stained. We would like to thank Harald Hess and C. Shan Xu at Janelia Farm Howard Hughes Medical Institute for providing this dataset. 49 blocks of 100^3 have been partially segmented (covering about 2/3 of each volume) as gold standard. Note that both datasets have isotropic voxel size and are therefore amenable to a true 3D approach, as opposed to thick slice data, such as from TEM imaging.

A watershed transform yields an oversegmentation. Then, the voxel ground-truth is projected onto these supervoxels to create ground-truth for $c_2 \in C_2$. The feature vector $\boldsymbol{\alpha}^{(i)}$ describes a surface c_2 which separates two adjacent supervoxels c_3^A and c_3^B. Given different voxel features (smoothed data, 1$^{\text{st}}$ and 2$^{\text{nd}}$ derivative filters), several statistics over the voxels near the surface are computed. Additional features include topological and geometric features such as $\text{size}(c_2)$, $|\,\text{size}(c_3^A) - \text{size}(c_3^B)|$, ratio between circumference to area of c_2 and number of adjacent surfaces.

The gold standard of each dataset is divided into training and test blocks. We sample connected components of $|C_2| \approx 27$ surfaces and their gold standard labeling to create training samples $(\boldsymbol{x}^n, \boldsymbol{y}^n)$, $n = 1 \ldots 200$. In order to maximize the number of supervoxels involved in each sample, we only consider, for each sample, surfaces that intersect the



(a) raw data of validation set

(b) gold standard segmentation

Figure 5.8: For the *Drosophila dataset*, 50 cubes of 200^3 voxels each (a) were partially annotated as gold standard segmentation (b).

same axis-aligned plane. To capture the asymmetric distribution of edges $y_i = 0$ versus $y_1 = 1$, the sampling algorithm attempts to obtain samples with a ratio of $p(y_i = 1)/p(y_i = 0)$ estimated from the gold standard segmentation.

Enumerating all (hundreds of thousands, see Figure 5.4) segmentations for one sample takes only about a second, thanks to the efficient enumeration from Section 5.2. About 3 seconds are needed to precompute different loss functions (RE, VI, Hamming). Both the list of segmentations (compressed to 4 bytes per segmentation) and the losses are stored on disk to be reused during structured learning. Figure 5.4 shows that, for most samples, we have to consider about half a million possible segmentations. Finally the vector $\phi(\boldsymbol{x}, \boldsymbol{y})$ is precomputed for all training samples. When the separation oracle asks for the most violated constraint during cutting-plane training, we only need to compute a dot product of the current weight vector and ϕ for each segmentation and look up the segmentation's loss w.r.t. the gold standard. In addition, it parallelizes easily. Training usually needs about 200 iterations for convergence on 8 CPUs and takes about 20 minutes.

For prediction, the trained model is optimized on several blocks taken from the test portion of each dataset. We compare our results to unstructured methods. Different binary classifiers are trained using a training set which consists of the union of all surfaces involved in any training sample and their gold standard label. For Random Forest and Regression Forest classifiers, the probabilistic output is transformed into weights

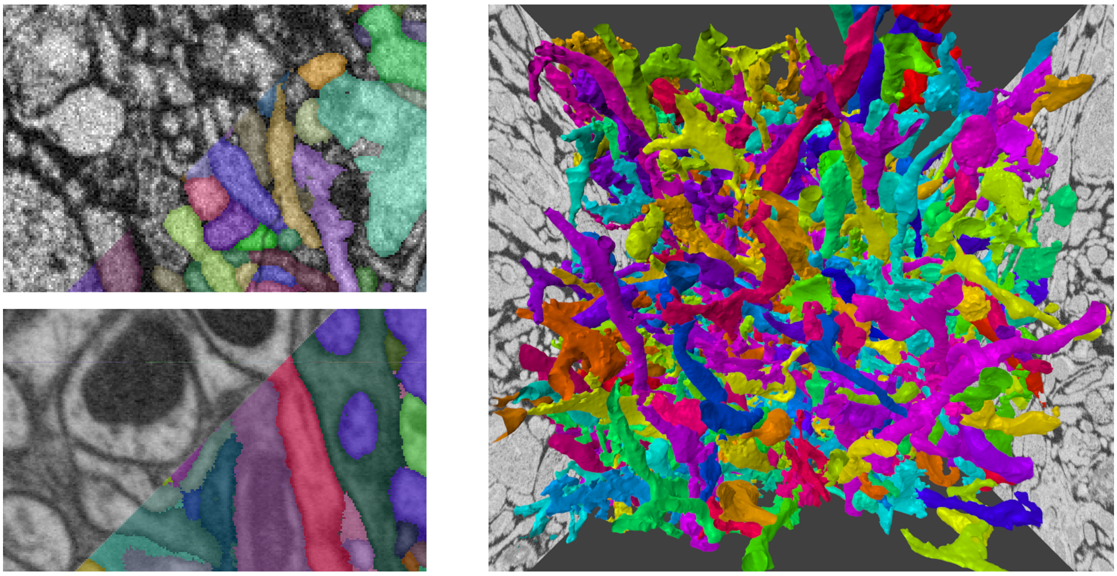
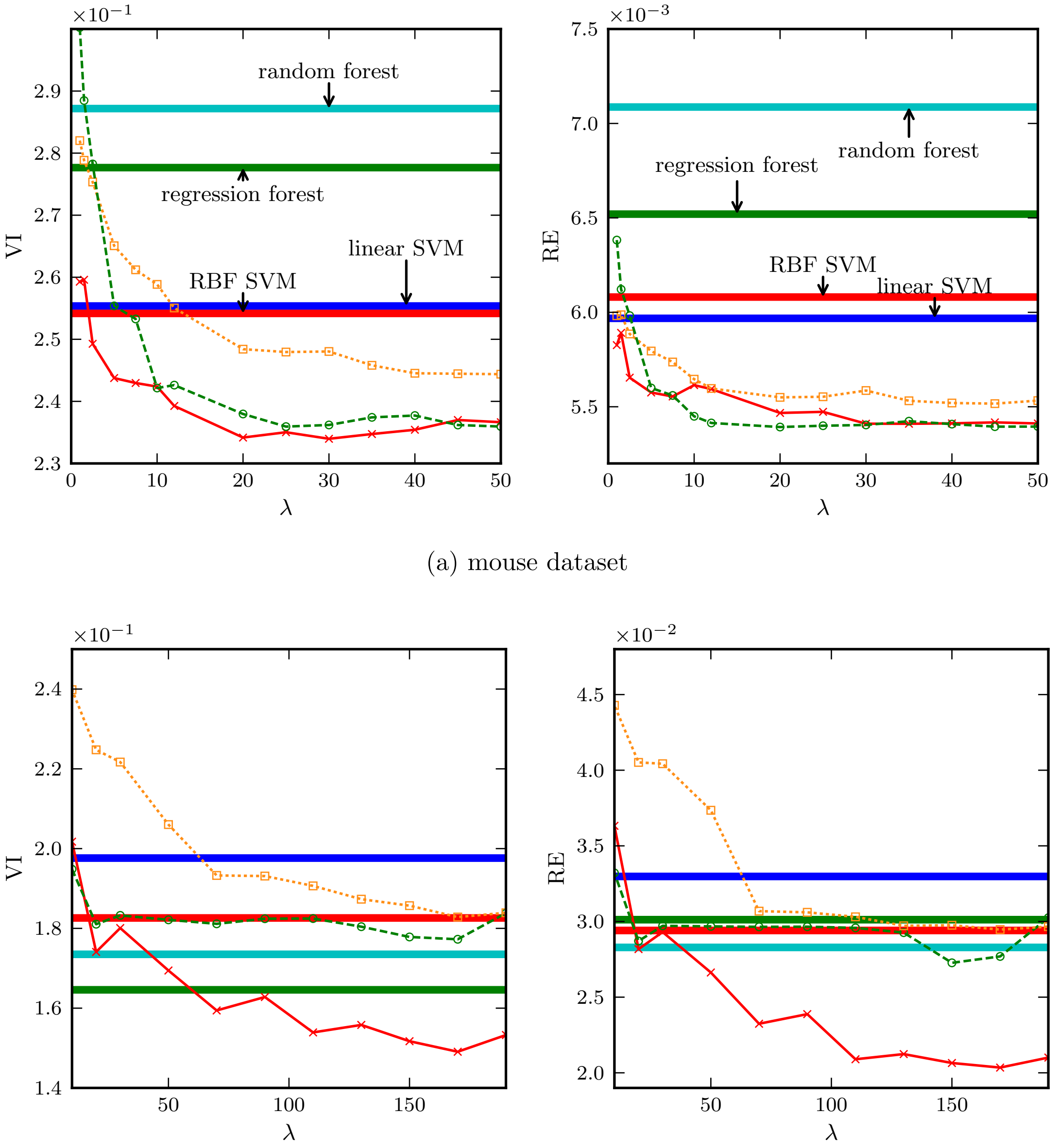


Figure 5.9: *Left*: Mouse dataset (above) and, drosophila dataset (below), with gold standard segmentations. *Right*: Segmentation result (showing only 900 objects) with structured prediction on 10^6 supervoxels with 10^7 variables, using blockwise hierarchical optimization.

$\boldsymbol{w}$ via (5.5); for linear SVM and RBF SVM classifiers, the weights are taken to be the distance from the margin. Hyperparameters (regression forest: tree depth; linear SVM: regularization strength λ; RBF SVM: γ, λ) are optimized via cross-validation.

Figure 5.10 shows VI and RE, averaged over multiple test blocks as a function of the regularization parameter λ with respect to the performance of unstructured methods. For the mouse dataset our approach is able to outperform both an unstructured learning of $\boldsymbol{w}$ as well as structured learning with decomposable Hamming loss, while being insensitive to the exact choice of hyperparameter λ. Choosing either VI and RE loss during learning yields similar performance (as measured by VI or RE) with respect to the gold standard. On the drosophila dataset, using VI for learning improves over unstructured methods; interestingly the RE loss does no better than unstructured learning. Note also that the relative performance of unstructured SVM and Random Forest is inverted between both datasets, emphasizing the distinctness of the two problems.



(a) mouse dataset

(b) drosophila dataset

Figure 5.10: VI and RE (lower is better) as a function of the regularization parameter λ. Results have been averaged over multiple blocks. For struct-SVM, using the structured loss functions VI "×" or RE "○" gives substantially better results than the unstructured Hamming loss "□".

5.5 Conclusion

This chapter addressed the problem of learning a supervised segmentation algorithm with arbitrary loss functions. A structured support vector machine has been used to learn weights for correlation clustering on an edge-weighted region adjacency graph.

Our first contribution is an efficient exhaustive enumeration of segmentations for small subsets of the training data for loss-augmented prediction. This allows to train on the same structured loss functions (RE or VI) as are used for evaluation of the segmentation quality. We find that, for a neighborhood of $|C_2| \approx 27$ and the linear struct-SVM classifier, structured learning with a structured loss function can beat more complex, but unstructured classifiers in two different microscopic modalities. However, the final segmentation still fails to match the quality of a human expert. Our second contribution, a hierarchical blockwise scheme for structured prediction, enables us to analyze a 1000^3 dataset involving over 10 million variables, which was broken up initially into 3,000 blocks. The complete hierarchical blockwise segmentation takes about a day, but can be easily parallelized over the independent subproblems. Figure 5.9, right, shows 900 objects from the final partitioning.

Chapter 6

Cut, Glue & Cut – Approximate Multicut Segmentation

Recently, unsupervised image segmentation has become increasingly popular. Starting from a superpixel segmentation, an edge-weighted region adjacency graph is constructed. Amongst all segmentations of the graph, the one which best conforms to the given image evidence – as measured by the sum of cut edge weights – is chosen (Chapters 4 and 5).

Since this problem is NP-hard, we propose a *new approximate solver* based on the move-making paradigm: first, the graph is recursively partitioned into small regions (cut phase). Then, for any two adjacent regions, we consider alternative cuts of these two regions defining possible moves (glue & cut phase). For planar problems, the optimal move can be found, whereas for non-planar problems, efficient approximations exist.

We evaluate our algorithm on published and new benchmark datasets, which we make available online. The proposed algorithm finds segmentations that, as measured by a loss function, are as close to the ground-truth as the global optimum found by exact solvers. It does so significantly faster than existing approximate methods, which is important for large-scale problems as proposed in this thesis for segmentation of neurites in electron microscopy volume images.

This chapter is based on [20] (at the time of writing still under review), also see page 145.

6.1 Introduction

Segmentation is an important problem in computer vision as a first step towards understanding an image. Many algorithms start with an over-segmentation into superpixels, which are then clustered into "perceptually meaningful" regions (cf. Section 2.4.5). Usually, the number of regions into which the image should be partioned is not known beforehand.

Recently, the multicut formulation [39] (sometimes called *correlation clustering*, [16]) has become increasingly popular for unsupervised image segmentation. Given an edge-weighted region adjacency graph, the problem is to find the segmentation which minimizes the cost of the cut edges. Such an approach has been shown to yield state-of-the-art results on the Berkeley Segmentation Dataset [5, 159, 4].

Unfortunately, solving the multicut problem is in general NP-hard. Any exact solver will therefore be plagued by scalability issues. However, segmentation of images with ever finer superpixel partitionings and of large volume images in computational neuroscience (Chapters 4 and 5, [10, 98]) demand solutions to large scale multicut problems. In this regime of large-scale problems, existing solvers either fail to find any solution after a reasonable time, rely on suitable edge weights (such that the problem decomposes naturally into independent subproblems) or yield an approximate solution possibly far away from the optimum.

Contribution.

(i) This chapter provides a new perspective on solving the multicut problem by *local* move-making methods together with

(ii) a new approximate multicut solver called *Cut, Glue & Cut (CGC)*. Furthermore,

(iii) our method avoids re-solving the same moves by tracking their "dirtiness", which decreases the runtime.

(iv) An extensive evaluation on existing and new benchmark datasets shows that CGC provides results close to optimality and equal application performance significantly faster than all its competitors, both exact and approximative methods, and gives new insights concerning the applicability of competing methods.

(v) Our C++ implementation of CGC is available online.

Organization. In Section 6.2 we review two common formulations of the multicut problem. Next, related work is discussed in Section 6.3 and the max-cut problem in Section 6.4. Based on a general formulation of local partition moves on segmentations in Section 6.5, we describe our new *Cut, Glue & Cut algorithm* in Section 6.6. Extensive experiments on benchmark datasets are presented in Section 6.7. Finally, we discuss our findings.

6.2 Problem Formulation

Let $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{w})$ be a weighted region adjacency graph of nodes $\mathcal{V}$, representing superpixels, and edges $\mathcal{E}$. The function $\boldsymbol{w} : \mathcal{E} \to \mathbb{R}$ assigns a weight to each edge. A positive weight expresses the desire that two adjacent nodes should be merged, whereas a negative weight indicates that these nodes should be separated into two different regions. Chapter 4 has described this model in detail.

A *subgraph* $G_A = \{A, \mathcal{E}_A, \boldsymbol{w}\}$ consists of nodes $A \subseteq \mathcal{V}$ and edges $\mathcal{E}_A = \mathcal{E} \cap (A \times A)$. In the following, we will call a connected component of nodes $\mathcal{V}$ a *region* $R \subseteq \mathcal{V}$. Each node i is assigned a label $l_i \in \{0, \dots, |V| - 1\}$. Using the indicator function $\delta(\cdot)$, the *multicut problem* can be written as a node labeling problem [14]:

$$\underset{\boldsymbol{l}}{\operatorname{argmin}} \left\{ \sum_{e=(i,j)\in\mathcal{E}} w_e \cdot \delta(l_i \neq l_j) \right\} \quad , \qquad \text{(multicut – node labeling, 6.1)}$$

where $\delta(a) = 1$ if a is true and 0 else. While (6.1) looks like a common Potts energy without unary terms, there are some crucial differences:

- Any weight w_e with $e \in \mathcal{E}$ can be positive or negative.
- The lack of any unary data terms renders the problem much harder and introduces ambiguity.
- The label space is large. In general we have to set $|l_i| = |\mathcal{V}|$ to allow for the solution where every supervoxel becomes its own region without having to solve the graph coloring problem implicitly. For planar graphs, it would be sufficient that $l_i \in \{0, ..., 3\}$ because any planar graph is 4-colorable [11].

An alternative formulation of problem (6.1) is in terms of binary edge indicator variables $\boldsymbol{y} \in \{0, 1\}^{|\mathcal{E}|}$:

$$\underset{\boldsymbol{y}}{\operatorname{argmin}} \underbrace{\left\{ \sum_{e=(i,j)\in\mathcal{E}} w_e \cdot y_e \right\}}_{\text{CUT}_G(\boldsymbol{y})} \text{ s.t. } \boldsymbol{y} \in \text{MC}_G \ . \qquad \text{(multicut – edge labeling, 6.2)}$$

A segmentation $\boldsymbol{y}$ has an associated *energy* which is denoted by $\text{CUT}_G(\boldsymbol{y})$. Here, MC_G is the set of all multicut constraints [39] for graph G forming the so-called *multicut polytope* (see Equation 4.7). These constraints ensure that labeling $\boldsymbol{y}$ is *consistent* (Section 4.3.2): if $y_{ij} = 1$ then nodes i and j should be in separate regions even after connected component labeling. Intuitively, dangling line segments are forbidden in two dimensional images, and punctured walls or faces are ruled out in three dimensional images.

While any segmentation $\bigcup_i R_i \equiv \mathcal{V}$ is represented by exactly one edge labeling $\boldsymbol{y}$ with $\boldsymbol{y} \in \text{MC}_G$, many node labelings $\boldsymbol{l}$ represent the same segmentation. Therefore, a multicut representation (6.2) is preferable to avoid large multiplicities of local optima. However, an efficient handling of the multicut polytope is essential.

