DISS. ETH NO. 22125

Automatic Neuron Reconstruction from Anisotropic Electron Microscopy Volumes

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES OF ETH ZURICH

(Dr. sc. ETH Zurich)

presented by
JAN FUNKE
Dipl.-inf. TU Dresden
born on 18.03.1985
citizen of Germany

accepted on the recommendation of

Prof. Richard Hahnloser
Prof. Fred Hamprecht
Dr. Matthew Cook
Dr. Albert Cardona

2014

Contents

Abstract

The work presented in this thesis addresses the problem of the automatic extraction of the wiring diagram of a nervous system from anisotropic electron microscopy volumes with high x- and y-resolution but low z-resolution, as obtained by serial section electron microscopy imaging procedures. A necessary step towards this goal is the segmentation of neural tissue to separate neuron cell interior from membrane and extracellular space, and thus reveal the 3D shape of each neuron, a process called *neuron reconstruction*.

The core of this thesis is a novel method for the reconstruction of neurons from serial section electron microscopy images. Due to the anisotropy of serial section imaging methods, we treat the data as a stack of 2D images, rather then a continuous 3D volume. However, the detection of neuron slices (*i.e.*, cross-sections of neural processes) in 2D images is difficult due to ambiguities in the data. Therefore, we propose to enumerate several diverse and possibly contradictory candidate neuron slices by identifying separating membranes with varying thresholds for each image individually. Between candidates of adjacent images in the stack, we enumerate assignments that reflect possible ways to follow a neural process from one image to another. We assign costs to each candidate and assignment and formulate constraints that ensure consistency between the assignments. We show how a globally cost-minimal segmentation of neuron slices and assignments between images can be found jointly and efficiently. Furthermore, we derive a structured learning formulation to learn the assignment costs from annotated ground truth and show its effectiveness compared to other methods.

Since the candidate selection is a crucial step in our model, we also introduce an alternative candidate generation method that samples candidates from a conditional random field (CRF) based on convolutional neural network predictions. The CRF is designed and trained to capture the statistics of 2D electron microscopy images of neural tissue. We show that sampling from this model produces plausible neuron slice candidates that are well suited for our reconstruction method, while additionally providing labels for synapse, glia cells, and mitochondria.

For the application to very large datasets, inference has to be distributed. However, since our model performs a global optimization, this is not trivial. We tackle this problem by presenting a distribution scheme for our model that is based on dual decomposition and guarantees global optimality. For that, the original problem is decomposed into several regions that communicate with each other to find an agreement. If such an agreement can be found, the collected answers from all regions is provably optimal. We introduce a messaging strategy that ensures that such an agreement can always be found under suitable assumptions.

Finally, we review error measures used for neuron reconstruction algorithms and discuss their properties. We introduce a new measure that reflects the edit distance between a reconstruction and a ground truth within certain tolerated variations and compare it to existing measures.

Given the extremely high accuracy requirements for biological use cases and the challenging ambiguities encountered in EM images, the complete automatic reconstruction of neurons is still out of reach. Nevertheless, we believe that the methods introduced in this thesis made a significant contribution towards this goal and can already be used to assist the tedious manual reconstruction.

Zusammenfassung

Die Arbeit, die in dieser Dissertation vorgestellt wird, befasst sich mit dem Problem der automatischen Extraktion neuronaler Schaltkreise aus anisotropen Elektron-Mikroskopie-Volumina. Diese Volumina, die aus Elektron-Mikroskopie-Bildern von seriellen Schnitten neuronalen Gewebes zusammengesetzt sind, haben eine hohe x- und y-Auflösung, jedoch eine niedrige z-Auflösung. Ein notwendiger Schritt, um diese Schaltkreise zu erhalten, ist die Segmentierung von neuronalem Gewebe, so dass das Zellinnere der Neuronen von Membranen und extrazellulärem Raum getrennt wird. Somit wird das 3D Profil eines jeden Neurons offen gelegt – ein Prozess der als *Neuronenrekonstruktion* bezeichnet wird.

Der Kern dieser Arbeit ist ein neues Verfahren für die Neuronenrekonstruktion aus Elektron-Mikroskopie-Bildern von seriellen Schnitten neuronalen Gewebes. Auf Grund der Anisotropie dieses bildgebenden Verfahrens betrachten wir das resultierende 3D Bild als einen Block aus 2D Bildern, statt eines kontinuierlichen 3D Volumens. Die Erkennung von Neuronenscheiben (d.h., Querschnitte von neuronalen Fortsätzen) in 2D Bildern ist jedoch aufgrund der Mehrdeutigkeiten in den Daten schwierig. Daher schlagen wir vor, verschiedene und möglicherweise widersprüchliche Neuronenscheiben-Kandidaten zu enumerieren, indem separierende Membranen mit unterschiedlichen Schwellwerten identifiziert werden. Zwischen den Kandidaten aufeinanderfolgender Bilder enumerieren wir Verbindungen, welche verschiedene Möglichkeiten reprsentieren einem Neuron von einem Bild zum Nächsten zu folgen. Wir weisen jedem Kandidaten und jeder Verbindung Kosten zu und formulieren Bedingungen, die Konsistenz zwischen den Verbindungen sicherstellen. Wir zeigen, wie eine global kostenminimale Segmentierung von Neuronenscheiben und Verbindungen zwischen den Bildern gemeinsam und effizient gefunden werden kann. Des Weiteren leiten wir eine strukturelle Lernformulierung für das Lernen der Verbindungskosten anhand annotierter Beispiele her und zeigen dessen Effektivität im Vergleich mit anderen Methoden.

Da die Kandidatenauswahl in unserer Methode ein entscheidender Schritt ist, führen wir zusätzlich ein alternatives Verfahren ein, welches Kandidaten als Stichproben eines *conditional random fields* (CRF) generiert. Dieses CRF wurde entworfen und trainiert um die Statistiken von 2D Elektron-Mikroskopie-Bildern von neuronalem Gewebe zu reproduzieren. Wir zeigen, dass die Stichproben von diesem Modell plausible Kandidaten produzieren, welche gut geeignet für unser Neuronenrekonstruktions-Verfahren sind und zusätzlich Annotationen für Synapsen, Glia-Zellen und Mitochondrien liefern.

Für die Anwendung auf sehr großen Datensätzen muss die Inferenz verteilt werden. Da unser Verfahren jedoch eine globale Optimierung durchführt, ist diese Verteilung nicht trivial. Wir begegnen diesem Problem indem wir ein Verteilschema präsentieren, welches auf der dualen Dekomposition basiert und globale Optimiliät garantiert. Dafür wird das originale Problem in mehrere Regionen aufgeteilt, welche miteinander kommunizieren um eine Übereinstimmung zu finden. Falls eine solche Übereinstimmung gefunden wird, ist die gemeinsame Lösung aller Regionen bewiesenermaßen optimal. Wir führen eine Kommunikationsstrategie zwischen den Regionen ein, die sicher stellt, dass eine solche Übereinstimmung unter geeigneten Annahmen gefunden wird.

Wir schließen diese Arbeit mit einer Studie über Fehlermaße, die für die Neuronenrekonstruktion benutzt werden und diskutieren deren Eigenschaften. Wir führen ein neues Maß ein, welches die Entfernung zwischen einer Rekonstruktion und einem manuell annotierten Beispiel anhand Veränderungsschritte misst, dabei aber gewisse Variationen toleriert, und vergleichen diesen Maß mit Existierenden.

Unter Berücksichtigung der extrem hohen Anforderungen an die Fehlerfreiheit für biologische Anwendungsfälle und der Mehrdeutigkeiten in EM-Bildern ist die komplett automatische Neuronenrekonstruktion noch nicht in Greifweite. Nichtsdestotrotz glauben wir, dass die Verfahren, die wir in dieser Arbeit vorstellen, einen signifikanten Beitrag zum Erreichen dieses Ziels leisten und bereits jetzt genutzt werden können um die ermüdende manuelle Rekonstruktion zu erleichtern.

Acknowledgements

Foremost, I would like to thank Matthew Cook, who's group I joined at INI. Matthew's contribution to my work can not be emphasized enough – I am grateful for the many fruitful discussions that we had, out of which I mostly left with new ideas, and only rarely with confusion. Matthew's constant support and friendship helped me to stay positive in uneasy times and the freedom that he gave me helped to reduce those times to a minimum.

I would also like to thank Fred Hamprecht, who invited me to visit his group in Heidelberg that soon became my second "home". Not only did I enjoy the exceptionally friendly atmosphere in his group, but also the high density of knowledge and smartness that I encountered in discussions with its members. In particular, I would like to thank Björn Andres and Bernhard Kausler for infecting me with their excitement about discrete optimization methods and structured learning, which obviously left a trace in this thesis.

Whenever I got too excited about the beauty of theory, it was Albert Cardona who grounded me by pointing out what the neuroscience community really needs. Albert's enthusiasm about our project is one of the main driving forces for me. Like no one else, Albert gave me the feeling how important our work is. Also his suggestions as an expert neuron tracer were of great value and influenced the methods developed in this thesis.

A special thank goes to Julien Martel, who joined our group as a PhD student a year ago and put a happy end to my peer isolation at INI. Although not his main project, Julien dedicated himself and his rich background in computer vision and deep learning techniques to our project. This dedication was not just very fruitful, but also increased the level of fun in our lab by several orders of magnitude. Many coffees have been drunk already over inspiring conversations, and I hope there will be many more coffees to drink.

Equally, I would like to thank Stephan Gerhard for his great commitment and his ability to connect people. A lot of experiments would not have been carried out and a lot of code would not have been written without Stephan's initiative. It is thanks to Stephan's first prototype that we know how to best make the methods presented in this thesis available to neuroscientists. Stephan is also our single point of contact for our collaboration with Dan Ciresan, Alessandro Giusti, and Luca Gambardella, who I would like to thank here as well for their contributions in improving and training deep networks for our application.

I would also like to thank my official supervisors Rodney Douglas and later Richard Hahnloser for their interest in my project and their valuable comments.

Many more people contributed to my work with discussions and support. I am grateful for all the interest and comments I received from the CO2 group at INI (thank you Cosimo Riday, Jakob Buhmann, Thanuja Ambegoda, Peter Diehl, Turlough Neary, Jonas Klein, and Mitra Javadzadeh), for the enjoyable collaboration with the MPI-CBG in Dresden (thank you Florian Jug, Tobias Pietzsch, and Dagmar Kainmüller) and the HCI Heidelberg (thank you Thorben Kröger, Anna Kreshuk, Martin Schiegg, Carsten Haubold, Ullrich Köthe, and many more), for the interesting visit in the VCG Group in Harvard (thank you Hanspeter Pfister, Verena Kaynig-Fittkau, Seymore Knowles-Barley, and Amelio Vazquez-Reina), as well as the fruitful work with the Janelians (thank you Stephan Saalfeld, Tom Kazimiers, and Larry Lindsey).

Very much appreciated was the support and interest from my friends in Zürich and Dresden and my family that kept very positive although they did not always understand what all these flies are about and how to make a living out of it.

Finally, I would like to thank my wife Sharon for her constant support and love, and for making my life outside the lab most enjoyable.

Chapter 1

Introduction[1]

Decades of research in computer vision have provided us with life-changing and impressive automatic systems. Today, we take it for granted that even point-and-shoot cameras automatically detect faces and focus on them. Mail-sorting systems have almost no problems determining the receiver of a hand-written letter. Our cars can detect traffic signs and oncoming traffic to warn us or control the upper beam headlights. Game consoles recognize players and allow them to interact with body motion only. In some cases, automatic systems outperform humans. This is especially true for monotonic tasks that require constant attention.

The MNIST dataset of hand-written digits [67] impressively shows the capabilities of computer vision algorithms. The best approach as of today is a convolutional neural network from Ciresan et al. [19]. It can correctly classify 99.77% of a set of never seen digits. Despite being a very specific task in a controlled environment, and despite the long time that researchers tried to improve the accuracy on the MNIST dataset – if we had the same accuracy for the automatic detection of neurons in 2D electron microscopy images, it would not be good enough[2].

The automatic detection of neurons and synapses from electron microscopy images is a problem much more challenging than digit recognition. First, the accuracy requirements are extremely high: Already a fraction of a percent of errors in the detection of neurons in electron microscopy images renders the result useless for biologists. Second, we have to deal with large amounts of data. Even the small brain of the fruit fly larva results in multiple terabytes of images. Third, the data obtained from electron microscopy is very noisy and contains a lot of ambiguous situations that can only be resolved by experts with domain specific priors.

These are properties of a lot of problems in computer vision, for which no general solution exists. Be it the interpretation of aerial images; the detection of structures in biomedical images coming from MRT, CT, X-ray, or light microscopy; landmark detection and mapping from moving cameras for autonomous agents; or object recognition and scene understanding for augmented reality – we always have to deal with noisy data, missing information, and domain specific prior knowledge to resolve ambiguities. Most of the times, solutions need to be efficient and very accurate, as well.

Addressing the problem of automatically detecting neurons and synapses in electron microscopy images can therefore teach us a lot about computer vision in general. It is clear that any solution will have to deal with the aforementioned challenges. Thus, it is very likely that a satisfying solution will be applicable to other problems with similar challenges

[1] Parts of this chapter are taken from our publication [26] and the research plan for this thesis.

[2] As an example, a local interneuron of a *Drosophila melanogaster* 3rd-instar larva has a cable length of about $320\mu m$. With current imaging techniques, this means that about 8000 2D neuron cross-sections have to be identified to reconstruct this neuron from electron microscopy images. Assuming a precision of 99.77%, we would still have to expect about 18 errors per neuron – enough to drastically change the obtained connectivity graph.

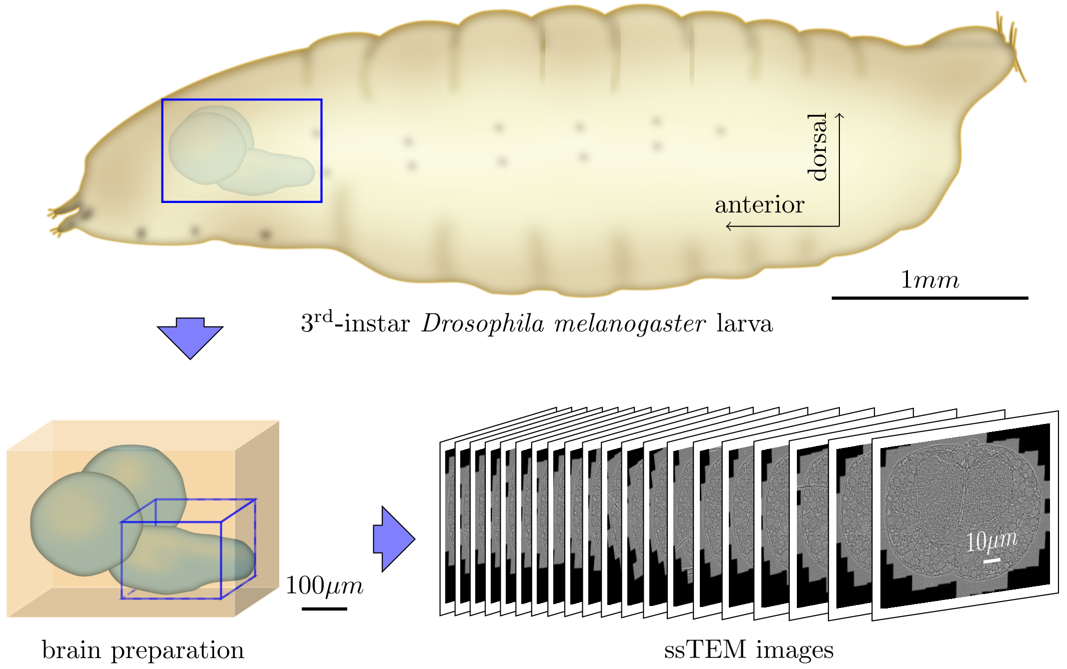


Figure 1.1: Sample preparation and imaging for the 3rd-instar *Drosophila melanogaster* larva (larva shown in top figure, head is left). The brain of the larva (highlighted with a blue rectangle) is manually isolated from the body and embedded in a block of resin (bottom left). With a diamond cutter, about 40*nm* thick sections are scraped off this block and imaged individually with a serial section transmission electron microscope (ssTEM) that delivers an x-y-resolution of about 4*nm*. The retrieved images are stitched and aligned to obtain a stack of 2D images[3] (bottom right). The example shows a subset of EM images from the ventral nerve chord (highlighted with a blue box in the brain preparation), each having a resolution of 25000×18000 pixels.

as well. In this sense, this problem is an interesting use case to assess the performance of current computer vision techniques and to develop new, general methods.

At the same time, solving the problem of the detection of neurons and synapses in electron microscopy images is a necessary step towards further insights in neuroscience. Obtaining the complete neuronal wiring diagram is an important landmark towards a better understanding of the functional capabilities of a nervous system. The comparatively sparse data about the connectivity that is available today is already used to formulate constraints for modelling neural circuits [22, 96]. In order to obtain larger and more detailed wiring diagrams for further investigations, millimeter-sized volumes of neural tissue are currently imaged at nanometer resolution to resolve synapses and thin axons and dendrites [22, 53].

Reconstructing whole nervous systems by hand – that is, identifying axons, dendrites, and synaptic connections – is very impractical. Currently, only selected circuits or single specimen of small animals like the *Drosophila melanogaster* larva (see Figure 1.1) or *C. elegans* are reconstructed manually [7, 12, 13, 15, 17, 36, 37, 80]. As an example, Figure 1.5b shows a single reconstructed neuron from light microscopy, where synapses can not be resolved due to resolution limitations. Reconstructing this neuron took weeks of labor for an expert neuroscientist. But even a *Drosophila melanogaster* brain has 10^5 neurons. High enough resolution electron microscopy images to allow the reconstruction of synapses would have a size of several hundreds of terabytes in order to cover the whole nervous system. With this amount of image data, a complete manual reconstruction of even a small organism like the *Drosophila melanogaster* exceeds the amount of time a neuroscientist has in their

[3]All ssTEM images used in this chapter are from the 3rd-instar ventral nerve cord of the *Drosophila melanogaster* larva, sectioned and imaged by Richard Fetter and the FlyEM project at Janelia.

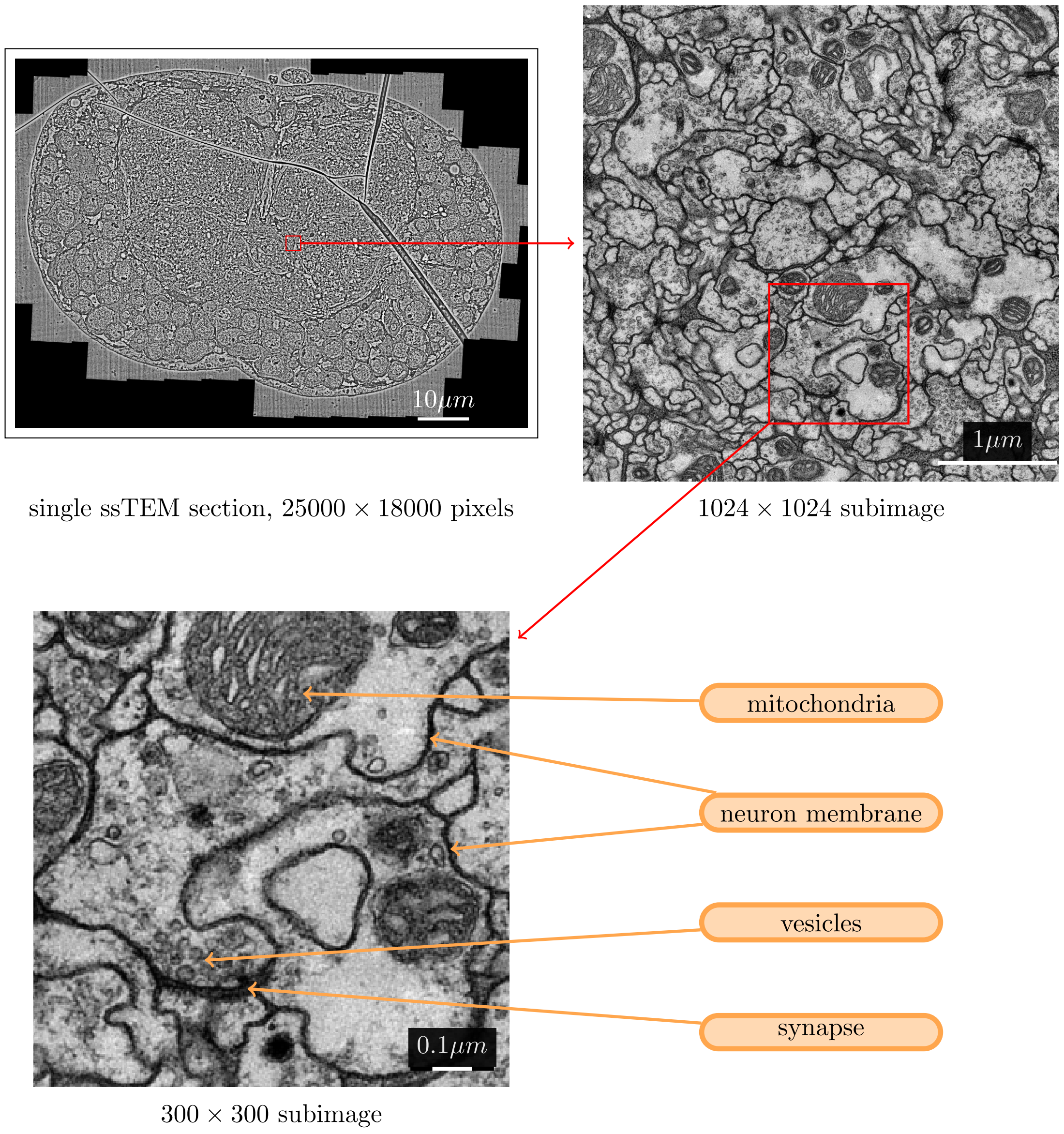


Figure 1.2: Sample of a single section of neural tissue of the 3rd-instar *Drosophila melanogaster* larva, imaged using ssTEM. With this imaging technique, individual neurons can be identified by their dark membranes. Synaptic connections between the neurons are visible, and pre-synaptic partners can be determined via the presence of vesicles.

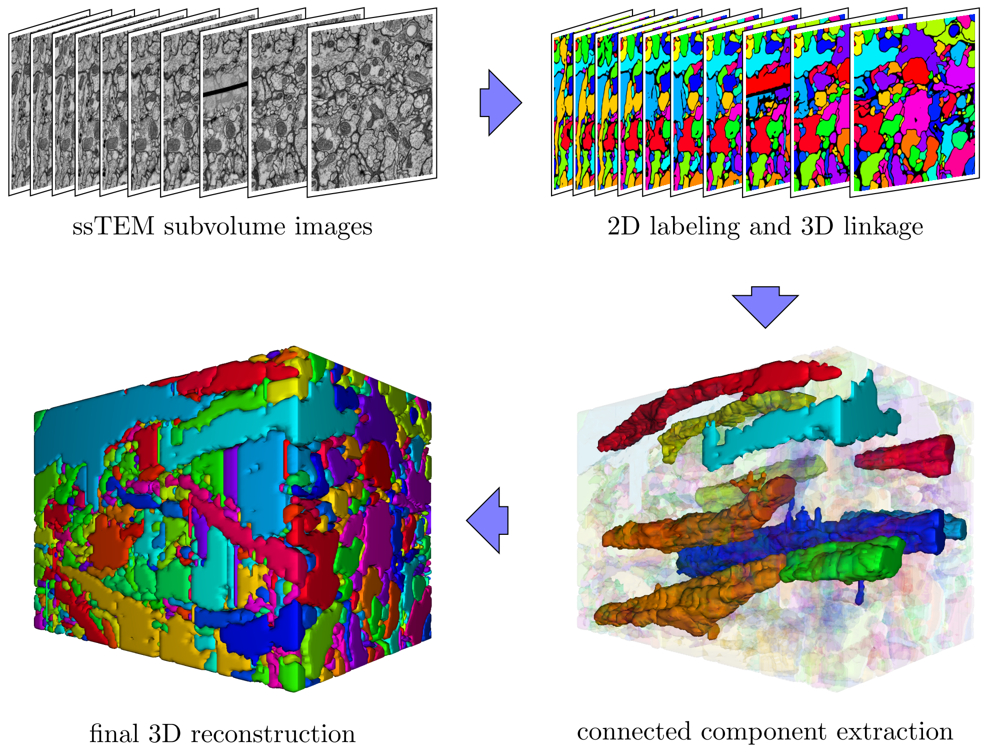


Figure 1.3: High-level view of the neuron reconstruction problem for a stack of images from a subvolume of neural tissue. 2D neuron slices need to be identified and linked across sections. The 2D shape and linkage information yields a 3D reconstruction of the neurons of the subvolume.

career [81]. In this light, the complete manual reconstruction of a mouse, cat, or even a human brain is by far out of reach.

In this thesis, we address the problem of automatically reconstructing neurons from serial section transmission EM (ssTEM). In the following section, we briefly describe this imaging technique with its peculiarities and the needed pre-processing steps to obtain image stacks of neural tissue. These stacks constitute the starting point for both the manual and automatic reconstruction of neurons. We highlight some of the challenges of the reconstruction task in Section 1.2 and provide an overview of the methods that we introduce with this thesis in Section 1.3.

1.1 Data Acquisition and Pre-Processing

In the last decade, substantial progress has been made in the development of EM technologies [7, 15, 21, 35, 54]. Currently, the available EM techniques for blocks of neural tissue can be divided into two groups: the well-established serial section EM techniques and the more recent serial block-face scanning EM techniques. Both produce a stack of 2D images from a 3D block of neural tissue. The main difference between the groups is how these 2D images are obtained. The serial section EM techniques require cutting the block of tissue first into very thin sections, which are then imaged individually. In contrast, serial block-face scanning techniques implement an iterative process that alternates between scanning directly a face of a block and removing a very thin layer from this face. The 3D volumes obtained this way do not show registration artifacts [21, 54]. In the case of FIBSEM (focused ion beam scanning EM, a type of serial block-face scanning) [54], the obtained volumes are even isotropic, *i.e.*, the spatial resolution of voxels is the same in x, y, and z. But despite the clear advantages in image quality of serial block-face scanning techniques, so far only serial section techniques are capable of imaging the very large volumes of tissue needed to contain complete neural circuits with a resolution that is sufficient to resolve synapses and terminal dendrites [7, 15]. For this reason, the methods presented in this thesis focus on images obtained from ssTEM, which we briefly describe in the following section.

1.1.1 ssTEM Imaging

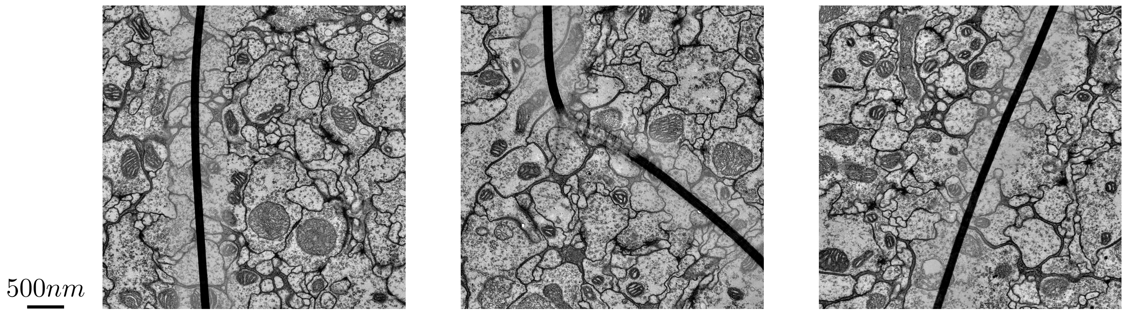
Since its first systematic and practical description more than 30 years ago [94], ssTEM became the most widely used technique to image large volumes of neural tissue [12]. For ssTEM, a sample of neural tissue is first fixed with aldehyde or high-pressure freezing [66] and then embedded in a block of a polymeric material [12] (see Figure 1.1 for an illustration with a *Drosophila melanogaster* larva as a model organism). Serial sections are obtained by cutting the block with a diamond knife into so-called "ultra-thin" sections of about $40nm$ to $50nm$ thickness. Attached to the diamond knife is a container filled with water, such that the scraped off sections directly float onto the surface of the water. From there, one or multiple sections can be picked up using a grid, *i.e.*, a small metal plate with a hole that is covered with a transparent support film, such that the section is carried by the support film above the hole. Staining of the tissue with heavy metals that increase the contrast in the later EM imaging can be done before fixation or after the cutting process. The grid is then placed inside a transmission electron microscope, where several overlapping images with an x-y-resolution of about $4nm$ are recorded to cover the whole section. For that, the amount of electrons that are transmitted through the section at every location is measured, which is less in locations where the heavy metal stain blocks their way. The great amount of detail that is revealed by this method is shown in an example ssTEM image in Figure 1.2.

Although very popular, ssTEM has several drawbacks, compared to the serial block-face scanning approaches, that stem from the cutting and preparation of the sections. The most severe error is the loss of whole sections during the manual cutting process and in particular the delicate placement of the sections on the grid. There are also minor artifacts that make the interpretation of the obtained images difficult: The cutting itself deforms the sections by squishing them in the cutting direction. This has to be accounted for in later processing steps. Another cutting artifact is the occasional ripping of sections, which leaves small holes in the sections. Furthermore, the thickness of the cut sections can vary and thus produces images with different statistics. Occasionally, the support film on the grid contains small folds, which produce dark lines in the EM images. Similarly, the staining precipitates produce cloud-like structures in the images. But also the anisotropy of the sections itself can be seen as an artifact: In some situations, the section thickness exceeds the diameter of small neural processes, making it very hard to follow processes that run parallel to the cutting plane. Examples of some of the mentioned artifacts are shown in Figure 1.4.

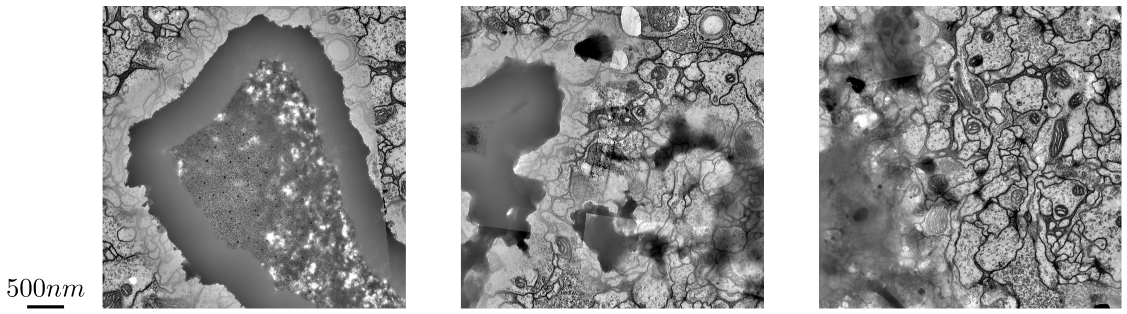
1.1.2 Registration

Two more problems need to be addressed, before we can use the images recorded by the transmission electron microscope for manual or automatic neuron reconstruction. First, the microscope does not produce a single image per grid that was loaded into it. To cover the comparatively large section, several overlapping tiles are imaged. The tile images need to be stitched together to obtain a single image per section. Due to non-linear deformations during the imaging process (similar to deformations with optical lenses), this task is not trivial. Second, the images of different sections need to be registered against each other. Besides the obvious changes in translation and rotation, it is also necessary to account for non-linear deformations of the sections that stem from the cutting process.

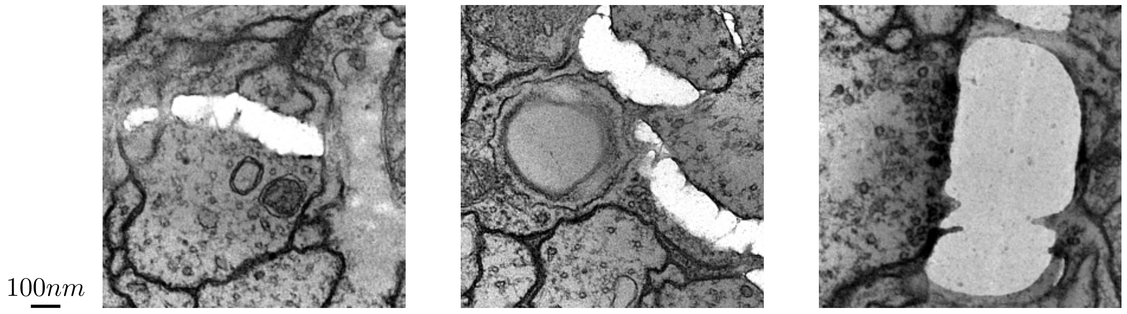
Recently, Saalfeld et al. made impressive progress in solving both problems jointly, even for very large datasets consisting of several thousand sections [87]. In their approach, SIFT



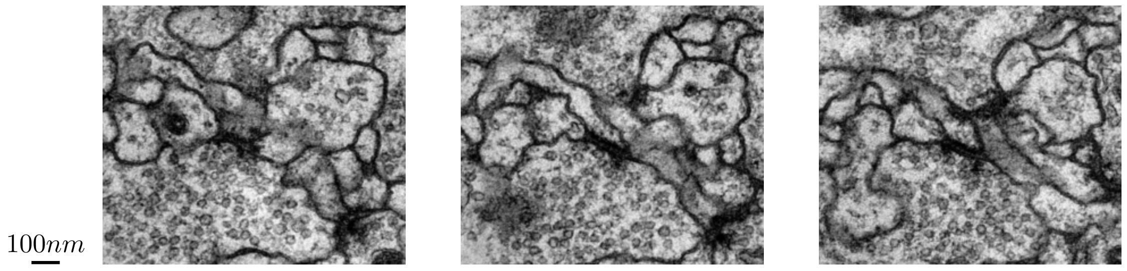
(a) support film folds (3.2%)



(b) staining precipitates (2.6%)

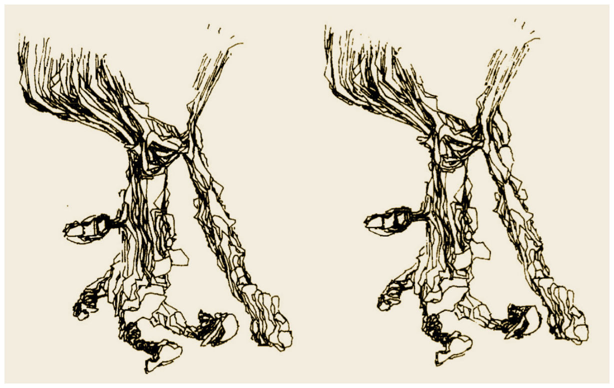


(c) section rips (1.0%)

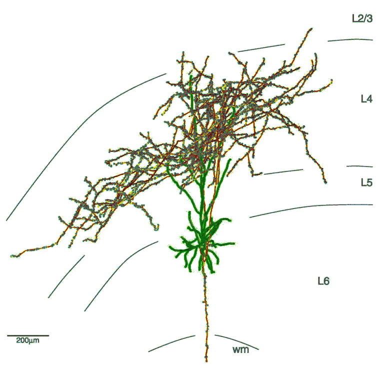


(d) low z-resolution (ambiguous situations in consecutive sections)

Figure 1.4: Examples of ssTEM imaging artifacts. The numbers in parentheses are the percentage of images that expose the corresponding artifact in a randomly chosen subvolume of 500 consecutive images of size 1024×1024 in the *Drosophila melanogaster* dataset. In addition to the artifacts shown above, 4.4% of the images in the subvolume have been missing due to lost sections.



(a) ssTEM (figure from [94])



(b) light microscopy (figure from [6])

Figure 1.5: Two examples of manually reconstructed neurons. (a) Stereo image of the 3D reconstruction of a single neuron in 1980. ssTEM images have been manually analyzed and the profiles of a neuron were digitalized for each image to yield a 3D model of the neuron. (b) A single reconstructed neuron from cat primary visual cortex. Dendrites are shown in green, axons are red, boutons are shown as dots. The reconstruction of this neuron from light microscopy, where synapses are not even visible, took several weeks.

feature [71] correspondences between the tile images of the same and adjacent sections are identified to find a coarse registration. This initial registration is then refined by modelling each tile image as a mesh of springs. For the nodes of these meshes, additional springs are introduced to visually similar points in adjacent tile images. These points are found using normalized cross-correlation in a neighborhood determined by the initial coarse registration. The final spring mesh setup is relaxed in a simulation, where springs inside a tile image favour rigidness and springs across tile images try to pull similar image contents towards each other. After convergence, the shifted spring mesh nodes give the non-linear registration of the tile images into a single 3D volume. With this approach, the quality of the registration is – with the exception of a few outliers – more than good enough for manual reconstruction, and leaves only little correction work to do for automatic reconstruction methods.

1.2 Neuron Reconstruction

Neuron reconstruction refers to the process of retrieving the shape and possibly also the connectivity of neurons from microscopic images of neural tissue. Depending on the application, the required level of detail of the shape can vary. For plain circuit reconstruction, tracing the skeletons of the neurons is sufficient, speeding up the manual reconstruction process. In contrast to that, a volumetric reconstruction delivers the exact 3D shape of each neuron but is much more tedious.

The investigation of neural circuits with EM technologies has a surprisingly long history, that almost dates back to the commercial availability of electron microscopy. One of the first works of this kind was the study of the squid giant axon by Richards et al. in 1943 [83]. Almost four decades later, EM from serial sections became popular and the protocol was refined to an extent that allowed neuroscientists to collect several hundred consecutive sections, and this marked the beginning of manual neuron reconstruction from EM. In Figure 1.5a we see the result of an early work by Stevens et al. [94], where the 3D shape of a neuron was reconstructed from a "movie" of EM images by digitizing manually painted profiles.

Today, we are less limited by the available imaging techniques and can record EM volumes of a whole neural system of small model organisms like the *Drosophila melanogaster* larva[4] or *C. elegans* [108]. The amount of manual interaction needed to obtain the reconstruction has also decreased significantly with the development of better tools and faster hardware. However, in the long run, systems for automatic neuron reconstruction will be needed.

1.2.1 Current State of Automatic Methods

The attempts to automate the processing of EM images of neural tissue are as diverse as the biological questions that naturally arise if data of this detail and amount is available. In the same way that the imaging techniques can be divided into block-face scanning and serial section methods, the automatic approaches can be divided into isotropic and anisotropic methods.

Methods for isotropic data usually exploit the fact that the stack of EM images can be interpreted as a continuous 3D volume. Consequently, the developed algorithms are isotropic as well, *i.e.*, they do not distinguish between the dimensions of the data. The main focus of these methods lies in the detection of membranes to separate individual neurons [1, 2, 38, 40–42, 64, 99]. Although early work formulated this task as a plain membrane pixel detection problem [40–42], recent approaches formulate the reconstruction problem as finding optimal cuts in an affinity graph between voxels or supervoxels [1, 2, 64, 99]. Finding these cuts is a challenging problem on its own, which was recently shown to have an elegant solution by Andres et al., who formulate the optimization as an integer linear program [2]. The optimization problem has exponentially many constraints, which can be added in a cutting-plane like fashion until global optimality is achieved. To obtain a connectivity matrix of the neurons in a block of neural tissue, synapses have to be identified as well. For isotropic data, Kreshuk et al. pioneered this field with an approach that is based on pixel-wise predictions obtained from a random forest classifier [11] on local features [62, 63], which was further improved by Becker et al. in a context-aware approach [5]. Finally, also the reconstruction of cell organelles like mitochondria gained attention in isotropic volumes [72, 73]. Similar to the reconstruction of neurons, mitochondria are found by segmenting supervoxels using a conditional random field with unary terms learned in the structured learning framework [98].

Methods for anisotropic EM data, as obtained by serial section EM, usually treat the EM image stack as a collection of 2D images. The reconstruction of neurons is mostly achieved by processing the 2D images individually to obtain an initial oversegmentation, which is then merged within and between the EM images in the stack [50, 69, 77, 89, 102, 103, 105, 110]. Due to the anisotropy of the data, the identification of synapses is much more challenging. Nevertheless, recent progress has been made with the development of more descriptive local image features [39, 61]. We will discuss the current approaches for neuron reconstruction in anisotropic EM data in more detail in the related work sections of subsequent chapters.

1.3 Thesis Overview

The linchpin of this thesis is a novel method for the reconstruction of neurons from serial section EM images. We present this model in Chapter 2 and several extensions in subsequent chapters. The following list gives a brief overview of each chapter's content.

Chapter 2: We introduce an assignment-based model for neuron reconstruction. In this model, binary indicator variables model the assignments between candidate slices of neurons in individual sections. We show how the costs for each assignment can be learnt with a random forest classifier on positive assignment examples only.

[4] The whole central nervous system of a 1st-instar *Drosophila melanogaster* larva has been sectioned and imaged by Richard Fetter and the FlyEM project at Janelia and registered by Stephan Saalfeld and Albert Cardona.

Chapter 3: The candidate neuron slice selection is a crucial step in our model. We introduce an alternative candidate generation method that samples candidates from a conditional random field based on convolutional neural network predictions. This improves results compared to our previous and other methods.

Chapter 4: The training of the costs for assignments with random forest classifiers has several drawbacks. We show how the framework of structured learning for feature weights in a linear cost function can be applied and provides better results.

Chapter 5: For the application to very large datasets, inference has to be distributed. We show how this can be done with our model while still guaranteeing global optimality.

Chapter 6: We review error measures used for neuron reconstruction algorithms and discuss their properties. We introduce a new measure that reflects the edit distance between a reconstruction and a ground truth within certain tolerated variations and compare it to existing measures.

Chapter 2

An Assignment-Based Model for Neuron Reconstruction

2.1 Introduction

In this chapter[1] we present our approach to addressing the problem of segmenting neural tissue in anisotropic volumes with high x- and y-resolution but low z-resolution, as obtained by serial section electron microscopy (EM) imaging procedures [37]. The segmentation is supposed to separate neuron cell interior from membrane and extracellular space, and thus reveal the 3D shape of the neurons. In the following, we refer to this segmentation as neuron reconstruction.

