DISSERTATION
submitted to the
Combined Faculty of Natural Sciences and Mathematics
of Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by

Constantin Pape
Born in Göttingen, Germany

Oral examination: 24.06.2021

# Scalable Instance Segmentation for Microscopy

Referees: Prof. Dr. rer. nat. Fred A. Hamprecht
Prof. Dr. rer. nat. Robert Strzodka

**Abstract** Modern microscopy techniques acquire images at very high rates, high spatial resolution and with a large field of view. To analyze the large image data-sets acquired with such microscopes, accurate and scalable automated analysis is desperately needed. A key component is the instance segmentation of structures of interest, such as cells, neurons or organelles.

In this thesis, we develop scalable methods for boundary based instance segmentation. We make use of Lifted Multicut graph partitioning and develop a method achieving state-of-the-art results on challenging benchmark data-sets. In order to scale this approach up, we introduce a new approximate solver for Multicut and Lifted Multicut, which can solve problems that were previously infeasible. We further establish a method to incorporate domain knowledge into the segmentation problem, which can significantly improve quality. To overcome the brittleness of seeded watersheds, used extensively in segmentation for microscopy, we introduce the Mutex Watershed. This efficient algorithm can segment images directly from pixels without the need for seeds or thresholds. Finally, we apply our methods in collaborative work, demonstrating their utility to answer biological research questions.

In summary, our contributions enable scalable instance segmentation, thus eliminating one of the major obstacles to the automated analysis of large microscopy data-sets.

**Zusammenfassung** Moderne Mikroskopie Verfahren ermöglichen Aufnahmen mit sehr hoher Rate, hoher räumlicher Auflösung und großem Blickfeld. Um große Datensätze, die mit solchen Mikroskopen aufgenommen wurden, zu analysieren werden akkurate und skalierbare automatisierte Analyseverfahren dringend benötigt. Ein zentraler Bestandteil ist die Instanz Segmentierung der zu untersuchenden Strukturen, wie etwa Zellen, Neuronen oder Organellen.

In dieser Arbeit entwickeln wir skalierbare Methoden zur Membran-basierten Instanz Segmentierung. Wir benutzen Lifted Multicut Graph Partitionierung und zeigen, dass unsere Methode beste Ergebnisse auf schwierigen Referenz-Datensätzen erzielen. Um diesen Ansatz auf großen Daten anzuwenden entwicklen wir neue approximative Lösungsverfahren für Multicut und Lifted Multicut, die zuvor undurchführbare Problem lösen können. Ausserdem führen wir eine Methode zur Einbeziehung von domänen-spezifischen Wissen ein, welche die Segmentierungsqualität signifikant verbessern kann. Um den marker-basierten Wasserscheiden Algorithmus, der häufig in der Segmentierung von Mikroskopie Bildern verwendet wird, zu verbessern, führen wir den Mutex Watershed ein. Dieser effiziente Algorithmus kann Bilder direkt von Pixeln segmentieren, ohne Marker oder Grenzwerte zu benötigen. Wir wenden die von uns entwickelten Methoden im Rahmen mehrerer Forschungs-Kollaborationen an und demonstrieren ihre Tauglichkeit um biolgische Fragestellungen zu beantworten.

Unsere Arbeit ermöglicht skalierbare Instanz Segmentierung und beseitigt somit eines der größten Hindernisse zur automatisierten Analyse von großen Mikroskopie Datensätzen.

# Acknowledgments

First and foremost, I want to thank my two main supervisors, Professor Fred Hamprecht and Doctor Anna Kreshuk, for their continued supervision and support of my research projects. Since I have joined Fred's research group as a master student, he sparked my interest in image analysis and enabled me to work on exciting topics at the interface of computer science and biology. Since working on my PhD, I have been jointly supervised by Anna, first as a senior postdoc in Fred's group, now as a group leader at EMBL Heidelberg. When she moved to EMBL, I joined her group as a visiting scientist, giving me the opportunity to work on exciting research projects in a great collaborative environment.

I want to thank the current and former members of the Image Analysis and Learning Lab at Heidelberg University for many discussions and a great working environment. In particular, I want to thank Steffen Wolf for countless hours working on joint research projects, including our joint development of the Mutex Watershed. Thorsten Beier for introducing me to the wonders of C++ and his nifty software libraries, that I still use on a daily basis. Nasim Rahaman, our former resident neural network wizard, for helping me getting started with deep learning. I also want to thank Alberto Bailoni for interesting joint work on segmentation algorithms, Roman Remme and Lorenzo Cerrone for invaluable contributions to the Covid-IF project and Ullrich Köthe for many fruitful discussions.

I want to thank the members of the Kreshuk Lab at EMBL Heidelberg, especially Adrian Wolny for working on exciting ideas, Valentyna Zinchenko, Dominik Kutra, Fynn Beuttenmüller, Maxim Novikov, Alex Matskevytch, Qin Yu and all other members of the lab.

I want to thank Stephan Saalfeeld and Davi Bock for giving me the opportunity to join Janelia Research Campus for an extended research visit and work on the forefront of problems in connectomics. And I want to thank Larissa Heinrich, Philipp Hanslovsky, John Bogovic, Scott Lauritzen and Arlo Sheridan for the joint projects we have worked on during my visit.

I want to thank Christian Tischer and Kimberly Meechan for our joint work on MoBIE and the PlatyBrowser and Paola Bertucci, Giulia Mizzon, Jake Musser, Rachel Templin, Julian Hennies, Hernando Martinez, Yannick Schwab and Detlev Arendt for joint work and many discussions at EMBL. I also want to thank Vibor Laketa for joint work on the Covid-IF assay and introducing me to high-throughput microscopy in the process.

Finally, I want to thank Barbara Werner for being an administrative hero and Professor Robert Strzodka for agreeing to be the second referee for this thesis.

# Contents

# 1 Introduction

Microscopy is a key technique used to study the processes underpinning life. Many of the fundamental principles of biology were established due to the advent and improvement of light microscopes. This includes the identification of cells as the basic unit of life by van Leeuwenhoek in the 17th century or the discovery of neurons as the main component of information processing in the brain by y Cajal in the early 20th century. Modern biology relies on a variety of microscopy techniques for in situ, in vivo or in vitro imaging at different scales. For example, multi-photon [81, 61] or light-sheet microscopes [99, 88] can image samples in situ with a sub-micrometer resolution and a field of view of several millimeters. Serial section or transmission electron microscopes [51, 85, 30] can image samples in vitro with a resolution of a few nanometers and a field of view of several hundred micrometers. Cryo electron microscopes [54, 157] can image samples in vitro with a resolution of a few angstroms and a field of view of a few micrometers. The resulting image data can easily exceed a terabyte (TB) in size for a single experiment due to the high spatial resolution and/or large field of view. For such large data-sets systematic manual analysis is prohibitively time consuming and has to be automated as much as possible. Take for example *connectomics*, a sub-field of neuroscience with the goal of reconstructing the neural wiring diagram of (small) animals. This diagram is made up of the neurons of the central nervous system and their connections via synapses. The smallest neuronal processes have diameters of less than 100 nanometer and synapses can be as small as 40 nanometer, while projections of a single neuron can encompass the whole nervous system. Hence, a reliable reconstruction of the wiring diagram is only possible with volume electron microscopy (EM). Even the brains of small animals imaged with this technique results in very large data-sets: two recently acquired fruit-fly brains [242, 187] measure about 50 TB of raw image data each. Figure 1.1 shows neuron reconstructions from [242], illustrating that this tasks involves several scales. Manual or semi-automated reconstruction of specific circuits takes multiple years of work for expert annotators [103, 57, 204, 150] and the reconstruction of the full wiring diagram is infeasible without substantial automation. Other fields like cell biology make use of EM to study the ultra-structure of cells [239, 152] or the cellular composition of organisms [218, 43], resulting in data-sets of similar size.

A key requirement for automated analysis is the identification and delineation of the structures of interest, such as organs, cells, neurons or organelles, e.g. mitochondria. This task can be solved via *instance segmentation*, a well-studied problem in computer vision. In recent

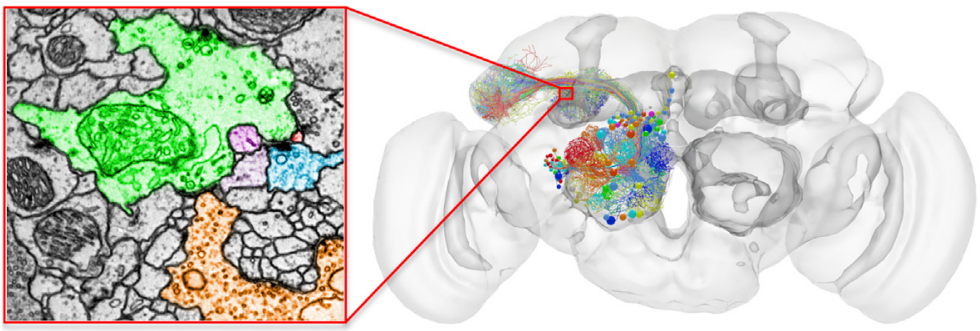


Figure 1.1: Neuron reconstruction in the fruit-fly brain. On the right, the outline of the whole brain and the manual reconstructions of several neurons are shown. The left shows an EM zoom-in, with reconstructed neurons overlaid in color. Figure from [242].

years, instance segmentation methods have improved significantly due to the adoption of convolutional neural networks (CNN), see Section 1.1. While research focuses mainly on natural images, very similar methods can be applied to microscopy images. However, scaling instance segmentation to large volumetric data is challenging and has hindered the full adoption of these methods for EM and other volumetric microscopy.

In this thesis, we have developed high-quality *and* scalable instance segmentation methods for volumetric data beyond the TB scale. Initially developed with the use-case of EM connectomics in mind, we have used them to segment neurons in large EM volumes. They have already been integrated in distributed computing solutions for large-scale neuron reconstruction [133]. Their versatility also enabled the application in other fields of biology. See Figure 1.2 for an overview of segmentation results for problems arising in connectomics, cell biology, evolutionary-developmental biology and virology. This includes a cellular atlas of *Platynereis dumerilii*, a small marine worm, build from the segmentation of all cells, nuclei and cellular ultra-structure; representing the first EM instance segmentation for a complete small animal.

## 1.1 Machine Learning for Image Segmentation

Segmentation is a fundamental task in computer vision. It consists of finding and delineating objects in a digital image or image volume. Different versions of this task exist: in *semantic segmentation*, each pixel is assigned to a discrete set of categorical labels, e.g. $\{road, tree, car, sky, pedestrian\}$. In *instance segmentation*, each pixel is assigned to the instance of a specific category, e.g. $\{car1, car2, car3, ...\}$, or background. The number of instances is usually unknown a priori. Combining these two tasks, *semantic instance segmenta-*

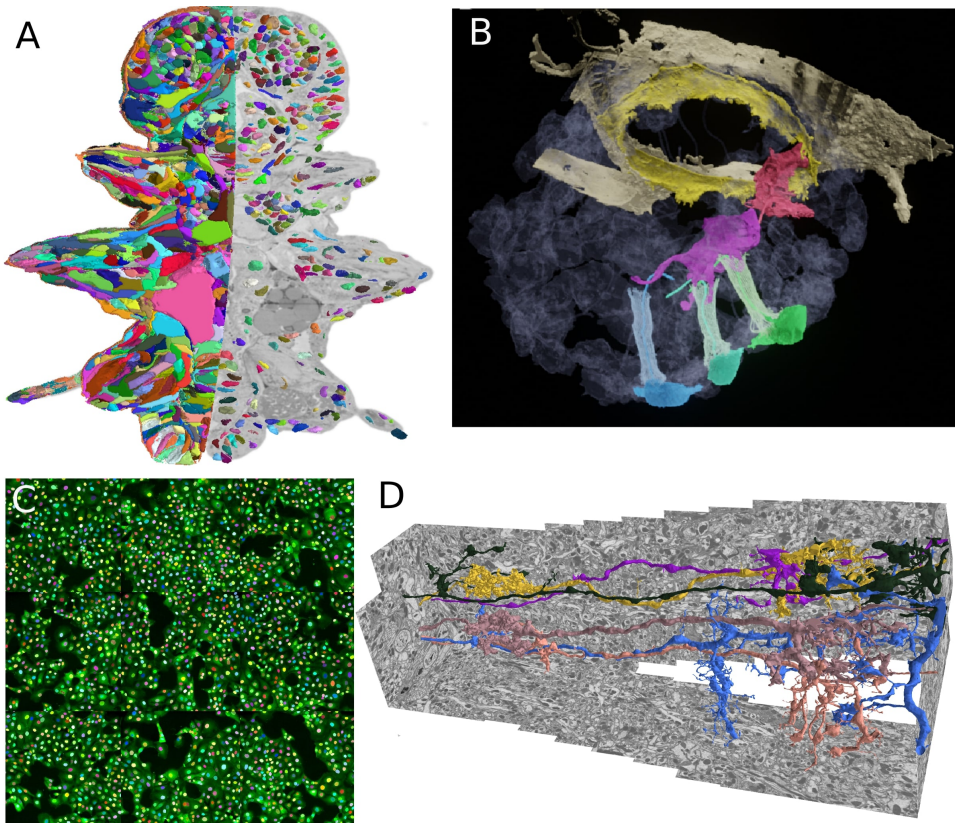



Figure 1.2: Applications of large-scale instance segmentation in this thesis. (**A**) cross section of the segmentation of all cells and nuclei in an EM volume containing a *Platynereis dumerilii* larva (Section 6.1). (**B**) cells and ultra-structure segmented from an EM volume of a sponge choanocyte chamber (Section 6.2). (**C**) nuclei segmented in immunofluorescence images for a Sars-CoV-2 antibody test (Section 6.3). (**D**) selection of neurites segmented in an EM volume of fruit-fly neural tissue (Section 4.4.2).

*tion* asks for an assignment to both a semantic class label and, for non-background classes, to an instance label. This thesis focuses on instance segmentation, which often needs a semantic segmentation as prerequisite. Hence, we will briefly summarize the current leading approaches for both tasks.

Most current semantic segmentation approaches are based on deep learning. This branch of machine learning uses artificial neural networks for predictive tasks. Loosely inspired by the nervous system, artificial neural networks consist of layers of non-linear functions, the

"neurons", connected by learnable weights. Given an input image, semantic segmentation networks predict the class membership probabilities for each pixel [130]. Here, we will focus on methods trained via supervised learning: during training the network at hand is presented with pairs of images and label maps and predicts the pixel-wise probabilities. The difference between label map and prediction is measured with a loss function, e.g. cross entropy [68]. The networks weights are updated using stochastic gradient descent [182] or one of its modern variants [104, 55]. These steps are repeated until the loss converges or some other stopping criterion is met.

Most networks for computer vision are based on the convolutional neural network (CNN) architecture. It is feed-forward, i.e. layers only receive input from previous layers, and neurons are only connected in a local neighborhood. This reduces the number of parameters, while keeping spatial context and introducing shift invariance. Unlike architectures for image classification [112, 79], which predict a scalar label per image, networks for semantic segmentation transform the input image into a label map with a fully convolutional architecture. Modern architectures have improved the initial fully convolutional network [130] by introducing skip connections [180], atrous convolutions [39] or multi-scale feature pyramids [127].

In the case of instance segmentation the label maps are permutation invariant. Take the example of segmenting cars: here, it is arbitrary which car is labeled $car1, car2, car3$ and so on. Hence, semantic segmentation methods, which use a loss function that is *not* permutation invariant, cannot be applied directly. Various solutions have been proposed. For example, predicting pixel-wise embeddings that can be clustered to obtain instances and are trained via a contrastive [49] or spatial [156] loss function. Other approaches first predict object bounding boxes, followed by the prediction of an instance mask per bounding box [78], or iteratively predict object masks [91].

Here, we solve the instance segmentation problem via graph partitioning methods, see Section 1.1.1, using CNN boundary predictions to construct the graph and estimate edge costs.

### 1.1.1 Graph-based Instance Segmentation

Images or volumes can be represented in terms of a graph $G = (V, E)$ with nodes $V$ and edges $E$. Such a representation can be obtained either via a *grid graph*, where nodes correspond to pixels and edges connect pixels in their local neighborhood, or a *region adjacency graph*, where nodes correspond to groups of pixels, often called superpixels, and edges connect superpixels that share a boundary. To obtain an instance segmentation, the nodes of a graph are grouped into partition elements based on costs[1] $C$ associated with the edges[2]. The costs estimate the

[1] In the literature also often called weights.

[2] The partitioning procedure can also be informed by costs associated with the nodes; here we restrict ourselves to the case with only edge costs.

likelihood of the incident nodes being part of the same partition element or of different partition elements. They can be obtained from properties of the image, e.g. the gradient, directly or predicted by an edge classifier.

An important graph-based segmentation methods is the seeded watershed [48, 145, 144], which finds a partition by flooding a height-map, in our case corresponding to the edge costs, starting from a set of seed nodes. It can be efficiently computed as a minimum spanning forest via Kruskal's algorithm [48, 144].

Another class of methods [62, 67, 161] is based on agglomerative clustering [224]. They agglomerate nodes based on the edge costs until a stopping criterion, e.g. a cost threshold or a preset number of partition elements, is met.

Both watershed and agglomerative clustering typically operate on unsigned graphs, i.e. graphs which have only positive (attractive) edges costs. Hence, an additional stopping criterion, like seeds, thresholds or a preset number of elements, is required to partition the graph. Defining a good stopping criterion can be challenging, especially for large graphs where the number of instances is often unknown. In contrast, Multicut [5, 6, 41] (also known as correlation clustering [17]) operates on signed graphs with attractive and repulsive edge costs, removing the need for a stopping criterion. Furthermore, the recently introduced Lifted Multicut [87] extends the Multicut formulation by long range graph interactions, allowing a more expressive formulation of the partition problem. However, solving the (Lifted) Multicut problem is NP-hard. Thus, in practice efficient approximate solvers are required. See Section 2.2.4 and Section 2.2.6 for an in-depth review of Multicut and Lifted Multicut graph partitioning.

In this thesis, we make use of the seeded watershed at the pixel level to obtain superpixels, which are used to construct a region adjacency graph. Multicut and Lifted Multicut are used to partition this graph, yielding the desired instance segmentation. One of our main contributions is the introduction of approximate solvers for these two problems, making them applicable for TB-sized volumetric microscopy data-sets. Furthermore in Chapter 5 we introduce the Mutex Watershed an instance segmentation algorithm that is, similar to watershed, based on Kruskal's algorithm. Unlike watershed it can incorporate attractive and repulsive costs and can thus provide an instance segmentation from grid graphs without seeds.

### 1.1.2 Instance Segmentation for Microscopy

Segmentation for microscopy has recently improved significantly through the wide adoption of deep learning based methods. The U-Net [180, 35] is especially popular for semantic segmentation tasks. To obtain an instance segmentation the network predictions have to be further post-processed. For structures with "simple" shapes, this can be achieved by a seeded watershed [132], methods using a shape-prior [191, 225, 200] or combining boundary and bounding box predictions [86].

Such shape assumptions do not hold for many microscopy segmentation problems. Take for example the case of neuron reconstruction, where individual neurons can extend through significant parts of the volume and have a complex tree-like structure. The predominant approach for such complex objects proceeds by predicting boundaries, computing superpixels based on the boundary predictions, building a region adjacency graph from the superpixels and partitioning this graph to obtain a segmentation [67, 5, 161, 120, 212]. See Section 1.1.1 for an overview of graph-based partitioning. Alternative approaches extend single- [141, 91] or multi-object [142] predictions recursively with a neural network. However, this requires repeated network prediction per object, resulting in significantly larger computational requirements, which prohibits the use in most academic settings [91, 124].

Applying graph-based methods to large volumetric microscopy data requires parallelizing the boundary prediction, graph extraction and graph partitioning. The two former processing steps can be parallelized by decomposing the volume into chunks, processing all chunks in parallel and merging the results with a cheap global operation. However, parallelizing the graph partitioning is more challenging, as this step relies on sequential algorithms. Thus, it is often parallelized by computing the instance segmentation on chunks with overlap and then stitching the results by matching segments in the overlap [179, 176] or by rerunning the segmentation method on the overlap [141]. In contrast, the method we introduce in Chapter 3 solves the partitioning problem for sub-graphs extracted from chunks, uses the solutions to reduce the size of the global partitioning problem and then solves the reduced global problem. This approach yields superior results compared to the local stitching approaches, see Section 3.4.2.

## 1.2 Contributions

This thesis develops scalable methods for instance segmentation that provide high-quality segmentation for large microscopy data.

In Chapter 2 we give a more detailed overview of Multicut and Lifted Multicut partitioning. We establish an instance segmentation pipeline that combines these partitioning methods with boundary predictions from neural networks. It significantly improves the quality of neuron segmentation from volume EM data and achieves state-of-the-art results on challenging benchmark data-sets.

In Chapter 3 we introduce a new approximate solver for the Multicut and Lifted Multicut problem. It can solve problems arising from instance segmentation for large volumetric data, which were not feasible with extant approximate solvers. It operates by extracting sub-problems from a spatial tiling of the volume, solving them, using their solutions to reduce the size of the global problem and then solving the reduced global problem. Empirically, the segmentation quality of this approach is on par with solutions from more computationally expensive approxi-

mate solvers. We also show that solving a reduced global problem yields significantly better segmentation accuracy than more greedy approaches like stitching by overlap.

In Chapter 4 we use the Lifted Multicut to incorporate prior knowledge about the biological system underlying the segmentation problem. We present a generic recipe to map such priors to sparse lifted edges. It is applicable if priors can be spatially attributed. In this framework, the priors are expressed probabilistically and can thus improve segmentation results even with uncertain attribution. We show that this procedure improves segmentation results for a diverse set of problems.

While the approximate solvers introduced for Multicut and Lifted Multicut enable the segmentation of large volumes, these methods are still too costly to be applied directly on the pixel level and thus rely on an initial over-segmentation. The Mutex Watershed introduced in Chapter 5 tackles this issue: this greedy algorithm can be formulated similar to the seeded watershed, but it incorporates attractive and repulsive interactions. Thus, it can segment instances directly from pixels without the need for seeds or other stopping criteria.

The methods developed in this thesis have been applied to segment large microscopy data-sets in collaboration with biologists. In Chapter 6 we highlight three applications to show how large-scale segmentation can help to make the most of such data-sets.

- We segment all cells and nuclei, as well as selected organelles, in a multiple TB-sized EM volume containing the complete body of a *Platynereis dumerilii* larva. This effort constitutes the first complete cellular segmentation of an animal imaged in EM. Our segmentation is combined with genetic markers to build a cellular atlas of morphological and genetic information at an unprecedented scale.

- We segment the cells and cellular ultra-structure in the choanocyte chamber of a sponge. This enables the study of spatial interactions between different cell types in the volume, strengthening the hypothesis that sponges contain pre-cursor neuronal cell types.

- We develop an image analysis pipeline for the quantitative evaluation of a high-throughput microscopy Sars-Cov-2 antibody test. This work is based on extant segmentation methods, but leverages the tools to process TB-sized image data developed in this thesis.

Our methods and tools for scalable instance segmentation are available as open-source software. This includes formats for efficient storage of large volumetric data and tools for distributed computation, see further descriptions in Appendix A. They are already used in a large-scale connectomics pipelines [133, 80] and used as dependencies in other microscopy segmentation tools [236, 22].

# 2 Multicut and Lifted Multicut for EM Segmentation

We have outlined the importance of instance segmentation for the analysis of microscopy images in Chapter 1. The predominant approach for complex shapes consists of three steps (cf. Section 1.1.2): boundary prediction, superpixel generation and graph-based merging of superpixels. Agglomerative clustering techniques are widely used for merging superpixes (e.g. [67, 212, 160]). In contrast, we merge superpixels by solving the Multicut problem [41, 17, 6] or the Lifted Multicut problem [87], an extension that can incorporate long-range information.

In Section 2.2 we explain the foundations of the instance segmentation methods developed in this thesis. In Section 2.3 we present results of our approach for the task of EM neuron segmentation. These results are based on [21], a publication multiple authors have contributed to: Thorsten Beier has adapted the Lifted Multicut problem to neuron segmentation, Nasim Rahaman has implemented the CNN for boundary prediction and Timo Prange has developed the watershed procedure for superpixel generation. We have developed edge features suitable for anisotropic and isotropic data and integrated the separate steps into a common segmentation pipeline. In addition, this chapter includes results we have contributed to [16]. At the time of publication of [21] our approach achieved state-of-the-art on three challenging benchmark data-sets. It is still the best performing method in the CREMI challenge [66], the latest and most challenging of the benchmarks.

## 2.1 Introduction

In an effort to understand the physical correlates of information processing in animals the connectomics community is acquiring volumetric EM images of neural tissue at an unprecedented rate [4, 96, 27, 121, 205, 23, 31]. The rate of acquisition and the size of the data is far exceeding the human capacity for neuron tracing, even when allowing for massively parallel annotation [82, 30, 103, 34, 184, 98, 223, 125]. Hence, reliable automatic segmentation is urgently needed for upcoming whole-brain data-sets (>50 TB per volume). In response, the computer vision community is developing segmentation approaches to decrease the manual proof-reading effort, with the ultimate goal of generating accurate segmentations fully automatically.

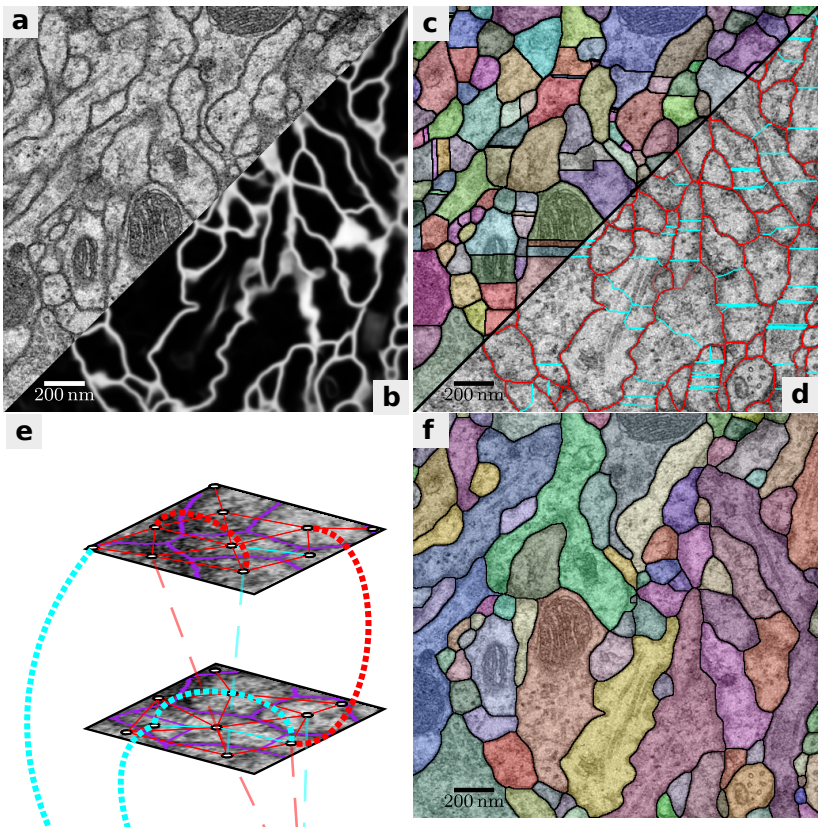
The progress of automated segmentation methods for connectomics has been largely driven by 'blind' challenges. They provide a training set comprising of raw data and labels and a test set, which only provides raw data. Segmentation results for the test data can be submitted to a server, where they are scored against the private test labels. The first neuron EM segmentation challenge has been organized at the ISBI 2012 conference [13], followed by a challenge at the ISBI 2013 conference [12] and the CREMI challenge organized at the MICCAI 2016 conference [66]. While the 2012 challenge offers a 3D data-set, it is highly anisotropic and results are evaluated only in 2D. The two later challenges are evaluated fully in 3D and provide significantly more, and more diverse, data. All three challenges remain open for submission, but the test labels for [12] have been published as part of a larger connectomics data-set [96].

Since many neurites are locally similar, most segmentation procedures rely primarily on boundary information. Such cell membrane probabilities can be estimated using neural networks [90, 210, 42] or shallow classifiers such as a random forest [7], possibly augmented with a conditional random field [97]. More recently, fully convolutional networks [130] have been adopted for this task and architectures using skip connections [180, 35] or residual skip connections [120] have improved boundary predictions significantly. Ideally, the connected components of thresholded boundary predictions would already correspond to neurites. Given that all boundary predictions available to date are imperfect, it is useful to first conservatively group pixels into superpixels that afford the extraction of more expressive features. These superpixels can then be grouped into tentative neurites in a second step, using approaches based on agglomerative clustering [7, 217, 129, 92, 64, 160, 212, 67] or graph partitioning [6].

Following this approach, we develop a three step segmentation pipeline. First, we apply a cascaded random forest, which needs few training labels, or a CNN, yielding higher accuracy at the cost of more training labels, to predict membrane probabilities. See Section 2.2.1 for an overview of the different boundary estimators. Second, we aggregate the pixels into superpixels to coarse grain the problem and extract higher order region information. Here, we use a watershed algorithm based on the distance transform of the boundary predictions. This approach yields large superpixels while being robust against gaps in the predictions, see also Section 2.2.2. Finally, we merge superpixels into segments by solving the Multicut [6] or Lifted Multicut [87] graph partitioning problem. See Section 2.2.3 and Section 2.2.4 for details on the Multicut set-up and problem; Section 2.2.5 and Section 2.2.6 for details on the Lifted Multicut. Figure 2.1 shows an overview of our segmentation pipeline applied to the ISBI2012 challenge.

This segmentation pipeline achieved state-of-the-art results on the three neuron segmentation challenges [1], narrowing the gap between automated and manual segmentation accuracy. In Section 2.3.2 we perform a lesion study for our pipeline, with the table in Figure 2.1 showing how the accuracy degrades when deviating from the optimal choices.

[1] Results for [13] and [12] have since been improved upon. We are still state-of-the-art in [66].




g

| CNN | DT WS | MC | LMC | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ |
|---|---|---|---|---|---|
| ✗ | ✗ | ✗ | ✗ | 0.89390 | 0.95440 |
| ✗ | ✓ | ✓ | ✗ | 0.97907 | 0.98844 |
| ✓ | ✗ | ✗ | ✗ | 0.97735 | 0.98865 |
| ✓ | ✗ | ✓ | ✗ | 0.98039 | 0.98855 |
| ✓ | ✓ | ✓ | ✗ | 0.98257 | 0.98946 |
| ✓ | ✓ | ✗ | ✓ | 0.98262 | 0.98946 |

Figure 2.1: Our neuron segmentation pipeline. (**a**) Example of ISBI 2012 data. Membrane probabilities (**b**) are used to find superpixels (**c**). Superpixel pairs are associated with attractive (cyan) or repulsive (red) costs, informed by local appearance (**d**). We consider next neighbor interactions (straight lines and straight dashed lines) in the region adjacency graph (**e**) and additional longer range interactions (curved lines) for the Lifted Multicut. Solving the respective partition problem yields tentative neurons (**f**). Results in (**g**) show the performance reached on the ISBI 2012 challenge using: a cascaded random forest; the same with distance transform watershed superpixels (DT WS) and Multicut (MC); our CNN; the same with standard watershed superpixels and MC; the same with DT WS and MC; and finally the proposed pipeline. Accuracies are measured by scores derived from the Rand index (RI) and the variation of information (VI). Higher scores are better.

## 2.2 Methods

The instance segmentation method developed in this chapter and used throughout the thesis operates in three steps: First boundary probabilities are predicted with a CNN or random forest, see Section 2.2.1. Given these predictions, pixels are grouped into superpixels via seeded watershed, see Section 2.2.2. The instance segmentation is then obtained via Multicut (Section 2.2.4) or Lifted Multicut (Section 2.2.6). These graph partition problems are formulated on the region adjacency graph derived from the superpixels and costs for regular (Section 2.2.3) and lifted (Section 2.2.5) edges.

### 2.2.1 Boundary Prediction

Boundary probability prediction is a binary segmentation task corresponding to the separation of pixels into foreground (membrane boundary) and background (inner-cellular space). To solve this task, we use two different supervised machine learning approaches. The first approach uses a CNN, see also Section 1.1. We use two different CNN architectures in this chapter, the first based on inception modules [202], the second based on a 3D U-Net [35]. In the second case, we predict affinities instead of boundaries. Affinities encode the pixel connectivity in multiple output channels, each channel corresponding to the likelihood that a label transition occurs between two pixels at a defined offset. For example, nearest neighbor affinities in 3d contain three channels, each channel encoding the label transitions to the adjacent pixels in z-, y- and x-direction. Figure 2.2 shows a raw image and ground-truth segmentation from the ISBI2012 training data-set and the boundary / affinity labels derived from the segmentation with the corresponding network predictions.

While CNNs currently yield the best segmentation results, they require a large amount of training data. In addition, the training data should be annotated *densely*, i.e. each pixel in the image should be labeled[2]. Hence, we have also developed a "shallow" classifier for this task, a cascaded random forest. It can be trained from less *and* sparse annotations.

**Cascaded Random Forest** While providing high-quality segmentation results, the many free parameters in neural networks call for copious amounts of densely labeled training data. Seeking to alleviate this requirement, we use a cascaded random forest to predict membrane probabilities. It is trained interactively with sparse labels and uses convolutional filters (e.g. gaussian smoothing, laplacian of gaussian) applied to the raw data as features. Following ideas from autocontext [209, 93], we perform two rounds of training and prediction. We have

[2] Training CNNs with sparse labels is possible, but in most cases yields inferior results.

implemented the random forest training in ilastik [22], a software package for interactive machine learning.

In more detail, the first random forest is trained interactively, using multiple semantic classes: membrane, cytoplasm, mitochondria, mitochondrial membrane, synaptic sites and "everything else". In the second round, the predictions for all classes are concatenated to the raw data as new channels and the features are computed for all channels. Labels are again provided interactively in ilastik, this time annotating only "boundary" and "background".

**ICv1** The network architecture, called ICv1, used for ISBI2012 and SNEMI experiments is based on inception modules [202], uses strided 2D convolutions, max-pool operations and deconvolutions in a fully convolutional setting [130]. Similar to [180] it combines high- and low-resolution pathways to aggregate geometric and semantic features. Deviating from recent trends [194], we use relatively large convolution kernels (up to $9 \times 9$ pixels) to enhance the field of view without introducing significant memory overhead. See Figure A.3 for a detailed overview of the architecture.

The network makes extensive use of train and test time data augmentation, using random linear and nonlinear transformations: flips, rotations and elastic deformations. During training, these transformations are applied to both raw images and labels to increase the effective amount of training data. During prediction, a transformation is applied to the raw images, followed by the network and the inverse transformation applied to the network's prediction. The process is repeated for different transformations and the results are averaged. This augmentation method makes used of the fact that the transformed images are clearly distinct from the originals, but in appearance still similar to real tissue images. The network is trained using pixelwise binary cross-entropy loss and using the Adam optimizer [104], see Section A.2 for all training hyperparameters.

**3d U-Net** For the CREMI experiments we use a 3D U-Net [35], implementing the architecture proposed in [67], and predict long-and-short range affinities as described in [120]. In addition to the standard training data augmentation techniques of random rotations, flips and elastic deformations (see also previous paragraph), we simulate artifacts that are present in the CREMI data-set, see also Section 2.3.5. In more detail, we randomly zero-out slices, decrease the contrast of slices, simulate tears, introduce alignment jitter and paste artifacts extracted from the training data. Both [67] and [120] have shown that these kinds of augmentations can help to alleviate issues caused by EM-imaging artifacts. Test-time data augmentation is not used. We use the L2 loss and the Adam optimizer to train the network.

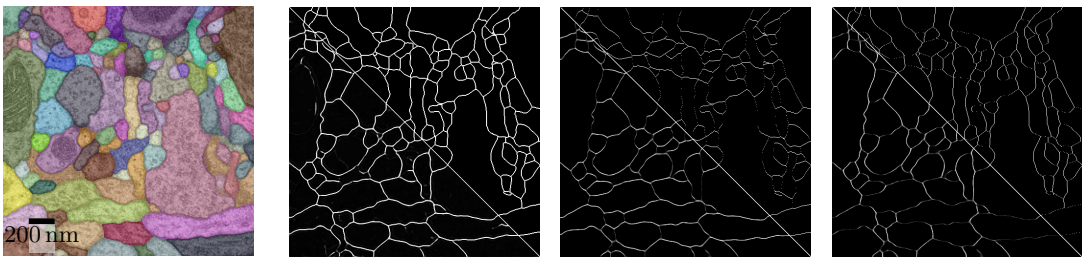



Figure 2.2: Training data and network predictions. From left to right: raw data with overlay of ground-truth instance segmentation. Boundary target labels (upper diagonal) and corresponding network prediction (lower diagonal); the image intensity encodes the pixel-wise boundary likelihood. Affinity target channels (upper diagonal) and corresponding network predictions (lower diagonal) for transition to the next pixel in x/y direction; the image intensity encodes the likelihood of a label transition in x/y direction. Data from the ISBI2012 challenge, for details on the network see Section 5.4.1.

### 2.2.2 Superpixel Generation

The field of view of neural networks is limited, typically a few hundred pixels in each spatial dimension. Given that neurites are tube- or tree-like structures that extend across several thousands of pixels at standard EM resolutions, this is a practically relevant limitation. Furthermore computation on the pixel level is impractical for large image or volumetric data. To overcome these limitation, pixels that, with a high confidence, belong to the same neurite can be grouped into a superpixel. In the subsequent processing steps the original image can then be represented in terms of superpixels, reducing the number of objects significantly. Sperpixels also allow the extraction of more informative features that are not accessible at the pixel level. For example, at the level of superpixels it is possible to answer questions like "how many mitochondria are in this superpixel" or "what is the diameter of this section of neurite" that would be ill-posed at the level of individual pixels. In the downstream processing of superpixels, they can usually only be merged and not be split. As a consequence, a good superpixel should encompass as many pixels as possible, but not extend beyond a single neurite.

For natural images, SLIC [1] or other algorithms that group pixels of similar appearance are popular. Such algorithms are not useful for the task at hand: neurites usually do not differ much in local appearance, but are separated by a membrane. In this setting, the watershed [25, 48, 145, 144] works well when applied to a (smoothed) pixelwise boundary probability map [7] and seeding from local minima.

Standard electron microscopy sample preparation and staining protocols often lead to perforations or slashes of neurite boundaries. These artifacts are a nuisance for the standard watershed, as it may result in undesirable superpixels that span two adjacent neurites. As a

consequence, we here propose to first threshold the boundary probabilities; to then filter out small connected boundary components; and to then compute the smoothed signed Euclidean distance transform on the remaining boundaries. See Figure 2.3 for an illustration. This is related to prior work on the stochastic watershed [8] and has the effect of closing slashes that are narrower than the diameter of either adjacent neurite in the proximity of the slash.

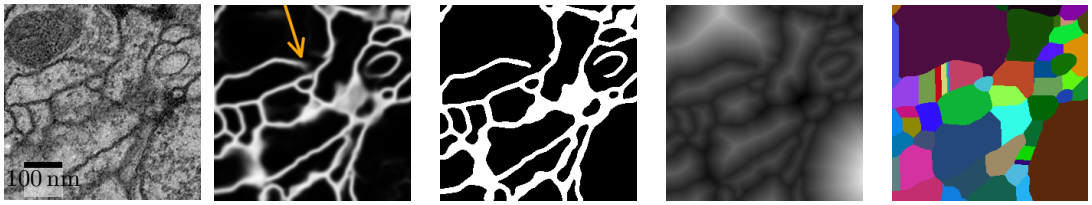



Figure 2.3: Generating distance transform watershed superpixels. From left to right: raw data, boundary probability map, thresholding thereof, distance transform on the latter, and watershed superpixels seeded by distance maxima. Note that the resulting superpixels are confined to single neurites, in spite of ambiguous boundary evidence (arrow).