6.3 Related Work

Applying state-of-the-art solvers for the labeling problem (6.1) is challenging since any permutation of the labelings transforms an optimal solution into another optimal solution, see [80] for more details and a recent review.

In general, $\boldsymbol{y} \in \mathrm{MC}_G$ can be enforced by an exponential number of constraints [39], but in practice – for a given objective function – a small subset of those are sufficient. Therefore, a major branch of research has focused on cutting plane approaches, either by solving a relaxation of (6.2) by a sequence of linear programs [86, 87, 80, 53] or by solving problem (6.2) exactly by a sequence of integer linear programs [79, 5, 10, 80, 3, 4].

For the latter, one can avoid the cutting plane procedure by lifting to the fully connected graph at the expense of problem size [3]. For many image segmentation problems, however, the size of the complete graph is prohibitively large.

Alush and Goldberger [3, 4] suggest to use partial optimality in order to decompose the problem into its positively connected components, defined as the connected components of the graph $G' = (\mathcal{V}, \mathcal{E}')$ with $\mathcal{E}' = \{e \in \mathcal{E} | w_e > 0\}$. These problems are then separately solved to global optimality. While this recovers the true solution, it depends on the weights $\boldsymbol{w}$ whether a significant reduction of the problem is possible.

Yarkony et al. [159] suggest a method, called PlanarCC, which solves a relaxed linear program by iteratively solving weighted two-coloring problems. Because these subproblems require planarity to be tractable, PlanarCC is restricted to problems with planar structure.

Another branch of research [85, 162, 14] has focused on specialized move-making methods that take the degeneracy of the solution, due to label permutations in (6.1), into account. Move-making algorithms maintain a valid solution throughout the optimization procedure (a segmentation defined via a node labeling $\boldsymbol{l}$ or, equivalently, by an edge labeling $\boldsymbol{y} \in \mathrm{MC}_G$). In each step, a set of possible moves transforming the current segmentation is considered, and the move which realizes the maximal energy decrease is chosen. Therefore, the energy decreases monotonically until a (local) optimum is found. The Kerninghan-Lin method [85], used in circuit-layout design, applies a sequence of greedy local moves to neighboring regions. Bagon [14] recently proposed an extension to the α-expansion move-making algorithm for solving multicut problems, which handles the large label space and label ambiguity in (6.1) by using dynamic label sets and additional label fixing, respectively.

6.4 Max-Cut

A basic subproblem which has to be solved in the Kerninghan-Lin method [85], Expand & Explore [14], PlanarCC [159] and also our method is the max-cut problem [50] (also known as weighted 2-coloring).

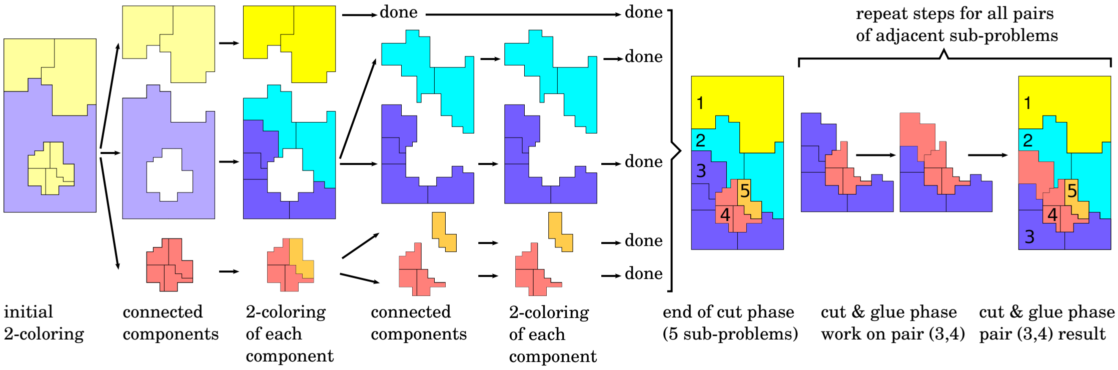

done
done
done
done
done
done
repeat steps for all pairs of adjacent sub-problems
1
2
3
4
5
1
2
5
4
3
initial 2-coloring
connected components
2-coloring of each component
connected components
2-coloring of each component
end of cut phase (5 sub-problems)
cut & glue phase work on pair (3,4)
cut & glue phase pair (3,4) result

Figure 6.1: *Left*: Illustration of the *cut phase*. The steps are visualized as a tree. First, a weighted 2-coloring problem (6.3) is solved on the initial superpixel adjacency graph of an image. A connected-component labeling yields the child nodes. For each node, the sequence of solving (6.3) and connected component labeling are repeated. Whenever a weighted 2-coloring yields only a single component, the energy cannot be decreased any further and the node has no children ("done"). *Right*: Illustration of the *glue&cut phase*. In the illustrated move, the regions labeled 3 and 4 are first merged and then a new, better partition is sought, which leads to a better global segmentation. This step is repeated until no more improvement is possible.

The *max-cut problem*[1] is the specialization of the node labeling problem (6.1) for the case of *binary* labels $l_i = \{0, 1\}$, given by

$$\underset{\boldsymbol{l} \in \{0,1\}^{|\mathcal{E}|}}{\operatorname{argmin}} \left\{ \sum_{e=(i,j)\in\mathcal{E}} w_e \cdot \delta(l_i \neq l_j) \right\} . \qquad \text{(max cut, 6.3)}$$

This should not be confused with the min-cut problem, which refers to sub-modular "graph cut problems" with non-negative edge weights $\boldsymbol{w}$, for which a max-flow problem can be formulated [93]. Intuitively, restricting the label space to two labels (0 and 1) simplifies the problem. For planar graphs, (6.3) can be solved by the Blossom algorithm [134] in polynomial time. In general this problem is still NP-hard. In response, Kernighan and Lin have suggested a greedy method for (6.3) used in the KL algorithm [85]. Recently, Bagon [14] suggested to use QPBO-I [129] for (6.3). While this LP relaxation often gives non-integral solutions for many nodes, the improving extension (-I) [129] iteratively fixes such nodes and therefore also deals with the ambiguity of the problem.

In our experiments, we found that using QPBO-I performs better than the greedy method used in the KL algorithm, but slightly worse than optimal solvers (Section 6.7).

[1] For reasons of consistency, we consider wlog. minimization instead of maximization.

6.5 Partition Moves

In this section we define a class of moves which can be used to iteratively improve segmentations. The *Partition Move Theorem* shows that under some technical conditions, the improvement of the local sub-problem leads to monotonous improvement of the global energy.

Theorem (Partition Moves)
Let $\boldsymbol{y}^t \in MC_G$ represent the segmentation of G at step t. Local partition moves are defined over a subset $S \subset \mathcal{V}$ for which all edges crossing its borders have to be labeled one: $\forall\, e \in \mathcal{E} \cap (S \times (\mathcal{V} \setminus S)) : y_e^t = 1$.

Let G_S be the respective subgraph of G and $\hat{\boldsymbol{y}} \in MC_{G_S}$ represent a given valid segmentation of S. For the combined segmentation

$$y_e^{t+1} := \begin{cases} \hat{y}_e & \text{if } e \in \mathcal{E}_S \\ y_e^t & \text{else} \end{cases} \quad , \tag{6.4}$$

the following holds:

(i) $\boldsymbol{y}^{t+1} \in MC_G$,

(ii) $CUT_{G_S}(\hat{\boldsymbol{y}}_{\mathcal{E}_S}) \leq CUT_{G_S}(\boldsymbol{y}^t_{\mathcal{E}_S}) \Rightarrow CUT_G(\boldsymbol{y}^{t+1}) \leq CUT_G(\boldsymbol{y}^t)$.

Proof.

(i) Let us extend the local segmentation $\hat{\boldsymbol{y}}$ to the global graph G with

$$\hat{\boldsymbol{y}}_G := \begin{cases} \hat{y}_e & \forall e \in \mathcal{E}_S \\ 1 & \forall e \in \mathcal{E} \cap S \times (\mathcal{V} \setminus S) \\ 0 & \forall e \in \mathcal{E} \cap (V \setminus S) \times (\mathcal{V} \setminus S) \ . \end{cases} \tag{6.5}$$

Since (a) $\hat{\boldsymbol{y}}_G \in \mathrm{MC}_G$, (b) $\boldsymbol{y}^{t+1} = \mathrm{OR}(\hat{\boldsymbol{y}}_G, \boldsymbol{y}^t)$, and (c) the set of multicuts is closed under the OR operation, (i) holds.

(ii) For $\mathrm{CUT}_{G_S}(\hat{\boldsymbol{y}}_{\mathcal{E}_S}) \leq \mathrm{CUT}_{G_S}(\boldsymbol{y}^t_{\mathcal{E}_S})$, the energy after the move can be written as:

$$\begin{aligned} \mathrm{CUT}_G(\boldsymbol{y}^{t+1}) &= \sum_{e \in \mathcal{E}_S} w_e \cdot y_e^{t+1} + \sum_{e \in \mathcal{E} \setminus \mathcal{E}_S} w_e \cdot y_e^{t+1} && (6.6a) \\ &\leq \sum_{e \in \mathcal{E}_S} w_e \cdot y_e^{t} + \sum_{e \in \mathcal{E} \setminus \mathcal{E}_S} w_e \cdot y_e^{t+1} && (6.6b) \\ &= \sum_{e \in \mathcal{E}_S} w_e \cdot y_e^{t} + \sum_{e \in \mathcal{E} \setminus \mathcal{E}_S} w_e \cdot y_e^{t} && (6.6c) \\ &= \mathrm{CUT}_G(\boldsymbol{y}^t) \quad \square && (6.6d) \end{aligned}$$

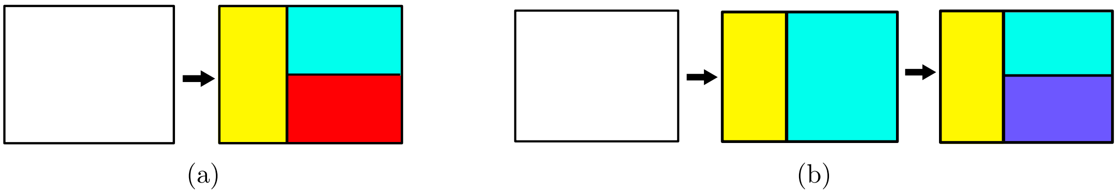


Figure 6.2: Solving the labeling problem (6.1) with more than two colors (a) allows for T-junctions. A sequence of two colorings can often model the same T-junction (b).

To obtain a segmentation $\hat{\boldsymbol{y}}$ on G_S, one can either

(a) solve the multicut problem over G_S (which is smaller than G) or

(b) restrict the subset of possible segmentations, e.g. to all two-colorable segmentations of G_S.

If we solve the problem to optimality and the current segmentation is in the feasible set of the move, it is guaranteed that $\boldsymbol{y}^{t+1}$ never increases the energy. For approximate solutions this is not the case; here the move should only be accepted in case of energy decrease. While (b) restricts the set of possible moves, the problems become easier and in most cases, a sequence of two-coloring moves (b) can generate the same segmentation as one single N-coloring move (a) as sketched in Figure 6.2.

Algorithm 6.1: Cut, Glue & Cut algorithm

```
Input: weighted graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{w})$
Output: approx. solution $Q$ to (6.1) for $G$
$Q^0 \leftarrow$ segmentation of $G$ into positively connected components
for $n = 1 \dots n_{iter}$ do
    $Q^n \leftarrow$ cut_phase$(G, Q^{n-1})$   //Algorithm 6.2
    $Q^n \leftarrow$ glue_cut_phase$(G, Q^n)$   //Algorithm 6.3
    if $Q^n = Q^{n-1}$ then
        exit
    end
end
```

6.6 Cut, Glue & Cut Algorithm

The CGC algorithm (Algorithm 6.1) always maintains a valid partitioning of the graph, which is iteratively improved by *local* moves which act on single or neighboring regions. It works in two distinct phases: (i) recursive *cut phase*, and (ii) *glue & cut phase*.

The cut phase recursively splits regions by solving max-cut problems (6.3), finally yielding a finer, lower energy segmentation driven by the problem's weights. In the glue & cut phase, any two neighboring segments are first merged and then a new cut is sought between them by solving (6.3). These two phases are repeated and the process stops when the energy cannot decrease any further.

Cut Phase

In the *cut phase*, the regions are recursively split into smaller and smaller regions tuned to the weights $\boldsymbol{w}$ until a local optimum is found. As *cut moves* we consider the max-cut solution (6.3) for a single given region R to find a better segmentation of the region $\hat{\boldsymbol{y}} \in \mathrm{MC}_{G_R}$, and accept the local move if the energy could be decreased compared to the previous solution $\mathrm{CUT}_{G_R}(\boldsymbol{y}_R) = 0$. The Partition Move Theorem then guarantees a monotonically decreasing global energy for the segmentation.

The *cut phase* is illustrated in the left part of Figure 6.1, and is given as Algorithm 6.2. All regions are first inserted into a queue Q. While Q is not empty, take a region $R \in Q$, then solve the max-cut problem (6.3) for G_R. Finally, a connected component labeling of the solution defines a new set of regions[2]. If the suggested cut reduces the local energy, we add the induced regions to Q. Otherwise, the energy of region R cannot further be reduced by this type of move; region R is marked as "done" and added to a list Q'.

[2] Note that in general there may be more than two connected regions, cf. Figure 6.1, initial 2-coloring.

[3] Note that by using $\mathcal{E}'$ instead of $\mathcal{E}$, we consider only one representative edge for each boundary between two regions for performance reasons.

Algorithm 6.2: Cut phase

```
Input: weighted graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{w})$, segmentation into regions given as queue $Q$
Output: segmentation into smaller regions $Q'$
$Q' \leftarrow \emptyset$
while $Q \neq \emptyset$ do
    $R \leftarrow$ queue_pop($Q$)
    $\boldsymbol{y}_{\mathcal{E}_R} \leftarrow$ solve max-cut (6.3) for $G_R$
    if $CUT_{G_R}(\boldsymbol{y}_{\mathcal{E}_R}) < 0$ then
        $Q \leftarrow Q \cup$ connected_components($\boldsymbol{y}_{\mathcal{V}_R}$)
    end
    else
        $Q' \leftarrow Q' \cup R$
    end
end
```

Glue & Cut Phase

In this phase, we consider *Glue & Cut moves* of pairs of adjacent segments until the energy cannot be decreased any further, yielding a better solution.

Intuitively, we expect that two common local operations can decrease the energy: (i) merging two segments or (ii) moving the boundary between two adjacent segments. This motivates the following algorithm.

We again apply the Partition Move Theorem from Section 6.5. Given two adjacent regions R_1, R_2 in the current segmentation, we consider the *merged region* $R = R_1 \cup R_2$ (*glue*), and find a new segmentation $\hat{\boldsymbol{y}} \in \mathrm{MC}_{G_R}$ (*cut*) by solving the max-cut problem (6.3). The local move is accepted if the energy $\mathrm{CUT}_{G_R}(\hat{\boldsymbol{y}})$ is lower than the energy of the previous cut. The Partition Move Theorem then guarantees a monotonically decreasing energy for the segmentation.

Formally, (Algorithm 6.3 and Figure 6.1, right), the glue & cut phase starts from a given segmentation Q. We first obtain its edge labeling $\bar{\boldsymbol{y}}$. Initially, all edges $e \in \mathcal{E}$ are marked "dirty". The following is repeated (line 3): for each pair (R_1, R_2) of adjacent regions we pick a single representative edge from the shared boundary $\mathcal{E} \cap (R_1 \times R_2)$ to form the set $\mathcal{E}'$ (line 5). Then, for each such representative $e = (i, j)$ with $y_e = 1$ and which is marked dirty, we find the best glue & cut move as described above. If, after processing all edges $e \in \mathcal{E}'$, no move could be performed, we break out of the outer loop (line 26).