The development of our approach was motivated by the reconstruction of neurons in the ventral nerve chord of the *Drosophila melanogaster* larva [15]. In this nerve chord, neural processes are mostly running perpendicular to the section cutting planes. In such a setup, the reconstruction of a neuron can be seen as a tracking of 2D neuron slices over the z-dimension, where it is possible that neurons continue, end, begin, split, and merge. We address this tracking problem by considering many diverse and contradictory local tracking hypotheses. These tracking hypotheses are possible assignments of 2D neuron slices across sections. We find a globally consistent and cost-minimal subset of assignments for the whole volume. This is achieved by formulating the inference problem as a linear constraint binary model, which can be solved with standard integer linear program solvers.

2.1.1 Overview

An overview of our model is shown in Figure 2.1. Due to the anisotropy of the EM volume, often also accompanied by registration errors, we prefer to treat the image data as a stack of 2D images instead of a continuous 3D volume. We generate different segmentation hypotheses for each image of the stack individually, *i.e.*, without consideration of spatial context in the z-direction. This is achieved by a sequence of parameterized graph-cut segmentations on membrane predictions, that we explain in detail in Section 2.2. We consider the connected components of each of those segmentations as 2D neuron candidates, *i.e.*, hypotheses for the cross-section of a neural process in a 2D image. Between each pair of consecutive images, all possible assignments of these 2D neuron candidates are enumerated and represented by binary assignment variables. In Section 2.3 we show how each possible continuation, branch, and end of a neural process is explicitly represented by an assignment variable of its own. To keep the number of variables linear in the volume size, we discard assignments between candidates with an x-y-displacement that is above a certain

[1]This chapter is based on our paper [26], which was published in 2012. The results in this chapter represent the current state at the time of publication.

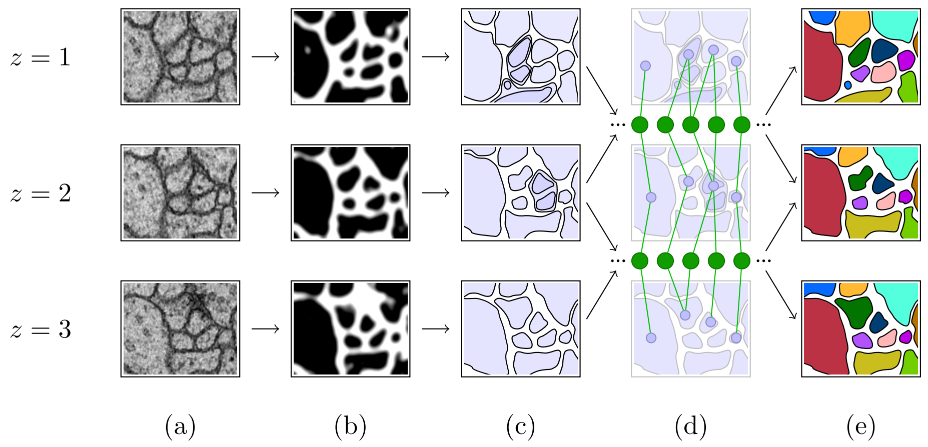


Figure 2.1: **Overview of our approach.** For all images in the stack (a), membrane predictions are obtained from low-level image features. With a sequence of graph-cuts, different 2D neuron candidates are extracted from those predictions (c). All possible assignments of these candidates across adjacent images are enumerated (d) and, according to assignment specific costs (obtained by a trainable classifier), a global optimum of both 2D neuron segmentations and assignments across sections is found (e).

threshold, assuming a coarse registration of the volume [86]. In Section 2.5 we show how we can train an assignment cost function on positive assignment samples. We use this cost function together with the graph-cut energy of the involved 2D neuron candidates to obtain the costs for selecting an assignment variable. The final segmentations of the images and assignments between them are then jointly found as the cost-minimal solution of a linear constraint binary model, as we show in Section 2.4. The results in Section 2.6 show that our approach improves the state of the art in anisotropic neuron reconstruction, while still being fast enough for practical applications and interactive correction. We also show how a confidence measure derived from our model has the potential to be used for directing an expert's attention during proof-reading and semi-automated segmentation correction. An implementation of our method is publicly available [25].

2.1.2 Related Work

Several authors have contributed to the problem of neural reconstruction from electron microscopy (EM) data for isotropic volumes [2, 40, 41, 43, 72] as well as for anisotropic volumes [46, 51, 52, 102–105, 110]. Although the majority of these publications focus on the detection of neuron boundaries for direct reconstruction [2, 40–42, 46, 51, 52, 102, 103, 105, 110], there have also been successful efforts in the detection of cell organelles like mitochondria [72], the increase of depth resolution in anisotropic volumes [104], and the invention of meaningful low-level feature detectors [103], all aimed to facilitate the detection of neurons.

Most approaches for the processing of anisotropic volumes are based on an oversegmentation that is carried out for each image individually and then merged within and between the images [102, 103, 105, 110]. On the other hand, it has also been shown that the direct segmentation of neuron membranes is a possible approach: either by using a series of neural networks [46] or by extension of the graph-cut formulation with a "good-continuation" term that is also incorporating context from adjacent images [52]. It has been noted that the joint segmentation of several images is a promising technique to increase the overall accuracy [84]. Merging of segments found between (and possibly within) images is a special case of this technique and a common problem for neuron reconstruction that has mainly been solved approximately, for instance, by problem relaxations [105] or greedy assignment strategies [51, 110]. A noteworthy exception to these solutions, which is also the closest one to our approach, solves the merging problem optimally [103]. As in that work, we enumerate possibly contradictory 2D neuron candidates in individual images, similar to the region selection approach in [65]. The candidates in [103] are fused in pairs between two images, while our work also allows split and merge assignments. A simple linear function of a few weighted features of the involved image patches is used in [103] to compute the cost of accepting a pair, while we use a random forest classifier on many features. The resulting optimization problem, of which assignments to accept, is solved efficiently by an integer linear program (ILP) that ensures consistency of the resulting segmentation. Our work goes beyond [103] by handling branching of neurons and providing confidence feedback. The approach in [103] is also restricted to a small number of features for computing the costs of an assignment: since there is no obvious way to learn the weight of each feature, the difficulty of tuning the weights limits the number of features. In contrast, we use random forest classifiers, which are well known to be able to handle a large number of features.

2.2 2D Neuron Candidate Extraction

In this section we describe our method to extract 2D neuron candidates from individual images of the stack using a sequence of graph-cuts. We also show how the number of considered candidates can be reduced without altering the topological properties of the possible results.

Practically, in order to identify neurons in images of a stack, we face a binary segmentation problem: it suffices to detect *membrane* pixels versus inner *neuron* pixels. Every connected component labeled as *neuron* would be the cross-section of one neuron. To the best of our knowledge, there exists no reliable method to discriminate between *neuron* and *membrane* pixels in a single image. In situations with high ambiguity, even human experts cannot unequivocally draw boundaries without inspection of adjacent images.

However, direct incorporation of spatial context in the segmentation task is impractical. The presence of registration errors – even in the scale of the width of a membrane – makes it very hard to model the influence of one pixel to its partners in adjacent images [52]. Therefore, we propose to extract a set of *possible* connected components that might represent neuron cross-sections. These connected components, that we will call 2D neuron candidates or just candidates in the following, are allowed to overlap and thereby contradict each other. Thus, we increase the scale of ambiguity from pixels to larger regions. It remains to find consistent subsets of candidates and assign them across adjacent images in the stack. In Section 2.3 we show how this can be done with consideration of all images at once.

2.2.1 Graph-Cut Sequence

The 2D neuron candidates are extracted for each image of the stack independently by performing a series of graph-cuts to optimize a parameterized energy term [9]. Given a field of per-pixel local features $\mathbf{x}$, a binary segmentation $\mathbf{y}$ is obtained by minimizing

$$E(\mathbf{y}, \mathbf{x}) = \sum_{i \in \Omega} (D(x_i, y_i) + \lambda_N y_i) + \sum_{(i,j) \in \mathcal{N}} \lambda_S (1 - \delta_{y_i = y_j}). \tag{2.1}$$

Here, $\Omega \subset \mathbb{N}^2$ is the image domain and $D(x_i, y_i) = -log\, p(y_i | x_i)$ is the log-likelihood of pixel $i \in \Omega$ belonging to *membrane* or *neuron*, as given by a pre-trained random forest classifier [11]. The set $\mathcal{N} \subset \Omega \times \Omega$ contains all pairs of 8-connected neighboring pixels. The second term in the exponent ensures smoothness of the segmentation, where δ is the Kronecker delta.

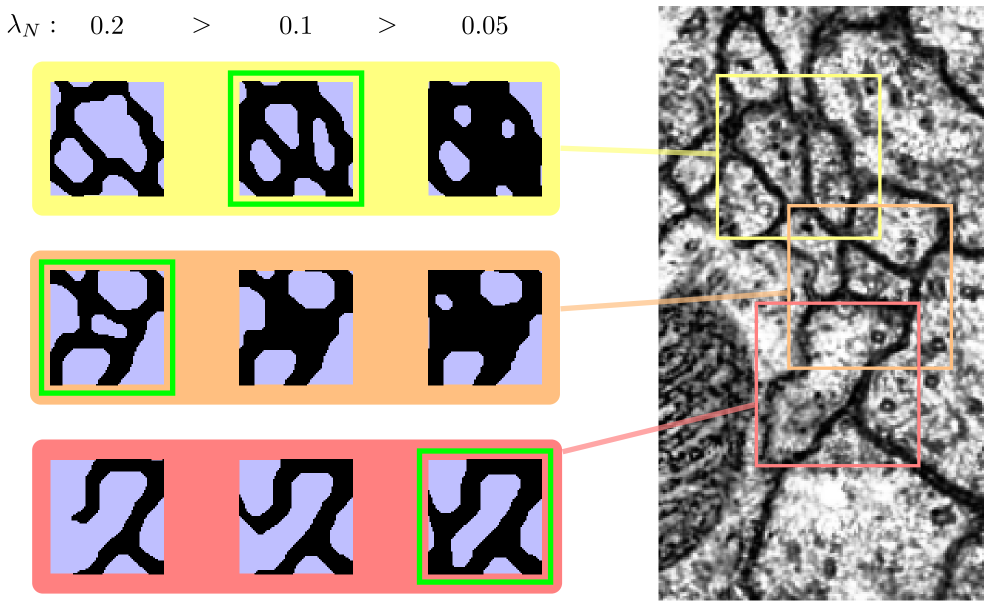


Figure 2.2: Segmentation results of nearby image regions for different values of λ_N. This parameter determines how many of the pixels are being labeled as *neuron* (white). The desired segmentation result for each case is highlighted in green, showing that there is no single value λ_N that segments the shown image correctly.

The parameter λ_N is a prior on the expected number of pixels assigned to *neuron* and λ_S controls the influence of the smoothness term. Finding an optimal set of parameters is a non-trivial task and one cannot expect a fixed set of parameters to perform well on all images [57]. We claim that a fixed set of parameters cannot even be expected to perform well on all areas of one image, see Figure 2.2 for an example. Therefore, we enumerate several local segmentation hypotheses by variation of λ_N. This can be done efficiently by several warm-started graph-cuts [55], from which we obtain a series of segmentations. Each segmentation consists of a set of connected components $\boldsymbol{C}_i \subset \Omega$ that are labelled as *neuron*. Each of these components is considered as one 2D neuron candidate. As λ_N decreases, these components can grow and merge, thus establishing a tree-shaped subset hierarchy, the so-called *component tree* [45], shown in Figure 2.3. Let $\mathcal{C}_z$ denote the set of all candidates of the image with index z in the stack. Any consistent subset $\mathcal{S}_z \subseteq \mathcal{C}_z$ of these candidates yields a valid segmentation of this slice. A subset is consistent if none of the containing components overlap, *i.e.*, $\boldsymbol{C}_i \cap \boldsymbol{C}_j = \varnothing$ for all $\boldsymbol{C}_i, \boldsymbol{C}_j \in \mathcal{S}_z$ with $i \neq j$.

2.2.2 Downsampling of Component Trees

To reduce the number of 2D neuron candidates, we propose to discard candidates that are already well represented by others and do not introduce a new interpretation of the image. In particular, we are not interested in *only children* of the component trees, *i.e.*, components that are the only child of their parent. For an example see Figure 2.3. These candidates are the only subset of their parent and therefore carry the same information (there is a 2D neuron slice) on a smaller set of pixels. In other words, if there are different conflicting candidates with the same topological properties, we choose to consider the biggest one only.

The effect of this downsampling is that the average distance between neighboring candidates is reduced – a fact that makes sense, considering that membranes are fairly thin compared to the diameter of neurons.

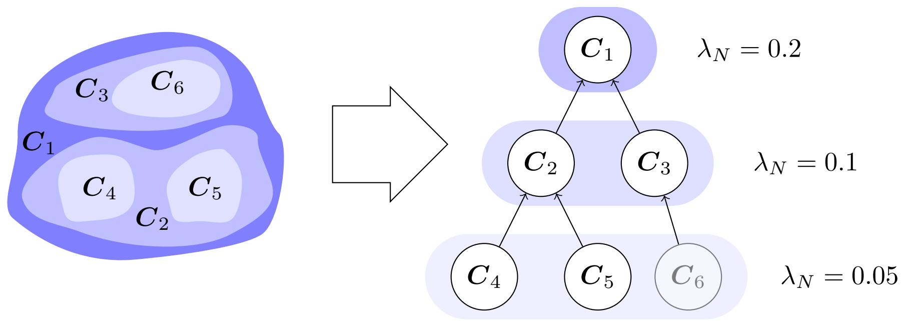


Figure 2.3: Visualisation of the 2D neuron candidate extraction. For different values of the prior parameter λ_N (different shades of blue), connected components of the segmentation are found (left side). The subset relation of these connected components define the component tree (right side). The candidate $\boldsymbol{C}_6$ gets removed since it is the only child of its parent.

2.3 Assignment Model

In this section we describe our model for possible assignments of the 2D neuron candidates across images. We introduce binary assignment variables for each possible assignment and costs for selecting them. We show how we ensure the consistency of any solution with linear constraints and how to find the optimal solution by solving an integer linear program (ILP). We also present our training method for the assignment costs, as well as a confidence measure that can be used to evaluate the solution.

2.3.1 Assignment Variables

For each possible assignment of a 2D neuron candidate in one image to a candidate in the previous or next image, we introduce one binary assignment variable. This variable is set to 1 if the involved candidates and their mutual assignment are accepted.

Let m be the number of all possible assignments. A binary vector $\mathbf{a} \in \{0,1\}^m$ of assignment variables is created similarly to the method proposed in [78]. Each possible continuation of a candidate $\boldsymbol{C}_i$ in image z to $\boldsymbol{C}_j$ in image $z+1$ is represented by a variable $a_{i\to j}$. A split of $\boldsymbol{C}_i$ in image z to $\boldsymbol{C}_j$ and $\boldsymbol{C}_k$ in image $z+1$ is represented as $a_{i\to j,k}$. Similarly, each possible merge is encoded as $a_{i,j\to k}$. Appearances and disappearances of neurons are encoded as assignments to a special end node $\boldsymbol{E}$, i.e., for each candidate $\boldsymbol{C}_i$ we introduce two variables $a_{i\to\boldsymbol{E}}$ and $a_{\boldsymbol{E}\to i}$. For the possible assignments, only candidates within a threshold distance θ_D to each other are considered. Thus, the number of assignment variables is linear in the number of extracted candidates. See Figure 2.4 for examples of assignments of a single candidate.

For each assignment variable we define costs representing the compatibility of the involved candidates. For that, a vector $\mathbf{c} \in \mathbb{R}^m$ is constructed. The costs for one-to-one assignments are modelled as:

$$c_{i\to j} = A_C(\boldsymbol{C}_i, \boldsymbol{C}_j) + \theta_S S(\boldsymbol{C}_{ij}), \tag{2.2}$$

where we write $\boldsymbol{C}_{ij}$ as a shorthand for $\boldsymbol{C}_i \cup \boldsymbol{C}_j$. The term $A_C(\boldsymbol{C}_i, \boldsymbol{C}_j)$ is the negative log-probability of a continuation, as obtained by a random forest classifier (see Section 2.5 for details). The term $S(\boldsymbol{C})$ gives the cost of assigning all pixels of $\boldsymbol{C}$ to *neuron* as given by Eq. (2.1), *i.e.*,

$$S(\boldsymbol{C}) = \sum_{j\in\boldsymbol{C}} \left(D(x_j,0) - D(x_j,1)\right) + \sum_{\substack{(j,k)\in\mathcal{N};\\ j\in\boldsymbol{C},\ k\notin\boldsymbol{C}}} \lambda_S. \tag{2.3}$$

In a similar way, we define the costs for splits:

$$c_{i\to j,k} = A_B(\boldsymbol{C}_i, \boldsymbol{C}_j, \boldsymbol{C}_k) + \theta_S S(\boldsymbol{C}_{ijk}), \tag{2.4}$$

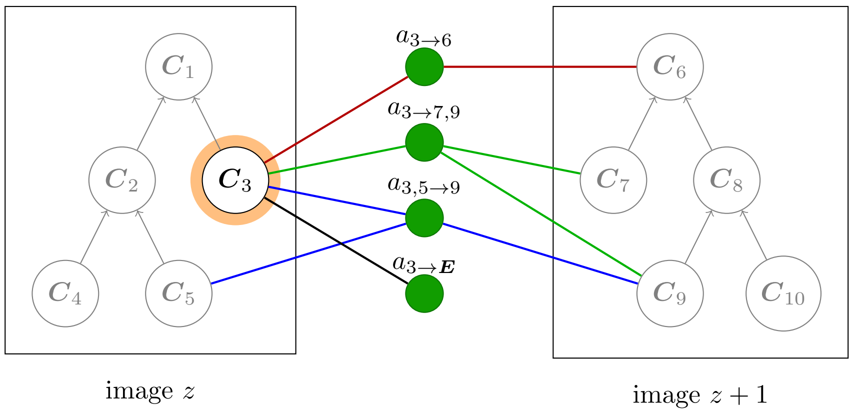


Figure 2.4: Examples of the four outgoing assignment types for a single candidate ($\boldsymbol{C}_3$, orange): A continuation (red lines) is modelled for each candidate in the next image that is within a threshold distance. Possible splits (green lines) and merges (blue lines) are enumerated for neighboring candidates in the respective images. The possible sources or targets of splits and merges are again candidates within a threshold distance. The disappearance of a neuron is represented by a single assignment (black line).

where $A_B(\boldsymbol{C}_i, \boldsymbol{C}_j, \boldsymbol{C}_k)$ is the negative log-probability of a branching. The merge cases are defined analogously and the appearance or disappearance of neurons cause the following costs:

$$c_{i\to \boldsymbol{E}} = c_{\boldsymbol{E}\to i} = A_E(\boldsymbol{C}_i) + \theta_S S(\boldsymbol{C}_i). \tag{2.5}$$

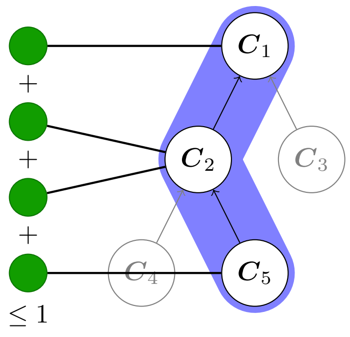
The segmentation likelihood weight θ_S is a free parameter of the model.

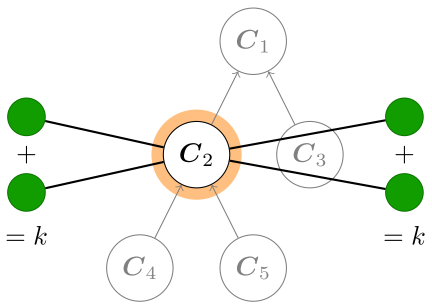
With costs defined like this, the optimal solution would favour candidates that are high in the component tree, since this would minimize the number of assignments that need to be made and there is a size-independent cost contribution from the random forest classifier for each assignment (A_C, A_B, and A_E). Therefore, we scale the random forest contribution for each possible assignment by the sum of leaves that are under the involved candidates. The intuitive meaning of this strategy is the following: An assignment involving a big candidate should only be preferred if its cost is less than the costs of any set of assignment involving subset candidates.

2.3.2 Consistency Constraints

A solution to both the segmentation of images and the assignments between the segments can be obtained by finding the subset of assignment variables with minimal costs: Every 2D neuron candidate that is involved in a selected assignment is taken to be a valid segmentation of a neuron.

However, overlapping candidates are contradictory and thus impose constraints on the assignment variables that can be picked. In particular, we have to ensure that for every path $\boldsymbol{P}$ in every component tree (*i.e.*, for every set of overlapping candidates) the number of assignments that connect them in any direction (to the previous or next image) is at most one. We call this the *candidate consistency* constraint. Furthermore, we have to ensure that a candidate that was picked from an assignment to a previous image will also be picked by an assignment to the next image. Even if a neural process ends, we want to force the solution to pick the end segment. We call this the *explanation consistency* constraint, since it ensures a continuous sequence of assignments, one continuing where the previous one ended. See Figure 2.5 for a visualisation of these constraints. Both types of constraints can





(a) candidate consistency (b) explanation consistency

Figure 2.5: Visualization of the two types of consistency constraints. The candidate consistency (a) ensures that no pixel is assigned to more than one candidate: For each path of the component tree (blue), the sum of all *incoming* assignment variables (green) has to be at most one. The explanation consistency (b) ensures a continuous sequence of assignments: For each candidate (orange), the sum of all *incoming* assignment variables (from the previous image) has to be equal to the sum of all *outgoing* assignment variables (to the next image). The incoming or outgoing assignment variables for a candidate are all assignment variables that have the component as target or source, respectively.

be expressed by the following linear (in)equalities:

$$\sum_{i \in \boldsymbol{P}} \sum_{a \in \mathbf{a}_{\rightarrow i}} a \leq 1 \qquad \forall \boldsymbol{P} \in \mathcal{P} \tag{2.6}$$

$$\sum_{a \in \mathbf{a}_{\rightarrow i}} a - \sum_{a' \in \mathbf{a}_{i \rightarrow}} a' = 0 \qquad 1 \leq i \leq n, \tag{2.7}$$

where $\mathcal{P}$ is the set of all paths in every component tree and n the number of all candidates. The sets $\mathbf{a}_{\rightarrow i}$ and $\mathbf{a}_{i \rightarrow}$ denote all assignment variables that involve the candidate $\boldsymbol{C}_i$ to the previous or next slice, respectively.

2.3.3 Confidence Measure

To provide a confidence measure, we exploit the candidate consistency constraints on the assignment variables: For each assignment variable a_i that is part of the solution, we determine the minimal cost $\bar{c}_i$ of any of its directly conflicting assignment variables:

$$\bar{c}_i = \min(\mathbf{c}_{\lightning i}), \tag{2.8}$$

where $\mathbf{c}_{\lightning i}$ denotes the set of costs of all assignment variables that are in direct conflict with a_i. According to the candidate consistency constraint, these are all assignments that link to any candidate that shares a path with the candidates that are involved in a_i. We define the confidence of an assignment as the ratio of the costs of the assignment to the minimal cost of any conflicting assignment

$$\text{conf}(a_i) = \frac{c_i}{\bar{c}_i}. \tag{2.9}$$

2.4 Inference

Given the costs $\mathbf{c}$ as defined in the previous section, we find a cost-minimal and consistent set of candidates and assignments between them by solving the following integer linear program:

$$\begin{aligned}
\min_{\mathbf{a}} \quad & \langle \mathbf{c}, \mathbf{a} \rangle \\
\text{s.t.} \quad & \sum_{i \in \boldsymbol{P}} \sum_{a \in \mathbf{a}_{\to i}} a \leq 1 && \forall \boldsymbol{P} \in \mathcal{P} \\
& \sum_{a \in \mathbf{a}_{\to i}} a - \sum_{a' \in \mathbf{a}_{i \to}} a' = 0 && \forall i = 1, \dots, n \\
& a_i \in \{0, 1\} && \forall i = 1, \dots, n
\end{aligned} \tag{2.10}$$

Although in general NP-hard, problems of this form can be solved efficiently in practice if the costs are strong (*i.e.*, they favour only a few solutions) and the constraints are local. In Section 2.6 we show empirically that this is the case for our problem.

2.5 Learning

The training of our assignment model consists of learning a random forest classifier on possible assignments between 2D neuron candidates. The idea to use a classifier to help merging regions is not entirely new [51, 105]. The biggest difference here is that we use the same classifier for continuation, branching, and end assignments. For that, we create a feature vector for each assignment variable. This vector has the same components regardless of which assignment case it refers to. For features that do not exist in all cases (for example, there is no center distance measurable in end assignments) the respective components are set to a dummy value.

To train the classifier, we need both positive and negative samples of assignments. However, only positive samples need to be provided by a user. Due to the candidate consistency constraints on the assignment variables, every positive assignment has a number of conflicting assignments that we take as negative samples.

2.5.1 Assignment Features

The features we are using to train the random forest classifier belong to two groups: geometry features and texture features. For the geometry features, we measure the distance between the centers of the candidates, the symmetric set difference of their pixels and the plain size of them. For the texture features, we perform a cross-correlation between image patches surrounding the candidates in question and use the position and value of the maximum in two different scales. In addition, we use the difference of normalized intensity histograms between the candidates that are supposed to be assigned to each other. A complete list of features can be found in Table 2.1.

Geometry Features

For each pair of 2D neuron candidates that is involved in a continuation assignment, the distance between the centroids projected to the x-y-plane is taken as a feature (CD). To account for the similarity in size and shape, we also measure the symmetric set difference (SE) of the pixels after translating the candidates such that the centroids line up. Here, the z-coordinate of the pixels is ignored. To not punish large candidates more than small ones, we normalize the set difference by the sum of the pixel sizes of the two candidates and take this value as an additional feature (SR).

The geometry features for the branching assignments are extracted in a very similar way to the continuation cases: the two candidates that are in one image are treated as one candidate to extract the aforementioned features (CD,SE,SR).

The only geometry feature for an end assignment is the size of the respective candidates.

Category		Feature	Description	Assignment Case
Geometry		CD	center distance	C,B
		SE	set difference	C,B
		SR	set difference ratio	C,B
		SI	size	E
Texture	coarse correlation	CX	offset x	C,B
		CY	offset y	C,B
		CV	max value	C,B
	fine correlation	FX	offset x	C,B
		FY	offset y	C,B
		FV	max value	C,B
	histogram	H0	bin 0	C,B,E
		H1	bin 1	C,B,E
		H2	bin 2	C,B,E
		H3	bin 3	C,B,E
		H4	bin 4	C,B,E
		H5	bin 5	C,B,E
		H6	bin 6	C,B,E
		H7	bin 7	C,B,E
		H8	bin 8	C,B,E
		H9	bin 9	C,B,E
	histogram normalized	N0	bin 0	C,B,E
		N1	bin 1	C,B,E
		N2	bin 2	C,B,E
		N3	bin 3	C,B,E
		N4	bin 4	C,B,E
		N5	bin 5	C,B,E
		N6	bin 6	C,B,E
		N7	bin 7	C,B,E
		N8	bin 8	C,B,E
		N9	bin 9	C,B,E

Table 2.1: List of all features and assignment cases they are used in: C: continuation, B: branching, E: end.

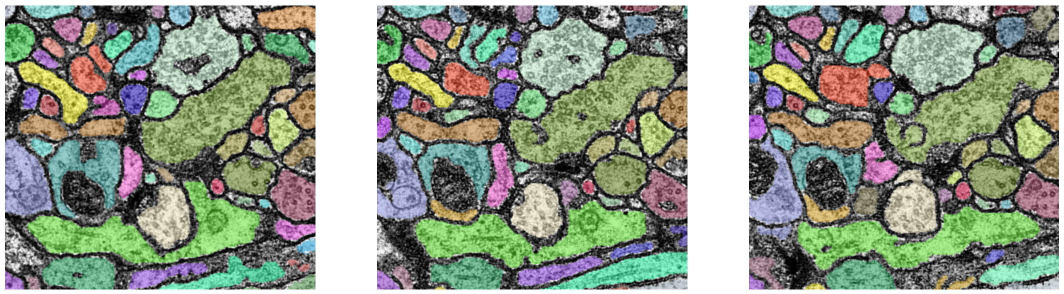


Figure 2.6: Segmentation result of three subsequent images. Assignments between images are represented by the use of the same color.

Texture Features

For each possible continuation assignment between two candidates, we carry out two normalized cross-correlations with different pre-scaling of the images. A coarse cross-correlation is performed after Gaussian smoothing with $\sigma = 10$ pixels in a window of three times the size of the involved candidates. This correlation is meant to account for the context of the candidate. A finer cross-correlation in a window of 1.5 times the size of the involved candidates measures the similarity of the interior of the candidates. In either cases, we perform a cross-correlation in both directions (local window from one candidate against the neighborhood of the other one, and vice versa). The two maxima positions relative to the respective centroid are added and constitute the offset feature (CX, CY and FX, FY). By adding the relative maximum positions, we deliberately ignore any displacement between the candidates, since this is taken care of already with the center distance feature (CD). Consequently, two candidates that agree about their displacement to each other will have an offset of zero. The maximum of both normalized cross-correlation values is taken as another feature (CV and FV).

To further evaluate the interior similarity of candidates we also compute histograms with ten bins of the pixel intensities for each candidate. The bin-wise differences between two candidates in a continuation assignment are added to the feature vector (H0 to H9). The same is done for normalized histograms that sum up to one (N0 to N9).

For branching assignments we also compute correlation and histogram features as for the continuation assignments. For the correlation features, we regard the branching as two continuations with the same source. The correlation features CX, CY, CV and FX, FY, FV are now taken as the average values of the respective continuation features. For the histogram features we simply add the histograms of the two candidates that are in one image and proceed as described for the continuation features.

The only texture features for end assignments are the plain histogram values of the involved candidate.

2.6 Results

We evaluated the performance of our approach on an annotated sample of *Drosophila melanogaster* larva neural tissue [15]. This publicly available dataset consists of 30 serial sections, imaged with transmission electron microscopy at a resolution of 4x4x50 nm/pixel. The dataset covers a 2x2x1.5 micron cube of neural tissue and provides labels of cellular membranes, cytoplasms and mitochondria of 170 neural processes. Figure 2.6 shows qualitative results of our approach on three subsequent images of the volume. In Figure 2.7 we show examples of successfully found branchings.

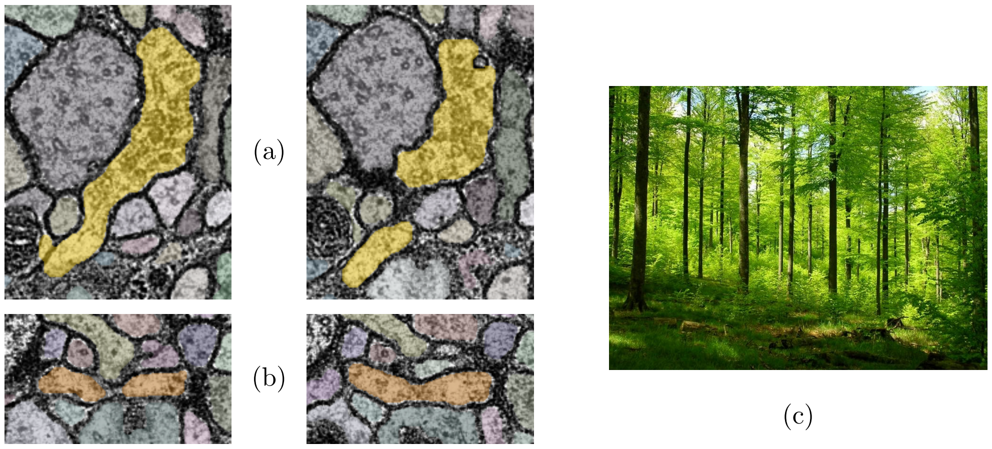


Figure 2.7: Examples of successfully found neuron branchings. (a): the central neuron (left) bisects into two parts (right) and is correctly segmented (yellow color). (b): two smaller processes (left) merge into one neuron (right). Again, the segmentation is correct (brown color). (c): A random forest.

2.6.1 2D Neuron Candidate Generation

We trained a random forest classifier to predict membranes in 2D images (not to be confused with the random forest classifier used for the assignment costs) using the tool Ilastik [90]. From these predictions, we extracted 2D neuron candidates that we used for all experiments. For that, we took 100 equidistant samples of the neuron prior λ_N in an interval that ranged from obvious over- to undersegmentation. The limits of this interval and the weight of the Potts term in the segmentation energy Eq. (2.1) have been found by visual inspection on the first slice of the training dataset using an interactive graph-cut implementation.

After the downsampling of the component trees as described in Section 2.2 we were left with 5800 candidates, distributed in 3633 trees with a mean depth of 0.252 ($\pm$0.491).

2.6.2 Training Data

We tested our approach by splitting the dataset into two parts: the first 5 images have been used for the training, while the remaining 25 images have been used for evaluation. From the ground truth of the first 5 images we extracted 510 positive assignment samples, for which our assignment model found 10340 negative assignment samples that have been made impossible (details on the selection of negative training samples can be found in Section 2.5).

2.6.3 Error Measure

In order to evaluate the performance of our approach, we use an error measure that reflects the number of steps a human expert would have to take at least to restore the ground truth from the result [43]. For that, we distinguish between errors that have been made between images (additional or missing assignments) and within images (additional or missing segments). We will call these errors *inter-* and *intra-slice errors*, respectively.

For the inter-slice errors, we say that two segments are *linked*, whenever there was an assignment selected that involves both of them. Hence, a selected continuation assignments introduces one link between two segments, a branching assignment introduces two links between three segments, and an end assignment has no effect. We count every link that is in the ground truth but not in the result as a *false split* (FS) error and every link in the result that is not in the ground truth as a *false merge* (FM) error. The intra-slice errors are being computed similarly: Every 2D neuron slice of the ground truth that has

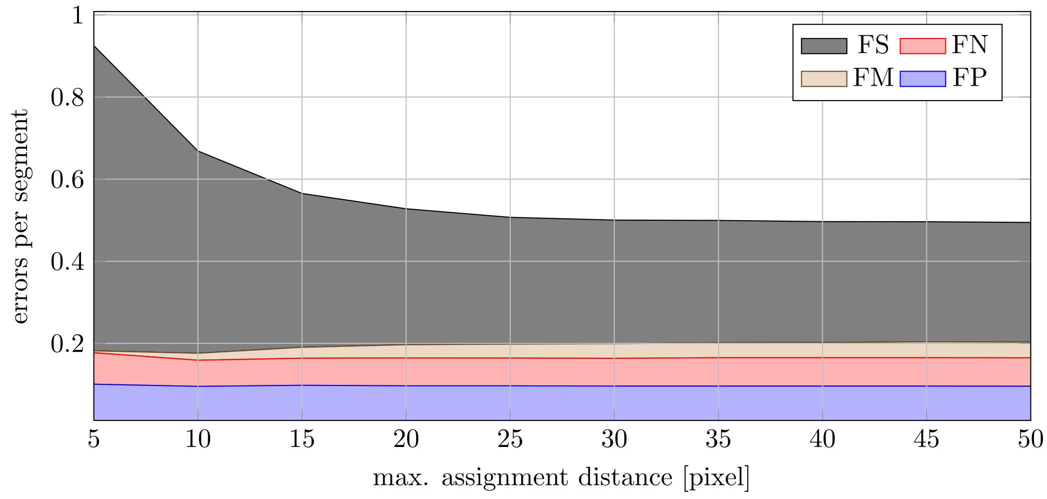


Figure 2.8: Influence of the maximal assignment distance on the errors of the result. After about 30 pixels there is no significant improvement.

no corresponding 2D neuron slice in the result is a *false negative* (FN) error, and vice versa for the *false positive* (FP) error. Details on how to compute this error measure are given in Section 6.2.5 on page 71. We normalized these errors by the number of segments in the test dataset, *i.e.*, by the sum of all neuron sections over all images.

2.6.4 Pipeline Parameters

Given a trained assignment classifier, only two parameters of our pipeline are free: the distance threshold θ_D for the generation of the assignment variables and the weight θ_S for the influence of the segmentation energy in the assignment costs. Errors for different values of θ_D can be found in Figure 2.8. Beyond a distance of about 30 pixels there is no significant improvement on the accuracy anymore. Changes due to perturbations of θ_S were very little. We found a value of 1.0 to work well.

2.6.5 Comparison

We compared our method to the segmentation fusion (SF) approach presented in [103], which was shown to be superior to other existing methods. There, the cost function for assignments is a linear function of weighted features of the involved candidates (x-y-displacement and cross-correlation). Since their approach does not provide a way to learn the feature weights from a training set, we performed a simple grid-search to pick weights that provided good results. However, what constitutes a good result is an application dependent question: A false positive might be harder to fix than a false negative. Therefore, instead of just taking the result with the best F1-score (SF-1), we decided to take two more results that reflect the range of the trade-off between false positives and negatives: One that had the smallest FN error (SF-2) and one that had the smallest FS error (SF-3). To keep the comparison fair, we performed the grid-search on the same five images that we used for training and not on the whole dataset. We also used the same candidates for both approaches to obtain an objective view on the assignment model performances. The results on the evaluation dataset can be seen in Figure 2.9. In the same figure we also give the errors of a best-effort solution based on the extracted candidates, *i.e.*, errors that stem from missing or wrong candidates that cannot be compensated for by the assignment search.

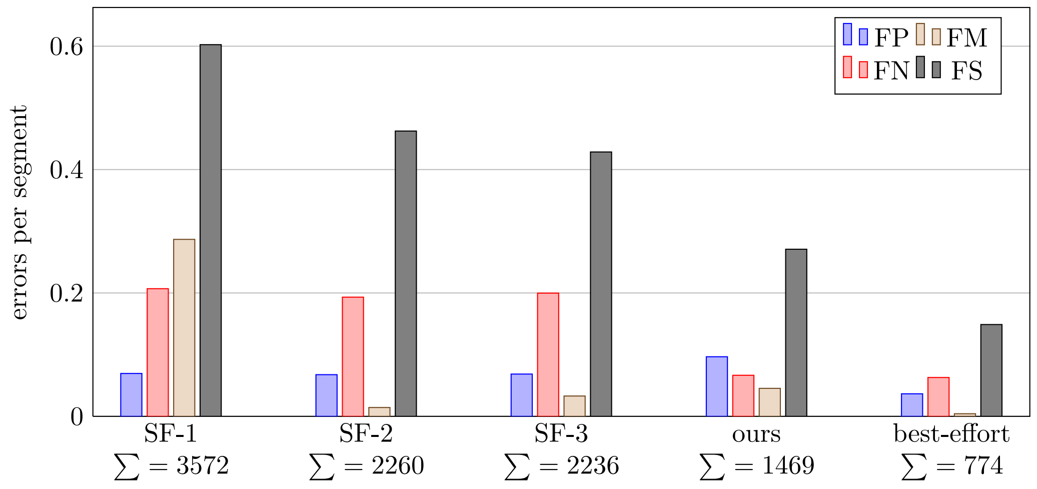


Figure 2.9: Comparative results to three different weight sets of the segmentation fusion approach [103] (SF-1 to SF-3) that have been found via grid-search on the test dataset. In total our results are closer to the lower error bound that is given by the extracted candidates.

2.6.6 Training

The training has been carried out with different numbers of (positive) training samples and classifier parameters, *i.e.*, the number of trees in the random forest and the number of features per node in the trees. The results with respect to the number of training samples can be seen in Figure 2.10. The performance converges after about 300 training samples. Regarding the training parameters, we found an optimum at 100 trees and 20 features per node, where further increase did not have a substantial effect.

2.6.7 Assignment Feature Importance

The importance of the assignment features according to the variable importance measure of the random forest classifier [11] can be found in Figure 2.11. The center distance between the candidates of an assignment is the most discriminative feature, followed by the differences of the normalized histogram and the set difference ratio.

2.6.8 Confidence Measure

To show the usability of our proposed confidence measure, we sorted all selected assignment variables by their confidence values and evaluated the errors that are contributed by the upper k%, *i.e.*, by the assignment variables our model is most confident about. For that, we limited our attention to the false positive and false merge errors, since the false negative and false split errors lose their meaning if we deliberately remove assignments from the solution. The errors have additionally been normalized by the number of assignment variables in the respective fraction to compensate for the decrease of false positive errors due to selecting less assignments.

The results can be seen in Figure 2.12. The 20% most confident assignment variables contribute only 0.00163 FMs per segment and 0.047 FPs per segment. In the top 10% there were no FMs at all and only 0.015 FPs per segment. We also found that the assignment variables with the lowest confidence measure contributed the most FMs and FPs.

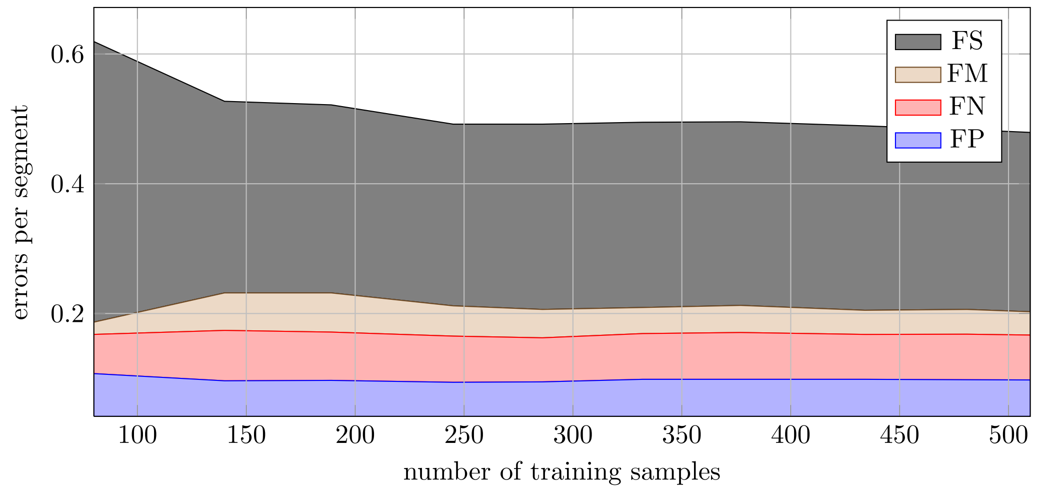


Figure 2.10: Influence of the number of training samples on the accuracy. After about 300 training samples there is almost no improvement.