For isotropic data, the superpixels are generated in 3D. For data with a high degree of anisotropy, no good 3D superpixels can be generated using this approach, because large gaps in between membranes in adjacent slices often lead to a single superpixel spanning several neurites. Hence, for such data we generate 2D superpixels for the individual images and stack them across the anisotropic axis.

In the presence of myelin, we follow [111] and use ilastik [22] pixel classification to train a myelin detector. Large myelin components are then turned into additional superpixels, see Figure 2.4.

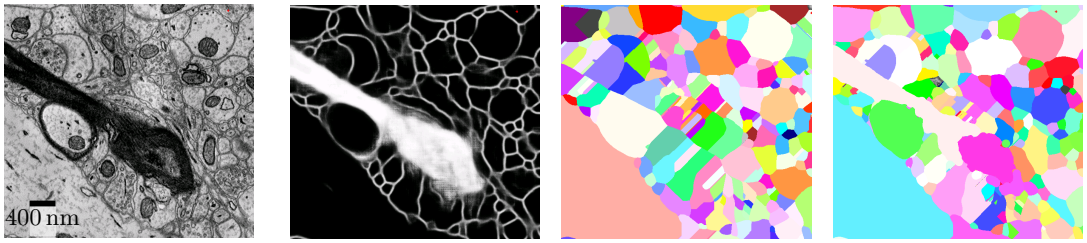



Figure 2.4: Generating superpixels in the presence of myelin. From left to right: Part of SNEMI3D test data, boundary probability map, distance transform watershed before and after myelin correction.

### 2.2.3 Edge Costs

Given the superpixel over-segmentation, we can compute the region adjacency graph to obtain a graph representation of the segmentation problem. In this graph each node corresponds to a superpixel and two nodes are connected by an edge if their superpixels share a boundary. To partition it via Multicut, we also need edge costs that represent the likelihood of the incident nodes being joined (attractive) or separated (repulsive) in the partition.

These costs can be computed by accumulating the boundary probabilities (Section 2.2.1) over the boundary pixels and then taking the average. This approach works well for accurate boundary predictions, but can lead to inferior results if the predictions are of lower quality. It does not make use of additional information like superpixel appearance or shape. Edge costs can also be computed with a classifier, such as a random forest, based on features associated with the edges. This classifier has to be trained on separate data with binary edge labels. Depending on the type of edge features, it can take information beyond local boundary evidence into account.

Here, we compute features taking into account boundary appearance, region appearance and region shape. Recall that we use planar superpixels in the case of data with high anisotropy (cf. Section 2.2.2). For planar superpixels the edges can be divided into two categories: *intra-slice edges* connecting two superpixels in the same image plane and *inter-slice edges* connecting two superpixels in adjacent planes. Intra-slice edges have a one dimensional boundary while inter-slice edges have a two dimensional boundary.

**Edge features based on boundary appearance** For these features, we first compute the response of convolutional filters (Gaussian smoothing, Hessian of Gaussian eigenvalues, Laplacian of Gaussian, each at scales $\sigma = \{1.6, 4.2, 8.3\}$) using the raw data and boundary probabilities as input. For anisotropic data, these filters are calculated either in 2D (for high anisotropy) or in 3D but with reduced sigma in the anisotropic direction. The filter responses are accumulated over superpixel boundaries using the following aggregate statistics: mean, sum, minimum, maximum, variance, skewness, kurtosis and the $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ quantiles.

**Edge features based on superpixel appearance** For each superpixel, we compute: its size, the eigenvalues of its inertial tensor, as well as the histogram (64 bins), kurtosis, maximum, minimum, $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ quantiles, skewness, sum and variance of the raw data accumulated over the superpixel. These values are mapped to the edges by taking the minimum, maximum, sum and absolute difference of the incident superpixels' values. In addition, we compute the squared distance between the incident supervoxels' centers of mass, using uniform pixel weights and weighted by the raw data intensities.

**Edge features for inter-slice edges** For anisotropic data with planar superpixels we compute the following features for inter-slice edges: the size of the union (when projecting orthogonally onto the same slice) of the adjacent superpixels, the size of their intersection and the ratios of intersection and union. In addition, we compute the ratio of the superpixels' area to circumference, mapped to edges via minimum, maximum and absolute difference.

This results in 625 features per edge. They are used by a random forest to predict, for each edge, the probability of the incident nodes being separated in the partition. The training data is obtained by mapping the ground-truth segmentation to superpixels and determining the corresponding edge labels. The random forest predicts pseudo probabilities $p$ in the range $[0, 1]$, but the costs $c$ in the Multicut objective Equation 2.2 are expected to be in the range $]-\infty, \infty[$. Negative values correspond to repulsive edges and positive values to attractive edges. To obtain costs from probabilities, we use the negative log-likelihood:

$$c = \log\frac{1-p}{p} + \log\frac{1-\beta}{\beta}. \tag{2.1}$$

The boundary bias $\beta$ influences the solution towards less ($\beta \in ]0.0, 0.5[$) or more ($\beta \in ]0.5, 1.0[$) partition elements. We normalize the cost for each edge by its length (for 1D edges) or area (for 2D edges). For anisotropic data with intra- and inter-slice edges we empirically find that it is best to train a single classifier for both edge types, setting the the inter-slice edge features to zero for intra-slice edges.

### 2.2.4 Multicut

Partitioning the nodes $V$ of a graph $G = (V, E)$ with signed costs $C$ associated with the edges $E$ is a well studied problem [100, 41]. When the number of partition elements is unknown in advance, it is known as the Multicut problem [94, 6, 5] or correlation clustering [17]. Following [6], it can be formulated as Integer Linear Problem (ILP). In the following, we refer to an edge as *cut* if the incident nodes are separated in the partition, i.e. part of different partition elements. The ILP is formulated by introducing binary edge indicator variables $y_e$ that take the value 1 if $e$ is cut and 0 otherwise. The Multicut problem is then expressed as finding the minimal sum of costs for cut edges under closedness constraints:

$$\tilde{\mathbf{y}} = \arg\min_{Y} \sum_{e} y_e \, c_e \qquad \text{subject to} \tag{2.2}$$

$$\forall\, C \in \text{cycles}(G) : \forall e \in C : y_e \leq \sum_{\hat{e} \in C \setminus \{e\}} y_{\hat{e}}. \tag{2.3}$$

Only a subset of binary edge labelings $Y$ result in a valid partition. For example, consider a triangular graph with nodes $A, B, C$. The labeling $\{(AB) = 0, (BC) = 0, (CA) = 1\}$ is not a

valid partition: A and C are assigned to the same partition element via their mutual connection to B. At the same time, they should be in distinct elements due to the cut edge between them; a contradiction. Equation 2.3 enforces a valid partition by introducing cycle constraints. For each edge $e$, they forbid edge cycles containing $e$ that have a sum of indicator variables, excluding $y_e$, that is smaller than $y_e$. In other words, if an edge is cut there must not be any path in the graph connecting the incident nodes via edges that are not cut. Without these constrains, the solution to Equation 2.2 would be to simply cut all repulsive edges. Figure 2.5 illustrates the cycle constraints on a small graph together with the corresponding instance segmentation problem.

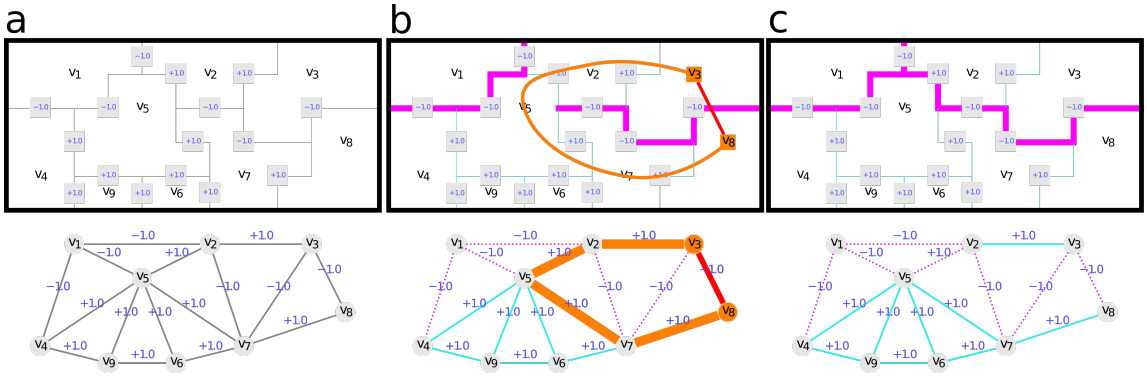



Figure 2.5: Multicut constraints: **a** over-segmentation (top) and the corresponding region adjacency graph (bottom) with edge costs. **b** partition when *only* optimizing Equation 2.2 without taking the constraints expressed by Equation 2.3 into account. Edges that are cut (separating the incident nodes) are bold in the segmentation and dashed in the graph. The edge that is violating the cycle constraint is marked in red, the edges in the corresponding cycle are orange. **c** the optimal partition under Multicut constraints.

In our context $G$ is given by the region adjacency graph and the edge costs are derived from pseudo probabilities for separating incident nodes, see also Section 2.2.3. Solving the Multicut problem to optimiality is NP-hard and thus only feasible for instance segmentation problems of modest size. Up to a few hundred thousand edges are feasible according to [6]. Approximate solvers (e.g. [159, 14, 19]) that scale to larger problem sizes exist. Please refer to Chapter 3 for the solver contributed in this thesis and Section 3.2 for more details on other approximate solvers.

### 2.2.5 Lifted Edge Costs

The Lifted Multicut problem [87] introduces *lifted edges*, which connect nodes that are not connected by a *regular* edge in the initial graph. It then extends the Multicut objective to obtain a partition consistent with the connectivity induced by initial graph in the presence of lifted edges. See also Section 2.2.6. The lifted edges can be used to incorporate long range information not available for adjacent pairs of superpixels, leading to more accurate segmentation results. There are multiple approaches for introducing lifted edges and deriving the associated costs. They can, for example, be derived from prior knowledge about the segmentation problem as in Chapter 4. In this chapter, we add lifted edges between all pairs of nodes within a fixed distance in the initial graph. The costs are estimated with a random forest as described in Section 2.2.3, but using a different set of features.

**Lifted edge features based Multicut connectedness** The Multicut partition is computed for five boundary bias values (Equation 2.1) in the range $[0.3, 0.7]$. This results in diverse partitions, which merge nodes more or less aggressively. We use binary variables that indicate whether the two incident nodes are part of the same or different partition elements as features.

**Lifted edge features based on ultrametric distance** We apply the ultrametric contour map transform [10] to generate a complete dendrogram. The ultrametric distance (the height in the dendrogram at which the two regions merge) is used as an additional feature for each lifted edge.

**Lifted edge features based on region appearance** Equivalent to those described in "Edge Features based on superpixel appearence" in Section 2.2.3.

To account for the fact that we have a considerably larger number of lifted than regular edges, we normalize the costs of both edge types by their number. For anisotropic data with planar superpixels, inter-slice lifted edges that are $d$ slices apart are further weighted by a factor of $1/(1+d)$.

### 2.2.6 Lifted Multicut

The Lifted Multicut [87] is an extension of the Multicut, which introduces a new set of edges $F$ called lifted edges. They connect nodes $v$ and $w$ not adjacent in the initial graph $G$. These edges differ from "regular" graph edges by providing only a contribution to the energy (sum of cut edges), but not inducing connectivity. They are introduced due to the observation that

non-local features can often inform the connectivity of (super)pixels. The presence of a non-local attraction should however not result in "air bridges", i.e. non-local edges that connect two (super)pixels without a connection via regular edges.

With the sets of original edges $E$, lifted edges $F$, binary indicator variables $Y$, and costs $C$ associated with all edges in $E \cup F$ the Lifted Multicut objective can be formulated as

$$\min_{y_e \in Y_{EF}} \sum_{e \in E \cup F} c_e \, y_e \quad \text{subject to} \tag{2.4}$$

$$\forall C \in \text{cycles}(G) : \forall e \in C : y_e \leq \sum_{\hat{e} \in C \setminus \{e\}} y_{\hat{e}} \tag{2.5}$$

$$\forall vw \in F \quad \forall P \in vw - \text{paths}(G) : y_{vw} \leq \sum_{e \in P} y_e \tag{2.6}$$

$$\forall vw \in F \quad \forall c \in vw - \text{cuts}(G) : 1 - y_{vw} \leq \sum_{e \in C} (1 - y_e). \tag{2.7}$$

The constraints (2.5) correspond to the Multicut constraints (Equation 2.3) and enforce a valid partition induced by the regular edges. The constraints (2.6) ensure that for each cut lifted edge $vw$, no path of not cut regular edges between the nodes $v$ ad $w$ exists. In other words, if two nodes are separated according to the lifted edge between them, they must be in different partition elements. Conversely, (2.7) ensures that for each not cut lifted edge $vw$, there exists no cut, i.e. a binary partition, in $G$ that is separating $v$ and $w$. In other words, if two nodes are joined according to the lifted edge between them, they must be in the same partition element. See Figure 2.6 for an illustration of these constraints on a small graph.

The ILP corresponding to the Lifted Multicut contains more constraints than the Multicut one, hence it is computationally even more expensive to find the optimal solution. Several approximate solvers have been introduced, e.g. in [18, 102], and enable the application to larger problems. Please refer to Chapter 3 for the Lifted Multicut solver contributed in this thesis and to Section 3.2 for more details on other approximate solvers.

## 2.3 Results

We have evaluated our segmentation method on the blind ISBI2012 and CREMI challenges, as well as on the publicly available SNEMI3D and Neuroproof data-sets. They each contain separate training and test EM image volumes. We further perform a lesion study on the ISBI2012 data to investigate the influence of the different configurations of our method.

For evaluation, we report the measures $V_{0.5}^{\text{rand}}$ and $V_{0.5}^{\text{info}}$ as defined in [13]. These are the F1-scores derived from the structured segmentation accuracy measures "Rand index" and

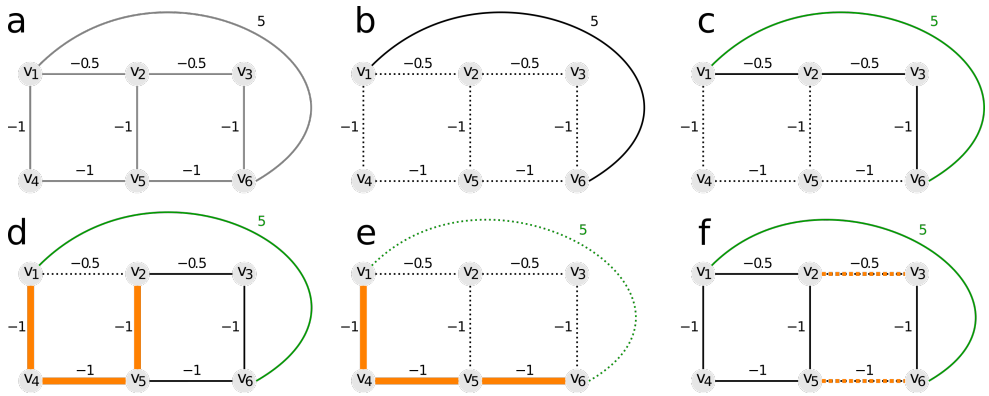



Figure 2.6: Lifted Multicut constraints. **a** Graph and costs; regular connectivity and lifted edge (1,6). **b** Multicut solution when treating (1,6) as regular edge; nodes 1 and 6 are joined, but not connected by regular edges. **c** Lifted Multicut solution, nodes 1, 2, 3 and 6 are part of one partition element. **d** Invalid partition: the cut regular edge (1,2) is violating Equation 2.5 due to a path of not cut regular edges (orange). **e** Invalid partition: the cut lifted edge (1,6) is violating Equation 2.6 due to a path of not cut regular edges (orange). **f** Invalid partition: the not cut lifted edge (1,6) is violating Equation 2.7 due to cut regular edges (orange).

"Variation of Information" [13]. Both measures go beyond aggregating single pixel errors. Instead, they summarize statistics of point pairs, verifying if they are in the same segment as prescribed by the ground-truth.

### 2.3.1 ISBI2012 challenge

The ISBI 2012 challenge [13] is the most popular and competitive connectomics challenge to date. It provides a training data-set with raw data and labels as well as a test data-set for which the raw data is publicly available and segmentation results can be uploaded for evaluation. The evaluation is performed using the measures $V_{0.5}^{\text{rand}}$ and $V_{0.5}^{\text{info}}$ for the image planes individually and then averaged. Both data-sets cover approximately $2 \times 2 \times 1.5$ microns of tissue from the fruit-fly larval brain.

We segment the test volume using the Multicut and Lifted Multicut segmentation approach outlined in Section 2.2, following the approach for anisotropic data. We use boundary probability maps produced by the network ICv1 and the cascaded random forest. The ground-truth consist of a pixel-wise labeling of membrane boundaries vs inner-cellular space. It can be used directly to train the neural network; the cascaded random forest is trained based on labels provided manually in ilastik. In order to train the random forest for edge cost prediction, we transform the provided labels into an instance segmentation by performing connected compo-

nents of the inner-cellular label for each image plane. Based on this instance segmentation, we derive binary edge labels, see Figure 2.7. Following this procedure we can derive edge labels only for the intra-slice edges. The inter-slice edges are still used in setting up the partition problem, but their costs are implicitly learned through the training data available for intra-slice edges. The Lifted Multicut results reported for this data-set were produced using all the features described in Section 2.2.5 and lifted edges were created between all superpixels within a distance of four in the initial graph.

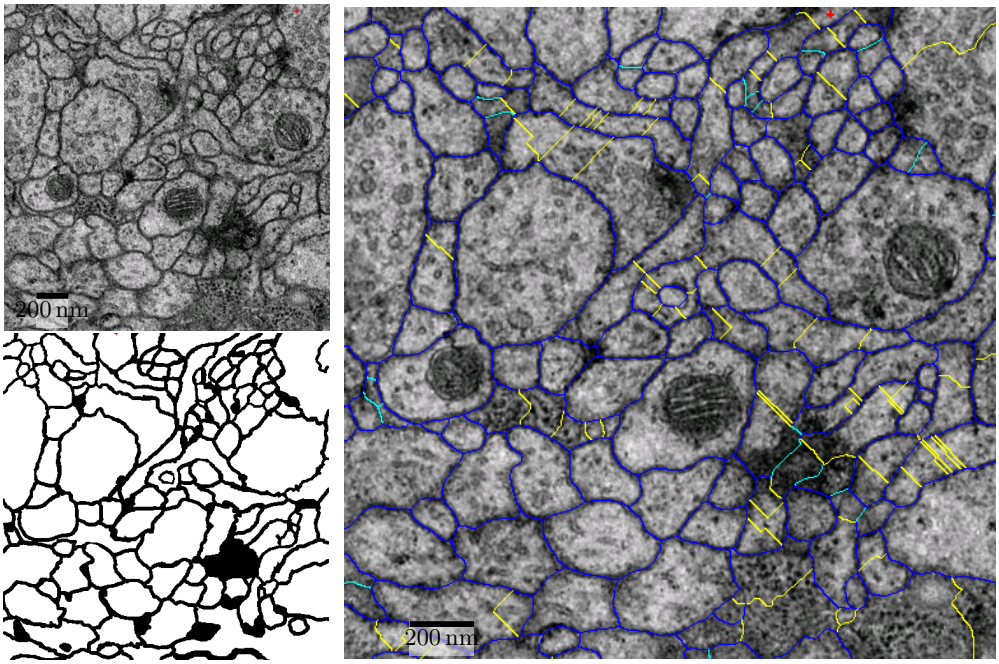



Figure 2.7: ISBI2012 training data generation. Left: Raw data and annotations. Right: edge labels derived for training the random forest. Edges are labeled either as repulsive (blue), attractive (yellow) or are treated as unlabeled (cyan) if the attribution of the corresponding superpixels to ground-truth instances was unclear.

At the time of publication of [21] the segmentation produced by the Lifted Multicut with ICv1 predictions was the highest scoring entry in the ISBI2012 challenges leaderboard, see Table 2.1. Since then several methods have improved upon these results, including our own follow-up work (Chapter 5). Note that these improvements mainly stem from advancements in network architecture or network training procedures and that several of the leading entries use our segmentation method as post-processing [192, 226, 177]. See Section 2.3.2 for a thorough comparison of the different configurations of our segmentation pipeline.

| ISBI 2012 accuracy | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ |
|---|---|---|
| LMC + ICv1 | 0.98262 | 0.98946 |
| Quan et al. [177] | 0.97804 | 0.98995 |
| Wiehman et al. [230] | 0.97714 | 0.98753 |
| Chen et al. [38] | 0.97682 | 0.98743 |
| UNet [180] | 0.97276 | 0.98662 |

Table 2.1: ISBI2012 challenge leaderboard at the time of publication of [21].

### 2.3.2 Lesion Study

To study the effectiveness of different configurations of our method, we have conducted a lesion study with the data from the ISBI2012 challenge, the best-studied of the four data-sets.

First, we have investigated the use of different classifiers for predicting the membrane probabilities. Table 2.2 shows the scores for the probability maps without additional post-processing. The segmentation accuracy is evaluated by thresholding the probability maps at different values, performing connected components, computing the accuracy measures and reporting the best measure. In addition the border thinning described in [13] is used. Our neural network, ICv1, performs substantially better than the cascaded random forest.

| ISBI 2012 accuracy | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ |
|---|---|---|
| Cascaded RF | 0.89390 | 0.95440 |
| ICv1 | 0.97735 | 0.98865 |

Table 2.2: Results of different probability maps on the ISBI test data.

Table 2.3 shows that (Lifted) Multicut segmentation significantly improves the segmentation accuracy beyond what even the best boundary probability estimators can deliver. Apparently, the (Lifted) Multicut can compensate for many errors made in the estimation of the probability map; this additional processing substantially reduces the accuracy gap between ICv1 and the cascaded random forest. Furthermore, we compare the superpixel generation using standard and distance transform watersheds. The latter yields better results in all combinations.

The results shown above were obtained by solving the (Lifted) Multicut problem obtained from a region adjacency graph set up in 3D. This approach pulls in 3D context through the inter-slice edges and yields better results than solving a 2D problem separately for each slice, see Table 2.4.

| ISBI 2012 accuracy | standard watershed | | DT watershed | |
|---|---|---|---|---|
| | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ |
| Cascaded RF & MC | 0.96229 | 0.98436 | 0.97907 | 0.98844 |
| Cascaded RF & LMC | 0.97040 | 0.98375 | 0.97852 | 0.98798 |
| ICv1 & MC | 0.98039 | 0.98855 | 0.98257 | 0.98946 |
| ICv1 & LMC | 0.97510 | 0.98757 | 0.98262 | 0.98946 |

Table 2.3: Performance of different boundary probability estimators, and different superpixel generators combined with Multicut and Lifted Multicut.

| ISBI 2012 accuracy | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ |
|---|---|---|
| 2D | 0.97775 | 0.98886 |
| 3D | 0.98257 | 0.98946 |

Table 2.4: 2D vs. 3D Multicut on the ISBI test set.

### 2.3.3 SNEMI3D Data-set

The SNEMI3D data-set [12] has been a blind challenge from 2013 to 2015, when a labeled superset of training and test data was released with [96]. It contains data from the murine cortex.

There are a number of important differences in the nature of raw data and annotations compared to ISBI2012. First, the training ground-truth is given in terms of a 3d instance segmentation (as opposed to a stack of binary membrane labels in ISBI2012). Second, the data is better resolved along the z-axis. Third, segment boundaries in the ground truth are not always properly aligned with the actual membranes in the raw data. Fourth, individual neurites are separated by a "negative" class that covers membranes, but also inter-cellular space, myelin sheaths and a few erroneously omitted thin processes.

To exploit the higher z-resolution, we supply each slice along with its two adjacent slices as input to the network ICv1. To mitigate the inaccuracies in the training labels, we weight the positive pixels (intra-cellular space) uniformly, but down weight the negative pixels (covering everything else) with increasing distance from the positive regions, reaching zero weight beyond a distance of seven pixels. Seeds for the distance transform watershed are found in 3d, and superpixels are grown in 2d on a smoothed probability map. We observed that watershed quality suffered for myelinated axons and replaced them with connected components of a binary myelin classifier, see Section 2.2.2 for details. Training labels for intra- and inter-slice superpixel pairs

are obtained as usual. Lifted edges are introduced between all pairs of superpixels within a distance of three in the region adjacency graph.

The SNEMI3D website also provides boundary probability maps shared by the authors of [42], which we also use in our segmentation pipeline, see Table 2.5. At the time of publication of [21] the segmentation produced by the Lifted Multicut based on ICv1 was the highest scoring entry in the SNEMI3D challenge leaderboard, see Table 2.6. Since then several methods have improved upon it [120, 91], again mostly leveraging advancements in network architectures and training procedures.

| SNEMI3D accuracy $\left(V_{0.5}^{\text{rand}}\right)$ | ICv1 | Ciresan [42] |
|---|---|---|
| Multicut | 0.92698 | 0.92568 |
| Lifted Multicut | 0.93122 | 0.92892 |

Table 2.5: Results of Lifted and standard Multicut on SNEMI3D for probability maps generated with the proposed ICv1 architecure and the Ciresan network [42]. The evaluation is done completely in 3D, and higher numbers are better.

| SNEMI3D accuracy | $V_{0.5}^{\text{rand}}$ |
|---|---|
| Human value | 0.94002 |
| LMC + ICv1 | 0.93122 |
| GALA [161] | 0.8995 |

Table 2.6: SNEMI3D challenge leaderboard at the time of publication of [21]. The challenge organizers have determined the human accuracy by comparing a second set of manual annotations against the ground-truth used for scoring the submissions.

### 2.3.4 Neuroproof Data-set

To study performance on isotropic data, we have turned to the example data-set[3] provided with the NeuroProof software [205]. It consists of two FIBSEM image volumes, the corresponding ground-truth segmentations as well as pre-computed membrane probability maps and superpixels. One of the volumes is used as training data, the other to evaluate the segmentation results. We use the pre-computed probabilities and superpixels as input for the partition step of our method, see the results in Table 2.7. For the Lifted Multicut, only the ultrametric distance and region features were used, with lifted edges added for superpixels within a distance of

[3] https://github.com/janelia-flyem/neuroproof_examples

two in the region adjacency graph. The resulting segmentation accuracies are higher than the published state-of-the-art [205]. The numbers given are computed for an updated ground-truth segmentation for the test data-set, where we have fixed a large segmentation error.

| Neuroproof accuracy | $V_{0.5}^{\text{rand}}$ | $V_{0.5}^{\text{info}}$ |
|---|---|---|
| Multicut | 0.93646 | 0.96173 |
| Lifted Multicut | 0.94047 | 0.96400 |

Table 2.7: Results for the isotropic neuroproof data-set. The evaluation is done completely in 3D.

### 2.3.5 CREMI challenge

The CREMI challenge [66] is the latest blind neuron segmentation challenge. It contains significantly more data than the previous challenges: three blocks of raw data and ground-truth annotations for training as well as three blocks to test the algorithm performance. The blocks come from different parts of the female adult fruit-fly brain, each block is $5 \times 5 \times 5$ microns large. Results for the test data can be uploaded to an evaluation server where they are scored according to the geometric mean of $1 - V_{0.5}^{\text{rand}}$ and the Variation of Information [140].

The data from the CREMI challenge is anisotropic and contains artifacts like missing sections, staining precipitations and folds in the support film. To alleviate difficulties stemming from misalignment, we use a version of the data that was elastically realigned by the challenge organizers with the method of [185]. We apply the Lifted Multicut segmentation pipeline, based on boundary predictions from a 3D U-Net, see Section 2.2.1 for details. At the time of writing this approach holds the top entry in the CREMI leaderboard, see Table 2.8.

## 2.4 Conclusion

At the time of publication, [21] established Lifted Multicut segmentation as the state-of-the-art method for EM neuron segmentation. Since then, methods for neuron segmentation have mainly improved the neural network architectures and training procedures. Most notably, CNNs using 3D convolutions, many based on the 3D U-Net [35], have been adopted [67, 120] and better loss function have been introduced: either adding a structured loss term [67] or an auxiliary loss [120]. In addition, some novel approaches break with the common steps of boundary prediction, superpixel generation and agglomeration: the Mutex Watershed [233] can agglomerate network predictions from pixels directly (see also Chapter 5), Flood-Filling Networks [91] predict a single object at a time and iterate this process to segment the full volume and 3C [142] can

| Method | CREMI-Score (lower is better) |
|---|---|
| 3D U-Net + LMC | 0.221 |
| PNI [120] | 0.228 |
| 3D U-Net + Clustering [16] | 0.241 |
| MALA [67] | 0.276 |
| CRU-Net [240] | 0.566 |
| LFC [169] | 0.616 |

Table 2.8: Current leading entries in the CREMI challenge leaderboard. The scores are averaged over three test data-sets and the CREMI-Score is computed via the geometric mean of $1 - V_{0.5}^{\text{rand}}$ and the Variation of Information. Using boundary probabilities from a 3D U-Net, Lifted Multicut segmentation holds the first rank.

segment multiple objects directly in a number of passes growing logarithmic with the number of objects.

Nevertheless, the three step agglomeration approach, and especially Lifted Multicut, is still relevant. Thanks to the superpixel over-segmentation and new solvers (see Chapter 3) these methods scale better to large data than other approaches and can improve even high-quality boundary maps to push the state-of-the-art: with a modern network architecture, our method is leading on the most challenging connectomics benchmark (see Section 2.3.5) and several high ranking methods in the ISBI2012 challenge use it to post-process their results [192, 226, 177]. Furthermore, the Lifted Multicut allows to incorporate long-range interactions not accessible in the limited field of view of a CNN. This enables, for example, the expression of biological priors in a probabilistic fashion (see Chapter 4).

While [120] have reported better than human performance on the SNEMI3D challenge, in practice EM neuron segmentations are still not of sufficient quality to be used without substantial proof-reading and correction efforts. All benchmark data-sets, even the CREMI challenge, contain volumes of very modest size compared to real data-sets imaged for circuit reconstruction and a quantitative evaluation of neuron segmentation accuracy at scale is still lacking.

# 3 Scalable Multicut and Lifted Multicut Segmentation

The instance segmentation method described in Chapter 2 yields accurate results for EM neuron segmentation or other boundary based segmentation problems. However, solving the (Lifted) Multicut to optimality is an NP-hard problem, prohibiting its direct application to large data-sets. Several authors have introduced approximate solvers both for Multicut, e.g. [19, 20, 122], and Lifted Multicut, e.g. [18, 102]. They scale to larger problems while still yielding accurate results. Still, these solvers are not efficient enough to optimize the problems arising from TB-sized microscopy data-sets.

Here, we describe two novel approximate solvers, one for Multicut and the other for Lifted Multicut, that scale to problems of sizes that were previously not feasible. Both solvers make use of the over-segmentation that is underlying the graph to be partitioned. They decompose the global problem into feasible sub-problems, using the over-segmentation as spatial domain. Then, they solve the sub-problems and use the solutions to reduce the size of the global problem. This step can be iterated, using increasingly larger sizes for the spatial decomposition, until the reduced global problem is feasible with an extant solver. Note that this approach relies on the spatial decomposition and is thus not applicable for (Lifted) Multicut problems without underlying over-segmentation, or a similar spatial domain. This chapter is based on the publication [166], the experiments have been updated. The extension of the approximate solver to the Lifted Multicut was introduced in [167] (Chapter 4).

## 3.1 Introduction

Connectomics is a domain of neuroscience that strives to understand structure-function relations in neural circuits from the directed graph of neural connections. The graph itself – the wiring diagram of a nervous system – is usually reconstructed from very large stacks of neural tissue images acquired by EM [30, 83]. Reconstruction of the graph consists of two major sub-problems: tracing of neurons through the image stack and detecting synapses that connect the neurons.

Unlike fluorescent labeling methods, which typically reveal a sparse subset of neurons, the heavy metal stains used in EM label all cell membranes in a piece of brain tissue. Consequently,

neuron tracing or segmentation have to be based on boundary evidence and have to be performed on images of very high resolution. At the same time, neural cells spread over very large volumes; hundreds of GB of images have to be analyzed to reconstruct a graph which would be relevant for biological analysis. Figure 3.1 shows one image extracted from such a data-set. Note how the segmented neurons on the left pass through the whole image, while the diameter of some processes in the inset on the right is so small they can only be distinguished at full resolution. Recently, the first complete brain of an adult fruit-fly has been imaged [242], the data-set measures over 50 TB. The on-going efforts to image the brains of small vertebrates are anticipated to produce data-sets of equal or greater size, for example [85, 149, 121].

Until very recently, most of the effort of the computer vision community has concentrated on solving the automated reconstruction problem correctly at small scale. Since the reconstruction accuracy has not been sufficiently high for direct biological analysis, neuroscientists have resorted to collective manual tracing [184, 28] or manual proof-reading of automatically generated segmentations [103, 96, 175]. The hurdle of fully automated segmentation, however, does not seem insurmountable any more: as we have demonstrated in Chapter 2, the gap to human segmentation accuracy is shrinking on the relevant benchmark data-sets ISBI2012 [13], SNEMI3D [12] and CREMI [66]. Recently, Lee et al. [120] have even demonstrated better-than-human accuracy on the SNEMI3D challenge data-set.[1] The question of scaling these methods up from small challenge data to TB-sized data-sets without substantial loss in accuracy is now starting to be addressed [179, 176, 141]. So far, this is achieved by performing the segmentation block-wise and merging the solutions based on local evidence. Here, we propose to optimize a global objective function instead, which allows to over-rule the local block solutions if evidence at larger scale suggests it would lead to a better overall segmentation.

Our approach builds on the neuron segmentation pipeline of [21], see also Chapter 2, which is based on (Lifted) Multicut graph partitioning. While this problem is NP-hard [41], practical approximate solvers have recently been introduced. We extend the pipeline to perform the global superpixel graph construction in a distributed manner (Section 3.3, Section A.1.2) and propose a hierarchical block-wise approximate solver for the Multicut problem (Section 3.3.1) and Lifted Multicut problem (Section 3.3.2). While the resulting solution is no longer globally optimal, we demonstrate sufficient accuracy on data from the CREMI challenge (Section 3.4.2) and excellent scaling behavior on a large cutout from the whole fruit-fly brain data-set of [242] (Section 3.4.3).

[1] Although this claim is so far specific to the SNEMI3D data and does not hold true for larger data-sets yet.

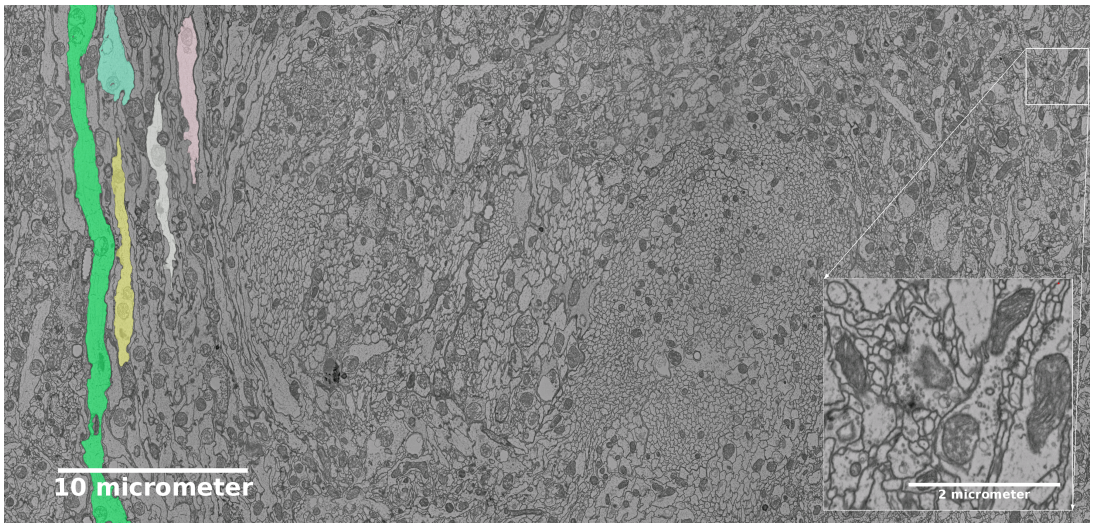



Figure 3.1: Part of an image from the whole fruit-fly brain data-set [242]. Highlighted neurons were segmented automatically with the proposed Multicut solver (some post-processing was applied to merge small fragments). The inset on the right shows some of the smaller process as well as a synaptic contact site, roughly in the middle.

## 3.2 Related Work

Most neuron segmentation pipelines in use today, e.g. [161, 64, 212], follow the same sequence of steps: boundary probability prediction with a neural network, over-segmentation with a seeded watershed and merging of the watershed superpixels into segments. See Section 1.1.2 and Section 2.1 for a more detailed discussion of this approach. Most existing methods for neuron segmentation at scale [179, 176, 141] use GALA [160, 161], a method following the three step approach, which uses agglomerative clustering with learned edge weights for merging superpixels. To apply it at scale, these approaches partition the volume into blocks, potentially with overlap, and apply GALA to each block independently and in parallel. The individual solutions are then stitched to obtain a solution for the complete volume. In [179] this is achieved by stitching segments across block boundaries according to the largest overlap. The authors of [176] employ a similar strategy, but include heuristics to prevent false merges. In [141] the solutions are stitched by re-applying the algorithm on the block overlaps. Flood-Filling Networks [91], which predict a single object mask at a time, can be scaled without the need for stitching. It is, however, necessary to apply the networks several times per object, with the exact number of inference steps depending on the size of the object. While the inference can be parallelized over objects, applying it to dense segmentation of neurons in a large data-set

requires computational resources that are unrealistic in an academic setting: it takes over a week of computation with hundreds to thousands of graphics processing units or tensor processing units[2] to segment all neurons in a TB-sized data-set [91, 124].

The Multicut has first been applied to instance segmentation problems in [94, 5] and to EM neuron segmentation in [6]. In these publications, the Multicut problem has been formulated as an ILP, see also Equation 2.2, and solved to optimality with a cutting planes based approach. However, being a NP-hard problem, this approach is not suitable for large problems. Consequently, many authors have proposed approximate solvers that do not solve the Multicut to optimality, but scale to problems of larger size.

An important class of approximate solvers is based on local search algorithms [122]: *Greedy Additive Edge Contraction* [102] starts by placing all nodes in their own partition element and putting the edges into a priority queue sorted by cost in descending order. It then draws edges from the queue, merges the incident nodes and updates the costs accordingly. It terminates when no more edges with positive cost exist. *Greedy Fixation* [122] proceeds in a similar fashion, but fills the priority queue according to absolute cost. If an edge with negative cost is drawn, it introduces a cannot link constraint between the corresponding partition elements. This constraint prevents merging the two elements later in the algorithm. *Cut, Glue and Cut* [20] operates in two phases: in the cut phase, the graph is recursively bipartitioned. In the glue and cut phase, pairs of neighboring clusters are visited and bipartitioned via max-cut. *Kernighan-Lin* [100] starts from an initial partition and tries to decrease the objective by moving nodes between the boundaries of neighboring partitions.

A different approach is taken by *Fusion Moves* [19], which iteratively fuses the current partition with new proposed partitions. In the context of structured learning for edge costs, [114] have also introduced a hierarchical Multicut solver. Further approximate solvers based on continuous relaxations of the ILP [159], message passing on a dual representation of the graph [201], dual re-weighting applied to local search algorithms [117] and discrete optimization heuristics [14] have been proposed. An interesting approach is taken by the *Decomposition Solver* [2]: it decomposes the problem into components separated by only repulsive edges and solves the problems for the individual components. This procedure does not sacrifice optimality but whether it improves efficiency depends on the structure of the problem. For example, Multicut problems arising in connectomics often have a dominating component when decomposed in this manner, see also Section 3.3. In contrast to the approximate solvers introduced so far, the (Lifted) Multicut solver proposed here, makes use of the spatial domain afforded by the underlying over-segmentation. This enables an efficient spatial decomposition of the problem and improves the efficiency of the solver significantly. However, it is only applicable if the Multicut problem arises from an over-segmentation, or has a similar spatial

[2] Special purpose hardware for neural network inference.

domain.