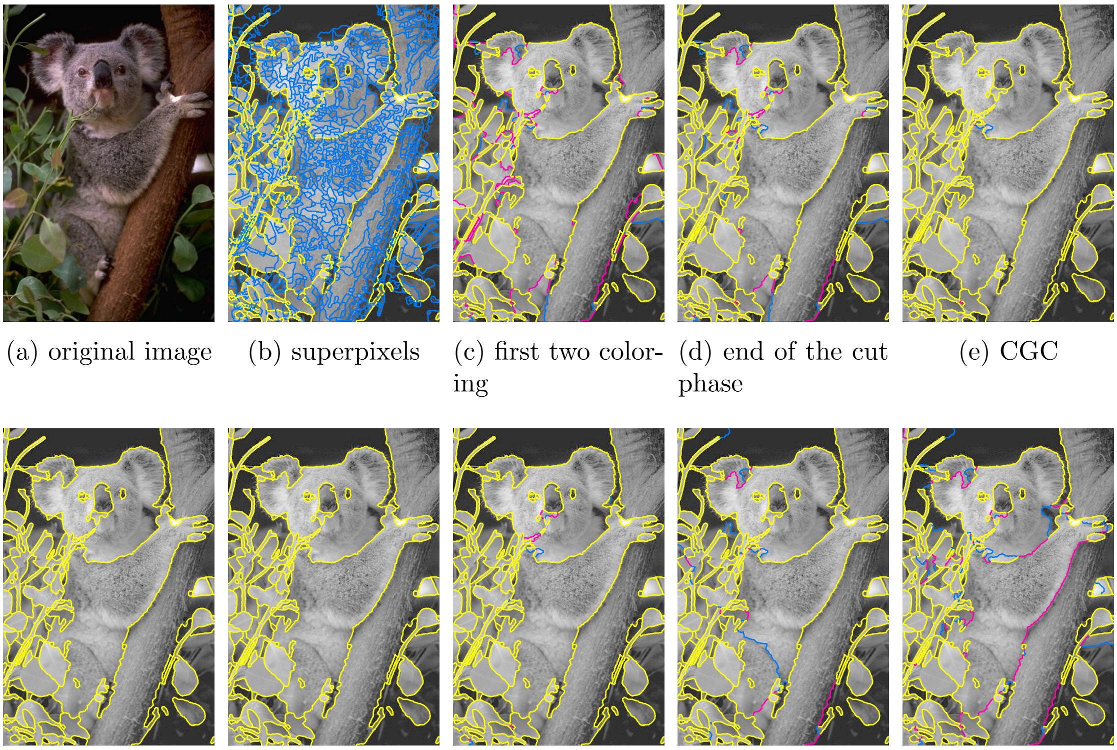
Algorithm 6.3: Glue & Cut phase

```
Input: weighted graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{w})$, segmentation $Q$
Output: improved segmentation $Q$ wrt. (6.1)
mark all edges $e \in \mathcal{E}$ as dirty
$\bar{\boldsymbol{y}} \leftarrow$ edge_labeling($Q$)
while true do
    $c \leftarrow 0$
    $E' \leftarrow \{e \in E \cap (R_1 \times R_2) | R_1, R_2 \in Q$ adjacent$\}$[3]
    for $e = (i, j) \in \mathcal{E}'$ do
        if $\bar{y}_e = 0$ or $e$ is clean then
            continue
        end
        find regions $R_1, R_2 \in Q$, s.t. $i \in R_1$, $j \in R_2$
        $S \leftarrow R_1 \cup R_2$    //glue
        $\boldsymbol{y}_{\mathcal{E}_S} \leftarrow$ solve (6.3) for $G_S$    //cut
        mark edges $\mathcal{E} \cap S \times S$ clean    ($\star$)
        if $CUT_{G_S}(\boldsymbol{y}_{\mathcal{E}_S}) < CUT_{G_S}(\bar{\boldsymbol{y}}_{\mathcal{E}_S})$ then
            $c \leftarrow c + 1$
            mark edges $\mathcal{E} \cap S \times (V \setminus S)$ dirty    ($\star$)
            CC $\leftarrow$ connected_components($\boldsymbol{y}_{\mathcal{E}_S}$)
            if $|CC| > 2$ then
                mark edges $\mathcal{E} \cap (S \times S)$ dirty    ($\star$)
            end
            $Q \leftarrow Q \setminus \{R_1, R_2\} \cup CC$
            $\bar{\boldsymbol{y}} \leftarrow$ edge_labeling($Q$)
        end
    end
    if $c = 0$ then
        break
    end
end
```

Book-keeping of Modified Boundaries

In order to avoid re-solving the same problem multiple times, the algorithm marks edges as "dirty" or "clean". If two adjacent regions are only separated by clean edges, a glue & cut move is *not* considered for this pair. In Algorithm 6.3, statements related to book-keeping are marked with ($\star$).

Imagine an accepted glue & cut move which yields exactly two regions R_1 and R_2. We



(a) original image (b) superpixels (c) first two coloring (d) end of the cut phase (e) CGC

(f) MC-I (global optimum) (g) MC-R (h) PlanarCC (i) Expand & Explore (j) Kerninghan-Lin

Figure 6.3: Comparison of the different multicut algorithms for a model from [5], based on the superpixel segmentation in (b). The colored boundaries indicate true positives (yellow), false negatives (red), false positives (blue) and true negatives (invisible) with respect to the globally optimal solution (f). *Top:* The image (a) is partitioned into superpixels (b). A single two-coloring leads to (c) and the cut phase ends with (d). The final output of the CGC algorithm is (e). *Bottom:* Results of various competitive methods.

mark the boundary between R_1 and R_2, $\mathcal{E} \cap (R_1 \times R_2)$, as clean, since re-solving (6.3) for $R_1 \cup R_2$ would again lead to the same two-coloring into R_1 and R_2. However, if an adjacent pair (R_1, R_3) is chosen subsequently, a glue & cut move could alter region R_1 into R_1'. Therefore, a move between R_1' and R_2 could improve the energy again. To allow this move it is necessary to mark the boundary of $R_1' \cup R_2$ as dirty. Note that for the case where an accepted move yields *more than two regions*, the above would not be correct and the internal edges have to be marked dirty.

6.7 Experiments

We evaluate the performance of our Cut, Glue & Cut algorithm on two different 2D segmentation benchmarks as well as on a new 3D volume segmentation benchmark.

Algorithms

For our CGC algorithm, we consider three variants: CGC-$\overline{B}$ does not do book-keeping, CGC-$\overline{P}$ does not use the globally optimal Blossom solver for planar max-cut problems, but rather uses the approximate QPBO-I. CGC-$\overline{PB}$ does not do book-keeping while using QPBO-I as max-cut solver.

We compare against the following algorithms: Kerninghan-Lin (KL, [85]), Planar Correlation Clustering (PlanarCC, [159]), Expand & Explore [14], LP-based cutting plane method of a relaxed problem (MC-R), integer linear program based cutting plane method (MC-I), which always finds the globally optimal solution. MC-R and MC-I use facet-defining separation procedures and bounding techniques as described in [80]. We use the publicly available C++ implementation in OpenGM 2.1.1 [9] for KL, MC-R and MC-I. For Expand & Explore, we use the publicly available code[4] of the corresponding authors. For PlanarCC, we kindly obtained the implementation by the authors of [159]. While both are implemented in MATLAB, all computation-heavy parts are delegated to C++ functions via MEX wrappers, such that our comparison is fair.

For PlanarCC, we follow the suggestion of the authors to stop the algorithm after 40 iterations for better runtime. Without this, for several instances, PlanarCC does not converge after one hour. With more iterations, the energy of the solutions improves slightly but at the expense of significantly longer runtime.

Note that in the Expand & Explore algorithm, each binary sub-problem includes by construction unary terms, which leads to non-planar sub-problems even when the original problem is planar.

6.7.1 2D Segmentation

We consider planar 2D segmentation problems derived from the Berkeley Segmentation Dataset [107]:

(i) models from Andres et al. [5] (BSD 300 test data, 100 instances),
(ii) models from Yarkony et al. [159] (BSD 300 training data, 200 instances).

While the former uses local edge likelihoods learned by a Random forest, the latter uses global probability of boundary (gPb). Furthermore, they are distinguished in the way

[4] http://www.wisdom.weizmann.ac.il/~bagon

they derive the edge weights w from the edge probabilities $p(e)$. In [5],

$$w_e = \log\left(\frac{p(y_e = 0)}{p(y_e = 1)}\right) + \log\frac{1-\beta}{\beta} \tag{6.7}$$

is chosen while [159] use

$$w_e = \log\left(\frac{1 - \text{gPb}_e}{\text{gPb}_e}\right) + \gamma \ . \tag{6.8}$$

Results for both datasets and different values of β and γ are shown in Figures 6.4 and 6.5. MC-I finds the global optimum for all instances. The plots show the average energy distance (mean gap) to the optimum. Among all approximate methods, CGC performs best. Concerning run times, CGC has robustly low runtime for a wide range of parameters β and γ.

A more detailed comparison for $\beta = 0.33$, as used in [78], and $\gamma = 0.3$ can be found in Tables 6.1a and 6.2a. These tables additionally show how often the methods find the global optimum (best), and how often they were able to verify the optimality by themselves (ver. opt). For Table 6.1a, ground truth segmentations and superpixels were available, such that we can calculate the Variation of Information (VI, [109]). Interestingly, the globally optimal solution does not necessarily give the best segmentation as measured by the VI (but approximately, VI distance gets larger with increasing energy as expected). This means that in practice, it is sufficient to run the much faster CGC algorithm, see also Figure 6.3.

6.7.2 3D Segmentation

We have created a new public benchmark dataset derived from the experiments in Chapter 4. Using the FIBSEM dataset, we derived a set of problem instances over a range of different problem sizes from volumes of 30^3 up to 450^3 voxels.

Results are shown in Figure 6.6 and for volumes of 400^3 are detailed in Table 6.4. Based on the energy difference of CGC and CGC-$\overline{P}$ for the planar models, we expect reduced performance for the non-planar case of 3D volume image segmentation (Figure 6.6). Still, with a runtime as fast as KL (the fastest algorithm considered), CGC-$\overline{P}$ is able to obtain much lower-energy solutions and lies between MC-I, MC-R and Expand & Explore. For instances with 400^3 voxels, MC-I was only able to find for 7 out of 8 instances the global optimum within one hour (Table 6.4).

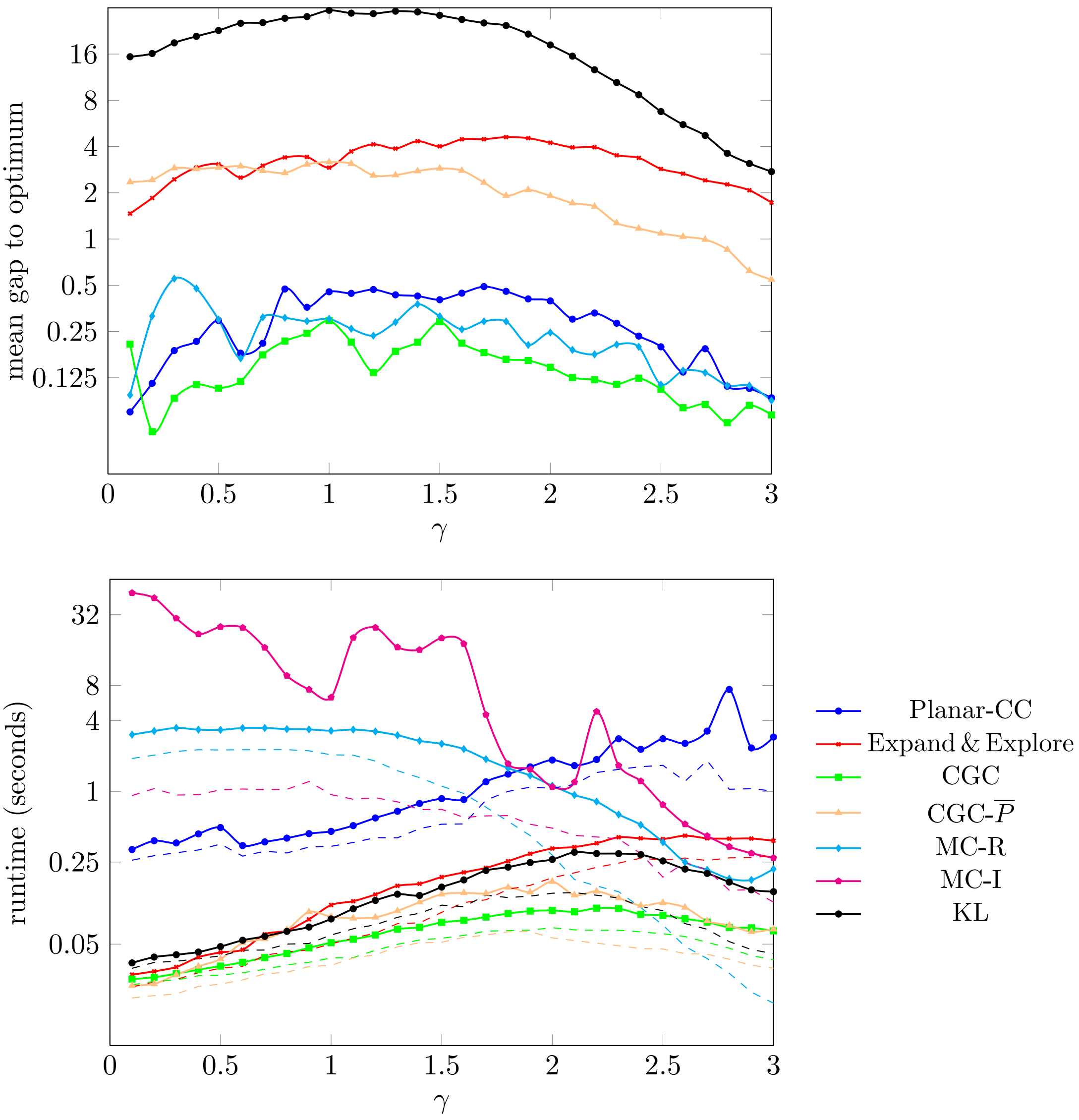


Figure 6.4: Evaluation on the planar model from Yarkony et al. [159] for different boundary penalties γ, averaged over 200 instances.
Top: Gap to the global optimal energy (MC-I) averaged over all instances.
Bottom: Mean runtime is shown as solid curves, median runtime in dashed curves. For both 2D and 3D images, CGC outperforms all competitors in terms of runtime. On 2D images, CGC gives better results in terms of energy than all competitive approximative methods. Only Integer Multicut (MC-I) gives better partitions in terms of energy but is significantly slower.

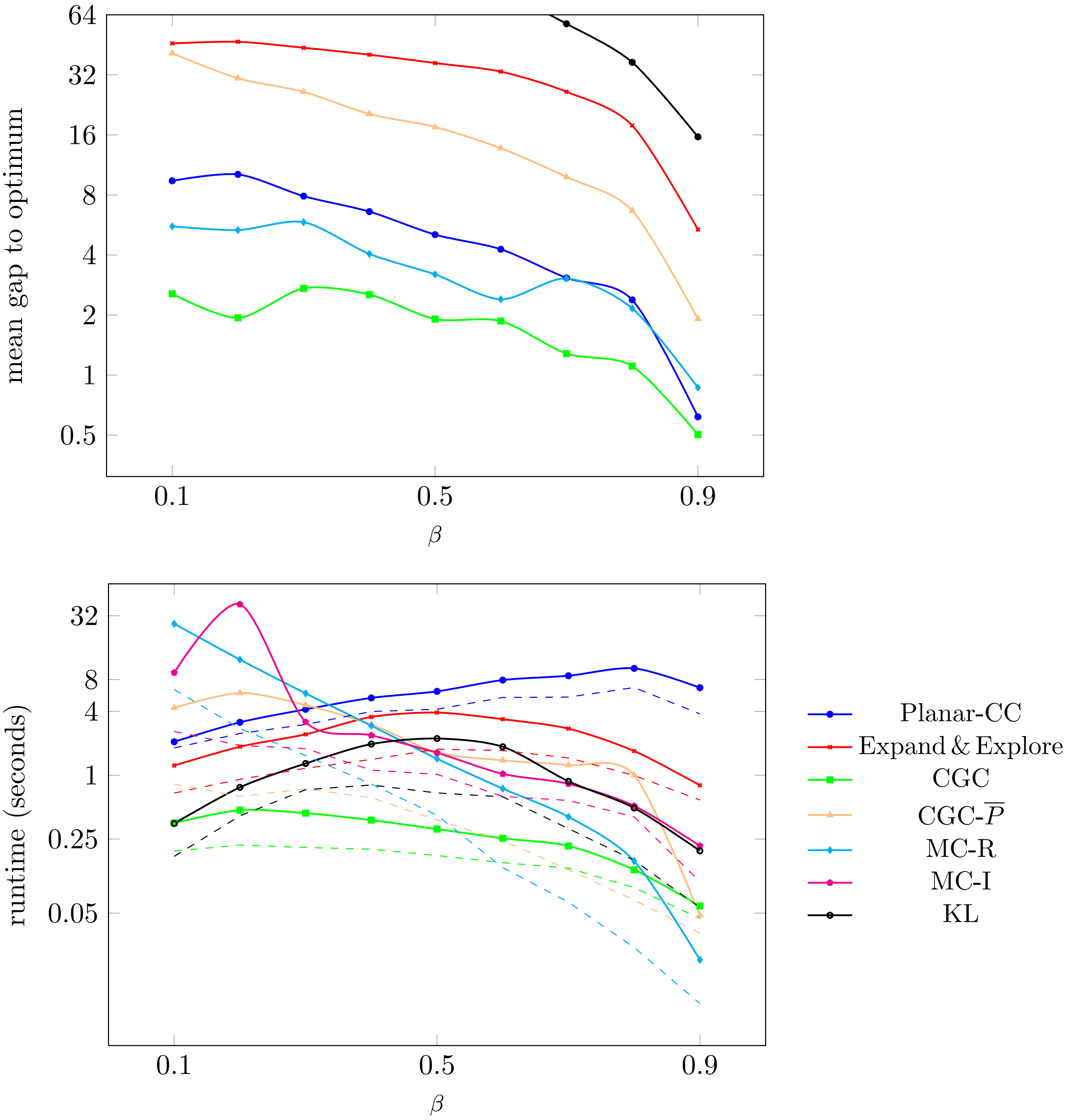


Figure 6.5: Evaluation on the planar models from Andres et al. [5] for different boundary penalties (β), averaged over 100 instances.
Top: Gap to the global optimal energy (MC-I) averaged over all instances.
Bottom: Mean runtime is shown as solid curves, median runtime in dashed curves. For both 2D and 3D images, CGC outperforms all competitors in terms of runtime. On 2D images, CGC gives better results in terms of energy than all competitive approximative methods. Only Integer Multicut (MC-I) gives better partitions in terms of energy but is significantly slower.