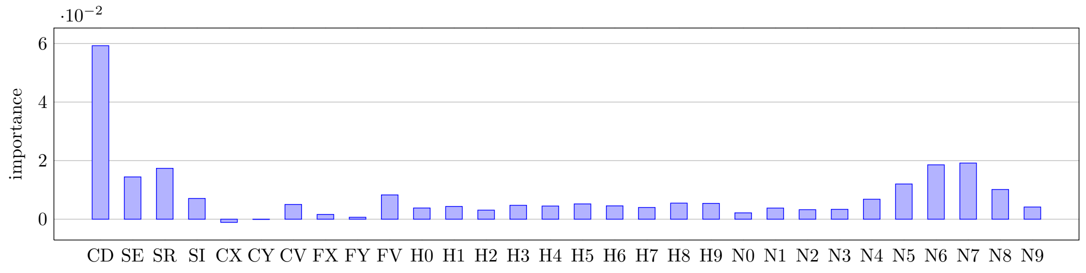


Figure 2.11: Variable importance of the used features as reported by the random forest classifier.

2.6.9 Inference Time

We solved the ILP to find the best set of assignments using the Gurobi solver[2], which is free for academic use. The average inference time for the whole stack was 6.2 seconds ($\pm$ 0.13), giving 2.02348 milliseconds ($\pm$ 0.04242) per segment on a 12 core Intel Xeon CPU at 3.47GHz.

2.7 Discussion

The presented automatic neuron reconstruction approach can be seen as an attempt to jointly segment and track neurons across a stack of 2D images. To keep inference tractable, the huge set of all possible 2D segmentations has been limited to use the 2D neuron candidates extracted from a sequence of graph-cuts. Although these 2D candidates still allow a lot of flexibility and help to resolve ambiguous cases, it should be noted that this limitation of segmentations does not come for free: About half of the final errors can be attributed to wrong or missing 2D neuron candidates. There is no doubt that the presented approach will benefit from better membrane predictions and hypotheses extraction strategies. In Chapter 3, we investigate the use of non-deterministically generated 2D neuron candidates by sampling from a probability distribution.

For the learning of the assignment costs, we could show that the use of a random forest classifier on a variety of features provides fewer final errors than a grid-search on a linear

[2]Gurobi Optimizer Version 4.6, www.gurobi.com

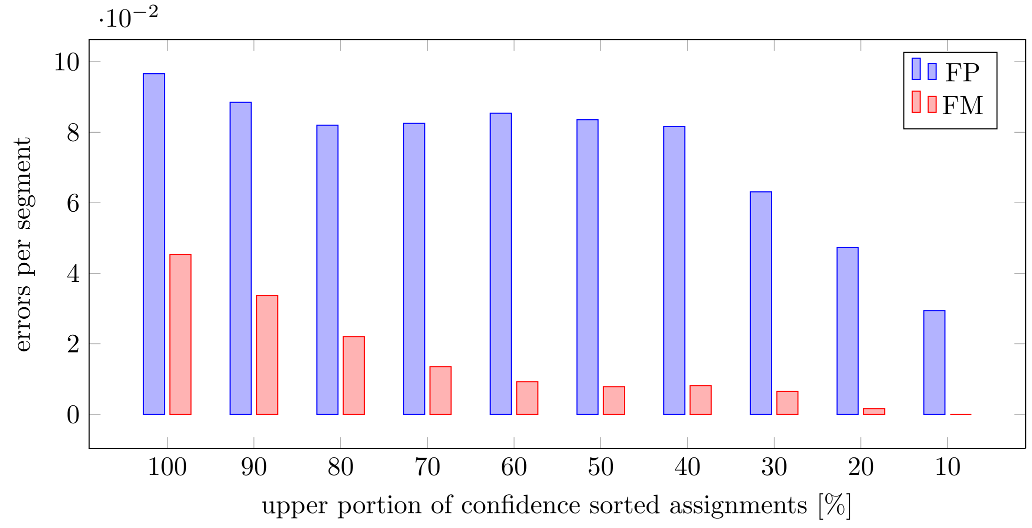


Figure 2.12: Relative FM and FP errors contributed by the upper $k\%$ of the confidence sorted assignments. To account for the differences in the solution sizes, the values have been normalized by the number of assignments in the respective portion. The upper 10% of the confidence sorted assignments do not contain FMs.

combination of a few features. This is surprising insofar as the random forest classifier was trained to classify positive and negative assignment samples, which does not necessarily relate to minimizing the final number of errors. An interesting question is whether a linear combination of the same set of features could perform better, if trained on a sensible objective. We answer this question in Chapter 4 in the context of the structured learning framework.

Currently, the output of any neuron segmentation algorithm must in practice be proofread by human experts. The confidence measure of our approach can be used to prioritize areas for human review. However, as we can see from our experiments, even the 10% most confident assignments contain some errors. To further improve this figure, we believe it is necessary to evaluate the confidence measure on a higher level then the one the model is using. In general, the confidence measure should take prior knowledge into account that is too hard to express directly in the model.

An interesting aspect of the presented approach is its efficiency. Although the model formulation is in general NP-hard, we observe small inference times that grow linearly with the problem size. Together with the fast training of the random forest classifier for assignment costs, this will allow us to use this method in an interactive user interface. Every correction made by a user can be used to retrain the random forest classifier and the problem can be resolved to propagate the changes made by the user.

Chapter 3

Sampling of 2D Neuron Candidates

3.1 Introduction

In this chapter[1], we present an alternative method for the extraction of 2D neuron candidates that are needed by the approach presented in Chapter 2. Instead of extracting candidates by a sequence of graph-cuts, we investigate candidates generated by sampling from a probabilistic model that was trained to label 2D images of neural tissue.

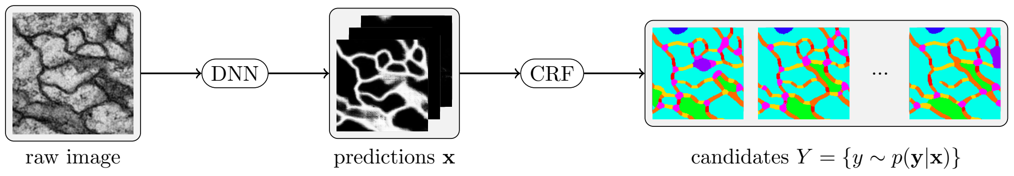
This work was motivated by the observation that about half of the errors of our approach can be attributed to missing or wrong 2D neuron candidates. Partly, this is due to the limitation that candidates are contained in component trees, as we obtain them by the sequence of graph-cuts that we used so far. This limitation makes it impossible to represent plausible candidates that are contradictory but not subsets of each other. By generating candidates as labeling samples from a probabilistic model, we overcome this limitation. Furthermore, the method used so far is not well suited to label ambiguous regions, since no prior about plausible candidates is incorporated. Here, we design and train a model to reflect the appearance statistics of several labels like membrane, cell interior, mitochondria, glia, and synapses in 2D sections of neural tissue.

3.1.1 Overview

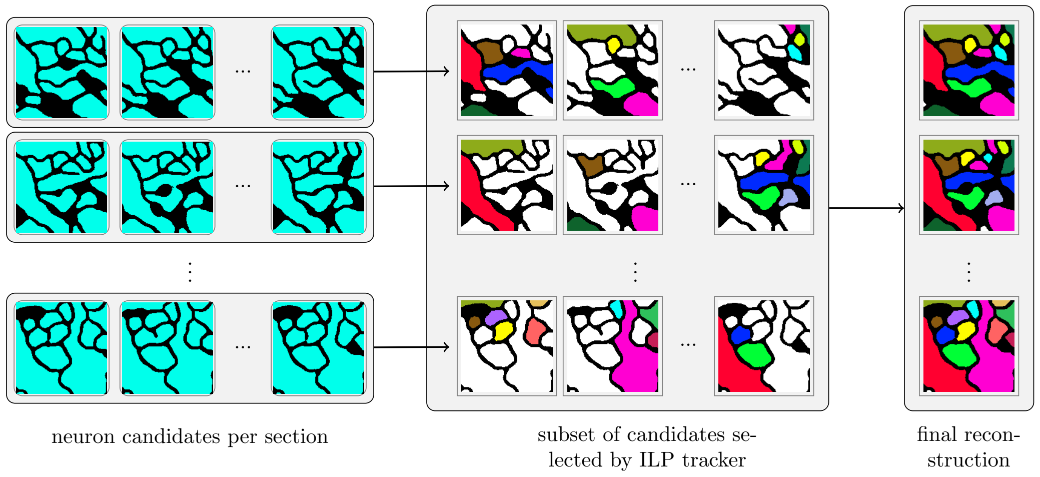
The key of our method is the generation of 2D neuron candidates as samples from a probabilistic model. Instead of generating a fixed and heuristic set of candidates, we draw samples from a conditional random field (CRF) that is designed and trained to label 2D sections of neural tissue. An overview of the proposed method is shown in Figure 3.1.

There are two important design choices in the proposed CRF to capture the appearance statistics of 2D sections of neural tissue and to keep inference tractable. First, the CRF exploits the orientation of cut membranes in EM sections to learn sensible priors for membranes, which are usually thin and elongated. As we show in Section 3.3, this helps to produce more plausible section labelings, and thus better 2D neuron candidates. Second, we restrict the interactions in the CRF to form a bipartite graph. With this trick, Gibbs sampling [29] on the model can be parallelized and carried out on a GPU with a speed-up of factor 39 compared to a single core CPU implementation. This allows us to model a large

[1]This chapter is based on our paper [27]. The results in this chapter represent the current state at the time of submission, and are outperformed by recent findings presented in Chapter 4. The training of the deep neural network used for the experiments of this chapter has been carried out by Dan C. Ciresan, Alessandro Giusti, and Luca M. Gambardella, members of Jürgen Schmidhuber's group at IDSIA, Lugano. Ground truth annotation and training data preparation was carried out together with Stephan Gerhard and Julien Martel, PhD students at the Institute of Neuroinformatics, UZH/ETH Zürich.



(a) Candidate generation for a single section.


neuron candidates per section
subset of candidates selected by ILP tracker
final reconstruction

(b) Neuron reconstruction for the whole stack.

Figure 3.1: Overview of the proposed method. (a) For each section, a deep neural network (DNN) predicts per-pixel labels. Using these predictions, 2D neuron candidates are generated by sampling from a CRF. (b) An integer linear program (ILP) tracker is used to find a consistent subset of the candidates for the whole stack at once.

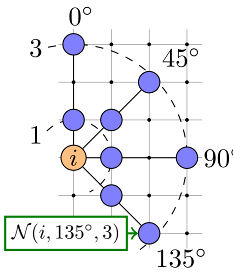
number of neighbor interactions for each pixel (and thus improve accuracy) and to use a rich set of biologically relevant labels (cell interior, mitochondria, glia cells, synapses, and different orientations of membranes) while still being fast enough to process large amounts of data. Interestingly, we found that this restriction does not decrease the quality of the generated candidates compared to a non-bipartite version of our model.

In Section 3.2 we give details about the proposed CRF that we use to generate 2D neuron candidates. In Section 3.3, we show results on a stack of *Drosophila melanogaster* neural tissue, where the sampled candidates produce 30% less errors compared to other candidate generation methods.

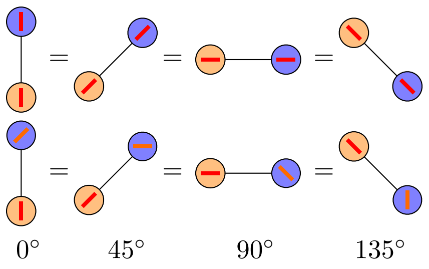
3.1.2 Related Work

Current state-of-the-art approaches for anisotropic neuron reconstruction require the extraction of 2D neuron candidates for each section [26, 50, 103]. For that, membranes are identified based on predictions of local image patch classifiers like deep neural networks (DNNs) [18] or random forest classifiers [26, 50, 103]. From these predictions, candidates are obtained by a sequence of watersheds [103] or a sequence of graph-cuts [26, 50]. A cost-minimal and globally consistent subset of the 2D neuron candidates and their assignment across sections is then found by tracking the candidates across sections via an integer linear program (ILP) formulation. The ILP tracker ensures that a non-contradictory subset of candidates is chosen and that the candidates are connected to optimize a global criterion like a smooth continuation.

In all those approaches, the accuracy and efficiency of the reconstruction is limited by



(a) neighborhood



(b) rotation equivariance

Figure 3.2: (a) Neighbors (blue) of a single pixel (i, orange) for orientations $\Phi = \{0°, 45°, 90°, 135°\}$ and distances $\Delta = \{1, 3\}$. Neighbors are approximated to the closest pixel on the grid in the given direction and distance. (b) Illustration of the rotation equivariant interactions. The values of rotated interactions are equal for accordingly rotated labels. For the evaluation of a pairwise interaction, the labels are rotated in the opposite direction of the interaction orientation and thus are normalized to the vertical (0°) case.

the quality of the generated 2D neuron candidates. It has been noted that the generation of 2D neuron candidates is responsible for about 50% of the final error [26]. Although the ILP tracker can ignore wrong candidates to some extend, it can not fantasize missing correct candidates. Thus, it is important to generate enough candidates with high variation to make sure that the correct candidate is amongst them. However, too many candidates increase the computational overhead and may lead to spurious results.

3.2 2D Neuron Candidate Sampling

To generate 2D neuron candidates, we repeatedly draw samples from a labeling distribution on each section individually and transform them into binary neuron/membrane segmentations. Let $\Omega \subset \mathbb{N}^2$ be the pixel domain of a single EM section. We model the distribution of labelings $\mathbf{y} = \left(y^{(i)} \in K \mid i \in \Omega\right)$ of assigning each pixel in Ω to a label of a discrete set K with a conditional random field (CRF).

For the possible values in K, we distinguish between *oriented* labels K_Φ (for membranes) and *non-oriented* labels K_N (for mitochondria, glia cells, *etc.*), such that $K = K_\Phi \cup K_N$ and $K_\Phi \cap K_N = \varnothing$. Each $k_\alpha \in K_\Phi$ represents a label k at a certain discrete orientation $\alpha \in \Phi$.

The CRF is conditioned on a pixel-wise prediction vector field $\mathbf{x} = \left(\mathbf{x}^{(i)} \in \mathbb{R}^F \mid i \in \Omega\right)$ with vectors of size F. We write $y^{(i)}$ or $\mathbf{x}^{(i)}$ to refer respectively to the label or prediction vector of pixel $i \in \Omega$ and $x_f^{(i)}$ to refer to the f$^{\text{th}}$ prediction component of $\mathbf{x}^{(i)}$. The lateral interactions of each location in the CRF are modeled with pairwise factors to neighbors in different directions Φ (as for the oriented labels) and distances $\Delta = \{d_1, \ldots, d_D\}$, approximated to the closest neighbor on the pixel grid (see Figure 3.2a for an illustration). We write $\mathcal{N}(i, \alpha, d)$ to denote the closest grid neighbor of i in direction α and distance d. We achieve rotation equivariance by re-using factors of same distance for interactions in different directions to directly model the rotation invariant statistics of our data. Furthermore, the CRF is homogeneous, *i.e.*, the same factors are used at every location.

Formally, we model the labeling distribution $p(\mathbf{y}|\mathbf{x})$ as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left[-E_P(\mathbf{y}) - E_D(\mathbf{y}, \mathbf{x})\right], \tag{3.1}$$

where $Z(\mathbf{x})$ is the partition function and the energies E_P and E_D correspond to the prior and the data term, respectively. The data term is a linear combination of the components of the prediction vectors with weights $\boldsymbol{w}^k \in \mathbb{R}^F$ for each label $k \in K$, *i.e.*,

$$E_D(\mathbf{y}, \mathbf{x}) = \sum_{i\in\Omega} \langle \boldsymbol{w}^{y^{(i)}}, \mathbf{x}^{(i)} \rangle. \tag{3.2}$$

The prior E_P models the pairwise interactions of each location to neighbors defined by a set of discrete orientations Φ and distances Δ:

$$E_P(\mathbf{y}) = \sum_{i\in\Omega} \sum_{\alpha\in\Phi} \sum_{d\in\Delta} R_{\alpha,d}(y^{(i)}, y^{(\mathcal{N}(i,\alpha,d))}), \tag{3.3}$$

where $R_{\alpha,d}$ determines the costs for the joint labeling of i and its approximate neighbor in direction α at distance d. The costs $R_{\alpha,d}$ are shared across different orientations, *i.e.*,

$$R_{\alpha,d}(k,l) = \tilde{R}_d \left(k^{-\alpha}, l^{-\alpha}\right), \tag{3.4}$$

where we write k^α to denote the rotated version of label k by α degrees: oriented labels change their orientation subscript and non-oriented labels do not change at all. This rotation operation ensures that the resulting CRF is rotationally equivariant by treating every pairwise interaction in the same way as the $0°$ interaction (see Figure 3.2b for an illustration). Finally, $\tilde{R}_d$ is a lookup table with entries

$$\tilde{R}_d\left(k,l\right) = v_d^{k,l} \tag{3.5}$$

for the joint labeling of neighboring pixels with distance d.

3.2.1 Learning

The parameters of our model are $\boldsymbol{\theta} = (\mathbf{v}, \boldsymbol{w})$, where

$$\mathbf{v} = \left\{ v_d^{k,l} \mid d \in \Delta;\ k,l \in K \right\} \tag{3.6}$$

are the costs for pairwise interactions and

$$\boldsymbol{w} = \left\{ w_f^k \mid f \in 0, \ldots, F;\ k \in K \right\} \tag{3.7}$$

are the weights of the prediction values. Given an annotated learning sample $(\mathbf{y}, \mathbf{x})$, we obtain the parameters by maximum likelihood learning [95]. The conditional log-likelihood of a learning sample is the negative log-probability

$$L(\boldsymbol{\theta}) = -\log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}), \tag{3.8}$$

that we optimize using stochastic gradient descent. Substituting Eq. (3.1) and building the derivative with respect to the two types of parameters yields the following gradients:

$$\frac{\partial L(\boldsymbol{\theta})}{\partial w_f^k} = \sum_{\substack{i\in\Omega: \\ y^{(i)}=k}} x_f^{(i)} - \sum_{i\in\Omega} p(y^{(i)} = k|\mathbf{x}) x_f^{(i)} \tag{3.9}$$

$$\frac{\partial L(\boldsymbol{\theta})}{\partial v_d^{k,l}} = \sum_{\substack{i\in\Omega \\ \alpha\in\Phi}} \delta_{y^{(i)}=k^\alpha} \delta_{y^{(i')}=l^\alpha} - \sum_{\substack{i\in\Omega \\ \alpha\in\Phi}} p(y^{(i)} = k^\alpha, y^{(i')} = l^\alpha|\mathbf{x}), \tag{3.10}$$

where δ is the Kronecker-delta and i' is a shorthand for $\mathcal{N}(i,\alpha,d)$. The marginal probabilities are obtained via Gibbs sampling, that we describe in the following section.

3.2.2 Parallelized Sampling

We use Gibbs sampling [29] to draw samples from $p(\mathbf{y}|\mathbf{x})$ that are needed for the training and generation of 2D neuron candidates. To tackle the slow convergence properties of Gibbs sampling, we restrict the interactions in the CRF to form a bipartite graph. For that, we divide the image domain Ω into "odd" and "even" locations, following a checkerboard pattern on the pixel grid, such that $\Omega = \Omega_O \cup \Omega_E$ and $\Omega_O \cap \Omega_E = \varnothing$. By modifying $\mathcal{N}$ to find the closest neighbor in the given direction and distance *of opposite parity*, we obtain a bipartite CRF. We write $\mathbf{y}^{(O)}$ and $\mathbf{y}^{(E)}$ to refer to the labeling of the pixels in Ω_O and Ω_E, respectively. It follows that

$$p(\mathbf{y}^{(E)}|\mathbf{y}^{(O)}, \mathbf{x}) = \prod_{i \in \Omega_E} p(y^{(i)}|\mathbf{y}^{(O)}, \mathbf{x}), \tag{3.11}$$

and

$$p(\mathbf{y}^{(O)}|\mathbf{y}^{(E)}, \mathbf{x}) = \prod_{i \in \Omega_O} p(y^{(i)}|\mathbf{y}^{(E)}, \mathbf{x}), \tag{3.12}$$

i.e., labels in one partition are conditionally independent given the labels in the other half [33]. Samples $y^{(i)} \sim p(y^{(i)}|\mathbf{y}^{(E)})$ for $i \in \Omega_O$ and $y^{(i)} \sim p(y^{(i)}|\mathbf{y}^{(O)})$ for $i \in \Omega_W$ can be drawn independently and in parallel. We exploit this property by sampling a whole section in two half-steps, each of which carried out with our GPU implementation [23].

3.3 Results

We evaluate the performance of our 2D neuron candidate generation method on two stacks of 20 EM images of size 1024×1024 from *Drosophila melanogaster* larva neuropil with 4.6nm xy-resolution and 50nm section thickness [31]. These stacks have been selected to capture well the typical variations found in EM images. The first stack was used to train the DNN architecture proposed in [18] to predict the pixel labels (shown in Figure 3.3) that are used as features $\mathbf{x}$ in the CRF. The second stack, for which we manually generated ground truth, is used to compare different candidate generation methods. To investigate the effect of different neighborhood sizes, we created three instances of our model with different distance sets: model D_1 (a baseline of our CRF) uses $\Delta = \{1\}$, D_4 uses $\Delta = \{1, 5, 9, 15\}$, and D_8 uses $\Delta = \{1, 3, 5, 7, 9, 15, 21, 31\}$. All models use the same labels (shown in Figure 3.3) and the same set of rotations $\Phi = \{0°, 45°, 90°, 135°\}$. From each model, we drew 20 samples per section to generate the 2D neuron candidates. Each sample is the last labeling after 100 Gibbs iterations on the whole section, starting from a random initialization.

3.3.1 Error Measure

The errors made by current neuron reconstruction methods render a manual proof reading necessary. Minimizing the time that has to be spent on correcting the result is therefore a sensible objective. Ideally, the evaluation measure on automatic reconstruction methods should reflect the time and effort needed to correct the errors made by an automatic method. Therefore, we measure the error of a reconstruction in terms of the edit distance to the ground truth, since this directly reflects the amount of time needed to fix it. Due to the different types of errors in anisotropic reconstruction, we suggest to provide errors in four categories: "FP" (false positives) are spuriously detected neuron slices in a section, "FN" (false negatives) are missed neuron slices in a section, "FS" (false splits) missed links between neuron slices across sections, and "FM" (false merges) are spurious links across sections. A neuron slice is considered to be detected, if it overlaps to at least 50% with the ground truth. Details about this error measure are given in Section 6.2.5 on page 71.

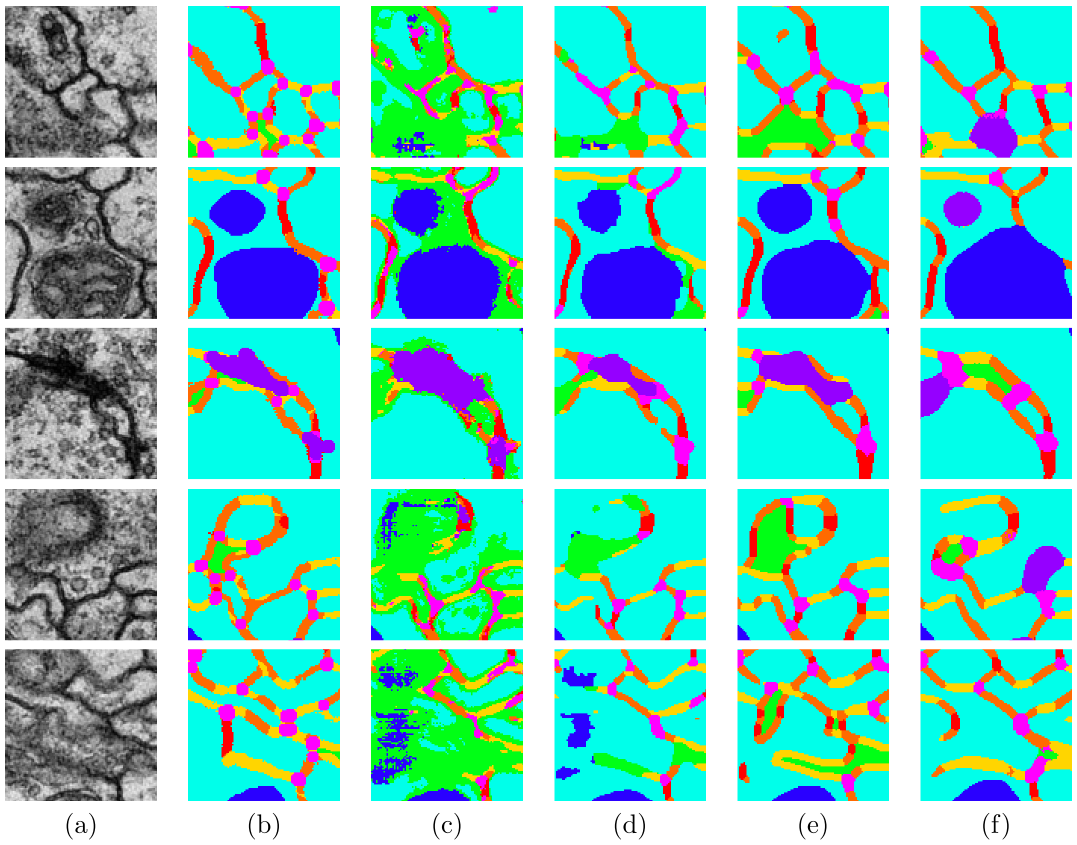


Figure 3.3: Examples of samples drawn from our CRF for ambiguous cases. The used labels are membrane (four orientations |,/,—,\ and junctions ●), cell interior (●), mitochondria (●), glia (●), and synapse (●). (a) shows the raw images, (b) the ground truth labelings, (c) the max-prediction labeling, (d-f) a representative sample from D_1, D_4, and D_8, respectively.

3.3.2 Comparison

We compare our model instances D_1, D_4, and D_8 to the current state of the art in neuron candidate generation: A series of graph-cuts (GraphCuts), applied on the *membrane* predictions of the DNN, as proposed in [26], and gap completion (GapComp) [52] as proposed in [50]. As a baseline, we also generate candidates from component trees (CompTrees) [45], extracted from the same predictions. We reconstructed neurons in the test stack for each candidate generation method using the publicly available ILP tracker SOPNET [26]. The trainable parts of this tracker have been trained and validated for each method individually on the first 10 sections of the test stack. The test results on the last 10 sections are shown in Figure 3.4. Model D_4 provides by far the best result, improving the error rate by 30% compared to the best competing approach (GraphCuts). Surprisingly, the larger neighborhood of model D_8 is inferior to D_4.

2D neuron candidates generated with our and the compared methods are shown in Figure 3.6. Each method is using the predictions of our trained DNN. No post-processing was applied. Compared to the other methods, hypotheses generated with our method (D_4 in the figure) appear smoother and propose a clearer segregation between neurons.

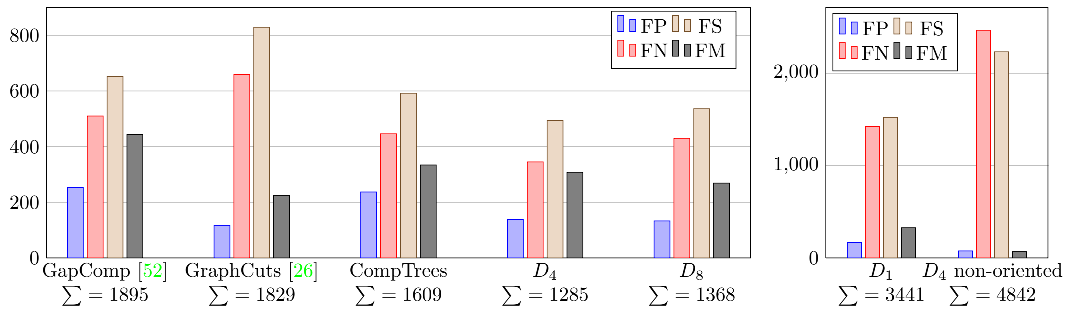


Figure 3.4: Reconstruction errors for different 2D neuron candidate generation methods (a). Results are given as false positives (FP), false negatives (FN), false splits (FS), and false merges (FM), see text for details. Our method D_4 produces 30% less errors compared to the best competing approach (GraphCuts). The importance of a rich neighborhood and oriented membranes in the proposed model is shown in (b): Decreasing the neighborhood radius to one (D_1) or ignoring membrane orientations (D_4 non-oriented) dramatically sacrifices reconstruction accuracy.

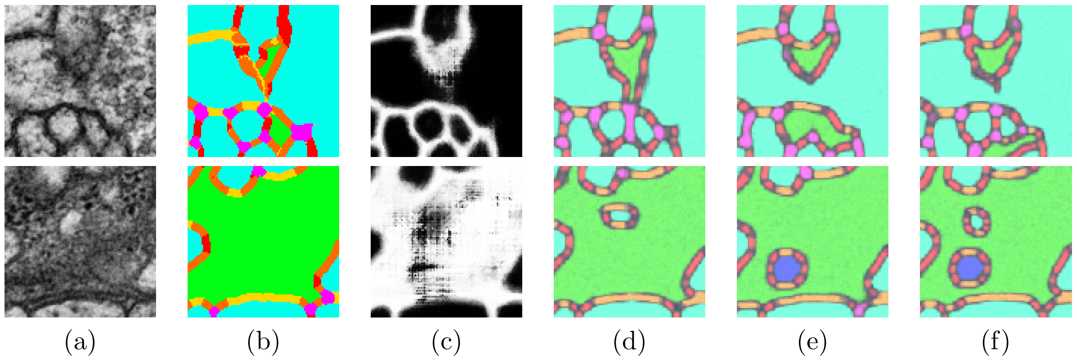


Figure 3.5: Demonstration of the candidate generation capabilities of model D_4 (for label legend see caption of Figure 3.3). (a) shows the raw image, (b) the ground truth labeling and (c) the prediction for the class *membrane*. Images (d-f) show three different samples drawn from D_4.

3.3.3 Model Properties

To show the importance of oriented membrane labels, we trained and evaluated a version of D_4 with non-oriented membrane labels (right bars in Figure 3.4). Due to undersegmentation, the model proposed only a few large candidates and thus missed a lot of neurons, which is reflected in a high number of false negatives (FN) and false splits across sections (FS). To investigate the effect of the bipartite restriction of our CRF, we also implemented a non-bipartite version of model D_4. The reconstruction errors differ by 1% in favour of the bipartite version and may well be attributed to sampling noise.

3.3.4 Inference Time

Tested on a NVIDIA Quadro 4000, we were able to draw 100 complete samples of size 1024×1024 from D_4 using our GPU implementation in $4.4s$. A single core CPU Gibbs sampler implementation[2] took $174s$ on a Intel Xeon CPU at 3.47 GHz to achieve the same, resulting in a speed-up factor of 39.54.

[2]from OpenGm: http://hci.iwr.uni-heidelberg.de/opengm2/

3.4 Discussion

We showed that replacing fixed sets of 2D neuron candidates with samples drawn from a distribution increases the reconstruction accuracy. This scheme might not just be limited to neuron reconstruction, but also be applied to other approaches that rely on the quality of initial candidates, like super-pixel based algorithms for image segmentation.

In our case, an interesting side effect of our method is the rich labeling of the candidates: Besides membrane locations, the sampled candidates also propose the locations of mitochondria, synapses, and glia cells. These propositions are not only of biological relevance, but could also be used to improve the reconstruction accuracy by, for instance, exploiting the fact that mitochondria are surrounded by neuron and synapses are separating neurons.

An open question in our method is how many samples need to be drawn to obtain good results with little computational overhead. An interesting solution might be to start with few initial samples and draw more on demand for locations that are unlikely according to higher level priors, like a sudden change in direction or an unexpected end of a neural process.

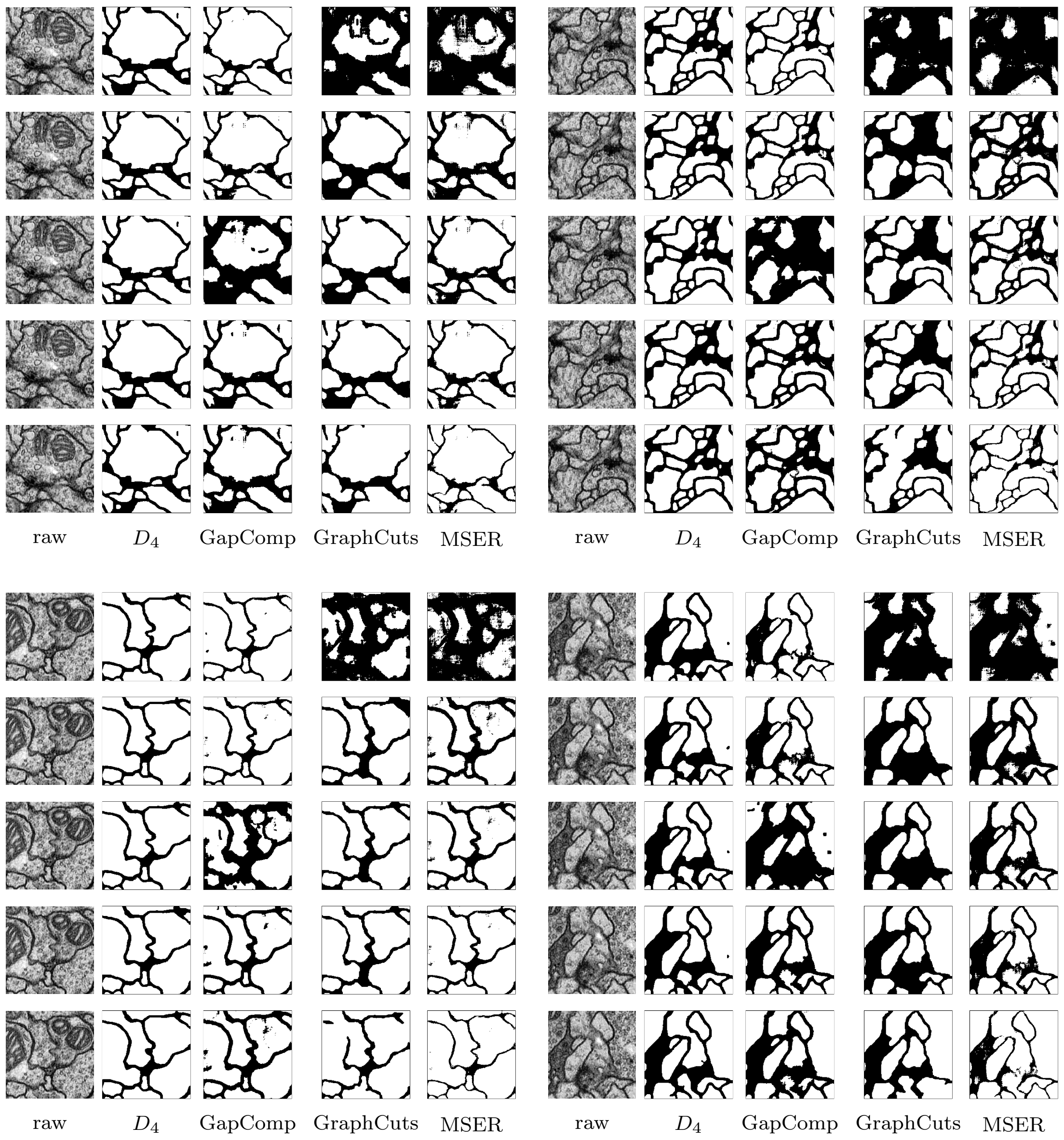


Figure 3.6: Qualitative comparison of the 2D neuron candidates of the used methods. Each block shows five randomly selected neuron vs. background segmentation hypotheses (columns) for each of the used methods (D_4 being our method) on a sample patch of size 200×200.

Chapter 4

Structured Learning of Assignment Costs[1]

4.1 Introduction

A complete automatic neuron reconstruction pipeline like the one we described in the previous chapters naturally has a lot of free parameters. Starting with the low-level image processing, there are a number of local features that are combined in a non-linear way to obtain per-pixel predictions. We achieved the best results by using the convolutional neural network (CNN) of Ciresan et al. for that, which introduces thousands of weight parameters [18]. From the CNN predictions, 2D neuron candidates are extracted. Using the sampling method introduced in Chapter 3 introduces another few hundred parameters for the CRF. From these candidates, possible assignments across sections are enumerated. For each of these assignments, costs have to be determined to rate their likelihood to be part of the final reconstruction.

Given annotated ground truth, each of the mentioned parts can be naively trained with dedicated algorithms. The CNN was trained by Ciresan et al. to minimize the classification error on a set of labeled image patches. The parameters of our CRF are obtained via maximum likelihood learning on fully labeled EM sections. Finally, we learn the assignment costs from positive and negative assignment samples using a random forest classifier [11].

However, this part-wise learning of parameters is not optimal: The individual training objectives may only poorly reflect the overall objective, which is to make less split and merge errors in the final reconstruction. For example, a CNN trained to classify as many patches correctly as possible does not necessarily do the best to avoid reconstruction errors. A few mislabeled pixels might be responsible for a split or merge in the final result, even though they contributed only little to the pixel-wise error rate. We hypothesize that the part-wise learning does not exploit the full capacity of the classifiers in the pipeline to minimize the final reconstruction error. Due to noise, ambiguous data, and only local information we can assume that none of the classifiers is perfect. Given that, we might want to shift the attention of the classifier towards critical cases. For instance, it might not be necessary to learn that a certain assignment type has high costs, if it would never be selected by the higher inference model anyway. Such a situation can arise if another conflicting assignment is already very likely to be part of the final reconstruction.

For these reasons, it would be desirable to train all free parameters of the neuron reconstruction pipeline at once to minimize the final error. Unfortunately, it is not obvious

[1]The training of the deep neural network used for the experiments of this chapter has been carried out by Dan C. Ciresan, Alessandro Giusti, and Luca M. Gambardella, members of Jürgen Schmidhuber's group at IDSIA, Lugano. Ground truth annotation and training data preparation was carried out together with Stephan Gerhard and Julien Martel, PhD students at the Institute of Neuroinformatics, UZH/ETH Zürich.

how to derive a learning signal for early parts of the pipeline based on errors in the resulting reconstruction. Already the training of a CNN on pixel classification error is suffering from vanishing gradients and non-convexity. This situation gets even worse if the learning signal depends on the pipeline outcome: There is no explicit functional relation between a parameter in early parts of the pipeline and the final error.

We can, however, expect improvements for the training of late parts in the pipeline. In particular, the learning of the costs for assignments can be made aware of the final reconstruction error. In the following, we show how this can be done within the structured learning framework[98]. In Section 4.2, we give an overview over this learning framework. We provide an implementation of this framework for discrete energy based models on arbitrary loss functions. Our implementation is explained in detail in Section 4.3. In Section 4.4 we show how the training of the assignment costs of our neuron reconstruction pipeline can be carried out with our implementation.

In Section Section 4.5, we investigate the effect of structured learning on the final error for different candidate generation methods. The results with structured learning are superior to learning the assignment costs using a random forest classifier (as described in Section 2.5), a support vector machine, and optimizing for maximum overlap only (which we found to work very well in practice so far). We also show that the outcome is robust to changes in the regularizer weight (the only free parameter in the structured learning framework), thus leaving effectively no free parameters in the last part of the neuron reconstruction pipeline.

4.2 Structured Learning

Structured learning addresses the problem of estimating the parameters of a mapping function $f : \mathcal{X} \mapsto \mathcal{Y}$ of some observations $\mathbf{x} \in \mathcal{X}$ to a discrete response $\mathbf{y} \in \mathcal{Y}$ [98]. Given training samples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(l)}, \mathbf{y}^{(l)}) \in \mathcal{X} \times \mathcal{Y}$, the parameters are found to minimize the empirical risk, which is defined in terms of an application specific cost function for responses that deviate from the desired training sample responses. This learning framework is called *structured* to account for the fact that the $\mathbf{y} \in \mathcal{Y}$ can represent application specific structures in contrast to being a plain classifier. This way it is possible to train directly on the desired output of a given application, for instance body part positions of a stickman model fitted to an image [10, 76, 109].

More formally, the mapping f is assumed to be the minimizer of an energy based model that scores pairs of observations and responses, *i.e.*,

$$f(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\arg\min} \quad E(\mathbf{x}, \mathbf{y}). \tag{4.1}$$

We assume that the energy $E(\mathbf{x}, \mathbf{y})$ is a linear combination of arbitrary real valued features on the combined input and output space $\phi_i(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, *i.e.*,

$$E(\mathbf{x}, \mathbf{y}) = \langle \boldsymbol{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle, \tag{4.2}$$

where we write $\phi(\mathbf{x}, \mathbf{y})$ to denote the vector of all features $\phi_i(\mathbf{x}, \mathbf{y})$. Note that this is a very general formulation and does not limit the expressiveness of the model. In particular, this formulation is compatible to the energy term of the very broad class of exponential family models, where the $\phi_i(\mathbf{x}, \mathbf{y})$ have a natural interpretation as sufficient statistics [106].