The Lifted Multicut has been introduced as an extension to Multicut in [87], see also Section 2.2.6. Some of the approximate Multicut solvers have already been adapted to it: Greedy Additive Edge Contraction in [102], Kernighan-Lin in [102] and Fusion Moves in [18].

## 3.3 Methods

To evaluate the performance of extant Multicut solvers and our proposed solver, we set up problems of different sizes. The problems are set up similar to [21], see also Section 2.2. Here, we use a CNN based on the 3d U-Net architecture [35], see also Section 2.2.1. The edge costs are estimated by averaging pixel-wise boundary probabilities, instead of using a random forest based on edge features.

The problem set-up is implemented in a distributed fashion to run on a compute cluster. We make heavy use of chunked data storage formats such as HDF5 or n5. See Section A.1.2 and Section A.1.1 for details.

### 3.3.1 Multicut Solver

Here, we propose a Multicut solver for very large problems, such as the ones arising from neuron segmentation in EM connectomics. We postulate that minimizing a global objective, even for very large segmentation problems, is superior to the local approaches pursued in previous work (cf. Section 3.2) and validate this claim experimentally in Section 3.4.2. To this end, we exploit the fact that our partition problems arise from a region adjacency graph. Hence, all nodes in the graph have a spatial domain afforded by their superpixels.

In more detail, our solver operates in multiple steps. First, it extracts sub-problems by tiling the input space with (overlapping) blocks and assigning nodes to sub-problems according to the overlap of their superpixels with the blocks. It then solves the sub-problems in parallel, using an existing solver, and reduces the global problem by merging the nodes that are unambiguously merged in the sub-solutions. The sub-problem extraction, solution and reduction can be iterated with increasing block size until the reduced global problem is feasible for an existing solver. See algorithm 2 for pseudo-code and Figure 3.2 for a graphical illustration.

The solver starts from a global problem, composed of a graph and edge costs (b), which are derived from a superpixel over-segmentation (a). In the first iteration step (arrow 2), it extract sub-problems (c) from the global problem by tiling the input space with (overlapping) blocks. A node is assigned to the blocks that have *any* overlap with its corresponding superpixel. Hence, a single node can be assigned to multiple sub-problems.

In the second iteration step (arrow 3) the sub-problems are solved in parallel and the binary

edge indicators are saved; (d) shows the sub-problem results with cut edges, i.e. edges to be preserved in the partition, indicated by bold lines and edges to be merged indicated by dotted lines.

In the third iteration step (arrow 4), the sub-results are projected to the global graph (e). Here, we differentiate three types of edges (f):

- *Connecting edges* (green lines), for which the incident nodes belong to different sub-problems.
- *Shared edges* (red lines), for which the incident nodes belong to multiple sub-problems.
- *Unique edges* (blue lines), for which the incident nodes belong to a single sub-problem.

Connecting edges are not merged in the reduction step. Shared edges are merged only if they are merged in all sub-solutions containing this edge. Unique edges are merged if they are merged in the corresponding sub-solution. The graph is reduced according to these merge decisions and costs for the new edges are computed by summation (g).

These three steps can be iterated with increasing block size until the reduced problem is feasible with an approximate solver. It is then solved (arrow 5) and the resulting partition (h) is projected to the initial problem (arrow 6) to obtain the segmentation (i).

Note that merges performed in the reduction step cannot be undone later. Due to this fact, the solver proposed here is approximate, even if an exact solver is used for the sub-problems. Hence, we take a conservative approach: we do not merge connecting edges and only merge shared edges if all corresponding sub-results vote for a merge. While the problem extraction is described for a tiling with overlapping blocks in this section, we have found that overlaps are not necessary in practice and have performed all experiments without it. Further note that the solver is only applicable if a spatial domain exists and can be used for problem extraction.

### 3.3.2 Lifted Multicut Solver

Here, we adapt the block-wise Multicut solver from the previous section for Lifted Multicut partitioning. Following the same steps, this solver extracts sub-problems by tiling the volume into blocks, solves these sub-problems in parallel and uses the solutions to contract nodes to reduce the problem size. This approach can be repeated for an increasing block size, until the reduced problem becomes feasible with another approximate solver.

To apply it to Lifted Multicut partition, we also extract lifted edges during the sub-problem extraction. Here, only lifted edges with incident nodes that are part of the same sub-problem are taken into account. In the reduction step, the costs of lifted edges whose incident nodes have not been merged are updated by summation. See algorithm 2 for pseudo-code.

**Data:** globalProblem, blockShape, overlap, nLevels
**Result:** nodePartition
1 problem = globalProblem;
**for** *l in nLevels* **do**
2  subProblems = extractSubproblems(problem, blockShape, overlap);
3  subSolutions = solveMulticutsInParallel(subProblems);
4  problem = reduceProblem(problem, subProblems);
  blockShape *= 2;
**end**
5 nodePartition = solveMulticut(problem);
6 nodePartition = projectToGlobalGraph(nodePartition, globalProblem);

**Algorithm 1:** Block-wise Multicut solver. A given problem is reduced by solving sub-problems and merging the graph accordingly for a given number of iterations. The reduced problem is then solved and projected back to the global solution. Line numbers correspond to the numbered arrows in Figure 3.2.

Note that lifted edges that are not part of any of the sub-problems at a given level will still be considered at a later stage. This strategy, where we ignore lifted edges crossing block boundaries, is in line with the idea that lifted edges contribute to the energy of the solution, but do not induce connectivity. The considerations about optimality and necessity of a spatial domain from Section 3.3.1 apply here as well.

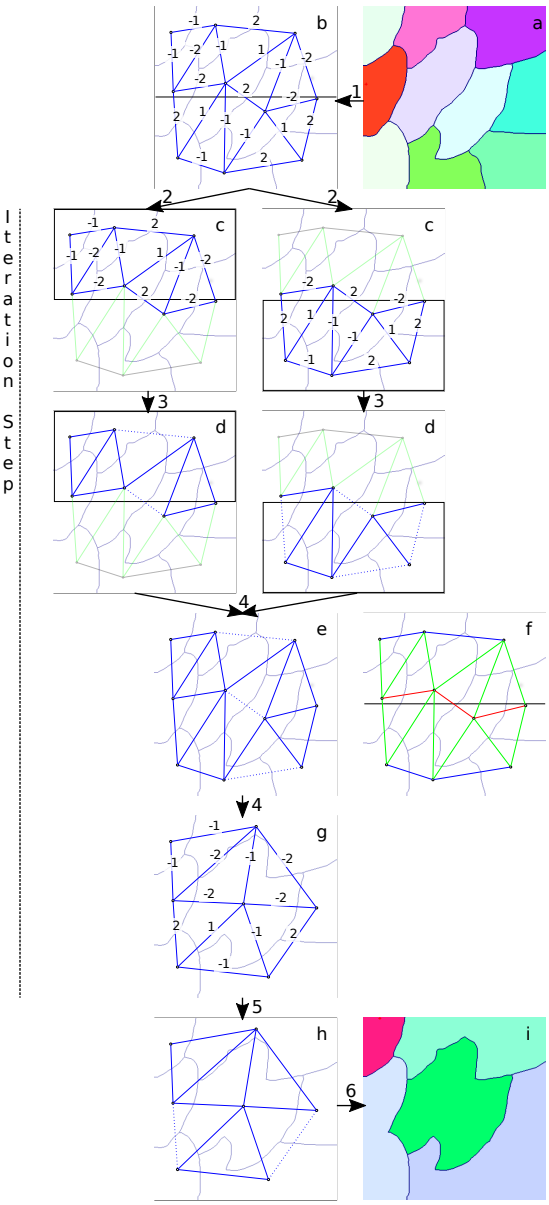



Figure 3.2: Block-wise Multicut solver: from the global problem (a, b) sub-problems are extracted (c) and solved in parallel (d). The sub-solutions are projected to the global problem (e, g), blue and red edges (f) are merged according to the sub-results. After a fixed number of iterations (or when feasible), the reduced problem is solved (h) and projected to a segmentation (i).

**Data:** graph $G$, edge costs $W_E$, lifted edges and costs $F$ and $W_F$, nLevels, blockShape
**Result:** node partition $P$
$\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F = G, F, W_E, W_F$;
**for** *n in nLevels* **do**
1 blocks = getBlocks(blockShape);
subPartitions = [];
`/* this for-loop is parallelized */`
**for** *block in blocks* **do**
2 $G_{sub}, W_E^{sub}$ = getSubproblem($\hat{G}$, $\hat{W}_E$, block);
3 $F_{sub}, W_F^{sub}$ = getLiftedEdges($G_{sub}$, $\hat{F}$, $\hat{W}_F$);
4 $P_{sub}$ = solveLiftedMulticut($G_{sub}, W_E^{sub}, F_{sub}, W_F^{sub}$);
subPartitions.append($P_{sub}$);
**end**
5 $\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F$ = reduceProblem($\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F$, subPartitions);
blockShape *= 2;
**end**
$P$ = solveLiftedMulticut($\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F$);
$P$ = projectToInitialGraph($G$, $P$);

**Algorithm 2:** Block-wise Lifted Multicut solver. (1): getBlocks tiles the volume with blocks of size blockShape. (2): getSubproblem extracts the sub-graph and corresponding edge costs from the given block. (3): getLiftedEdges extracts the lifted edges, which have both incident nodes in the given sub-problem, and costs. (4): solveLiftedMulticut solves the lifted multicut problem using an approximate solver. (5): reduceProblem: reduces the graph by merging nodes according to the sub-partition results. It also updates edge costs as well as lifted edges and their costs accordingly.

## 3.4 Results

In Section 3.4.1 we evaluate extant Multicut solvers for medium size problems in order to determine their suitability as internal solver in our proposed algorithm. We evaluate the performance and accuracy of the proposed solver on the same problems in Section 3.4.2. In Section 3.4.3, we evaluate its performance and scaling behavior on a much larger problem. The extension to the Lifted Multicut is evaluated in Section 3.4.4.

We set up three problems of medium size for evaluating the performance of extant solvers and our proposed Multicut solver. They are derived from three EM volumes made available by the CREMI challenge [66]. Each volume measures $8\times12\times12$ micrometer and shows a cutout from the fruit-fly brain imaged at $40\times4\times4$ nanometer [242]. In addition, each volume contains a $5\times5\times5$ micrometer crop with a ground-truth neuron segmentation. The entries for Level 0 in Table 3.2 contain the properties of the Multicut problems.

### 3.4.1 Performance of extant Multicut Solvers

We evaluate the performance of several Multicut solvers on the level 0 problems from Table 3.2. We use the local search based algorithms Greedy Additive Edge Contraction (*gaec*), Greedy Fixation (*gf*), Kernighan Lin (*kl*) and Cut, Glue and Cut (*cgc*). We also use the Fusion Moves solver (*fm*), indicating the solver used for sub-problems by concatenating its shorthand. In addition, the ILP based solver (*ilp*) is used as internal solver for *fm*; by itself it does not scale to the problems solved here. We warm-start *kl* with the solution from *gaec* and warm-start *fm* and *cgc* with the solution from *kl*. Note that the decomposition solver is not suited to speed-up the problems here, because they have one component of dominating size, see Table 3.2.

The results for all solvers and the three samples are shown in Table 3.1. We report the runtime, the energy (Equation 2.2) and the validation score. This score is computed on cutout of the data, for which ground-truth segmentation is available. We use the same score as used in the CREMI challenge, the geometric mean of Adapted Rand Error [13] and Variation of Information [140]. To avoid decreased scores from merges that happen outside of the ground-truth region, we run connected components on the cropped segmentation before scoring. Note that the scores reported here are significantly worse than the current state-of-the-art (Section 2.3.5), in spite of a very similar problem set-up. This can be explained by the fact that we have trained networks in a leave-one-out fashion, in order to ensure that all data is un-seen during prediction.

The two greedy solvers *gaec* and *gf*, run significantly faster than other solvers, but converge to sub-optimal solutions, which is also reflected in worse scores. The solver *gaec* is faster and converges to a better solution than *gf*. The *kl* solver converges after less than 4 minutes for all samples and can improve the energy and scores significantly. The *fm_kl* solver can further improve energy and scores at the expense of a significantly longer run-time. All other solvers

either do not improve energies over *kl* (*fm_gaec*) or have a significantly longer runtime with only marginal improvements (*cgc*, *fm_ilp*). In Section A.3.1 we further examine the runtime performance of the different solvers.

| | Sample A | | | Sample B | | | Sample C | | |
|---|---|---|---|---|---|---|---|---|---|
| Solver | Time [s] | Energy | Score | Time [s] | Energy | Score | Time [s] | Energy | Score |
| cgc | 4873.3 | -7.0698e+06 | 0.8819 | 4794.5 | -6.0740e+06 | 0.8510 | 4808.9 | -7.4642e+06 | 0.9074 |
| fm_gaec | 203.0 | -7.0688e+06 | 0.9005 | 289.8 | -6.0737e+06 | 0.8499 | 295.7 | -7.4636e+06 | 0.9072 |
| fm_ilp | 5726.3 | -7.0703e+06 | 0.8582 | 6777.1 | -6.0746e+06 | 0.8226 | 5494.8 | -7.4654e+06 | 0.8995 |
| fm_kl | 587.8 | -7.0690e+06 | 0.8809 | 4344.8 | -6.0745e+06 | 0.7774 | 2675.0 | -7.4642e+06 | 0.8528 |
| gaec | 4.6 | -7.0541e+06 | 0.9862 | 4.4 | -6.0659e+06 | 0.9221 | 5.8 | -7.4541e+06 | 0.9089 |
| gf | 25.2 | -7.0505e+06 | 1.0005 | 20.9 | -6.0593e+06 | 0.9274 | 28.4 | -7.4497e+06 | 0.9694 |
| kl | 78.3 | -7.0688e+06 | 0.9005 | 206.7 | -6.0737e+06 | 0.8499 | 165.6 | -7.4636e+06 | 0.9072 |

Table 3.1: Evaluation of extant approximate Multicut solvers on three problem arising from neuron segmentation in EM. For accurate runtime comparisons we have performed all experiments with a Intel Xeon Gold 6136 Processor, all solvers run in a single thread.

### 3.4.2 Performance and Accuracy of the Block-wise Multicut Solver

We evaluate our proposed block-wise solver (*BMC*) for the three medium sized problems. Table 3.2 shows the properties of the initial problems (Level 0) and the properties after one and two reduction steps (Level 1, 2). See Table A.1 for the configuration of the solver used here. The first step reduces the problem size by about one order of magnitude. The reduction afforded by the second step is only marginal. The reduced problems also change qualitatively: the decompositions[3] of the initial problems all have a component that contains over 90% of the nodes. The reduced problems, especially at Level 2, decompose into more and smaller components. The biggest component is still significantly larger than all others, but it does not dominate the complete problem as is the case at Level 0. This fact makes the decomposition solver (cf. Section 3.2) applicable for the reduced problems.

In Table 3.1, we compare our solver with the best Multicut results (*MC*) from Table 3.1. In addition, we compare with several baseline stitching algorithms: For *overlap*, we perform stitching by largest overlap [179, 176], starting from the same tiling as *BMC*, but adding a halo of $5\times50\times50$ pixels to the blocks to achieve spatial overlaps. We also compare to *stitch-mc*, where we solve the Multicut problems arising from overlaps and stitch based on their solutions,

[3] Obtained by thresholding at cost zero and applying connected components.

| Sample | Level | # Nodes | # Edges | # Components | Max component size (%) |
|---|---|---|---|---|---|
| A | 0 | 686,232 | 4,551,762 | 8,201 | 98.23 |
| | 1 | 29,109 | 171,100 | 16,234 | 31.07 |
| | 2 | 22,100 | 122,357 | 17,369 | 09.31 |
| B | 0 | 584,336 | 3,736,610 | 30,943 | 92.73 |
| | 1 | 72,299 | 391,192 | 56,021 | 15.89 |
| | 2 | 64,622 | 335,687 | 58,414 | 06.19 |
| C | 0 | 709,171 | 4,611,307 | 30,698 | 93.06 |
| | 1 | 75,188 | 439,740 | 56,289 | 16.82 |
| | 2 | 65,857 | 372,910 | 58,776 | 05.70 |

Table 3.2: Properties of the Multicut problems: the number of nodes and edges in the graph, the number of connected components when thresholding edges at cost 0 and the size of the largest component as percentage of the number of nodes. Level 0 are the initial problems, Level 1 and 2 after one and two iterations of problem reduction.

similar to the approach in [141]. Finally, we compare to *greedy*, where the sub-results are stitched by merging attractive edges between the blocks (cf. connecting edges described in Section 3.3.1).

We observe that the solution of *BMC* is significantly better than the *MC* solution for Sample A, on par for Sample B and slightly worse for Sample C. The improved results for Sample A are most likely an artifact due to low quality edge costs caused by EM imaging defects, see Section A.3.2 for a closer discussion. Nevertheless, we observe that the *BMC* solutions do not differ significantly in quality compared to *MC*, while being able to solve the problem 10 to 20 times faster. In contrast, the segmentation quality does suffer for the baseline stitching approaches. Note that we report the best value over a range of overlap / merge thresholds for the *overlap* / *greedy* stitching approaches. Still, their score is significantly worse on Samples B and C. The results of *stitch-mc* are even worse. We assume that the problem resulting from the overlap of $5 \times 50 \times 50$ pixels are too small to yield an informative segmentation problem. We have not optimized this parameter.

In addition, we compare different configurations for the *BMC* solver in Section A.3.2. Overall, the algorithm is fairly robust against the choice of parameters; the block size is the most important one. We also show in Table A.3 that the decomposition solver can indeed be used to speed up solving the global problem.

| | Sample A | | | Sample B | | | Sample C | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | Time | Energy | Score | Time | Energy | Score | Time | Energy | Score |
| MC | 587.8 | -7.0690e+06 | 0.8809 | 4344.8 | -6.0745e+06 | 0.7774 | 2675.0 | -7.4642e+06 | 0.8528 |
| BMC | 67.0 | -7.0524e+06 | 0.5624 | 182.0 | -6.0419e+06 | 0.7821 | 139.0 | -7.4346e+06 | 1.0441 |
| overlap | - | -7.0314e+06 | 0.8148 | - | -6.0341e+06 | 1.0600 | - | -7.4393e+06 | 1.3189 |
| greedy | - | -6.8761e+06 | 0.9649 | - | -5.8962e+06 | 1.2288 | - | -7.3447e+06 | 1.3022 |
| stitch-mc | - | -6.4730e+06 | 1.2370 | - | -5.5767e+06 | 1.3029 | - | -7.2528e+06 | 1.5072 |

Table 3.3: Comparison of different methods on the three problems extracted from CREMI. For *MC*, we report the result of *fm_kl* from Table 3.1. *BMC* is our proposed block-wise solver, *overlap* stitches sub-solutions by overlap, *greedy* by merging attractive edges between the sub-solutions and *stitch-mc* solves a Multicut on the overlap segmentation. All methods extract the sub-problems from blocks of size 25×256× 256 pixels and use the *fm_ilp* solver for sub-problems. *BMC* uses the *fm_kl* solver for the reduced global problem; the other methods use an overlap size of 5×50 ×50 pixels. For *overlap* and *greedy*, we compute the solution for different overlap / merge thresholds and report the result with the lowest energy. *BMC* uses 24 threads to solve sub-problems in parallel, using a Intel Xeon Gold 6136 Processor. We have not optimized the efficiency for the baseline methods and thus do not report runtimes.

### 3.4.3 Scaling Behavior of the Block-wise Multicut Solver

To evaluate the scaling behavior of our proposed solver, we construct a large Multicut problem. It is based on a 95 × 60 × 30 micrometer cutout from the full fruit-fly-brain [242], which is the same data-set used for extracting the smaller problems. For this volume, we do not have any ground-truth annotations available and can only compare the energies of different partitions. Here, we also evaluate our distributed implementation of the problem set-up, see Table 3.4 for an overview of the runtimes on a compute cluster. Table 3.5 shows the properties of the Multicut problem, both at the initial stage (Level 0) and after up to three iterations of problem reduction (Level 1 to 3).

Table 3.6 shows the results of our solver applied to the large Multicut problem. Here, we have used the Kernighan-Lin solver for sub-problems and have used the Decomposition solver, using Kernighan-Lin internally, for the global problems. The results show that the energy of the solution increases with the number of reduction iterations, while the runtime decreases due to the further reduction of the global problem. Our solver can solve this, previously infeasible, problem in under 40 minutes when using two or more reduction iterations.

We further investigate the scaling behavior of our solver, by plotting strong and weak scaling for sub-problem extraction and solution, problem reduction and global solution. To this end, we run these steps with a varying number of workers for a fixed problem size (strong scaling)

| Stage | #GPUs | #Cores | #Threads | Runtime [min] |
|---|---|---|---|---|
| Inference | 12 | - | - | 87.6 |
| Watershed | - | 200 | - | 15.1 |
| Problem | - | 200 | 8 | 171.0 |

Table 3.4: Runtimes for the problem set-up on a compute cluster for the large Multicut problem. Only the neural network inference uses gpus. For the other tasks we report the number of cores used for "map-like" tasks, which run on multiple nodes and the number of threads for "reduce-like" tasks, which run on a single node. See Section A.1.2 for details.

| Level | # Nodes | # Edges | # Components | max component size % |
|---|---|---|---|---|
| 0 | 85,012,571 | 556,282,289 | 5,260,500 | 90.77 |
| 1 | 13,650,761 | 82,095,065 | 8,256,407 | 32.13 |
| 2 | 11,260,960 | 66,021,137 | 8,767,569 | 16.34 |
| 3 | 10,237,143 | 58,574,608 | 9,062,027 | 07.62 |

Table 3.5: Properties of the large Mulicut problem. We report both the initial problem (Level 0) and the problems after up to three iterations of problem reduction. Similar to Table 3.2 we observe the largest reduction in the first iteration and observe that problems decompose better after reduction. For details on the problem reduction please refer to Table 3.6.

or construct problems of different size and process them with a fixed number of workers (weak scaling). For the construction of smaller problems, we use a spatial cutout of the complete volume. Here, we start with an initial size of $512 \times 512 \times 512$ pixels and iteratively increase the size in each dimension by a factor of two until the complete volume size is reached.

The scaling of the sub-problem extraction and solution is shown in Figure 3.3. In the strong scaling plot we see an almost linearly decrease of the runtime for up to 100 workers, until hitting a plateau for 200 workers or more. Here, the number of workers is given by the number of jobs times the number of threads per job. In the weak scaling plot we see a linear increase with the problem size, which is measured by the number of edges in the graph. For this task, individual jobs first load the complete graph, and then extract and solve the sub-problems in parallel using threads. When only few sub-problems have to be processed per job, the serial step of loading the graph starts to dominate the runtime, explaining the plateau in the strong scaling plot. Note that each job loads the complete graph in order to extract the sub-problems. It is possible, but more complex, to implement the problem extraction without having the

| Level | Sub-problems [min] | Reduction [min] | Global [min] | Total [min] | Energy |
|---|---|---|---|---|---|
| 1 | 6.8 | 21.4 | 80.1 | 108.3 | -8.60967e+08 |
| 2 | 0.7 | 4.1 | 5.9 | 38.9 | -8.60410e+08 |
| 3 | 0.4 | 2.0 | 1.0 | 36.4 | -8.60232e+08 |

Table 3.6: Block-wise Multicut solver results for a large Multicut problem. The columns "Sub-problems", "Reduction" and "Global" contain the runtimes for solving the sub-problems, reducing the problem and solving the reduced global problem at the given level. "Total" contains the full runtime for the given level. All computations were performed on Intel Xeon Gold 6136 Processors, using 32 jobs, each running with 16 threads, for solving the sub-problems. Reduction and global solver run in a single job with 16 threads.

complete graph in memory. Potentially, this would improve the strong scaling behavior. Given that the sub-problem extraction is currently not the limiting step (cf. Table 3.6), and that the weak scaling suggests applicability to much larger problem sizes, we have not done so yet.

Figure 3.4 shows the scaling of the problem reduction. The strong scaling plot shows an initial linear decrease of the runtime until hitting a plateau for more than four workers. The weak scaling plot shows roughly linear scaling with the problem size. The reduction task runs on a single node and parallelizes the computation of new edge costs and the serialization of the reduced problem with threads. The dominating sequential step is merging the nodes, which is implemented with a union find datastructure. In order to improve the strong scaling, one could switch to a parallel implementation of this datastructure [136, 195]. Unfortunately, to the best of our knowledge, such an implementation is not available in any standard library yet; we use the union find implementation provided by boost.

Figure 3.5 shows the scaling of the global problem solution. The strong scaling plot shows that the runtime is independent of the number of workers used, while the weak scaling shows roughly linear scaling with the problem size. Here, we use the Decomposition solver, which thresholds the graph edges at cost zero and extracts sub-problems from the resulting connected components, for parallelization. In this case, more threads do not improve the runtime because the biggest component is significantly larger than all others: for the Level 1 problem we find that it comprises of 32.13% of the nodes, the next biggest component is only made up of $4.2 * 10^{-4}\%$. Nevertheless, the reduction in size of the largest component affords a very significant speed up compared to the Kernighan-Lin solver, cf. Table A.4. Further improvement of the strong scaling of this step will prove challenging, because most extant Multicut solvers are sequential algorithms. An exception is the Fusion Moves solver, for which new proposal partitions can be generated in parallel. However, fusing the of proposals still has to be executed

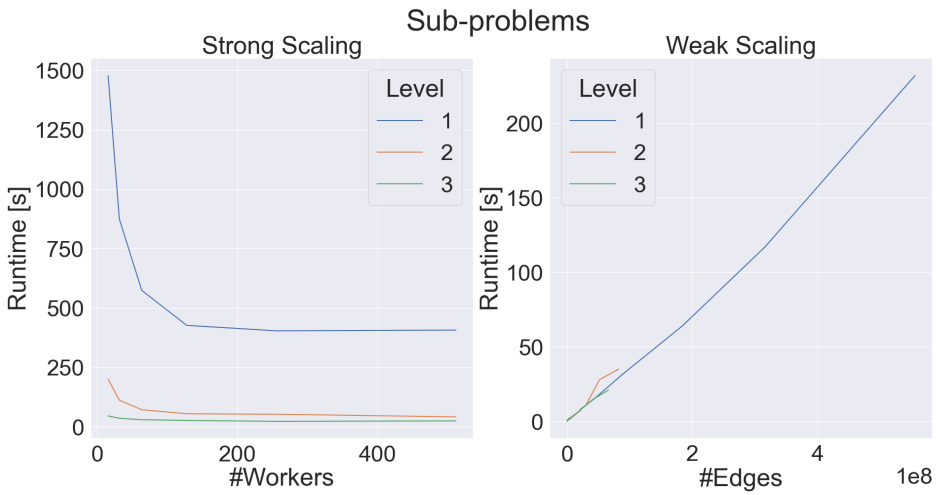



Figure 3.3: Scaling of the sub-problem extraction and solution. (Left) Strong scaling behavior when increasing the number of workers for the large Multicut problem. Here, the number of workers is the product of jobs and threads per job. (Right) Weak scaling behavior with fixed number of workers and increasing problem size, measured by the number of edges. Here, we use 128 workers. In both plots, the different levels are plotted separately.

sequentially. It is thus unclear if adopting this strategy will improve the scaling significantly, especially since the overall runtime of (serial) Fusion Moves is much larger compared to the Decomposition solver. We have only evaluated a sequential implementation of Fusion Moves here. Given the weak scaling behavior, the ability to decrease the problem size by further reduction iterations and the runtime of less than 40 minutes for a very large problem, we expect our solver to scale to the problems arising from even large microscopy segmentation data-sets as is.

### 3.4.4 Lifted Multicut solver

We evaluate the extension of our solver to the Lifted Multicut (*BLMC*) in Table 3.7. Here, we set up a Lifted Multicut problem for benchmarking purposes following a simplified procedure: we extract a central $1\times10\times$ 10 micrometer volume from the fruit-fly neural tissue data-set used in Section 4.4.2 and compute the region adjacency graph and costs for local edges based on mean accumulated boundary probabilities. Then, we introduce lifted edges between all nodes within a graph distance of two, setting their cost to the minimal edge cost along the weighted shortest path between the incident nodes. The resulting problem contains approximately 34,000 nodes, 244,000 regular edges and 2,384,000 lifted edges.

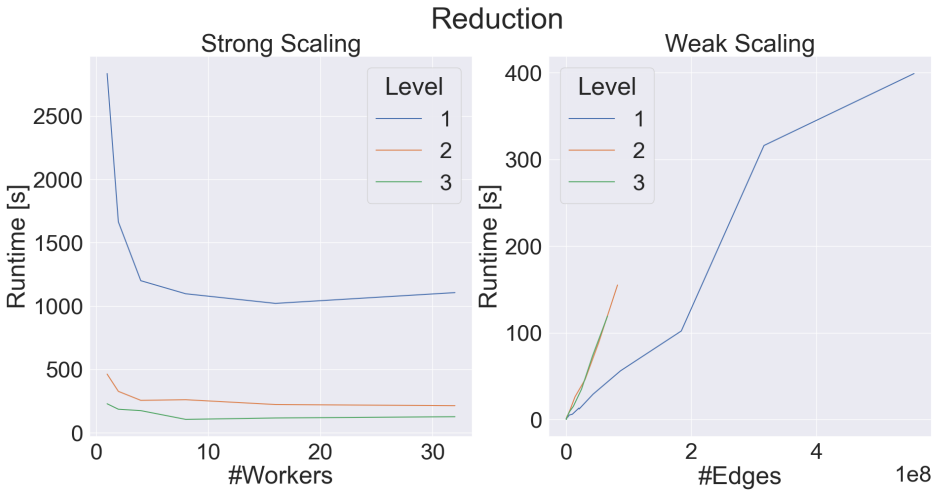



Figure 3.4: Scaling of the problem reduction. (Left) Strong scaling behavior when increasing the number of workers for the large Multicut problem. (Right) Weak scaling behavior with fixed number of threads and increasing problem size. Here, we use eight threads.

We compare to the approximate solvers Greedy Additive Edge Contraction [102] (*gaec*), Kernighan-Lin [102] (*kl*) and Fusion Moves [18] (*fm*). We warm-start *kl* with the *gaec* solution and warm-start *fm* with the *kl* solution. The *BLMC* algorithm uses *kl* to solve sub-problems and the reduced global problem.

The energies obtained by our solver are comparable with *kl* and *fm*, but it only needs a fraction of their runtime. It is almost as fast as *gaec*, which yields inferior energies on its own. While we only apply our solver for a single problem size here, we have used it for much larger Lifted Multicut problems in Chapter 4 and Section 6.1.

| | Energy | Time [s] |
|---|---|---|
| gaec | -1585593.5 | 2.03 |
| kl | -1645876.7 | 174.69 |
| fm_kl | -1645876.7 | 181.48 |
| BLMC | -1630274.3 | 3.29 |

Table 3.7: Our proposed Lifted Multicut solver compared to three other approximate solvers. The problem at hand arises from neuron segmentation in fruit-fly neural tissue (cf. Section 4.4.2) and contains approximately 34,000 nodes, 244,000 regular edges and 2,384,000 lifted edges.

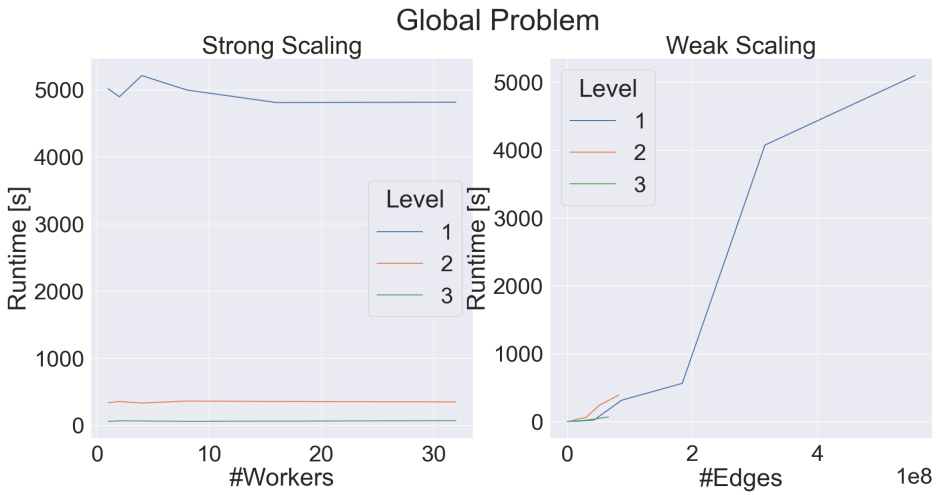



Figure 3.5: Scaling of the global problem solution. (Left) Strong scaling behavior when increasing the number of workers for the large Mulicut problem. The solution runs on a single node and uses the decomposition solver to parallelize sub-solutions via threads. (Right) Weak scaling behavior with fixed number of threads and increasing problem size. Here, we use eight threads.

## 3.5 Conclusion

We have introduced a hierarchical block-wise solver for the Multicut and Lifted Multicut in this chapter. By evaluation on data from a neuron segmentation challenge and a much larger data-set, we have demonstrated excellent scalability without a significant loss in segmentation quality. We have shown that this approach is advantageous compared to greedy local scaling strategies, which result in inferior segmentation results. The largest Multicut problem solved here contains about 85 million nodes and 556 million edges. We can solve it in under 40 minutes, with a total runtime of approximately five hours for the segmentation of the complete 150 GB raw volume. In Section 6.1, we use our solvers for even larger Multicut and Lifted Multicut problems to segment the cells in an EM volume containing a complete small animal. Hence, solving the global partition problem can no longer be considered the bottleneck for instance segmentation in the automated analysis of large microscopy image volumes.

The core idea of our algorithm – reducing the global partitioning problem instead of greedily stitching sub-solutions – is not restricted to the Multicut problem. It can also be applied to agglomerative clustering. However, given the inferior solutions obtained when applying only *gaec* here, this approach would most likely be detrimental to the segmentation quality.

# 4 Improving Segmentation with Lifted Priors

In Chapter 2, we have shown that Lifted Multicut can improve segmentation accuracy by introducing lifted edges in a local graph neighborhood with generic edge features. Here, we show that lifted edges can be used to express prior knowledge about the biological system underlying the segmentation problem; improving accuracy based on information not available otherwise due to their sparsity and/or large spatial distance.

This chapter, which is based on the publication [167], introduces a recipe to generate lifted edges from such priors, given that they can be attributed spatially. The procedure proves to be effective for a diverse set of problems, using rules such as "each cell should contain exactly one nucleus" or "different neuronal tissue types should not be mixed in the same compartment".

## 4.1 Introduction

Large-scale EM imaging is becoming an increasingly important tool in different fields of biology. The technique was pioneered by the efforts to trace the neural circuitry of small animals at synaptic resolution to obtain their connectome – a map of the neurons and the synapses connecting them. In the 1980's White et al. [228] mapped the complete connectome of *C. elegans* in a manual tracing effort which spanned over a decade. Since then, throughput has increased by several orders of magnitude thanks to innovations in EM image acquisition, such as multi-beam serial section EM [56], TEM camera arrays [27], hot-knife stitching [76] or gas cluster milling [75]. These innovations enable imaging larger volumes, up to the complete brain of the fruit-fly larva [57] and even the adult fruit-fly [242]. Recently, studies based on large-scale EM have become more common in other fields of biology as well [158, 164, 183, 172].

In light microscopy (LM), very large image volumes became routine even earlier [181, 115, 99], with Terabyte-scale acquisitions not uncommon for a single experiment. While the question of segmenting cell nuclei at such scale with high accuracy has been addressed before [3], cell segmentation based on membrane staining remains a challenge and a bottleneck in analysis pipelines.

Given the enormous amount of data generated, automated analysis of the acquired images is crucial; one of the key steps being instance segmentation of cells or cellular organelles. In recent years, the accuracy of automated segmentation methods has increased significantly due to the adoption of CNNs for semantic and instance segmentation [211, 42, 21, 67, 120, 91]. Still, it is not yet good enough to completely forego human proof-reading. Out of all microscopy image analysis problems, neuron segmentation in volume EM turned out to be particularly difficult [91] due to the small diameter and long reach of neurons and astrocytes. Many other EM segmentation problems have not yet been fully automated either. Aside from the complex morphology of objects of interest, EM sample preparation also renders the segmentation problems more difficult: the heavy metal staining used in the sample preparation labels all cellular components indiscriminately and forces segmentation algorithms to rely on membrane detection to separate them. The same problem arises in the analysis of light microscopy volumes with membrane staining, where methods originally developed for EM segmentation also achieve state-of-the-art results [65, 236].

One of the major downsides of CNN-based segmentation approaches lies in their limited field of view, making them overly reliant on local boundary evidence. Staining artifacts, alignment issues or noise can severely weaken this evidence and often cause *false merge errors* where separate objects get merged into one. On the other hand, membranes of cellular organelles or objects with a small diameter often cause *false split errors* where a single structure gets split into several objects in the segmentation.

Human experts avoid many of these errors by exploiting additional prior knowledge about the expected object shape or constraints from higher-level biology. Following this observation, several algorithms have recently been introduced to enable detection of morphological errors in segmented objects [178, 244, 53, 138]. By looking at complete objects rather than a handful of pixels, these algorithms can significantly improve the accuracy of the initial segmentation. In addition to purely morphological criteria, Krasowski et al. in [110] suggested an algorithm to exploit biological priors such as an incompatible mix of ultrastructure elements.

Building on such prior work, we introduce a general approach to leverage domain-specific knowledge in order to improve the accuracy of boundary based segmentation methods. Our method can be understood as a post-processing step for CNN predictions that pulls in additional sparse and distant sources of information. It allows to incorporate a large variety of rules, explicit or learned from data. It only requires that these rules can be expressed as the likelihood of certain locations in the image to belong to the same object or to different objects. These locations can be sparse and/or spatially distant.

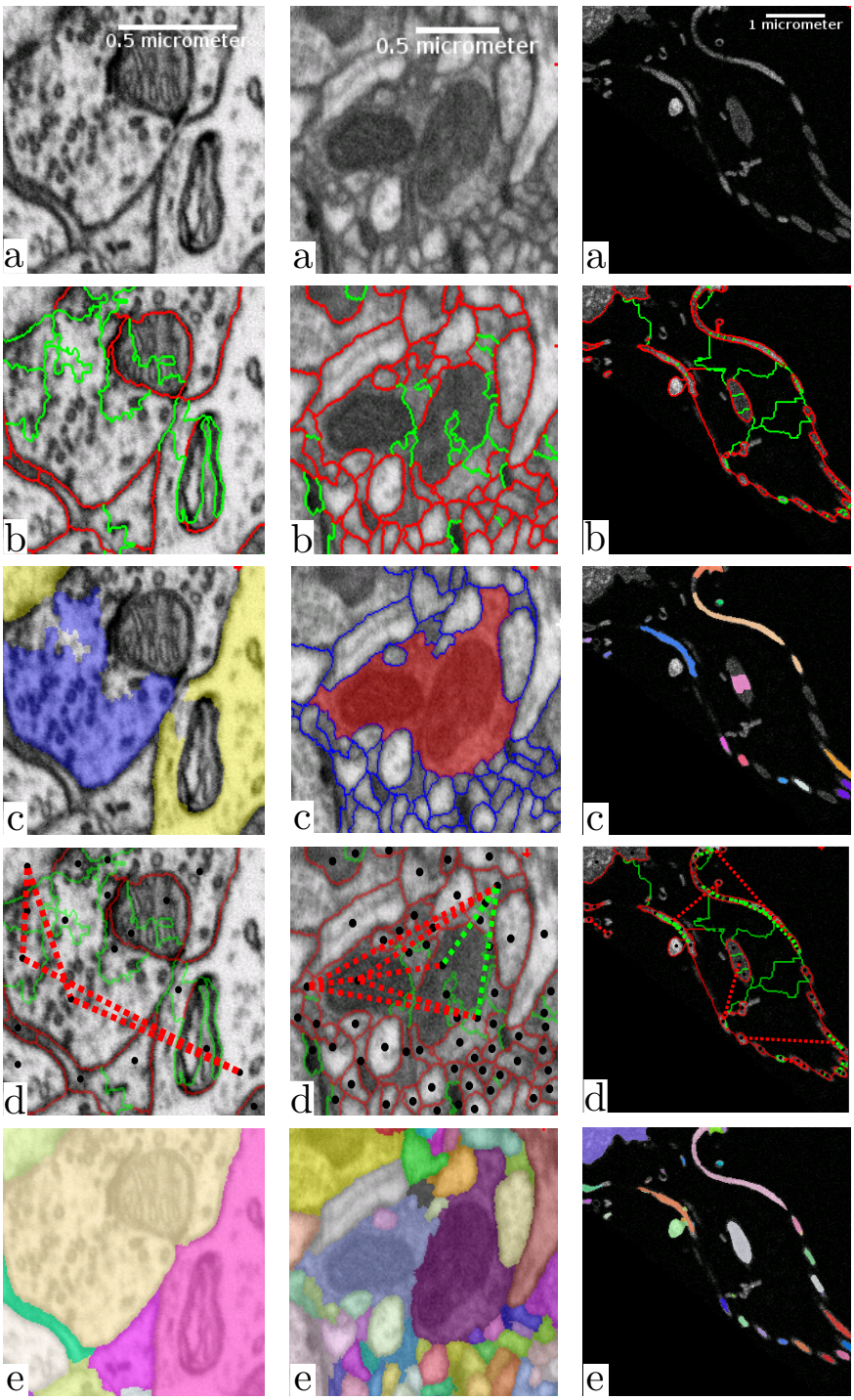

0.5 micrometer
0.5 micrometer
1 micrometer
a
a
a
b
b
b
c
c
c
d
d
d
e
e
e

Figure 4.1: Sparse lifted edges from domain knowledge for mammalian cortex (left), drosophila brain (middle) and sponge choanocytes (right). **a** Raw EM data. **b** Superpixel edges with attractive (green) and repulsive (red) costs derived from local boundary evidence. **c** Domain knowledge mapped to superpixels: axon (blue) and dendrite (yellow) attribution (left); an object with implausible morphology (red, center); shape priors for different organelles (one color per segmented organelle, right). **d** Attractive (green) and repulsive (red) lifted edges derived from **c**. **e** Lifted Multicut segmentation.