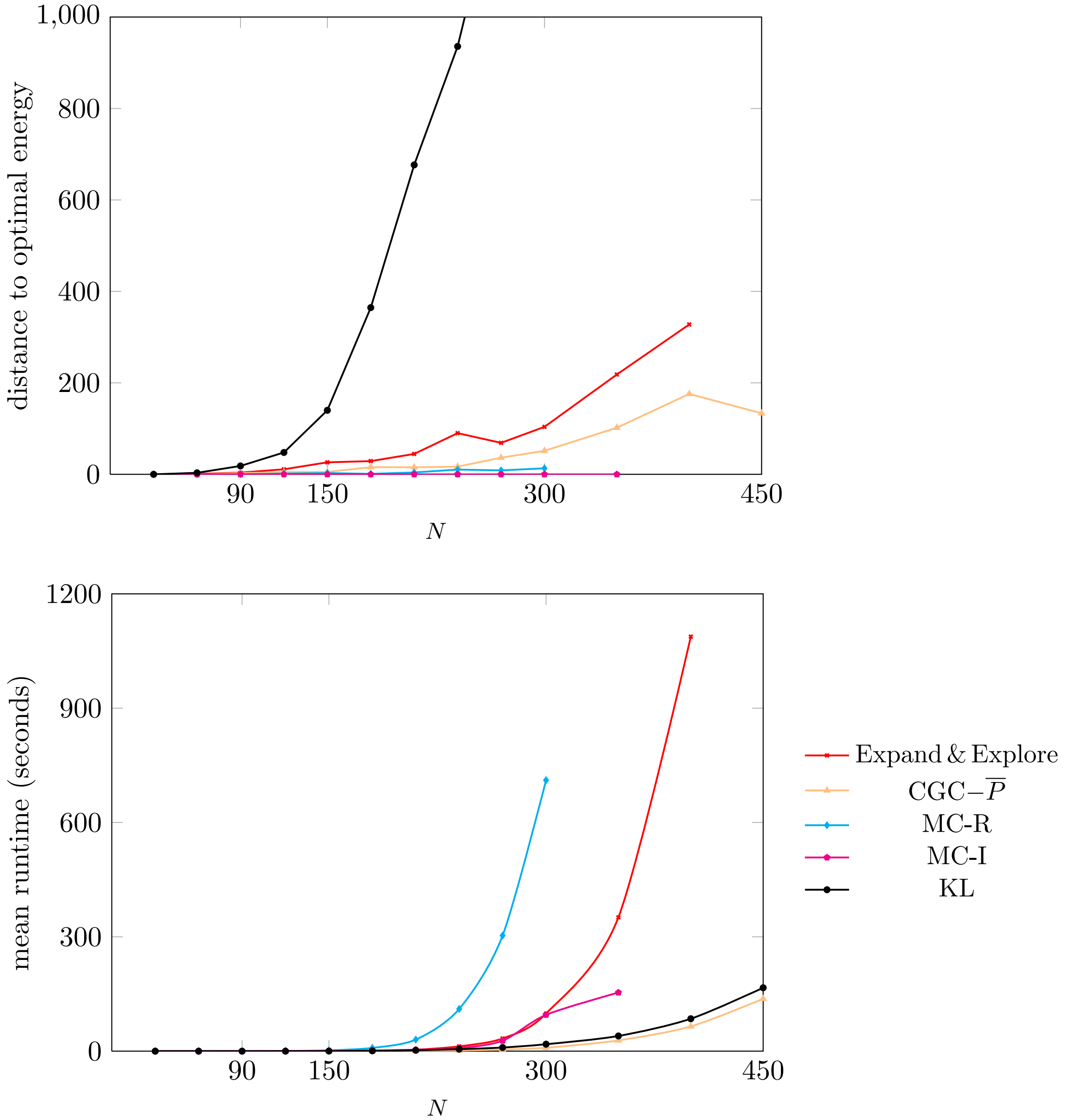


Figure 6.6: Evaluation on a 3D segmentation benchmark (derived from Chapter 4) for different volume sizes N^3 averaged over 8 instances. For 3D volume image segmentation, the approximate MC-R method beats CGC, though at significant runtime cost.

algorithm	runtime	value	bound	best	ver. opt.	VI
KL	4.96 sec	4608.57	$-\infty$	0	0	2.6431
Expand & Explore	2.90 sec	4486.57	$-\infty$	1	0	2.9153
CGC-$\overline{PB}$	6.35 sec	4466.80	$-\infty$	1	0	**2.5247**
CGC-$\overline{P}$	5.35 sec	4466.80	$-\infty$	1	0	**2.5247**
CGC-$\overline{B}$	0.63 sec	4445.06	$-\infty$	23	0	2.5355
CGC	**0.42** sec	4445.06	$-\infty$	23	0	2.5355
MC-R	5.16 sec	4447.47	4442.34	35	35	2.5490
Planar-CC	5.20 sec	4450.73	4437.29	9	8	2.5603
MC-I	2.20 sec	**4442.64**	**4442.64**	**100**	**100**	2.5363

(a) Models from Andres et al. [5].

algorithm	runtime	value	bound	best	ver. opt.
KL	0.04 sec	−73.41	$-\infty$	45	0
Expand & Explore	**0.03** sec	−89.90	$-\infty$	130	0
CGC-$\overline{PB}$	**0.03** sec	−89.45	$-\infty$	104	0
CGC-$\overline{P}$	**0.03** sec	−89.45	$-\infty$	104	0
CGC-$\overline{B}$	**0.03** sec	−92.25	$-\infty$	185	0
CGC	**0.03** sec	−92.25	$-\infty$	185	0
MC-R	3.48 sec	−91.70	−92.39	181	180
Planar-CC	0.36 sec	−92.16	−92.39	184	174
MC-I	29.80 sec	**−92.35**	**−92.35**	**200**	**200**

(a) Models from Yarkony et al. [159].

Table 6.3: Mean runtime, energy and bound (if available) for different multicut solvers. Also shown is how often each method finds the global optimum ("best"), and how often a method is able to verify the optimality by itself ("ver. opt"). Execution was aborted after one hour. Best values are marked in bold. The VI column reports the Variation of Information [109].
(a) Summary over the models from [5], 100 instances. (a) Summary over the models from [159], 200 instances. Without superpixel maps available, VI could not be calculated.

algorithm	runtime	value	bound	best	ver. opt.
KL	85.06 sec	-53476.75	$-\infty$	0	0
Expand & Explore	1087.84 sec	-57054.25	$-\infty$	0	0
CGC$-\overline{P}$	**64.95** sec	-57206.18	$-\infty$	0	0
MC-R	4121.08 sec	-20111.68	-58774.97	0	0
MC-I	745.53 sec	$-\mathbf{57319.41}$	$-\mathbf{57386.73}$	**7**	**7**

Table 6.4: Performance of various multicut solvers on instances derived from [10] (cf. FIB-SEM dataset, Chapter 4). Eight instances with cube length $N = 400$ voxels were used. See Table 6.3 for a column legend.

6.8 Conclusion

In this Chapter I have presented a new approximate solver for multicut problems – called *Cut, Glue & Cut* (CGC). The solver can be used both for planar and non-planar problems and is based on the move-making paradigm. It works in two phases:

- In the *Cut phase*, a low energy segmentation tuned to the problem's weights is created by recursively solving 2-coloring problems, either using the Blossom method (for planar problems) or QPBO-I.
- In the *Glue & Cut phase*, two adjacent sub-problems are first merged (glue) and then a possibly better cut is sought between them (cut). This process is repeated until no energy improvement is possible anymore.

The experimental evaluation shows that the new algorithm is considerably faster than existing methods while able to match an exact solver in quality, as measured by the Variation of Information on 2D images.

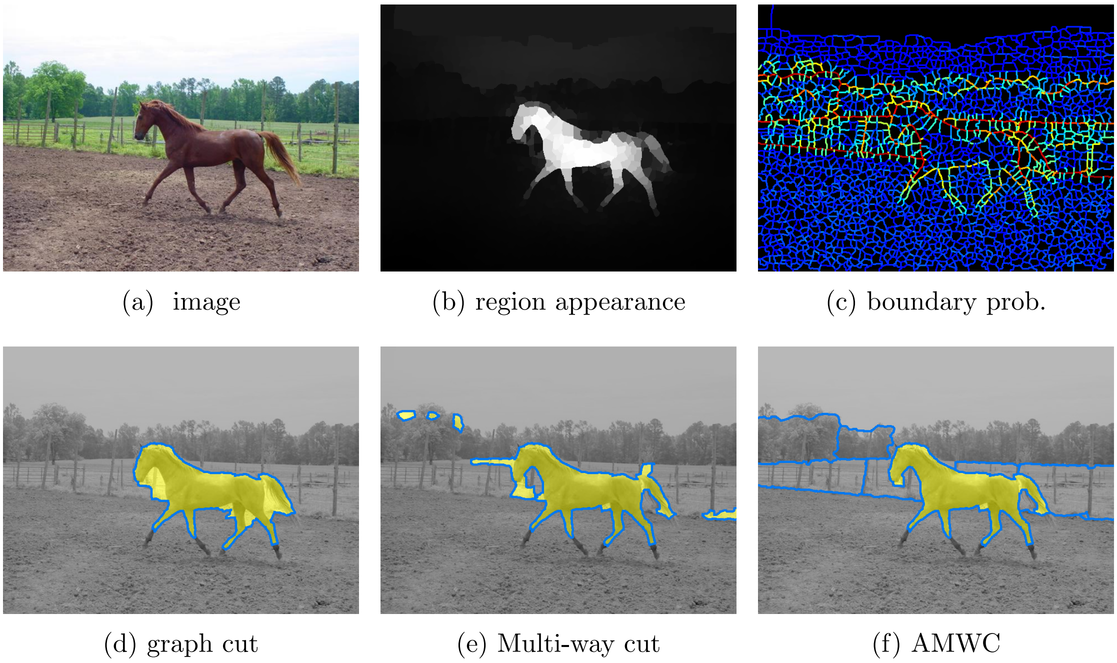
Chapter 7

Towards Joint Segmentation and Labeling

In this chapter, I propose the *Asymmetric Multi-way cut* model in order to jointly segment neurites as well as label cell organelles (e.g. mitochondria) in volumetric electron microscopy data of neural tissue.

For image segmentation, recent advances in optimization of non-submodular pairwise energy functions make it possible to combine noisy region appearance terms with pairwise terms which can not only *discourage*, but also *encourage* label transitions, depending on boundary evidence. These general multi-label second-order Conditional Random Fields have the potential to overcome problems inherent to graph cut algorithms, such as the shrinking bias. However, with the ability to encourage label transitions comes a different problem: strong boundary evidence can overrule weak region appearance terms to create new regions out of nowhere. We make the observation that some label classes exhibit strong internal boundaries, such as the background class which is the pool of all objects which are not of interest and for which no separate region appearance terms are available. Other label classes, meanwhile, should be modeled as a single region, even if some internal boundaries are visible.

We therefore propose in this chapter to treat label classes asymmetrically: for some classes, we allow a further partitioning into their constituent objects as supported by boundary evidence; for other classes, further partitioning is forbidden. We show how such a model can be expressed as both a labeling problem with a large label space as well as a binary edge labeling problem, for which we give a succinct formulation. This formulation allows us to obtain an exact solver for our new type of models by extending the optimizer of Kappes et al. In our experiments, we show where such a model can be useful for both 2D and 3D segmentation.


(a) image (b) region appearance (c) boundary prob.

(d) graph cut (e) Multi-way cut (f) AMWC

Figure 7.1: Segmentation of image (a) can combine information from both region appearance terms (b) and boundary probabilities (c). We examine different variants of pairwise Conditional Random Field models with Potts potentials. Graph cut (d) uses positive coupling strengths only, which leads to shrinking bias. Multi-way cut (e) uses both negative and positive coupling strengths, such that the creation of boundaries can be actively encouraged. However, this leads to some spurious labelings, induced by strong boundary evidence. Our proposed variant, (f), may yield a better segmentation by allowing boundaries within the background class.

7.1 Introduction

Image segmentation methods typically rely on two complementary sources of information: object appearance and boundary evidence. For example, in semantic labeling tasks [51] a set of object classes of interest is given. Each image can contain one or more of these instances, but might also contain many objects of unknown classes ("background"). One approach for semantic segmentation is to make use of (noisy) local object class probabilities – as obtained from learned appearance models – which can be regularized using local boundary cues.

On the other hand, pure partitioning problems, as in the Berkeley Segmentation Dataset [107], do not specify any object classes but rely on boundary evidence alone, e.g. [12, 5, 159].

In this chapter, I propose a combined semantic labeling *and* partitioning which can naturally deal with object classes which are known to have strong internal boundaries. Our model, called *Asymmetric Multi-way cut* (*AMWC*), jointly optimizes the region labeling, the boundaries between classes and the boundaries within classes. This model is motivated by the problem of simultaneously segmenting neurites as well as labeling mitochondria and other cell organelles in EM data of neural tissue (see Section 7.5.2).

Many segmentation algorithms, including AMWC, are formulated as second-order Conditional Random Fields over a discrete set of labels, in which the unary potentials transport local evidence for each object class. The pairwise potentials are usually chosen to be Potts functions with varying coupling strengths $w_e \in \mathbb{R}^+$, which may depend on boundary evidence. The optimal labeling (in the MAP sense) can then be found by minimizing the associated energy function.

Graph cut based algorithms have been extremely influential in the last decade [29, 93, 128], because they allow to find the optimal solution for binary labeling problems and approximate solutions for multi-label problems with non-negative coupling strengths in polynomial time. They regularize noisy detections by penalizing boundary length.

Unfortunately, this leads to "shrinking bias" [152], i.e. thin, elongated objects are cut off (Figure 7.1d and – in the context of neurite segmentation – Section 2.4.3). As a counter measure, the coupling strength can be chosen as an inverse function of boundary evidence, making label transitions *less* costly when strong boundary evidence exists and *more* costly when boundary evidence is weak. However, the general problem remains: positive coupling strengths cannot actively *encourage* label transitions.

Since negative coupling strengths w_e *encourage* label transitions and positive w_e *discourage* label transitions, a model with no restriction on the sign of w_e may be more expressive: with strong boundary evidence (resulting in $w_e < 0$) along a thin, elongated object, shrinking bias can be overcome.

Recently, Kappes et al. [79] presented a method that, contrary to others [92, 94], is able to find the globally optimal solution for these more general models. This allows us to evaluate the models without any error introduced by approximate optimization.

However, besides their increased computational hardness, models which can *encourage* label transitions also have a major drawback: spurious label transitions are provoked in highly cluttered background (Figure 7.1e) or within textured objects when strong boundary evidence overrules homogeneous region appearance terms.

In this chapter, we investigate a new subclass of labeling problems, which do allow intra-category boundaries. The energy function can still be expressed as a pairwise Conditional Random Field. Figure 7.1f shows the result of our new AMWC model, in which we allow internal edges in the "background" class, but disallow internal edges in the "foreground" class.

	can use appearance terms	can use positive coupling	can use negative coupling	can partition same-labeled regions
graph cut, e.g. [29, 93]	✓	✓	×	×
Multicut, e.g. [39, 5]	×	✓	✓	n/a
Multi-way cut, e.g. [79, 161]	✓	✓	✓	×
AMWC (this chapter)	✓	✓	✓	✓

Table 7.1: Overview of different subclasses of discrete labeling problems, which can all be expressed as pairwise Conditional Random Fields.

Contributions.

(i) A formulation of joint labeling and partitioning problems, where some classes may have internal boundaries and others may not. This addresses a gap in the literature, see Table 7.1.

(ii) An exact solver for AMWC problems based on a formulation using binary edge indicator variables [80].

(iii) Experiments that show when such a formulation is useful and when it is not.

Organization. Related work is surveyed in Section 7.2, followed by a review of the Multi-way cut formulation of the labeling problem in Section 7.3. Our new AMWC model is described in Section 7.4 together with a method to find the globally optimal solution. Experiments are presented and discussed in Section 7.5. Finally, we conclude with Section 7.6.

7.2 Related Work

Let $G = (\mathcal{V}, \mathcal{E})$ be a given pixel (or superpixel) adjacency graph. Each node $i \in \mathcal{V}$ can be assigned one of k discrete labels: $l_i \in \mathcal{L} = \{0, \ldots, k-1\}$. Many common pixel or superpixel labeling problems [29, 128, 145, 152, 14, 78] are then written as an energy minimization over the sums of unary and pairwise terms:

$$\underset{l \in \mathcal{L}^{|\mathcal{V}|}}{\operatorname{argmin}} \left\{ \sum_{i \in \mathcal{V}} E_i(l_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(l_i, l_j) \right\} . \qquad (2^{\text{nd}}\text{-order CRF, 7.1})$$

The *unary terms* are functions $E_i : \mathcal{L} \to \mathbb{R}$ and indicate the local preference of node i to be assigned a label. The *pairwise terms* are functions $E_{ij} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ that express the local joint preferences of two adjacent nodes i and j.

A common choice for the binary term is a *Potts* function [29]:

$$E_{ij}^{\boldsymbol{w}}(l_i, l_j) = \begin{cases} 0 & \text{if} \quad l_i = l_j \\ w_{ij} & \text{if} \quad l_i \neq l_j \end{cases} . \qquad \text{(Potts function, 7.2)}$$

Depending on the weight w_{ij} a Potts function can either *encourage* ($w_{ij} < 0$) or *discourage* ($w_{ij} > 0$) label transitions.

Binary labeling problems with $\mathcal{L} = \{0, 1\}$ and pairwise potentials for which $w_{ij} \geq 0$ for all $(i, j) \in \mathcal{E}$ (known as *graph cut problems*) can be solved in polynomial time with a max-flow algorithm [29, 93, 28]. Graph cut has been ubiquitous in image segmentation [128], but penalizes a weighted sum of cut edges which leads to the problem of shrinking bias [152], for which many sophisticated counter measures have been developed.