In the following, we assume the presence of only a single training sample $(\mathbf{x}', \mathbf{y}') \in \mathcal{X} \times \mathcal{Y}$, since multiple training samples can be concatenated into one by concatenating $\mathcal{X}$ and $\mathcal{Y}$ accordingly. Now, the optimal weight vector $\boldsymbol{w}^*$ is found by minimizing the empirical risk $J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$, *i.e.*,

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\arg\min} \quad J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}'), \tag{4.3}$$

where the empirical risk is

$$J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}') = \Omega(\boldsymbol{w}) + L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}'), \tag{4.4}$$

with $\Omega(\boldsymbol{w})$ being a convex regularizer on $\boldsymbol{w}$ and $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ the training sample loss. Finally, the loss establishes the conceptual link to maximum margin problems [98]: It is the maximal energy difference

$$\delta E(\mathbf{y}', \mathbf{y}) = E(\mathbf{x}', \mathbf{y}') - E(\mathbf{x}', \mathbf{y}) \tag{4.5}$$

of a response $\mathbf{y}$ to the ground truth $\mathbf{y}'$, modulated by an application specific cost $\Delta(\mathbf{y}', \mathbf{y})$. More formally,

$$L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}') = \max_{y \in \mathcal{Y}} \, \Gamma(\mathbf{y}', \mathbf{y}) \delta E(\mathbf{y}', \mathbf{y}) + \Delta(\mathbf{y}', \mathbf{y}), \tag{4.6}$$

where $\Delta(\mathbf{y}', \mathbf{y})$ is left undefined and fulfills

$$\Delta(\mathbf{y}', \mathbf{y}) \geq 0 \;\text{ and }\; \Delta(\mathbf{y}', \mathbf{y}') = 0, \tag{4.7}$$

and $\Gamma(\mathbf{y}', \mathbf{y})$ serves as a placeholder to switch between the margin rescaling variant (if $\Gamma(\mathbf{y}', \mathbf{y}) = 1$) and the slack rescaling variant (if $\Gamma(\mathbf{y}', \mathbf{y}) = \Delta(\mathbf{y}', \mathbf{y})$) [97]. With this formulation, the sample loss is the maximum over all possible responses $\mathbf{y}$ of the energy difference of $\mathbf{y}$ to the ground truth $\mathbf{y}'$, modulated either linearly or constantly with some costs $\Delta(\mathbf{y}', \mathbf{y})$ for false responses. This cost function plays an important role in the structured learning framework: Instead of just maximizing the energy margin between the ground truth $\mathbf{y}'$ and any other $\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}'\}$, the optimization takes into account how wrong a particular response is, and raises its energy the more the higher the corresponding $\Delta(\mathbf{y}', \mathbf{y})$ is. Provided enough training data, this minimizes the expected response cost over all possible observations $\mathbf{x}$.

4.3 Bundle Method for Structured Risk Minimization

In the following, we describe our implementation of a bundle method to minimize the structured risk $J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ introduced in the previous section for general models with quadratic regularizers.

For that, we assume without loss of generality[2], that the set $\mathcal{Y}$ of responses is the set of all binary vectors of length n subject to linear constraints

$$\mathcal{Y} = \{\mathbf{y} \in \{0, 1\}^n | A\mathbf{y} \preceq \mathbf{b}\}, \tag{4.8}$$

where we write $\mathbf{a} \preceq \mathbf{b}$ to say that $\mathbf{a}$ is element-wise smaller or equal to $\mathbf{b}$. Again without loss of generality[3], we assume further that the energy $E(\mathbf{x}, \mathbf{y})$ is defined in terms of feature vectors $\phi_i(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}^m$ for each component y_i of $\mathbf{y}$, where the binary y_i enable or disable the contribution of feature vector $\phi_i(\mathbf{x})$ to the energy. More formally, we define the energy to be of the form

$$E(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \langle \boldsymbol{w}, \phi_i(\mathbf{x}) \rangle \, y_i = \langle \boldsymbol{w}, \phi(\mathbf{x})\mathbf{y} \rangle \, , \tag{4.9}$$

where we write $\phi(\mathbf{x})$ to denote the matrix of all feature vectors $\phi_i(\mathbf{x})$ as columns. We refer to models satisfying Eq. (4.8) and Eq. (4.9) as linear constrained binary models.

In our implementation, we restrict $\Delta(\mathbf{y}', \mathbf{y})$ to be linear in $\mathbf{y}$ as well, *i.e.*,

$$\Delta(\mathbf{y}', \mathbf{y}) = \langle \Delta_\mathbf{l}, \mathbf{y} \rangle + \Delta_c, \tag{4.10}$$

where $\Delta_\mathbf{l} \in \mathbb{R}^n$ and $\Delta_c \in \mathbb{R}$ depend on $\mathbf{y}'$. This restriction is not sacrificing expressiveness of the cost function, since arbitrarily complex cost functions can be implemented by augmenting the vector $\mathbf{y}$ and introducing new constraints[4]. The resulting model will still be a linear constrained binary model.

[2]Every discrete energy model can be transformed into an equivalent model with binary indicator variables and linear constraints that ensure consistency between the indicator variables. See Appendix A.1 for the exact procedure.

[3]Every energy formulation of the form presented in Eq. (4.2) can be rewritten as in Eq. (4.9) after the transformation to binary indicator variables. See Appendix A.2 for the exact procedure.

[4]By treating the cost function as a factor graph, we can use the same procedure as described in Appendix A.1.

Put together, the restrictions in Eq. (4.8), Eq. (4.9), and Eq. (4.10) allow us to perform the maximization of the margin rescaling variant of the loss, *i.e.*,

$$L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}') = \max_{y \in \mathcal{Y}} \delta E(\mathbf{y}', \mathbf{y}) + \Delta(\mathbf{y}', \mathbf{y}), \tag{4.11}$$

by solving an integer linear program (ILP), without losing generality of the models that can be handled. As we will see soon, this maximization has to be carried out repeatedly to find the optimal weights.

Our implementation uses a quadratic regularizer, *i.e.*, $\Omega(\boldsymbol{w}) = \lambda|\boldsymbol{w}|^2$. In this case, the empirical risk $J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ is piecewise quadratic and convex in $\boldsymbol{w}$, despite the possible flexibility that $\Delta(\mathbf{y}', \mathbf{y})$ allows. As we see in Figure 4.1, the reason for that is that $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ is the maximum over a set of hyperplanes. Thus, $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ is convex in $\boldsymbol{w}$ and so is $J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$, which is just the sum of a quadratic regularizer and $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$. Theoretically, the optimal set of weights could thus be found by solving the quadratic program

$$\min_{\boldsymbol{w}, \xi} \lambda|\boldsymbol{w}|^2 + \xi, \text{ s.t. } \forall \mathbf{y} \in \mathcal{Y} : \langle \mathbf{a}_{\mathbf{y}}, \boldsymbol{w} \rangle + b_{\mathbf{y}} \leq \xi, \tag{4.12}$$

where $\langle \mathbf{a}_{\mathbf{y}}, \boldsymbol{w} \rangle + b_{\mathbf{y}} = 0$ defines the hyperplane corresponding to configuration $\mathbf{y}$. In practice, however, the enumeration of the exponentially many constraints (one for each configuration $\mathbf{y} \in \mathcal{Y}$) is intractable.

To avoid enumeration of all those hyperplanes, we exploit the fact that around the minimum, $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ can be described by only a few hyperplanes. We use a cutting plane method to discover those hyperplanes. For that, we repeatedly solve Eq. (4.11) for different weight vectors. Each found hyperplane is used to refine a lower bound $\mathcal{J}(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ of $J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$, which we minimize to obtain the next weight vector to sample. Our implementation follows the framework of *Bundle Methods for Regularized Risk Minimization* described in [97].

In Algorithm 1, Algorithm 2, and Algorithm 3 we provide details about the cutting plane technique we use. Our generic implementation for linear constraint binary models is made available in [24].

4.4 Application to Neuron Reconstruction

The application of the structured learning algorithm presented in the previous chapter to our model for neuron reconstruction (described in detail in Chapter 2) is almost straightforward. Our model is already a member of the class of linear constraint binary models, so that no further transformation is necessary. The components of the vectors $\mathbf{y}$ are binary variables, each indicating the selection and assignment of 2D neuron candidates across sections. The set $\mathcal{Y}$ is the set of all consistent solutions, *i.e.*, all solutions that fulfill the linear constraints introduced in Section 2.3. It remains to construct feature vectors for each binary assignment variable, find a way to obtain a training sample $(\mathbf{x}', \mathbf{y}')$, and to model a sensible cost function.

4.4.1 Feature Vectors

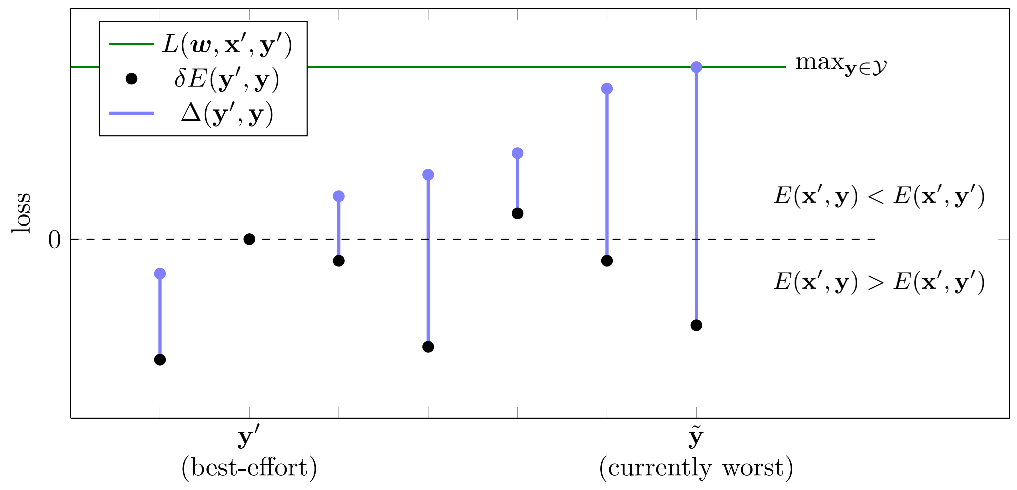
We reformulate the objective presented in Section 2.4 to combine only the assignment features linearly. More specifically, we replace the costs $\mathbf{c}$ obtained from a random forest classifier that we used so far with dot products to obtain

$$E(\mathbf{x}, \mathbf{y}) = \langle \mathbf{c}, \mathbf{y} \rangle \tag{4.13}$$

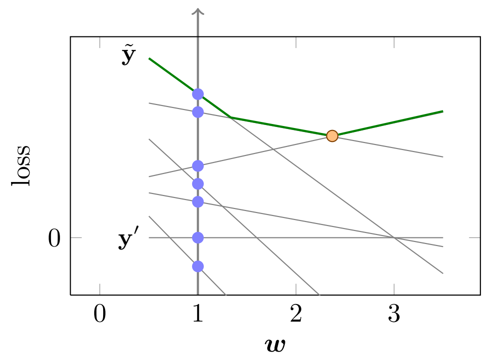
with

$$c_i = \langle \boldsymbol{w}, \phi_i(\mathbf{x}) \rangle, \tag{4.14}$$

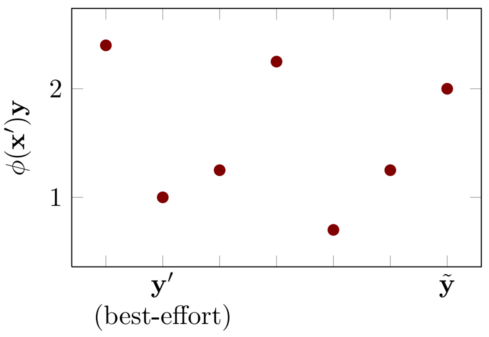
where $\phi_i(\mathbf{x})$ is the feature vector for assignment y_i. The feature vector is unified across the three different types of binary assignment variables (continuation, branch, and end)



(a) loss by $\mathbf{y}$ for $\boldsymbol{w} = 1$



(b) loss by $\boldsymbol{w}$ for each $\mathbf{y}$



(c) features $\phi(\mathbf{x}')\mathbf{y}$ for each $\mathbf{y}$

Figure 4.1: Visualization of the margin rescaling loss function $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ with $\Gamma(\mathbf{y}', \mathbf{y}) = 1$ on a toy problem with a single real-valued feature for each $\mathbf{y} \in \mathcal{Y}$ (shown in Figure (c)). Figure (a) shows the loss for a given weight vector $\boldsymbol{w} = 1$. $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ is the maximum of the individual losses over all $\mathbf{y} \in \mathcal{Y}$ (green line). The individual losses are composed of an energy difference $\delta E(\mathbf{y}', \mathbf{y}) = E(\mathbf{x}', \mathbf{y}') - E(\mathbf{x}', \mathbf{y})$ to the ground truth (shown as black dots) and the cost for misclassification $\Delta(\mathbf{y}', \mathbf{y})$ (shown as blue vertical lines). The goal of the learning procedure is to find feature weights $\boldsymbol{w}$ that minimize $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$, *i.e.*, the loss of the currently worst response $\tilde{\mathbf{y}}$. For that, responses with high costs have to be pushed down by increasing their energy and thus decreasing the difference $\delta E(\mathbf{y}', \mathbf{y})$, regardless whether their energy is already higher then the ground truth energy. The dependency between $\boldsymbol{w}$ and the energy difference $\delta E(\mathbf{y}', \mathbf{y})$ is linear, with the coefficients being the feature differences between $\mathbf{y}'$ and $\mathbf{y}$ (see Eq. (4.9)). Thus, $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ is piecewise linear, as shown in Figure (b). Although $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ is also convex, minimizing it is difficult because of the large number of possible responses.

Algorithm 1: Bundle method for structured risk minimization.

```
Input: regularizer weight $\lambda$, Tolerance of approximation, training sample $(\mathbf{x}', \mathbf{y}')$
Output: $\boldsymbol{w}^*$ minimizing structured risk $J(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$
$t = 0$;
/* initial weights                                              */
$\boldsymbol{w}_0 = \mathbf{0}$;
/* initial bundle set                                           */
$\mathcal{B}_0 = \emptyset$;
/* upper bound of minimum                                       */
$u = \inf$;
/* approximation gap                                            */
$\epsilon = \inf$
/* iteratively refine lower bound                               */
while $\epsilon \geq$ Tolerance do
    $t = t + 1$;
    /* update bundle set                                        */
    $(\mathbf{a}_t, b_t) \leftarrow$ FindMaxHyperplane($\boldsymbol{w}_{t-1}$,$\mathbf{x}'$,$\mathbf{y}'$);
    $\mathcal{B}_t = \mathcal{B}_{t-1} \cup \{(\mathbf{a}_t, b_t)\}$;
    $J_t = \lambda|\boldsymbol{w}|^2 + \langle \mathbf{a}_t, \boldsymbol{w}_{t-1} \rangle + b_t$;
    /* find minimum position and value of lower bound           */
    $(\boldsymbol{w}_t, \mathcal{J}_t) \leftarrow$ MinimizeBundleApproximation($\lambda$, $\mathcal{B}_t$);
    /* compute gap                                              */
    $u = \min(u, J_t)$;
    $\epsilon = u - \mathcal{J}_t$;
return $\boldsymbol{w}_t$
```

Algorithm 2: FindMaxHyperplane

```
Input: current weights $\boldsymbol{w}$, training sample $(\mathbf{x}', \mathbf{y}')$
Output: gradient and value $(\mathbf{g}, v)$ of $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ at $\boldsymbol{w}$
/* $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}') = \max_y \delta E(\mathbf{y}', \mathbf{y}) + \Delta(\mathbf{y}', \mathbf{y})$ */
/*                 $= \max_y \langle \boldsymbol{w}\phi(\mathbf{x}'), \mathbf{y}' \rangle - \langle \boldsymbol{w}\phi(\mathbf{x}'), \mathbf{y} \rangle + \langle \Delta_{\mathbf{l}}, \mathbf{y} \rangle + \Delta_c$ */
$\mathbf{f} = \boldsymbol{w}\phi(\mathbf{x})$;
/*                 $= \max_y \langle \mathbf{f}, \mathbf{y}' \rangle - \langle \mathbf{f}, \mathbf{y} \rangle + \langle \Delta_{\mathbf{l}}, \mathbf{y} \rangle + \Delta_c$ */
$d = \langle \mathbf{f}, \mathbf{y}' \rangle + \Delta_c$;
/*                 $= \max_y \ d - \langle \mathbf{f}, \mathbf{y} \rangle + \langle \Delta_{\mathbf{l}}, \mathbf{y} \rangle$ */
$\mathbf{l} = \Delta_{\mathbf{l}} - \mathbf{f}$;
/*                 $= \max_y \ d + \langle \mathbf{l}, \mathbf{y} \rangle$ */
/* optimize with an ILP solver */
$\mathbf{y}^* = \arg\max_{\mathbf{y}} \ \langle \mathbf{l}, \mathbf{y} \rangle$, s.t. $A\mathbf{y} \preceq \mathbf{b}$;
/* compute gradient $\frac{\partial L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')}{\partial \boldsymbol{w}}$ at $\mathbf{y}^*$ */
$\mathbf{g} = \phi(\mathbf{x}')\mathbf{y}' - \phi(\mathbf{x}')\mathbf{y}^*$;
/* compute value $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$ at $\mathbf{y}^*$ */
$v = d + \langle \mathbf{l}, \mathbf{y}^* \rangle$;
return $(\mathbf{g}, v)$;
```

Algorithm 3: MinimizeBundleApproximation

```
Input: regularizer weight $\lambda$, bundle set $\mathcal{B}$
Output: minimum $\boldsymbol{w}^*$ and minimal value $\mathcal{J}^*$ of lower bound
/* optimize with a QP solver */
$\boldsymbol{w}^* \leftarrow \arg\min_{(\boldsymbol{w}, \xi)} \ \lambda|\boldsymbol{w}|^2 + \xi$, s.t. $\langle \boldsymbol{w}, \mathbf{a}_i \rangle + b_i \leq \xi \ \ \forall (\mathbf{a}_i, b_i) \in \mathcal{B}$;
$\mathcal{J}^* = \lambda|\boldsymbol{w}|^2 + \xi$;
return $(\boldsymbol{w}^*, \mathcal{J}^*)$
```

by concatenating the respective features. Features that are not available for a certain assignment type (there is, for instance, no *overlap* for end assignments) are just set to zero in the unified feature vectors of this assignment type.

4.4.2 Training Sample

The structured learning framework requires us to provide a training sample $(\mathbf{x}', \mathbf{y}')$ with $\mathbf{y}' \in \mathcal{Y}$. In addition to the difficulties in obtaining unambiguous human generated ground truth for the neuron reconstruction problem in the first place, the provision of $\mathbf{y}'$ is not trivial: We have to find a member of $\mathcal{Y}$, *i.e.*, the set of all possible reconstructions with the found 2D neuron candidates, that is as close as possible to a human annotated ground truth. First, we have to note that the extracted 2D neuron candidates can be imperfect and thus there might not be a $\mathbf{y} \in \mathcal{Y}$ that corresponds to the human annotated ground truth. Second, we have to find a sensible criterion by which we measure the closeness of a reconstruction to the ground truth. It is debatable[5] what exactly this measure should be. In conclusion, we have to accept that the training sample $\mathbf{y}'$ will only represent a best-effort reconstruction and not the ground truth.

We found the $\mathbf{y}'$ with the maximal spatial overlap to the human annotated ground truth to best capture our intuition of a best-effort solution. Since the overlap criterion can be decomposed into local contributions for each assignment, we find $\mathbf{y}'$ by solving a reconstruction problem, where the assignment costs are replaced with the overlap to the human annotated ground truth.

4.4.3 Cost Function

A proper choice of $\Delta(\mathbf{y}', \mathbf{y})$ is crucial for the success of the structured learning method. Ideally, we would use the error measure that we use to evaluate the results of our automatic reconstruction as $\Delta(\mathbf{y}', \mathbf{y})$. However, we have to make sure that the maximization in Eq. (4.6) is still tractable.

As a compromise between tractability and specificity of the cost function, we chose to use the Hamming distance, *i.e.*, the L1-norm of the difference between the binary vectors $\mathbf{y}$ and $\mathbf{y}'$:

$$\begin{aligned}
\Delta(\mathbf{y}, \mathbf{y}') &= \sum_i |y_i - y_i'| && (4.15)\\
&= \sum_{i:y_i'=1} (1 - y_i) + \sum_{i:y_i'=0} y_i && (4.16)\\
&= |\mathbf{y}'|^2 + \sum_{i:y_i'=1} -y_i + \sum_{i:y_i'=0} y_i && (4.17)\\
&= |\mathbf{y}'|^2 + \langle \mathbf{l}, \mathbf{y} \rangle && (4.18)
\end{aligned}$$

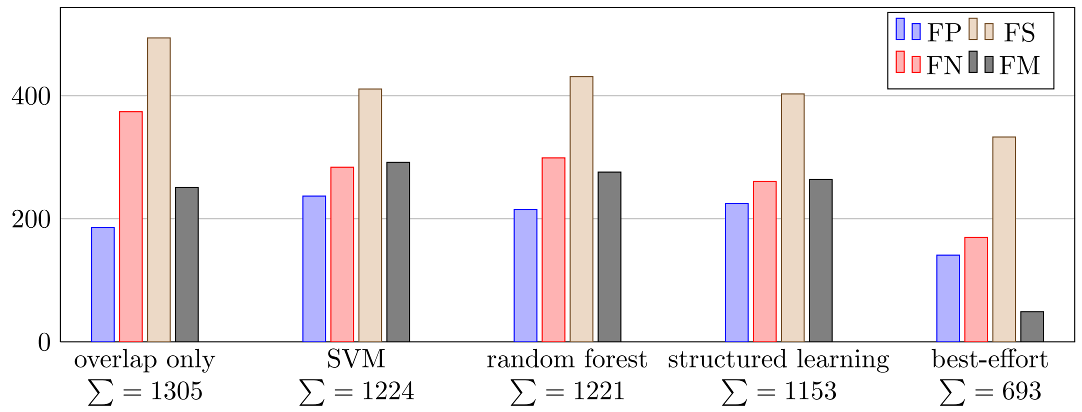
with $\mathbf{l}_i = 1 - 2y_i'$. The fact that the Hamming distance is linear in $\mathbf{y}$ allows us to merge it into the energy difference term in the loss $L(\boldsymbol{w}, \mathbf{x}', \mathbf{y}')$. Thus, the objective and structure of the maximization in Eq. (4.6) is very similar to the inference problem presented in Section 2.4, from which we know that it is tractable in practice.

4.5 Results[6]

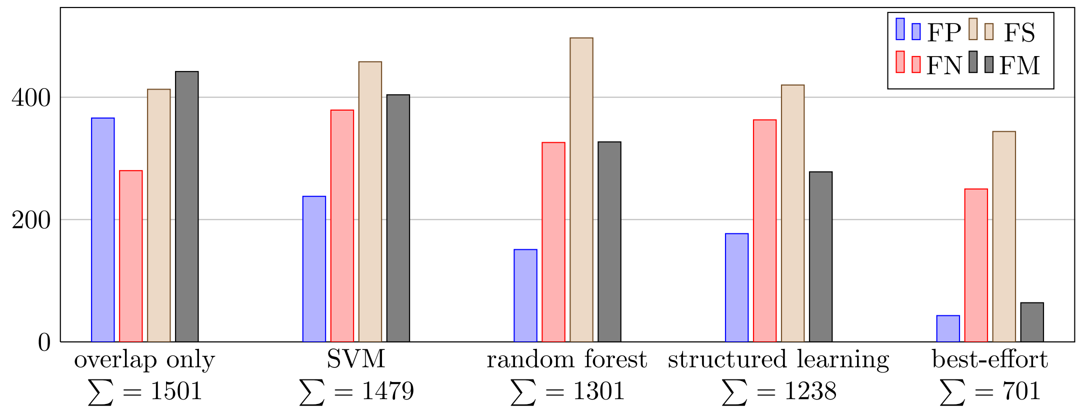
In Figure 4.3 we show how the errors change for different regularizer values for the structured learning experiments.

[5] In fact, we do debate it in Chapter 6.

[6] The SVM training for the models presented in this section has been carried out by Jonas Klein as part of his master thesis at the Institute of Neuroinformatics, UZH/ETH Zürich.



(a) Component trees.



(b) Samples from CRF.

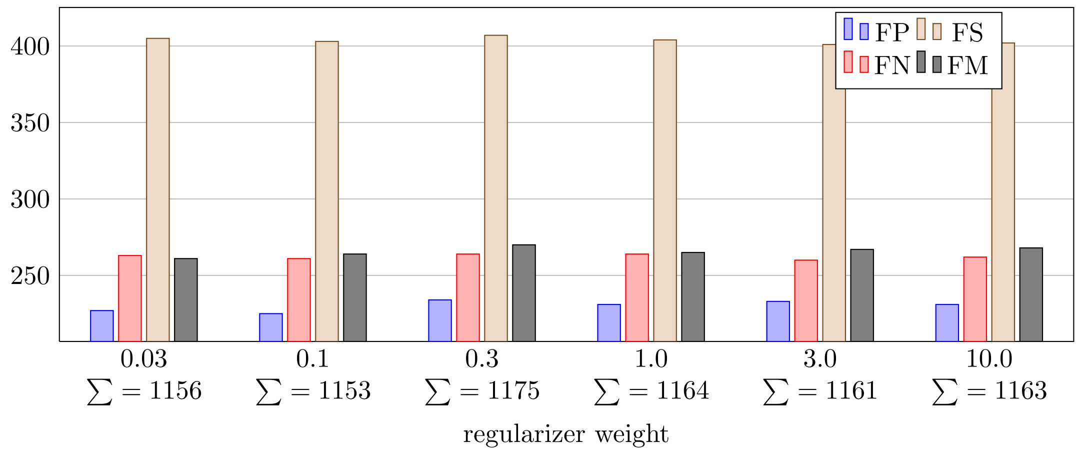
Figure 4.2: Comparison of different objective training methods for assignment costs on 10 sections of the *Drosophila melanogaster* dataset. The best results are obtained by using a linear combination of several features in the objective, with weights trained via structured learning to minimize the Hamming distance to the training sample. The bars labeled "best-effort" show the minimal numbers of errors possible with the given set of assignment candidates.

We compared the results of structured learning as described in the previous section against a random forest (RF) classifier [11], a support vector machine (SVM) [20], and a baseline model that maximizes the overlap of linked 2D neuron candidates. For that, we used two stacks of ssTEM images. The first stack consists of 20 EM images of size 1024×1024 from *Drosophila melanogaster* larva neuropil with $4.6nm$ xy-resolution and 50nm section thickness [31], for which we have human annotated ground truth. For this stack, all methods have been trained on the first five sections, validated on the second five sections, and were evaluated on the last 10 sections. The second stack consists of 100 EM images of size 1024×1024 from with $6nm$ xy-resolution and $30nm$ section thickness[7] from mouse cortex. For this stack, we trained on the first ten images, validated parameters on the second ten images, and evaluated the methods on the remaining 80 images.

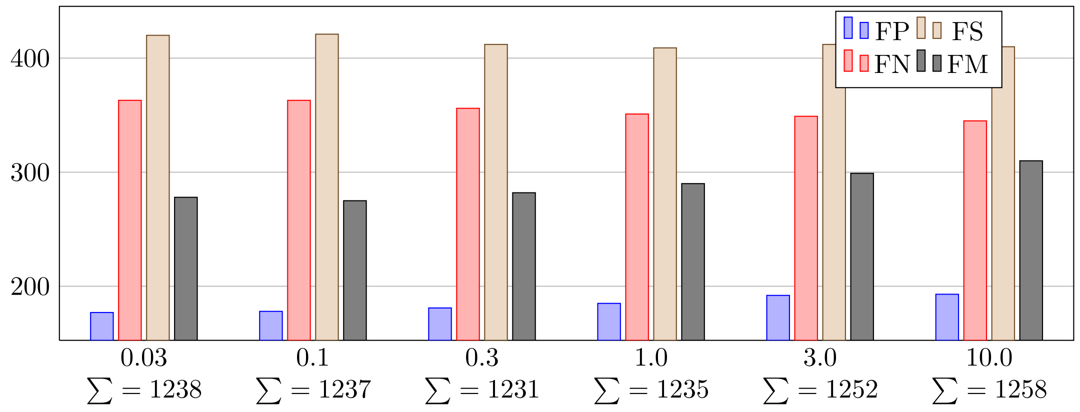
4.5.1 2D Neuron Candidate Sets

A separate stack of 20 sections was used to train a CNN to identify membranes [18] in the *Drosophila melanogaster* stack. For the mouse cortex stack, we used the available

[7]SNEMI3D challenge dataset, available at http://brainiac2.mit.edu/SNEMI3D.



(a) Component trees.



(b) Samples from CRF.

Figure 4.3: The effect of different regularizer weights in the structured learning experiments on the final errors. For both 2D neuron candidate sets, the result is very robust to changes of the regularizer weight, leaving basically no free parameter in the structured learning setup.

membrane predictions which have been produced by the same CNN architecture. From these predictions, we extracted 2D neuron candidates from the component trees of the median-filtered predictions. The component tree candidates have been generated as described in Section 2.2, but without spatial smoothness, *i.e.*, by just performing a sequence of thresholds. For the *Drosophila melanogaster* stack we additionally generated candidates by sampling a CRF as described in Chapter 3. For each of the candidate sets, we found a best-effort reconstruction $\mathbf{y}'$ as described in Section 4.4.2, that was used for all training methods.

4.5.2 Training, Validation, and Evaluation

The RF classifier has been trained on the feature vectors of positive and negative assignment samples of the best-effort reconstruction $\mathbf{y}'$ for the first five sections. Each assignment that is part of $\mathbf{y}'$ was taken as a positive sample, the remaining assignments as negative samples. After training, we use the negative logarithm of the RF prediction (which is the probability of being a correct assignment) as the cost for an assignment. The SVM classifier was trained on the same feature vectors. For the reconstruction objective, we use the signed distance to the learnt margin as the cost for an assignment. The structured learning was carried

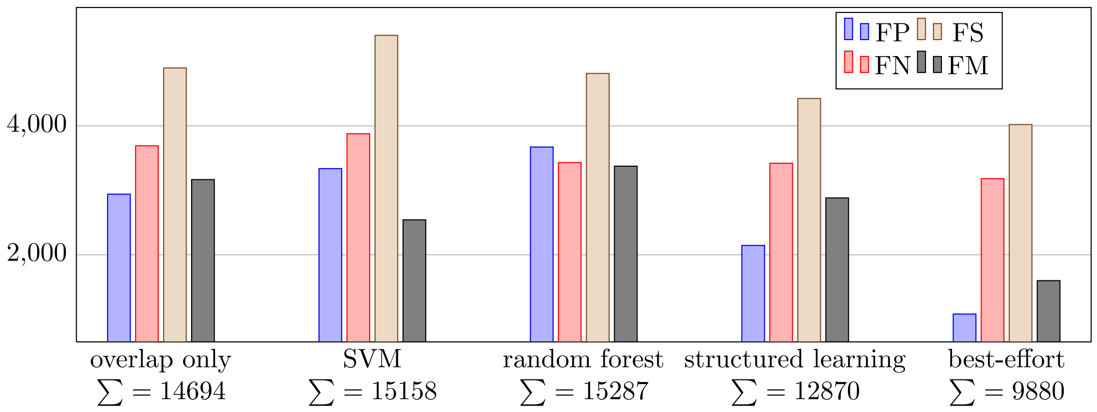


Figure 4.4: Comparison of different objective training methods for assignment costs on 80 sections of the mouse cortex dataset. The best results are obtained by using a linear combination of several features in the objective, with weights trained via structured learning to minimize the Hamming distance to the training sample. The bars labeled "best-effort" show the minimal numbers of errors possible with the given set of assignment candidates.

out to minimize the Hamming distance of the reconstruction to the best-effort solution, as described in Section 4.4. The baseline model (overlap only) was not trained at all.

Since the RF and SVM classifiers are trained on only positive and negative assignment samples, they do not have the scope they need to learn sensible priors for the different assignment types (continuation, branch, and end). In particular, the costs obtained from the RF classifier are always positive, thus giving no incentive to select any assignment at all. To make comparison fair, we added those priors explicitly as constant costs for each assignment type to the objective for the RF and SVM experiments. We found their values via grid search on the respective validation datasets by minimizing the sum of errors made. For the structured learning experiments, we validated the value of the regularizer weight in the same way.

Both for the validation and evaluation, we use errors in terms of the anisotropic edit distance, which counts false positives (FP, spuriously found 2D neuron slices), false negatives (FN, missed 2D neuron slices), false splits (FS, missed links between neuron slices of adjacent sections), and false merges (FM, spurious links between neuron slices of adjacent sections). In our experiments, we considered 2D neuron slices with at least 50% overlap with the ground truth as correctly found. Details on how to compute this error measure are given in Section 6.2.5 on page 71.

In Figure 4.2 we present the errors on the *Drosophila melanogaster* stack after training of each method. Additionally, we provide results for the baseline model that selects assignments by just maximizing the overlap of linked 2D neuron candidates across adjacent sections, *i.e.*, without any training. The reconstruction results of the best methods for component trees and CRF are shown in Appendix C.1 and Appendix C.2, respectively. Finally, Figure 4.4 shows the evaluation on the mouse cortex stack.

4.6 Discussion

Amongst the investigated training methods, structured learning provides the best results for both datasets (*Drosophila melanogaster* and mouse cortex) and the used 2D candidate generation methods. On the *Drosophila melanogaster* dataset, the results are closer to the best-effort solution by 12.9% (candidates from component trees) and 11% (candidates from CRF), compared to the RF classifier, which is the second best training method. On the mouse dataset, the results are even 62.1% closer to the best-effort solution, compared to the overlap-only baseline, which is the second best in this dataset. This demonstrates that even a linear combination of the features can perform very well if trained on a sensible objective.

The reason why the Hamming distance is such a sensible training objective might lie in the diversity of the 2D neuron candidates. Consider, for instance, the candidates extracted via component trees, following the method presented in Section 2.2. Every pair of conflicting assignments generated from these 2D neuron candidates proposes a topologically different reconstruction. In other words, there are no two assignment candidates that only differ slightly in the membrane locations. It follows that any deviation from the best-effort reconstruction results in a topologically different reconstruction. The Hamming distance happens to count the number of topological deviations and is thus a sensible objective to minimize.

A very appealing property of the structured learning formulation is its robustness to changes of the regularizer weight. Thus, the structured learning formulation comes with basically no free parameter. In practice, that means we can omit the tedious grid search on hyper parameters while still getting superior performance compared to the other methods.

Chapter 5

Distributed Inference

5.1 Introduction

A key property of our model for neuron reconstruction is that a globally optimal solution for a whole volume is found at once. Although we observe that the optimization problem can be solved efficiently in linear time in praxis, scaling up to larger volumes is not trivial. An adult fly brain, for example, will produce at least 100 terapixels of images of neural tissue. From our experiments we know that even our smallest models need about 100 kilobyte per megapixel to describe the optimization problem as an integer linear program (ILP)[1].For the whole fly brain, this means that just to represent the ILP we need at least 10 terabytes of memory – not counting the ILP data structures needed to actually solve the optimization problem. On current hardware, this is prohibitively large.

At the same time, we observe that the assignment variables in our model have a limited region of influence. A decision made in one part of the volume is very unlikely to change the decisions made in another part. This is in line with observations from the manual reconstruction of neurons: In most cases, it is sufficient to consider a small and constant-sized region around a reconstruction site to resolve ambiguities. There are only a few examples where the knowledge of the shape of a complete neuron helps a human annotator to resolve ambiguities, but these far-distance relations are not part of our model, yet. Intuitively, it seems very likely that our inference problem can be decomposed and solved sequentially.

Motivated by this intuition, we investigate in this chapter[2] how our model for neuron reconstruction can be solved in a distributed way. In particular, we propose a decomposition strategy into subproblems that iteratively refines the partial solutions of the subproblems to guarantee optimality of the global problem.

5.1.1 Overview

We use the method of dual decomposition [92] to obtain constant size subproblems of the original large problem. For that, we define overlapping regions on a factor graph representation of our model. Each of the regions represents one subproblem and can be solved independently of other subproblems. In general, the optimal solutions of neighboring subproblems (*i.e.*, subproblems from overlapping regions) can disagree on the values of shared variables. By passing messages between neighboring subproblems, the subproblems are encouraged to agree on the values of shared variables. If all subproblems agree on all shared variables, the globally optimal solution was found. However, in the standard dual decom-

[1]Our smallest model produces ILP matrices with about 6000 non-zeros per megapixel of image data. Assuming that every non-zero is represented by a double (8 bytes) and two unsigned integers (each 4 bytes), we obtain a memory consumption for the description of the ILP of about 100 kilobyte per megapixel.

[2]The findings presented in this chapter are results from joint work with Tobias Pietzsch and Florian Jug from the MPI-CBG in Dresden.

position formulation these messages are not guaranteed to achieve agreement. We present a strategy to modify the subproblems in a way that guarantees that every pair of neighboring subproblems will always find an agreement. We show that if the subproblem neighborhood graph is tree-shaped, our strategy will also find the global optimal solution to the original problem.

In Section 5.2 we show how a general factor graph model can be decomposed into subproblems using the framework of dual decomposition. We show how an optimal set of messages can be found to encourage agreement between neighboring subproblems and discuss the situation in which this strategy fails due to the duality gap. In Section 5.3 we present our strategy to modify pairs of subproblems to close the duality gap between them and show that this strategy achieves the globally optimal solution for tree-shaped neighborhood graphs. We evaluate our strategy experimentally on in Section 5.4 and discuss its usefulness for practical implementations in Section 5.5.

5.1.2 Related Work

Dual decomposition as a method to distribute approximate inference has extensively been studied, for an overview see Sontag et al. [92]. Dual decomposition is a special case of Lagrangian relaxation [30], where consistency between subproblems is first ensured with the introduction of constraints that are then relaxed into Lagrangian multipliers. As such, dual decomposition is mainly used to achieve approximate maximum a-posteriori (MAP) solutions for large and combinatorially difficult Markov random fields.

Consequently, many publications address the problem of tightening these relaxations. Recent advances have been made by decomposing the original problems into subproblems that describe spanning trees on the original problem [56, 107]. It was shown by Komodakis et al. that they are instances of a more general dual decomposition framework on Markov random fields that provably solves the dual relaxation corresponding to the chosen decomposition [60]. In these works, the coefficients of the Lagrangian multipliers can be seen as messages that are sent between neighboring subproblems. These messages change the objective values of the subproblems, such that the net effect of all changes to the global optimization problem is zero. By optimizing the dual formulation of the decomposed problem, an optimal set of messages can be found to encourage neighboring subproblems to agree on shared variable values. However, if the decomposition contains cycles, there might not exist a set of messages that leads to an agreement. In these cases, the conflicting variable assignments are usually repaired using a heuristic to find an approximate solution to the MAP problem [92].

Despite its main use to obtain good approximate MAP solutions, exiting progress has been made in using the dual decomposition framework for optimal MAP inference. So-called cluster-pursuit algorithms iteratively add higher order consistency constraints to the problem decomposition to tighten the dual relaxation and thus to suppress the influence of frustrated cycles [4, 59, 91]. Even though the number of additional constraints needed to achieve a tight relaxation is in general exponential in the problem size, empirically it could be shown to be tractable for a wide range of problems.

A remaining bottle-neck of dual decomposition approaches is the slow convergence rate of the optimization in the dual to find the optimal set of messages. This issue is addressed by smoothing the dual formulation [44] or by using bundle methods [48] that are well suited to optimize the non-smooth dual objective. Recently, a multigrid-like approach was shown to improve the performance of this optimization for computer vision problems [58].

In the field of automatic neuron reconstruction, problem decomposition has hardly been a topic so far. A noteworthy exception is the neuron reconstruction pipeline by Kaynig et al. [50], which decomposes large volumes of neural tissue into overlapping blocks. These blocks are solved independently and the found neuron objects are merged or split in pairs with neighboring blocks depending on the overlap of the objects in voxels.

5.2 Dual Decomposition

In this section we show how the framework of dual decomposition can be used to decompose a large inference problem into smaller subproblems. For that, we assume that the inference problem is given in form of a graphical model that defines energies on a set of variables with discrete domains. The optimization problem consists of finding variable values that minimize this energy. This very general formulation allows us in particular to express optimization problems on linear constraint binary models, as we obtain them from our model for neuron reconstruction presented in Chapter 2.

5.2.1 Problem Definition

We start by defining the original and possibly large inference problem in terms of a graphical model, where we distinguish between variable and factor nodes. This so-called factor graph representation corresponds to a factorization of the energy that we wish to minimize [106].

Definition 5.2.1 (Factor Graph) Let a graphical model be given as a factor graph $G = (F, V, E, \theta, D)$. F and V are strictly ordered sets of m factor nodes (or just factors) and n variable nodes (or just variables), respectively:

$$F = \{f_i : 1 \leq i \leq m\} \text{ and } V = \{v_i : 1 \leq i \leq n\}.$$

Factors and variables are connected with edges $E \subseteq F \times V$ to form a bipartite graph. To each factor, we associate a function

$$\theta = \{\theta_f : f \in F\},$$

and to each variable a discrete and finite domain

$$D = \{D_v : v \in V\}.$$

We write V_f for the strictly ordered set of all variables that are connected to factor f and F_v for the strictly ordered set of all factors connected to variable v. We denote an assignment of variables to elements of their discrete domains as

$$\mathbf{x} = \{x_v : v \in V, x_v \in D_v\}.$$

We write $\mathbf{x}_f = \{x_v \in \mathbf{x} : v \in V_f\}$ for the partial assignment of all variables connected to factor f. Consequently, we write $\theta_f(\mathbf{x}_f)$ for the function value of θ_f under the assignment $\mathbf{x}_f$. The energy $G(\mathbf{x})$ associated to a graphical model G under an assignment $\mathbf{x}$ is the sum over all factor functions under their partial assignments:

$$G(\mathbf{x}) = \sum_{f \in F} \theta_f(\mathbf{x}_f). \tag{5.1}$$

The global optimization problem consists now of finding an assignment $\mathbf{x}$ that minimizes the energy as defined by the factor graph.