To incorporate such rules, we make use of the Lifted Multicut [87] graph partition problem. Briefly, this extension of the Multicut partition problem introduces additional edges between non-adjacent nodes, so-called lifted edges. They carry an energy contribution, but do not induce connectivity. See Section 2.2.6 for details.
When domain knowledge can be expressed as rules that certain locations must or must not belong to the same object, it can be distilled into lifted edges between the graph nodes corresponding to these locations. We show that the cost of such lifted edges can be derived from the strictness of the rules, which can colloquially range from "usually do / do not belong to the same object" to "always / never belong to the same object". We demonstrate the versatility of this approach by applying it to four segmentation problems, three in EM and one in LM. In these problems, we make use of very different kinds of domain knowledge:

- Based on the knowledge that axons are separated from dendrites in mammalian cortex, we use indicators of axon/dendrite attribution to avoid merges between different neuronal processes (Figure 4.1(left)).
- Based on the knowledge of plausible neuron morphology, we correct false merge errors in the segmentation of neuronal processes (Figure 4.1(center)).
- Based on the knowledge that certain organelles form long continuous objects, we reduce the number of false splits in instance segmentation of sponge choanocytes (Figure 4.1(right)).
- Based on the knowledge that a cell should only contain one nucleus, we improve the segmentation of plant lateral root cells (Figure 4.5).

## 4.2 Related Work

Neuron segmentation for connectomics has been the main driver of the recent advances in boundary-based segmentation for microscopy. As described in Section 1.1.2 and Section 2.1,

most methods follow a three step procedure: they predict boundaries, then compute a superpixel over-segmentation and finally agglomerate the superpixels to obtain a segmentation.
Krasowski et al. [110] showed that this three-step procedure can be modified to incorporate sparse biological priors in the superpixel agglomeration step. They use the Asymmetric Multi-Way Cut (AMWC) [113], a generalization of the Multicut for joint graph partition and node labeling. Their method is based on the fact that, given the field of view of modern electron microscopes, axon- and dendrite-specific ultrastructure should not belong to the same neuronal compartments in mammalian cortex. While this approach can be generalized to other domain knowledge, it has two important drawbacks. First, it cannot encode attractive information. Second, it is only applicable when the information is of semantic nature. For example, morphology-based false merge correction does not fit this category, because it detects segmentation errors rather than providing semantic node labels.
The Lifted Multicut formulation has been used for neuron segmentation before by [21] (Chapter 2). However, the lifted edges were added densely in a local graph neighborhood. Edge costs were not derived from domain knowledge but rather learned in a supervised approach. These lifted edges made the segmentation algorithm more robust against single missing boundaries, but did not counter the problem of the limited field of view of the boundary predictor or prevent biologically implausible objects. Note that this approach can be seen as a special case of the framework proposed here, using generic, but weak knowledge about local morphology and graph structure of segments. Besides Lifted Multicut, the recently introduced Mutex Watershed [233, 231] (Chapter 5) and generalized agglomerative clustering [16] can also exploit long-range information.
While all the listed methods demonstrate increased segmentation accuracy, they do not offer a general recipe on how to exploit domain-specific knowledge in a segmentation algorithm. We propose a versatile framework that can incorporate such information from diverse sources by mapping it to sparse lifted edges in the Lifted Multicut problem.

## 4.3 Methods

In the following, we describe our general recipe to map domain-specific knowledge to the lifted edges, see Section 4.3.1. We then describe the four specific applications, each using a different source of knowledge. In general, we set up the local problem, i.e. the region adjacency graph and local edge costs, similar to Section 2.2. For a review of the Lifted Multicut, please refer to Section 2.2.6.

### 4.3.1 Sparse Lifted Edges

Our main contribution is a general recipe how to express domain-specific knowledge via sparse lifted edges. They are only added between graph nodes where attribution of this knowledge is possible. The right side of Figure 4.2 illustrates this idea: nodes with attribution are shown by red and blue segments and sparse lifted edges by green dashed lines. The left side shows the approach of [21] (Chapter 2), where lifted edges are introduced between all nodes in a local graph neighborhood.

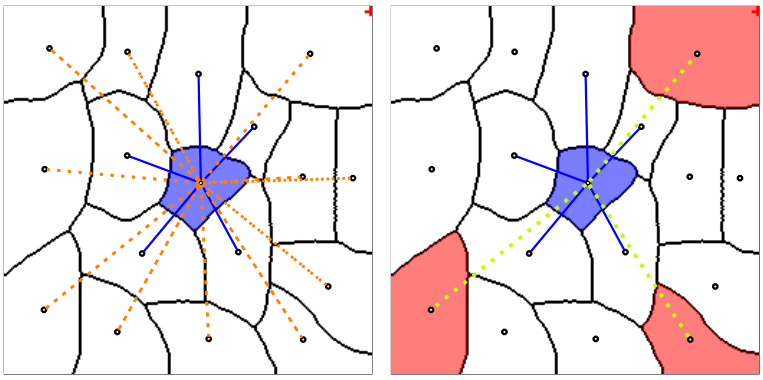


Figure 4.2: (Left) Graph neighborhood of a single node (blue shaded segment) with local edges (blue lines) and dense lifted edges (orange dotted edges). (Right) Neighborhood with sparse lifted edges (green dotted edges), connecting nodes with domain knowledge attribution (blue and red shaded segments).

The sparse lifted edges are constructed in several steps, see also Figure 4.1: we compute superpixels, construct the corresponding region adjacency graph and derive edge costs from local boundary evidence. Figure 4.1(b) shows the regular graph edges, marked green if attractive and red if repulsive. Then, we map the domain specific knowledge to nodes of the graph, as shown in (c). We then introduce sparse lifted edges between nodes with such attribution, shown as dashed lines in (d). The sign and strength of the lifted edge can either be learned supervisedly or derived explicitly, always indicating the likelihood of incident nodes being part of the same segment or not. In Figure 4.1 (d), attractive lifted edges are marked green, repulsive lifted edges red. If the likelihoods are expressed as (pseudo-) probabilities, they are transformed into signed costs via Equation 2.1. Finally, we solve the resulting Lifted Multicut problem to obtain an instance segmentation, see Figure 4.1(e).

**Mouse Cortex Segmentation, EM** This application shows how our method can be used to incorporate the axon/dendrite attribution first introduced in [110]. We detect the axon-

and dendrite-specific elements and map them to graph nodes in the same way as [110], see Figure 4.1(c), with blue shading for axon and yellow for dendrite attribution. The difference to [110] comes in the next step: instead of introducing semantic node labels for "axon" and "dendrite" classes, we add repulsive lifted edges between nodes with different semantic attribution. The lifted edge costs are derived from an edge classifier trained in a supervised fashion. Section 4.4.1 offers more details on the problem set-up and results.

**Drosophila brain segmentation, EM** For neurons in the insect brain, the axon/dendrite separation is not pronounced and the approach described in the previous paragraph cannot be applied. Instead, morphological information can be used to identify and resolve errors in segmented objects. This was first demonstrated by [178], where a CNN was trained on downsampled segmentation masks to detect merge errors. Meirovitch et al. in [141] detect merge errors with a simple shape-based heuristic and then correct these with a MaskExtend algorithm. Similarly, Zung et al. [244] have combined CNN-based error detection and flood filling network-based correction. In their formulation both false merge and false split errors can be corrected. Recently, [53, 138] have introduced an approach based on CNN error detection followed by a simple heuristic to correct false merges and Lifted Multicut graph partitioning to correct false splits.
This prior work convincingly demonstrates that false segmentation merge errors can be reliably detected. Hence, we concentrate our efforts on *error correction*, emulating the detection step with an oracle. We start from an initial segmentation and skeletonize all objects in this segmentation. Next, for all skeletons, the oracle predicts for all paths between terminal nodes if a false merge is present, i.e. whether the path crosses a boundary that was missed in the initial segmentation. The oracle predictions are probabilistic and we evaluate the performance of our method for different levels of noise applied to these predictions. We introduce lifted edges only for the objects that contain at least one path with a false merge according to the oracle. For these objects, we introduce repulsive lifted edges between the terminals of paths with false merge predictions and attractive lifted edges between the terminals of all other paths. We use the pseudo-probabilities generated by the oracle to compute costs for these lifted edges using Equation 2.1. See Figure 4.1 for an example: the red object in the middle of panel (c) has been identified as a false merge and the lifted edges introduced for this object are shown in panel (d).

**Sponge segmentation, EM** In this example, we tackle a segmentation problem in a sponge choanocyte chamber [153], see Section 6.2 for a summary of the study. These structures are built from several outer cells, the choanocytes, that enclose a cavity. These cells interact with one or two central cells per chamber via flagella, which are surrounded by a collar of microvilli. Our goal is to segment cell bodies, flagella and microvilli. This task is challenging

due to the large difference in sizes of these structures. Especially the segmentation of the thin flagella and microvilli is difficult. Without the use of shape priors, the Multicut algorithm splits them up into many small pieces.
In order to alleviate these false split errors, we predict which pixels in the image belong to flagella and microvilli and compute an approximate instance segmentation via thresholding and connected components. We map these instances to nodes of the region adjacency graph, see right column in Figure 4.1(c). Then, we introduce attractive lifted edges between the nodes attributed to the same component and repulsive lifted edges between nodes attributed to different components, see panel (d).

**Lateral root segmentation, LM** Finally, we tackle a challenging segmentation problem in light-sheet microscopy data: the segmentation of root cells in *Arabidopsis thaliana*. This data was imaged with two channels, one with a staining for cell membranes the other with a staining for nuclei. We use the first channel for boundary prediction, which forms the basis for the usual problem set-up. The second channel is used to segment individual nuclei. We then express the rule that a cell should only contain a single nucleus using sparse lifted edges. To this end, repulsive lifted edges are introduce between nodes whose superpixels are mapped to different nuclei instances. See Section 4.4.4 for more details on the problem set-up and results that show the prevention of false merges in the cell segmentation.

## 4.4 Results

We study the proposed method on four different problems: i) neuron segmentation in murine cortex with priors from axon/dendrite attribution, ii) neuron segmentation in the drosophila brain with priors from morphology-based error detection, iii) instance segmentation in a sponge choanocyte chamber with priors from attribution of semantic classes, iv) cell segmentation in plant roots with priors from the "one nucleus per cell" rule. Table 4.1 summarizes the different problem set-ups. We evaluate segmentation quality using the variation of information (VI) [140], which can be separated into split and merge scores, and the adapted Rand index [13]. For all quality measures used here, a lower value corresponds to higher segmentation quality.

### 4.4.1 Mouse Cortex Segmentation, EM

We present results on a volume of murine somatosensory cortex that was acquired by FIBSEM at $5 \times 5 \times 6$ nanometer resolution. The same volume has already been used in [110] for a similar experiment. To ensure a fair comparison between the two methods for incorporating axon/dendrite priors, we obtained derived data from the authors and use it to set-up the

| Edges | Regular | Dense Lifted | Sparse Lifted |
|---|---|---|---|
| Drosophila EM | Mean boundary evidence | - | False merge oracle predictions |
| Murine EM | RF based on edge features | RF based on region/ clustering features | Axon/dendrite attribution |
| Sponge EM | Mean boundary evidence | - | Small class semantic segmentation |
| Arabidopsis LM | Mean boundary evidence | - | Nucleus instance segmentation |

Table 4.1: Overview of the four problem set-ups. RF stands for random forest.

segmentation problem.
This derived data includes probability maps for neuron membranes, mitochondria, axon and dendrite attribution as well as a watershed over-segmentation and a ground-truth instance segmentation. From this data, we set up the graph partition problem as follows: we build the region adjacency graph $G$ from the watersheds and compute costs for the regular edges with a random forest based on the edge and region appearance features, using the same features as [21]. Next, we introduce dense lifted edges up to a graph distance of three. We use a random forest based on features derived from region appearance and clustering to predict their costs, again see [21] for details. In addition to the region appearance features, which are only based on raw data in [21], we also take into account the mitochondria attribution here. Next, we map the axon/dendrite attribution to the nodes of $G$ and introduce sparse lifted edges between nodes mapped to *different* classes. We infer costs for these edges with a random forest based on features from the statistics of the axon and dendrite node mapping. We use the fusion move solver of [18] for optimizing the Lifted Multicut objective.
We divide the volume into a $1 \times 3.5 \times 3.5$ micrometer block that is used to train the random forests for edge costs and a $2.5 \times 3.5 \times 3.5$ micrometer block used for validation. The random forest predicting pixel-wise probabilities was trained by the authors of [110] on a separate volume, using ilastik [22].
We compare the Multicut and AMWC results from [110] with different variants of our methods, see Table 4.2. As a baseline, we compute the Lifted Multicut only with dense lifted edges and without features from mitochondria predictions (LMC-D). We compute the full model with dense and sparse lifted edges (LMC-S) with and without additional mitochondria features. In addition, we compare to an iterative approach (LMC-SI) similar to the error correction approach in Section 4.4.2. For this approach, we perform LMC-D segmentation first and introduce sparse lifted edges only for objects that contain a false merge (identified by presence of both axonic and dendritic nodes in the same object).
The LMC-D segmentation quality is on par with the AMWC, although it does not use any input

from the priors, showing the importance of dense lifted edges. Our full model with sparse lifted edges shows significantly better quality compared to LMC-D. Mitochondria-based features provide a small additional boost. The segmentation quality of the iterative approach LMC-SI is inferior to solving the full model LMC-S, demonstrating the importance of joint optimization of the full model with dense and sparse lifted edges.

| Method | VI-Split | VI-Merge | Rand Error |
|---|---|---|---|
| MC [110] | **0.3471** | 0.6347 | 0.0787 |
| AMWC [110] | 0.4578 | 0.4935 | 0.0754 |
| LMC-D | 0.4144 | 0.4445 | 0.0891 |
| LMC-S | 0.4133 | **0.3788** | **0.0362** |
| LMC-S (No Mitos) | 0.4038 | 0.3966 | 0.0363 |
| LMC-SI | 0.5054 | 0.3998 | 0.0586 |

Table 4.2: Variants of our approach compared to [110]. The Rand Error measures the over-all segmentation quality, VI-Split measures the degree of over-segmentation and VI-Merge the degree of under-segmentation. For all measures, lower scores corresponds to better segmentations.

### 4.4.2 Drosophila brain segmentation, EM

We test the false merge correction approach on a $68 \times 38 \times 44$ micrometer FIBSEM volume of the *Drosophila* medulla. The data was imaged at $8 \times 8 \times 8$ nanometer in [205], who also provide a ground-truth segmentation for the whole volume.
First, we train a 3D U-Net for boundary prediction on a separate $2 \times 2 \times 2$ micron volume imaged with the same microscope. We use this network to predict boundaries on the whole volume and run watershed over-segmentation based on these predictions. Then, we set up an initial Multicut problem with edge costs derived from the mean accumulated boundary evidence. We obtain an initial segmentation by solving this problem with the block-wise solver of [166] (Section 3.3.1).
In order to demonstrate segmentation improvement based on morphological features, we skeletonize all sufficiently large objects using the method of [118] implemented in [215]. We then predict false merges along all paths between skeleton terminal nodes, using the ground-truth segmentation as oracle predictor. Note that [53] have shown that it is possible to train a very accurate CNN to classify false merges based on morphology information in this set-up. Given these predictions, we set up the Lifted Multicut problem by selecting all objects that

have at least one path with a false merge detection. For these objects, we introduce lifted edges between the terminal nodes of all paths in the skeleton. We derive costs for these edges from the false merge probability. Note that we use an imperfect oracle for some experiments, so the merge predictions are not absolutely certain. We solve two different variants of this problem: LMC-S, where we solve the complete problem using the solver introduced in Section 3.3.2. And LMC-SI, where we only solve the sub-problems arising for the individual objects with false merge detections. For this, we use the Fusion Moves solver of [18].
Table 4.3 compares the results of the initial Multicut (MC) with LMC-S and LMC-SI (using a perfect oracle) as well as the current state-of-the-art segmentation from Flood Filling Networks (FFN)[1] [91]. We adopt the evaluation procedure of [91] and use a cutout of size 23 $\times$ 19 $\times$ 23 micrometer for validation. We use two different versions of the ground-truth, the full segmentation and only a set of white-listed objects that were more carefully proofread. The FFN segmentation and validation ground-truth was kindly provided by the authors of [91]. The results show that our initial segmentation is inferior to FFN in terms of merge errors, but using LMC-SI we can improve the merge error to be even better than the FFN. Interestingly, LMC-SI performs better than LMC-S. We suspect that this is due to the fact that we only add lifted edges inside of objects with a false merge detection. Hence, LMC-S does not contain more information than LMC-SI, while having to solve a much bigger optimization problem.
Figure 4.3 shows the segmentation quality when using an imperfect oracle by tuning its F-score from 0.5 to 1.0 and measuring VI-split and VI-merge of the segmentation result. LMC-SI is fairly robust against this noise; it starts with a better VI-merge than MC, even for F-Score 0.5. Its VI-split gets close to MC for F-Scores larger than 0.75. In Figure 4.4 we show the initial segmentation and three examples of corrected merges.

### 4.4.3 Sponge segmentation, EM

The two previous applications mostly profit from repulsive information derived from ultra-structure or morphology. In order to show how attractive information can be exploited, we turn to an instance segmentation problem in a sponge choanocyte chamber. The EM volume was imaged with FIBSEM at a resolution of 15$\times$15$\times$ 15 nanometer. We aim to segment structures of three different types: cell bodies, flagella and microvilli. Flagella and microvilli have a small diameter, which make them difficult to segment with a boundary based approach. On the other hand, cell bodies have a much larger diameter and touch each other, which makes boundary based segmentation appropriate.
In order to set-up the segmentation problem, we first compute probability maps for boundaries as well as microvilli and flagella attribution using the autocontext workflow of ilastik [22]. We

[1] Note that the FFN was trained with more ground-truth data than our network.

| | Full | | | Whitelist | | |
|---|---|---|---|---|---|---|
| | VI-Split | VI-Merge | Rand Error | VI-Split | VI-Merge | Rand Error |
| MC | 1.5246 | 1.9057 | 0.6055 | 1.2189 | 0.6532 | 0.4143 |
| LMC-S | 1.6110 | 0.9405 | 0.4501 | 1.3050 | 0.2544 | 0.3891 |
| LMC-SI | 1.5773 | **0.5403** | 0.3335 | 1.2369 | **0.0122** | 0.2943 |
| FFN | **1.4653** | 0.6340 | **0.2838** | **0.8702** | 0.0559 | **0.1963** |

Table 4.3: Results on the drosophila EM data: Multicut (MC), Lifted Multicut solved for the complete problem (LMC-S), Lifted Multicut solved separately for sub-problems derived from false merge predictions (LMC-SI) and FFN [91]. We use a cutout for validation and evaluate with the complete ground-truth segmentation (Full) and a subset of proof-read objects (Whitelist).

set-up the Lifted Multicut problem by computing watersheds based on the boundary maps, extracting the region adjacency graph and computing costs for regular edges by mean accumulated boundary probability. We do not introduce dense lifted edges. For sparse lifted edges, we compute an additional instance segmentation of flagella and microvilli by thresholding the corresponding probability maps and running connected components. Then, we map the components of this segmentation to graph nodes. We connect nodes that are mapped to the same component with attractive lifted edges and nodes mapped to different components with repulsive lifted edges. We use the solver introduced in Section 3.3.2.
We run our segmentation approach on the whole volume of $70\times75\times$ 50 micrometer. For evaluation, we use three cutouts of size $15\times15\times1.5$ micrometer with ground-truth for instance and semantic segmentation. We split the evaluation into separate scores for objects belonging to the three different semantic classes. See Table 4.4 for the evaluation results, comparing the sparse Lifted Multicut (LMC) to the Multicut baseline (MC). As expected the quality of the segmentation of cell bodies is not affected, because we do not introduce lifted edges for those. The split rate in flagella and microvilli decreases significantly leading to a better overall segmentation for these structures.

### 4.4.4 Lateral root segmentation, LM

We segment cells in light-sheet image volumes of the lateral root primordia of *Arabidopsis thaliana* from [236]. These volumes are taken from a time-lapse video consisting of 51 time points, obtained in vivo in close-to-natural growth conditions. Each time point corresponds to a 3D volume of size $2048\times1050\times486$ voxels with resolution $0.1625\times0.1625\times0.25$ micrometer. The volume has two channels, one showing a membrane marker, the other a nucleus marker.


Figure 4.3: Quality of the error detection based Lifted Multicut when tuning the F-Score of the oracle false merge predictor from 0.5 to 1. We compare the approach where the complete problem is solved (LMC-S) to the approach where sub-problems from false merge predictions are solved individually (LMC-SI) with the Multicut baseline (MC).

We work on two selected time points: $T_{45}$ and $T_{49}$ taken from the later stages of development where the instance segmentation problem is more challenging due to growing number of cells. The time points have dense ground-truth segmentation for a $1000\times450\times200$ voxels cutout centered on the root primordia. Both cell and nucleus ground-truth segmentations are available. A variant of 3D U-Net [35] was trained in order to predict cell membranes and nuclei respectively. The two networks were trained on dense ground-truth from time points that are not part of our evaluation. Apart from the primary task of predicting membranes and nuclei respectively, both networks have an auxiliary task of predicting long-range affinities similarly to [120].
With these networks, we predict cell boundary probabilities and nucleus foreground probabilities. We use the nucleus predictions to obtain a nucleus instance segmentation by thresholding the probability maps at $0.9$ and running connected component analysis.
We compute superpixels from the watershed transform on the membrane predictions and compute costs for the regular edges via mean accumulated boundary probabilities. We set up lifted edges by mapping the nucleus instances to superpixels and connecting all nodes whose superpixels were mapped to different nuclei with repulsive lifted edges.
Table 4.5 shows the evaluation of segmentation results on the ground-truth cutouts. We can see that LMC-S clearly improves the merge errors as well the overall Rand Error while only marginally diminishing the split quality. Figure 4.5 shows an overview of the LMC result and

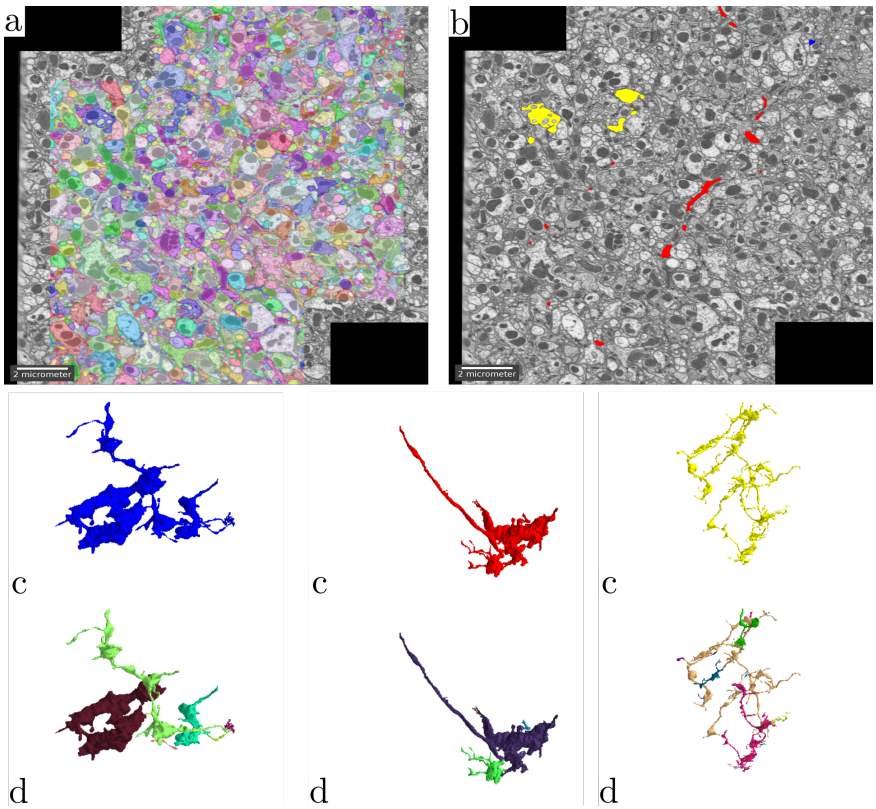



Figure 4.4: Drosophila EM segmentation. We detect merges in the initial segmentation (a) using an oracle. The red, blue and yellow segments in (b) were flagged as false merges. (c) and (d) show merged / correctly resolved objects.

two qualitative comparisons of MC and LMC results, highlighting merges that were prevented by LMC. Note that not all merges can be prevented, even if the nuclei were segmented correctly, because they can occur already in the watershed over-segmentation.

## 4.5 Discussion

Here, we have proposed a general purpose strategy to leverage domain-specific knowledge for instance segmentation problems in microscopy image analysis. This strategy expresses the domain knowledge in the (long-range) lifted edges of the Lifted Multicut graph partition problem. We apply the proposed strategy to a diverse set of difficult instance segmentation problems in

| Method | VI-Split | VI-Merge | Rand Error |
|---|---|---|---|
| Cells | | | |
| MC | 0.6058 | 0.0116 | 0.0783 |
| LMC | 0.6004 | 0.0116 | 0.0782 |
| Flagella | | | |
| MC | 0.4728 | 0.0812 | 0.1205 |
| LMC | 0.2855 | 0.0812 | 0.0429 |
| Microvilli | | | |
| MC | 3.1760 | 1.1101 | 0.7409 |
| LMC | 2.2745 | 1.1807 | 0.6973 |

Table 4.4: Quality of the sponge chonanocyte segmentation for cells, flagella and microvilli.

| | MC | | | LMC-S | | |
|---|---|---|---|---|---|---|
| | VI-Split | VI-Merge | Rand Error | VI-Split | VI-Merge | Rand Error |
| Timepoint 45 | 0.3596 | 0.5918 | 0.1641 | 0.3740 | 0.5527 | 0.1517 |
| Timepoint 49 | 0.4586 | 0.7116 | 0.2019 | 0.5153 | 0.5485 | 0.1873 |

Table 4.5: Comparison of Multicut and Lifted Multicut segmentation results for two time points taken from the light-sheet root primordia data.

light and electron microscopy and consistently show an improvement in segmentation accuracy. The improvement can be demonstrated even for imperfect prior information: segmentation quality only starts to degrade at fairly high error levels in the lifted edge costs, see results in Section 4.4.2.

For an application with ultrastructure based priors, we also observe that the Lifted Multicut based formulation yields higher quality results than the AMWC formulation of [110]. We believe that this is due to joint exploitation of dense short-range and sparse long-range information. A complete joint solution, with both lifted edges and semantic labels, has recently been introduced in [123]. We look forward to exploring the potential of this objective for the neuron segmentation problem.

Clearly, not every kind of domain knowledge can be expressed in this form, and the final

accuracy improvement depends on the information content of the prior knowledge. Also, our method cannot fix merge errors in the watershed segmentation underlying the graph, even if priors indicating such an error are available, see results in Section 4.4.4.
Similar to the findings of [113], we demonstrated that prevention of merge errors is more efficient than their correction: the joint solution of LMC-S is more accurate than iterative LMC-SI. However, not all prior information can be incorporated directly into the original segmentation problem. For these priors we demonstrate how to construct an additional resolving step which can also significantly reduce the number of false merge errors. In the future we plan to further improve our segmentations by other sources of information: matches of the segmented objects to known cell types, manual skeletons or correlative light microscopy imaging.

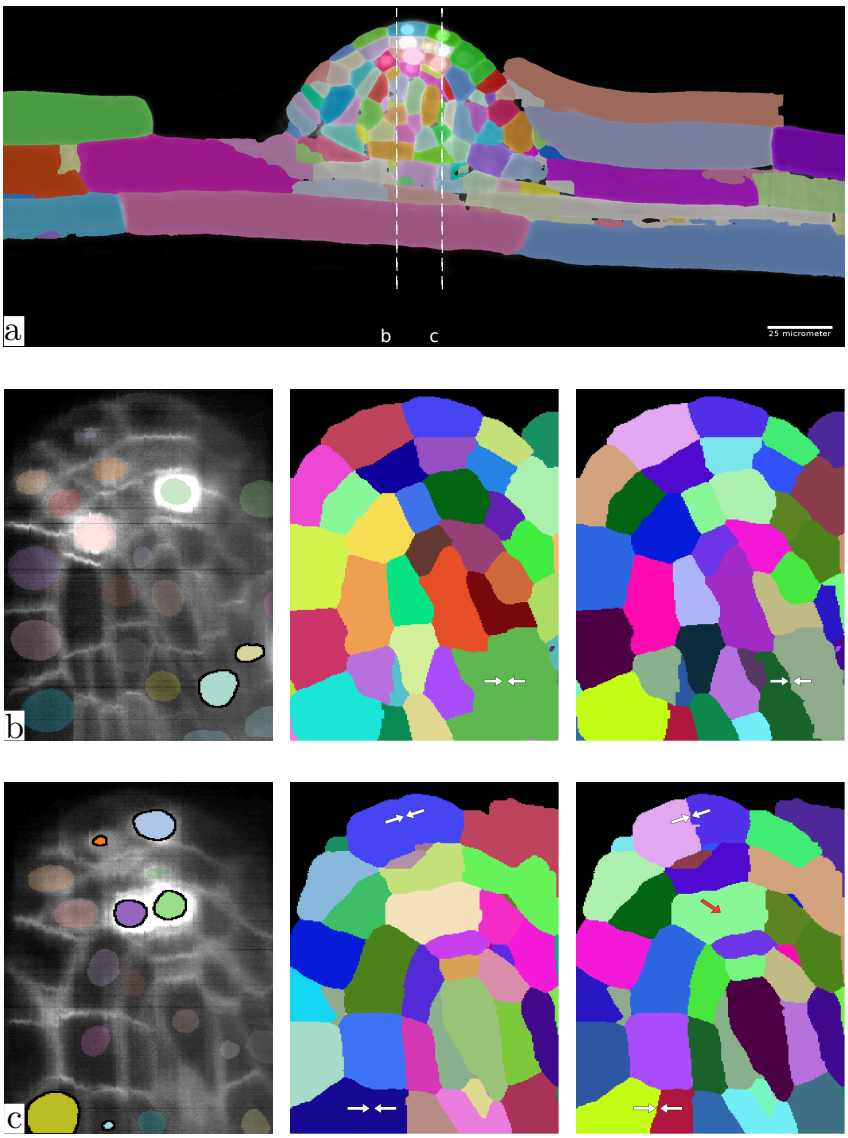



Figure 4.5: Segmentation results on light-sheet volumes of plant roots. (a) shows one complete image plane with membrane channel and LMC segmentation for timepoint 49. (b) and (c) show zoom ins of the yz plane with raw data and nucleus segmentation (left), MC segmentation (middle) and LMC segmentation (right) with avoided merge errors marked by white arrows. The two dashed lines in (a) show the cut planes for the zoom-ins. In (b) and (c) the nuclei instances inside of cells falsely merged in the MC segmentation are highlighted. Note that not all merge errors can be resolved by LMC; in some cases the watersheds are already merged, see red arrow in (c).

# 5 The Mutex Watershed

The segmentation approaches introduced so far have been based on an initial superpixel oversegmentation followed by a merging step. In certain cases it is difficult to generate satisfactory superpixels, for example because they tend to omit objects with a small diameter or do not adhere well to fuzzy boundaries. Thus, an approach that segments objects based on neural network predictions directly, skipping the superpixel generation, is desirable. Unfortunately (Lifted) Multicut segmentation does not scale to direct pixel-level segmentation for meaningful volume sizes, even with the approximate solvers introduced in Chapter 3.
Here, we introduce the Mutex Watershed, an algorithm that fulfills the above requirement: it introduces long-range repulsive interactions in a watershed-like algorithm. These interactions remove the need for seeds or thresholds, enabling segmentation directly from affinity predictions of a neural network. This chapter is based on the publication [233], which is joint work with Steffen Wolf, who has equally contributed to algorithm development and experiments.

## 5.1 Introduction

Most image partitioning algorithms are defined over a graph encoding purely attractive interactions. No matter whether a segmentation or clustering is then found agglomeratively (as in single linkage clustering / watershed) or divisively (as in spectral clustering or iterated normalized cuts), the user either needs to specify the desired number of segments or a termination criterion. An even stronger form of supervision is in terms of seeds, where one pixel of each segment needs to be designated as such either by a user or automatically. Unfortunately, clustering with automated seed selection remains a fragile and error-fraught process, because every missed or hallucinated seed causes an under- or oversegmentation error. Although the learning of good edge detectors boosts the quality of classical seed selection strategies (such as finding local minima of the boundary map, or thresholding boundary maps), non-local effects of seed placement along with strong variability in region sizes and shapes make it hard for any learned predictor to place *exactly one* seed in every true region.
In contrast to the above class of algorithms, Multicut / correlation clustering partitions vertices with both attractive and repulsive interactions encoded into the edges of a graph. Multicut has the great advantage that a "natural" partitioning of a graph can be found, without needing to

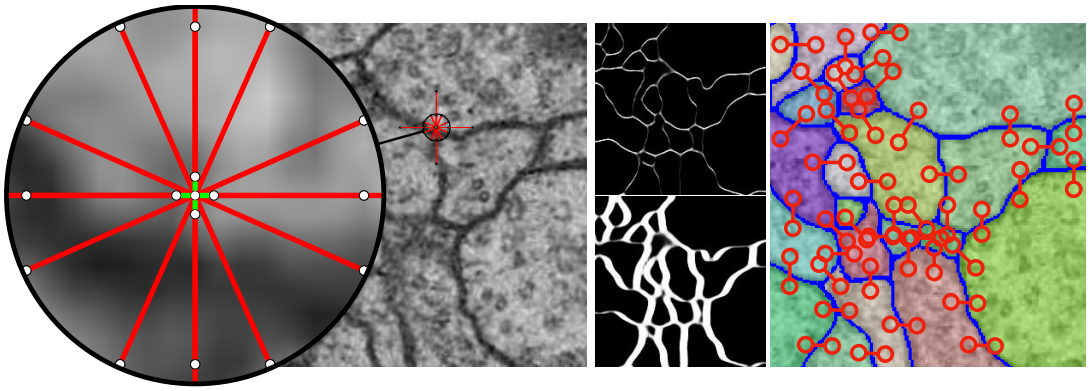


Figure 5.1: Left: Overlay of raw data from the ISBI 2012 EM segmentation challenge and the edges for which attractive (green) or repulsive (red) interactions are estimated for each pixel using a CNN. Middle: vertical / horizontal repulsive interactions at intermediate / long range are shown in the top / bottom half. Right: Active mutual exclusion (mutex) constraints that the proposed algorithm invokes during the segmentation process.

specify a desired number of clusters, or a termination criterion, or one seed per region. Its great drawback is that its optimization is NP-hard.

The main insight of this contribution is that when both attractive and repulsive interactions between pixels are available, then a generalization of the watershed algorithm can be devised that segments an image *without* the need for seeds, stopping criteria or thresholds. It examines all graph edges, attractive and repulsive, sorted by their weight and adds these to an active set iff they are not in conflict with previous, higher-priority, decisions. The attractive subset of the resulting active set is a forest, with one tree representing each segment. However, the active set can have loops involving more than one repulsive edge. See Figure 5.1 for a visual abstract.

In summary, our principal contribution, the Mutex Watershed, is a "best of both worlds" algorithm that combines the Multicut's desirable lack of hyperparameters with the small computational footprint of Kruskal-type watershed algorithm.

The algorithm is presented in Section 5.3. In Section 5.4 we evaluate it against very strong baselines. We choose a challenging data-set for neuron segmentation from EM image stacks as benchmark. For this task, watershed segmentation is a key component: EM staining only highlights membrane boundaries, discouraging the use of region cues for segmentation. By incorporating long-range repulsions into the watershed procedure, we can obtain an accurate segmentation from this step already, avoiding costly agglomeration as a post-processing step. In addition, we present exploratory results on the BSDS500, demonstrating the applicability of the proposed method to natural images.

## 5.2 Related Work

In the original watershed algorithm [220], seeds were automatically placed at all local minima of the boundary map. Unfortunately, this leads to severe over-segmentation. Defining better seeds has been a recurring theme of watershed research ever since. The simplest solution is offered by the seeded watershed algorithm [25]: It relies on an oracle (an external algorithm or a human) to provide seeds and assigns each pixel to its nearest seed in terms of minimax path distance. In the absence of an oracle, automatic seed selection is challenging because *exactly one* seed must be placed in every region. Simple methods, e.g. defining seeds by connected regions of low boundary probability, do not work: The segmentation quality is usually insufficient because multiple seeds are in the same region and/or seeds leak through the boundary.

This problem is typically addressed by biasing seed selection towards over-segmentation (with seeding at all minima being the extreme case). The watershed algorithm then produces superpixels that are merged into final regions by more or less elaborate post-processing. This works better than using watersheds alone because it exploits the larger context afforded by region adjacency graphs. Many criteria have been proposed to identify the regions to be preserved during merging, e.g. region dynamics [71], the waterfall transform [24], extinction values [214], region saliency [155], and $(\alpha, \omega)$-connected components [197]. A merging process controlled by criteria like these can be iterated to produce a hierarchy of segmentations where important regions survive to the next level. Variants of such hierarchical watersheds are reviewed and evaluated in [173].

These results highlight the close connection of watersheds to hierarchical clustering and minimum spanning trees/forests [146, 154], which inspired novel merging strategies and termination criteria. For example, [186] simply terminated hierarchical merging by fixing the number of surviving regions beforehand. [135] incorporate predefined sets of generalized merge constraints into the clustering algorithm. Graph-based segmentation according to [62] defines a measure of quality for the current regions and stops when the merge costs would exceed this measure. Ultrametric contour maps [9] combine the gPb (global probability of boundary) edge detector with an oriented watershed transform. Superpixels are agglomerated until the ultrametric distance between the resulting regions exceeds a learned threshold. An optimization perspective is taken in [105], which introduces $h$-increasing energy functions and builds the hierarchy incrementally such that merge decisions greedily minimize the energy. The authors prove that the optimal cut corresponds to a different unique segmentation for every value of a free regularization parameter.