For $k > 2$ labels, (7.1) becomes a multi-label energy minimization problem that is NP-hard in general, even for non-negative weights $\boldsymbol{w}$. We will refer to the general problem of (7.1) with $2 \leq k \ll |\mathcal{V}|$ and $\boldsymbol{w} \in \mathbb{R}^{|\mathcal{E}|}$ as the *multi-way cut problem* because the optimal solution can be found as a cut in a graph with special structure (reviewed in Section 7.3). Approaches to solve this problem approximately are, amongst others, move-making algorithms [29, 14], linear programming [91, 95, 80, 78] and dual decomposition [94]. An integer linear program to which violated constraints are added in a cutting-plane fashion [79, 80] is able to find the globally optimal solution on many problem instances, see Section 7.3.

A special case of the labeling problem with $E_i(l_i) \equiv 0$, a virtually unlimited set of labels $k = |\mathcal{V}|$ and no restriction on the sign of w_{ij} is called the *correlation clustering* [16] or *multicut problem* [39]. The multicut formulation has recently become popular for unsupervised image segmentation [5, 10, 14, 87, 159, 4, 98] where the weights $\boldsymbol{w}$ are either learned in a supervised fashion [5, 14, 10, 87, 98], or derived from boundary detectors such as gPb [106] as in [159]. In particular, Chapters 4 and 5 have introduced such models for automatic segmentation of neurites in electron microscopy volume images of neural tissue. Multicut models have an inherent model-selection ability [14] such that they recover the optimal number of regions needed for an accurate segmentation automatically, based only on boundary evidence. However, as region appearance is not taken into account, the resulting segments are not given a class label. Labels can be assigned to segments only in a post-processing step, which takes the segmentation as given.

Table 7.1 summarizes the different subclasses of pairwise Conditional Random Field models considered in this chapter. Note that we review only unified formulations with a specified objective function here and omit workflows that chain several processing steps.

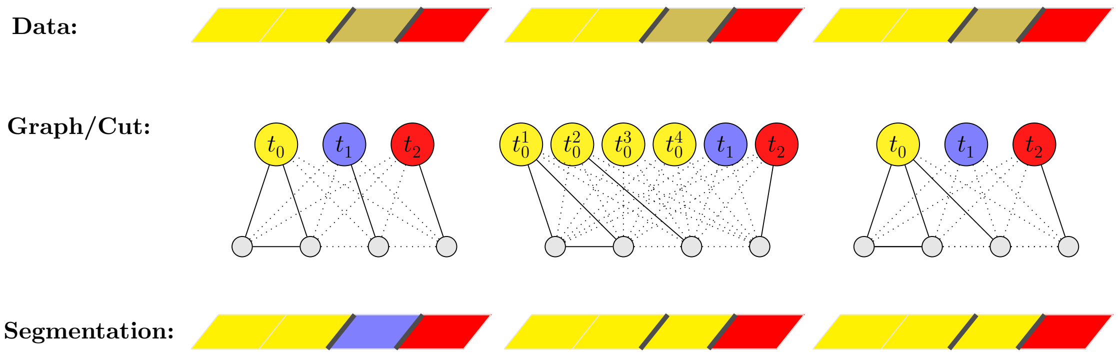


Figure 7.2: Illustration of the graph representation of (7.1), as given by (7.3). Shown is a 4 × 1 pixel image strip ("Data" row). There are three possible classes $\mathcal{L} = \{\text{red}, \text{blue}, \text{yellow}\}$. From the left to the right pixel, the region appearance terms indicate strong preference for the yellow class (pixels 1, 2), a preference for yellow over the blue class (pixel 3, p_3), and a strong preference for the red class for the last pixel. Furthermore, there are strong boundaries to the left and right of p_3. *Left:* Multi-way cut solution. Due to the strong boundary evidence, p_3 is assigned the blue label contrary to its unary potential. *Middle:* The AMWC model with $\mathcal{A} = \{\text{yellow}\}$ is obtained by duplicating the terminal nodes for the yellow class $|\mathcal{V}| = 4$ times. *Right:* With the modified constraint (7.5c′), the number of terminal nodes does not have to be increased to yield the same solution.

7.3 Multi-Way Cut Formulation

In order to discuss our new AMWC model in Section 7.4, we first review a different representation of the labeling problem (7.1) with Potts potentials (7.2), based on Kappes et al. [79, 80].

We call the nodes $\mathcal{V}$ the *internal nodes* and edges $\mathcal{E}$ the *internal edges*. We then define a new graph $G' = (\mathcal{V}', \mathcal{E}')$, where a set of *terminal nodes* $T = \{t_0, \dots, t_{k-1}\}$, representing k labels, has been added (Figure 7.2, left). Furthermore, *terminal edges* are introduced between each pair of internal and terminal nodes as well as between all pairs of terminal nodes:

$$\begin{aligned} \mathcal{V}' &= \mathcal{V} \cup T \quad , \\ \mathcal{E}' &= \mathcal{E} \cup \Big\{ (t, v) \quad \mid t \in T, v \in \mathcal{V} \Big\} \\ &\quad \cup \Big\{ (t_i, t_j) \mid 0 \le i < j < k \Big\} \quad . \end{aligned} \tag{7.3}$$

Then, the node labeling problem (7.1) with Potts potentials (7.2) can be written using binary indicator variables $\boldsymbol{y}$ for the edges $\mathcal{E}'$ and weights $\boldsymbol{w}'$, derived from the potential functions $E_i(\cdot)$ and $E_{ij}(\cdot,\cdot)$, see Kappes et al. [79]:

$$\underset{\boldsymbol{y}\in\{0,1\}^{|\mathcal{E}'|}}{\operatorname{argmin}} \left\{ \sum_{(i,j)\in\mathcal{E}'} w'_{ij} \cdot y_{ij} \right\} \quad \text{s.t. } \boldsymbol{y} \in \mathrm{MWC}_G \ , \tag{7.4}$$

where MWC_G is the multi-way cut polytope as defined by the following set of linear constraints [80]:

$$\begin{aligned}
\sum_{(i,j)\in P} y_{ij} &\geq y_{uv} && \forall\,(u,v)\in\mathcal{E} \\
& && P \in \mathrm{Path}(u,v) \subseteq \mathcal{E} && \text{(7.5a)} \\
y_{tt'} &= 1 && \forall\,(t,t')\in T,\ \ t\neq t' && \text{(7.5b)} \\
y_{tu}+y_{tv} &\geq y_{uv} && \forall\,(u,v)\in\mathcal{E},\ \ t\in T && \text{(7.5c)} \\
y_{tu}+y_{tv} &\geq y_{tv} && \forall\,(u,v)\in\mathcal{E},\ \ t\in T && \text{(7.5d)} \\
y_{tv}+y_{uv} &\geq y_{tu} && \forall\,(u,v)\in\mathcal{E},\ \ t\in T\ . && \text{(7.5e)}
\end{aligned}$$

For internal nodes, the *cycle constraint* (7.5a) [39, 5] intuitively forbids "dangling" boundaries (discussed in detail in Section 4.3.2). Constraint (7.5b) ensures that all terminals are always separated. Finally, (7.5c)-(7.5e) constitute cycle constraints for all cycles of three nodes involving one terminal. In particular, (7.5c) says that, if there is a label transition ($y_{uv} = 1$), label t cannot belong to both u and v (which would be the case for $y_{tu} = 0$ and $y_{tv} = 0$).

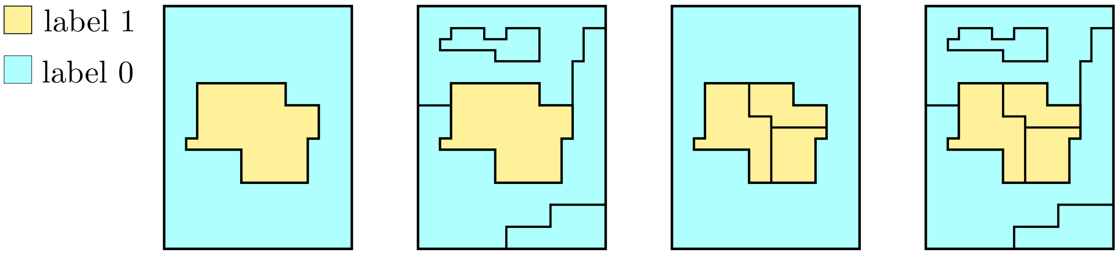
Although a complete description of MWC_G needs a possibly exponential number of constraints, in practice only a small set of active constraints is needed to find the (valid) globally optimal solution. The cutting plane method [79, 80] first formulates an unconstrained integer linear program and then identifies violated constraints in the solution, which are subsequently added to the problem. This is repeated until a solution does not violate any of the constraints in (7.5a)-(7.5e), yielding the globally optimal solution.

7.4 The Asymmetric Multi-Way Cut Model

In the proposed Asymmetric Multi-way cut model, we want to *allow* internal boundaries within regions labeled as $l \in \mathcal{A}$, and *disallow* internal boundaries in all regions labeled $l \in \{0,\ldots,k-1\} \setminus \mathcal{A}$. This is illustrated in Figure 7.3.

7.4.1 Formulation within the Binary Edge Labeling Framework

We first give the advantageous formulation of AMWC in terms of the binary labeling of $\mathcal{E}'$ in (7.4). One way to formulate the model is to replace every terminal node $t_a \in \mathcal{A}$ with a set $T_a = \{t_a^0, \ldots, t_a^{|\mathcal{V}|-1}\}$, as shown in Figure 7.2, middle and Figure 7.4, for which



$\mathcal{A} = \emptyset$ $\quad$ $\mathcal{A} = \{0\}$ $\quad$ $\mathcal{A} = \{1\}$ $\quad$ $\mathcal{A} = \{0, 1\}$

Figure 7.3: Illustration of allowed cuts, depending on $\mathcal{A}$.
Blue denotes background (label 0) and yellow foreground (label 1). From left to right we demand no label transitions within foreground and background regions ($\mathcal{A} = \emptyset$), we allow label transitions within the background class ($\mathcal{A} = \{0\}$), or only within the foreground class ($\mathcal{A} = \{1\}$) or in both classes ($\mathcal{A} = \{0, 1\}$). In all cases, we admit only closed contours (black boundaries).

edge weights are copied from the existing terminal edges. These new nodes can represent a partitioning of class a into subclasses, which are separated by salient boundaries in the image. Similar to the multicut formulation, the label space has to be increased dramatically. By setting $|T_a| = |\mathcal{V}|$, solutions where every node is assigned a different label from the set T_a are made possible.

However, instead of adding $|\mathcal{A}| \cdot (|\mathcal{V}| - 1)$ additional terminal nodes, the same effect can be achieved by simplifying a single constraint in the binary edge labeling formulation of Section 7.3. We relax constraint (7.5c) to be

$$y_{tu} + y_{tv} \geq y_{uv} \quad \forall\, (u, v) \in \mathcal{E}, \quad t \in T \setminus \mathcal{A} \ . \tag{7.5c$'$}$$

In practice, we extend the implementation of [80] such that constraint (7.5c) is only added in the cutting-plane procedure when $t \notin \mathcal{A}$ holds.

With this change, we obtain a method that is able to solve our new AMWC-type models to global optimality.

7.4.2 Formulation as a Node Labeling Problem

Our AMWC model can still be formulated as a second-order Conditional Random Field (7.1) with Potts potentials. Starting from the edge labeling formulation from the previous section, we construct the corresponding model by choosing a new set of labels $\mathcal{L}'$ as well as constructing appropriate unary and pairwise potentials $E'_i(\cdot)$ and $E'_{ij}(\cdot, \cdot)$. Let

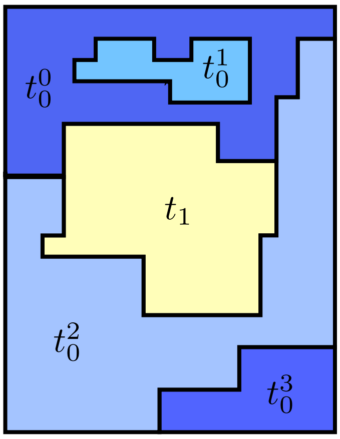


Figure 7.4: In this example with $\mathcal{A} = \{0\}$, the background class (label zero, shown in blue) actually consists of several distinct segments t_0^0, t_0^1, t_0^2 and t_0^3 (different shades of blue), all separated by boundaries (black). Another region, $t_1 \notin \mathcal{A}$ cannot be split into sub-segments.

the original set of labels be $\mathcal{L} = \{t_0, \dots, t_{k-1}\}$. Then we set

$$\mathcal{L}' = \bigcup_{a \in \mathcal{A}} \left(\{t_a^0, \dots, t_a^{|\mathcal{V}|-1}\} \right) \cup (\mathcal{L} \setminus \mathcal{A}) \ , \tag{7.6a}$$

$$E_i'(l_i') = \begin{cases} E_i(l_i') & \text{if} \quad l_i' \in (\mathcal{L} \setminus \mathcal{A}) \\ E_i(a) & \text{if} \quad l_i' = t_a^j \end{cases} \ , \tag{7.6b}$$

$$E_{ij}'(l_i', l_j') = E_{i,j}(a, b) \quad \text{with } t_a^? = l_i, t_b^? = l_j \ . \tag{7.6c}$$

In (7.6a), we introduce $|\mathcal{V}| - 1$ additional labels for each label class for which internal boundaries are allowed. These extra labels are assigned the same weights in the new unary terms (7.6b) as the original label class. Finally, the Potts terms do not change (7.6c).

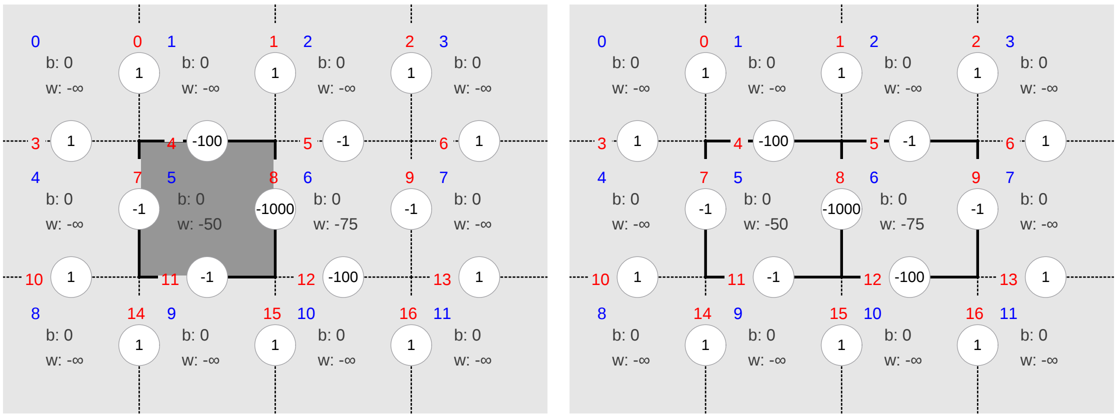
The inflated label space makes this formulation unwieldy for practical optimization methods: similar to multicut models, a large number of different labelings, obtained by label permutations, have the same energy.

7.4.3 Labeling of Regions and Boundaries

At first sight, it may seem that an optimal solution of AMWC can also be obtained by the following algorithm:

1. Run the *Multi-way cut* algorithm using the original label set $\mathcal{L} = \{0, \dots, k-1\}$ to obtain a segmentation into regions $R_1, \dots, R_n$.
2. For each region R_i assigned a label $l \in \mathcal{A}$, run the *multicut* algorithm to obtain an internal partitioning.

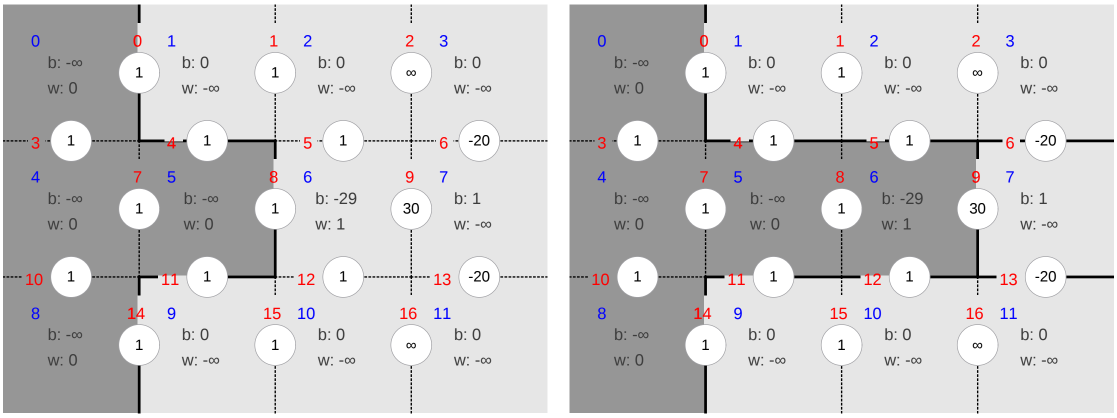
We give two toy examples with $\mathcal{L} = \{\mathtt{b}, \mathtt{w}\}$ in which this decomposition is not possible, both with the setting $\mathcal{A} = \{\mathtt{w}\}$. In Figure 7.5, the $\mathtt{b}$-object *shrinks* when using the



(a) $\mathcal{A} = \emptyset$ (standard multi-way cut)

(b) $\mathcal{A} = \{\mathtt{w}\}$ (AMWC)

Figure 7.5: Toy example showing how AMWC can lead to changes in the extent of objects. Here, we consider a binary labeling problem of a 4 × 3 pixel grid with classes $\mathcal{L} = \{\mathtt{b}, \mathtt{w}\}$, which are shown in dark and light shades of gray, respectively. In the center of each square, unary potentials are given for b and w. On each boundary, the circled number gives the weight parameterizing the associated Potts potential. *Left:* Using the standard Multi-way cut model, the segmentation yielding the best energy marks a single pixel as background. *Right:* With the same weights, the AMWC model with $\mathcal{A} = \{\mathtt{w}\}$ yields a segmentation in which the b-object shrinks and the w-object grows. See Figure 7.6 for an example of the opposite case.