Definition 5.2.2 (P1) The optimization problem *P1* is defined as

$$\min_{\mathbf{x}} G(\mathbf{x}). \tag{5.2}$$

We denote the minimal value of P1 as e^* and the minimizer by $\mathbf{x}^*$.

5.2.2 Decomposition into Subproblems

There are two difficulties in optimizing P1. First, P1 is in general NP-hard. As such, there might not exist an efficient optimization scheme at all. For many "real world" problems in computer vision, and in particular for our model of neuron reconstruction, this is in practice not a problem [49]. Second, the graphical model corresponding to P1 might be too large to be dealt with at once. This case is the main motivation for the decomposition strategy that we present in the following.

The general idea of the decomposition is to partition the original problem into subproblems that are as large as possible, *i.e.*, they can still be represented and solved efficiently on available hardware. We define these subproblems as subsets of factors of the original factor graph. Variables that are used in factors of different subproblems are said to be shared. These variables play an important role in the decomposition: As we will show soon, the subproblems have to agree on the optimal values of these variables in order to achieve global optimality – without having the scope to look at the whole problem at once. We show in the following how the subproblems can be encouraged to agree by following the dual decomposition principle.

Definition 5.2.3 (Decomposition) A decomposition of G is a tuple (R, W), where R is a strictly ordered set of l possibly overlapping regions

$$R = \{R_i : 1 \leq i \leq l, R_i \subseteq F\}$$

and W is a set of factor weights

$$W = \{w_f^r \in [0, 1] : r \in R, f \in r\}.$$

We denote by $R_f = \{r \in R : f \in r\}$ the subset of all regions containing a factor f and by $R_v = \bigcup_{f \in F_v} R_f$ the subset of all regions whose factors are involving v. The factor weights w_f^r for every factor f and all regions $r \in R_f$ that contain this factor have to sum up to one, *i.e.*,

$$\sum_{r \in R_f} w_f^r = 1 \;\; \forall f \in F. \tag{5.3}$$

We say that a decomposition *spans* the graphical model, if every factor appears in at least one region, *i.e.*, R_f is non-empty for all $f \in F$. Note, that a region r can contain factors with zero weight, *i.e.*, $\{f \in F : w_f^r > 0\} \subseteq r$ for all $r \in R$.

The introduced factor weights for each region are important to guarantee that the corresponding set of subproblems partitions the energy defined on the original graphical model. We achieve this by replacing the functions associated to the factors in each region with scaled versions.

Definition 5.2.4 (Subproblem) Given a decomposition (R, W), the subproblem corresponding to a region $r \in R$ is a factor graph $G^r = (V^r, r, E^r, \theta^r, D^r)$ where

$$V^r = \bigcup_{f \in r} V_f$$

and E^r and D^r are corresponding subsets of the original E and D for V^r. The factor functions θ^r are scaled copies of the original functions, *i.e.*,

$$\theta^r = \{\theta_f^r = w_f^r \theta_f : f \in r\}.$$

The optimization problem P1 on the original factor graph can now be rewritten in terms of the subproblems of a decomposition (R, W):

Definition 5.2.5 (P2) The subproblem optimization problem *P2* is defined as

$$\min_{\mathbf{x}} \sum_{r \in R} \sum_{f \in r} \theta_f^r(\mathbf{x}_f). \tag{5.4}$$

So far, the decomposition was only cosmetic and did not change the original optimization problem.

Lemma 5.2.1 P1 is equal to P2.

The proof can be found in Appendix B.1. In order to decouple the subproblems, we introduce local variable assignments $\mathbf{x}^r = \{x_v^r : v \in R_v\}$ for every region $r \in R$. We write $\mathbf{x}^R = \{\mathbf{x}^r : r \in R\}$ to denote the vector of all local assignments. With additional constraints we enforce consistency between the assignments for shared variables across regions:

Definition 5.2.6 (P3) The constraint optimization problem *P3* is defined as

$$\min_{\mathbf{x}^R} \sum_{r \in R} \sum_{f \in r} \theta_f^r(\mathbf{x}_f^r) \tag{5.5}$$

$$\text{s.t.} \quad x_v^r = x_v^{r'} \qquad \forall v \in V \; \forall r, r' \in R_v : r < r'. \tag{5.6}$$

Lemma 5.2.2 P2 is equal to P3.

The proof can be found in Appendix B.2. Coming back to our initial assumption that each subproblem can efficiently be solved on its own, we realize that P3 could be easily solved as well if we could drop the consistency constraints across regions. Without these constraints, the optimization in P3 would decompose into local optimizations within each subproblem. We eliminate those constraints by using the method of Lagrangian relaxation [30]. For that, we introduce Lagrangian multipliers

$$\boldsymbol{\lambda} = \{\lambda_{v,x}^{r,r'} : v \in V, x \in D_v, r, r' \in R : r < r'\} \tag{5.7}$$

for each consistency constraint to obtain the new objective

$$L(\boldsymbol{\lambda}, \mathbf{x}^R) = \sum_{r \in R} \sum_{f \in r} \theta_f^r(\mathbf{x}_f^r) + \sum_{v \in V} \sum_{\substack{r, r' \in R_v \\ r < r'}} \sum_{x \in D_v} \lambda_{v,x}^{r,r'} \left(\mathbb{1}_{\{x_i^r = x\}} - \mathbb{1}_{\{x_i^{r'} = x\}} \right), \tag{5.8}$$

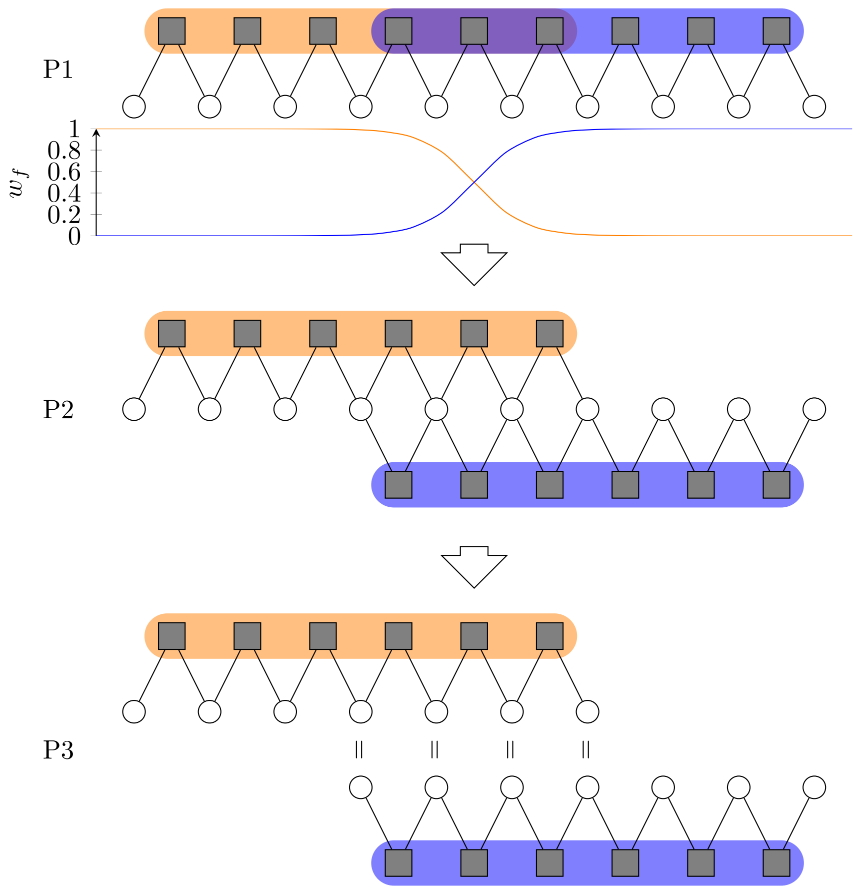


Figure 5.1: A simple example to illustrate the proposed decomposition of a factor graph (variables are circles, factors are squares) into two regions (orange and blue). P1 is the original problem that we wish to decompose. By duplicating and weighting of the factors according to w_f for each region, we obtain P2, which has the same optimal solution as P1. Finally, P3 is obtained by additionally duplicating all variables that are shared across regions and enforcing consistency between them.

where we rewrote the consistency constraints as linear equalities on the values[3] of their respective x_i. These equalities are non-zero whenever two regions disagree on the value of a joint variable. This implies that if the constraints in Eq. (5.6) hold, the minimum of $L(\boldsymbol{\lambda}, \mathbf{x}^R)$ is achieved at the solution to P3, regardless of the $\boldsymbol{\lambda}$.

Interestingly, for a given $\boldsymbol{\lambda}$, the $\mathbf{x}^R$ minimizing $L(\boldsymbol{\lambda}, \mathbf{x}^R)$ can be found by local optimizations. We denote this optimization problem by

$$L(\boldsymbol{\lambda}) = \min_{\mathbf{x}^R} L(\boldsymbol{\lambda}, \mathbf{x}^R), \tag{5.9}$$

which is the dual formulation corresponding of the given decomposition. This dual is a lower bound on the minimal energy e^* of the original problem.

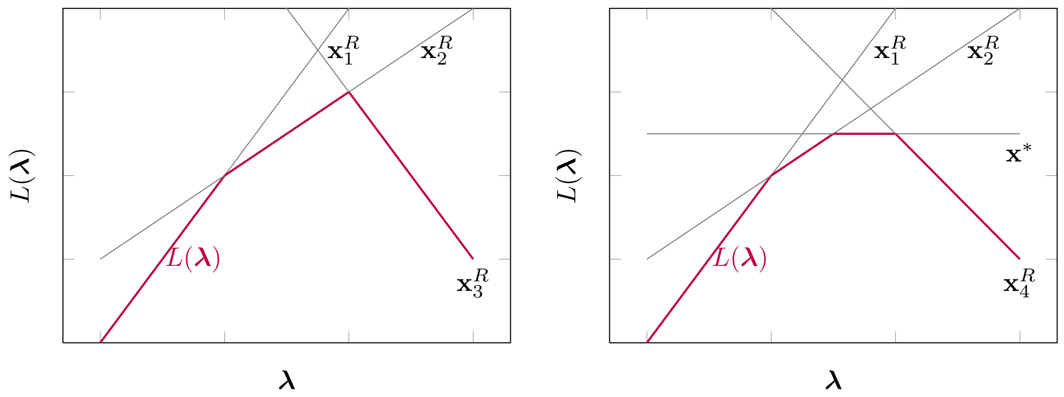
Theorem 5.2.3 For any given $\boldsymbol{\lambda}$, $L(\boldsymbol{\lambda})$ is a lower bound on the energy of the solution of P1. Furthermore, if the $\mathbf{x}^R$ at the optimum are consistent according to Eq. (5.6), they represent the optimal solution to P1.

The proof can be found in Appendix B.3. Since we are interested in finding the minimal energy solution to P1 we wish to maximise $L(\boldsymbol{\lambda})$, knowing that if $L(\boldsymbol{\lambda})$ is able to provide us the solution at all, it can only be the case at the maximum. To see that this can be achieved by local operations, we rewrite $L(\boldsymbol{\lambda})$ as follows, where we write $|v|_r$ to denote the degree of a variable in region r, *i.e.*, the number of factors involving it:

$$\begin{aligned}
L(\boldsymbol{\lambda}) &= \min_{\mathbf{x}^R} L(\boldsymbol{\lambda}, \mathbf{x}^R) && (5.10)\\
&= \min_{\mathbf{x}^R} \sum_{r\in R}\sum_{f\in r} \theta_f^r(\mathbf{x}_f^r) + \sum_{v\in V}\sum_{\substack{r,r'\in R_v\\ r<r'}}\sum_{x\in D_v} \lambda_{v,x}^{r,r'}\left(\mathbb{1}_{\{x_i^r=x\}} - \mathbb{1}_{\{x_i^{r'}=x\}}\right) && (5.11)\\
&= \min_{\mathbf{x}^R} \sum_{r\in R}\sum_{f\in r} \theta_f^r(\mathbf{x}_f^r) + \sum_{v\in V}\sum_{\substack{r,r'\in R_v\\ r<r'}}\left(\sum_{x\in D_v} \lambda_{v,x}^{r,r'}\mathbb{1}_{\{x_i^r=x\}} - \sum_{x\in D_v} \lambda_{v,x}^{r,r'}\mathbb{1}_{\{x_i^{r'}=x\}}\right) && (5.12)\\
&= \min_{\mathbf{x}^R} \sum_{r\in R}\sum_{f\in r} \theta_f^r(\mathbf{x}_f^r) + \sum_{v\in V}\sum_{\substack{r,r'\in R_v\\ r<r'}}\left(\lambda_{v,x_v^r}^{r,r'} - \lambda_{v,x_v^{r'}}^{r,r'}\right) && (5.13)\\
&= \min_{\mathbf{x}^R} \sum_{r\in R}\sum_{f\in r} \underbrace{\left(\theta_f^r(\mathbf{x}_f^r) + \sum_{v\in V_f} |v|_r^{-1}\left(\sum_{\substack{r'\in R_v\\ r'>r}} \lambda_{v,x_v^r}^{r,r'} - \sum_{\substack{r'\in R_v\\ r'<r}} \lambda_{v,x_v^r}^{r,r'}\right)\right)}_{\overline{\theta}_f^r(\mathbf{x}_f^r)} && (5.14)\\
&= \sum_{r\in R}\min_{\mathbf{x}^r}\sum_{f\in r}\overline{\theta}_f^r(\mathbf{x}_f^r). && (5.15)
\end{aligned}$$

This reformulation shows that we can think of the $\boldsymbol{\lambda}$ as changes to the factors of P3. These changes are incorporated in a way that doesn't change the overall energy for a given assignment $\mathbf{x}^R$.

[3]We reformulated the consistency constraints such that we have one constraint for each *value* of each variable, instead of one for each variable as we see it in Eq. (5.6). This is for technical purposes: Unless there is a good reason to prefer a conflicting assignment which happens to be close to each other (*e.g.*, one region says that x_A should be 1 and another one says it should be 2 is better then 1 vs. 3 – this is usually not the case) we want every disagreement to be treated equally.



(a) non-tight (b) tight

Figure 5.2: Illustration of the $L(\boldsymbol{\lambda})$ function (purple). $L(\boldsymbol{\lambda})$ is the minimum of a set of hyperplanes, one for each possible assignment $\mathbf{x}^R$. If the relaxation is tight (b), the hyperplane for $\mathbf{x}^*$ bounds the maximum of $L(\boldsymbol{\lambda})$.

5.2.3 Optimizing the Decomposition Relaxation

Optimizing $L(\boldsymbol{\lambda})$ can be interpreted as optimizing a relaxed version of problem P3, where we don't enforce consistency across regions explicitly.

Definition 5.2.7 (P3R) The optimization problem *P3R* is defined as

$$\max_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}). \tag{5.16}$$

We define $\boldsymbol{\lambda}^*$ to be the maximizer

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} \; L(\boldsymbol{\lambda}) \tag{5.17}$$

of P3R, and $\mathbf{x}^{R*}$ to be the assignment at the optimum:

$$\mathbf{x}^{R*} = \arg\max_{\mathbf{x}^R} \; L(\boldsymbol{\lambda}^*, \mathbf{x}^R). \tag{5.18}$$

From the definition of $L(\boldsymbol{\lambda})$ in Eq. (5.9), and also from its reformulation in Eq. (5.15), we can see that it is convex and piecewise linear. It is the minimum over a set of functions (one for each assignment $\mathbf{x}^R$), that are linear in $\boldsymbol{\lambda}$ (see Figure 5.2).

Intuitively, $L(\boldsymbol{\lambda})$ can be optimized by a gradient method. However, there are some difficulties to address: First, $L(\boldsymbol{\lambda})$ is not differentiable everywhere, but this has almost no practical importance. Second and more importantly, it is unclear how to set the step width to efficiently arrive at the optimum.

These difficulties are well tackled by bundle methods [48, 97]. These methods maintain a piecewise linear upper bound $\mathcal{L}(\boldsymbol{\lambda})$ on the objective $L(\boldsymbol{\lambda})$ by iteratively sampling values and subgradients from $L(\boldsymbol{\lambda})$. A regularized version of the upper bound is then maximized to propose the next sample position $\boldsymbol{\lambda}^t$ and tighten the gap estimate ϵ_t, *i.e.*, the difference between the largest function value seen so far and the maximum value of the bound. This gap estimate allows to quantify the closeness to the optimal solution.

5.2.4 Tightness of the Relaxation

If $L(\boldsymbol{\lambda})$ is tight, it must be bound in its maximum by the hyperplane corresponding to the assignment $\mathbf{x}^{R*}$. Otherwise, it is blocked by a set of hyperplanes with a non-zero gradient, each corresponding to an inconsistent assignment (see Figure 5.2 for an illustration).

An interesting question to ask is: under what conditions is a relaxation guaranteed to be tight? It is difficult to answer this question exactly, but a link to linear programming (LP) relaxation allows us to define at least a sufficient criterion for tight relaxations [88, 107].

Corresponding LP Problem

Every problem decomposition following the dual decomposition principle has a corresponding LP relaxation problem. For the construction of this LP relaxation, it is helpful to collapse the variables and factors that are exclusively owned by a region into a single factor. This emphasizes the fact that the subproblems of the regions are hiding the structure of their internal factor graphs: For every configuration of the shared variables of a subproblem, there is a unique minimal energy contribution, that we represent as a single factor on the shared variables. See Figure 5.3 for an illustration.

The optimization problem of the collapsed problem consists now of finding values for the shared variables that minimize the global energy. This optimization can be formulated as an integer linear program on indicator variables $\mathbb{I}$ with costs $\phi_{f,\mathbf{x}_f}$ for each configuration of variables of each factor (see Appendix A.1 for the exact procedure):

$$\begin{aligned}
\min_{\mathbb{I}} \quad & \sum_{f \in F} \sum_{\mathbf{x}_f} \phi_{f,\mathbf{x}_f} \mathbb{I}_{f,\mathbf{x}_f} & \\
\text{s.t.} \quad & \sum_{\mathbf{x}_f} \mathbb{I}_{f,\mathbf{x}_f} = 1 & \forall f \in F \\
& \sum_{\mathbf{x}_f : \mathbf{x}_f(i) = z} \mathbb{I}_{f,\mathbf{x}_f} - \sum_{\mathbf{x}_g : \mathbf{x}_g(i) = z} \mathbb{I}_{g,\mathbf{x}_g} = 0 & \forall i \in V : \ \forall f, g \in F_i : \ \forall z \in D_i \\
& \mathbb{I}_{f,\mathbf{x}_f} \in \{0, 1\} & \forall f \in F : \forall \mathbf{x}_f.
\end{aligned}$$

In this formulation, the linear constraints ensure consistency between the indicators of factors that share variables. In some sense, they ensure that the marginalization of the integral indicator variables is consistent. By relaxing the integrality condition on the indicators and thus allowing the indicators to take real values, *i.e.*,

$$\mathbb{I}_{f,\mathbf{x}_f} \in [0, 1] \quad \forall f \in F : \forall \mathbf{x}_f,$$

the LP relaxation is obtained. It is well known that the dual of the LP relaxation obtained this way is equivalent to the objective of the dual decomposition, see for example Wainwright et al. [107]. A direct consequence is that if the LP relaxation is tight, the corresponding dual decomposition relaxation is tight as well. LP relaxations are provably tight for graphical models that do not contain cycles. Brought back into the context of dual decomposition, this implies that the dual is tight, if the following (sufficient but not necessary) two conditions hold: (1) The decomposition is tree-shaped, that is, the neighborhood graph of overlapping regions does not contain cycles, and (2) the subgraphs that are shared between overlap regions do not contain cycles.

In the following, we will assume that the first condition, a tree-shaped decomposition, is fulfilled. Whether this assumption can be made depends on the concrete application. It remains to address the problem of possible cycles in the overlap between neighboring regions.

5.3 Closing the Duality Gap

Assuming that the decomposition is tree-shaped, it remains to ensure that the subgraphs that are shared between neighboring regions are cycle free. Naively, this can be done by

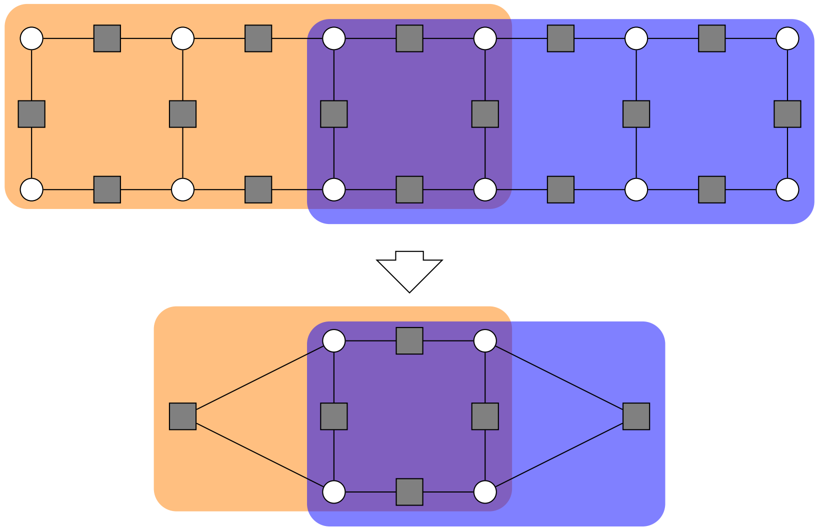


Figure 5.3: Illustration of the region collapsing to obtain the linear program relaxation that corresponds to a given dual decomposition scheme. In this example, a simple factor graph (top figure) with variables represented as circles and factors represented as squares is decomposed into two regions (orange and blue). Variables and factors that belong exclusively to a region can conceptually be collapsed into a single factor (bottom figure). Possible cycles in the internal factor graph of a subproblem are thus hidden from the structure of the decomposition and have no influence on the theoretical guarantees on the tightness of the dual decomposition. On this collapsed model, an integer linear program can be formulated to minimize the energy over all shared variables. The dual of the LP relaxation of this ILP is provably the same as the objective function of the dual decomposition. It follows that if the dual of the LP relaxation is tight, so is the dual decomposition objective.

collapsing all the variables in the overlap into a single variable with as many states as there are configurations in the overlap, similar to how we collapsed the variables that are owned exclusively by a single region. However, in contrast to the collapsing of exclusive variables, the collapsing of the overlap variables is not just a conceptual trick to obtain theoretical guarantees: For this strategy to work, we would have to maintain very large factors and tables of messages to be sent between neighboring regions. The number of configurations in an overlap region grows exponentially with the number of variables in it.

Instead, we propose to follow a strategy inspired by cluster-pursuit algorithms [4,59,91]. We iteratively solve the dual $L(\boldsymbol{\lambda})$ and investigate the result in the primal: If a consistent labeling of the graph was found, we found the globally optimal solution. Otherwise, we are left with a conflicting assignment of shared variables. We add additional binary variables that indicate those conflict assignments and introduce new messages by augmenting $\boldsymbol{\lambda}$ for the new variables. See Figure 5.4 for an illustration of one step of this strategy. By iteratively adding indicator variables for remaining conflicts, we widen the scope of the messages that can be sent between the subproblems. In the worst case, this strategy will introduce an indicator for every possible assignment of the variables in the overlap between two subproblems. This case is equivalent to collapsing all the variables in the overlap between two subproblems into a single variable and deriving the messages (*i.e.*, the Lagrangian multipliers) for each value of this collapsed variable. Although we hope that this is in practice not necessary, it shows that our strategy eventually eliminates cycles between neighboring regions and thus will find a globally optimal solution. Pseudo-code of our strategy is shown in Algorithm 4.

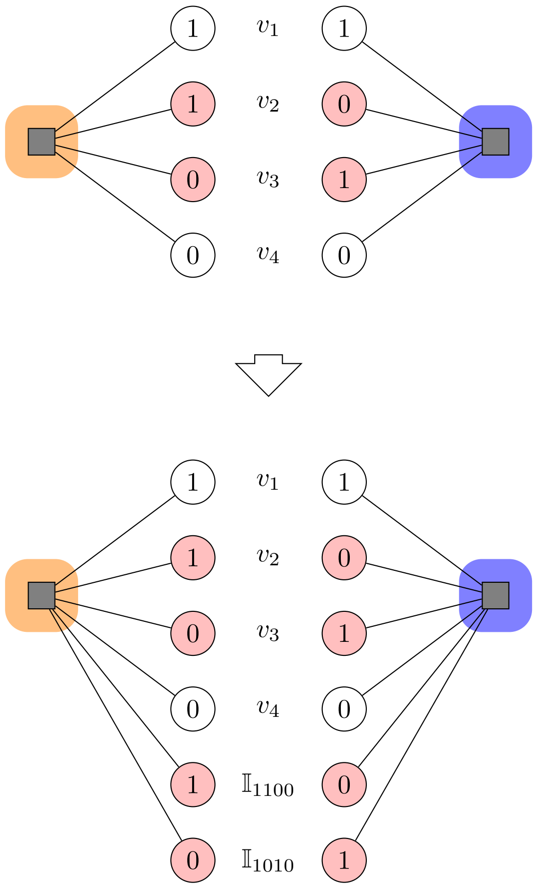


Figure 5.4: Illustration of the proposed gap closing strategy for conflicting assignments (highlighted in red). The upper figure shows two regions (represented by a single factor in orange and blue) with shared variables v_1 to v_4 and their assignments after optimizing $L(\boldsymbol{\lambda})$. Shared factors are omitted for clarity. We add indicator variables $\mathbb{I}$ for both proposed assignments of the shared variables to each region (lower figure). This way, we introduce additional messages that can be sent between the regions to negotiate the conflict assignments. The updated problem is solved again and further indicator variables are added until both regions agree on the assignment of the shared variables.

Algorithm 4: Gap closing strategy for tree-shaped decompositions.

```
Input: original problem $G$, tree-shaped decomposition $(R, W)$
Output: $\mathbf{x}^*$ minimizing $G(\mathbf{x})$
/* decompose problem into subproblems */
$\mathcal{S} = \{G^r : r \in R\}$
while $\mathbf{x}^*$ not found do
    /* solve dual */
    $\mathbf{x}^{R*}$ = OptimizeDual($\mathcal{S}$)
    if $\mathbf{x}^{R*}$ is consistent then
        /* we found the globally optimal solution */
        $\mathbf{x}^* \leftarrow \mathbf{x}^{R*}$
        return $\mathbf{x}^*$
    foreach conflict $\bar{\mathbf{x}}_p$, $\bar{\mathbf{x}}_q$ of shared variables between regions $p, q \in R$ do
        /* update $G^p$ */
        $G^p \leftarrow$ AddIndicators($G^p$, $\bar{\mathbf{x}}_p$, $\bar{\mathbf{x}}_q$)
        /* update $G^q$ */
        $G^q \leftarrow$ AddIndicators($G^q$, $\bar{\mathbf{x}}_p$, $\bar{\mathbf{x}}_q$)
```

5.3.1 Conflict Detection

We are using a proximal bundle method similar to the one proposed by Kappes et al. [48] that optimizes the dual $L(\boldsymbol{\lambda})$ to arbitrary precision to obtain λ^*. The corresponding solutions $\mathbf{x}^{R*}$ of the subproblems can directly be read from the last update step of the bundle method. From $\mathbf{x}^{R*}$ we can easily read conflicting assignments of shared variables between neighboring subproblems.

There is one technical difficulty that we need to address: As long as there is a duality gap, we might end up in the optimum of the $L(\boldsymbol{\lambda})$ that corresponds to a conflict assignment for which we already added indicator variables. In this case, we would add redundant indicator variables and are likely stuck in a loop.

This problem can be circumvented by analyzing the hyperplanes around the local optimum of $L(\boldsymbol{\lambda})$. At least one of them is "blocking" our ascent to the $\mathbf{x}^*$ hyperplane (see Figure 5.2 for an illustration) and does not correspond to any of the previous conflict assignments for which we added indicator variables. We are not guaranteed to end up on this hyperplane after optimizing $L(\boldsymbol{\lambda})$, but only ϵ close to it. However, the sought hyperplane is a member of the current set of bundles (otherwise, it would not block our way). Therefore, it is sufficient to find a hyperplane in the current set of bundles that corresponds to a conflict assignment for which no indicator variable was added before. There might be several hyperplanes with this property in the current bundle set. In practice, we take the hyperplane with the closest value at $\boldsymbol{\lambda}^*$ to $L(\boldsymbol{\lambda}^*)$

5.3.2 Auxiliary Indicator Variables

Let us assume that $\mathbf{x}^{R*}$ does contain conflicts. In this case, we can identify pairs of regions r_1 and r_2 with disagreements in the assignment of their shared variables $V^{r_1,r_2} = V^{r_1} \cap V^{r_2}$. Let us denote the conflicting assignments of V^{r_1,r_2} as $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$, respectively.

We add auxiliary binary indicator variables $\mathbb{I}_{\bar{\mathbf{x}}_1}$ and $\mathbb{I}_{\bar{\mathbf{x}}_2}$ to each of the two involved subproblems for both conflicting assignments $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$. With two additional factors we ensure that the value of those indicator variables is one, if and only if the shared variables assume the respective assignments.

In practice, we exploit the fact that our subproblems are solved using integer linear programs. For that, we transform the subproblems into linear constraint binary form, as described in Appendix A.1. In this case, the addition of a binary indicator variable $\mathbb{I}$ (we omit the subscript in the following for better readability) for an assignment $\bar{\mathbf{x}}$ of n variables requires just two constraints. For the construction of those, let us assume for the moment the existence of a linear quantity s that counts the number of variables that do not take the desired label from $\bar{\mathbf{x}}$ under the current assignment $\mathbf{x}$. We will show shortly how this quantity can be constructed. The constraints that ensure that $\mathbb{I}$ is always properly set can now be expressed as

$$-s - \mathbb{I} \leq -1 \tag{5.19}$$

$$s + n\mathbb{I} \leq n. \tag{5.20}$$

The first constraint ensures that if the number of disagreements s is zero (*i.e.*, the desired assignment is selected) the indicator has to be one. The second constraints ensures that as soon as the number of disagreements is larger than zero (*i.e.*, the desired assignment is not selected), the indicator has to be zero. The quantity s is indeed a linear expression of the assignment $\mathbf{x}$ we wish to optimize over. To show that, we exploit the fact that our variables have binary domains. Let V be the set of variables that $\bar{\mathbf{x}}$ assigns values to. The number of disagreeing labels with respect to $\mathbf{x}$ is

$$s = \sum_{v \in V} \mathbb{1}_{\{\mathbf{x}_v = \bar{\mathbf{x}}_v\}} = \sum_{v \in V : \bar{\mathbf{x}}_v = 0} \mathbf{x}_v + \sum_{v \in V : \bar{\mathbf{x}}_v = 1} 1 - \mathbf{x}_v, \tag{5.21}$$

which we can now plug into the above mentioned constraints.

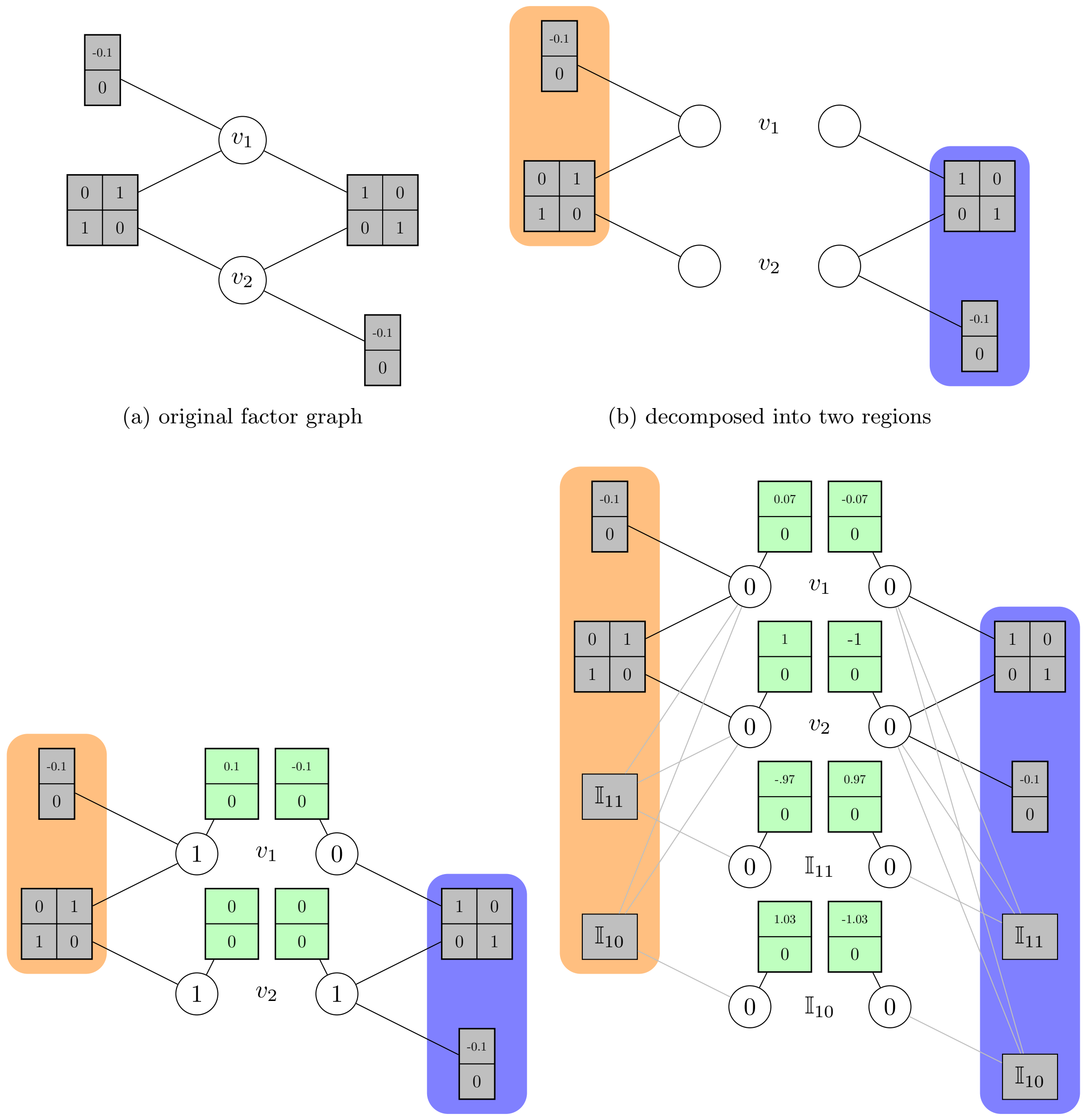
5.4 Results

5.4.1 Toy Example

To illustrate our gap closing strategy, we first show results on a simple toy example. This example consists of only two binary variables v_1 and v_2 in a factor graph with a frustrated cycle (see Figure 5.5a): Two binary factors prefer the assignments of the variables to be equal or unequal, respectively. The tie is broken by two unary factors that both prefer the assignment of 0. The decomposition into two regions (Figure 5.5b) and the following dual optimization are not able to find a consistent assignment for the shared variables (Figure 5.5c), since the corresponding LP relaxation to the decomposition is not tight. By adding indicator variables after the conflicting assignments have been identified, the global optimal solution can be obtained anyway (Figure 5.5d). The progress of the bundle method before and after the addition of indicator variables is shown in Figure 5.6.

5.4.2 Neuron Reconstruction

We evaluate the convergence properties of the proposed method in two simple setups. For that, we take the neuron reconstruction model on component trees (presented in Section 4.5 on page 44) that was trained using the structured learning formulation described in Chapter 4. We decompose and solve the according inference problem on a small test stack of *Drosophila melanogaster* larva neuropile. For that, we first decompose the inference problem into two subproblems of different size and overlaps to investigate the efficiency of the dual optimization. In a second setup, we decompose the inference problem into three subproblems of different size and overlaps to investigate the effect of more than two interacting subproblems.



(a) original factor graph

(b) decomposed into two regions

(c) $\boldsymbol{\lambda}$ changes (green) after first optimization

(d) $\boldsymbol{\lambda}$ changes (green) after adding indicator variables

Figure 5.5: Toy example to illustrate the gap closing strategy. The factor graph in (a) contains a frustrated cycle, the optimal solution is $v_1 = v_2 = 0$ with a value of 0.8. After the decomposition in (b), there is a duality gap: In the optimum of $L(\boldsymbol{\lambda})$, the regions still prefer conflicting assignments, as shown in (c), where we indicate the changes contributed by the $\boldsymbol{\lambda}$ as green factors. After adding indicator variables $\mathbb{I}_{11}$ and $\mathbb{I}_{10}$ for $v_1 = 1, v_2 = 1$ and $v_1 = 1, v_2 = 0$, the $L(\boldsymbol{\lambda})$ optimization is augmented by two new dimensions and achieves agreement between the shared variables.

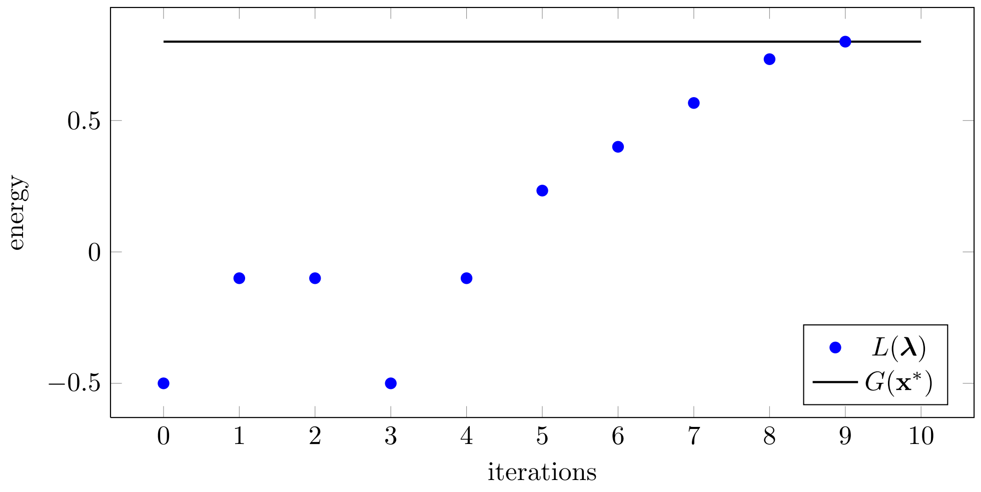


Figure 5.6: Optimization of $L(\boldsymbol{\lambda})$ for the toy example. After two iterations, the desired precision of the initial $L(\boldsymbol{\lambda})$ optimization was reached, but no consistent assignment was found. After adding an indicator variable for the conflict assignments and restarting the optimization in iteration 3, the duality gap could be closed and a consistent assignment was obtained.

Two Subproblems

We decomposed the inference problem of a stack of 20 EM images of size 1024×1024 from *Drosophila melanogaster* larva neuropil with 4.6nm xy-resolution and 50nm section thickness [31] into two subproblems of different size and overlap. The parameters of the proximal bundle method to optimize the dual $L(\boldsymbol{\lambda})$ have been left at their default values, which are a regularizer weight of 1.0, and an optimality threshold of $\epsilon = 10^{-5}$. Serious steps are performed if the optimization improvement exceeds $10^{-3}\epsilon_t$, where ϵ_t is the current gap estimate between the bundle approximation $\mathcal{L}(\boldsymbol{\lambda})$ and $L(\boldsymbol{\lambda})$. The progress of the bundle method for the subproblem sizes of 5 and 10 images with overlaps of 1, 2, 3, and 4 images are shown in Figure 5.7.

Interestingly, none of the pairs of subproblems suffered from a duality gap. After about 60 iterations for an overlap of one image, and about 200 iterations for an overlap of four images, the bundle method found $\boldsymbol{\lambda}$ that resulted in a consistent assignment. We also observe that the number of iterations needed to optimize $L(\boldsymbol{\lambda})$ depends primarily on the size of the overlap, suggesting that a smaller overlap is better.

Three Subproblems

In this setup, we decomposed the same inference problem used before into three subproblems of different size and overlaps. The parameters of the proximal bundle are left unchanged. The progress of the bundle method for the subproblem sizes of 5 and 6 images with overlaps of 1 and 2 images are shown in Figure 5.8.

For the experiments with an overlap of one image, the globally optimal solution was not obtained in the first round of the $L(\boldsymbol{\lambda})$ optimization. After adding indicator variables for the conflict assignment in the optimum, a second run found a consistent and hence globally optimal assignment across all three regions. The close up in Figure 5.9 shows that the first round of optimization ended up at almost the same energy as the second round with additional indicator variables. Nevertheless, the indicator variables have been needed to obtain a consistent assignment.