An important line of research is based on the observation that superior partitionings are obtained when the graph has both attractive and repulsive edges. Solutions that optimally balance attraction and repulsion do not require external stopping criteria such as predefined

number of regions or seeds. This generalization leads to the NP-hard problem of correlation clustering or (synonymous) Multicut (MC) partitioning, see Section 2.2.4 for details.
Another beneficial extension is the introduction of additional long-range edges. Thanks to their larger field of view, the strength of these edges can often be estimated with greater certainty than is achievable for the local edges used in standard watersheds. This has been used in [241] to represent object size constraints by repulsive long-range edges, which is still an MC-type problem. When long-range edges are also allowed to be attractive, the problem turns into the more complicated lifted Multicut (LMC) [87], see Section 2.2.6 for details. Long-range edges are also used in [120], as a side loss for the boundary detection CNN; but they are not used explicitly in any downstream inference.
In general, striking progress in watershed-based segmentation has been achieved by learning boundary maps with CNNs. This is nicely illustrated by the evolution of neuron segmentation for connectomics. CNNs were introduced to this application in [90] and became, in much refined form [42], the winning entry of the ISBI 2012 Neuro-Segmentaion Challenge [13]. Boundary maps and superpixels were further improved by progress in CNN architectures and data augmentation methods, using U-Nets [180], FusionNets [177] or inception modules [21]. Subsequent post-processing with the GALA algorithm [161, 107], conditional random fields [213] or the Lifted Multicut [21] (Chapter 2) pushed the envelope of final segmentation quality. MaskExtend [141] applied CNNs to both boundary map prediction and superpixel merging, while flood-filling networks [91] eliminated superpixels all together by training a recurrent neural network to perform region growing one object at a time.
Most networks mentioned so far learn boundary maps on pixels, but learning works equally well for edge-based watersheds, as was demonstrated in [243, 169] using CNN-generated edge weights according to [211, 210]. Tailoring the learning objective to the needs of the watershed algorithm by penalizing critical edges along minimax paths [210] or end-to-end training of edge weights and region growing [234] improved results yet again.
Outside of connectomics, [15] obtained superior boundary maps from CNNs by learning not just boundary strength, but also its gradient direction. Holistically-nested edge detection [238, 108] couples the CNN loss at multiple resolutions using deep supervision and is successfully used as a basis for watershed segmentation of medical images in [32].
The present paper combines all these concepts (hierarchical clustering, attractive and repulsive interactions, long-range edges, and CNN-based learning) into a novel efficient segmentation framework. It can be interpreted as a generalization of [135], because we also allow for soft constraints (which can be overridden by strong attractive edges), and constraints are generated on the fly by a neural network rather than predefined. Our method is also related to greedy additive edge contraction (GAEC) according to [102] and greedy fixation [122], but we handle attractive and repulsive interactions separately and define edge strength between clusters by a maximum instead of an additive rule.

## 5.3 Methods

### 5.3.1 Definitions and notation

We consider the problem of clustering a graph $G(V, E^+ \cup E^-, W^+ \cup W^-)$ with both attractive and repulsive edge attributes. The scalar attribute $w_e^+ \in \mathbb{R}_0^+$ associated with edge $e \in E^+$ is a merge affinity: the higher this number, the higher the inclination of the two incident vertices to be assigned to the same cluster. Similarly, $w_e^- \in \mathbb{R}_0^+$ for $e \in E^-$ is a split tendency: the higher this number, the greater the tendency of the incident vertices to be in different clusters.
In our application, each vertex corresponds to one pixel in the image to be segmented. Two vertices may have no edge connecting them; or an attractive edge $e \in E^+$; or a repulsive edge $e \in E^-$; or two edges at the same time, one attractive and one repulsive. Edges can be either *local/short-range* (when connecting two pixels that are immediately adjacent in the image) or *long-range*.
The Mutex Watershed algorithm, defined in Section 5.3.3, maintains disjunct active sets $A^+ \subseteq E^+$, $A^- \subseteq E^-$, $A^+ \cap A^- = \emptyset$, that encode merges and mutual exclusion constraints, respectively. Clusters are defined via the "connected" predicate:

$$\begin{aligned} \forall i, j \in V: \qquad \Pi_{i \to j} &= \{\text{path } \pi \text{ from } i \text{ to } j \text{ with } \pi \subseteq E^+\} \\ \text{connected}(i, j) &\Leftrightarrow \exists \text{ path } \pi \in \Pi_{i \to j} \text{ with } \pi \subseteq A^+ \subseteq E^+ \\ \text{cluster}(i) &= \{i\} \cup \{j : \text{connected}(i, j)\} \end{aligned}$$

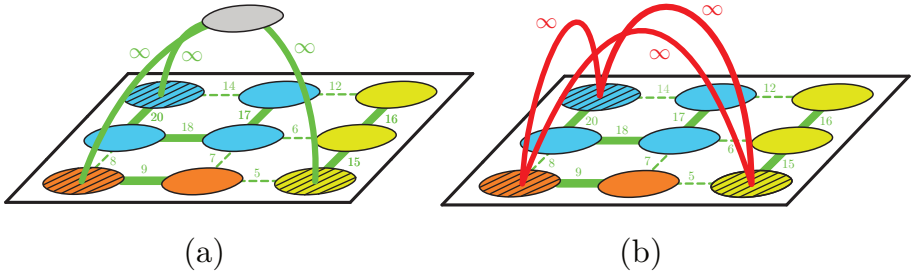
Conversely, the active subset $A^- \subseteq E^-$ of repulsive edges defines mutual exclusion relations by using the following predicate:

$$\begin{aligned} \text{mutex}(i, j) \quad \Leftrightarrow \quad & \exists\, e = (k, l) \in A^- \text{ with} \\ & k \in \text{cluster}(i) \text{ and } l \in \text{cluster}(j) \text{ and} \\ & \text{cluster}(i) \neq \text{cluster}(j) \end{aligned}$$

Admissible active edge sets $A^+$ and $A^-$ must be chosen such that the resulting clustering is consistent, i.e. nodes engaged in a mutual exclusion constraint cannot be in the same cluster: $\text{mutex}(i, j) \Rightarrow \text{not}\,\text{connected}(i, j)$. The "connected" and "mutex" predicates can be efficiently evaluated using a union find data structure.

### 5.3.2 Seeded watershed from a mutex perspective

One interpretation of the proposed method is in terms of a generalization of the edge-based watershed algorithm [145, 144, 146] or image foresting transform [60]. This algorithm can only ingest a graph with purely attractive interactions, $G(V, E^+, W^+)$. Without further constraints,




(a) (b)

Figure 5.2: Two equivalent representations of the seeded watershed clustering obtained using (a) a maximum spanning tree computation or (b) Algorithm 3. Both graphs share the weighted attractive (green) edges and seeds (hatched nodes). The infinitely attractive connections to the auxiliary node (gray) in (a) are replaced by infinitely repulsive (red) edges between each pair of seeds in (b). The two final clusterings are defined by the active sets (bold edges) and are identical. Node colors indicate the clustering result, but are arbitrary.

the algorithm would yield only the trivial result of a single cluster comprising all vertices. To obtain more interesting output, an oracle needs to provide seeds, namely precisely one node per cluster. These seed vertices are all connected to an auxiliary node (see Figure 5.2 (a)) by auxiliary edges with infinite merge affinity. A maximum spanning tree (MST) on this augmented graph can be found in linearithmic time; and the maximum spanning tree (or in the case of degeneracy: at least one of the maximum spanning trees) will include the auxiliary edges. When the auxiliary edges are deleted from the MST, a forest results, with each tree representing one cluster [146, 145, 60].

We now reformulate this well-known algorithm in a way that will later emerge as a special case of the proposed Mutex Watershed: we eliminate the auxiliary node and edges, and replace them by a set of infinitely repulsive edges, one for each pair of seeds (Figure 5.2 (b)). Algorithm 3 is a variation of Kruskal's MST algorithm operating on the seed mutex graph just defined, and gives results identical to seeded watershed on the original graph.

This algorithm differs from Kruskal's only by the check for mutual exclusion in the if-statement. Obviously, the modified algorithm has the same effect as the original algorithm, because the final set $A^+$ is exactly the maximum spanning forest obtained after removing the auxiliary edges from the original solution. In the sequel, we generalize this construction by admitting less-than-infinitely repulsive edges. Importantly, these can be dense and are hence much easier to estimate automatically than seeds with their strict requirement of only-one-per-cluster.

**Input:** weighted graph $G(V, E^+, W^+)$ and seeds $S \subseteq V$, such that $E^- = \{(s_i, s_j)|i, j \in 1, \ldots, |S|; i \neq j\}$ is the set of infinitely repulsive edges between all pairs of seeds;
**Output:** clusters defined by activated edges $A^+$;
**Initialization:** $A^+ = \emptyset$; $A^- = E^-$;
**for** $(i, j) = e \in E^+$ in descending order of $w^+$ **do**
  **if not** $\mathrm{connected}(i, j)$ **and not** $\mathrm{mutex}(i, j)$ **then**
    $A^+ \leftarrow A^+ \cup e$ ;
      ▷ merge $i$ and $j$ and inherit the mutex constraints of the parent clusters

**Algorithm 3:** Mutex version of seeded watershed algorithm.

### 5.3.3 Mutex Watersheds

We now introduce our core contribution: an algorithm that is empirically no more expensive than a MST computation; but that can ingest both attractive and repulsive cues and partition a graph into a number of clusters that does not need to be specified beforehand. There is no requirement of one seed per cluster, and not even of a hyperparameter that would implicitly determine the number of resulting clusters.

The Mutex Watershed, Algorithm 4, proceeds as follows: given a graph with sets of attractive and repulsive edges $E^+$ and $E^-$, with edge weights $W^+$ and $W^-$ respectively, do the following: sort all edges $E^+ \cup E^-$, attractive or repulsive, by their weight in descending order into a priority queue. Iteratively pop all edges from the queue and add them to the active set one by one, provided that a set of conditions are satisfied. More specifically, if the next edge popped from the priority queue is attractive and its incident vertices are not yet in the same tree, then connect the respective trees provided this is not ruled out by a mutual exclusion constraint. If on the other hand the edge popped is repulsive, and if its incident vertices are not yet in the same tree, then add a mutual exclusion constraint between the two trees.

The crucial difference to Algorithm 3 is that mutex constraints are no longer pre-defined, but created dynamically whenever a repulsive edge is found. However, new exclusion constraints can never override earlier, high-priority merge decisions. In this case, the repulsive edge in question is simply ignored. Similarly, an attractive edge must never override earlier and thus higher-priority must-not-link decisions.

**Input:** weighted graph $G(V, E^+ \cup E^-, W^+ \cup W^-)$;
**Output:** clusters defined by activated edges $A^+$;
**Initialization:** $A^+ = \emptyset$; $A^- = \emptyset$;
**for** $(i, j) = e \in E^+ \cup E^-$ in descending order of $W^+ \cup W^-$ **do**
  **if** $e \in E^+$ **then**
    **if not** $\mathrm{connected}(i, j)$ **and not** $\mathrm{mutex}(i, j)$ **then**
      merge(i, j): $A^+ \leftarrow A^+ \cup e$;
        ▷ merge $i$ and $j$ and inherit the mutex constraints of the parent clusters
  **else**
    **if not** $\mathrm{connected}(i, j)$ **then**
      addmutex(i, j): $A^- \leftarrow A^- \cup e$;
        ▷ add mutex constraint between $i$ and $j$

**Algorithm 4:** Mutex Watershed

### 5.3.4 Time Complexity Analysis

Before analyzing the time complexity of Algorithm 4, we first review the complexity of Kruskal's algorithm. Using a union-find data structure the time complexity of $\mathrm{merge(i,\ j)}$ and $\mathrm{connected(i,\ j)}$ is $\mathcal{O}(\alpha(V))$, where $\alpha$ is the slowly growing inverse Ackerman function, and the total runtime complexity is dominated by the initial sorting of the edges $\mathcal{O}(E \log E)$[46]. To check for mutex constraints efficiently, we maintain a set of all active mutex edges

$$M[C_i] = \{(u,\ v) \in A^- | u \in C_i \vee v \in C_i\}$$

for every $C_i = \mathrm{cluster}(i)$ using hash tables, where insertion of new mutex edges (i.e. $\mathrm{addmutex}$) and search have an average complexity of $\mathcal{O}(1)$. Note that every cluster can be efficiently identified by its union-find root node. For $\mathrm{mutex(i,\ j)}$ we check if $M[C_i] \cap M[C_j] = \emptyset$ by searching for all elements of the smaller hash table in the larger hash table. Therefore $\mathrm{mutex(i,\ j)}$ has an average complexity of $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$. Similarly, during $\mathrm{merge(i,\ j)}$, mutex constraints are inherited by merging two hash tables, which also has an average complexity $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$.
In conclusion, the average runtime contribution of attractive edges $\mathcal{O}(|E^+| \cdot \alpha(V) + |E^+| \cdot M)$ (checking mutex constrains and possibly merging) and repulsive edges $\mathcal{O}(|E^-| \cdot \alpha(V) + |E^-|)$ (insertion of one mutex edge) result in a total average runtime complexity of Algorithm 4:

$$\mathcal{O}(E \log E + E \cdot \alpha(V) + EM). \tag{5.1}$$

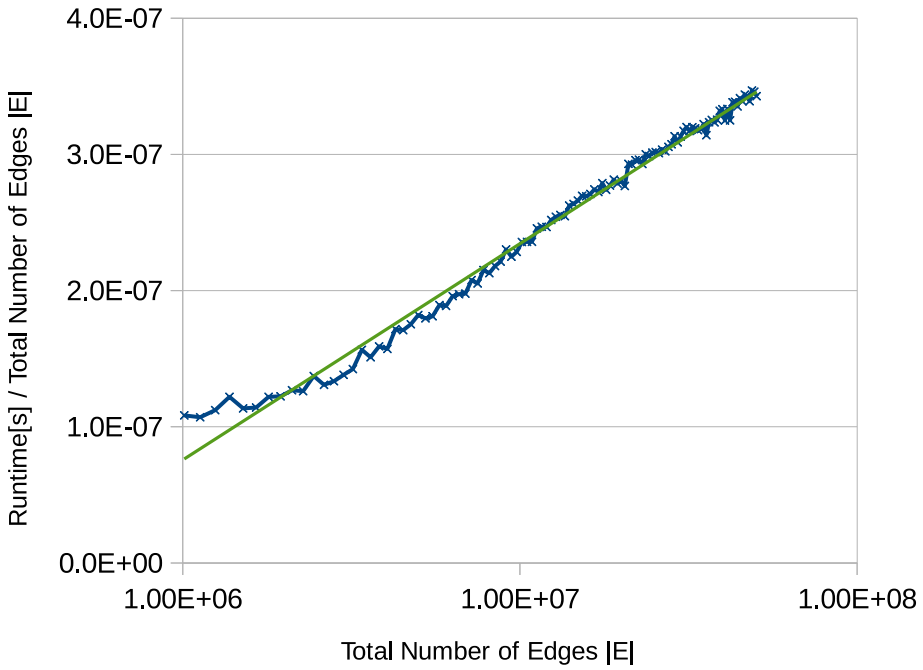



Figure 5.3: Runtime $T$ of Mutex Watershed (without sorting of edges) measured on differently sized sub-volumes of the ISBI challenge data, thereby varying the total number of edges $E$. We plot $\frac{T}{|E|}$ over $|E|$ in a logarithmic plot, which makes $T \sim |E|log(|E|)$ appear as straight line. A logarithmic function (green line) is fitted to the measured $\frac{T}{|E|}$ (blue crosses) with ($R^2 = 0.9896$). The good fit suggests that empirically $T \approx \mathcal{O}(E \log E)$.

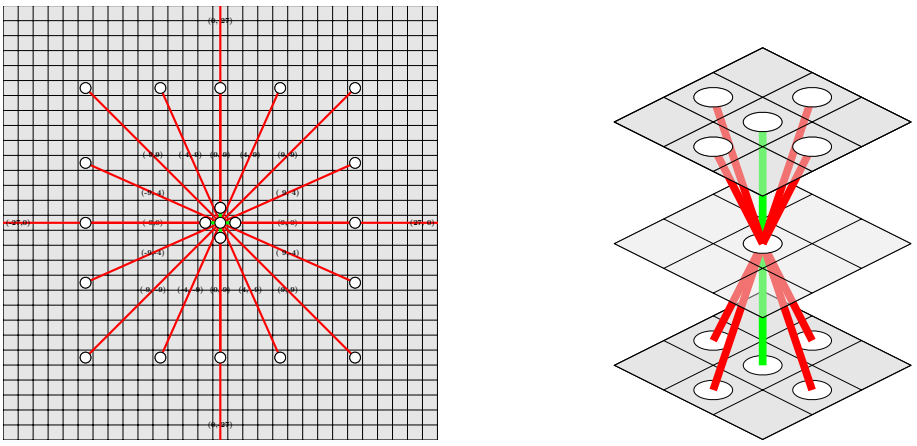
where $M$ is the expected value of $\min(|M[C_i]|, |M[C_j]|)$. Using $\alpha(V) \in \mathcal{O}(\log V) \in \mathcal{O}(\log E)$ this simplifies to $\mathcal{O}(E \log E + EM)$.
In the worst case $\mathcal{O}(M) = \mathcal{O}(E)$, the Mutex Watershed Algorithm has a runtime complexity of $\mathcal{O}(E^2)$. Empirically, we find that $\mathcal{O}(EM) \approx \mathcal{O}(E \log E)$ by measuring the runtime of Mutex Watershed for different sub-volumes of the ISBI challenge (see Figure 5.3), leading to an empirical complexity of $\mathcal{O}(E \log E)$.

## 5.4 Results

We evaluate the Mutex Watershed on the challenging task of neuron segmentation in EM image volumes. This application is of key interest in connectomics, the effort to reconstruct neural wiring diagrams spanning complete central nervous systems. Neuron segmentation from EM images is a challenging endeavor, since segmentation has to be based only on boundary information (cell membranes) and some of the boundaries are not very pronounced. Besides, cells contain membrane-bound organelles, which have to be suppressed in the segmentation. Some of the neuron protrusions are very thin, but all of those have to be preserved in the segmentation to arrive at the correct connectivity graph. While a lot of progress has been made recently, only manual tracing yields sufficient accuracy for correct circuit reconstruction [189]. We validated the Mutex Watershed the most popular neural segmentation challenge: ISBI2012 [13]. We estimate the edge weights using a CNN as described in Section 5.4.1 and compare with other entries in the leaderboard as well as with other common post-processing methods for the same network predictions in Section 5.4.2.

### 5.4.1 Estimating edge weights with a CNN

The common approach to EM segmentation is to predict which pixels belong to a cell membrane using a CNN. Different post-processing methods are used on top to obtain a segmentation, see Section 1.1.2 for an overview of these methods. The CNN can be either trained to predict boundary pixels [42, 21] or undirected affinities [120, 67], which express how likely it is for a pixel to belong to a different cell than its neighbors. In this case, the output of the network contains three channels, corresponding to left, down and next imaging plane neighbors in 3d. The affinities do not have to be limited to immediate neighbors - in fact, [120] have shown that introduction of long-range affinities is beneficial for the final segmentation, even if they are only used as auxiliary loss during training. Building on their work, we train a CNN to predict short and long-range affinities and then use those directly as weights for segmentation with the Mutex Watershed.
We estimate the affinities/edge weights for the neighborhood structure shown in Figure 5.4. To that end, we define local attractive and long-range repulsive edges. The choice of this structure has to be motivated by the underlying data - we use a different pattern for in-plane and between-plane edges due to the anisotropy of the validation data at hand. In more detail, we picked a sparse ring of in-plane repulsive edges and additional longer-range in-plane edges which were necessary to split regions reliably (see Figure 5.4a). We also added connections to the indirect neighbors in the lower adjacent slice to ensure correct 3D connectivity (see Figure 5.4b).
In total, $C^+$ attractive and $C^-$ repulsive edges are defined for each pixel, resulting in $C^+ + C^-$



(a) XY-plane neighborhood with local attractive edges, a sparse repulsive edges with approximate radius 9 and further long-range connections with distance 27

(b) Due to the high anisotropy of the data we limit the Z-plane edges to a distance of 1. The direct neighbors are attractive; the indirect neighbors are repulsive.

Figure 5.4: Local neighborhood structure of attractive (green) and repulsive (red) edges in the Mutex Watershed graph. Due to point symmetry to the origin, we only predict half of the directions with the neural network.

output channels in the network. We partition the set of attractive / repulsive edges into subsets $H^+$ and $H^-$ that contain all edges at a specific offset, attractive edges: $E^+ = \bigcup_c^{C^+} H_c^+$ and repulsive edges analogously. Each element of the subsets $H_c^+$ and $H_c^-$ corresponds to a specific channel predicted by the network. We further assume that weights take values in $[0, 1]$ and adopt the same conventions for attractiveness / repulsion as in Section 5.3. For more details on network architecture and training see Section A.4.1.
In our experiments, we pick a subset of repulsive edges, by using strides of 2 in the XY-plane in order to avoid artifacts caused by occasional very thick membranes. Note that this stride is not applied to local (attractive) edges, but only to long-range (repulsive) edges.

### 5.4.2 ISBI Challenge

The ISBI 2012 EM Segmentation Challenge [13] is the neuron segmentation challenge with the largest number of competing entries. The challenge data contains two volumes of dimensions $1.5 \times 2 \times 2$ microns with a resolution of $50 \times 4 \times 4$ nm per pixel. The groundtruth is provided as binary membrane labels, which can easily be converted to a 2D segmentation. To train a 3D model, we follow the procedure described in [21] (Section 2.2).
The test volume has private groundtruth; results can be submitted to the leaderboard. They are

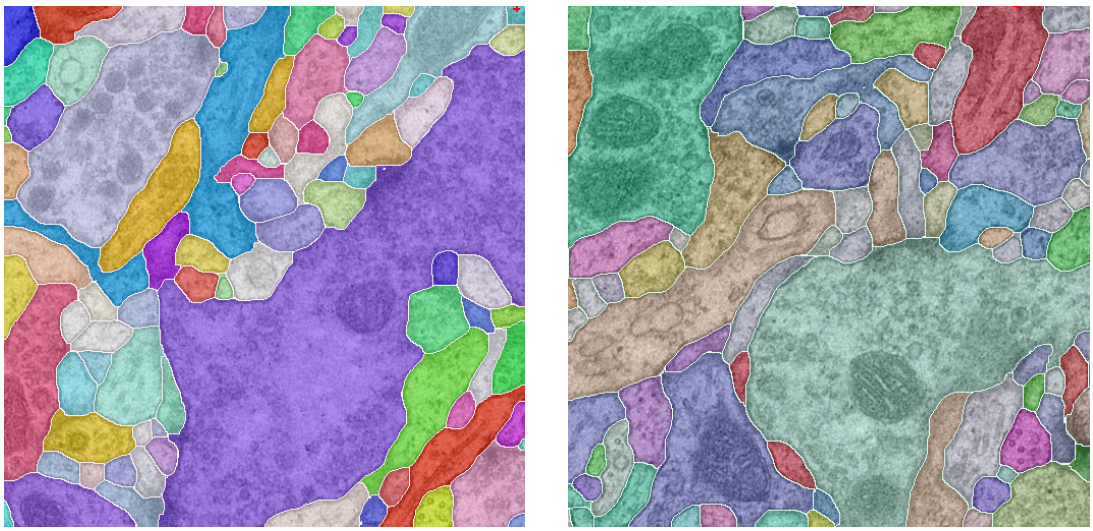


Figure 5.5: Mutex Watershed applied on the ISBI Challenge test data. For further images and a detailed comparison to the baseline segmentation methods see Section A.4.2

evaluated based on the Adapted Rand Score (Rand-Score) and the Variation of Information Score (VI-Score) [13], separately for each 2D slice.
Our method held the top entry in the challenge's leader board at the time of publication, see Table 5.1a. [1]. This is especially remarkable, because it is simpler than the other high-scoring methods. Similar to us, they rely on a CNN to predict boundary locations, but post-process its output with the complex pipeline described in [21], that involves a NP-hard partitioning step.
In addition, we compare to baseline post-processing methods starting from our network predictions: thresholding (THRESH), two watershed variants (WS, WSDT), and one Multicut variant (MC-LOCAL) only take into account short-range predictions. Lifed Multicut (LMC) and another Multicut variant (MC-FULL) also use long-range predictions. For these baseline methods we have only produced 2D segmentations for the individual slices, either because the 3D results were inferior (THRESH, WS, WSDT) or infeasible to obtain (MC, LMC). In contrast, the Mutex Watershed benefited from 3D segmentation. See Table 5.1b for the evaluation results and see Section A.4.2 for further details on the baseline methods and a qualitative comparison.
The three methods that use short- and long-range connectivity perform significantly better than the other methods. Somewhat surprisingly, MWS performs better than MC-FULL and LMC, which are based on a NP-hard partition problem. This might be explained by the lack of 3D

[1] Currently, the method proposed in [134], which also uses the Mutex Watershed, holds the top entry.

information in the two latter two approaches (solving the 3D model was infeasible).

| Method | Rand-Score | VI-Score |
|---|---|---|
| UNet + MWS | **0.98792** | **0.99183** |
| M2FCN + LMC [192] | 0.98788 | 0.99072 |
| SCN + LMC [226] | 0.98680 | 0.99144 |
| FusionNet + LMC [177] | 0.98365 | 0.99130 |
| ICv1 + LMC [21] | 0.98262 | 0.98945 |

(a) Top five entries at time of submission. Our Mutex Watershed (MWS) is state-of-the-art without relying on complex lifted Multicut post-processing used by all other top entries.

| Method | Rand-Score | VI-Score | Time [s] |
|---|---|---|---|
| MWS | **0.98792** | **0.99183** | 43.32 |
| MC-FULL | 0.98029 | 0.99044 | 9415.8 |
| LMC | 0.97990 | 0.99007 | 966.0 |
| THRESH | 0.91435 | 0.96961 | 0.2 |
| WSDT | 0.88336 | 0.96312 | 4.4 |
| MC-LOCAL | 0.70990 | 0.86874 | 1410.7 |
| WS | 0.63958 | 0.89237 | 4.9 |

(b) Comparison to other segmentation strategies, all of which are based on our CNN.

Table 5.1: Results on the ISBI 2012 EM Segmentation Challenge.

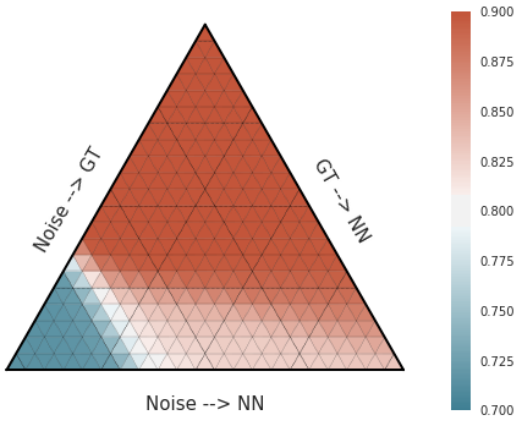
### 5.4.3 Study on natural image segmentation

We conducted exploratory experiments on the Berkeley segmentation data-set BSD500 [137] to study the Mutex Watersheds applicability to natural images. Training a state-of-the-art edge detection network on this small data-set requires a set of data-set specific optimization tricks such as training with external data, multi resolution architectures and auxiliary losses [108]. In this exploratory study we train a 2D version of the network used for the ISBI experiments to predict the 2D connectivity pattern depicted in Figure 5.4a. To alleviate the small size of the

training set, we present this network with predictions from [238] as additional input channel.
In order to isolate the influence of the quality of the underlying affinities, we run ablation experiments where we interpolate (via weighted average) between (a) affinities as predicted by our neural network, (b) those obtained from the ground-truth and (c) uniform noise. We obtain Mutex Watershed segmentations from the interpolated affinities for the BSD testset, size-filter them (as the only post-processing step) and evaluate with the Rand Index. The "phase transition diagram" resulting from these experiments is shown in Figure 5.6a; Figure 5.6b shows Rand Index and Variation of Information obtained for several points on this diagram.
Observe that the vertices corresponding to (a) and (c) can be interpreted as structured and unstructured noise on the ground-truth affinities (respectively). Hence, the results of our experiments show that the Mutex Watershed is fairly robust against both types of noise; when mixing the GT with noise, the quality of the segmentations is unaffected up to 60 % noise. When mixing GT with NN predictions, it is unaffected to an even higher degree.
In addition, we compare to the result of [102], who use an approach similar to ours and solve a Lifted Multicut based on long range potentials extracted from a pre-computed probability map. In Section A.4.3, we show the segmentations resulting at different stages of interpolation between GT, NN predictions and noise.

## 5.5 Conclusion

We have presented a fast algorithm for the clustering of graphs with both attractive and repulsive edges. The ability to consider both obviates the need for the kind of stopping criterion or even seeds that all popular algorithms, except for correlation clustering, need. The proposed method has low computational complexity in imitation of its close relative, Kruskal's algorithm.
At the time of publication of [233], the proposed algorithm, presented with informative edge weights from a neural network, outperformed all known methods on a competitive bio-image segmentation benchmark, including methods that operate on the very same network predictions. Since then, this result was improved upon by [134], who have also used the Mutex Watershed as part of their approach. In addition, the Mutex Watershed has also been included into another connectomics segmentation approach [119] based on affinities derived from learned pixel embeddings.
Furthermore, we have shown a close theoretical connection between the Mutex Watershed and Multicut in follow-up work [231] and generalized the algorithm to semantic instance segmentation, a common task in natural image segmentation challenges [44, 128, 151], in [232]. We have also used the Mutex Watershed for large scale microscopy segmentation problems, see Section 6.1 for details.




(a) BSD500 segmentation quality of MWS algorithm, given affinities from ground truth (top corner), from a neural network (right corner) or pure noise (left corner); plus hundreds of experiments on weighted combinations of the above. MWS segmentation quality (evaluated with Rand index) degrades only once a large amount of noise is added to the affinities.

| NN | GT | Noise | RI | VI |
|---|---|---|---|---|
| 100% | 0% | 0% | 0.826 | 1.722 |
| 0% | 100% | 0% | 0.901 | 0.927 |
| 0% | 38% | 62% | 0.897 | 0.976 |
| 0% | 33% | 66% | 0.820 | 1.912 |
| 80% | 20% | 0% | 0.878 | 1.247 |
| 43% | 0% | 57% | 0.813 | 2.127 |
| 43% | 14% | 43% | 0.838 | 1.636 |
| Keuper et al. [102] | | | 0.82 | 1.75 |

(b) BSD500 scores at various interpolations between the neural network predictions (NN), ground-truth (GT) and noise. See Section A.4.3 for example images of the interpolated affinities. We include [102] as a reference point, because they also use long range potentials in their segmentation method.

# 6 Applications of Large-scale Segmentation

We have applied the methods developed in this thesis to large-scale segmentation problems arising from biological research questions. Here, we highlight three applications that show the versatility of our approaches. For each application, we explain the biological relevance and show how our contribution enables further insight.
The first application (Section 6.1) is the segmentation of all cells, nuclei and selected organelles in a 6 day old larva of *Platynereis dumerilii* imaged in EM. This work has been foundational to the collaborative effort of building a detailed cellular atlas for this animal [218]. Section 6.1 is based on the publication [218], where several authors have equally contributed. Our main contribution to this work are the different segmentations from EM, developing tools for data exploration and sharing as well as integrating the segmentations with down-stream analysis. The most relevant contributions from other authors for the results described here come from Christian Tischer, who spearheaded the development of the viewer, Kimberly Meechan, who performed the analysis of the cell morphology, gene expression and their correlation and Hernando Vergara, who imaged the in-situ gene expression profiles and contributed to their registration.
In Section 6.2, we describe a contribution to the publication [153] of Musser et al. Here, we have contributed the automated segmentation of cells, microvilli and flagella in a sponge choanocyte chamber and, together with Jakob Musser and Giulia Mizzon, contributed to the subsequent analysis and visualization of this resource.
Finally, in Section 6.3 we describe an immunofluorescence microscopy assay for the detection of SARS-CoV-2 antibodies, where we have contributed to the image analysis pipeline developed to automatically score the antibody responses. This work is based on the publication [168], where several authors have contributed; especially Roman Remme, who has set up the infected cell classification and Vibor Laketa, who has designed the assay and performed all microscopy imaging.
Besides these three publications, we have contributed to several other studies: the study [236] investigates plant development using light-sheet microscopy and makes use of several of the segmentation methods developed in this thesis. In [50] we have helped to set up an automated segmentation approach used to investigate the development of the nervous system in mice. The

data exploration tools developed for [218] have been adapted for EM volumes of SARS-CoV-2 infected cells in [47]. Finally, we have applied the assay developed in [168] for a population level serological study in [208] and adapted the assay for a drug reporter system in [165].

## 6.1 A cellular atlas for *Platynereis dumerilii*

Cells are the basic units of life. In multicellular organisms, distinct sets of genes are expressed in different cells, producing the individual cellular traits that we call cell types [11]. Deciphering how the genotype is decoded into a multicellular phenotype is therefore critical to understand the development, structure and functioning of an entire body. Hence, we need to establish the link between gene expression profiles and cellular morphologies. To achieve this integration, data from different modalities has to be brought together: detailed cellular morphologies can be reconstructed from high-resolution EM image volumes that stain the membranes of cells and innner-cellular structures; gene expression profiles can be obtained from markers imaged with light microscopy at lower resolution.
Here, for the first time, we combine cellular morphology and gene expression at sub-cellular resolution for a full organism, a six day old larva of *Platynereis dumerilii*, a marine worm. To this end, we segment all cells and nuclei in an EM volume of a specimen and register a gene expression atlas [219] to this volume. Thus, we can assign gene expression information to cells for the entire animal. We find that an unbiased clustering based on the gene expression defines groups of cells that correlate well with anatomical units, in particular defining groups of neurons that form coherent tissues separated by morphological boundaries. In order to integrate, analyze and share the data derived from multiple modalities, we develop an open-source viewer that we call MoBIE [1] and make all data available through it. We expect that the methods developed here can be used to build similar cellular atlases for animals of comparable size.
The results described here have been achieved in a collaborative effort. Our main contribution is the extension and application of the segmentation methods and integrating the segmentation results with further analysis steps as well as contributing to the software for exploring and sharing the data. The next section summarizes the segmentation contributions, followed by an overview of further analysis results that highlight the methodological advances enabled by large scale segmentation. While we have contributed to these results, these efforts have been led by other partners in the collaboration: Hernando Martinez Vergara has composed the gene expression atlas, Christian Tischer has led the efforts for registration and viewer software development and Kimberly Meechan has done the main work behind the morphology analysis and clustering.

[1] https://github.com/mobie/mobie

### 6.1.1 Segmentation

The EM image volume at hand captures a 6 day old larva of *Platynereis dumerilii*. It was imaged at a pixel size of 10×10×25 nanometer with serial block face microscopy, resulting in 11,416 images. These images were aligned using SBEMImage [206] to obtain an image volume of 275×260× 285 micrometer in physical dimensions and a size of 2.5 TB (compressed).
Given the complete specimen at ultra-structural resolution, we provide the segmentations of all cells, all nuclei, the cuticle (epidermal skin layer), selected tissues and body parts as well as nuclear chromatin and cilia for selected organs. In particular the segmentation of cells in this data-set is challenging: the appearance of cells and their membrane is diverse throughout the animals' body. Membranes of sub-cellular structures often resemble cellular membranes and some membranes are missing due to staining artifacts. To alleviate these issues, we make use of the segmentations of structures discernible at coarser resolution, nuclei, tissues and organs, and use them to inform the cellular segmentation.
We provide segmentations for the following tissues and organs of the animal: coelomic cavity, glands, gut, secretory cells and yolk. These are large structures with a pronounced region boundary. To segment them, we have used the ilastik carving workflow [22], which is based on watersheds from seeds provided by the user, on downsampled data (80×80 ×100 nanometer).
To segment the nuclei, we have used the Mutex Watershed algorithm [233] (Chapter 5). First, a 3D U-net [35] predicts short- and long-range pixel affinities and, for each pixel, predicts whether it belongs to a nucleus or not. Based on these predictions, we obtain a nucleus instance segmentation with the Mutex Watershed. The computations for the nucleus segmentation are performed on data downscaled to a pixel size of 80×80 ×100 nanometer. Still, the whole volume could not be processed in one piece, so we have applied the algorithm to blocks of size 512×512×64 pixels and then combined the results into a full segmentation by solving a Multicut problem following [166] (Chapter 3). The network was trained with 12 blocks of manually annotated training data (each of size 400×400×120 pixels). We initially tried to segment the nuclei with a method based on watershed over-segmentation, but observed frequent merges of nuclei that touch across a small portion of their boundary, an issue observed only very rarely when using the Mutex Watershed approach.
Cilia and cuticle are segmented with the same method as the nuclei, using the full resolution (10×10×25 nanometer pixel size) data for the cilia segmentation, but only for the region of segmented nephridia cells (see below). The cuticle segmentation is based on data downscaled to 40×40×50 nanometer. For both tasks, separate 3D U-nets were trained on manual segmentations of the corresponding structures.
For the cell segmentation we rely on the Lifted Multicut segmentation approach of [167], (Chapter 4). We obtain cell membrane predictions from a 3D U-net trained to predict short- and long-range affinities. Here, we additionally insert the edges of tissue and organ segmentations

(see above), in order to avoid missing boundary signal due to the very different appearance of some organ and tissue boundaries. The groundtruth annotations for this network consist of 8 blocks of 628×628×130 pixels and a larger block of size 1280×1280×120 pixels. Based on the network predictions, we compute superpixels, region adjacency graph and edge costs. We additionally exploit the nucleus segmentation to enforce separation of cells containing different nuclei. To this end, we introduce lifted edges between nodes whose superpixels map to the segmented nuclei, attractive for the nodes mapped to the same nucleus and repulsive for the nodes mapped to different nuclei. Lifted edges are introduced up to a graph distance of four and the attractive / repulsive edge cost is set to the maximum / minimum of the local edge costs. The Lifted Multicut problem is solved by the hierarchical solver introduced in [167] (Section 3.3.2). Since the repulsive lifted edges are only included up to a certain graph distance, there are still objects in the cell segmentation that contain more than one segmented nucleus. We find these in post-processing and separate them individually by running a graph watershed seeded from the nodes mapped to the nuclei. The cell segmentation was performed on data downscaled to a pixel size of 20×20×25 nanometer. Processing the whole volume took a total of 30 hours. The first 10 hours were spent on neural network affinity prediction, using 6 GPUs for parallelization. The remaining 20 hours were spent on the problem set-up and solving the Lifted Multicut, using up to 600 CPU cores for parallelization. See Section 3.4.3 and Section A.1.2 for implementation details.

We leverage prior information from nuclei and tissue segmentation to mitigate segmentation errors arising from ruptured cellular membranes and diverse appearance of cell membranes. While these issues could in theory be mitigated by providing additional training data, this process is very laborious for 3D segmentation. Instead, we rely on the nucleus segmentation - a simpler problem which our algorithm solves to 99.0% accuracy (see below) - and tissue segmentation, which can be achieved at lower resolution. Our use of nuclei as prior knowledge assumes that every cell should contain only one nucleus, a constraint that is expected to be true for almost all of the cells at this developmental stage. Furthermore, we proofread some cells that are likely to contain a false merge, measured by a morphology based score. We iterate through the top 1000 cells ranked by this score and correct the cells that contain a false merge via graph watershed from user-generated seeds. This correction procedure was applied to 154 falsely merged cells. In addition, we use Paintera [72] to perform more fine grained proof-reading for some selected cells.