(a) $\mathcal{A} = \emptyset$ (standard multi-way cut)

(b) $\mathcal{A} = \{\mathtt{w}\}$ (AMWC)

Figure 7.6: Toy example showing how AMWC can lead to changes in the extent of objects. *Left:* Optimal solution for the standard Multi-way cut model. *Right:* Using the same weights, the AMWC model with $\mathcal{A} = \{\mathtt{w}\}$ yields a segmentation in which the extent of the b-object changes. (For notation, also see Figure 7.5).

AMWC model with respect to the Multi-way cut solution; in Figure 7.6, the b-object *grows* when using the AMWC model with respect to the Multi-way cut solution.

7.5 Experiments

In this section, we show qualitatively when the AMWC model is useful and when it is not. A quantitative analysis is subject of future work.

7.5.1 Application to 2D Segmentation of Photographs

We consider foreground/background segmentation problems with $\mathcal{L} = \{0, 1\}$, in which internal boundaries are only allowed in the background: $\mathcal{A} = \{0\}$. For all images, we first compute an over-segmentation into superpixels using a seeded watershed algorithm (Section 2.4.5) on an elevation map combining gradient magnitude and the output of the generalized probability of boundary (gPb) detector from [106].

Then, the superpixel adjacency graph $G = (\mathcal{V}, \mathcal{E})$ defines the structure of the Conditional Random Field (7.1). For each edge $(i, j) \in \mathcal{E}$ which represents the shared boundary between superpixels i and j, we compute the mean boundary probability, $\mathbf{f}_{ij}$, as given by the gPb detector. Weights $w_{ij} \in \mathbb{R}$ are then obtained as follows

$$w_{ij} = \log \frac{1 - \mathbf{f}_{ij}}{\mathbf{f}_{ij}} + \log \frac{1 - \beta}{\beta} \quad , \tag{7.7}$$

where $\beta \in [0, 1]$ is a hyper-parameter giving the prior boundary probability. As region appearance terms, we use the output of the object saliency detector [123], or region appearance terms derived from manually placed object bounding boxes. Again, there is a bias hyper-parameter α which gives the prior foreground probability. Finally, we write the energy function (7.1) as

$$\underset{\boldsymbol{l} \in \mathcal{L}^{|\mathcal{V}|}}{\operatorname{argmin}} \left\{ \gamma \cdot \sum_{i \in \mathcal{V}} E_i(l_i) + \sum_{(i,j) \in \mathcal{E}} E^{\boldsymbol{w}}_{ij}(l_i, l_j) \right\} \quad , \tag{7.8}$$

where the hyper-parameter γ weights unary and pairwise terms.

Figure 7.7 gives examples where the AMWC formulation can help and where it cannot. Column (a) shows the original images, taken from benchmark datasets [121, 107, 27]. Column (b) shows foreground maps either obtained using [123] or given as manual bounding box annotations (rows 1-4). The boundary probability for superpixel edges is visualized in column (c). The Multicut algorithm, column (d), ignores the region appearance terms and gives a decomposition into regions which are shown with random colors. Column (e) shows the visually best solution of a standard graph cut model. Finally, columns (f) and (g) show results for both the multi-way cut model as well as the AMWC model with $\mathcal{A} = \{\text{background}\}$.

For each row, hyper-parameters α, β and γ – shared among Multicut, Multi-way cut and Asymmetric Multi-way cut models – were chosen to give reasonable and comparable results for these three algorithms.

For the pedestrian detection (Figure 7.7, rows 1–4) both the local appearance and edge detection terms are weak. The latter leads to regions which "leak" into the background when using multicut segmentation. Classical graph cut methods suffer severely from the very rough local data terms and show many artifacts caused by shrinking bias. While Multi-way cut generates foreground artifacts due to strong edges present in the *background*, the proposed AMWC model can handle these by introducing closed contours in the background at these locations. Sometimes, however, strong within-foreground contours can produce some artifacts (rows 2 and 4).

For the examples using saliency detection as the region appearance model (Figure 7.7, rows 5–8), both the region and edge terms are more confident. However, graph cut still shows shrinking artifacts and AMWC sometimes "hallucinates" foreground-regions in the background, though this is no longer as significant as with the weaker data terms in rows 1–4.

The run times for Multicut, Multi-way cut and AMWC are comparable with less than 10 seconds per image.

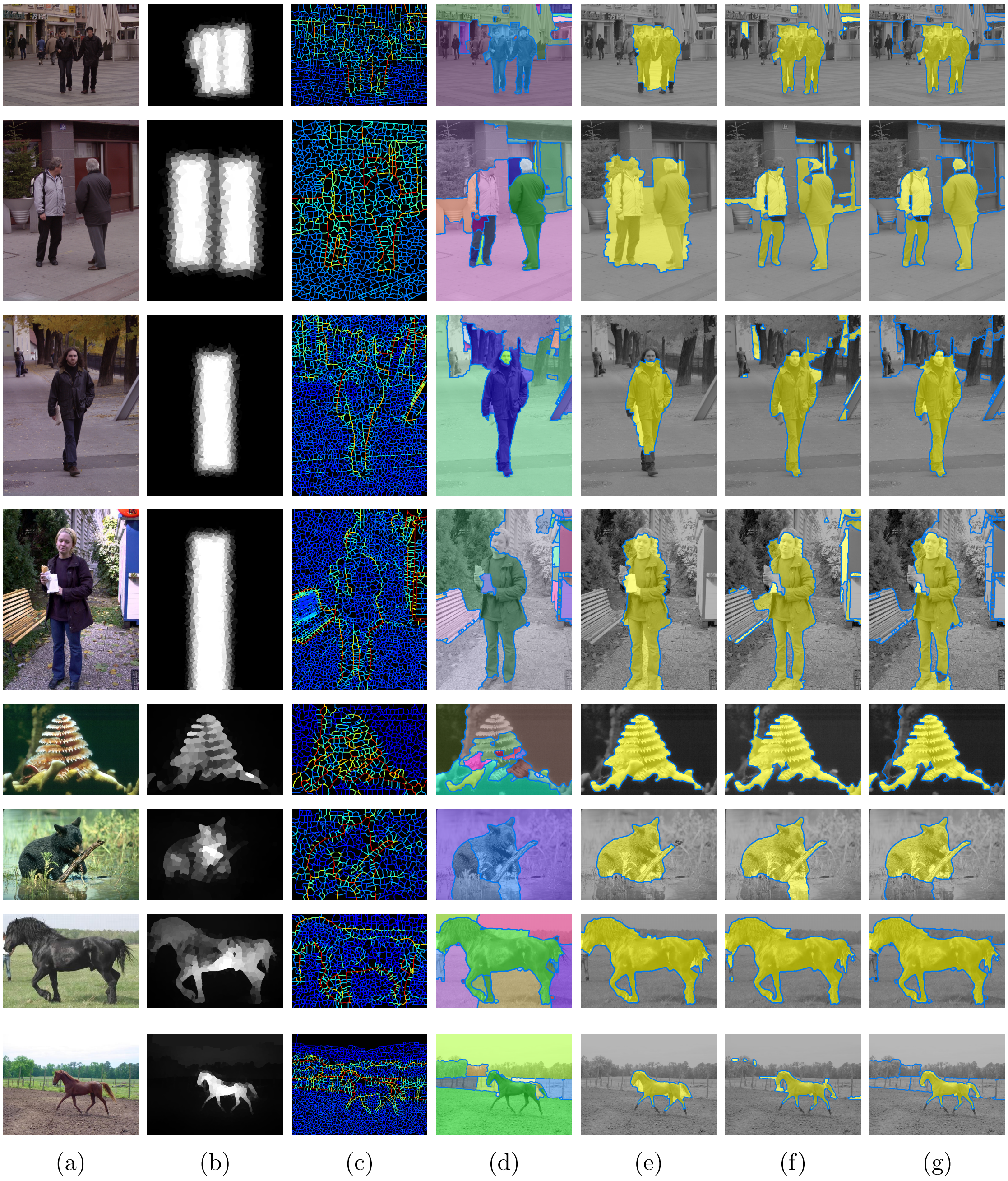


Figure 7.7: Example segmentations of various pairwise Conditional Random Fields Models. (a) original image, (b) region terms, (c) boundary terms, (d) Multicut, (e) best graph cut, (f) Multi-way cut and (g) AMWC. For details see text.

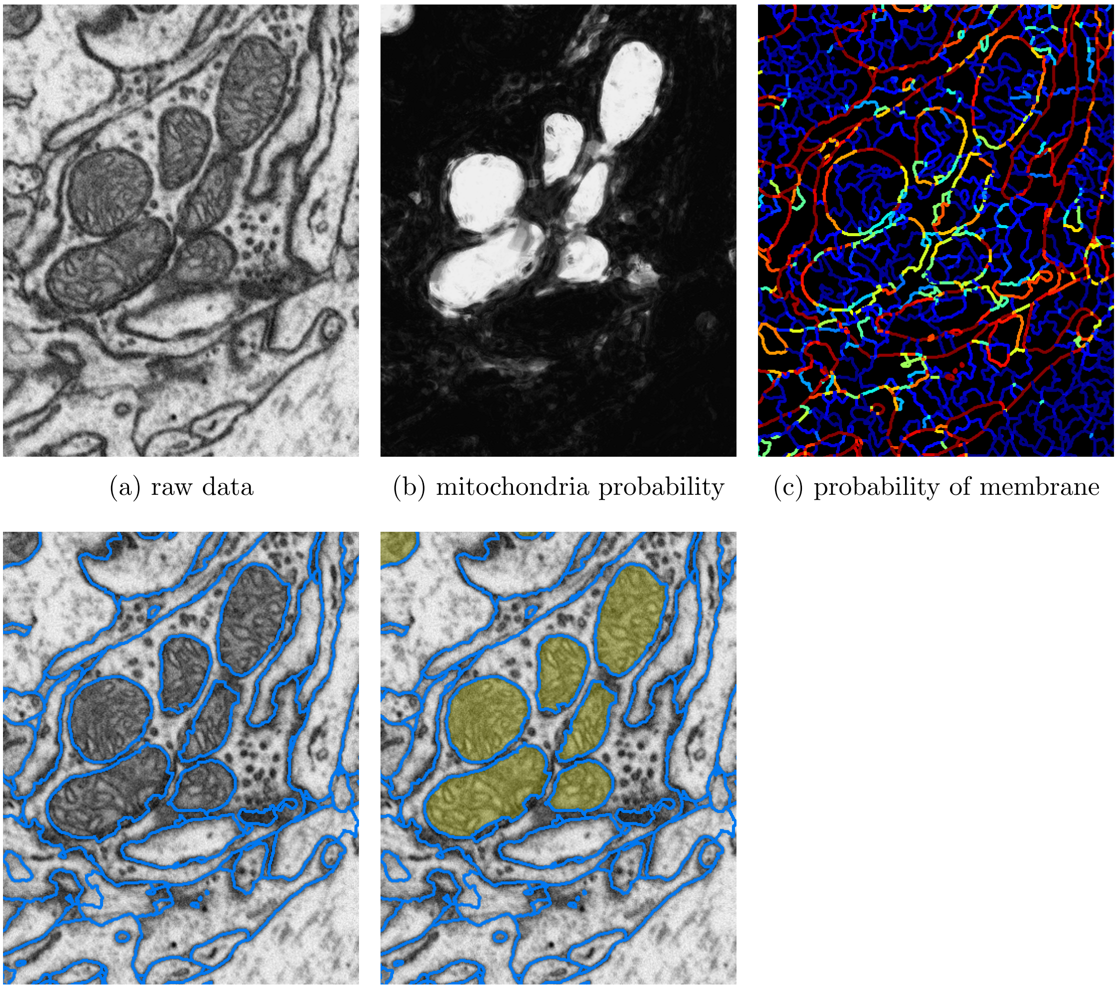
7.5.2 Application to Joint Neurite and Cell Organelle Segmentation

In Chapter 4, we have shown the benefits of applying the multicut model to automatic segmentation of neurites in high-resolution electron microscopy volume images of neural tissue. In the SBFSEM dataset (Section 4.5.1), we could benefit from the chosen staining, which suppressed internal structures of neurites, such as mitochondria, vesicles and postsynaptic densities. However, researchers are worried that it is too difficult to reliably detect synapses based on shape-cues alone (Shawn Mikula, personal communication). This is one reason why the more complicated conventional staining, such as the one used for the FIBSEM dataset in Section 4.5.2, needs to be dealt with.

In the past, automatic methods to segment mitochondria [103] or synapses [97, 19], have been considered independently from the neurite segmentation problem (cf. Section 2.3). The Asymmetric Multi-way cut model now allows to combine both problems into a single, joint model.

In our preliminary experiments, we consider two classes, mitochondrion m and cytoplasm c. In the future, we plan to consider also vesicle clusters and postsynaptic densities. We set $\mathcal{A} = \{c\}$ to allow internal boundaries within the cytoplasm class — accounting for the strong boundary evidence of neurite membranes, which separate regions of cytoplasm into distinct neurites.

Figure 7.8 shows results on the FIBSEM dataset from Section 4.5.2, where we combine learned pixel-wise mitochondrion probabilities with learned membrane (boundary) probabilities. Starting from the raw data (7.8a), we train a Random Forest classifier using ILASTIK's pixel classification workflow (Section 3.1) to distinguish mitochondria voxels from background. The resulting probability map is shown in (7.8b). Additionally, we train a boundary classifier using the interactive boundary learning tool FaceLabeler from Section 3.7. Edge weights derived from this Random Forest classifier are depicted in Figure 7.8c. In Figure 7.8d, the multicut segmentation method from Chapter 4 has been applied by disregarding the region terms. Finally, Figure 7.8e shows the result of the AMWC algorithm. This is the first time that a *joint* partitioning into neurons and labeling of intracellular structures becomes possible.



(a) raw data

(b) mitochondria probability

(c) probability of membrane

(d) Multicut

(e) Asymmetric Multi-way cut model

Figure 7.8: From the raw volume image (a), a mitochondria versus background probability map (b) was obtained using ILASTIK [140]. Boundary probabilities are obtained from a Random Forest classifier using local edge features (c). The multicut algorithm (d) yields a decomposition of the volume image into segments, relying only on the boundary evidence in (c). Finally, Asymmetric Multi-way cut (e) is the first method that can jointly find a similar segmentation to the one in (d), while at the *same* time labeling regions by their appearance (here: mitochondria (yellow) vs. cytoplasm).

7.6 Conclusion

In this chapter, I have introduced a new subclass of non-submodular pairwise multi-label Conditional Random Fields with Potts potentials in which

(i) label transitions can be both discouraged as well as encouraged and

(ii) some labels, such as background, are allowed to have internal boundaries. As a consequence, strong boundaries within these classes can be naturally accommodated by a further partitioning.

The proposed model can be solved exactly using an extension of an existing Multi-way cut solver. We expect this model to be most useful in a regime where regional appearance terms and boundary evidence are both noisy, but, and this is crucial: complementary. In this setting, this chapter has offered a principled unified approach to simultaneous labeling and partitioning.

For application to automated neurite segmentation, we hope that this method will prove useful for jointly segmenting all neurites as well as labeling cell organelles, such as mitochondria.

Chapter 8

Conclusion and Outlook

The research presented in this thesis has focused on the challenging problem of *automated neurite segmentation* in electron microscopy volume images of neural tissue. Towards solving this problem, I have proposed several machine learning-based segmentation approaches. These approaches all start from an over-segmentation of the volume image into supervoxels. In particular, I have considered a *simultaneous merge decision* of all pairs of adjacent supervoxels based on image evidence.

Contributions

- In Chapter 4 and reference [10] we have adapted a probabilistic graphical model – multicut segmentation – which ensures that merge decisions are consistent and surfaces of final segments are closed. Although the model is NP-hard to optimize, we show that our cutting-plane approach coupled with an integer linear programming solver can find the global optimum for real-world 3D problems of up to 10^6 variables. For good performance, adding only facet-defining constraints and a fast, parallel search for violated constraints is crucial. In our quantitative analysis we show that in terms of segmentation quality measures, the closed-surface regularization is beneficial.

- The decision to merge or not to merge adjacent supervoxels is driven by the weights $\boldsymbol{w}$ which are derived from a learned likelihood in Chapter 4. Next, we have investigated whether these weights can instead be chosen directly such as to minimize the expected loss (Chapter 5 and reference [98]). As loss functions, we consider Rand Index and Variation of Information, which are *global* measures based on the contingency table. We use structured learning coupled with an efficient exhaustive enumeration over small subsets of variables to find the most violated constraint. In a quantitative analysis on two different datasets we found that this method can improve upon unstructured learning.

- Whether optimization of the multicut objective is tractable in practice depends on the quality of the edge weights (how many "holes" are created by thresholding the weights). In order to scale to challenging and large datasets, I have developed an approximate, blockwise and hierarchical optimization scheme for multicut segmentation in Chapter 5 and reference [98].

- Furthermore, in Chapter 6, we propose a novel *move-making approach* to finding low-energy solutions to the multicut segmentation problem. We call this method the *Cut, Glue & Cut* algorithm. Compared to the optimal solution, this algorithm can find locally optimal solutions which give equal performance as measured by segmentation quality measures. However, our algorithm finds these solutions significantly faster than competing methods.

- Chapter 7 discusses a novel approach to *jointly* find a closed-surface segmentation as well as to label the resulting segments. In order to allow for *within-class cuts*, I have modified the Multi-way Cut formulation of image segmentation. Here, the application in mind is the simultaneous segmentation of neurites together with the semantic labeling of cell organelles such as mitochondria. Towards this goal, we have reported qualitative experiments that show when the proposed *Asymmetric Multi-way Cut* model can be beneficial.