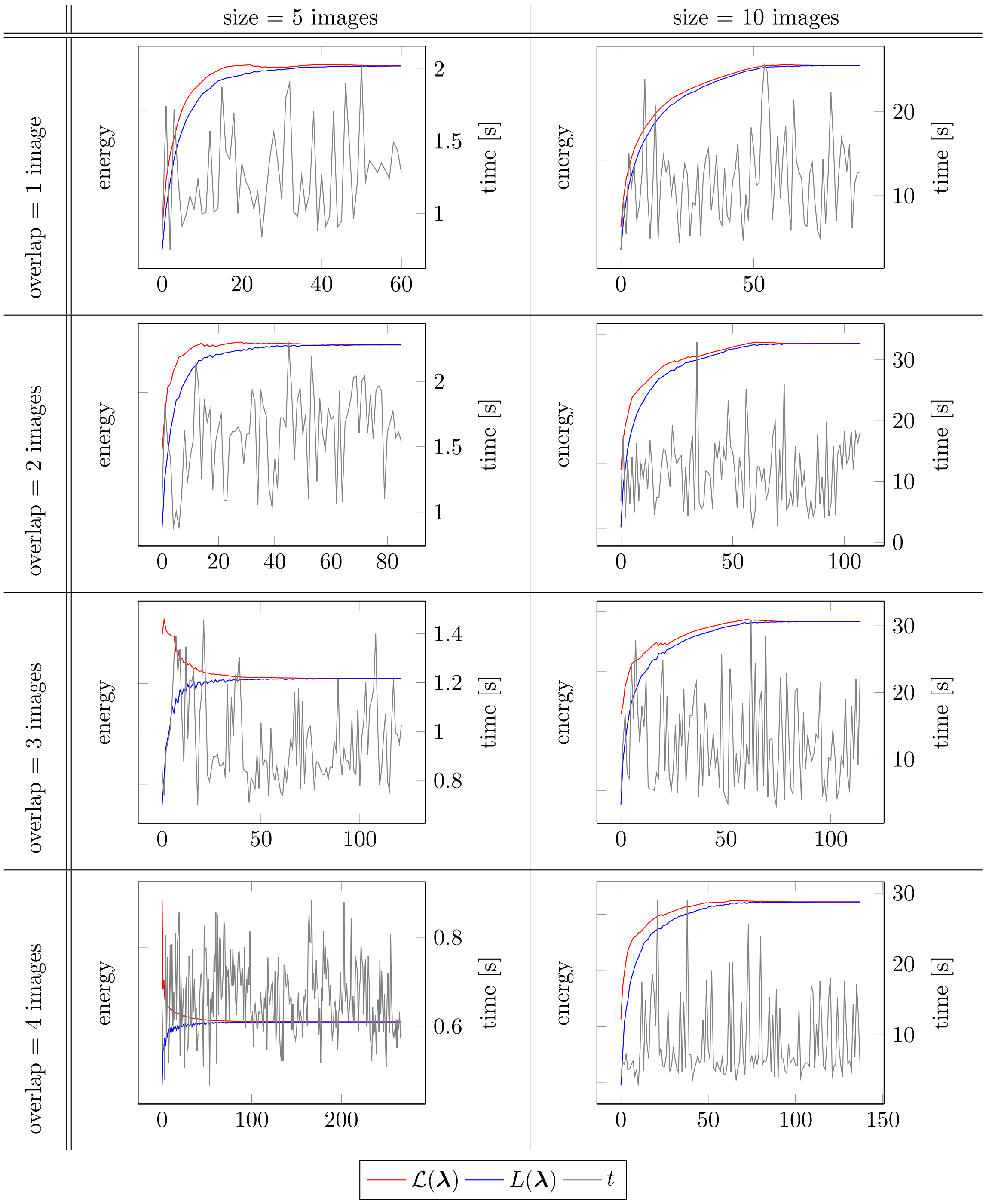


Figure 5.7: Progress of the bundle method optimizer for pairs of subproblems with different sizes (columns) and overlaps (rows). The subproblems have been generated for the model *structured learning* on component trees, as presented in Section 4.5 on page 44. Each plot shows over the number of iterations (x-axis) the current values of the dual $L(\boldsymbol{\lambda})$ (blue) and its upper bound approximation $\mathcal{L}(\boldsymbol{\lambda})$ (red) defined by the current set of hyperplanes. The maximal time needed to solve the subproblems in each iteration are superimposed with a black plot (t). In all runs, global optimality was achieved without adding indicator variables.

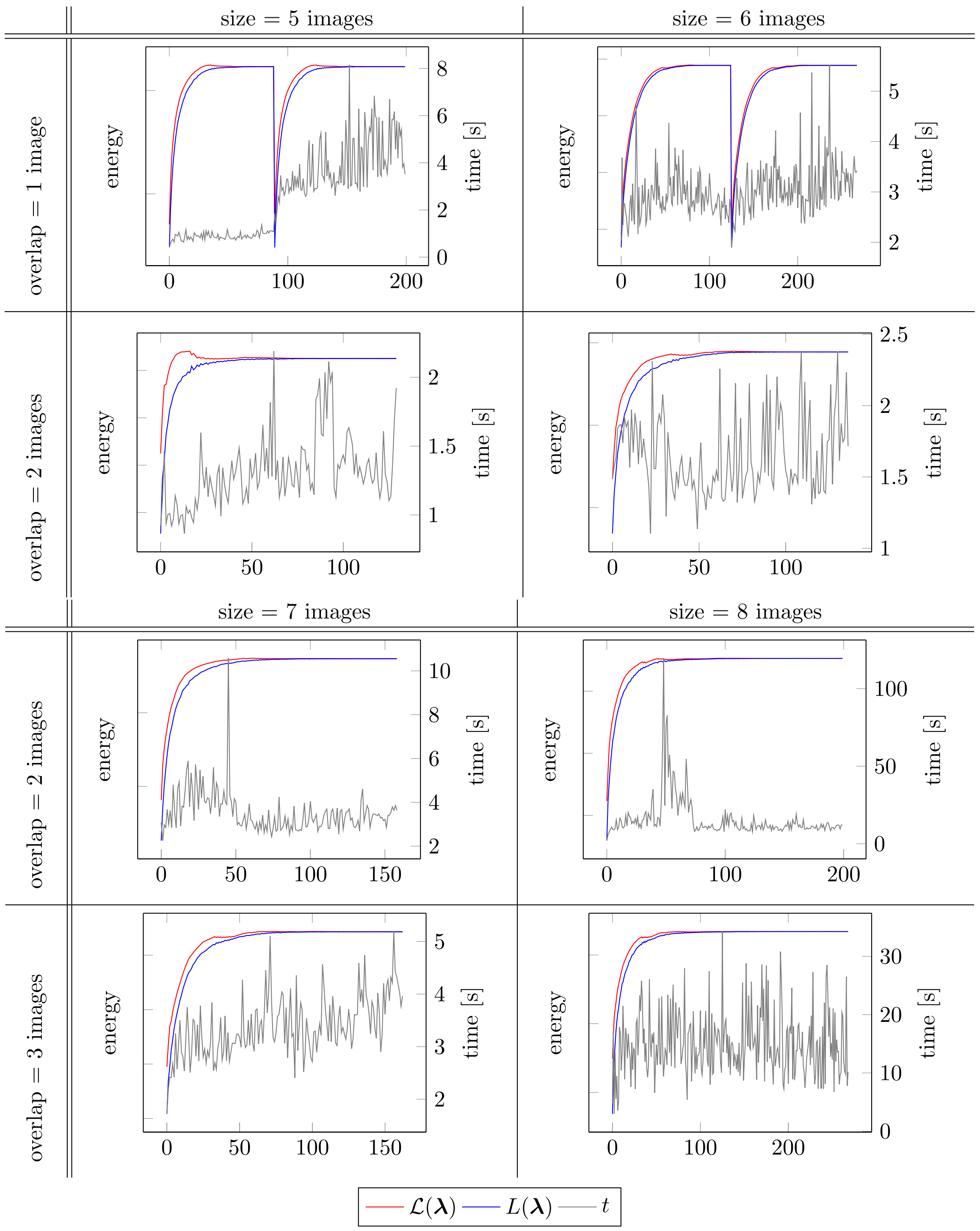


Figure 5.8: Progress of the bundle method optimizer for a decomposition into three subproblems with different sizes (columns) and overlaps (rows). The subproblems have been generated for the model *structured learning* on component trees, as presented in Section 4.5 on page 44. Each plot shows over the number of iterations (x-axis) the current values of the dual $L(\boldsymbol{\lambda})$ (blue) and its upper bound approximation $\mathcal{L}(\boldsymbol{\lambda})$ (red) defined by the current set of hyperplanes. The maximal time needed to solve the subproblems in each iteration are superimposed with a black plot (t). For the experiments with an overlap of one image (top row), indicator variables have been added and the optimization was restarted (discontinuity in the plots). After that, the globally optimal solution was found.

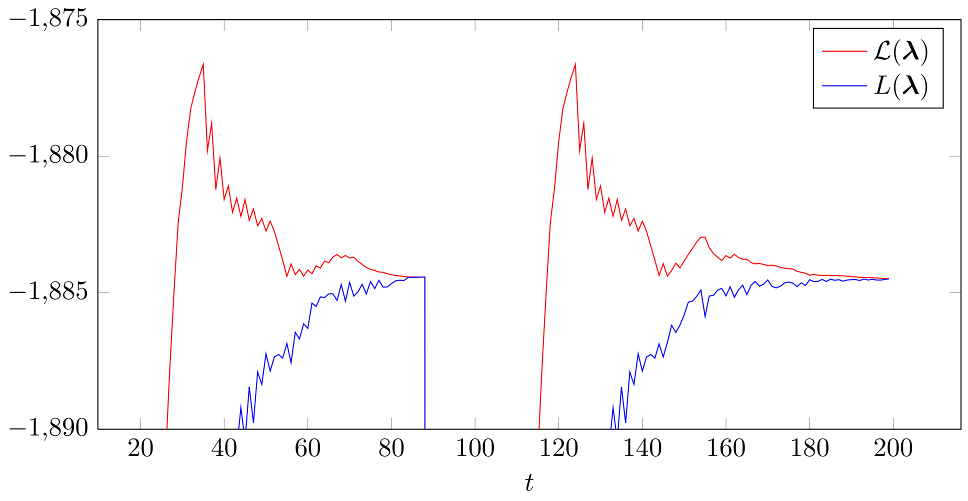


Figure 5.9: Close up of plot for size = 5 images and overlap = 1 image in Figure 5.8.

5.5 Discussion

The proposed problem decomposition and optimization scheme guarantees global optimality, as long as the decomposition graph is tree-shaped. It should be noted that this is a sufficient criteria and loopy decomposition graphs might deliver optimal solutions as well. For large subproblems, and in particular for the models presented in this thesis, frustrated cycles between the subproblems are very unlikely. This is confirmed by our experiments, where even between relatively small subproblems a globally optimal solution was found without the need to add indicator variables. This implies that the dual relaxation of the corresponding LP problem was tight, which in turn indicates that there have been no frustrated cycles in the overlaps between the subproblems.

Hence, there are two modes of operation imaginable for the application of our decomposition scheme to neuron reconstruction:

If the volume to process has only one very large dimension, subproblems could be formulated to decompose this large dimension into a chain. The resulting optimization problem could then be distributed for parallel or memory efficient computation. In this case, global optimality is guaranteed.

If, on the other hand, the volume is too big to allow a decomposition along one dimension only, a block-wise decomposition would need to be carried out. The resulting decomposition graph would be loopy and we lose the optimality guarantee of the resulting optimization. However, we would still be able to tell if a globally optimal solution was found, since a consistent assignment $\mathbf{x}^{R*}$ is a certificate for optimality. If there was indeed a duality gap, we would be able to detect this as well. In an interactive neuron reconstruction system, these cases could be reported to a user for manual inspection.

Finally, we would like to point out that we deliberately used the conjunctive in the previous paragraphs. Even the fastest $L(\boldsymbol{\lambda})$ optimizations in our experiments needed about 60 oracle calls, *i.e.*, each subproblem had to be solved that many times. In the context of large scale neuron reconstruction, this is a very high price to pay. It is questionable, whether the globally optimal solution is worth this extra efforts, especially in the light of the many reconstruction errors that current automatic methods still make. In fact, a simple strategy that solves overlapping subproblems only once and merges the conflicting assignments between them with a simple heuristic might just be as good.

Chapter 6

Error Measures for Neuron Reconstruction

6.1 Introduction

In the last decade the automatic volumetric reconstruction of neurons from electron microscopy has become an acknowledged part of computer vision research. As such, the reconstruction is an interesting computer vision problem on its own, assessing the ability of our methods to deal with noisy signals and missing data in a stereotypical setup with strong priors. The reconstruction of neurons can teach us valuable lessons about other similar problems in computer vision.

But besides being a challenge for computer vision, the automatic neuron reconstruction has a clear objective to meet biological needs. To serve as a tool to study the structure and function of nervous systems, the reconstruction needs to provide accurate 3D shapes of the neurons and has to be able to identify synaptic connections.

The reconstruction of the shape of neurons is important to identify individual neurons, to study neuron types, and to formulate neuron models. For stereotypical neural systems like *Drosophila melanogaster*, the neuron morphology can be used to identify neurons by matching them to light-microscopy based neuron atlases [79]. For less stereotyped neural systems, like the mammalian CNS, the morphology allows to determine a neuron's type and thus to formulate theories about their function [22]. For biophysically realistic neuron simulation, volumetric information is needed to model action potential time dynamics, and to understand and simulate information processing capabilities of single neurons [70].

The detection of synapses and the identification of synaptic partners is important to analyze the function of neural circuits and to understand the effects of learning. The connectivity information allows to test hypotheses about how individual neurons affect the function of a neural circuit [22]. The size and number of synaptic connections can be used to study the effects of learning [8].

6.1.1 Shape and Synapse Accuracy

From the use cases of neuron reconstruction, we can identify two requirements for automatic reconstruction approaches: the *shape accuracy* and the *synapse accuracy*.

A high shape accuracy is achieved if the volumetric reconstruction of neurons follows the true morphology within certain tolerance bounds on the cell boundary location. The reasons for the tolerance in the boundary location are two-fold: First, the EM preparation protocol can alter the volume of a neural process, such that it is hard to know what the true size was [93]. Second, the location of membranes in EM images can be ambiguous, such that it is hard to clearly identify the boundary of a neuron. In fact, it is commonly assumed that there is no unique "ground truth" for the boundary location [16]. More important than the exact location of the cell boundary is the topological correctness. Missing or falsely added branches can change the morphology of the neuron dramatically.

The synapse accuracy reflects the correct identification of neurons and their synaptic connections, such that a connectivity graph can be obtained. For that, it is important to identify pairs of pre- and postsynaptic sites and match them to the neurons they belong to. A high shape accuracy is in general not a prerequisite of a high synapse accuracy, although in many practical implementations this will be the case.

6.1.2 Ground Truth

It has been noted that finding a "ground truth" for neuron reconstruction is a challenging task on its own [14, 99]. This is especially true for minor ambiguities like boundary shifts, which do not change the resulting connectivity graph and only marginally change the shape statistics of the neuron. However, we also note from our experience that in some situations – namely in the presence of noise, and thin and densely packed structures – even human experts fail to resolve ambiguities. In these cases, the annotators have to resort to assign lower confidence values to their decisions. These values can help to remind annotators to revisit an area later, when a larger reconstruction provides the context to resolve the ambiguity.

The sizes of the volumes used for the training and testing of automated reconstruction approaches do usually not provide the context needed to resolve all of the mentioned ambiguous situations. As a consequence, we have to accept that a "ground truth" labeling is most likely not free of errors.

Nevertheless, what we call the ground truth can be seen as a best-effort solution to the reconstruction, given the available data. To train algorithms to imitate the experts who created the labeling is still desirable, despite subtle errors in the labeling. This is especially true since we observe that, currently, the number of errors made by algorithms is orders of magnitudes higher than the number of unconfident decisions in the ground truth. Therefore, for the purpose of the discussion in this paper, we assume the existence of a unique ground truth.

6.1.3 Outline

The error measures used in current publications on automatic neuron reconstruction are the Rand index (RI), variation of information (VOI), the anisotropic edit distance (AED), and the warping error (WE). In the following section, we give a short survey of these measures. We introduce a new error measure, the tolerant edit distance (TED) in Section 6.3. We also discuss a potential measure to evaluate the synapse accuracy in Section 6.4 and highlight its current limitations. We investigate the suitability of the TED to assess the shape accuracy in Section 6.5 and compare it against the currently most used measures. In Section 6.6 we discuss the properties of the TED in the light of automatic neuron reconstruction and its specific needs.

6.2 Survey of Current Error Measures

In the computer vision literature, several approaches to asses the quality of contour detection and segmentation algorithms can be found. Most of these measures have been designed to capture the intuition of what humans consider to be similar results. In particular, these measures are supposed to be robust to certain tolerated deviations, like small shifts of contours. For the contour detection in the Berkeley segmentation dataset [74], for example, the precision and recall of detected boundary pixels within a threshold distance to the ground truth became the widely used standard [3, 75]. Contour error measures are, however, not a good fit for segmentation comparison, since small errors in the detection of a contour can lead to the split or merge of label regions. Therefore, several alternatives have been proposed, like the variation of information, the Rand index [82], as well as the probabilistic Rand index [100] and the normalized probabilistic Rand index [101], and the segmentation covering measure [3].

With the advent of automatic methods for neuron reconstruction, some of the segmentation error measures were adopted to compare the quality of different approaches. The error measures were chosen to represent what was called the topological errors in neuron reconstruction. This follows the intuition that merging or splitting a neuron is a more severe error than a boundary shift, even if the number of affected pixels is smaller.

In the remainder of this section, we give a short survey of the most popular error measures used for neuron reconstruction. These error measures motivated the design of the tolerant edit distance (TED) that we introduce in Section 6.3.

6.2.1 Notation

We assume that a neuron reconstruction can be represented as a label function

$$y : \Omega \mapsto K_y, \tag{6.1}$$

where $\Omega \subset \mathbb{N}$ is a discrete set of pixel locations in an EM volume of neural tissue and

$$K_y = \{0, 1, 2, \ldots\} \tag{6.2}$$

is a set of neuron identifiers used by the labeling y, where we use 0 to denote an optional background label. We say that y imposes a partition Y of Ω into non-empty subsets Y_k. Formally, this partition is given by

$$Y = \{Y_k \mid Y_k = \{i \in \Omega \mid y(i) = k\}, Y_k \neq \varnothing, k \in K_y\}. \tag{6.3}$$

Note that this formulation does not require regions with the same label to form connected components in the volume. This way, we support labelings of anisotropic volumes with registrations errors (*i.e.*, the 2D images do not line up perfectly) or missing data.

6.2.2 Rand index

The Rand index (RI) directly compares two partitions X and Y in terms of the agreement of pairs of elements in both partitions. Two partitions are said to agree on a pair $(i, j) \in \Omega^2$, if both partitions assign the pair to the same subset, *i.e.*,

$$x(i) = x(j) \text{ and } y(i) = y(j), \tag{6.4}$$

or both partitions assign the pair to two different subsets, *i.e.*,

$$x(i) \neq x(j) \text{ and } y(i) \neq y(j). \tag{6.5}$$

Let a be the number of agreeing pairs and $n = \binom{|\Omega|}{2}$ the number of all pairs in Ω. The Rand index is then given by

$$RI(X, Y) = \frac{a}{n}, \tag{6.6}$$

i.e., the fraction of agreeing pairs on all possible pairs. Hence, the Rand index is 1 for two equivalent partitions and 0 in the worst case where one partition contains only one subset of size $|\Omega|$ and the other one $|\Omega|$ partitions of size one[1]. In practice, where we have one of the partitions fixed as the ground truth, a Rand index of zero is therefore never reached.

[1]Proof by contradiction: Assume that one of the worst case partitions contains equal and unequal pairs, then we can find a triple (i, j, l) such that $i = j$, $i \neq l$, $j \neq l$. In the other partition, to find no agreement, we would need to have $i \neq j$, $i = l$, and $j = l$, which is not possible. Hence, the partitions can contain either only unequal pairs or only equal pairs.

6.2.3 Variation of Information

The variation of information (VOI) measures the similarity of two partitions X and Y indirectly by considering two random variables A and B and their conditional probabilities. A and B represent the chance of obtaining a certain label k in X and a label l in Y when randomly and uniformly selecting a location $i \in \Omega$. Thus,

$$p(A = k) = \frac{|X_k|}{|\Omega|} \text{ and } p(B = l) = \frac{|Y_l|}{|\Omega|}, \tag{6.7}$$

and their conditional distributions are

$$p(A = k|B = l) = \frac{|X_k \cap Y_l|}{|Y_l|} \text{ and } p(B = l|A = k) = \frac{|X_k \cap Y_l|}{|X_k|}. \tag{6.8}$$

Now, the VOI of these two random variables is the sum of the two conditional entropies of one variable given the other:

$$\mathrm{VOI}(A, B) = H(A|B) + H(B|A). \tag{6.9}$$

The conditional entropy $H(A|B)$ gives the number of bits needed to determine the value of A, if B is already known; and vice versa for $H(B|A)$. More formally,

$$H(A|B) = \sum_l p(B = l)H(A|B = l) \tag{6.10}$$

$$= \sum_l p(B = l) \sum_k -p(A = k|B = l) \log(p(A = k|B = l)). \tag{6.11}$$

Intuitively, the two summands of the VOI give a measure for the split or merge error of a reconstruction in comparison to the ground truth. Assume that A corresponds to the labels of the ground truth, and B of the reconstruction. If a ground truth neuron with label k was accidentally split into two neurons in the reconstruction, the number of bits needed to determine B for all locations where $A = k$ will be bigger than zero and at most one. Conversely, if two ground truth neurons got accidentally merged into a single neuron with label l, the number of bits needed to determine A for all locations where $B = l$ is bigger than zero. An example of this measure for the split of a single region can be seen in Figure 6.1.

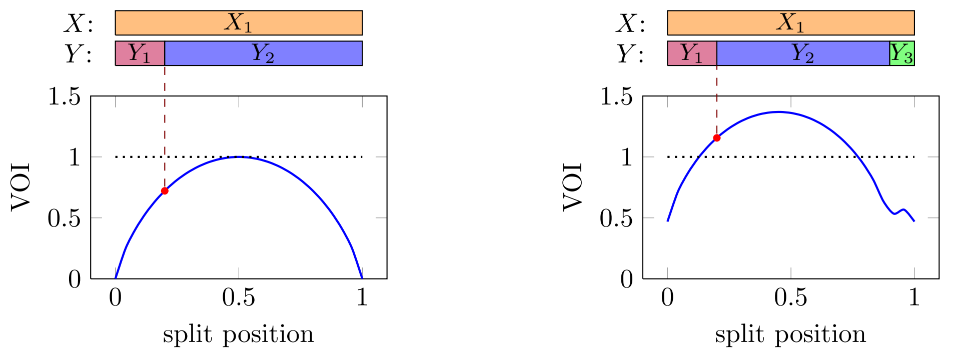
6.2.4 Warping Error

The warping error by Jain et al. aims to measure the difference between a ground truth and a reconstruction in terms of their topological differences [40]. As such, it is the first error measure for neuron reconstruction that deals with the delicate question up to which point boundary shifts are not considered as errors.

The ground truth and the reconstruction are assumed to be given as foreground vs. background labelings, such that each connected component of the foreground color represents a neuron. This implies that two neurons have to be separated by an at least one pixel wide background sheet. This is in contrast to our definition of a labeling (see Section 6.2.1), where the identity of a neuron is maintained by the label instead of the connectedness.

The warping error is the minimal Hamming distance between two binary labelings x and y over a set of possible *warpings* of y. A warping is a sequence of topology-preserving relabelings of single locations $i \in M \subseteq \Omega$. This set M is called the mask and encodes the tolerable deviation between the ground truth labeling and the reconstruction: Labels within M can be changed, as long as the topology of the involved connected components does not change. In particular, this allows the shift of object boundaries within M, so that a slightly mislabeled boundary does not contribute to the final error measure. More formally, the warping error between two labelings x and y is

$$W(x, y) = \min_{y' \lhd y} \sum_{i \in \Omega} (x(i) - y'(i))^2, \tag{6.12}$$



(a) VOI for one split as a function of the split position.

(b) VOI for two splits as a function of one of the split positions.

Figure 6.1: VOI shown on an example of splitting one region X_1 into two regions Y_1 and Y_2 (a) and three regions Y_1, Y_2, and Y_3 (b). The plots show the effect of moving (a) the split position and (b) one of the split positions along the whole Y. It can clearly be seen that the VOI does depend on the split position. Furthermore, there are several situations where splitting into three regions gives a lower VOI then the worst case VOI for splitting into two regions (dotted line).

where $y' \lhd y$ denotes a warping y' of a labeling y.

The mask M is defined as the set of all image locations that are within a threshold Euclidean distance to the background label of the ground truth. Thus, reconstructed neurons are allowed to grow and shrink by a certain amount, unless they touch or split.

Since finding the exact warping error by solving Eq. (6.12) is not trivial, Jain et al. propose to follow an efficient greedy optimization strategy. Of all possible relabelings of single locations in y that would reduce the Hamming distance, one of them is performed randomly until no further improvements can be made.

6.2.5 Anisotropic Edit Distance

The anisotropic edit distance (AED) that we introduced in [26] counts the errors of a reconstruction in terms of the edit distance to the ground truth. Four different types of errors that are specific to anisotropic neuron reconstruction are counted: missed and spurious neuron slices within a section, and missed and spurious links between the neuron slices across sections (see Figure 6.2 for an illustration).

The central concept of this error measure is the matching of neuron slices (*i.e.*, 2D neuron cross-sections) in each section between the reconstruction and the ground truth. Neuron slices can either be matched one-to-one (if they are similar), or not at all. Let S_R be the set of all neuron slices of the reconstruction in all sections, and S_G the corresponding set of slices in the ground truth. A matching $m = (V, E)$ of the neuron slices is a bipartite graph with

$$V = S_R \cup S_G \tag{6.13}$$
$$E \subset S_R \times S_G, \tag{6.14}$$

where the order of nodes is at most one to ensure that each neuron slice is matched at most once.

For any given matching m of neuron slices, we evaluate the AED in the following way: Every not matched neuron slice in the reconstruction is a false positive (FP), every not matched neuron slice in the ground truth is a false negative (FN). Let $d_m : V \mapsto \{0, 1\}$ be the degree of a slice $v \in V$ under a mapping m, *i.e.*, it is one if the slice v was matched and

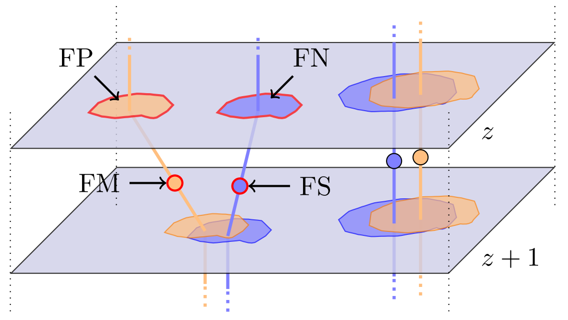


Figure 6.2: Visualization of four possible error types in the anisotropic edit distance (AED). An example reconstruction is shown in orange, the ground truth is shown in blue. Reconstruction slices (*i.e.*, 2D neuron cross-sections) are matched with the ground truth based on a threshold on their overlap ratio. Ground truth slices that have not been matched are counted as false negatives (FN), reconstruction slices that have not been matched are counted as false positives (FP). Links connecting slices across sections are matched between the reconstruction and ground truth, if the involved slices match. Ground truth links that have not been matched are counted as false splits (FS), reconstruction links that have not been matched are counted as false merges (FM). If there are multiple ways to match slices between the reconstruction and ground truth, the matching that minimizes the total number of errors is chosen.

zero otherwise. The FP and FN errors can now be expressed as

$$\mathrm{FP}(m) = \sum_{v \in S_R} 1 - d_m(v) \tag{6.15}$$

$$\mathrm{FN}(m) = \sum_{v \in S_G} 1 - d_m(v). \tag{6.16}$$

Further, we match the links (*i.e.*, the connections between the neuron slices of two subsequent sections) between the ground truth and the reconstruction. Two links are said to match, if both of the involved slices match. Every not matched link in the reconstruction is a false merge (FM), every not matched link in the ground truth is a false split (FS). Let $L_R \subset S_R \times S_R$ be the set of links in the reconstruction, and L_G the corresponding set of links in the ground truth. The FS and FM errors can now be expressed as

$$\mathrm{FS}(m) = \sum_{(s,t) \in L_R} 1 - \sum_{(s',t') \in L_G} \mathbb{1}_{\{(s,s') \in E\}} \mathbb{1}_{\{(t,t') \in E\}} \tag{6.17}$$

$$\mathrm{FM}(m) = \sum_{(s,t) \in L_G} 1 - \sum_{(s',t') \in L_R} \mathbb{1}_{\{(s,s') \in E\}} \mathbb{1}_{\{(t,t') \in E\}}, \tag{6.18}$$

where $\mathbb{1}_\alpha$ returns 1 if α is true, otherwise 0.

Since this procedure is sensitive to the matching m of the neuron slices, we select an optimal matching that minimizes the total number of errors in the following way: First, all possible neuron slice matchings between the ground truth and the reconstruction are enumerated for each section individually. For that, only neuron slices that share at least a certain percentage k of the union of their pixels are considered as match candidates. This gives a set M_k^z of possible matchings for each section z. Each matching $m^z \in M_k^z$ proposes a number of $\mathrm{FP}(m^z)$ and $\mathrm{FN}(m^z)$ errors for this particular section. Similarly, each pair

$$(m^z, m^{z+1}) \in M_k^z \times M_k^{z+1}$$

of matchings of consecutive sections z and $z+1$ proposes a number of $\mathrm{FS}(m^z \cup m^{z+1})$ and $\mathrm{FM}(m^z \cup m^{z+1})$ errors.

Let n be the number of sections in the stack, and x and y the reconstruction and ground truth labeling, from which the sets of neuron slices S_R, S_G and the sets of links L_R, L_G are extracted. We denote the set of all possible matchings for the whole stack as

$$\mathcal{M}_k = M_k^1 \times \ldots \times M_k^n.$$

The AED is now the minimal number of total errors for a similarity threshold k, and can be found by the following minimization:

$$\mathrm{AED}_k(x,y) = \min_{m \in \mathcal{M}_k} \sum_{z=1}^{n} \mathrm{FP}(m^z) + \mathrm{FN}(m^z) + \sum_{z=2}^{n} \mathrm{FS}(m^{z-1} \cup m^z) + \mathrm{FM}(m^{z-1} \cup m^z). \quad (6.19)$$

We find the matching minimizing the AED in a forward-backward like scheme, similar to inference on a Markov chain, where each node represents the possible matchings of a section. If the similarity threshold k is smaller or equal to 50%, there is only one possible matching per section. In this case, there is no need to perform an optimization.

6.3 Tolerant Edit Distance

The tolerant edit distance (TED) is a generalization of the warping error described in Section 6.2.4 and is based on two observations: (1) Certain types of errors like boundary shifts or holes in the reconstructed neurons are tolerable to some extent. Which errors are tolerable and to what extent is application dependent and should be left as a parameter of the measure. (2) The errors made by current automatic reconstruction methods render manual proofreading necessary. Minimizing the time that has to be spent on fixing the errors is therefore a sensible objective.

The TED measures the difference between two labelings x and y as defined in Section 6.2.1 in terms of the minimal number of splits and merges (and in the presence of a background label also false positives and false negatives) over a set of possible relabelings of y. How y is allowed to be relabeled is defined on a tolerance criterion, *e.g.*, the maximal displacement of an object boundary.

Let a tolerance function T be a binary indicator on two labeling functions y and y':

$$T(y,y') = \begin{cases} 1 & \text{if } y \text{ is similar to } y', \\ 0 & \text{otherwise.} \end{cases} \quad (6.20)$$

Further, let $\mathcal{Y}$ be the set of all labeling functions $y' : \Omega \mapsto K_y$, *i.e.*, all possible labelings of Ω using the labels of y, and let $\mathcal{Y}^+(y) = \{y' \in \mathcal{Y} \mid T(y,y') = 1\}$ be the set of all tolerated relabelings of y.

For two labelings x and y, let $M(x,y)$ be the matching matrix, *i.e.*, a matrix that counts how many times a label k from x coincides with a label l from y. Formally,

$$M(x,y)_{kl} = \sum_i \mathbb{1}_{\{x(i)=k \wedge y(i)=l\}}, \quad (6.21)$$

where $\mathbb{1}_\alpha$ returns 1, if α is true, and 0 otherwise. Each row k in $M(x,y)$ that contains more than one non-zero entry indicates a split of label k from x into several labels from y. Analogously, each column l with more than one non-zero entry indicates a merging of several labels from x into label l from y. Hence, the number of split and merge errors is given by

$$\mathrm{splits}(x,y) = \sum_{k \in K_x} \left(\left[\sum_{l \in K_y} \mathbb{1}_{\{M(x,y)_{kl} > 0\}} \right] - 1 \right) \quad (6.22)$$

and

$$\mathrm{merges}(x,y) = \sum_{l \in K_y} \left(\left[\sum_{k \in K_x} \mathbb{1}_{\{M(x,y)_{kl} > 0\}} \right] - 1 \right). \quad (6.23)$$

The TED is now the minimal number of split and merge errors over all tolerable relabelings of y:

$$\mathrm{TED}(x,y) = \min_{y' \in \mathcal{Y}^+(y)} \mathrm{splits}(x,y') + \mathrm{merges}(x,y'). \tag{6.24}$$

6.3.1 Optimization Problem for Local Tolerance Functions

In the following, we show how the TED can be computed by solving an integer linear program, if the tolerance function is local. We say that a tolerance function is local, if for two given labelings y and y' we can determine its value by considering each location $i \in \Omega$ independently[2]. An example for a local tolerance function is the boundary shift with threshold θ: To determine whether a label k can be assigned to a location i, it is sufficient to figure out whether label k appears on other locations j within a distance of θ around i. More generally, local tolerance functions allow each location to be relabeled from a set of labels $A_i \subseteq K_y$, including the original label $y(i)$. Choosing from any of these labels yields a tolerated relabeling. See Figure 6.3 for an example. Formally, the local tolerance function is

$$T(y,y') = \prod_{i \in \Omega} \mathbb{1}_{\{y'(i) \in A_i\}} \tag{6.25}$$

for a set of relabel possibilities $\mathcal{A} = \{A_i | i \in \Omega\}$.

In contrast to the warping error, the tolerable relabelings defined this way do not preserve the connectedness of regions in general. This is not needed, here, since the identity defining criterion is not connectedness, but the label of the region.

To find the sought labeling y' that minimizes the sum of errors, we introduce one binary indicator variable $v_{i=l}$ for each location $i \in \Omega$ and each possible label $l \in A_i$ it can assume. We formulate the following constraints to ensure that exactly one of the labels gets chosen for each location and that each original label has to appear at least once:

$$\sum_{l \in A_i} v_{i=l} = 1 \quad \forall i \in \Omega, \tag{6.26}$$

$$\sum_{i \in \Omega} v_{i=l} \geq 1 \quad \forall l \in K_y. \tag{6.27}$$

Further, we introduce binary variables a_{kl} that indicate the joint assignment of label k from x and label l from y' at at least one location. With the following constraints we make sure that $a_{kl} = 1$ if and only if there is at least one location $i \in \Omega$ such that $x(i) = k$ and $y'(i) = l$:

$$a_{kl} - v_{i=l} \geq 0 \quad \forall k \in K_x, \forall l \in K_y, \forall i \in \Omega : x(i) = k, \tag{6.28}$$

$$\sum_{i \in \Omega : x(i)=k} v_{i=l} - a_{kl} \geq 0 \quad \forall k \in K_x, \forall l \in K_y. \tag{6.29}$$

For each label k from x, we can now introduce an integer s_k, counting the number of times this label is split in y'. This is the number of times k was matched with any other label minus one (see Eq. (6.22)):

$$s_k - \sum_{l \in K_y} a_{kl} = -1 \quad \forall k \in K_x. \tag{6.30}$$

Similarly, we introduce variables m_l, counting the number of merges caused by label l from y':

$$m_l - \sum_{k \in K_x} a_{kl} = -1 \quad \forall l \in K_y. \tag{6.31}$$

[2]In practice, it is not necessary to consider each voxel $i \in \Omega$ separately. By intersecting the ground truth and the reconstruction regions, a set of super-voxels can be identified and used to replace Ω (see Figure 6.3 for an example). This is both more efficient and allows for more elaborate tolerance criteria. Therefore, we assume an Ω of some general locations in the following.

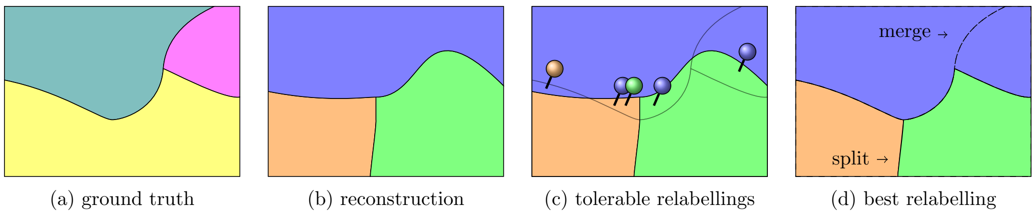


Figure 6.3: Illustration of the tolerant edit distance (TED) between a ground truth labeling (a) and a reconstruction labeling (b). (c) According to a local tolerance criterion, the reconstruction is allowed to be relabeled to match the ground truth as closely as possible. For that, regions obtained by intersecting the ground truth and reconstruction are considered: For each of these regions, the tolerance function allows a set of alternative labels (indicated by colored pins) that would change the labeling only within certain bounds. This can be, for instance, a boundary shift criterion, allowing relabelings that correspond to boundary shifts up to a certain distance. Regions without a pin can not be relabeled according to the tolerance function and have to keep their reconstruction label. All other regions can choose between their original label or any of the pin labels. (d) From all the possible ways to relabel the reconstruction, the relabeling minimizing the number of split and merge errors compared to the ground truth is chosen by solving an integer linear program.

The final split and merge numbers s and m are just the sums of the label-wise split and merges:

$$s - \sum_{k \in K_x} s_k = 0 \quad \text{and} \tag{6.32}$$

$$m - \sum_{l \in K_y} m_l = 0. \tag{6.33}$$

The TED is now the optimal objective value of the following optimization problem:

$$\min_{v} \quad m + s \tag{6.34}$$

$$\text{s.t.} \quad \text{Eq. (6.26) - Eq. (6.33).} \tag{6.35}$$

6.4 Synapse Error

Let us assume for a moment that the desired outcome of a neuron reconstruction approach is only the connectivity structure between neurons in terms of synapses, referred to as the *connectome* [34, 68]. Such a connectivity structure could be obtained by methods that indicate the presence of synapses and their pre- and post-synaptic partner locations. Together with a labeling function x for neuron identities in the volume, this would allow us to match pre- and post-synaptic locations that belong to the same neuron.

The connectivity structure can be represented by a labeled directed graph $G = (V, E, s)$ where V is the set of all neurons, $E \subseteq V \times V$ is a set of ordered tuples indicating the existence of pre- to post-synaptic connections between pairs of neurons, and $s : E \mapsto \mathbb{N}$ is a labeling function on the edges indicating the number of synaptic connections.

Given a ground truth labeling x and a reconstruction y together with corresponding synapse annotations yields two connectivity graphs G_x and G_y. The synapse error (SE) can now be defined as the minimal graph edit distance [28] between G_y and G_x. For that, we allow to add and remove vertices (neurons), add and remove edges (synaptic connectivity), and alter the edge labelings (number of synaptic connections). Although finding the minimal number of edit steps to match graphs is in general NP-hard, polynomial time approximations can be found by relaxing an integer linear program formulation [47].

It is, however, questionable if such an error measure would make sense in practice. First, the location and morphology of the involved neurons are not considered. From a biological perspective, it is desirable to get the morphology together with the connectivity information of neural circuits. How well an automatic reconstruction reflects both is not expressed by this measure. Second, we assume the availability of ground truth large enough to contain several connected neurons to make the error measure meaningful. Although there are current efforts to create datasets of that size in the form of skeleton reconstructions [32], these datasets take a long time to be created by hand and their size might make the evaluation of the synapse error and the training to minimize the synapse error intractable. Finally, current methods for the automatic reconstruction of neurons provide a shape accuracy that is orders of magnitudes below the human performance. Hence, we would expect a lot of errors in the connectivity graph. In those situations, the minimal edit distance will be expensive to be computed and also loses its meaning.

In summary, the synapse error as defined above is limited to cases where only the connectivity structure of neurons is of interest and not their identity or morphology. For computer vision based approaches, this is usually not the case. There are other interesting directions like the sequencing of the connectome via virus injections [111], but these methods do not allow to extract a ground truth for comparison. We believe that the neuron identities and morphologies should be considered as well to assess the quality of computer vision based approaches in terms of the connectivity structure. This could be achieved by altering the synapse error formulation above to constraint the possible edit steps to structures that are spatially close and similar. However, current methods do not predict synaptic connections, their shape accuracy is not sufficient, and we do not have ground truth datasets large enough to get reliable estimates of the synapse error. We have to conclude that with current methods we are not ready to evaluate the errors on the connectivity structure.

6.5 Experiments

In this section, we review several error measures with respect to their ability to quantify the shape accuracy of the reconstruction. First, we show how different error measures respond to the shift of an object boundary on artificial data. In a second setup, we take a human generated volumetric ground truth labeling that we modify in several ways: we grow the membrane label, and we split and merge neurons at randomly picked locations. The modified ground truth is considered as a possible reconstruction and is compared against the original ground truth. Finally, we take the best automatic reconstruction result obtained with the method described in Chapter 4 and investigate how the TED changes for different tolerance criteria.

We restrict our analysis to error measures that can be used both for isotropic and anisotropic volumes, *i.e.*, the Rand index (RI), variation of information (VOI), and the tolerant edit distance (TED).

6.5.1 Shift of Object Boundary

To compare the behaviour of different error measures in the case of object boundary displacements, we created a simple artificial 1D labeling consisting of two regions. We show the errors of reconstructions obtained by shifting the object boundary in Figure 6.4.