The cellular segmentation contains 11,402 cells with nuclei, see Figure 6.1(A) and achieves accurate segmentations for different kinds of cells such as epidermal cells (B), muscles (C) and nephridia (D). We measured nucleus sizes in the range from 33.6 to 147.5 $\mu m^3$, and cell sizes in the range from 59.8 to 1224.6 $\mu m^3$. Note that neurites in the neuropil have not been segmented, as they are not sufficiently preserved in the EM volume for automated segmentation. The detail in the 3D ultrastructure volume and the cell segmentation provide a framework

for anatomical classification in a complete and unbiased manner. Here, we focused on the larval nephridia (Figure 6.1 (D)). Previous transmission EM analysis had shown that the larval nephridia are embedded between the body surface muscles and oblique muscles, and that the tubule wall is formed from single cells wrapping around a tight lumen with six cilia constantly present [74]. However, transmission EM on single sections could not resolve how many cells contribute to this structure and how many distinct cilia they protrude into the lumen. This task is now possible with our resource: we segmented all nephridial cilia as proof of principle that comprehensive ultrastructure segmentation is possible with our resource and methods. The two nephridia stereotypically comprise 7 cells per side, and each cell contributes several cilia to the continuous central bundle. The bundles are made up of 85 and 78 cilia on the left and right sides of the body, respectively. Furthermore, we observed a similar distribution of cilia per cell for both sides and found that for a given cross-section of the lumen cilia belong almost exclusively to one cell, see Figure 6.2 (B).
To quantify the segmentation accuracy, we validate the cell and nucleus segmentations against 8 slices (4 transversal, 4 horizontal) manually annotated by domain experts. The annotators have marked nuclei and cell centers with circular annotations; each slice is annotated twice and then the consensus annotation is used. For the cell segmentation, we distinguish two types of errors: false merges, which are given by a segmented cell that is matched to more than one annotation, and false splits, which are given by a cell without matching annotation or an annotation matched to more than a single cell. Based on 406 annotations, we find 6.30 % false splits and 3.23 % false merges. For the nucleus segmentation, we find false positive errors, corresponding to a segmented nucleus without matching annotation, and false negative errors, corresponding to an annotation without matching nucleus. We observe a false positive rate of 0.49 % and a false negative rate 0.55 % based on 2888 annotations. See Figure 6.2 (A) for examples of annotations and the types of errors.

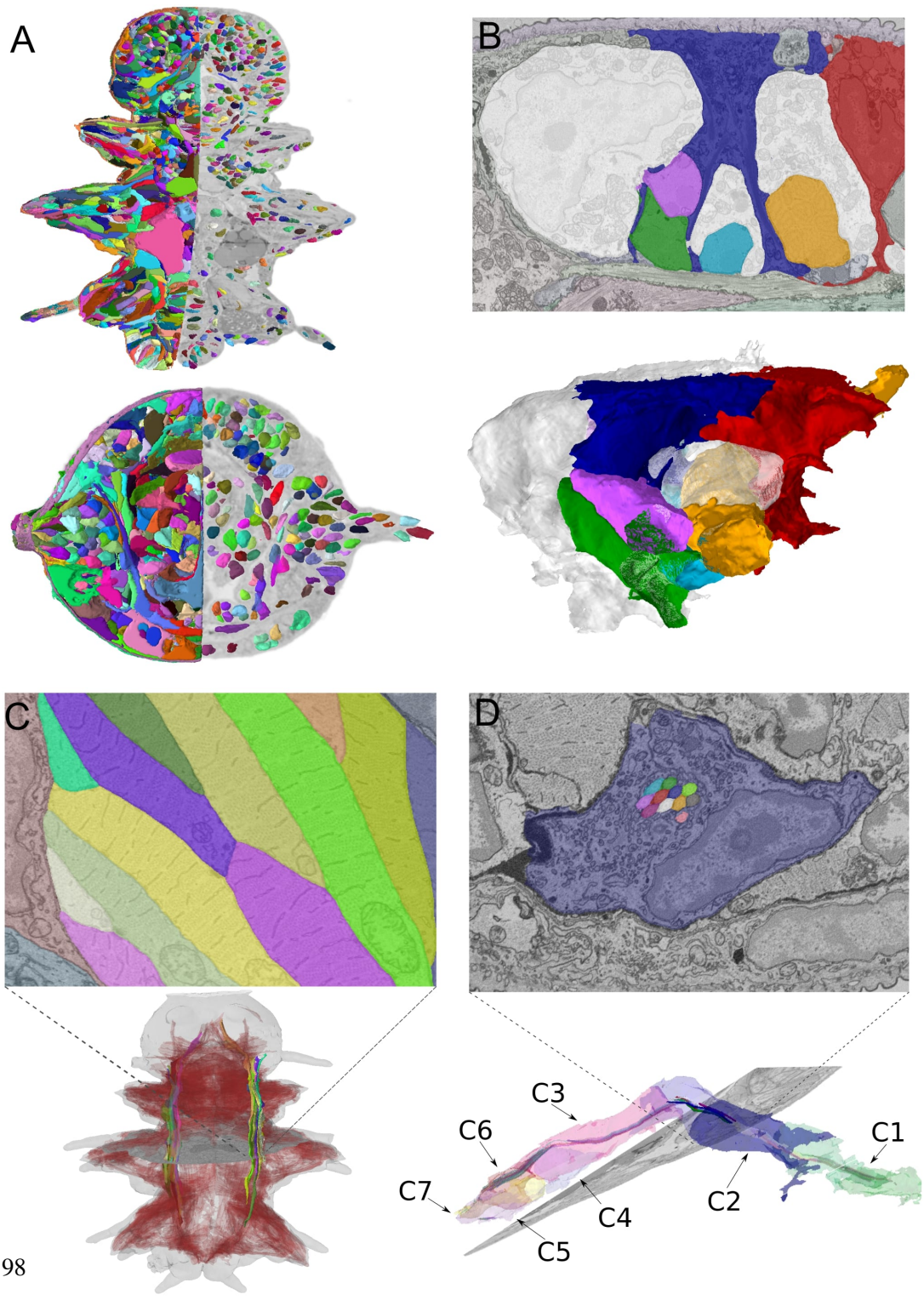

A
B
C
D
C3
C6
C1
C7
C4
C2
C5

Figure 6.1: (A) Cells and nuclei are segmented in 3D in the whole EM volume. Horizontal section (top) and transversal section (bottom) with 3D renderings of all cells (left half) and nuclei (right half). The cellular segmentation yields accurate 3D reconstructions for the different cell types. (B) Intertwining epithelial cells are reconstructed, see colored segments overlayed with EM (top) and 3D rendering (bottom). (C) Long stretching muscles are segmented correctly, see overlay with EM in the top image. This bundle of muscles is highlighted in the bottom rendering, with the corresponding bundle on the other side colored less brightly and all other reconstructed muscles rendered in brown. (D) We studied the nephridia in more detail and also reconstructed their cilia bundle. Seven cells contribute to this bundle, see the top image for an intersection of one of these cells and the bundle and the bottom image for a 3D rendering of the seven cells and cilia where each cilium is colored the same as the cell it is attached to.

### 6.1.2 Further Analysis

Based on the complete cellular segmentation, we can compute detailed morphological descriptions and use the segmented cells to integrate data from different modalities. Here, we first map gene expression data to the EM data-set, then compute morphological descriptors of the cells and correlate gene expression and morphology.

**Gene Expression Atlas and Registration** The number and position of cells at a given developmental phase is stereotypical in *Platens* [222], which allows for the generation of gene expression atlases for the whole body with cellular resolution. Here, we build on the gene expression atlas from [219]. It was obtained by imaging genetic markers for 153 specimens and registering the individual specimen to a common reference coordinate system via nucleus positions. The marker images for individual specimen were then averaged to obtain a single volume per marker. The gene expression atlas contains 205 image volumes for the different markers at a pixel size of $0.55\times0.55\times0.55$ micrometer, so at a significantly coarser resolution than the EM data-set ($10\times10\times25$ nanometer).
We register the gene expression atlas to the EM coordinate system based on the nucleus signal, which is available via DAPI[2] in the former and via the segmented nuclei in the latter. The registration is performed by first applying a similarity transformation followed by a sequence of B-Spline transformations, using the software package elastix [106]. Based on 43 landmarks that were manually identified in both data-sets, we find a median landmark discrepancy of 2.99 micrometer, which is less than one cell diameter. Figure A.9 illustrates the registration process and shows examples for several gene expression maps overlaid with the EM after registration.

[2]DAPI is a fluorescent marker that binds to parts of the DNA and is used for fluorescent imaging of nuclei.

**Cell Morphology and Clustering** The combined resource of EM image volume, segmented tissues, cells, nuclei and ultra-structure as well as gene expression data provides a rich resource, which enables to investigate the link between gene expression and cellular morphology. As a first step, we investigate how the segmented cells can be characterized based on their morphology only and compute a set of 140 morphological descriptors per cell, derived from cell and nucleus shapes, intensities and texture. Some of these descriptors are also derived from a further segmentation of the nuclei into heterochromatin (electron dense part) and euchromatin (less electron dense), using ilastik pixel classification [22]. The chromatin patterns are a proxy for the DNA configuration inside the nucleus.
We visualize the high-dimensional space defined by the morphological descriptors using UMAP [139] and further cluster the cells with graph-based community detection [26], resulting in 11 distinct clusters. Figure 6.2 (D) shows the clusters mapped onto the UMAP and (E) shows them mapped to a cross-section of the EM: individual clusters cover anatomically distinct parts of the animal. To further quantify the usefulness of our morphological descriptors, we use them to recover pairs of cells that are bilateral partners, i.e. two corresponding cells in the left and right part of the body[3]. For each segmented cell, we rank all other cells by their euclidean distance in the descriptor space and find the first neighbor that is a potential bilateral partner. The criterion for a potential partner is determined by the spatial distance of positions mirrored at the symmetry plane of the animal, using the mean distance of 202 manually identified bilateral partners as matching criterion. Figure 6.2 (C) shows the fraction of cells that have a bilateral partner as K-th neighbor for different descriptor sets: all descriptors, restricted to the descriptors derived from cell, nucleus or chromatin segmentation. These four sets of descriptors perform significantly better than random assignment, demonstrating that they characterize the cell morphology well.

[3] *Platynereis* has a bilateral symmetry, thus many, but not all, cells have a unique partner cell [219]

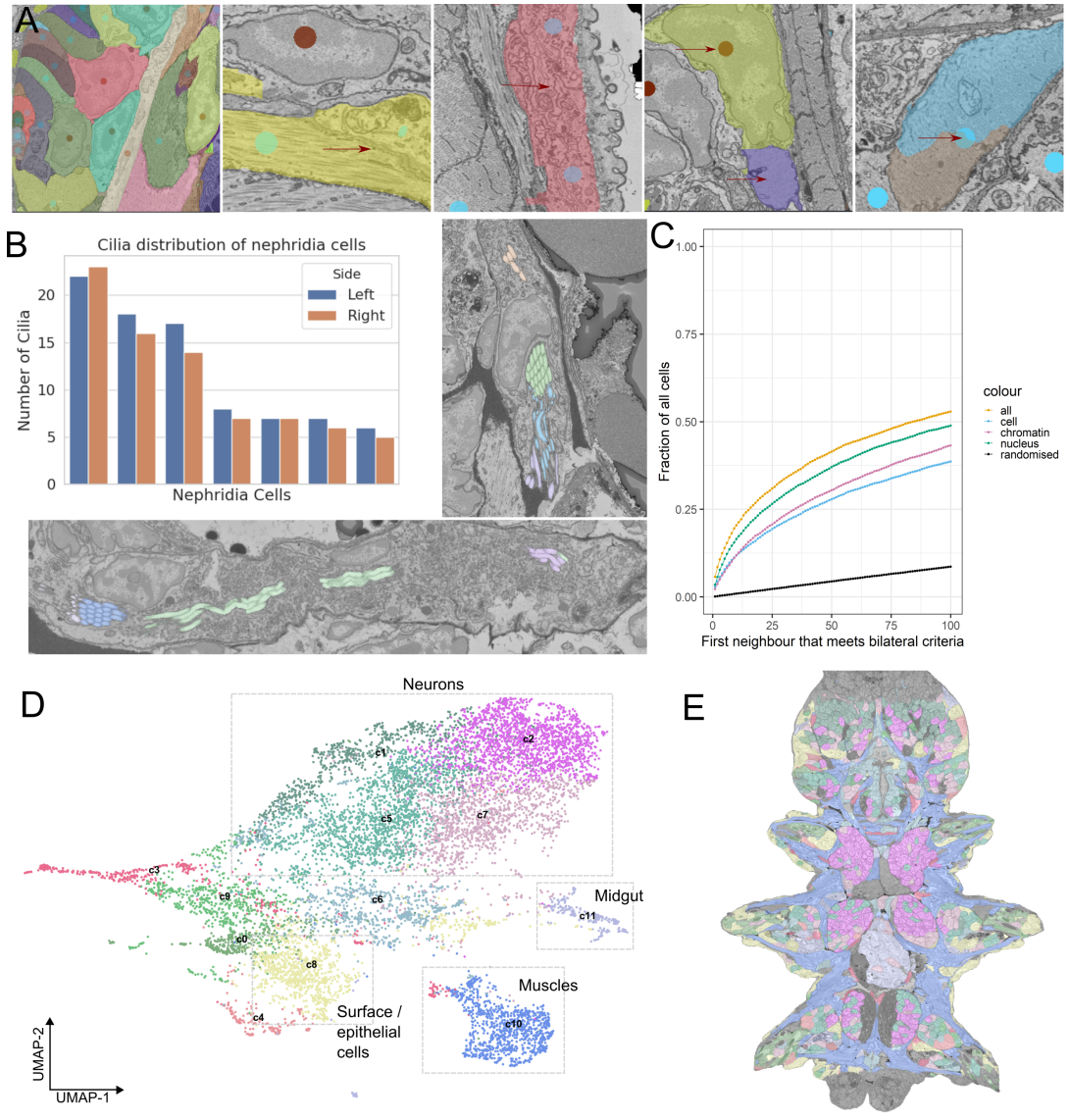

A
B
Cilia distribution of nephridia cells
Side
Left
Right
Number of Cilia
Nephridia Cells
C
Fraction of all cells
First neighbour that meets bilateral criteria
colour
all
cell
chromatin
nucleus
randomised
D
Neurons
Midgut
Muscles
Surface / epithelial cells
UMAP-2
UMAP-1
E

Figure 6.2: (A) Nuclei and cells were annotated by domain experts for 8 slices (4 transversal, 4 horizontal), see the leftmost image for example annotations. We use these annotations to find false merge errors, see the two middle images with arrows highlighting the cell membrane not picked up, and false split errors, see the two rightmost images with arrows highlighting parts of the cell that are split off in the segmentation. (B) The distribution of cilia per cell is stereotypical for the nephridia on both sides of the animal. Cilia in a given cross-section of the cilia bundle start off from the same cell, see segmented cilia colored by their origin cell overlayed with the EM. Upper image shows a cross-section of the right nephridium, lower image of the left one. (C) Bilateral pair analysis - the graph shows the fraction of cells that find a potential bilateral partner within a certain number of neighbors when ranking all other cells by the euclidean distance of morphological descriptors. The set "all" consists of all descriptors, "cell" only consists of descriptors derived from the cell shape, texture and intensity, same for "nucleus" and "chromatin", but for the corresponding segmentation. The baseline "randomized" is computed by taking the mean of 100 random rankings per cell. (D) UMAP of the space defined by all morphological descriptors, colored by the 11 clusters obtained from community based clustering. The gray boxes mark some distinct categories of cells that can be identified based on the morphological descriptors. (E) Clusters from (D) overlaid on a cross-section of the EM.

To further investigate the gene expression, we compute an expression value per gene for each cell. Here, we use the fraction of the spatial overlap of the cell mask and gene expression; resulting in a value in range 0 to 1. We visualize the space defined by the expression values using UMAP and cluster the gene expression space, again using the community based approach of [26]. This results in 15 distinct clusters, see Figure 6.3 (A). In (B) the same UMAP is displayed with an overlay of the the segmented organs and tissues, showing that these have distinct profiles in the gene expression space.

For a more detailed analysis of how gene expression relates to tissue boundaries, we focus on the head, the region with the highest gene expression density. It is already highly subdivided at six days [37]. We subdivide the head into nine anatomically distinct parts, referred to as ganglia in the following, by manually assigning segmented cells, see Figure 6.3 (C). We notice that six of the genetic clusters in the head show a clear spatial correlation to the anatomically defined ganglia. To quantify this relationship, we calculate a specificity score for each genetic cluster and the individual genes with the ganglia. We use the F1-score based on the fraction of cluster cells (cells expressing the gene) belonging to the ganglion and the fraction of ganglia cells that belong to the cluster (express the gene). Remarkably, for almost all ganglia the specificity values for the individual genes were considerably lower than the specificity values for the gene clusters (exceptions were ganglia 3 and 6 that are very small), see for example (D). Genes with

relatively high specificity values often encode known transcription factors[4]. Characterizing these factors further we found that expression occurs in coherent and overlapping domains, but covers several of the cluster-defined territories. These findings indicate that the *Platynereis* head is subdivided into domains that are combined by expression of several transcription factors, rather than a single factor, which largely correspond to morphologically distinct brain ganglia.

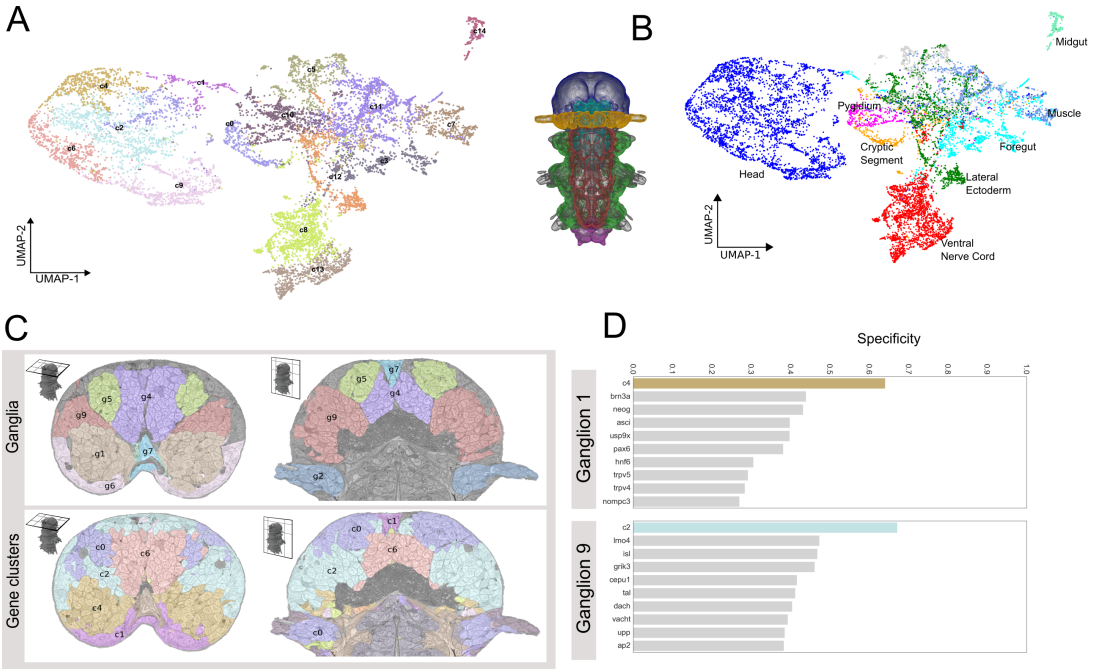



Figure 6.3: (A) UMAP of all cells based on the expression data of all 201 genes in the atlas. The points are colored by their membership to the different gene expression clusters c0-c14. (B) Segmented tissue and regions mapped onto the gene expression UMAP. (C) Comparison of anatomically defined ganglia (top row) with genetically defined clusters (bottom row) in the animal head. (D) Comparison of specificity of gene clusters and individual genes for the two ganglia g1 and g9, showing the top 10 scoring genes / clusters.

**Data Exploration** The EM image volume, the segmentations and the registered gene expression maps form a valuable resource, that we hope will enable further insights into the relation of genotype and phenotype in *Platynereis* and beyond. However, the number and size

[4]Transcription factors are proteins that control the rate of DNA transcription and are often differentially expressed across different tissues.

of image source, currently 231 sources adding up to over 2TB of lossless compressed data, make it difficult to explore and interrogate the data for scientific discovery. We have therefore developed a platform for multi-modal big image data exploration and sharing, short MoBIE, to explore and share the data-set at hand and simplify the interaction with large multi-modal image data in general. At its core, MoBIE is a viewer for large 3D data based on BigDataViewer [174]. It is available as a Fiji plugin [188]. It makes use of the main features of BigDataViewer: browsing large image data stored in a chunked pyramidal data format, arbitrary plane slicing of 3D data and on-the-fly affine transformations that allow to simultaneously display data of different resolutions, such as EM and genes in our case.
It provides additional features that are essential to efficiently interact with multi-modal data-sets and segmentations: a user interface to select an arbitrary number of image sources, look-up tables for displaying segmentations, tables linked with the segmented objects that can, for example, be used to look up and visualize the cell morphology descriptors or their gene expression. We also integrate 3D rendering of segmented objects via the Fiji 3D viewer [190] and bookmarks to save and navigate to points of interests. For data access without the need for downloading complete data-sets, the MoBIE viewer supports streaming data from any S3 compatible web object store, see Section A.1.3 for details. For an overview of the main elements of the MoBIE viewer, see Figure A.2. MoBIE is available as free open source software at `https://github.com/mobie/mobie-viewer-fiji`; all the data generated for the *Platynereis* data-set is available at `https://github.com/mobie/platybrowser-datasets`.

## 6.2 Segmentation of a Sponge Choanocyte Chamber

Sponges are sister to nearly all other animals [227, 193]. Unlike most animals, they lack bona fide neurons, muscles, and a gut. Rather, their body plan is composed of a few basic cell types that act to create an efficient water pump for filter-feeding and waste removal. Among these, the choanocytes form spherical chambers. They exhibit microvilli and a flagellum that beats to drive water through the sponges pore system. Despite their simple organization, sponges possess genes encoding conserved molecular machinery found in neurons and muscles of bilaterian animals. This includes components of the bilaterian neuronal pre- and postsynapse [126].
However, prior to the work of Musser et al. [153], which we have contributed to, cells with integrative signaling functions were unknown. There, single cell RNA sequencing revealed so-called *neuroid cells*, which express a genetic profile similar to neurons in biletarians, among the identifiable cell types. These cells potentially form an early stage in the evolution of neurons, presenting an important link in animal cell type evolution and the origins of the

nervous system. To determine their function, a 30 micrometer cubed FIBSEM volume was imaged and correlative light microscopy was used to find back neuroid cells, see Figure 6.4. The neuroid cells are located in the center of choanocyte chambers and interact spatially with the chaonocytes, suggesting that they coordinate the choancytes flagella movements. In order to analyze these interactions in more detail (Section 6.2.2), we have produced an instance segmentations of all cell bodies, flagella and microvilli in the choanocye chamber (Section 6.2.1).

### 6.2.1 Segmentation

To study the interaction of choancytes and neuroid cells in EM, we segment the relevant structures, namely cell bodies of choanocytes, apopylar and neuroid cells as well as microvilli and flagella. The size, shape and frequency of instances of these three classes differs greatly. In addition the intensity histogram shifts between parts of the data-set that were imaged in different sessions. These facts make the task at hand too challenging to be solved by a simple thresholding based approach. Thus, we employ the Lifted Multicut based segmentation workflow of [167], see also Section 4.3.
We use the autocontext workflow of ilastik [22] to generate boundary and semantic probabilities. This algorithm consists of multiple stages of pixel classification. In an individual stage a random forest classifier predicts semantic class probabilities for each pixel. The classifier is trained from sparse labels and uses the responses of convolutional filter banks applied to the input data as features. Each stage is presented with the raw data and the predictions of the previous stage, thus refining the predictions from the previous stage. Here, we perform three stages of autocontext: in the first and second stage, we predict six different classes: background, object boundary (corresponding to membranes of cells, flagella and microvilli), cytoplasm, nucleus, flagellum and microvillus. In the third stage, we only predict object boundary versus background. See Figure 6.5 (b) for exemplary predictions from the second stage.
We use the boundary predictions from the third stage and the class predictions from the second stage to perform Lifted Multicut based segmentation, see also Figure 6.5: first, distance transform watershed superpixels are generated from the boundary predictions, see panel (c). The region adjacency graph is built from the superpixels and edge costs are derived from the mean accumulated boundary probabilities. The graph is then augmented by sparse lifted edges: we first compute the connected component of the thresholded microvilli and flagella predictions, see panel (e). Then, we map the resulting segments to graph nodes via overlap with the corresponding superpixels. Attractive lifted edges are then introduced between nodes mapped to the same objects, see panel (f). The resulting Lifted Multicut problem is solved with the algorithm introduced in [167] (Section 3.3.2).
Note that segmentation via the Multicut problem based on the graph without lifted edges is also

possible. However, we observe that this segmentation heavily over-segments microvilli and flagella. Both structures have an elongated shape with small diameter, which results in small superpixels and non-robust estimates for the edge costs based only on boundary predictions. Hence, we add the attractive lifted edges derived from more robust semantic predictions to alleviate the degree of over-segmentation. The segmentation obtained via connected components on the thresholded predictions on its own is of inferior quality; it falsely merges almost all cell bodies and systematically lacks segments for microvilli. For a quantitative comparison between Multicut and Lifted Multicut segmentation, please refer to Section 4.4.3. Note that the problem set-up there also includes repulsive lifted edges. This is omitted here, because under-segmentation of microvilli was not a concern; see also Section 6.2.2.

### 6.2.2 Analysis

For further analysis, the cell bodies from the automated segmentation were proofread and microvilli forming the collar around the flagella were merged into a single object. This was achieved manually using the proof-reading tool paintera [72].
The volume contains two neuroid cells situated inside the chamber, see Figure 6.4 panels (a) to (c). The first neuroid cell is positioned in the center of the chamber (violet cell in (a)), with the second cell residing near the apopylar pore (red cell in (b)).
Our reconstruction revealed that both neuroid cells form multiple protrusions, each directed towards the collar of individual choanocytes, see panels (b) to (f). Strikingly, nearly all extensions from the neuroid cells contact and enwrap one or more microvilli (panels (d) to (e)), with each extension reaching to a different choanocyte collar. In several cases, we also observed neuroid cell extensions in close proximity to flagella, and even orienting themselves along the main flagella axis. Flagella in close contact with the neuroid protrusions emerged straight, and then bent, contrasting with the undulatory, corkscrew-like appearance of normal motile cilia, suggesting they may not be actively beating.
Overall these spatial interactions confirm that communication between neuroid cells and choanocytes through neurotransmitters is possible. Hence, neuroid cells are likely to play a key role in the coordination of choanocyte movements and are thus likely to represent an important stage in the evolution of neural cell types.

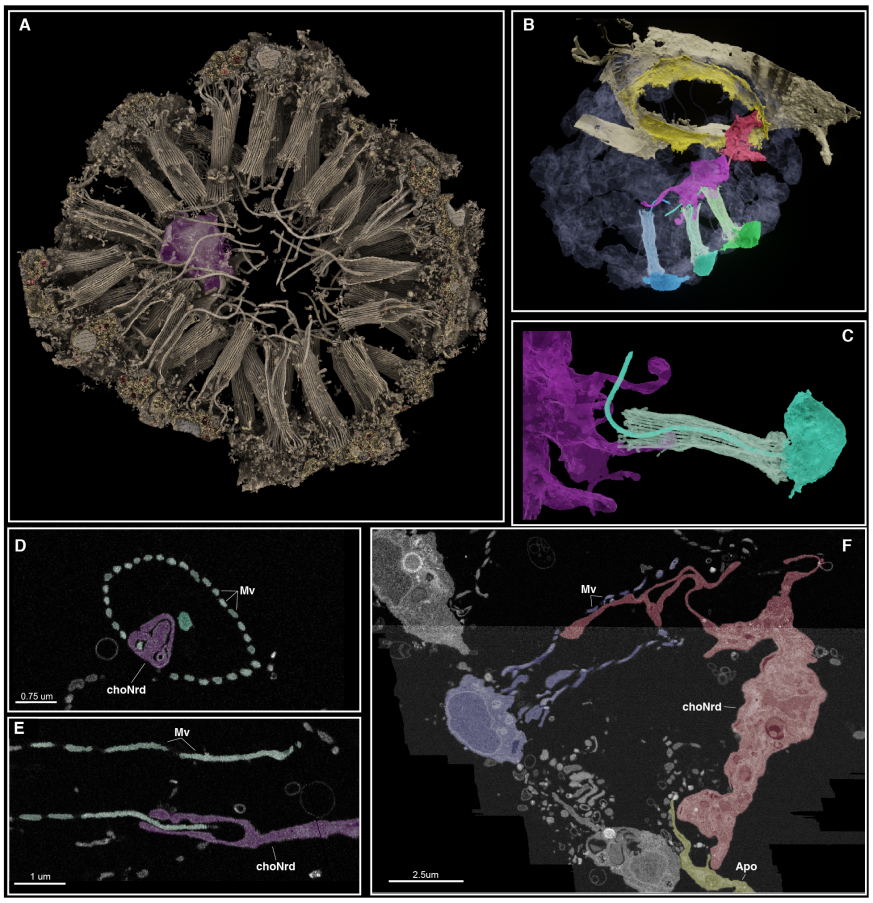



Figure 6.4: FIBSEM of neuroid-choanocyte interaction. (a) Rendered 3D volume of choanocyte chamber with neuroid cell (violet). (b) Segmented volume showing two neuroid cells (violet and red) contacting flagella and microvillar collars of three choanocytes (blue, turquoise, and green) and apopylar cells (yellow). (c) Segmented neuroid cell (violet) with filopodia extending into the microvillar collar (turquoise). (d-f) 2D images of neuroid cells (choNrd; purple and red) filopodia extending into, and enwrapping, choanocyte microvilli (Mv; turquoise and dark blue), and contacting apopylar cell (Apo; yellow).

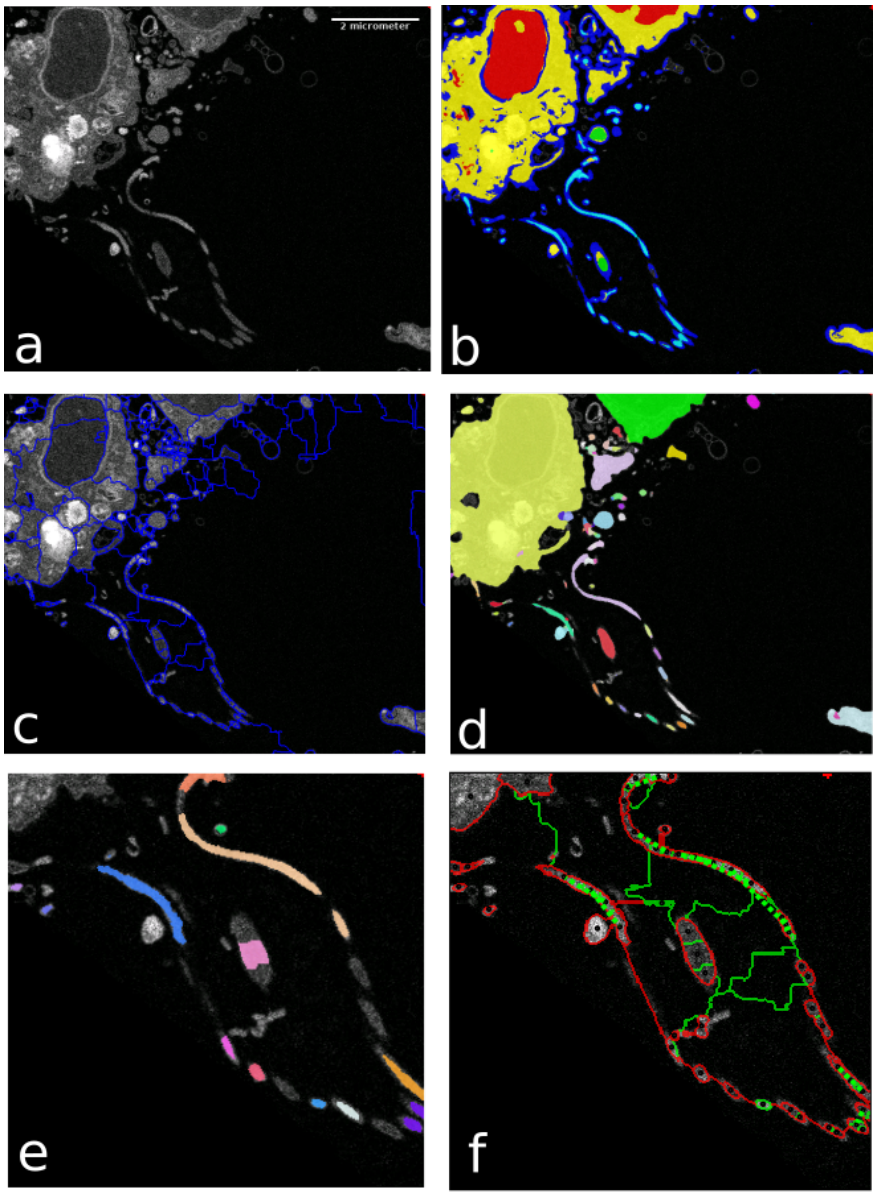



Figure 6.5: We derive semantic and boundary predictions (b) from the raw data (a) using the ilastik autocontext workflow. We compute a watershed over-segmentation (c) and build a graph with edge costs derived from boundary evidence; indicated by green (attractive) and red (repulsive) lines in (f). In addition, we compute an instance segmentation based on thresholding flagella and microvilli predictions from (b) and map these to watershed segments / graph nodes (e). We introduce attractive lifted edges (green dashed lines in (f)) between nodes that were mapped to the same segment. We solve the lifted Multicut problem to obtain the instance segmentation (d).

## 6.3 Immunofluorescence based Testing for SARS-CoV-2 Antibodies

The recent emergence of the novel coronavirus SARS-CoV-2 [237, 89] and the rapid pandemic spread of the virus has dramatic consequences in all affected countries. Testing for SARS-CoV-2 infection and tracking of transmission and outbreak events are of paramount importance to control the pandemic. The sequence of the viral genome became available only weeks after the initial reports on COVID-19 and allowed rapid development of reliable and standardized quantitative PCR based tests for direct virus detection [45]. While these tests are the key to identify acutely infected individuals, monitoring and tracking of SARS-CoV-2 specific antibody levels becomes highly important to understand the immune response against SARS-CoV-2 and monitor infection levels in the general population. Therefore, specific, sensitive and reliable methods for the quantitative detection of virus specific anti-bodies are urgently needed from the beginning of an emerging pandemic.

Compared to approaches for direct virus diagnostics by PCR, development of test systems for detection of SARS-CoV-2 specific antibodies proved to be more challenging. While ELISA [58] based antibody test kits for SARS-CoV-2 serological diagnostics are commercially available, the initially marketed test kits underwent a rapid development and approval process; thus only small sample sizes were used for validation. Consequently, sensitivity and specificity of the test systems often failed to meet the practical requirements [229]. Thus, complementary strategies to test for antiviral antibodies that can be rapidly deployed in situations where commercially available kits are either not yet developed or not available are an important addition to the diagnostic toolkit.

Immunofluorescence (IF) using virus infected cells as a specimen is a classical serological approach in virus diagnostics and has been applied to coronavirus infections [36]. The advantages of IF are that it does not depend on specific diagnostic reagent kits or instruments and that the specimen contains all viral antigens expressed in the cellular context, thus providing high information content. The main disadvantage of the IF approach, as it is typically used in serological testing, is its limited throughput capacity due to the involvement of manual microscopy handling steps and sample evaluation based on visual inspection. Furthermore, visual classification is subjective and thus not well standardized and yields only binary results. Here, we address those limitations, making use of high-throughput microscopy and image analysis methods developed for basic research. We establish and validate a semi-quantitative, semi-automated workflow for SARS-CoV-2 specific antibody detection, combining the advantages of IF with a reliable and objective readout and high throughput compatibility. The protocol described here was developed in response to the emergence of SARS-CoV-2, but it represents a general approach that can be adapted for the study of other viral infections and is

suitable for rapid deployment to support diagnostics of emerging viral infections in the future. It has already been applied to validate other serological tests in a large population study [208] and has been integrated into a new reporter system for drug discovery [165].

### 6.3.1 Immunofluorescence Assay and Image Acquisition

We set up the IF analysis using cells infected with SARS-CoV-2 as samples. African green monkey kidney epithelial cells [162] (Vero E6 cell line) are used for infection with SARS-CoV-2, virus production and IF. Our strategy involves a direct comparison of the IF signal when mixing the examined serum with infected and non-infected cells in the same sample. This enables the clear identification of positive reactivity in spite of a variable and sometimes high nonspecific background from human sera. Preferential antibody binding to infected compared to non-infected cells indicates the presence of specific SARS-CoV-2 antibodies in the examined serum. Under our conditions, infection rates of 40% to 80% of the cell population were achieved, allowing for a comparison of infected and non-infected cells in the same well of the test plate. An antibody that detects double-stranded RNA (dsRNA) produced during viral replication was used to distinguish infected from non-infected cells within the same field of view. The bound SARS-CoV-2 antibodies are detected using fluorophore-coupled secondary antibodies against human IgG, IgA or IgM.
In summary, the assay and image acquisition proceeds as follows: Vero E6 cells are seeded into 96-well plates infected and immunostained using anti-dsRNA antibody and patient serum, followed by indirect detection using a mixture of anti-IgG and anti-IgA/IgM secondary antibodies. Images are acquired using an automated widefield microscope. Figure 6.6 panel (A) provides an overview of these steps, panel (B) shows example IF images for a COVID-19 patient and a healthy donor.

### 6.3.2 Quantitative Analysis

To obtain a measure for specific antibody binding we perform automated segmentation of cells and classify them into infected and non-infected based on the dsRNA staining. We then measure fluorescence intensities in the serum channel per cell as a proxy for the amount of bound antibodies for both infected and non-infected cells. The ratio between these values for infected and non-infected cells is used to score the SARS-CoV-2 antibody response. To enable training of a machine learning approach for cell segmentation and to directly evaluate infected cell classification, we have manually labeled cells and annotated them as infected/non-infected in ten images, chosen from five positive and five control samples. Figure 6.7 presents an overview of all analysis steps; the full description of all steps can be found in Section A.6.1. Briefly, our approach works as follows:

First, we manually discard all images that contain obvious artifacts such as large dust particles or dirt and out-of-focus images. Then, images are processed to correct for the uneven illumination

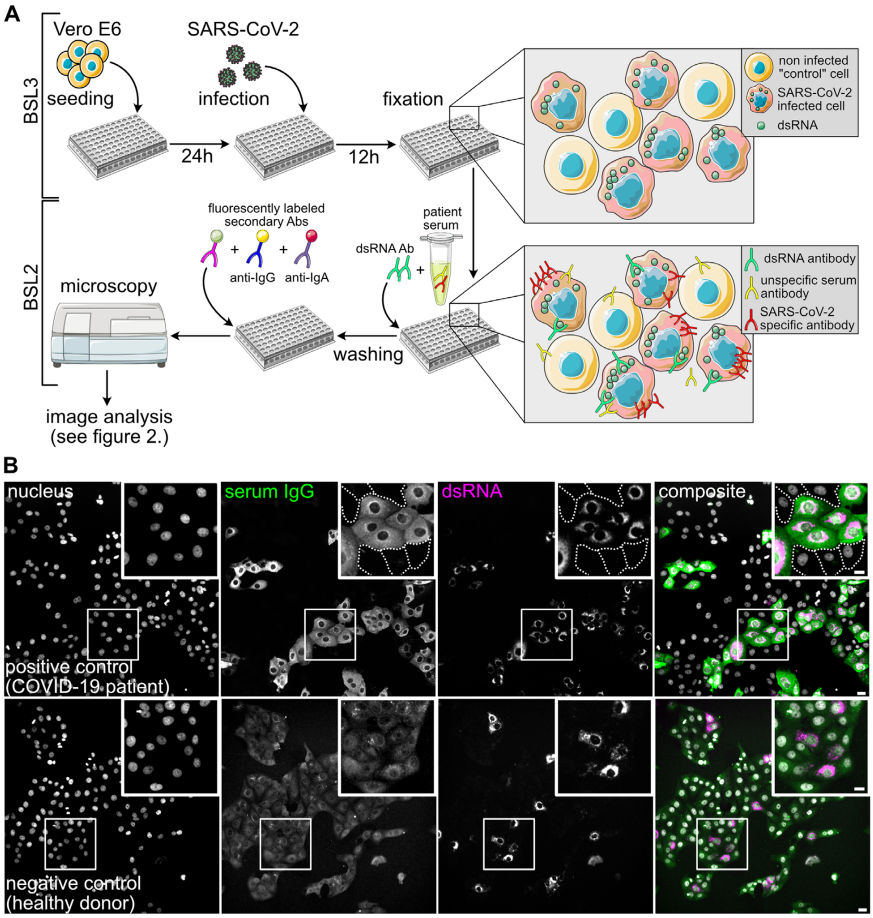



Figure 6.6: Principle of the immunofluorescence assay for SARS-CoV-2 antibody detection. (A) Scheme of the IF workflow and the SARS-CoV-2 antibody detection. (B) Representative images showing immunofluorescence results using a COVID-19 patient serum (positive control, upper panels) and a negative control serum (lower panels), followed by staining with a anti-IgG secondary antibody. Nuclei (gray), IgG (green), dsRNA (magenta) channels and a composite image are shown. White boxes mark the zoomed areas. Dashed lines mark borders of non-infected cells that are not visible at the chosen contrast setting. The insets show that infected cells have a much higher intensity than non-infected cells in the serum channel for the positive control, while showing the same intensities in the negative control.

profile in each channel. Next, we segment individual cells with a seeded watershed algorithm [25], using nuclei segmented via StarDist [191] as seeds and boundary predictions from a U-Net [180, 236] as height map. We have evaluated this approach using leave-one-out cross-validation and measured an average precision [59] of 0.77 ± 0.08 (i.e., on average 77% of segmented cells are matched correctly to the corresponding cell in the annotations). Combined with extensive automatic quality control, which discards outliers in the results, the segmentation was found to be of sufficient quality for our analysis. Especially since robust intensity measurements are used to reduce the effect of remaining errors.