- Finally, I have presented some of the software tools written for this thesis in Chapter 3. The open source ILASTIK software has been used for obtaining voxel probability maps, which are needed for generating an initial over-segmentation into supervoxels. Furthermore, ilastik's CARVING module was improved to facilitate the acquisition of gold standard segmentations for my quantitative experiments. As volume viewing and annotation component, ilastik utilizes the VOLUMINA library, which I have co-written.

  In order to work with label volumes, region adjacency graphs and cell complexes, I have written tools which integrate into volumina, such that a boundary classifier can be trained interactively. Furthermore, the implementation of hierarchical and blockwise approximate multicut segmentation builds upon novel software for working with hierarchies of cell complexes.

  To handle huge datasets, the open source BLOCKEDARRAY library provides blockwise operations, such as connected component analysis and in-memory compressed, blocked arrays.

Outlook

Segmentation Model.
Multicut segmentation can be formulated as an integer linear program. The formulation as an energy function is appealing, because it allows to separate the analysis of segmentation quality into (i) the performance of the *model* itself and (ii) the performance of model *optimization*. As we have shown, the globally optimal MAP solution can often be found. The block-based optimization scheme from Chapter 5 can scale the segmentation to large volume images. The empirical success of this approximate optimization scheme suggests to me that instead of working on optimization methods, research should now mainly focus on improving the model.

Recently, *agglomerative clustering* has seen renewed interest in the neurite segmentation community [69, 119, 76, 101, 26], as well as in the image segmentation community [126]. Because supervoxels are merged step-by-step, features for segments and surfaces can be re-computed after each merge. Possibly, this can improve agglomerative clustering methods enough to overcome problems related to greedy optimization.

As a compromise, one might envision the following *hierarchical scheme*. First, multicut segmentation is applied to a fine over-segmentation, but biased such as to yield a *coarser* over-segmentation. Next, new features are computed. As the over-segmentation is coarser, these features can be *more expressive*. For example, one could include the directionality of tubular segments, a feature that is uninformative on the very fine initial over-segmentation. Based on the output of a classifier, a new multicut problem is then set up and the MAP segmentation is found. The entire process is repeated multiple times.

Feature learning.
Assuming large enough fragments of neurites can be reliably detected, it is then important to come up with better features for classifying whether two adjacent supervoxels should be merged, for which [26] shows an interesting direction.

While we found that structured learning can indeed improve the quality of resulting segmentations (Chapter 5), recent research by Bogovic et al. [26] indicates that hundreds of features, together with highly non-linear classifiers (such as neural networks), may be necessary for further improvements in accuracy. Structured learning, however, employs a linear classifier and is typically used with only tens of features.

Biological Prior Knowledge.
Finally, in my opinion a long-term aim should be to incorporate long-range affinities between *any* pair of supervoxels (not only between adjacent supervoxels), into a new segmentation model. The current model “only” enforces closed membranes. In addition, biology dictates certain rules over larger scales. These rules should either not be violated (hard-constraints) or incur a penalty when violated (soft-constraints). In Section 4.7,

I have already described the prior that neurites should not be wholly contained within a small block of tissue without being connected to any soma. In addition, it may be useful to consider rules about branching angles of neurites, priors on geometry (tubular structures) as well as the interaction of neurites with cell organelles and synapses.

List of Publications

Peer-reviewed Conference Proceedings

- {Bjoern Andres, Thorben Kroeger}*, Kevin L Briggman, Winfried Denk, Natalya Korogod, Graham Knott, Ullrich Koethe, and Fred A Hamprecht. Globally Optimal Closed-Surface Segmentation for Connectomics. In *European Conference on Computer Vision (ECCV)*, volume 7574 of LNCS, pages 778 – 791. Springer, 2012 (*equal contribution)
- Thorben Kroeger, Shawn Mikula, Winfried Denk, Ullrich Koethe, and Fred A Hamprecht. Learning to Segment Neurons with non-local Quality Measures. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 8150 of LNCS, pages 419 – 427. Springer, 2013
- {Thorsten Beier, Thorben Kroeger}*, Joerg Kappes, Fred A Hamprecht. Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning. In *Computer Vision and Pattern Recognition (CVPR)*, under review. IEEE, 2014 (*equal contribution)

Contributions beyond the Scope of this Thesis

- Boray F Tek, Thorben Kroeger, Shawn Mikula, and Fred A Hamprecht. Automated Cell Nuclei Detection for Large-Volume Electron Microscopy of Neural Tissue. In *International Symposium on Biomedical Imaging (ISBI)*, to appear. IEEE, 2014
- Bjoern Andres, Ullrich Koethe, Thorben Kroeger, Moritz Helmstaedter, Kevin L Briggman, Winfried Denk, and Fred A Hamprecht. 3D segmentation of SBFSEM images of neuropil by a graphical model over supervoxel boundaries. *Medical Image Analysis*, 16(4):796805, 2012.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. 18, 30, 31

[2] Helmut Alt, Ulrich Fuchs, and Klaus Kriegel. On the number of simple cycles in planar graphs. In *Graph-Theoretic Concepts in Computer Science*, volume 1335 of *LNCS*, pages 15–24. Springer, 1997. 66

[3] Amir Alush and Jacob Goldberger. Ensemble segmentation using efficient integer linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1966–1977, 2012. 110

[4] Amir Alush and Jacob Goldberger. Break and Conquer: Efficient Correlation Clustering for Image Segmentation. In *Similarity-Based Pattern Recognition*, volume 7953 of *LNCS*, pages 134–147. Springer, 2013. 108, 110, 129

[5] Bjoern Andres, Jörg H Kappes, Thorsten Beier, Ullrich Köthe, and Fred A Hamprecht. Probabilistic image segmentation with closedness constraints. In *International Conference on Computer Vision (ICCV)*, pages 2611–2618. IEEE, 2011. 31, 66, 67, 68, 74, 77, 79, 84, 85, 94, 96, 108, 110, 117, 118, 119, 121, 123, 126, 128, 129, 131

[6] Bjoern Andres, Ullrich Koethe, Thorben Kroeger, and Fred A Hamprecht. How to Extract the Geometry and Topology from Very Large 3D Segmentations. *arxiv.org*, abs/1009.6215, 2010. 48, 51, 54, 58

[7] Bjoern Andres, Ullrich Koethe, Thorben Kroeger, Moritz Helmstaedter, Kevin L Briggman, Winfried Denk, and Fred A Hamprecht. 3D segmentation of SBFSEM images of neuropil by a graphical model over supervoxel boundaries. *Medical Image Analysis*, 16(4):796–805, 2012. 20, 21, 32

[8] Bjoern Andres, Ullrich Köthe, Moritz Helmstaedter, Winfried Denk, and Fred A Hamprecht. Segmentation of SBFSEM Volume Data of Neural Tissue by Hierarchical Classification. In Gerhard Rigoll, editor, *DAGM Symposium*, volume 5096 of *LNCS*, pages 142–152. Springer, 2008. 32

[9] Björn Andres, Thorsten Beier, and Jörg H. Kappes. OpenGM: A C++ Library for Discrete Graphical Models. *arxiv.org*, abs/1206.0111, 2012. 118

[10] {Bjoern Andres, Thorben Kroeger}*, Kevin L Briggman, Winfried Denk, Natalya Korogod, Graham Knott, Ullrich Koethe, and Fred A Hamprecht. Globally Optimal Closed-Surface Segmentation for Connectomics. In *European Conference on Computer Vision (ECCV)*, volume 7574 of *LNCS*, pages 778–791. Springer, 2012. 20, 21, 26, 63, 65, 94, 96, 108, 110, 124, 129, 141

[11] Kenneth Appel and Wolfgang Haken. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21(3):429–490, 1977. 109

[12] Pablo Arbeláez. Boundary Extraction in Natural Images Using Ultrametric Contour Maps. In *Computer Vision and Pattern Recognition Workshop*, pages 182–182. IEEE, 2006. 126

[13] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. 26, 94

[14] Shai Bagon and Meirav Galun. Large Scale Correlation Clustering Optimization. *arxiv.org*, abs/1112.2903, 2011. 109, 110, 111, 118, 128, 129

[15] Shai Bagon and Meirav Galun. A Multiscale Framework for Challenging Discrete Optimization. In *NIPS Workshop on Optimization for Machine Learning*, 2012. 94

[16] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. *Machine Learning*, 56(1-3):89–113, 2004. 67, 94, 99, 108, 129

[17] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36:493–513, 1988. 67

[18] Carlos Becker, Karim Ali, Graham Knott, and Pascal Fua. Learning context cues for synapse segmentation in EM volumes. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 7510 of *LNCS*, pages 585–592. Springer, 2012. 18

[19] Carlos Becker, Karim Ali, Graham Knott, and Pascal Fua. Learning Context Cues for Synapse Segmentation. *IEEE Transactions on Medical Imaging*, 32(10):1864–1877, 2013. 18, 138

[20] {Thorsten Beier, Thorben Kroeger}*, Joerg Kappes, and Fred A Hamprecht. Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014. 21, 63, 107

[21] Johanna Beyer, Ali Al-Awami, Narayanan Kasthuri, Jeff W Lichtman, Hanspeter Pfister, and Markus Hadwiger. ConnectomeExplorer: Query-Guided Visual Analysis of Large Volumetric Neuroscience Data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2868–2877, 2013. 14, 15

[22] Johanna Beyer, Markus Hadwiger, Ali Al-Awami, Won-Ki Jeong, Narayan Kasthuri, Jeff W Lichtman, and Hanspeter Pfister. Exploring the Connectome: Petascale Volume Visualization of Microscopy Data Streams. *IEEE Computer Graphics and Applications*, 33(4):50–61, 2013. 14, 15

[23] Christopher M Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006. 18, 21

[24] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov Random Fields for Vision and Image Processing.* The MIT Press, 2011. 27

[25] Davi D Bock, Wei-Chung Allen Lee, Aaron M Kerlin, Mark L Andermann, Greg Hood, Arthur W Wetzel, Sergey Yurgenson, Edward R Soucy, Hyon Suk Kim, and R Clay Reid. Network anatomy and in vivo physiology of visual cortical neurons. *Nature*, 471(7337):177–182, 2011. 8, 15

[26] John A Bogovic, Gary B Huang, and Viren Jain. Learned versus Hand-Designed Feature Representations for 3d Agglomeration. *arXiv*, 2013. 21, 143

[27] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *European Conference on Computer Vision (ECCV)*, volume 2351 of *LNCS*, pages 109–122. Springer, 2002. 135

[28] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. 129

[29] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001. 27, 127, 128, 129

[30] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 18, 22, 23, 40

[31] Kevin L Briggman and Davi D Bock. Volume electron microscopy for neuronal circuit reconstruction. *Current Opinion in Neurobiology*, 22(1):154–161, 2012. 4, 5, 9, 11

[32] Kevin L Briggman and Winfried Denk. Towards neural circuit reconstruction with volume electron microscopy techniques. *Current Opinion in Neurobiology*, 16(5):562–570, 2006. 9

[33] Kevin L Briggman, Moritz Helmstaedter, and Winfried Denk. Wiring specificity in the direction-selectivity circuit of the retina. *Nature*, 471(7337):183–188, 2011. 6, 76

[34] Dawen Cai, Kimberly B Cohen, Tuanlian Luo, Jeff W Lichtman, and Joshua R Sanes. Improved tools for the Brainbow toolbox. *Nature Methods*, 10(6):540–547, 2013. 5

[35] Albert Cardona. Towards Semi-Automatic Reconstruction of Neural Circuits. *Neuroinformatics*, pages 1–3, 2013. 19, 28

[36] Albert Cardona, Stephan Saalfeld, Stephan Preibisch, Benjamin Schmid, Anchi Cheng, Jim Pulokas, Pavel Tomancak, and Volker Hartenstein. An Integrated Micro-and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy. *PLoS biology*, 8(10):e1000502, 2010. 15

[37] Albert Cardona, Stephan Saalfeld, Johannes Schindelin, Ignacio Arganda-Carreras, Stephan Preibisch, Mark Longair, Pavel Tomancak, Volker Hartenstein, and Rodney J Douglas. TrakEM2 software for neural circuit reconstruction. *PLoS one*, 7(6):e38011, 2012. 14, 15

[38] Dmitri B Chklovskii, Shiv Vitaladevuni, and Louis K Scheffer. Semi-automated reconstruction of neural circuits using electron microscopy. *Current Opinion in Neurobiology*, 20(5):667–675, 2010. 19

[39] Sunil Chopra and MR Rao. The partition problem. *Mathematical Programming*, 59(1-3):87–115, 1993. 66, 67, 71, 75, 94, 95, 108, 109, 110, 128, 129, 131

[40] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Juergen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2852–2860, 2012. 20, 26, 27

[41] Marie-Christine Costa, Lucas Létocart, and Frédéric Roupin. Minimal multicut and maximal integer multiflow: a survey. *European Journal of Operational Research*, 162(1):55–69, 2005. 67

[42] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1362–1374, 2009. 32

[43] R Cameron Craddock, Saad Jbabdi, Chao-Gan Yan, Joshua T Vogelstein, F Xavier Castellanos, Adriana Di Martino, Clare Kelly, Keith Heberlein, Stan Colcombe, and Michael P Milham. Imaging human connectomes at the macroscale. *Nature Methods*, 10(6):524–539, 2013. 1, 10

[44] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2&#8211;3):81–227, February 2012. 22, 23

[45] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989. 24

[46] Juan A. de Carlos and José Borrell. A historical reflection of the contributions of Cajal and Golgi to the foundations of neuroscience. *Brain Research Reviews*, 55(1):8 – 16, 2007. 2

[47] Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2):172–187, 2006. 67

[48] Winfried Denk, Kevin L Briggman, and Moritz Helmstaedter. Structural neurobiology: missing link to a mechanistic understanding of neural computation. *Nature Reviews Neuroscience*, 13(5):351–358, 2012. 5, 6, 10

[49] Winfried Denk and Heinz Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS biology*, 2(11):e329, 2004. 8, 76

[50] Michel M Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15. Springer, 1997. 110

[51] Mark Everingham, Luc Van Gool, Christopher K Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 126

[52] John C Fiala. Reconstruct: a free editor for serial section microscopy. *Journal of Microscopy*, 218(1):52–61, 2005. 14, 15

[53] Thomas Finley and Thorsten Joachims. Supervised clustering with support vector machines. In *International Conference on Machine Learning (ICML)*, pages 217–224. ACM, 2005. 94, 110

[54] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision (ICCV)*, pages 670–677. IEEE, 2009. 31

[55] Jan Funke, Bjoern Andres, Fred A Hamprecht, Albert Cardona, and Matthew Cook. Efficient automatic 3D-reconstruction of branching neurons from EM data. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1004–1011. IEEE, 2012. 20, 29

[56] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979. 67

[57] Martin Grötschel and Yoshiko Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1-3):59–96, 1989. 66, 67, 74

[58] Kristen M Harris and Richard J Weinberg. Ultrastructure of Synapses in the Mammalian Brain. *Cold Spring Harbor Perspectives in Biology*, 4(5), 2012. 3, 11, 15

[59] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer, 2009. 21, 22, 23

[60] Ken J Hayworth, Narayanan Kasthuri, Richard Schalek, and Jeff W Lichtman. Automating the Collection of Ultrathin Serial Sections for Large Volume TEM Reconstructions. *Microscopy and Microanalysis*, 12:86–87, 8 2006. 8

[61] Moritz Helmstaedter. Cellular-resolution connectomics: challenges of dense neural circuit reconstruction. *Nature Methods*, 10(6):501–507, 2013. 4, 9, 10, 11, 17, 19

[62] Moritz Helmstaedter, Kevin L Briggman, and Winfried Denk. High-accuracy neurite reconstruction for high-throughput neuroanatomy. *Nature Neuroscience*, 14(8):1081–1088, 2011. 11, 14, 16, 35, 76

[63] Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, Sebastian Seung, and Winfried Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013. 6, 8, 9, 11, 16, 19

[64] Gary B Huang and Viren Jain. Deep and Wide Multiscale Recursive Networks for Robust Image Labeling. *arXiv*, 2013. 20, 27

[65] Vignesh Jagadeesh, James Anderson, Bryan Jones, Robert Marc, Steven Fisher, and BS Manjunath. Synapse classification and localization in Electron Micrographs. *Pattern Recognition Letters*, 2013. 18

[66] Viren Jain, Benjamin Bollmann, Mark Richardson, Daniel R Berger, Moritz Helmstaedter, Kevin L Briggman, Winfried Denk, Jared B Bowden, John M Mendenhall, Wickliffe C Abraham Abraham, Kristen M. Harris, Narayanan Kasthuri, Ken J Hayworth, Richard Schalek, Juan Carlos Tapia, Jeff W Lichtman, and Sebastian Seung. Boundary learning by optimization with topological constraints. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2488–2495. IEEE, 2010. 20

[67] Viren Jain, Joseph F Murray, Fabian Roth, Srinivas Turaga, Valentin Zhigulin, Kevin L Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. Supervised learning of image restoration with convolutional networks. In *International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007. 19, 24, 25

[68] Viren Jain, Sebastian Seung, and Srinivas C Turaga. Machines that learn to segment images: a crucial technology for connectomics. *Current Opinion in Neurobiology*, 20(5):653–666, 2010. 19, 20

[69] Viren Jain, Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and Sebastian Seung. Learning to agglomerate superpixel hierarchies. In *Advances in Neural Information Processing Systems (NIPS)*. 2011. 20, 21, 32, 95, 143