6.5.2 Shape Accuracy on Modified Ground Truth

We use ground truth of a volume [31] consisting of 20 sections of size 1024×1024 pixels, each voxel representing about $4 \times 4 \times 40nm$. We evaluated the error measures RI, VOI, and TED in three experiments: "grow", "split", and "merge". For the "grow" experiment, we dilated the neuron labels by $10nm$. For the "splits" experiment, we randomly picked 10 locations in the volume and manually removed the closest link of a neuron to the previous section. For

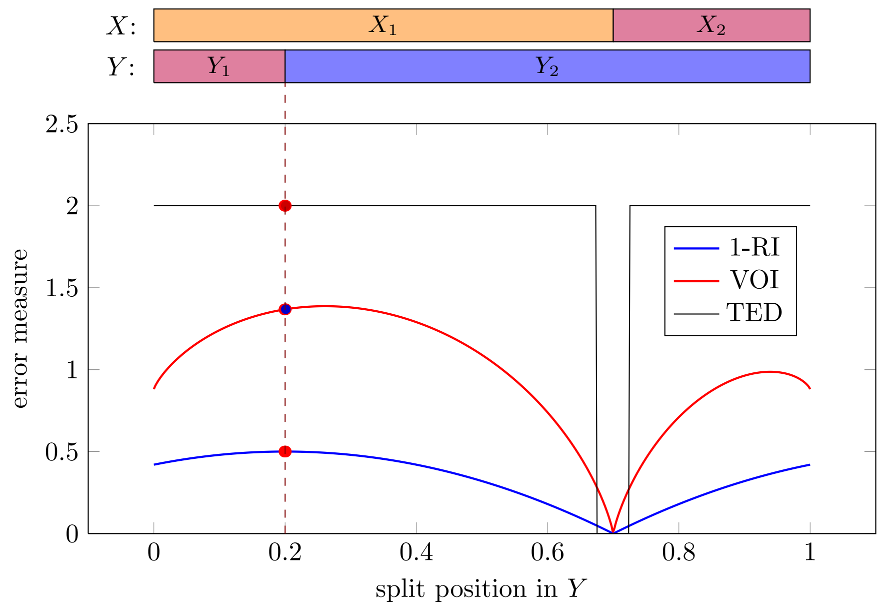


Figure 6.4: Comparison of Rand index (RI), variation of information (VOI), and tolerant edit distance (TED) as functions of object boundary displacements. Given a ground truth labeling X, the error measures are plotted as functions of the split position between two objects in a reconstruction Y. The TED is given as the sum of the possible errors, which is one split and one merge error unless the reconstruction object boundary is within the tolerated distance (0.025 in this example) to the true object boundary. VOI is in bits (lower is better) and 1-RI is 1 minus the ratio of agreeing pairs over all pairs (lower is better).

the "merges" experiment, we randomly picked 10 locations in the volume where a neural process ended and introduced a link to the closest neuron in the direction of the end. The results comparing the Rand index (RI), variation of information (VOI), and the tolerant edit distance (TED) with a boundary shift tolerance criterion of $20nm$ are shown in Figure 6.5. In this particular setup, the TED is the only measure that is able to disambiguate between minor boundary shifts and real morphological errors.

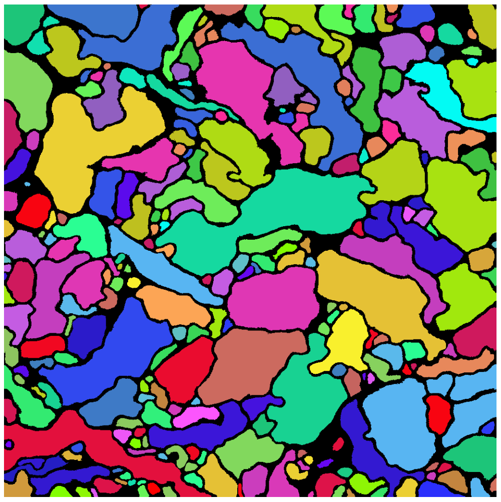
6.5.3 Influence of Tolerance Criteria for TED

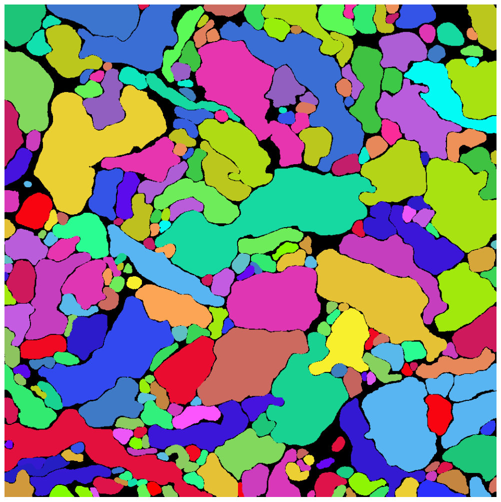
The tolerant edit distance is by design sensitive to the given tolerance criteria. In Figure 6.6 we investigate how the TED changes for different boundary shift tolerances on the reconstruction result of our method, as described in Chapter 4. By tolerating shifts up to $40nm$ (corresponding to $10 \times 10 \times 1$ pixels in the x-, y-, and z-direction), the TED reveals that our reconstruction suffers mainly from false negatives (*i.e.*, missed neural processes) and split neurons, which matches our observations. Further increasing the threshold shows that indeed the biggest errors are caused by false negatives and splits.

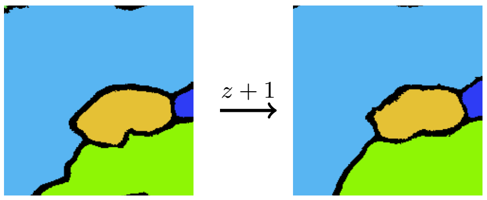
6.6 Discussion

In general, it is difficult to say what a good error measure for neuron reconstruction is. It depends on the concrete application and its accuracy requirements. However, we can safely say that being concious of the assumptions behind different error measures is important to interpret the results.

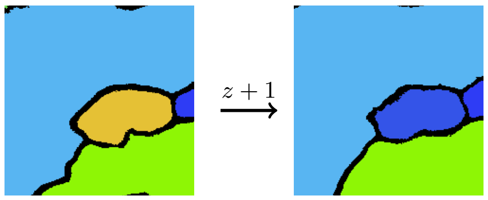
A key property of the TED is its explicitness. A user can directly state what exactly should be counted as an error. The boundary shift criterion that we used in the experiments


(a) original ground truth


(b) grown by 10nm



(c) original ground truth



(d) example split

experiment	RAND	VOI	TED			
			S	M	FP	FN
grow	0.964	1.098	0	0	0	0
10 splits	0.999	0.035	10	0	0	0
10 merges	0.999	0.048	0	10	0	0

(e) Error measure results for different ground truth modifications.

Figure 6.5: Comparison of error measures for different modifications of a human generated ground truth (a). In experiment "grow", the labels of the ground truth have been dilated by 10nm (b). For the "split" and "merge" experiments, ten random locations have been chosen where the ground truth neurons have been manually split or merged, respectively. An example split of a neuron (c) is shown in (d). Results are shown in (e). RI is the ratio of agreeing label pairs over all label pairs (higher is better), VOI is in bits (lower is better), and TED counts the errors as splits (S), merges (M), false positives (FP), and false negatives (FN). Both RI and VOI assign better scores to the split and merge experiments then to the grow experiment. The TED boundary shift tolerance was set to 20nm. With this setup, the TED counts only the true morphological errors (columns S and M).

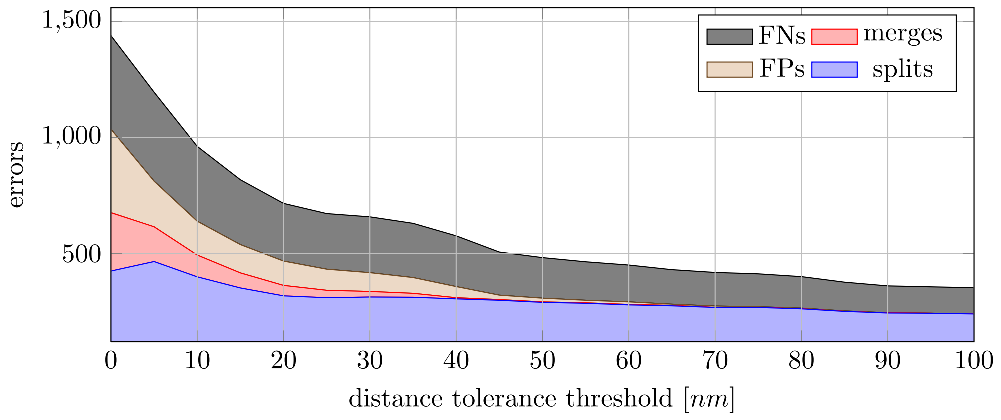


Figure 6.6: The tolerant edit distance (TED) on an automatically generated reconstruction as a function of the tolerated boundary shift. It can be seen that most of the errors occur within the range of about $40nm$. Depending on the biological needs, those errors might be tolerable.

in Section 6.5 is an example of such an error measure. But with our formulation it is also possible to count errors as deviations from non-volumetric skeletons in the ground truth. This is of special interest, since this kind of ground truth can quickly be produced and thus allows the evaluation of our algorithms on large volumes [85].

A consequence of the explicitness of the TED is that errors can be localized in the volume. This is an important feedback for the design of better neuron reconstruction approaches or the refinement of the tolerance criteria. It also eases the important communication with neuroscientists. Instead of having to report a number in bits (VOI) or pairs of agreeing pixels (RI), we can directly answer how many relevant errors we made, where they are, and what they look like.

Counting errors directly also allows us to provide time-to-fix estimates by assigning each error type an expected time needed to repair it. Although this seems very useful, we have to note that without a list of low confidence decisions made by an automatic reconstruction approach, a user would spend more time trying to find the errors in an automatically reconstructed volume than on fixing them. This highlights a very important need in our field: Unless the reconstruction approaches are perfect, we need a confidence measure that prioritizes locations that a user should check for errors. Combined with an explicit error measure like the TED, this would allow us to compute the expected time needed to fix all errors – a value that undoubtedly would be of great use. Minimizing this value should be the real objective of automatic neuron reconstruction approaches.

An interesting direction for future work is the question how the TED can be used to train neuron reconstruction approaches. The structured learning framework (see Chapter 4) provides a powerful tool to train on almost arbitrary cost functions and has already been used successfully to train on VOI [64]. For the structured learning, we have to repeatedly find a reconstruction that has in some sense the highest cost. Although the minimization that is needed to compute the TED can be carried out efficiently, the situation changes when we have to optimize over all possible reconstructions at the same time. It seems necessary to approximate the TED in those cases, either by first- or second-order approximations or by using a lower bound that can be obtained by ignoring all split and merge contributions of regions that are allowed to be relabeled.

Appendix A

Discrete Energy Models

A.1 Conversion to Linear Constraint Binary Model

We assume that a discrete energy model is given as a factor graph $G = (V, F, E)$ with factor nodes F and variable nodes denoted by integers $V = \{1, \ldots, n\}$ with discrete domains $D = \{D_1, \ldots, D_n\}$. Variable nodes and factor nodes are connected by edges $E \subseteq V \times F$ to form a bipartite graph. We write V_f for the strictly ordered set of all variable nodes connected to factor node f, and F_i for all factor nodes connected to variable node i. An assignment of variable nodes to discrete values is represented by a vector $\mathbf{y} = (y_1, \ldots, y_n)$ where y_i represents the value of variable node i. We write $\mathcal{Y} = D_1 \times \ldots \times D_n$ to denote the set of all possible assignments. Similarly, we write $\mathbf{y}_f = (y_i : i \in V_f)$ for the partial assignment of variable nodes connected to a factor node f and $\mathcal{Y}_f$ for its domain. As a shorthand, we write $\mathbf{y}(i)$ and $\mathbf{y}_f(i)$ to refer to the value of variable i under assignments $\mathbf{y}$ or $\mathbf{y}_f$, respectively. A partial assignment $\mathbf{y}_f$ is said to be compatible with an assignment $\mathbf{y}$, denoted by $\mathbf{y}_f \subseteq \mathbf{y}$, if and only if $\mathbf{y}_f(i) = \mathbf{y}(i), \forall i \in V_f$. To each factor $f \in F$ we assign a real valued function $\phi_f : \mathcal{Y}_f \mapsto \mathbb{R}$ that depends on the partial assignment $\mathbf{y}_f \in \mathcal{Y}_f$. Usually, these functions correspond to some features extracted from observations that we do not model explicitly, here. We can think of the functions ϕ_f to be the logarithms of the true factor functions that ought to be multiplied. With this trick, we can write the logarithm of the product encoded by the factor graph as the sum

$$E(\mathbf{y}) = \sum_{f \in F} \phi_f(\mathbf{y}_f), \tag{A.1}$$

which we refer to as *energy* of $\mathbf{y}$.

Each discrete factor graph model defined this way can be transformed into an equivalent constraint binary model in three steps: (1) introduce binary indicator variables for each possible assignment $\mathbf{y}_f$ of the variables nodes V_f of each factor $f \in F$, (2) add exclusivity constraints to ensure that at most one of the indicators of a factor is selected at a time, and (3) add consistency constraints to ensure that the indicator variables of different factors agree on the assignment of shared variables.

Let $\mathbb{I} = (\mathbb{I}_{f,\mathbf{y}_f} : f \in F)$ be binary indicator variables for each particular assignment $\mathbf{y}_f$ of the variable nodes V_f used by any factor f, *i.e.*, $\mathbb{I}_{f,\mathbf{y}_f} = 1$ if and only if $\mathbf{y}_f \subseteq \mathbf{y}$. Let us further define constants $\phi_{f,\mathbf{y}_f} := \phi_f(\mathbf{y}_f)$. Eq. (A.1) can now be rewritten as

$$E(\mathbf{y}) = \sum_{f \in F} \sum_{\mathbf{y}_f} \phi_{f,\mathbf{y}_f} \mathbb{I}_{f,\mathbf{y}_f}. \tag{A.2}$$

To reformulate this energy as a function on $\mathbb{I}$, we have to introduce two sets of constraints:

$$\sum_{\mathbf{y}_f} \mathbb{I}_{f,\mathbf{y}_f} = 1 \qquad \forall f \in F \tag{A.3}$$

$$\sum_{\mathbf{y}_f : \mathbf{y}_f(i)=z} \mathbb{I}_{f,\mathbf{y}_f} - \sum_{\mathbf{y}_g : \mathbf{y}_g(i)=z} \mathbb{I}_{g,\mathbf{y}_g} = 0 \qquad \forall i \in V : \forall f, g \in F_i : \forall z \in D_i. \tag{A.4}$$

The first set ensures that at most one indicator variable for each $\mathbf{y}_f$ is selected at a time. The second set of constraints ensures mutual agreement on assignments of variables that are shared between any pair of factors f and g. Let $\mathcal{I}$ be the set of all $\mathbb{I}$ that fulfill Eq. (A.3) and Eq. (A.4). It can now be seen that the original energy Eq. (A.1) is equivalent to

$$E(\mathbb{I}) = \sum_{f\in F} \sum_{\mathbf{y}_f} \phi_{f,\mathbf{y}_f} \mathbb{I}_{f,\mathbf{y}_f} \tag{A.5}$$

for all $\mathbb{I} \in \mathcal{I}$. In particular,

$$\min_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{y}) = \min_{\mathbb{I}\in\mathcal{I}} E(\mathbb{I}). \tag{A.6}$$

A.2 Feature Linearization

We are considering a linearly parameterized discrete energy model, where we assume that each factor function ϕ_f can be scaled by a real valued weight w_f. More formally, we extend the definition of a discrete energy given in Eq. (A.1) by a weight vector $\boldsymbol{w}$ as follows:

$$E(\boldsymbol{w}, \mathbf{y}) = \langle \boldsymbol{w}, \phi(\mathbf{y}) \rangle = \sum_{f\in F} w_f \phi_f(\mathbf{y}_f). \tag{A.7}$$

We want to show that this energy can be rewritten to yield a formulation that is linear in some discrete variables $\mathbf{y}'$ that replace the original variables $\mathbf{y}$, *i.e.*,

$$E(\mathbf{y}) = \langle \boldsymbol{w}, \phi(\mathbf{y}) \rangle = \langle \boldsymbol{w}, \boldsymbol{\phi}'\mathbf{y}' \rangle = \langle \boldsymbol{w}\boldsymbol{\phi}', \mathbf{y}' \rangle . \tag{A.8}$$

For that, we first transform Eq. (A.7) into linear constraint binary form, as shown in Appendix A.1, to obtain an equivalent energy formulation $E(\boldsymbol{w}, \mathbb{I})$ on indicator variables $\mathbb{I}$.

$$E(\boldsymbol{w}, \mathbb{I}) = \sum_{f\in F} \sum_{\mathbf{y}_f} (w_f \phi_{f,\mathbf{y}_f}) \mathbb{I}_{f,\mathbf{y}_f}. \tag{A.9}$$

We further introduce a *feature matrix* $\boldsymbol{\phi}'$, which is a block diagonal matrix with $|F|$ rows and $|\mathbb{I}|$ columns. We index the rows of this matrix with $f \in F$ and the columns with tuples $(f, \mathbf{y}_f)$ to correspond to the indicator variable that they represent. By defining

$$\boldsymbol{\phi}'_{f',(f,\mathbf{y}_f)} = \begin{cases} \phi_{f,\mathbf{y}f} & \text{if} f' = f \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

and $\mathbf{y}' = \mathbb{I}$, we see that

$$E(\boldsymbol{w}, \mathbb{I}) = \langle \boldsymbol{w}\boldsymbol{\phi}', \mathbf{y}' \rangle . \tag{A.11}$$

Appendix B

Problem Decomposition

B.1 Equivalence of P1 and P2

We want to show that P1 (Definition 5.2 on page 51) is equal to P2 (Definition 5.2.5 on page 53).

Proof: We show equivalence of the energy functions for P1 and P2. The indicator function $\mathbb{1}_\alpha$ returns one if α is true, zero otherwise.

$$\begin{aligned}
\sum_{f \in F} \theta_f(\mathbf{x}_f) &= \sum_{f \in F} \theta_f(\mathbf{x}_f) \sum_{r \in R_f} w_f^r && \text{(B.1)} \\
&= \sum_{f \in F} \sum_{r \in R_f} w_f^r \theta_f(\mathbf{x}_f) && \text{(B.2)} \\
&= \sum_{f \in F} \sum_{r \in R} \mathbb{1}_{\{f \in r\}} \theta_f^r(\mathbf{x}_f) && \text{(B.3)} \\
&= \sum_{r \in R} \sum_{f \in F} \mathbb{1}_{\{f \in r\}} \theta_f^r(\mathbf{x}_f) && \text{(B.4)} \\
&= \sum_{r \in R} \sum_{f \in r} \theta_f^r(\mathbf{x}_f) && \text{(B.5)}
\end{aligned}$$

$\blacksquare$

B.2 Equivalence of P2 and P3

We want to show that P2 (Definition 5.2.5 on page 53) is equal to P3 (Definition 5.2.6 on page 53).

Proof: Let $\mathcal{X}$ be all possible assignments of the variables of P2, and $\mathcal{X}^R$ all possible assignments of the variables of P3. Let us further denote all assignments $x^R \in \mathcal{X}^R$ that satisfy the constraints of P3 by $\mathcal{X}^{R*}$. These constraints ensure that there is a bijective mapping $m : \mathcal{X}^{R*} \mapsto \mathcal{X}$ between valid $\mathbf{x}^R$ and $\mathbf{x}$:

$$m(\mathbf{x}^R) = \{x_v = x_v^r : v \in V, r \in R_v\}. \tag{B.6}$$

P3 can now be rewritten as

$$\begin{aligned}
\min_{\mathbf{x}^R \in \mathcal{X}^{R*}} \sum_{r \in R} \sum_{f \in r} \theta_f^r(\mathbf{x}_f^r) &= \min_{\mathbf{x} \in m[\mathcal{X}^{R*}]} \sum_{r \in R} \sum_{f \in r} \theta_f(\mathbf{x}_f) && \text{(B.7)} \\
&= \min_{\mathbf{x} \in \mathcal{X}} \sum_{r \in R} \sum_{f \in r} \theta_f(\mathbf{x}_f) && \text{(B.8)}
\end{aligned}$$

which shows equivalence between P2 and P3. $\blacksquare$

B.3 Lower Bound

We want to show that $L(\boldsymbol{\lambda})$ (Eq. (5.9) on page 55) is a lower bound on the energy defined in P1 (Definition 5.2 on page 51) and that any consistent solution $\mathbf{x}^R$ obtained by maximizing $L(\boldsymbol{\lambda})$ is an optimal solution to P1.

Proof: First, we show that $L(\boldsymbol{\lambda})$ is a lower bound on the minimal energy of P1. It follows that any consistent solution we obtain by maximizing $L(\boldsymbol{\lambda})$ is a solution to P1.

From Lemma 5.2.1 and Lemma 5.2.2 we know that P3 is equal to P1. Minimizing the energy of P3 subject to the consistency constraints is equivalent to the following optimization:

$$\min_{\mathbf{x}^R} \max_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}, \mathbf{x}^R). \tag{B.9}$$

This can easily be verified by looking at the definition of $L(\boldsymbol{\lambda}, \mathbf{x}^R)$: Whenever one of the constraints in P3 does not hold, the corresponding Lagrangian term in Eq. (5.8) becomes non-zero. This allows the maximization over the $\boldsymbol{\lambda}$ to increase its value to infinity, such that the minimization over $\mathbf{x}^R$ would not pick this configuration. If, however, the constraints are fulfilled, the Lagrangian terms drop out and reveal the same objective as the one in P3. Hence, the maximization in Eq. (B.9) is either the true energy of an assignment $\mathbf{x}^R$, if the consistency constraints are fulfilled; or infinite, if they are not fulfilled. We obtain a lower bound on Eq. (B.9) by exchanging the minimization and maximization:

$$\min_{\mathbf{x}^R} \max_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}, \mathbf{x}^R) \geq \max_{\boldsymbol{\lambda}} \underbrace{\min_{\mathbf{x}^R} L(\boldsymbol{\lambda}, \mathbf{x}^R)}_{L(\boldsymbol{\lambda})}. \tag{B.10}$$

We see that the maximal value of $L(\boldsymbol{\lambda})$ is smaller or equal to the minimal energy of P1. Hence, we can conclude

$$L(\boldsymbol{\lambda}) \leq e^* \quad \forall \boldsymbol{\lambda}. \tag{B.11}$$

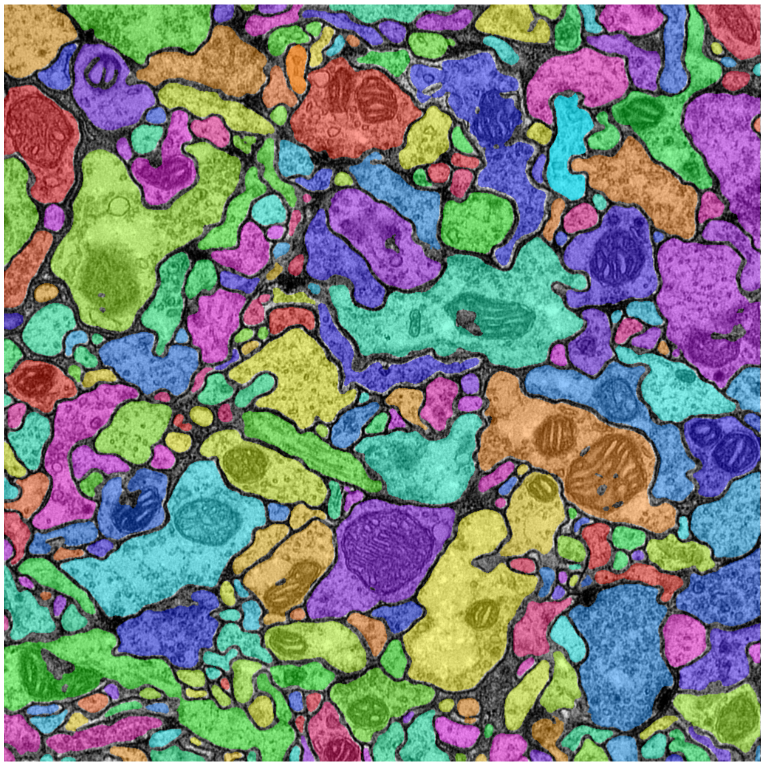
Knowing that $L(\boldsymbol{\lambda})$ is a lower bound on e^*, we can further conclude that any consistent solution $\mathbf{x}^{R*}$ at the maximum of $L(\boldsymbol{\lambda})$ can be mapped to the solution $\mathbf{x}^*$ of P1. ∎

Appendix C

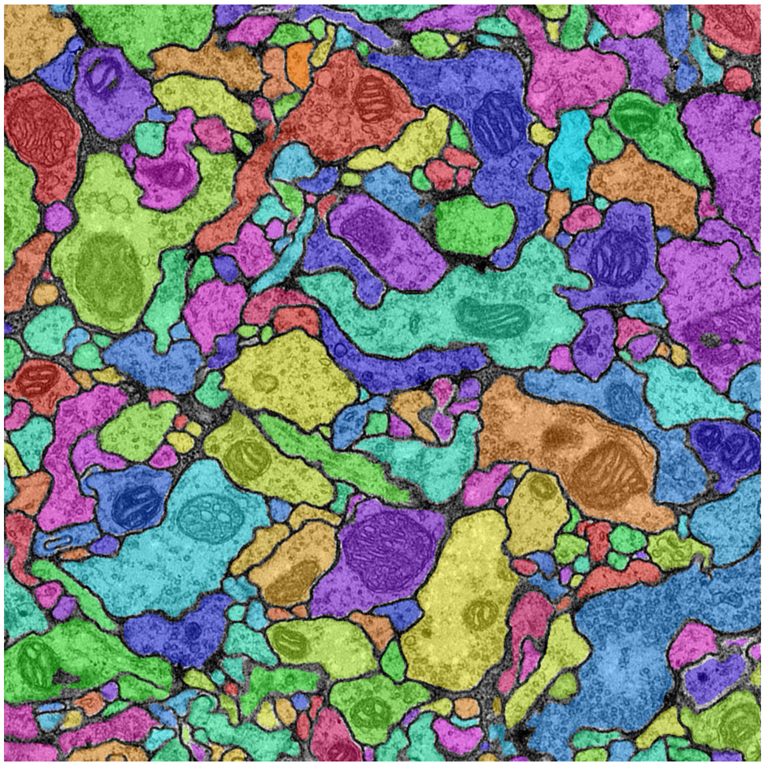
Reconstruction Results

C.1 Candidates from Component Tree

The following images show the automatic reconstruction result after structured learning of the assignment costs for 2D neuron candidates from component trees on the last 10 sections of the evaluation stack. Details about the setup can be found in Section 4.5.

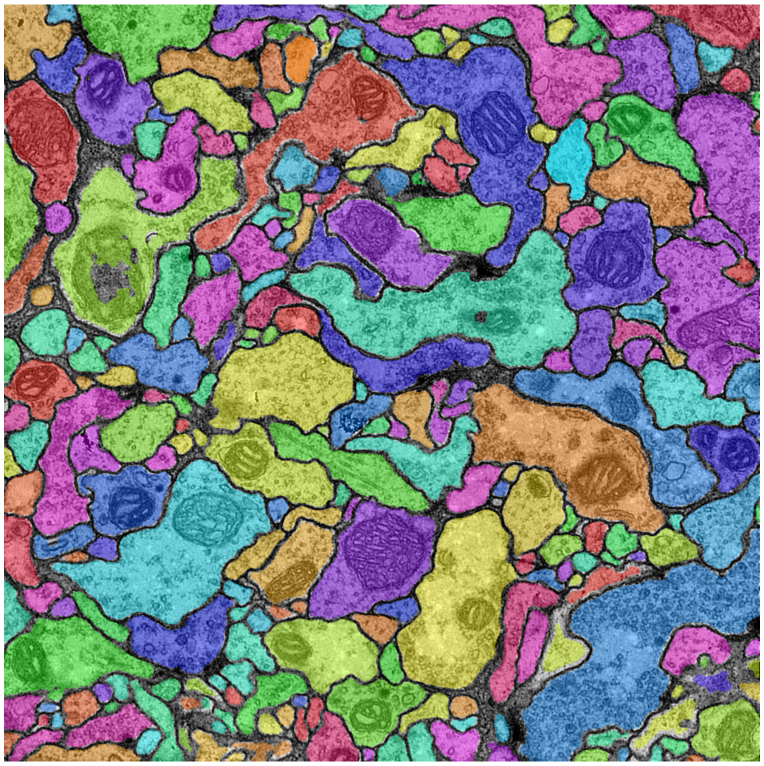


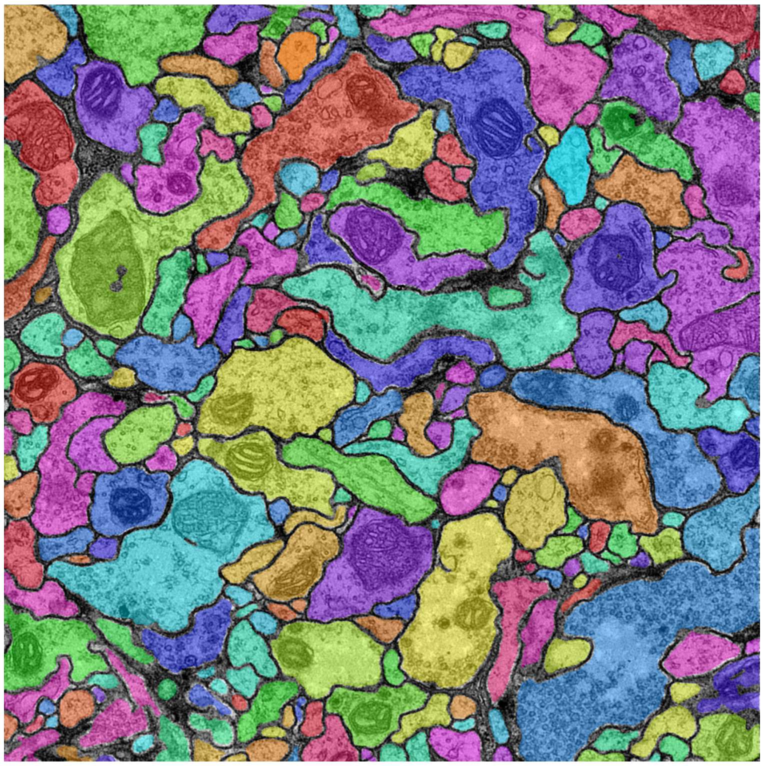
$z = 10$



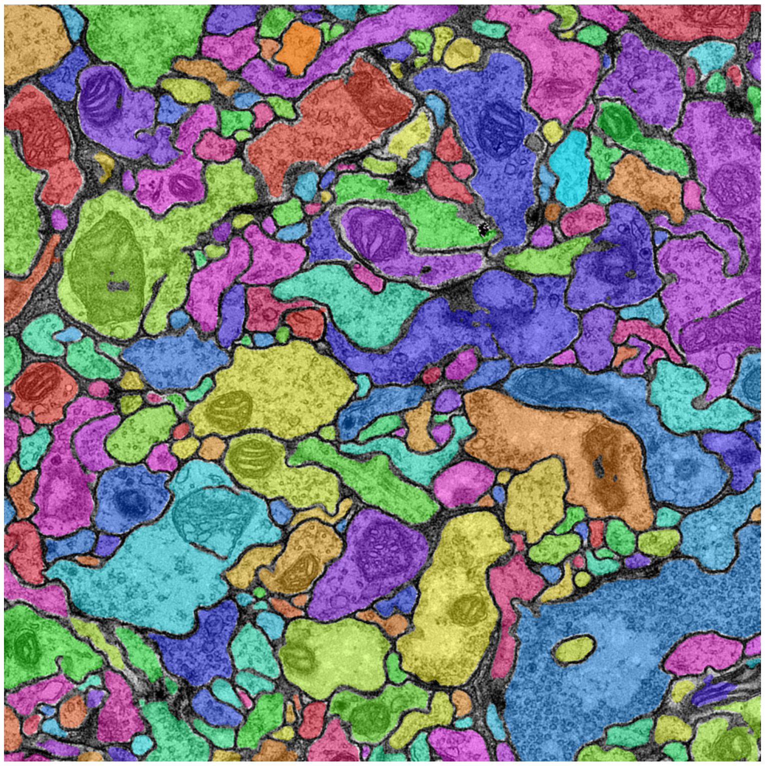
$z = 11$

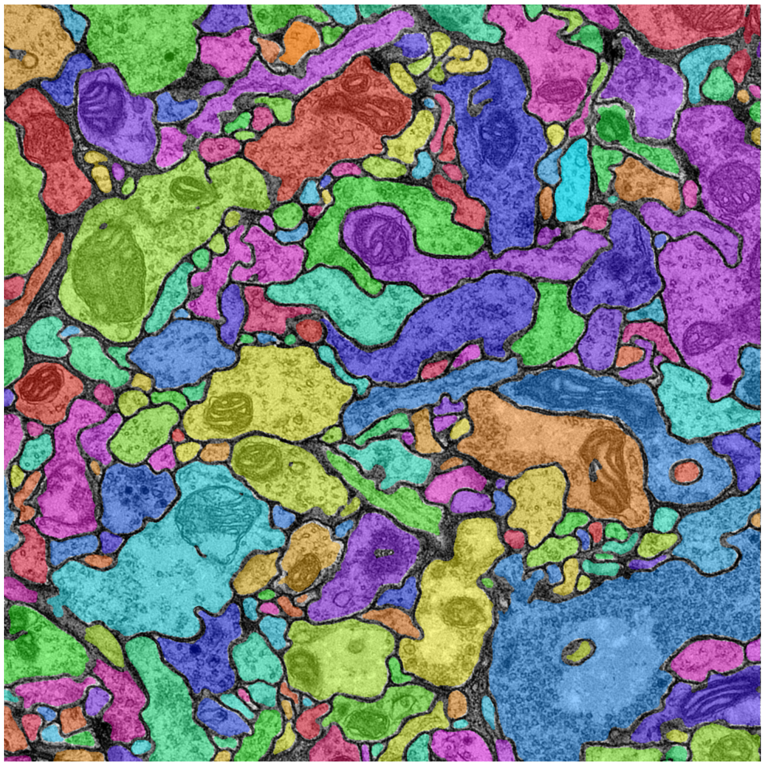
reconstruction result after structured learning,
2D neuron candidates from component trees


$z = 12$

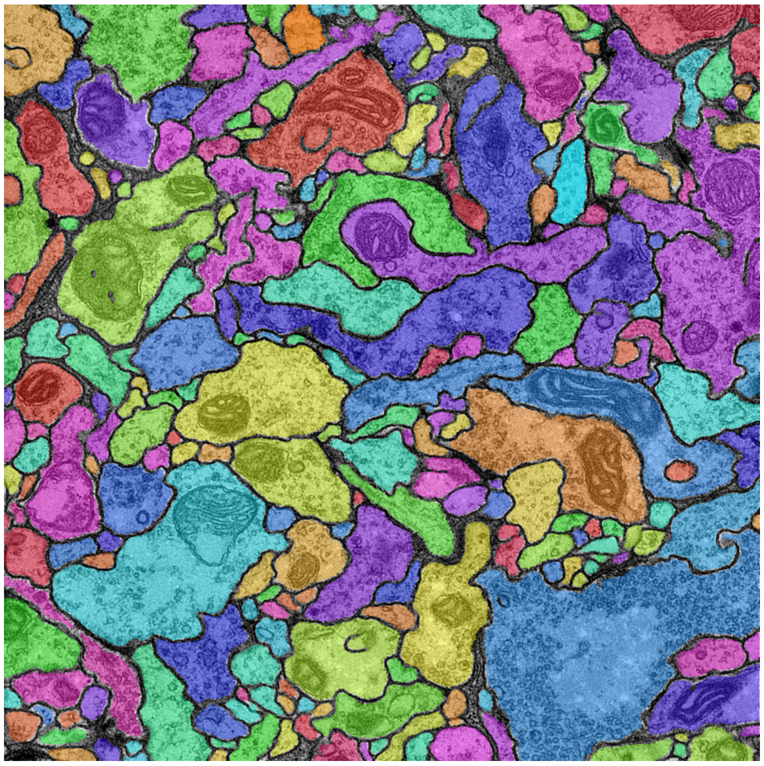

$z = 13$

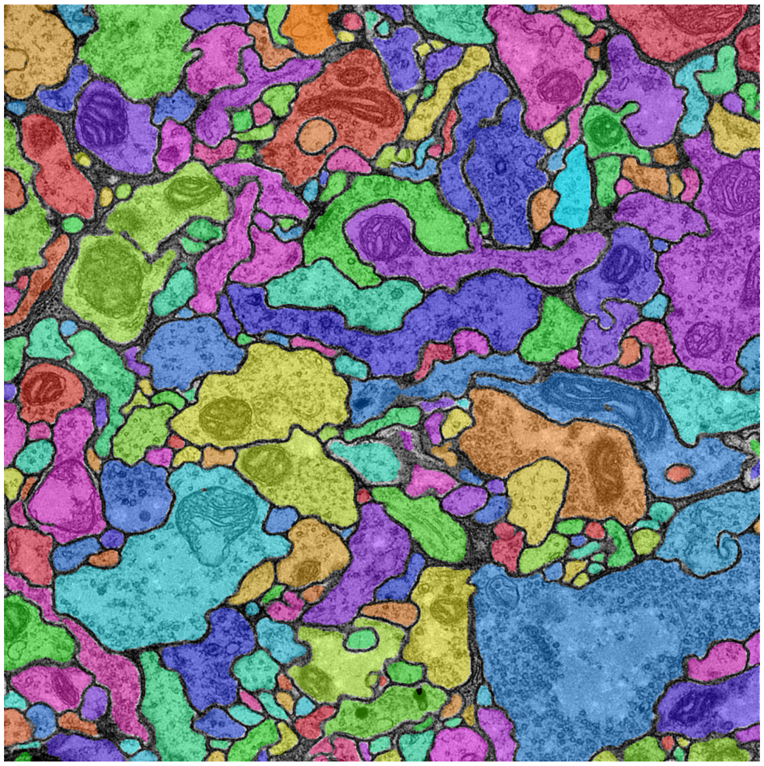
reconstruction result after structured learning,
2D neuron candidates from component trees


$z = 14$

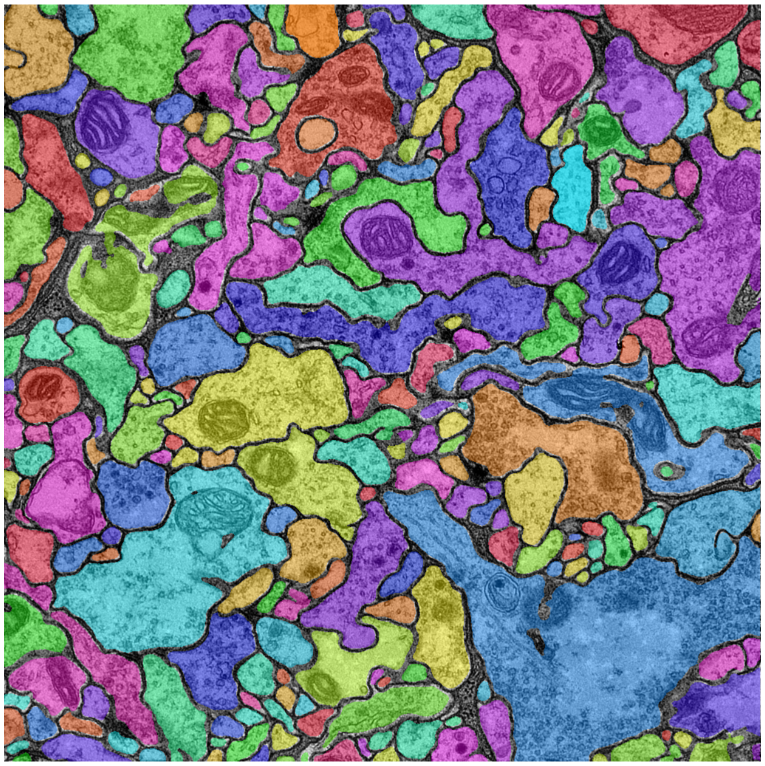

$z = 15$

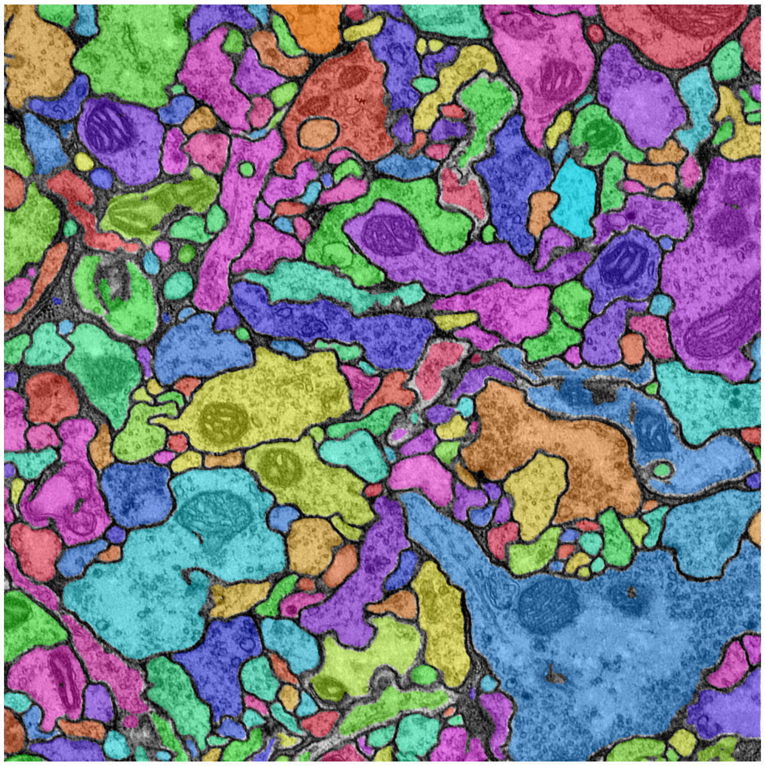
reconstruction result after structured learning,
2D neuron candidates from component trees


$z = 16$


$z = 17$

reconstruction result after structured learning,
2D neuron candidates from component trees