We then classify the segmented cells into infected and non-infected, by measuring the 95th percentile intensities in the dsRNA channel and classifying cells as infected if this value exceeded 4.8 times the noise level, determined by the mean absolute deviation. This factor and the percentile were determined empirically using grid search on the manually annotated images (see above). Using leave-one-out cross validation on the image level, we found that this approach yields an average F1-score of 84.3%.

In order to make our final measurement more reliable, we discard whole wells, images or individual segmented cells based on quality control criteria that were determined by inspection of initial results. Those criteria include a minimal number of non-infected cells per well; minimal and maximal number of cells per image; minimal cell intensities per image; and minimal and maximal sizes of individual cells.

To score each sample, we computed the intensity ratio:

$$r = \frac{mI}{mN} \tag{6.1}$$

Here, $mI$ is the median serum intensity of infected cells and $mN$ the median serum intensity of non-infected cells. For each cell, we compute this intensity via the mean pixel intensity in the serum channel, excluding the nucleus area, where we did not observe serum binding. This value is subtracted by the background intensity, which is measured on two control wells that do not contain any serum per plate. See Figure 6.8 for example segmentations and the corresponding scores.

We use efficient implementations for all processing steps and deploy the analysis software on a computer cluster to rapidly process the large amounts of image data. For visual inspection, we have further developed an open-source software tool (PlateViewer) for interactive visualization of high-throughput microscopy data. It can be used in a final quality control step to visually inspect positive hits. For example, inspection with PlateViewer allowed us to identify a characteristic spotted pattern co-localizing with the dsRNA staining, that was sometimes observed in the IgA channel upon staining with negative control serum. Samples displaying this pattern were discarded from further analysis.

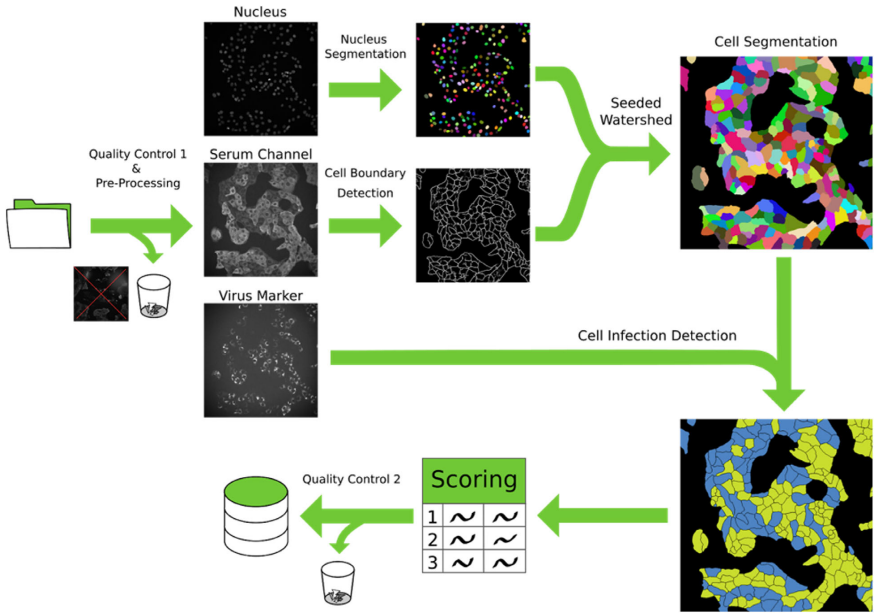



Figure 6.7: Schematic overview of the image analysis pipeline. Initially, images are subjected to the manual quality control, where images with acquisition defects are discarded. A pre-processing step is then applied to correct for barrel artifacts. Subsequently, a cell segmentation is obtained via seeded watershed, using a nucleus instance segmentation as seeds and boundary probabilities computed with a neural network as height map. Lastly, using the virus marker channel, each cell is classified as infected or not infected and the infection score is computed. A final automated quality control identifies and automatically discards non-conform results. All intermediate results are saved in a database to ensure full reproducibility of the results.

### 6.3.3 Assay Characterization and Validation

Here, we characterize our proposed assay and compare it with an established ELISA test approved for diagnostic use[5]. A main concern regarding serological assays for SARS-CoV-2 antibody detection is the occurrence of false positive results. In particular, these false positives may originate from cross-reactivity of antibodies that originated from infection with any of the four types of common cold Corona viruses (ccCoV) circulating in the population, as is the case for SARS-CoV and MERS-CoV [143]. Also, acute infection with Epstein-Barr virus (EBV) or cytomegalovirus (CMV) may result in unspecific reactivity in human sera [116, 69]. We therefore selected negative control samples consisting of 218 sera collected before the fall of 2019, comprising samples from healthy donors (n = 105, cohort B), patients that tested positive

[5]Euroimmun, Lübeck, Germany

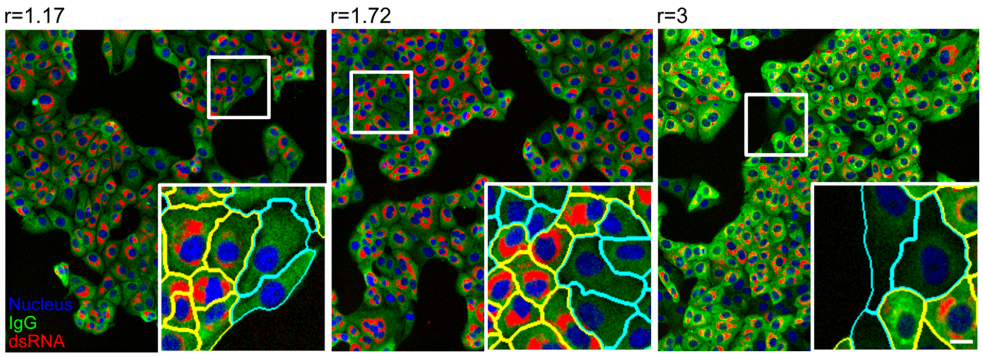



Figure 6.8: Examples of results from the automated image analysis pipeline. Panels display images that correspond to three different ratio scores, indicated above the images. White boxes mark the zoomed area. Cells in the insets are highlighted with yellow or cyan boundaries, indicating infected and non-infected cells, respectively.

for ccCoV several months before the blood sample was taken (n = 34, all four types of ccCoV represented; cohort A), as well as patients with diagnosed Mycoplasma pneumoniae (n = 22; cohort Z), EBV or CMV infection (n = 57, cohort E). We further selected 57 sera from 29 PCR confirmed COVID-19 patients, collected at different days' post symptom onset, as positive samples (cohort C).
Based on the scores (Equation 6.1) obtained for the samples of cohorts A, B, C and Z we defined the threshold separating negative from positive scores for each of the antibody channels. Since we envision the use of the assay for screening approaches, we decided to assign more weight to specificity at the cost of sensitivity and arrived at thresholds 1.39, 1.31, and 1.27 for IgA, IgG and IgM antibody channels based on ROC curve analysis [245]. For more details on this analysis, please refer to Section A.6.2. We validated the classification performance on negative control cohort E, that was not used for threshold selection, and detected no positive scores.

**Assay specificity characterization** While the majority of the control samples tested negative in ELISA measurements as well as in the IF analyses, some false positive readings were obtained in each of the assays, in particular in the IgA specific analyses, see Table 6.1. Roughly 10.6 % (IgA) or 3 % (IgG) of the samples were classified as positive or potentially positive by ELISA. The respective proportion of false-positive obtained based on IF, 0% for IgA and 0.9% for IgG, were lower, indicating higher specificity of the IF readout compared to the ELISA measurements. Importantly, false positive readings did not correlate between ELISA and IF, see Figure A.10. We conclude that applying both methods in parallel and using

the “double positive” definition for classification notably improves specificity of SARS-CoV-2 antibody detection.

| Negative cohort | IF IgM | IF IgA | IF IgG | ELISA IgA | ELISA IgA |
|---|---|---|---|---|---|
| B (n=105) | 1 | 0 | 1 | 7 | 5 |
| A (n=34) | 0 | 0 | 1 | 3 | 1 |
| Z (n=22) | 0 | 0 | 0 | 2 | 0 |
| E (n=57) | 0 | 0 | 0 | 11 | 1 |
| Total (n=218) | 1 (0.5%) | 0 (0.0%) | 2 (0.9 %) | 23 (10.6%) | 7 (3.2%) |

Table 6.1: Summary of positive results obtained by ELISA and IF for the negative control samples. The classification of ELISA results followed the definition of the test manufacturer. Cohort B: healthy donors, cohort A: patients that tested positive for ccCoV, cohort Z: patients with diagnosed Mycoplasma pneumoniae, cohort E: patients with EBV or CMV infection.

**Assay sensitivity characterization** In order to determine the sensitivity of our IF assay, we used the positive samples from cohort A. These samples have been collected from patients within 5 to 27 days post symptom onset. Again, samples were measured both in IF and ELISA. For an assessment of sensitivity, we stratified the samples according to the day post symptom onset, as shown in Table 6.2. The correlation between the semi-quantitative values was assessed in Figure A.11. For both methods, and for all antibody classes, mean values and the proportion of positive samples increased over time. In all cases, only positive values were obtained for samples collected later than 14 days post symptom onset, in accordance with other reports [163, 131]. At the earlier time points (up to day 14), a similar or higher proportion of positive samples was detected by IF compared to ELISA for IgG.

| Days after symptoms | IF IgM | IF IgA | IF IgG | ELISA IgA | ELISA IgG |
|---|---|---|---|---|---|
| <11 (n=17) | 7 (41%) | 9 (53%) | 7 (41%) | 11 (65%) | 3 (18%) |
| 11-14 (n=24) | 18 (75%) | 19 (79%) | 19 (79%) | 19 (79%) | 16 (67%) |
| >14 (n=16) | 16 (100%) | 16 (100%) | 16 (100%) | 16 (100%) | 16 (100%) |
| Total (n=57) | 42 (73%) | 44 (77%) | 42 (73%) | 46 (80%) | 34 (60%) |

Table 6.2: Positive results obtained for sera from COVID-19 patients collected at the indicated days post symptom onset.

# 7 Conclusion

In this thesis, we have established a method for boundary based segmentation problems in EM and other microscopy modalities based on the Lifted Multicut partition problem (Chapter 2). In order to scale this approach to large microscopy data-sets, we have introduced a block-wise solver for Mulicut and Lifted Multicut, that enables solving previously infeasible problems (Chapter 3). In addition, we have shown in Chapter 4, how lifted edges can be used to express domain specific knowledge and thus significantly improve the segmentation accuracy by making use of information not available locally. We have also introduced the Mutex Watershed (Chapter 5), an algorithm that, combined with high quality predictions from a CNN, enables instance segmentation directly from pixels, eliminating the often brittle step of seed generation for "normal" watersheds. Finally, we have applied these methods in several collaborative studies (Chapter 6), demonstrating their utility in biological research. In summary, our contributions enable instance segmentation for large microscopy data-sets, the previously missing link in automated image analysis for such data-sets. Given the increasing acquisition throughput and field of view of modern microscopes, we envision that our contribution will provide crucial help in the analysis of many such data-sets for a wide range of biological applications.

Largely due to the recent adoption and improvement of deep learning methods, the segmentation quality is often sufficient for downstream analysis tasks directly, or with only minor corrections. These methods, however, need large amounts of training data to produce such high-quality segmentations. Given the elimination of scaling as a bottleneck, generating enough training data has become the new rate-limiting step for (high-quality) automated instance segmentation. Especially, since ground-truth segmentations can only be produced by domain experts for many microscopy segmentation problems. Recent advances in self-supervised learning for image classification [40, 84, 70, 77] offer a perspective solution. By solving an unsupervised contrastive learning task, these methods learn to generate representations that can be fine-tuned for the target task with only small amounts of extra training data. Astonishingly, they often produce results of the same, or even higher, quality than methods trained fully supervised. In a very recent contribution, the authors of [216] have adapted these methods to semantic segmentation for natural images with some success. We are optimistic that similar methods will also be applicable in microscopy and in fact have contributed to preliminary work towards this goal [235].

Here, Mutex Watershed offers an interesting perspective: this algorithm is fast enough to

provide user corrections for instance segmentation in an interactive fashion. In fact, we have already developed a prototype for a correction tool based on this idea[1]. In the future, we hope to use this approach for providing the labels necessary to fine-tune unsupervised representations in an interactive manner.
Another promising future direction of research is the extension of the block-wise Multicut solver. On the one hand, it could be extended to other Multicut applications than instance segmentation by finding suitable spatial domains. On the other hand, this algorithm could be extended to more generalizations of the Multicut, such as higher order Multicuts [95], higher order Lifted Multicuts [101] or Multicut formulations for joint node partitioning and labeling [123, 113].

[1] https://github.com/constantinpape/affogato/blob/master/example/interactive/interactive_napari.py

# Appendices

# A Appendix

## A.1 Software

The methods developed as part of this thesis are made available as open-source software. They are already used by several other publications, e.g. [134, 119, 192, 226, 177, 16, 80] and as dependency of other software libraries: ilastik [22] and plantseg [236] use the Multicut functionality provided by elf (see below) and Neurokube [133], a framework for EM neuron segmentation via Kubernetes, makes use of our scalable segmentation implementation for large scale neuron segmentation.
While implementing our methods, we have used, whenever possible, available open source software. Especially numpy [73], scipy [221], scikit-image [215], scikit-learn [171] and vigra [109] are used for numerical algorithms, image analysis methods and machine learning algorithms. We made heavy use of napari [196] for visualization and use pytorch [170] for training and inference of neural networks.
Novel algorithms are implemented in C++ for efficiency reasons. They have either been contributed to nifty[1], a library for image segmentation and graph partitioning initially developed by Thorsten Beier or were implemented in affogato[2], a library for affinity based segmentation methods that we have implemented together with Steffen Wolf. Both libraries make use of xtensor[3] for multi-dimensional arrays and pybind11 [4] to generate python bindings.
For convenient training of neural networks, we have implemented torch-em[5], a wrapper around pytorch that provides commonly used functionality for microscopy segmentation.
To make the segmentation functionality and other methods developed in this thesis available in a convenient and unified manner, we have implemented elf[6]. This python library offers a high-level functional interface for the segmentation algorithms and other image analysis tools.

[1] https://github.com/DerThorsten/nifty
[2] https://github.com/constantinpape/affogato
[3] https://github.com/xtensor-stack/xtensor
[4] https://github.com/pybind/pybind11
[5] https://github.com/constantinpape/torch-em
[6] https://github.com/constantinpape/elf

### A.1.1 Chunked Data Storage

Efficient access to large volumetric data is often implemented using a chunked multi-dimensional data format. Instead of contiguously storing the data, these formats store it in multi-dimensional chunks of a fixed size. This data layout enables efficient access to sub-volumes because only chunks overlapping with the request have to be loaded, see Figure A.1.

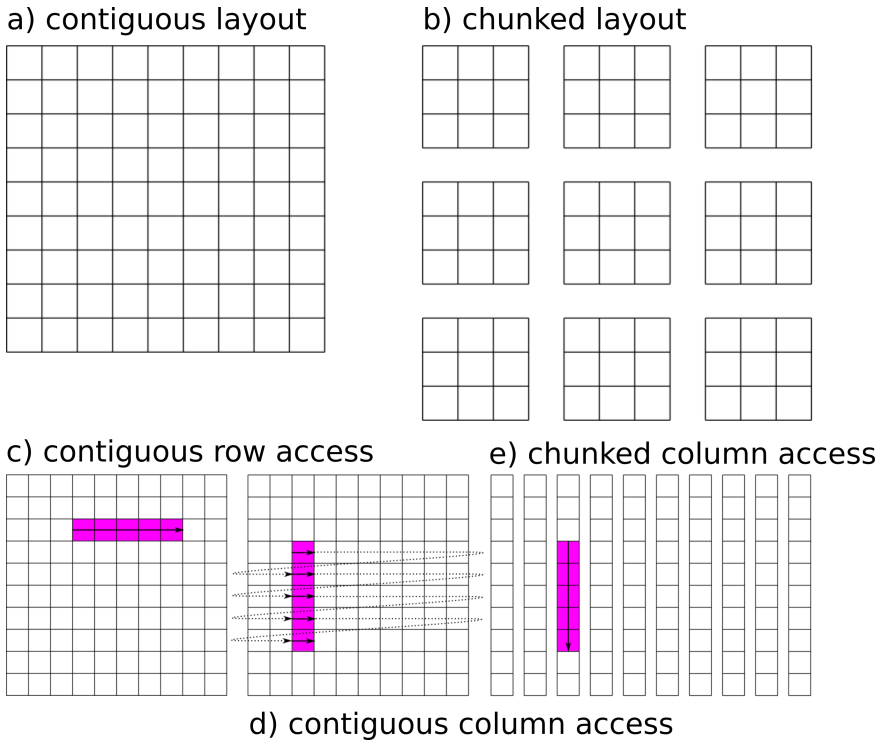



Figure A.1: Multi-dimensional (nd) data storage layouts. Example shown in 2d, extension to higher dimensions is trivial. (a) Contiguous storage: the data is serialized as single buffer, here using row-major layout. (b) Chunked storage: the data is serialized into multiple buffers, one for each chunk of fixed nd size. (c) Rows can be efficiently accessed in contiguously stored data. (d) Accessing columns is inefficient, because all interleaving rows (dashed lines) have two be loaded. (e) Loading columns is efficient for chunked storage and appropriate chunk size. The panels are extracted from `https://support.hdfgroup.org/HDF5/doc/Advanced/Chunking`.

The most popular for chunked data storage is HDF5 [63]. It implements it using a single file filesystem that also support hierarchical grouping of data containers. In addition, it enables

compression by applying compression filters to chunks. Chunks can be read from the data in parallel.
However, placing all chunks in a single file introduces two important drawbacks: it is not, naively[7], possible to write data in parallel, even if the chunks to be written do not overlap. It is also not straight-forward to access individual chunks in HDF5 files in a web store, for example an AWS S3 bucket.
To tackle these issues, two file formats, n5 [29] and zarr [147], have recently emerged in the bio-imaging community. Instead of storing chunks in a single file, they store them as individual files and also implement compression on a per chunk basis. This design enables naive parallel read *and* write operations and access in a web store on a per chunk basis. The effort to further standardize and merge these data formats, with the stated goal to establish a standard file format for microscopy images, is ongoing [148]. Note that HDF5 has introduced so-called region references [63], which enable placing chunks in different files as well. However, this feature is not supported by all HDF5 implementations yet and thus not widely adopted in the bio-imaging community.
In this thesis, we make heavy use of these emerging data-formats to enable parallel I/O. These data formats also enable the cloud based data access for MoBIE, see Section A.1.3. We have also implemented z5[8], a C++ library to read and write these file formats. It also offers python bindings that implement the same interface as the popular h5py[9] library.

### A.1.2 Distributed Computation

While the software for instance segmentation that we have described in the first paragraphs of Section A.1 is implemented with efficiency in mind, it does not, on its own, scale to large microscopy data-sets. In particular, it must be possible to process data that is significantly larger than main memory to deal with such data-sets. To this end, we have implemented cluster_tools[10], a high-level python library that implements our instance segmentation methods for distributed computing. It makes use of the n5 file format (see previous section) to enable processing volumetric data of arbitrary size, making use of chunked parallel read and write operations. It uses luigi[11] to build workflows and cache progress. It supports slurm and lsf compute clusters and is extensible to other scheduling systems.

[7] HDF5 can be used with MPI to support parallel write access [63]. This mode of operation is however not widely used by the bio-imaging community, which uses a wide variety of programming languages and tools, which often do not support MPI.
[8] https://github.com/constantinpape/z5
[9] https://github.com/h5py/h5py
[10] https://github.com/constantinpape/cluster_tools
[11] https://github.com/spotify/luigi

For embarrassingly parallel tasks, such as neural network inferences, it relies on inter-process communication through files to map each worker to its assigned part of the input volume. For some of the computational steps a map-reduce like approach is necessary. Take the example of building a region adjacency graph from a segmentation. In a first step, the sub-graphs for all chunks are computed using the embarrassingly parallel approach. In a second step the complete graph is constructed by iterating over the sub-graphs, identifying the unique nodes and edges and introducing a consecutive edge index. The "reduction" steps are implemented in nifty and run on a single node.
Within cluster_tools we provide scalable implementations of neural network inference, watershed segmentation, (Lifted) Multicut problem extraction, the block-wise (Lifted) Multicut solver and many other image analysis algorithms. See Section 3.4.3 for performance measurements when applied to a large segmentation problem.

### A.1.3 MoBIE and Data Sharing

We have developed MoBIE as part of [218] to explore and share this large data-set. Since then, we have extended the tool significantly and it is also being used for several other projects, including [47].
In a nutshell, MoBIE consists of two parts: first, a data specification for large image data, making use of chunked data formats (see Section A.1.1) and multi-scale image pyramids. It also supports tables, stored as tab separated values and defines metadata, stored according to a json schema. Second, a viewer that can load data stored according to the MoBIE specification, both from the filesystem or a S3 compatible web object store.
The viewer is available as a Fiji plugin [188]. It uses BigDataViewer [174] to display the image data. It implements additional functionality for exploring segmentations and data associated with the segmented objects, which is stored in tables. It further supports 3D rendering of segmented objects and image volumes, a convenient annotation mode, interactive scatter plots, grid views and bookmarks that can be used to serialize and recreate the full viewer state. Figure A.2 shows selected functionality for the data of [218].
The development of MoBIE has been joint work with Christian Tischer and Kimberly Meechan. It is available as open source software at `https://mobie.github.io/`; the Fiji viewer can be installed from the MoBIE Fiji update site.

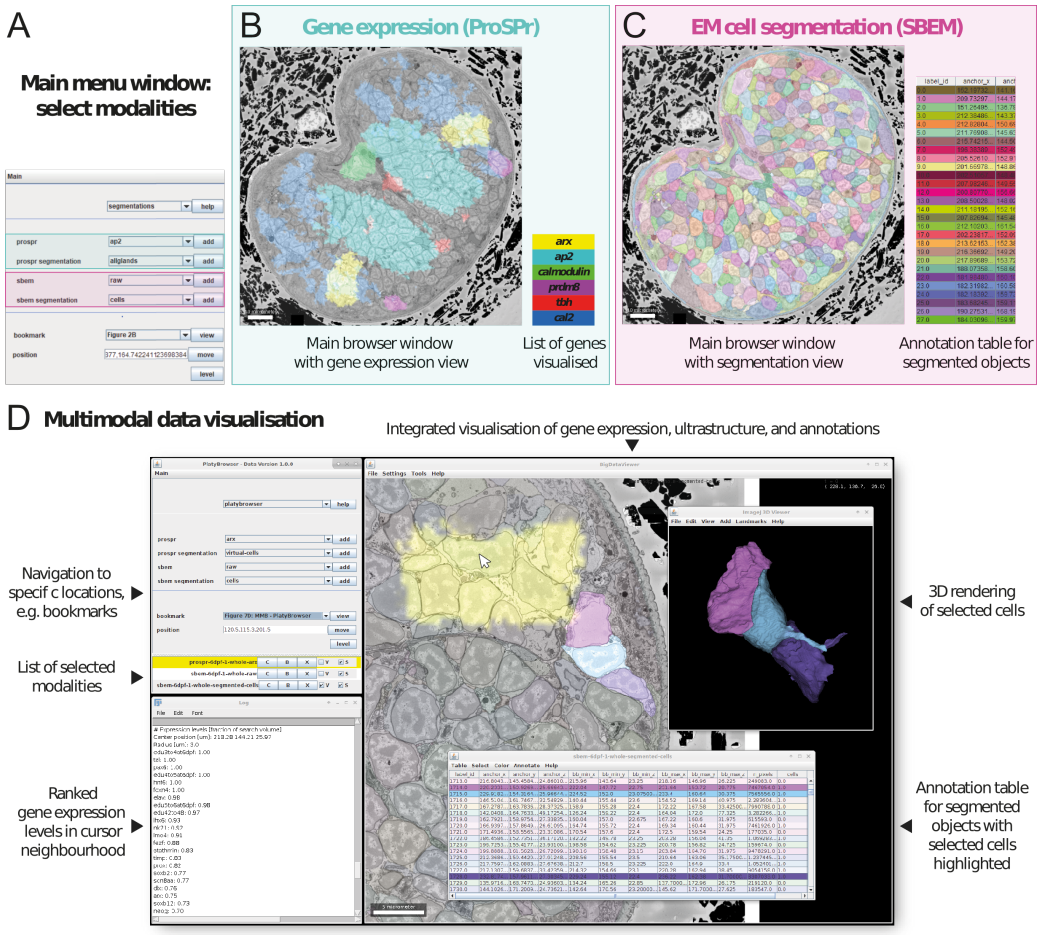



Figure A.2: Core functionality of MoBIE with example data from [218]: (A) User interface to select image sources, change their appearance, and navigate to specific locations in the animal. (B) viewer window showing the SBEM image in a region of the animals head, with the gene expression signal for six different genes. (C) viewer window of the same section as in (B), displaying the cellular segmentation and a table storing attributes for the segmented objects. (D) Screenshot of the full user interface illustrating the integration of modalities and additional functionalities: the expression of gene “arx” is shown in yellow; three segmented neurons are shown next to it; below is shown the annotation table highlighting the rows that correspond to the objects selected. Next to it, the 3D Viewer window shows a rendering of the selected cells; the colors for a given object are identical in the 2D Viewer overlay, 3D rendering and table. Below the main menu, the log window shows a ranked list of the gene expression where the mouse cursor is positioned (white arrow).

## A.2 Multicut and Lifted Multicut for EM Segmentation

**Training hyperparameters of the lCv1 network:** The network was trained using pixel-wise binary cross-entropy loss. Binary boundary / no-boundary target labels were smoothed [203] by computing the pixelwise negative exponential of the euclidean distance transform, The raw image batches were normalized to zero mean and unit variance. The optimizer Adam [104] was used with the following optimization hyperparameters: step size $\alpha = 0.0002$, exponential decay rates for moment estimates $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a fuzz factor $\epsilon = 10^{-8}$ for numerical stability. A L2 weight decay term with coefficient $\lambda = 0.0005$ was added to the objective and dropout [199] ($p = 0.5$) was used.

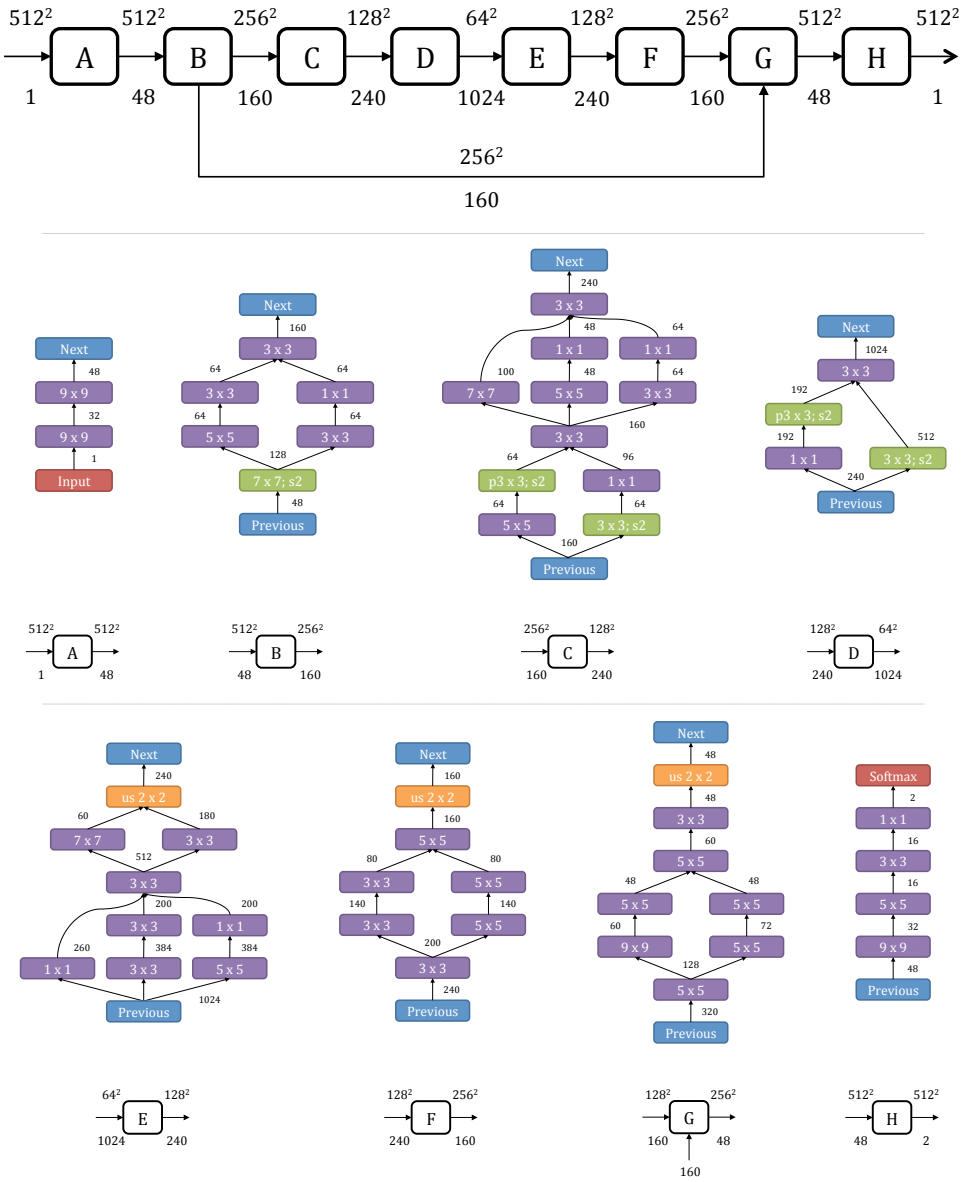



Figure A.3: ICv1 neural network architecture. Purple boxes represent convolutional layers with kernel size $M \times N$, ELU non-linearity, and "same" border handling (i.e. the input and output images have identical shapes). Green boxes annotated with $pM \times N;\ sS$ denote max-pooling layers over $M \times N$ patches with stride $S$; while those annotated with $M \times N;\ sS$ represent convolutional layers with stride $S$. Both layers reduce the image width and height by a factor $S$. Orange boxes annotated $us\ M \times N$ show upscaling layers, where pixels in the input feature maps are repeated $M$ times vertically and $N$ times horizontally. The numbers next to arrows specify the number of feature maps flowing from one layer to the next. Boxes with $M$ outgoing arrows imply that the output of the layer is replicated $M$ times, while boxes with $N$ incoming arrows imply that the layer receives $N$ inputs, which are concatenated depth-wise (i.e. along the feature axis) before being processed.

## A.3 Scalable Multicut and Lifted Multicut Segmentation

### A.3.1 Performance of extant Multicut Solvers

In addition to the results in Table 3.1, we plot the energy versus runtime for all three problems in Figure A.4, Figure A.5 and Figure A.6. The initial decrease in energy afforded by the greedy solvers *gaec* and *gf* happens on a much faster time-scale compared to the subsequent decrease by the other solvers. Here, *gaec* converges faster, and to a lower energy, than *gf*. The *kl* solver converged within less than four minutes for all three problems and affords a further improvement in energy compared to *gaec*. The *fm* based solvers take longer to converge and afford only a small improvement in energy; here *fm_kl* offers the best trade-off between improvement and additional runtime. The *cgc* solver also provides only a marginal improvement in the energy while taking long to converge.

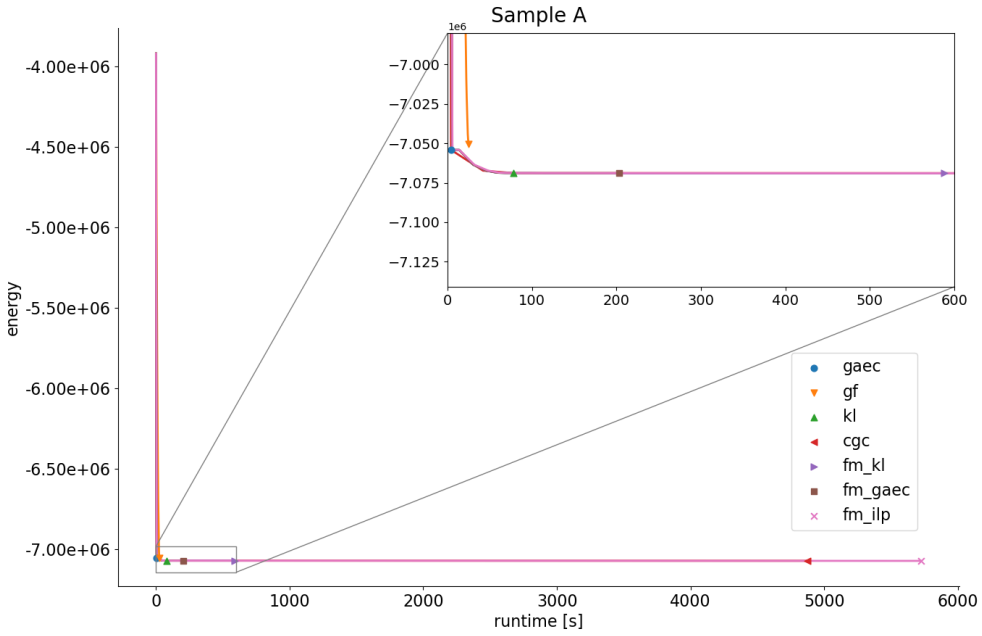



Figure A.4: Performance of different solvers for the Sample A Multicut problem.

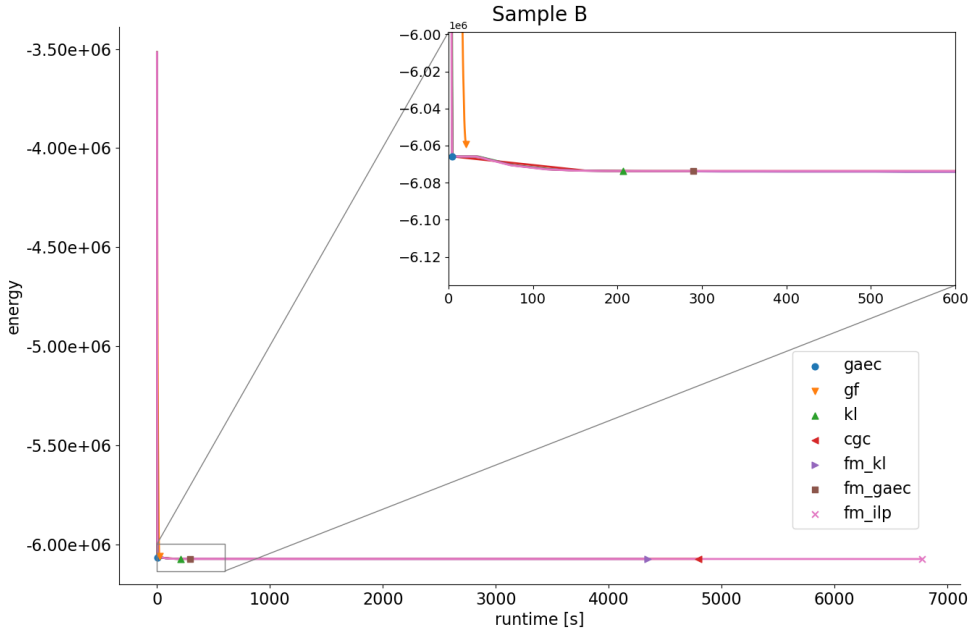



Figure A.5: Performance of different solvers for the Sample B Multicut problem.

### A.3.2 Performance and Accuracy of the Block-wise Multicut Solver

In addition to the comparison of the block-wise Multicut solver to other methods in Table 3.3, we compare different configurations of our solver. Table A.1 compares the performance and accuracy for iterating the extraction and reduction step up to three times. Table A.2 compares different block sizes for sub-problem extraction. Table A.3 compares different solvers for solving the sub-problems and solving the reduced global problem.
We find that the energy of the partition decreases with increasing number of iterations. This is expected, as the solver moves further away from the optimal solution when solving a global problem that is further reduced. However, the less optimal solution does not have an adverse effect on the segmentation quality; the scores are on par for all three configurations. For the medium size problems the runtime is roughly the same for the different iteration numbers. Note that this does not hold true for larger problems, where the global solution starts to dominate the runtime, see Section 3.4.3.
The size of the blocks used for sub-problem extraction has a more significant influence on the

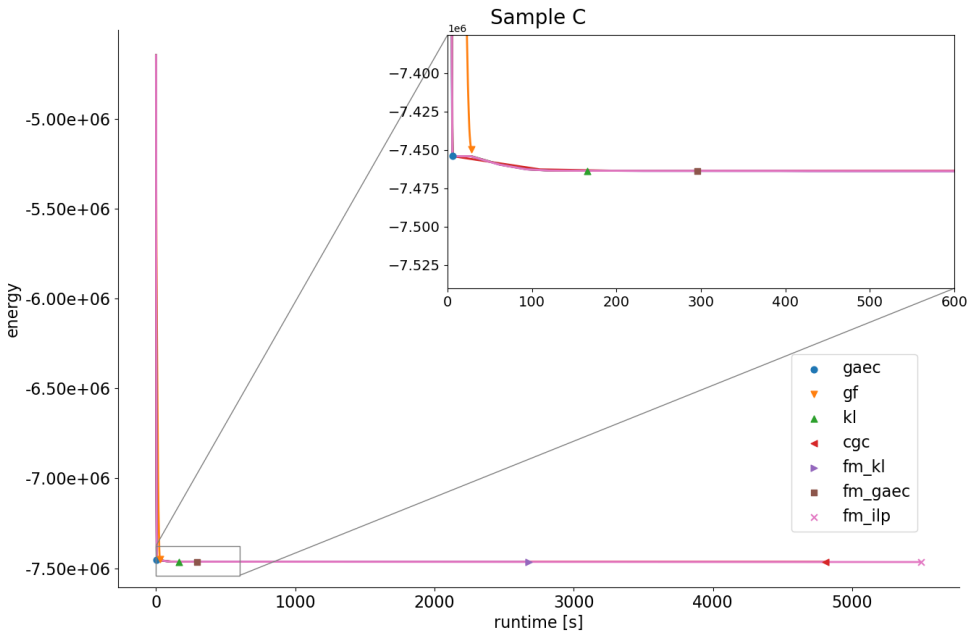



Figure A.6: Performance of different solvers for the Sample C Multicut problem.

segmentation quality. Here, quite surprisingly, using the smallest block size yields a significantly better score for Sample A and B, while achieving a worse energy. This fact hints at a modeling issue, i.e. incorrect edge costs that result in the optimal Multicut solution not corresponding to the best segmentation. While this effect needs further study, it is likely to be data-set specific. Our initial hypothesis is that it is caused by EM imaging artifacts and the resulting low-quality network predictions. This explanation is consistent with the observation that these artifacts are more prevalent in Samples A and B compared to Sample C. Furthermore, the runtime increases significantly for the largest block size.