[70] Won-Ki Jeong, Johanna Beyer, Markus Hadwiger, Rusty Blue, Charles Law, Amelio Vázquez-Reina, R Clay Reid, Jeff Lichtman, and Hanspeter Pfister. Ssecrett and neurotrace: interactive visualization and analysis tools for large-scale neuroscience data sets. *IEEE Computer Graphics and Applications*, 30(3):58–70, 2010. 14, 15

[71] Won-Ki Jeong, Johanna Beyer, Markus Hadwiger, Amelio Vázquez, Hanspeter Pfister, and Ross T Whitaker. Scalable and interactive segmentation and visualization of neural processes in EM datasets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1505–1514, 2009. 14, 15

[72] Cory Jones, Ting Liu, Mark Ellisman, and Tolga Tasdizen. Semi-automatic neuron segmentation in electron microscopy images via sparse labeling. In *International Symposium on Biomedical Imaging (ISBI)*, pages 1304–1307. IEEE, 2013. 17

[73] Cory Jones, Mojtaba Seyedhosseini, Mark Ellisman, and Tolga Tasdizen. Neuron segmentation in electron microscopy images using partial differential equations. In

International Symposium on Biomedical Imaging (ISBI), pages 1457–1460. IEEE, 2013. 28

[74] Elizabeth Jurrus, Antonio RC Paiva, Shigeki Watanabe, James R Anderson, Bryan W Jones, Ross T Whitaker, Erik M Jorgensen, Robert E Marc, and Tolga Tasdizen. Detection of neuron membranes in electron microscopy images using a serial neural network architecture. *Medical Image Analysis*, 14(6):770–783, 2010. 20, 27

[75] Elizabeth Jurrus, Shigeki Watanabe, Richard J Giuly, Antonio RC Paiva, Mark H Ellisman, Erik M Jorgensen, and Tolga Tasdizen. Semi-Automated Neuron Boundary Detection and Nonbranching Process Segmentation in Electron Microscopy Images. *Neuroinformatics*, 11(1):5–29, 2013. 20

[76] Elizabeth Jurrus, Shigeki Watanabe, Richard J Giuly, Antonio RC Paiva, Mark H Ellisman, Erik M Jorgensen, and Tolga Tasdizen. Semi-automated neuron boundary detection and nonbranching process segmentation in electron microscopy images. *Neuroinformatics*, 11(1):5–29, 2013. 21, 27, 29, 32, 143

[77] Eric Kandel, James Schwartz, and Thomas Jessell. *Principles of neural science*, volume 3. Appleton & Lange, 1991. 3, 4

[78] Jörg H Kappes, Bjoern Andres, Fred A Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X Kausler, Jan Lellmann, Nikos Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1328–1335, 2013. 119, 128, 129

[79] Jörg Hendrik Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schnörr. Globally optimal image partitioning by multicuts. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 31–44. Springer, 2011. 67, 94, 110, 127, 128, 129, 130, 131

[80] Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. Higher-order Segmentation via Multicuts. *arxiv.org*, abs/1305.6387, 2013. 110, 118, 128, 129, 130, 131, 132

[81] Bernhard X. Kausler. *Tracking-by-Assignment as a Probabilistic Graphical Model with Applications in Developmental Biology.* PhD thesis, University of Heidelbeg, 2013. 42, 44

[82] Verena Kaynig, Thomas Fuchs, and Joachim M Buhmann. Neuron geometry extraction by perceptual grouping in sstem images. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2902–2909. IEEE, 2010. 20, 28

[83] Verena Kaynig, Thomas J Fuchs, and Joachim M Buhmann. Geometrical Consistent 3D Tracing of Neuronal Processes in ssTEM Data. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 6362 of *LNCS*, pages 209–216. Springer, 2010. 20

[84] Verena Kaynig, Amelio Vázquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister. Large-Scale Automatic Reconstruction of Neuronal Processes from Electron Microscopy Images. *arXiv*, 2013. 17, 20, 26, 28, 29, 40

[85] Brian W Kernighan and Shen Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49(1):291–307, 1970. 110, 111, 118

[86] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D Yoo. Higher-order correlation clustering for image segmentation. In *Advances in Neural Information Processing Systems (NIPS)*, 2011. 67, 110

[87] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D Yoo. Task-Specific Image Partitioning. *IEEE Transactions on Image Processing*, 22(2):488–500, 2013. 31, 94, 95, 110, 129

[88] Max Knoll and Ernst Ruska. Das Elektronenmikroskop. *Zeitschrift für Physik*, 78(5-6):318–339, 1932. 5

[89] Graham Knott and Christel Genoud. Is EM dead? *Journal of Cell Science*, 126(20):4545–4552, 2013. 9

[90] Graham Knott, Herschel Marchman, David Wall, and Ben Lich. Serial Section Scanning Electron Microscopy of Adult Brain Tissue Using Focused Ion Beam Milling. *Journal of Neuroscience*, 28(12):2959–2964, 2008. 9, 81

[91] Pushmeet Kohli, Alexander Shekhovtsov, Carsten Rother, Vladimir Kolmogorov, and Philip Torr. On partial optimality in multi-label MRFs. In *International Conference on Machine Learning (ICML)*, pages 480–487. ACM, 2008. 129

[92] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006. 127

[93] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004. 18, 27, 111, 127, 128, 129

[94] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011. 127, 129

[95] Nikos Komodakis and Georgios Tziritas. Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, 2007. 129

[96] VA Kovalevsky. Finite Topology as Applied to Image Analysis. In *Computer Vision, Graphics, and Image Processing*, pages 141–161, 1989. 48, 51

[97] Anna Kreshuk, Christoph N Straehle, Christoph Sommer, Ullrich Koethe, Marco Cantoni, Graham Knott, and Fred A Hamprecht. Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images. *PLoS one*, 6(10):e24899, 2011. 18, 138

[98] Thorben Kroeger, Shawn Mikula, Winfried Denk, Ullrich Koethe, and Fred A Hamprecht. Learning to Segment Neurons with non-local Quality Measures. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 8150 of *LNCS*, pages 419–427. Springer, 2013. 21, 26, 63, 93, 108, 129, 141, 142

[99] Ritwik Kumar, Amelio Vázquez-Reina, and Hanspeter Pfister. Radon-like features and their application to connectomics. In *Computer Vision and Pattern Recognition Workshops*, pages 186–193. IEEE, 2010. 20

[100] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *International Conference on Computer Vision (ICCV)*, pages 277–284. IEEE, 2009. 67

[101] Ting Liu, Elizabeth Jurrus, Mojtaba Seyedhosseini, Mark Ellisman, and Tolga Tasdizen. Watershed merge tree classification for electron microscopy image segmentation. In *International Conference on Pattern Recognition (ICPR)*, pages 133–137. IEEE, 2012. 21, 26, 143

[102] Jean Livet, Tamily A Weissman, Hyuno Kang, Ju Lu, Robyn A Bennis, Joshua R Sanes, and Jeff W Lichtman. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature*, 450(7166):56–62, 2007. 5

[103] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-based segmentation of mitochondria in EM image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, 31(2):474–486, 2012. 18, 31, 81, 138

[104] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Vincent Lepetit, and Pascal Fua. A Fully Automated Approach to Segmentation of Irregularly Shaped Cellular Structures in EM Images. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 6362 of *LNCS*, pages 463–471. Springer, 2010. 18, 31

[105] Bohumil Maco, Anthony Holtmaat, Marco Cantoni, Anna Kreshuk, Christoph N Straehle, Fred A Hamprecht, and Graham W Knott. Correlative in vivo 2 photon and focused ion beam scanning electron microscopy of cortical neurons. *PLoS one*, 8(2):e57405, 2013. 17

[106] Michael Maire, Pablo Arbeláez, Charless Fowlkes, and Jitendra Malik. Using Contours to Detect and Localize Junctions in Natural Images. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. 129, 135

[107] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423. IEEE, 2001. 68, 118, 126, 135

[108] Jonathan Masci, Alessandro Giusti, Dan Ciresan, Jonathan Masci, Luca M Gambardella, and Juergen Schmidhuber. Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks. In *International Conference on Image Processing (ICIP)*, 2013. 20, 27

[109] Marina Meilă. Comparing Clusterings by the Variation of Information. In *Learning Theory and Kernel Machines*, volume 2777 of *LNCS*, pages 173–187. Springer, 2003. 119, 123

[110] Marina Meilă. Comparing clusterings – an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007. 36, 77, 81, 94

[111] Kristina D Micheva and Stephen J Smith. Array tomography: a new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55(1):25–36, 2007. 4

[112] Shawn Mikula, Jonas Binding, and Winfried Denk. Staining and embedding the whole mouse brain for electron microscopy. *Nature Methods*, 9(12):1198–1201, 2012. 6

[113] Yuriy Mishchenko, Tao Hu, Josef Spacek, John Mendenhall, Kristen M Harris, and Dmitri B Chklovskii. Ultrastructural analysis of hippocampal neuropil from the connectomics perspective. *Neuron*, 67(6):1009–1020, 2010. 11, 15

[114] Joshua L Morgan and Jeff W Lichtman. Why not connectomics? *Nature Methods*, 10(6):494–500, 2013. 4, 10

[115] Saket Navlakha, Joseph Suhan, Alison L Barth, and Ziv Bar-Joseph. A high-throughput framework to detect synapses in electron microscopy images. *Bioinformatics*, 29(13):i9–i17, 2013. 18

[116] Sebastian Nowozin and Stefanie Jegelka. Solution stability in linear programming relaxations: Graph partitioning and unsupervised learning. In *International Conference on Machine Learning (ICML)*, pages 769–776. ACM, 2009. 67

[117] Sebastian Nowozin and Christoph H Lampert. Global connectivity potentials for random field models. In *Computer Vision and Pattern Recognition (CVPR)*, pages 818–825. IEEE, 2009. 67

[118] Sebastian Nowozin and Christoph H Lampert. *Structured Learning and Prediction in Computer Vision*. Now Publishers Inc, 2011. 95, 96

[119] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. Machine Learning of Hierarchical Clustering to Segment 2D and 3D Images. *PLoS one*, 8(8):e71715, 2013. 20, 21, 26, 32, 143

[120] Donald J Olbris. FlyEM: Raveler. https://openwiki.janelia.org/wiki/display/flyem/Raveler, 2013. 14, 16

[121] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, 2006. 135

[122] Pavel Osten and Troy W Margrie. Mapping brain circuitry with a light microscope. *Nature Methods*, 10(6):515–523, 2013. 5

[123] Federico Perazzi, Philipp Krähenbühl, Yael Pritch, and Alexander Hornung. Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 733–740. IEEE, 2012. 135

[124] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. 26, 35, 77, 81, 94

[125] Mani Ranjbar, Greg Mori, and Yang Wang. Optimizing Complex Loss Functions in Structured Prediction. In *European Conference on Computer Vision (ECCV)*, volume 6312 of *LNCS*, pages 580–593. Springer, 2010. 95

[126] Zhile Ren and Gregory Shakhnarovich. Image Segmentation by Cascaded Region Agglomeration. pages 2011–2018, 2013. 26, 143

[127] Mike Roberts, Won-Ki Jeong, Amelio Vázquez-Reina, Markus Unger, Horst Bischof, Jeff Lichtman, and Hanspeter Pfister. Neural Process Reconstruction from Sparse User Scribbles. In Gabor Fichtinger, Anne Martel, and Terry Peters, editors, *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 6891 of *LNCS*, pages 621–628. Springer, 2011. 17

[128] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004. 27, 127, 128, 129

[129] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. 111

[130] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 25

[131] Michael J Rust, Mark Bates, and Xiaowei Zhuang. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM). *Nature Methods*, 3(10):793–796, 2006. 4

[132] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomančák. CATMAID: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–1986, 2009. 14, 15

[133] Stephan Saalfeld, Richard Fetter, Albert Cardona, and Pavel Tomančák. elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature Methods*, 9(7):717–720, 2012. 8

[134] Nicol N Schraudolph and Dmitry Kamenetsky. Efficient exact inference in planar Ising models. In *Advances in Neural Information Processing Systems (NIPS)*. 2008. 111

[135] Sebastian Seung. *Connectome – How the Brain's Wiring Makes Us Who We Are.* Mariner Books, 2013. 1, 2, 3, 4, 8, 10

[136] Mojtaba Seyedhosseini, Mark H Ellisman, and Tolga Tasdizen. Segmentation of mitochondria in electron microscopy images using algebraic curves. In *International Symposium on Biomedical Imaging (ISBI)*, pages 860–863. IEEE, 2013. 18

[137] Mojtaba Seyedhosseini, Ritwik Kumar, Elizabeth Jurrus, Rick Giuly, Mark Ellisman, Hanspeter Pfister, and Tolga Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images Using Multi-scale Context and Radon-Like Features. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 6891 of *LNCS*, pages 670–677. Springer, 2011. 20

[138] Mojtaba Seyedhosseini, Mehdi Sajjadi, and Tolga Tasdizen. Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *International Conference on Computer Vision (ICCV)*. IEEE, 2013. 20

[139] Stephen J Smith. Circuit Reconstruction Tools Today. *Current Opinion in Neurobiology*, 17(5):601–608, 2007. 4

[140] Christoph Sommer, Christoph Straehle, Ullrich Köthe, and Fred A Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *International Symposium on Biomedical Imaging (ISBI)*, pages 230–233. IEEE, 2011. 14, 16, 20, 40, 90, 139

[141] David Sontag and Tommi Jaakkola. New Outer Bounds on the Marginal Polytope. In *Advances in Neural Information Processing Systems (NIPS)*, 2008. 67

[142] Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS computational biology*, 1(4):e42, 2005. 10

[143] Christoph Straehle, Ullrich Koethe, Graham Knott, Kevin Briggman, Winfried Denk, and Fred A Hamprecht. Seeded watershed cut uncertainty estimators for guided interactive segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 765–772. IEEE, 2012. 16, 17, 46

[144] Christoph Straehle, Ullrich Köthe, Graham Knott, and Fred A Hamprecht. Carving: Scalable Interactive Segmentation of Neural Volume Electron Microscopy Images. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 6891 of *LNCS*, pages 653–660. Springer, 2011. 16, 17, 32, 46, 88

[145] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for Markov random fields. In *European Conference on Computer Vision (ECCV)*, volume 3952 of *LNCS*, pages 16–29. Springer, 2006. 128

[146] Shin-ya Takemura, Arjun Bharioke, Zhiyuan Lu, Aljoscha Nern, Shiv Vitaladevuni, Patricia K Rivlin, William T Katz, Donald J Olbris, Stephen M Plaza, Philip Winston, Ting Zhao, Jane Anne Horne, Richard D. Fetter, Satoko Takemura, Katerina Blazek, Lei-Ann Chang, Omotara Ogundeyi, Mathew A. Saunders, Victor Shapiro, Christopher Sigmund, Gerald M. Rubin, Louis K. Scheffer, Ian A. Meinertzhagen, and Dmitri B. Chklovskii. A visual motion detection circuit suggested by Drosophila connectomics. *Nature*, 500(7461):175–181, 2013. 8, 11, 32

[147] Daniel Tarlow and Richard S. Zemel. Structured Output Learning with High Order Loss Functions. In *International conference on Artificial Intelligence and Statistics (AISTATS)*, 2012. 95

[148] Boray F Tek, Thorben Kroeger, Shawn Mikula, and Fred A Hamprecht. Automated Cell Nuclei Detection for Large-Volume Electron Microscopy of Neural Tissue. In *International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2014. 39, 61, 63, 86

[149] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and Sebastian Seung. Maximin affinity learning of image segmentation. In *Advances in Neural Information Processing Systems (NIPS)*. 2009. 20, 25, 26, 95

[150] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010. 20

[151] Amelio Vázquez-Reina, Michael Gelbart, Daniel Huang, Jeff Lichtman, Eric Miller, and Hanspeter Pfister. Segmentation fusion for connectomics. In *International Conference on Computer Vision (ICCV)*, pages 177–184. IEEE, 2011. 20, 29

[152] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. 28, 67, 127, 128, 129

[153] Shiv N Vitaladevuni and Ronen Basri. Co-clustering of image segments using convex optimization applied to EM neuronal reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2203–2210. IEEE, 2010. 20, 94

[154] Randle W Ware and Vincent LoPresti. Three-dimensional reconstruction from serial sections. *International Review of Cytology*, 40:325–440, 1975. 7

[155] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 314(1165):1–340, 1986. 8, 9, 11, 16

[156] Wikimedia Commons. Drawing of Purkinje cells and granule cells from pigeon cerebellum by Santiago Ramón y Cajal, 1899. File: `PurkinjeCell.jpg`. 2

[157] Meng Xu, Travis A Jarrell, Yi Wang, Steven J Cook, David H Hall, and Scott W Emmons. Computer Assisted Assembly of Connectomes from Electron Micrographs: Application to Caenorhabditis elegans. *PLoS one*, 8(1):e54050, 2013. 11, 14, 16

[158] Huei-Fang Yang and Yoonsuck Choe. An interactive editing framework for electron microscopy image segmentation. In *International Symposium on Advances in Visual Computing (ISVC)*, pages 400–409. Springer, 2011. 17

[159] Julian Yarkony, Alexander Ihler, and Charless C Fowlkes. Fast Planar Correlation Clustering for Image Segmentation. In *European Conference on Computer Vision (ECCV)*, volume 7577 of *LNCS*, pages 568–581. Springer, 2012. 31, 94, 96, 108, 110, 118, 119, 120, 123, 126, 129

[160] Paul A Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006. 14, 16

[161] Lei Zhang and Qiang Ji. Image segmentation with a unified graphical model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1406–1425, 2010. 128

[162] Liang Zhao, Hiroshi Nagamochi, and Toshihide Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Mathematical Programming*, 102(1):167–183, 2005. 110