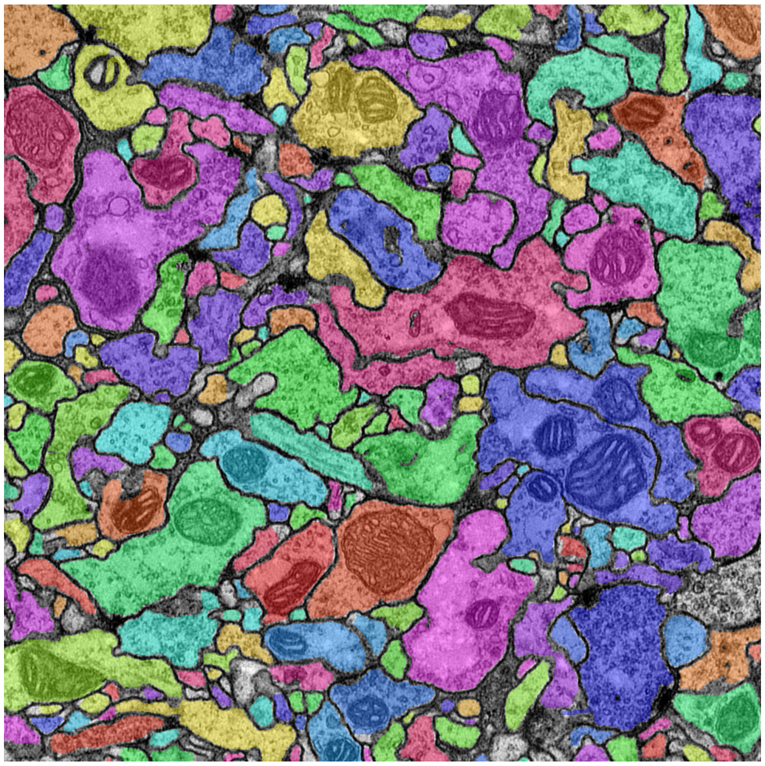

$z = 18$

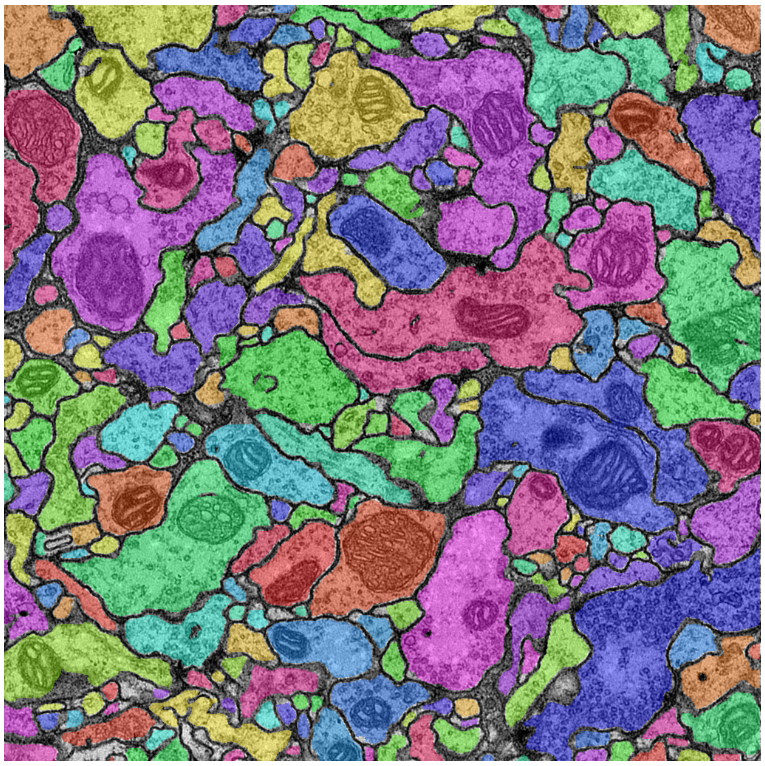

$z = 19$

reconstruction result after structured learning,
2D neuron candidates from component trees

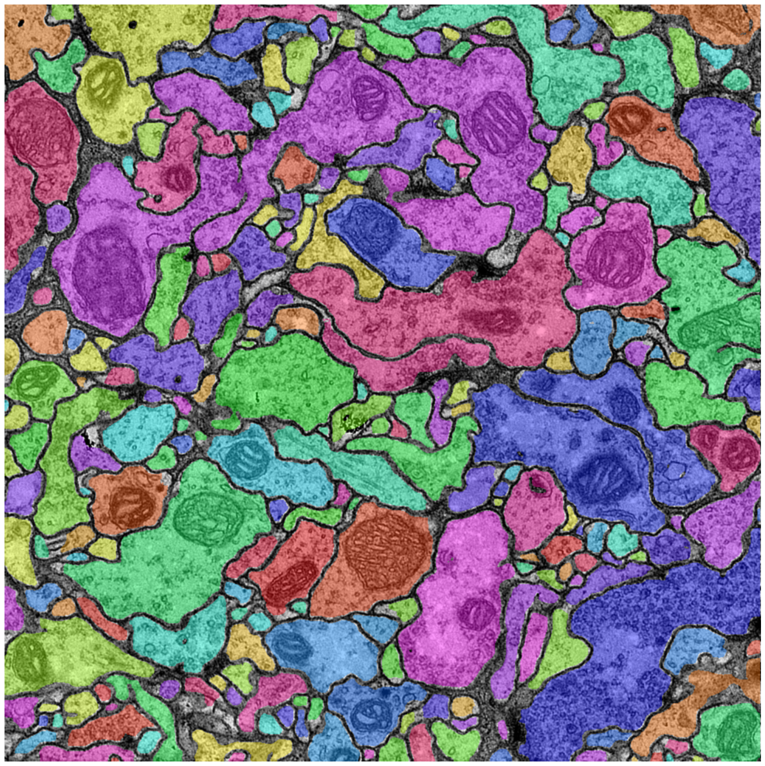
C.2 Candidates from CRF

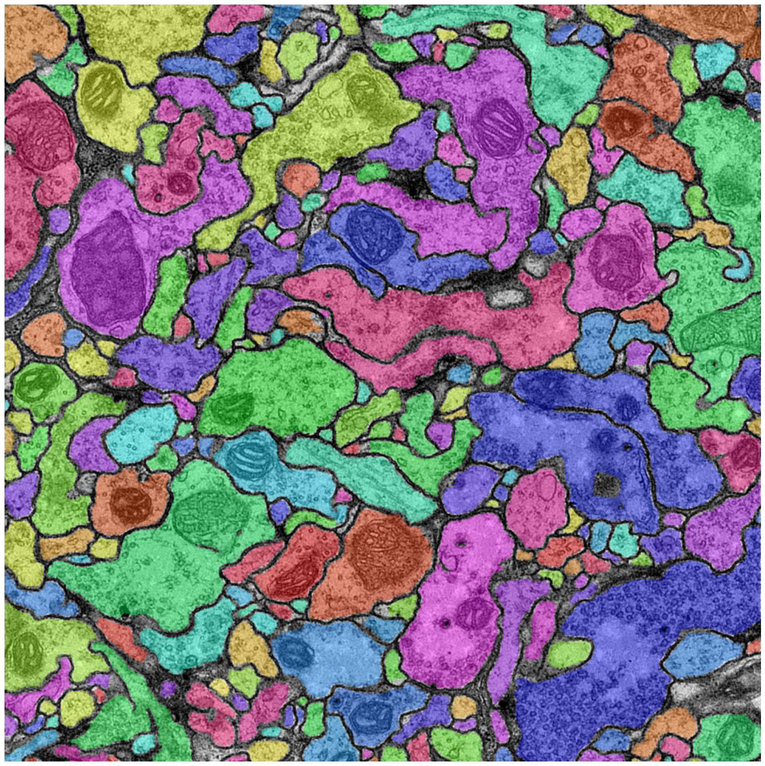
The following images show the automatic reconstruction result after structured learning of the assignment costs for 2D neuron candidates from the CRF described in Chapter 3 on the last 10 sections of the evaluation stack. Details about the setup can be found in Section 4.5.


$z = 10$

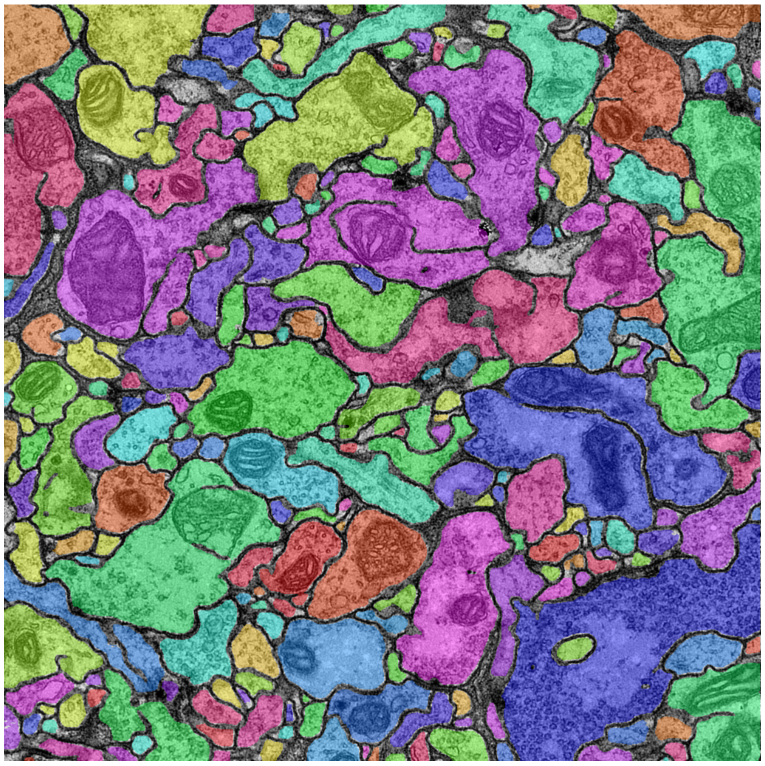

$z = 11$

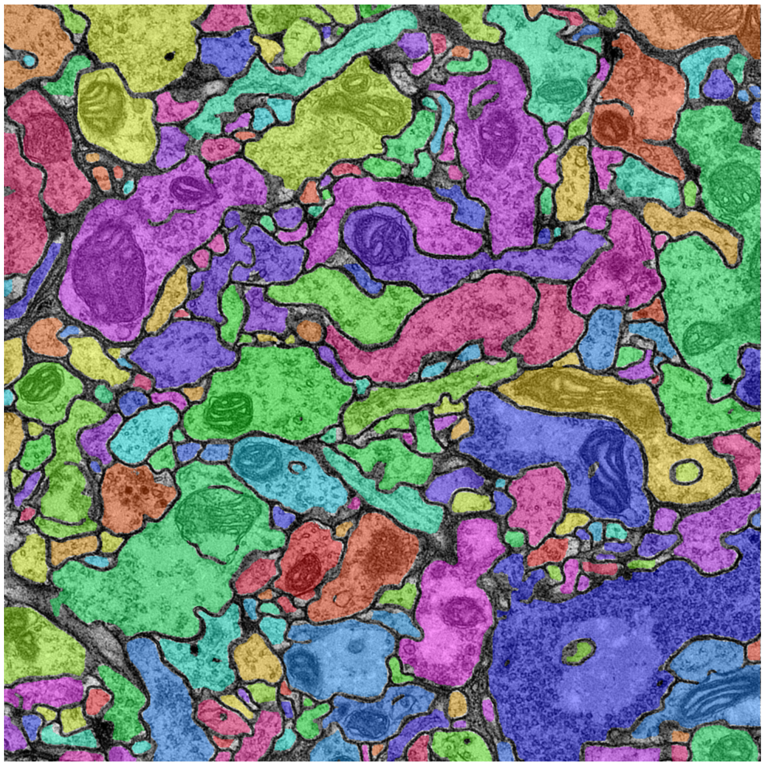
reconstruction result after structured learning,
2D neuron candidates from CRF


$z = 12$

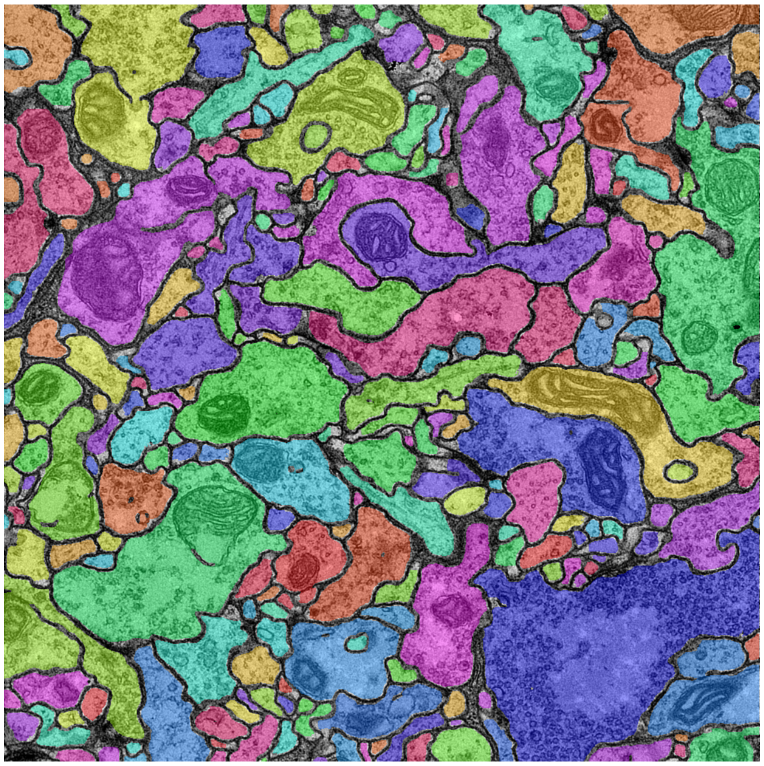

$z = 13$

reconstruction result after structured learning,
2D neuron candidates from CRF

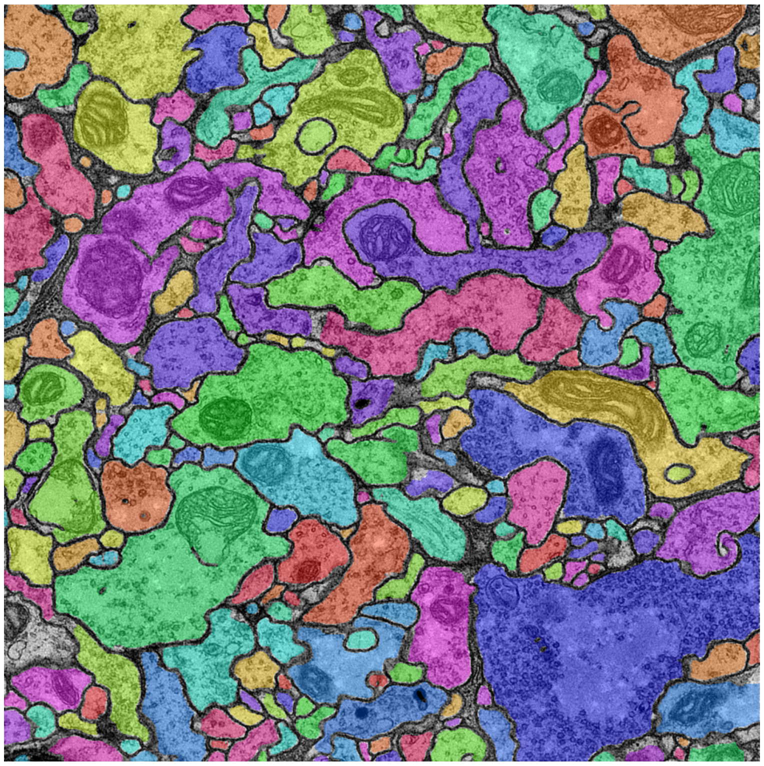

$z = 14$


$z = 15$

reconstruction result after structured learning,
2D neuron candidates from CRF

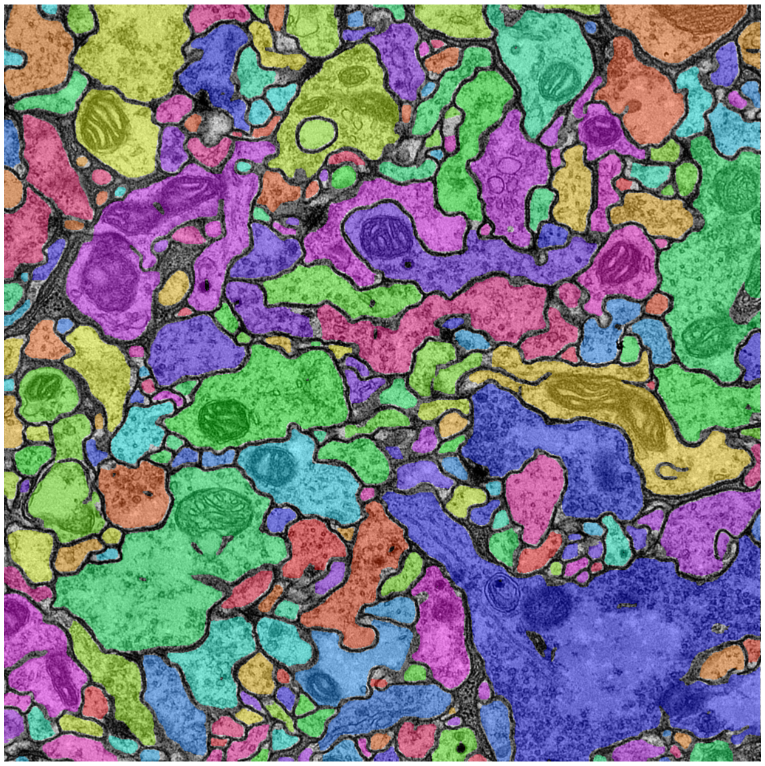


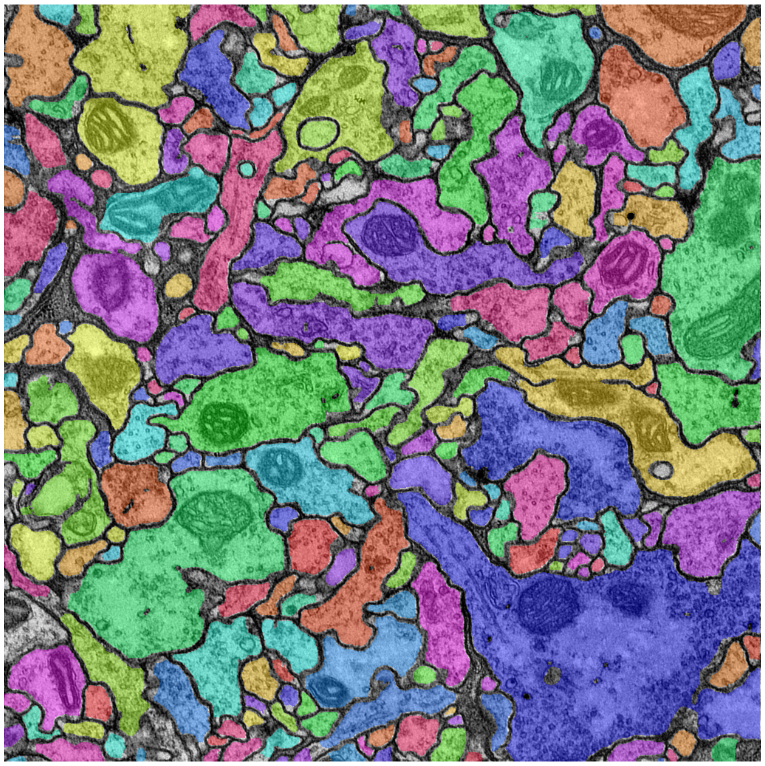
$z = 16$



$z = 17$

reconstruction result after structured learning,
2D neuron candidates from CRF


$z = 18$


$z = 19$

reconstruction result after structured learning,
2D neuron candidates from CRF

Bibliography

[1] 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 (2012). ↑8

[2] Bjoern Andres, Thorben Kröger, Kevin L. Briggman, Winfried Denk, Natalya Korogod, Graham Knott, Ullrich Köthe, and Fred A. Hamprecht, *Globally Optimal Closed-surface Segmentation for Connectomics*, ECCV, 2012. ↑8, 12

[3] Pablo Arbelez, Michael Maire, Charless Fowlkes, and Jitendra Malik, *From Contours to Regions: An Empirical Evaluation*, CVPR, 2009. ↑68, 69

[4] Dhruv Batra, Sebastian Nowozin, and Pushmeet Kohli, *Tighter Relaxations for MAP-MRF Inference: A Local Primal-Dual Gap based Separation Algorithm.*, AISTATS, 2011, pp. 146–154. ↑50, 58

[5] Carlos Joaquin Becker, Karim Ali, Graham Knott, and Pascal Fua, *Learning Context Cues for Synapse Segmentation*, IEEE Transactions on Medical Imaging (2013). ↑8

[6] Tom Binzegger, Rodney J. Douglas1, and Kevan A. C. Martin, *A Quantitative Map of the Circuit of Cat Primary Visual Cortex*, Journal of Neuroscience **24(39)** (2004), 8441–8453. ↑7

[7] DD Bock, WC Lee, AM Kerlin, ML Andermann, G Hood, AW Wetzel, S Yurgenson, ER Soucy, HS Kim, and RC Reid, *Network anatomy and in vivo physiology of visual cortical neurons*, Nature **47** (2011), no. 7337, 177–82. ↑2, 4, 5

[8] Jennifer N Bourne and Kristen M Harris, *Nanoscale analysis of structural synaptic plasticity.*, Current opinion in neurobiology (November 2011). ↑67

[9] 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** (2004sept.), no. 9, 1124 –1137. ↑13

[10] Steve Branson, Oscar Beijbom, and Serge Belongie, *Efficient Large-Scale Structured Learning*, CVPR, 2013. ↑38

[11] Leo Breiman, *Random Forests*, Machine Learning **45** (2001), 5–32. 10.1023/A:1010933404324. ↑8, 13, 23, 37, 45

[12] Kevin L Briggman and Winfried Denk, *Towards neural circuit reconstruction with volume electron microscopy techniques*, Current Opinion in Neurobiology **16** (2006), no. 5, 562 –570. ↑2, 5

[13] Kevin L. Briggman, Moritz Helmstaedter, and Winfried Denk, *Wiring Specificity in the Direction-Selectivity Circuit of the Retina*, Nature **471** (2011), 183–188. ↑2

[14] Albert Cardona, *Towards Semi-Automatic Reconstruction of Neural Circuits*, Neuroinformatics **11** (2012), 31–33. ↑68

[15] 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** (2010), no. 10, e100050. ↑2, 4, 5, 11, 20

[16] 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** (2012). ↑68

[17] D. Chklovskii, S. Vitaladevuni, and L. Scheffer, *Semi-Automated Reconstruction of Neural Circuits Using Electron Microscopy*, Current Opinion in Neurobiology **20(5)** (2010), 667–675. ↑2

[18] Dan Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber, *Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images*, NIPS, 2012, pp. 2843–2851. ↑28, 31, 37, 45

[19] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber, *Multi-column Deep Neural Networks for Image Classification*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012. ↑1

[20] Corinna Cortes and Vladimir Vapnik, *Support-Vector Networks*, Machine Learning **20** (1995), 273–297. ↑45

[21] W. Denk and H. Horstmann, *Serial Block-Face Scanning Electron Microscopy to Reconstruct Three-Dimensional Tissue Nanostructure*, PLoS Biology **2(11)** (2004). ↑4

[22] Winfried Denk, Kevin L Briggman, and Moritz Helmstaedter, *Structural neurobiology: missing link to a mechanistic understanding of neural computation*, Nature reviews Neuroscience (February 2012). ↑2, 67

[23] Jan Funke, *Parallel Implementation of a Gibbs Sampler on GPU*. http://github.com/funkey/prim. ↑31

[24] ______, *sbmrm - Bundle Method for Structured Risk Minimization*. http://github.com/funkey/sbmrm. ↑40

[25] ______, *SOPNET - C++ Neuron Reconstruction Pipeline*. http://github.com/funkey/sopnet. ↑12

[26] Jan Funke, Bjoern Andres, Fred A. Hamprecht, Albert Cardona, and Matthew Cook, *Efficient Automatic 3D-Reconstruction of Branching Neurons from EM Data*, CVPR, 2012, pp. 1004–1011. ↑1, 11, 28, 29, 32, 71

[27] Jan Funke, Julien Martel, Stephan Gerhard, Bjoern Andres, Dan C. Ciresan, Alessandro Giusti, Luca M. Gambardella, Juergen Schmidhuber, Hanspeter Pfister, Albert Cardona, and Matthew Cook, *Candidate Sampling for Neuron Reconstruction from Anisotropic Electron Microscopy Volumes*, MICCAI, 2014, pp. 17–24. ↑27

[28] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li, *A survey of graph edit distance*, Pattern Analysis and Applications **13** (2010), no. 1, 113–129 (English). ↑75

[29] S. Geman and D. Geman, *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6 (6)** (1984), 721–741. ↑27, 31

[30] Arthur M. Geoffrion, *Lagrangian Relaxation for Integer Programming*, Mathematical Programming Study **2** (1974), 82–114. ↑50, 53

[31] Stephan Gerhard, Jan Funke, Julien Martel, Albert Cardona, and Richard Fetter, *Segmented anisotropic ssTEM dataset of neural tissue*, 2013. http://dx.doi.org/10.6084/m9.figshare.856713. ↑31, 45, 63, 76

[32] Stephan Gerhard, Mark Longair, Stephan Saalfeld, Richard Fetter, Tom Kazimiers, Frank Midgley, Casey Schneider-Mizell, and Albert Cardona, *Collaborative online reconstruction and analysis of neural circuits.*, in preparation (2014). ↑76

[33] Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin, *Parallel Gibbs Sampling: From Colored Fields to Thin Junction Trees*, In Artificial Intelligence and Statistics (AISTATS), 2011May. ↑31

[34] Patric Hagmann, *From Diffusion MRI to Brain Connectomics*, Ph.D. Thesis, 2005. ↑75

[35] K. J. Hayworth, N. Kasthuri, R. Schalek, and J. W. Lichtman, *Automating the Collection of Ultrathin Serial Sections for Large Volume TEM Reconstructions*, Microscopy and Microanalysis **12** (2006), 86–87. ↑4

[36] Moritz Helmstaedter, Kevin Briggman, and Winfried Denk, *High-accuracy neurite reconstruction for high-throughput neuroanatomy*, Nat Neurosci **14** (2011), no. 8. ↑2

[37] Moritz Helmstaedter, Kevin L. Briggman, and Winfried Denk, *3D Structural Imaging of the Brain with Photons and Electrons.*, Current Opinion in Neurobiology **18** (2008), no. 6, 633–641. ↑2, 11

[38] Gary B. Huang and Viren Jain, *Deep and Wide Multiscale Recursive Networks for Robust Image Labeling*, ICLR, 2014. ↑8

[39] Vignesh Jagadeesh, James Anderson, Bryan Jones, Robert Marc, Steven Fisher, and B.S. Manjunath, *Synapse classification and localization in Electron Micrographs*, Pattern Recognition Letters **43** (2014), no. 0, 17 –24. {ICPR2012} Awarded Papers. ↑8

[40] Viren Jain, Benjamin Bollmann, Mark Richardson, Daniel R. Berger, Moritz N. Helmstaedter, Kevin L. Briggman, Winfried Denk, Jared B. Bowden, John M. Mendenhall, Wickliffe C. Abraham, Kristen M. Harris, Narayanan Kasthuri, Ken J. Hayworth, Richard Schalek, Juan Carlos Tapia, Jeff W. Lichtman, and H. Sebastian Seung, *Boundary Learning by Optimization with Topological Constraints*, CVPR, 2010. ↑8, 12, 70

[41] Viren Jain, Joseph F. Murray, Fabian Roth, Srinivas Turaga, Valentin P. Zhigulin, Kevin L. Briggman, Moritz Helmstaedter, Winfried Denk, and H. Sebastian Seung, *Supervised Learning of Image Restoration with Convolutional Networks*, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2007, pp. 1–8. ↑8, 12

[42] Viren Jain, H. Sebastian Seung, and Srinivas C. Turaga, *Machines that learn to segment images: a crucial technology for connectomics*, Current Opinion in Neurobiology (2010). ↑8, 12

[43] Viren Jain, Srinivas C. Turaga, Kevin L. Briggman, Moritz N. Helmstaedter, Winfried Denk, and H. Sebastian Seung, *Learning to Agglomerate Superpixel Hierarchies*, Advances in Neural Information Processing Systems, 2011. ↑12, 21

[44] Vladimir Jojic, Stephen Gould, and Daphne Koller, *Accelerated dual decomposition for MAP inference*, ICML, 2010. ↑50

[45] Component Trees for Image Filtering and Segmentation (1997) ↑14, 32

[46] Elizabeth Jurrus, Antonio R.C. 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** (2010), no. 6, 770 –783. ↑12

[47] D. Justice and A. Hero, *A binary linear programming formulation of the graph edit distance*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006Aug), no. 8, 1200–1214. ↑75

[48] J.H. Kappes, B. Savchynskyy, and C. Schnorr, *A bundle approach to efficient MAP-inference by Lagrangian relaxation*, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012June, pp. 1688–1695. ↑50, 56, 60

[49] Jörg H. Kappes, Bjoern Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Jan Lellmann, Nikos Komodakis, and Carsten Rother, *A Comparative Study of Modern Inference Techniques for Discrete Energy Minimization Problem*, CVPR, 2013. ↑52

[50] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister, *Large-Scale Automatic Reconstruction of Neuronal Processes from Electron Microscopy Images*, ArXiv e-prints (2013). ↑8, 28, 32, 50

[51] Verena Kaynig, Thomas Fuchs, and Joachim M. Buhmann, *Geometrical Consistent 3D Tracing of Neuronal Processes in ssTEM Data*, MICCAI 2010, Part II, 2010, pp. 209–216. ↑12, 13, 18

[52] ———, *Neuron Geometry Extraction by Perceptual Grouping in ssTEM Images*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 2902–2909. ↑12, 13, 32

[53] David Kleinfeld, Arjun Bharioke, Pablo Blinder, Davi D. Bock, Kevin L. Briggman, Dmitri B. Chklovskii, Winfried Denk, Moritz Helmstaedter, John P. Kaufhold, Wei-Chung Allen Lee, Hanno S. Meyer, Kristina D. Micheva, Marcel Oberlaender, Steffen Prohaska, R. Clay Reid, Stephen J. Smith, Shinya Takemura, Philbert S. Tsai, and Bert Sakmann, *Large-Scale Automated Histology in the Pursuit of Connectomes*, The Journal of Neuroscience **31** (November 2011), no. 45, 16125–16138 (en). ↑2

[54] 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 Neurscience **28(12)** (2008), 2959–2964. ↑4

[55] Pushmeet Kohli and Philip Torr, *Dynamic Graph Cuts and Their Applications in Computer Vision*, Computer Vision, 2010, pp. 51–108. ↑14

[56] Vladimir Kolmogorov, *Convergent Tree-Reweighted Message Passing for Energy Minimization*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **28** (2006Oct), no. 10, 1568–1583. ↑50

[57] Vladimir Kolmogorov, Yuri Boykov, and Carsten Rother, *Applications of Parametric Maxflow in Computer Vision*, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2007. ↑14

[58] Nikos Komodakis, *Towards More Efficient and Effective LP-Based Algorithms for MRF Optimization*, ECCV, 2010. ↑50

[59] Nikos Komodakis and Nikos Paragios, *Beyond Loose LP-Relaxations: Optimizing MRFs by Repairing Cycles*, Computer Vision ECCV 2008, 2008, pp. 806–820. ↑50, 58

[60] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas, *MRF Energy Minimization and Beyond via Dual Decomposition*, IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011), no. 3, 531–552. ↑50

[61] A. Kreshuk, U. Köthe, E. Pax, D. D. Bock, and F. A. Hamprecht, *Automated Detection of Synapses in Serial Section Transmission Electron Microscopy Image Stacks*, PLoS ONE **9** (2014), 2. ↑8

[62] A. Kreshuk, C. N. Straehle, C. Sommer, U. Köthe, M. Cantoni, G. Knott, and F. A. Hamprecht, *Automated Detection and Segmentation of Synaptic Contacts in Nearly Isotropic Serial Electron Microscopy Images*, PLoS ONE **6 (10)** (2011). ↑8

[63] A. Kreshuk, C. Strähle, C. Sommer, U. Köthe, G. Knott, and F. A. Hamprecht, *Automated Segmentation of Synapses in 3D EM Data*, Eighth IEEE International Symposium on Biomedical Imaging (ISBI). Proceedings, 2011, pp. 220–223. ↑8

[64] Thorben Kröger, Shawn Mikula, Winfried Denk, Ullrich Köthe, and Fred A. Hamprecht, *Learning to Segment Neurons with Non-local Quality Measures*, MICCAI, 2013. ↑8, 79

[65] M.P. Kumar and D. Koller, *Efficiently selecting regions for scene understanding*, Computer Vision and Pattern Recognition, 2010, pp. 3217 –3224. ↑13

[66] R. D. Leapman, *Novel Techniques in Electron Microscopy*, Current Opinion in Neurobiology **14** (2004), 591–598. ↑5

[67] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Intelligent Signal Processing, 2001, pp. 306–351. ↑1

[68] Jeff W. Lichtman and Joshua R. Sanes, *Ome sweet ome: what can the genome tell us about the connectome?*, Current Opinion in Neurobiology **18** (2008), 346–353. ↑75

[69] Ting Liu, Cory Jones, Mojtaba Seyedhosseini, and Tolga Tasdizen, *A modular hierarchical approach to 3D electron microscopy image segmentation*, Journal of Neuroscience Methods **226** (2014), no. 0, 88 –102. ↑8

[70] Michael London and Michael Husser, *Dendritic Computation*, Annual Review of Neuroscience **28** (2005), no. 1, 503–532. ↑67

[71] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision **60** (2004), 91–110. ↑7

[72] 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 **30** (2011), no. 11. ↑8, 12

[73] Aurlien Lucchi, Yunpeng Li, Kevin Smith, and Pascal Fua, *Structured Image Segmentation Using Kernelized Features*, Computer Vision ECCV 2012, 2012, pp. 400–413. ↑8

[74] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics*, ICCV, 2001, pp. 416–423. ↑68

[75] David R. Martin, Charless C. Fowlkes, and Jitendra Malik, *Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues*, PAMI (2004). ↑68

[76] Sebastian Nowozin and Christoph H. Lampert, *Structured Learning and Prediction in Computer Vision*, Foundations and Trends in Computer Vision, 2011. ↑38

[77] 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 (2013). ↑8

[78] Dirk Padfield, Jens Rittscher, and Badrinath Roysam, *Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis*, Medical Image Analysis **In Press, Corrected Proof** (2010). ↑15

[79] Hanchuan Peng, Phuong Chung, Fuhui Long, Lei Qu, Arnim Jenett, Andrew M. Seeds, Eugene W. Myers, and Julie H. Simpson, *BrainAligner: 3D registration atlases of Drosophila brains*, Nature Methods **8** (June 2011), no. 6, 493–498. ↑67

[80] S. Plaza, L. Scheffer, and M. Saunders, *Minimizing Manual Image Segmentation Turn-Around Time for Neuronal Reconstruction by Embracing Uncertainty*, PLoS ONE **7(9)** (2012). ↑2

[81] Stephen M Plaza, Louis K Scheffer, and Dmitri B Chklovskii, *Toward large-scale connectome reconstructions*, Current Opinion in Neurobiology **25** (2014), no. 0, 201 –210. Theoretical and computational neuroscience. ↑4

[82] William M. Rand, *Objective Criteria for the Evaluation of Clustering Methods*, Journal of the Americal Statistical Association **66** (1971), 846–850. ↑69

[83] A. Glenn Richards, H. Burr Steinbach, and Thomas F. Anderson, *Electron microscope studies of squid giant nerve axoplasm*, Journal of Cellular and Comparative Physiology **21** (1943), no. 2, 129–143. ↑7

[84] C. Rother, T. Minka, A. Blake, and V. Kolmogorov, *Cosegmentation of Image Pairs by Histogram Matching - Incorporating a Global Constraint into MRFs*, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, 2006june, pp. 993 –1000. ↑12

[85] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomancák, *CATMAID: Collaborative Annotation Toolkit for Massive Amounts of Image Data*, Bioinformatics **25** (2009). ↑79

[86] ———, *As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets.*, Bioinformatics **26** (2010), 57–63. ↑12

[87] Stephan Saalfeld, Richard D. Fetter, Albert Cardona, and Pavel Tomancák, *Elastic volume reconstruction from series of ultra-thin microscopy sections*, Nature Methods **9** (2012), 717–720. ↑5

[88] M.I. Schlesinger, *Syntactic analysis of two-dimensional visual signals in the presence of noise*, Cybernetics **12** (1976), no. 4, 612–628 (English). ↑57

[89] 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*, Proceedings of the Conference on Medial Image Computing and Computer-Assisted Intervention, 2011. ↑8

[90] Christoph Sommer, Christoph Straehle, Ullrich Koethe, and Fred A. Hamprecht, *"ilastik: Interactive Learning and Segmentation Toolkit"*, 8th IEEE International Symposium on Biomedical Imaging (ISBI) (2011). ↑21

[91] David Sontag, Do Kook Choe, and Yitao Li, *Efficiently Searching for Frustrated Cycles in MAP Inference*, CoRR **abs/1210.4902** (2012). ↑50, 58

[92] David Sontag, Amir Globerson, and Tommi Jaakola, *Introduction to Dual Decomposition for Inference*, 2010. ↑49, 50

[93] Gina E. Sosinsky, John Crum, Ying Z. Jones, Jason Lanman, Benjamin Smarr, Masako Terada, Maryann E. Martone, Thomas J. Deerinck, John E. Johnson, and Mark H. Ellisman, *The combination of chemical fixation procedures with high pressure freezing and freeze substitution preserves highly labile tissue ultrastructure for electron tomography applications*, Journal of Structural Biology **161** (March 2008), no. 3, 359–371. ↑67

[94] John K. Stevens, Thomas L. Davis, Neil Friedman, and Peter Sterling, *A systematic approach to reconstructing microcircuitry by electron microscopy of serial sections*, Brain Research Reviews **2** (1980), no. 13, 265 –293. ↑5, 7

[95] Charles Sutton and Andrew McCallum, *An Introduction to Conditional Random Fields for Relational Learning*, Department of Computer Science University of Massachusetts, 2006. ↑30

[96] 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 A. 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** (August 2013), no. 7461, 175–181. ↑2

[97] Choon H. Teo, S. V. N. Vishwanathan, Alex Smola, and Quoc Le, *Bundle Methods for Regularized Risk Minimization*, Journal of Machine Learning Research **11** (2010), 311–365. ↑39, 40, 56

[98] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer, *Large margin methods for structured and interdependent output variables*, Journal of Machine Learning Research **6** (2005), 1453–1484. ↑8, 38, 39

[99] Srinivas C. Turaga, Joseph F. Murray, Viren Jain, Fabian Roth Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H. Sebastian Seung, *Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation*, Neural Compuation (2010). ↑8, 68

[100] Ranjith Unnikrishnan and Martial Hebert, *Measures of Similarity*, Seventh IEEE Workshop on Applications of Computer Vision, 2005. ↑69

[101] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert, *Toward Objective Evaluation of Image Segmentation Algorithms*, PAMI **29** (2007), 929–944. ↑69

[102] Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller, *Multiple Hypothesis Video Segmentation from Superpixel Flows*, Proceedings of the European Conference on Computer Vision (ECCV), 2010. ↑8, 12

[103] Amelio Vazquez-Reina, Daniel Huang, Michael Gelbart, Jeff Lichtman, Eric Miller, and Hanspeter Pfister, *Segmentation Fusion for Connectomics*, ICCV, 2011. ↑8, 12, 13, 22, 23, 28

[104] Ashok Veeraraghavan, Alex V. Genkin, Shiv Vitaladevuni, Lou Scheffer, Shan Xu, Harald Hess, Richard Fetter, Marco Cantoni, Graham Knott, and Dmitri Chklovskii, *Increasing depth resolution of electron microscopy of neural circuits using sparse tomographic reconstruction*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1767 –1774. ↑12

[105] Shiv Naga Prasad Vitaladevuni and Ronen Basri, *Co-Clustering of Image Segments Using Convex Optimization Applied to EM Neuronal Reconstruction*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 2203 –2210. ↑8, 12, 13, 18

[106] Martin J. Wainwright and Michael I. Jordan, *Graphical Models, Exponential Families, and Variational Inference*, Now Publishers Inc., Hanover, MA, USA, 2008. ↑38, 51

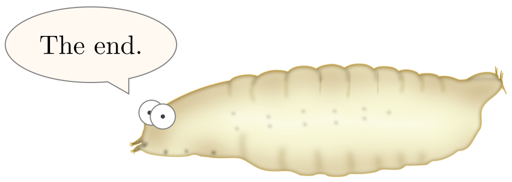
[107] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky, *MAP estimation via agreement on trees: message-passing and linear programming*, Information Theory, IEEE Transactions on **51** (2005Nov), no. 11, 3697–3717. ↑50, 57

[108] JG White, E Southgate, JN Thomson, and S Brenner, *The structure of the nervous system of the nematode Caenorhabditis elegans*, Philosophical Transactions of the Royal Society of London. B, Biological Sciences **314** (1986), no. 1165, 1. ↑8

[109] Yi Yang and Deva Ramanan, *Articulated Human Detection with Flexible Mixtures-of-Parts*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2013). ↑38

[110] Yuriy and Mishchenko, *Automation of 3D reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs*, Journal of Neuroscience Methods **176** (2009), no. 2, 276 –289. ↑8, 12, 13

[111] Anthony M. Zador, Joshua Dubnau, Hassana K. Oyibo, Huiqing Zhan, Gang Cao, and Ian D. Peikon, *Sequencing the Connectome*, PLoS Biol **10** (201210), no. 10, e1001411. ↑76


The end.