The choice of different solvers for sub-problems and global problem does not have a large influence on the segmentation quality. Only using *kl* for both leads to a slightly worse score compared to the other combinations. Using the expensive *ilp* solver for sub-problems leads to a significant increase in runtime.

| | Sample A | | | Sample B | | | Sample C | | |
|---|---|---|---|---|---|---|---|---|---|
| Iterations | Time | Energy | Score | Time | Energy | Score | Time | Energy | Score |
| 1 | 88.0 | -7.0600e+06 | 0.7187 | 130.0 | -6.0614e+06 | 0.8429 | 128.0 | -7.4506e+06 | 1.0777 |
| 2 | 79.0 | -7.0590e+06 | 0.7078 | 117.0 | -6.0598e+06 | 0.8538 | 106.0 | -7.4492e+06 | 1.0809 |
| 3 | 91.0 | -7.0587e+06 | 0.7009 | 140.0 | -6.0595e+06 | 0.8432 | 118.0 | -7.4491e+06 | 1.0858 |

Table A.1: Different number of problem reduction iterations. We have used an initial block size of 50 $\times$ 512 $\times$ 512 pixels and increase it by a factor of 2 in each dimension for every iteration. The solver *fm_ilp* is used to solve sub-problems and *fm_kl* to solve the global problem. Sub-problem extraction and reduction have used 24 workers for paralleization.

| | Sample A | | | Sample B | | | Sample C | | |
|---|---|---|---|---|---|---|---|---|---|
| Blocksize | Time | Energy | Score | Time | Energy | Score | Time | Energy | Score |
| 25, 256, 256 | 67.0 | -7.0524e+06 | 0.5624 | 182.0 | -6.0419e+06 | 0.7821 | 139.0 | -7.4346e+06 | 1.0441 |
| 50, 512, 512 | 89.0 | -7.0600e+06 | 0.7189 | 127.0 | -6.0614e+06 | 0.8429 | 170.0 | -7.4506e+06 | 1.0777 |
| 100, 1024, 1024 | 2423.0 | -7.0735e+06 | 0.6702 | 1677.0 | -6.0743e+06 | 0.8286 | 2012.0 | -7.4652e+06 | 1.0986 |

Table A.2: Different blocksizes. The solver *fm_ilp* is used to solve sub-problems and *fm_kl* to solve the global problem. Sub-problem extraction and reduction have used 24 workers for paralleization. We have only used a single extraction and reduction iteration.

| | Sample A | | | Sample B | | | Sample C | | |
|---|---|---|---|---|---|---|---|---|---|
| Solvers | Time | Energy | Score | Time | Energy | Score | Time | Energy | Score |
| fm_ilp-decomp | 65.0 | -7.0599e+06 | 0.7189 | 86.0 | -6.0614e+06 | 0.8429 | 78.0 | -7.4506e+06 | 1.0777 |
| fm_ilp, fm_kl | 88.0 | -7.0600e+06 | 0.7136 | 124.0 | -6.0614e+06 | 0.8429 | 123.0 | -7.4506e+06 | 1.0777 |
| fm_kl, kl | 13.0 | -7.0584e+06 | 0.7305 | 17.0 | -6.0604e+06 | 0.8721 | 17.0 | -7.4496e+06 | 1.0590 |
| ilp-fm, kl | 77.0 | -7.0605e+06 | 0.7230 | 5192.0 | -6.0614e+06 | 0.8472 | 475.0 | -7.4520e+06 | 1.0560 |
| kl, kl | 8.0 | -7.0556e+06 | 0.7396 | 8.0 | -6.0588e+06 | 0.9361 | 6.0 | -7.4482e+06 | 1.0758 |

Table A.3: Different solvers for sub-problems and global problem. We have only used a single extraction and reduction iteration and a blocksize of 50 $\times$ 512 $\times$ 512 pixels. Sub-problem extraction and reduction have used 24 workers for paralleization.

### A.3.3 Scaling Behavior of the Block-wise Multicut Solver

In addition to the results in Section 3.4.3, we compare the Decomposition, using Kernighan-Lin as internal solver, with using Kernighan-Lin to solve the complete problem. As shown in Table A.4, the solution of both approaches is identical while the Decompositon solver is significantly faster due to the decomposition into sub-problems, see Table 3.5.

| | Kernighan-Lin | | Decomposition | |
|---|---|---|---|---|
| Level | Time [min] | Energy | Time [min] | Energy |
| 1 | 468.5 | -8.60967e+08 | 80.1 | -8.609670e+08 |
| 2 | 70.1 | -8.60410e+08 | 5.9 | -8.604099e+08 |
| 3 | 16.4 | -8.60232e+08 | 1.0 | -8.602317e+08 |

Table A.4: Solving large Multicut problem with Kernighan-Lin and Decomposition solvers. Computations were performed on an Intel Xeon Gold 6136 Processors, using a single thread for Kernighan-Lin and 8 threads for Decompositon.

## A.4 The Mutex Watershed

### A.4.1 Network Architecture and Training

We use the 3D U-Net [180, 35] architecture, as proposed in [67].
Our training targets for attractive / repulsive edges $\hat{w}^{\pm}$ can be derived from a ground truth label image $\hat{L}$ according to

$$\hat{w}_e^+ = \begin{cases} 1, & \text{if } \hat{L}_i = \hat{L}_j \text{with } e = e_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{A.1}$$

$$\hat{w}_e^- = \begin{cases} 0, & \text{if } \hat{L}_i = \hat{L}_j \text{with } e = e_{ij} \\ 1, & \text{otherwise} \end{cases} \tag{A.2}$$

Here, $i$ and $j$ are the indices of image pixels and $e_{ij}$ denotes the edge connecting them. Next, we define the loss terms

$$\mathcal{J}_c^+ = -\frac{\sum_{e \in H_c^+} (1 - w_e^+)(1 - \hat{w}_e^+)}{\sum_{e \in H_c^+} ((1 - w_e^+)^2 + (1 - \hat{w}_e^+)^2)} \tag{A.3}$$

$$\mathcal{J}_c^- = -\frac{\sum_{e \in H_c^-} w_e^- \hat{w}_e^-}{\sum_{e \in H_c^-} ((w_e^-)^2 + (\hat{w}_e^-)^2)} \tag{A.4}$$

for attractive edges (i. e. channels) and repulsive edges (i. e. channels). Equation A.3&A.4 is the Sørensen-Dice coefficient [52, 198] formulated for fuzzy set membership values and a product T-norm. During training we minimize the sum of attractive and repulsive loss terms $\mathcal{J} = \sum_c^{C^+} \mathcal{J}_c^+ + \sum_c^{C^-} \mathcal{J}_c^-$. This corresponds to summing up the channel-wise Sørensen-Dice loss. The terms of this loss are robust against prediction and/or target sparsity, a desirable quality for neuron segmentation: since membranes are very thin, they occupy very few pixels in the volume. More precisely, if $w_e^+$ or $\hat{w}_e^+$ (or both) are sparse, we can expect the denominator $\sum_e ((w_e^+)^2 + (\hat{w}_e^+)^2)$ to be small, which has the effect that the numerator is adaptively weighted higher. In this sense, the Sørensen-Dice loss at every pixel $i$ is conditioned on the global image statistics, which is not the case for a Hamming-distance based loss like Binary Cross-Entropy or Mean Squared Error.
We optimize this loss using the Adam optimizer and additionally condition learning rate decay on the Adapted Rand Score [13] computed on the training set every 100 iterations. During training, we augment the data set by performing in-plane rotations by multiples of 90 degrees, flips along the x- and y- axis as well as elastic deformations. At prediction time, we use test time data augmentation, presenting the network with seven different versions of the input obtained by a combination of rotations by a multiple of 90 degrees, axis-aligned flips and transpositions. The network predictions are then inverse-transformed to correspond to the original image, and the results averaged.

### A.4.2 Baseline Post-processing Methods

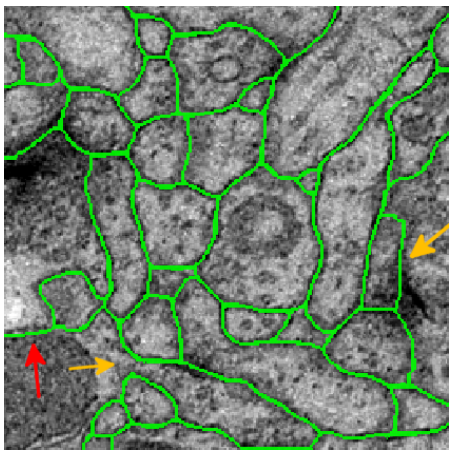
The predictions of out CNN can be post-processed directly by the Mutex Watershed algorithm. To ensure a fair comparison, we transform the same CNN predictions into a segmentation using other popular post-processing methods. We start from simple thresholding (THRESH) and seeded watershed. Since these cannot take long-range repulsions into account, we generate a boundary map by taking the maximum[12] values over the attractive (short-range) edge channels. Based on this boundary map, we introduce seeds at the local minima (WS) and at the maxima of the smoothed distance transform (WSDT). For both variants, the degree of smoothing was chosen such that each region receives as few seeds as possible, without however causing severe under-segmentation. The performance of these three baseline methods in comparison to Mutex Watershed is summarized in Table 5.1b. The methods were applied only in 2D, because the high

[12]The maximum is chosen to preserve boundaries.

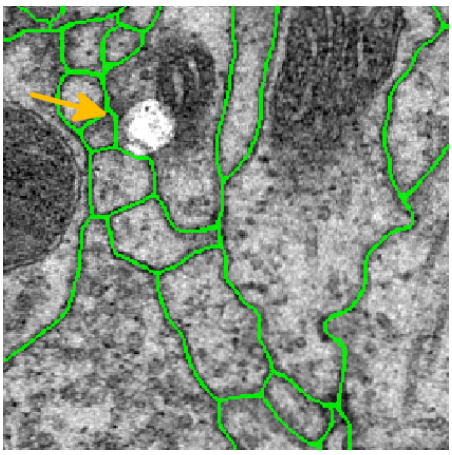
degree of anisotropy leads to deteriorating quality of results when applied in 3D. In contrast, the Mutex Watershed can be applied in 3D out of the box and yields significantly better 2D segmentation scores.
Qualitatively, we show patches from our results in Figure A.7. The major failure case for WS (Figure A.7e) and WSDT (Figure A.7f) is over-segmentation caused by over-seeding a region. The major failure case for THRESH is under-segmentation due to week boundary evidence (Figure A.7d). In contrast, the Mutex Watershed produces a better segmentation, only causing minor over-segmentation (Figure A.7a, Figure A.7b).
Note that, in contrast to most pixel-based post-processing methods, our algorithm can take long range predictions into account. To compare with methods which share this property, we turn to the Multicut and Lifted Multicut - based partitioning for neuron segmentations as introduced in [6] and [18]. We compute costs corresponding to edge cuts from the affinities estimated by the CNN via:

$$s_e = \begin{cases} \log \frac{w_e^+}{1-w_e^+}, & \text{if } e \in E^+ \\ \log \frac{1-w_e^-}{w_e^-}, & \text{otherwise,} \end{cases} \tag{A.5}$$

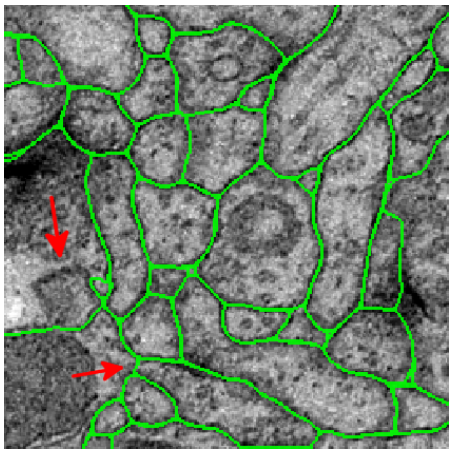
We set up two Multicut problems: the first is induced only by the short-range edges (MC-LOCAL), the other by short- and long-range edges together (MC-FULL). Note that the solution to the full connectivity problem can contain “air bridge”, i.e. pixels that are connected only by long-range edges, without a path along the local edges connecting them. However, we found this not to be a problem in practice. In addition, we set up a lifted Multicut (LMC) problem from the same edge costs.
Both problems are NP-hard, hence it is not feasible to solve them exactly on large grid graphs. For our experiments, we use the approximate Kernighan Lin [100, 102] solver. Even this allows us to only solve individual 2D problems at a time. The results for MC-LOCAL and MC-FULL can be found in Table 5.1b. The MC-LOCAL approach scores poorly because it under-segments heavily. This observation emphasizes the importance of incorporating the longer-range edges. The MC-FULL and LMC approaches perform well. However, the Mutex Watershed yields a better segmentation still, probably because it is evaluated for the full 3d problem, which is not feasible for the Multicut based segmentation.



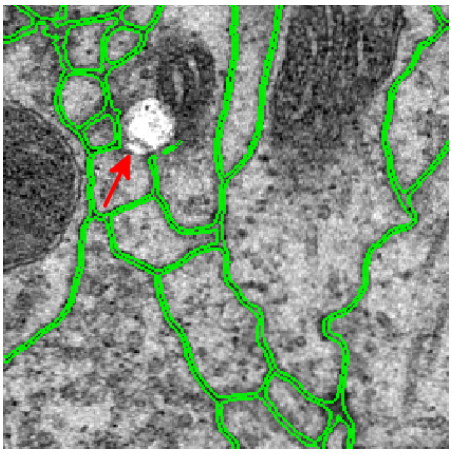
(a) Mutex Watershed



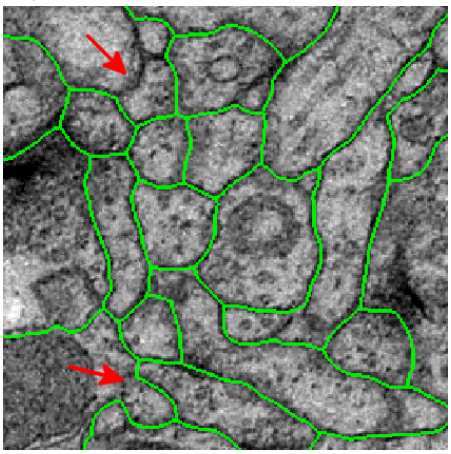
(b) Mutex Watershed



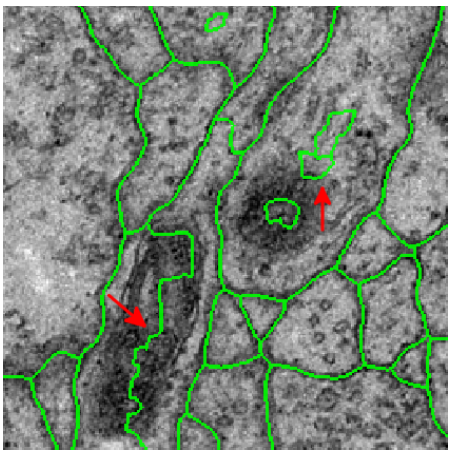
(c) Multicut partitioning based segmentation (MC-FULL)



(d) Thresholding of local boundary maps (THRESH)



(e) Watershed, seeded at local minima of the smoothed input map (WS)



(f) Distance Transformed Watershed (WSDT)



Figure A.7: Mutex Watershed and baseline segmentation algorithms applied on the ISBI Challenge test data. Red arrows point out major errors. Orange arrows point to difficult, but correctly segmented regions. All methods share the same input maps.

### A.4.3 Study on Natural Image Segmentation

Figure A.8 shows the segmentations we have obtained for different points of the phase transition diagram (Figure 5.6a).

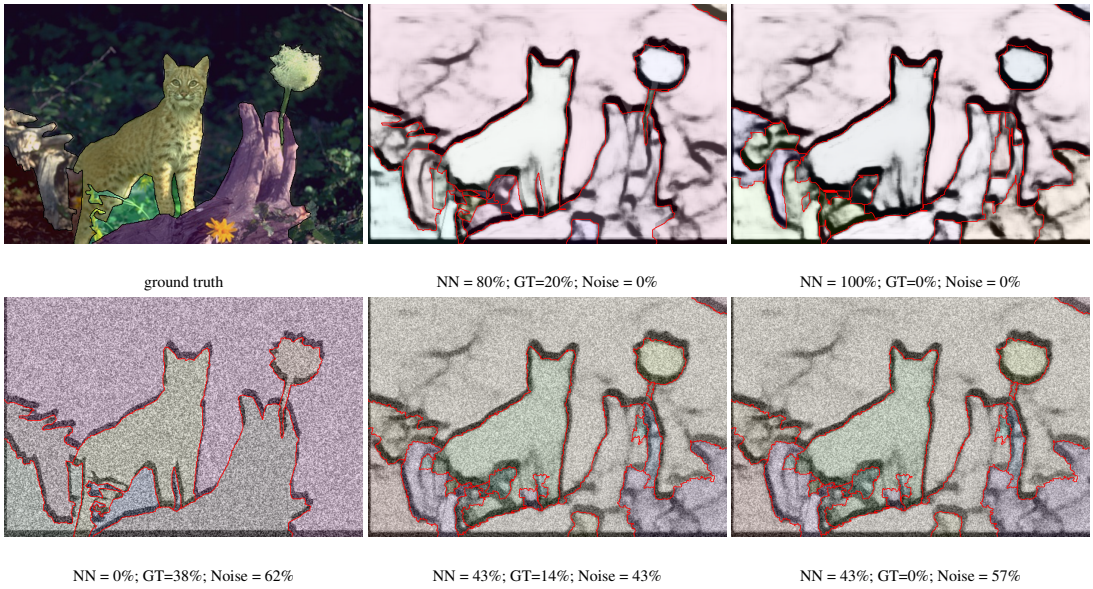



Figure A.8: MWS segmentations (red lines and color fill) of one arbitrary BSD500 image, given affinities (gray values) interpolated between neural net predictions (NN), ground-truth (GT) and uniform noise. The algorithm proved to be reasonably robust against noise in the affinities.

## A.5 A cellular Atlas for *Platynereis dumerilii*

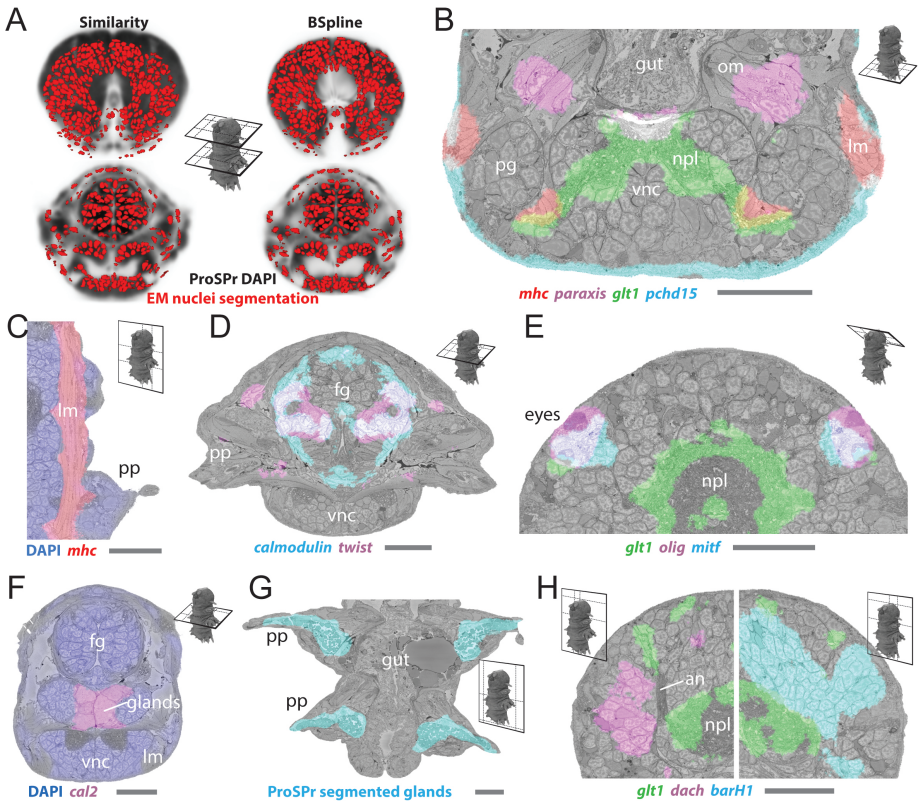



Figure A.9: Registration of gene expression signal to the EM data. (A) illustrates the registration procedure, where we register the DAPI signal to the segmented nuclei in EM, using a sequence of of non-deformable and deformable transformations. (B) to (H) show different gene expression maps overlaid with the EM data. The registration accuracy has a precision of about one cell radius and shows good adherence for know markers, such as myosin (mhc) for muscles in (C) or glutamate synthase (glt1) for neuropil in (E) and (H).

## A.6 Immunofluorescence based Testing for SARS-COV-2 Antibodies

### A.6.1 Quantitative Analysis

**Manual Annotations** Two of our processing steps require manually annotated data: in order to train the CNN used for boundary and foreground prediction, we need label masks for the individual cells. To determine suitable parameters for the infected cell classification, we need a set of cells classified as being infected or non infected. We have produced these annotations for 10 images with the following steps: first, we created an initial segmentation following the approach outlined in Section 6.3.2, using boundary and foreground predictions from the ilastik [22] pixel classification workflow, which can be obtained from a few sparse annotations. We then corrected this segmentation using the annotation tool BigCat[13]. After correction, we manually annotated these cells as infected or non-infected. Note that this mode of annotations can introduce two types of bias: the segmentation labels are derived from an initial segmentation. Small systematic errors in the initial segmentation, that were not found during correction, could influence the boundary prediction network. More importantly, when annotating the infected/non infected cells, both the serum channel and the virus marker channel have to be available to the annotators, in order to visually delineate the cells. This may result in subconscious bias, and the possibility of the observed intensity in the serum channel influencing the decision on the infection status of a cell.

**Preprocessing** We perform minimal preprocessing (i.e. flat-field correction) to compensate for uneven illumination of the microscope system [207]. First, we subtract a constant CCD camera offset $\text{ccd}_{\text{offset}}$. Then, we correct uneven illumination by dividing each channel by a corresponding corrector image $\text{flatfield}(x, y)$:

$$\text{processed}(x, y) = \frac{\text{raw}(x, y) - \text{ccd}_{\text{offset}}}{\text{flatfield}(x, y) - \text{ccd}_{\text{offset}}}. \tag{A.6}$$

The corrector image is obtained as the normalized average of all images of the corresponding channel, smoothed by a normalized convolution with a gaussian filter with a bandwidth of 30 pixels. This image has to be recomputed if the microscopy set-up changes. Full background subtraction is performed later in the pipeline using either the background measured on wells that (deliberately) do not contain any serum or, if not available, using a fixed value that was determined manually.

[13] https://github.com/saalfeldlab/bigcat

**Segmentation** Cell segmentation forms the basis of our analysis method. In order to obtain an accurate segmentation, we make use of both the DAPI and the serum channel. First, we segment the nuclei in the DAPI channel using the StarDist method [191] trained on data from [33]. Note that this method yields an instance segmentation: each nucleus in the image is assigned a unique ID. In addition, we predict per pixel probabilities for the boundaries between cells and for the foreground (i.e. whether a given pixel is part of a cell) using a 2D U-Net [180] based on the implementation of [236]. This method was trained using the ten annotated images, see above. The cells are then segmented by the seeded watershed algorithm [25]. We then use the nucleus segmentation, dilated by three pixels, as seeds and the boundary predictions as height map. In addition, we threshold the foreground predictions, erode the resulting binary image by 20 pixels and intersect it with the binarised seeds. The result is used as a foreground mask for the watershed. The dilation/erosion is performed to alleviate issues with very small nucleus segments/imprecise foreground predictions. In order to evaluate this segmentation method, we train ten different networks using leave-one-out cross-validation, training each network on nine of the manually annotated images and evaluating it on the remaining one. We measure the segmentation quality using average precision [59] at an intersection over union (IoU) threshold of 0.5. We measure a value of $0.77 \pm 0.08$ with the optimum value being 1.0.

**Infection Classification** To distinguish infected from non infected cells, we use the dsRNA virus marker channel: infected cells show a signal in this channel while the non-infected control cells should ideally not be visible, see Figure 6.8. We classify each cell in the cell segmentation individually, using the following procedure: first, we denoise the marker channel using a white tophat filter with a radius of 20 pixels. To account for inaccuracies in the cell segmentation (the exact position of cell borders is not always clear), we then eroded all cell masks with a radius of 5 pixels and thereby discard pixels close to segment boundaries. This step does not lead to information loss, since the virus marker is mostly concentrated around the nuclei. On the remaining pixels of each cell, we compute the 0.95 quantile $q$ of the intensity in the marker channel. For the background pixels (as predicted by the CNN), we compute the median intensity $b$ of the virus marker channel across all images in the current plate. Finally, we classify the cell as infected if $q$ exceeds $b$ by more than a given threshold:

$$q - \text{median}(b) > t. \tag{A.7}$$

For additional robustness against intensity variations, we adapt the threshold based on the variation in the background of the plate. Hence, we define it as a multiple of the mean absolute deviation of all background pixels of that plate with $M = 4.8$:

$$t = M \ \text{mad}(b). \tag{A.8}$$

To determine the optimal values of the parameters used in this procedure, we used the cells manually annotated as infected/non-infected and performed a grid search over the following parameters:

- quantiles: 0.9, 0.93, 0.95, 0.96, 0.97, 0.98, 0.99, 0.995
- $M$: 0 to 10 in intervals of 0.1

To estimate the validation accuracy, we performed leave-one-out cross-validation on the image level. This yields an average validation F1-score of 84.3%, precision of 84.3% and recall of 84.8%. These values are the arithmetic means of the individual results per split.

**Immunoglobulin Intensity Measurements** In order to obtain a relative measure of antibody binding, we determine the mean intensity and the integrated intensity in each segmented cell from images recorded in the IgG, IgA, or IgM channel. A comparative analysis revealed that the mean intensity was more robust against the variability of cell sizes, whereas using the integrated intensity as a proxy yielded a higher variance in non-infected cells. Thus, mean intensity per cell was chosen as a proxy for the amount of antibody bound. Non-specific auto-fluorescence signals require a background correction of the measured average serum channel intensities. For background normalization, we use cells (two wells per plate) that were not immunostained with primary antiserum. From this, we compute the background to be the median serum intensity of all pixels of images taken from this well. This value is subtracted from all images recorded from the respective plate. In case the control wells are not available, the background value for subtraction is determined manually.

**Scoring** The core interest of the assay is to measure the difference of antibody binding to cells infected with the coronavirus in comparison to non-infected cells. To this end, utilizing the results of the image analysis, we compute the following summary statistics of the background corrected antibody binding of infected cells, $I$, and of non-infected cells, $N$:

$$m_I = \text{median}(I) \tag{A.9}$$

$$m_N = \text{median}(N) \tag{A.10}$$

$$\sigma_N = \text{mad}(N) \tag{A.11}$$

Using these, the ratio $r$, difference $d$ and robust-z-score $z$ are computed:

$$r = \frac{m_I}{m_N} \tag{A.12}$$

$$d = m_I - m_N \tag{A.13}$$

$$z = \frac{m_I - m_N}{\sigma_N} \tag{A.14}$$

We compute above scores for each well and each image, taking into account only the cells that passed all quality control criteria (see below). While the final readout of the assay is well based, image scores are useful for quality control.

**Quality Control** We perform quality control of the images and analysis results at the level of wells, images and cells. The entities that do not pass quality control are not taken into account when computing the score during final analysis. We exclude wells that contain less than 100 non-infected cells, that have a median serum intensity of infected cells smaller than three times the noise level (measured by the median absolute deviation), or that have negative intensity ratios, which can happen due to the background subtraction. Out of 1736 wells corresponding to the samples processed as part of validation (Section 6.3.3), 94 did not pass the quality control, corresponding to 5.4% of wells. At the image level, we visually inspect all images and mark those that contain imaging artifacts using a viewer based on napari[14]. We distinguish the following types of artifacts during the visual inspection: empty, unstained or over-saturated images, as well as images covered by a large bright object. In addition, we automatically exclude images that contain less than 10 or more than 1000 cells. These thresholds are motivated by the observation that too few or too many cells often result from a problem in the assay. Thus, 296 of the total 15,624 images were excluded from further analysis, corresponding to 1.9% of images. Out of these, 295 were manually marked as outliers and only a single one did not pass the subsequent automatic quality control. Finally, we automatically exclude segmented cells with a size smaller than 250 pixels or larger than 12,500 pixels, that most likely correspond to segmentation errors. These limits were derived by the histogram of cell sizes investigated for several plates. Two percent of the approximate 5.5 million segmented cells did not pass this quality control. In addition, we have also manually inspected all samples scored as positives. For the IgA channel, we have found a dotty staining pattern in ten cases that produced positive hits based on intensity ratio in negative control cohorts, but does not appear to indicate a specific antibody response. We have also excluded these samples from further analysis.

**Implementation** In order to scale the analysis workflow to the large number of images produced by the assay, we implemented an open-source python library to run the individual analysis steps. This library allows rerunning experiments for a given plate for newly added data on demand and caches intermediate results in order to rerun the analysis from checkpoints in case of errors in one of the processing steps. To this end, we use a file layout based on hdf5 to store multi-resolution image data and tabular data. The processing steps are parallelized over the images of a plate, if possible. We use efficient implementations for the U-Net[236],

[14] https://github.com/napari/napari

StarDist[191] and the watershed algorithm [15] as well as other image processing algorithms[215]. We use pytorch [16] to implement GPU-accelerated cell feature extraction. The total processing time for a plate (containing around 800 images) is about 2 hours and 30 minutes, using a single GPU and 8 CPU cores. In addition, the results of the analysis as well as meta-data associated with individual plates are automatically saved in a centralized MongoDB database [17] at the end of the workflow execution. Apart from keeping track of the analysis outcome and meta-data, a user can save additional information about a given plate/well/image in the database conveniently using the PlateViewer. All source code is available open source under the permissive MIT license at https://github.com/hci-unihd/batchlib.

### A.6.2 Assay Characterization and Validation

**Decision Threshold Selection** In order to determine the presence of SARS-CoV-2 specific antibodies in patient sera, it was necessary to define the decision threshold $\hat{r}$. If a measured intensity ratio $r$ is above the decision threshold $\hat{r}$, then the serum is scored as positive for SARS-CoV-2 antibodies. For this an ROC analysis was performed [245]. Each possible choice of $\hat{r}$ for a test corresponds to a particular sensitivity/specificity pair. By continuously varying the decision threshold, we measured all possible sensitivity/specificity pairs, known as ROC curves, see Figure A.12. To determine the appropriate $\hat{r}$ we considered two factors:

- The undesirability of errors or relative cost of false-positive and false-negative classifications.
- The prevalence, or prior probability of disease.

These factors can be combined to calculate a slope in the ROC plot

$$m = \frac{\text{cost}_{\text{falsepositive}}}{\text{cost}_{\text{falsenegative}}} \frac{1-P}{P}, \tag{A.15}$$

where $P$ is the prevalence or prior probability of the disease. The optimal decision threshold $\hat{r}$, given the false-positive/false-negative cost ratio and prevalence, is the point on the ROC curve where a line with slope $m$ touches the curve. As discussed in Section 6.3.3, a major concern regarding serological assays for SARS-CoV-2 antibody detection is the occurrence of false-positive results. Therefore, we choose $m$ to be larger than one in our analysis. In particular, we determine $\hat{r}$ for the choice of $m = 10$ (see Figure A.12).

[15] http://ukoethe/github.io/vigra
[16] https://pytorch.org/
[17] https://www.mongodb.com

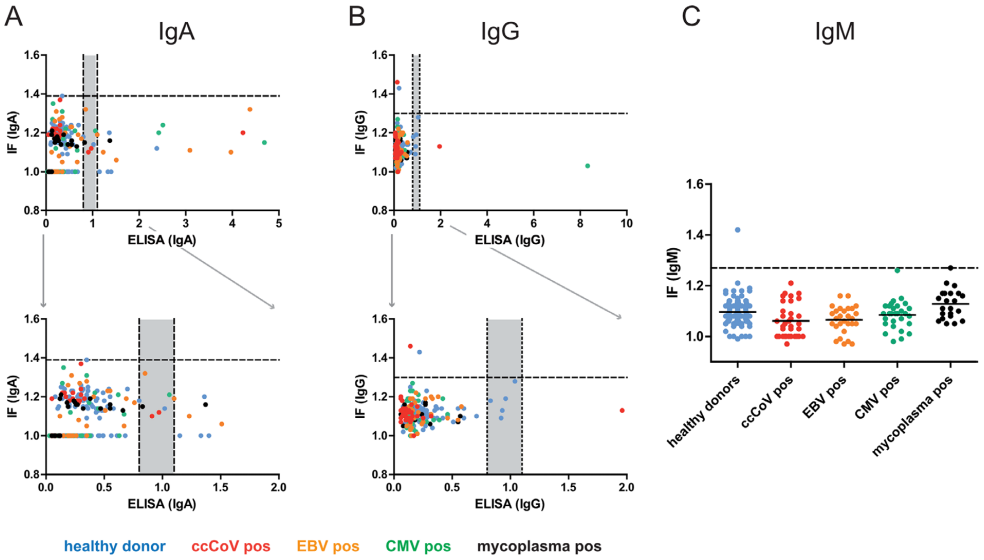



Figure A.10: Correlation between SARS-CoV-2 specific IF and ELISA results for the negative control panel obtained in IgA (A) or IgG (B) measurements. Each dot represents one serum sample. Blue, healthy donors; red, ccCoV positive; green, CMV positive; orange, EBV positive; black, mycoplasma positive. Bottom panels represent zoomed-in versions of the respective top panel to illustrate the borderline region. (C) IgM values for the indicated negative control cohorts determined by IF. Since a corresponding IgM specific ELISA kit from Euroimmun was not available, correlation was not analyzed in this case. In some cases, antibody binding above background was undetectable by IF in non-infected as well as in infected cells, indicating low unspecific cross-reactivity and lack of specific reactivity of the respective serum. In order to allow for inclusion of these data points in the graph, the IF ratio was set to 1.0. Dotted lines indicate the optimal separation cut-off values defined for sample classification, gray areas indicate borderline results in ELISA.

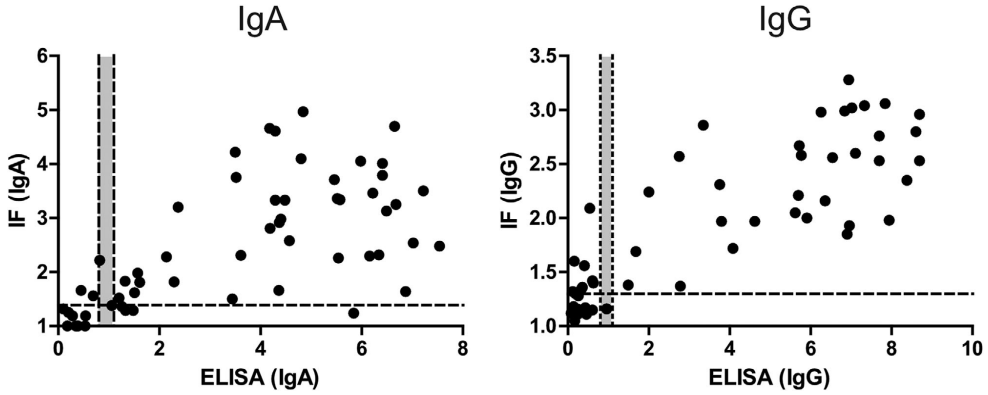



Figure A.11: Correlation between IgA or IgG values obtained by ELISA and IF for sera from 29 COVID-19 patients collected at different days' post infection. In some cases, antibody binding above background was undetectable by IF in non-infected as well as in infected cells, indicating low unspecific cross-reactivity and lack of specific reactivity of the respective serum. In order to allow for inclusion of these data points in the graph, the IF ratio was set to 1.0. Dotted lines indicate the cut-off values defined for classification of readouts, gray areas indicate borderline values.

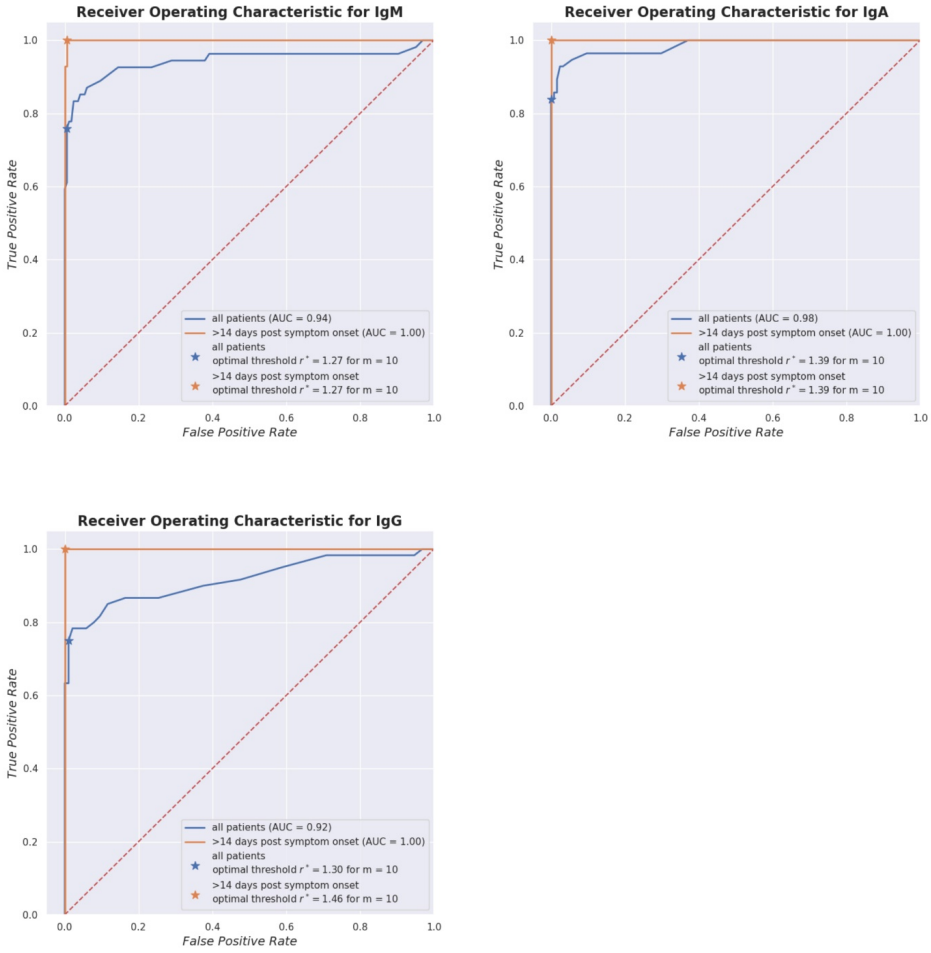



Figure A.12: ROC plot for immunofluorescence (IgM, IgG and IgA) assays. Solid lines show all possible pairs of (false-positive rate, true-positive rate) or, equivalently (1 - specificity, sensitivity) derived from varying the tests' decision threshold. Stars show the optimal threshold for our choice of prevalence and costs that corresponds to a slope of $m = (\text{cost}_{\text{falsepositive}} * (1 - P))/(\text{cost}_{\text{falsenegative}} * P) = 10$. Discriminating only between sera from negative samples and sera from COVID19 patients collected later than 14 days after symptom onset improves the accuracy of the test.

# Publications

I have contributed to the following publications as part of work on this thesis. I am first author or co-first author on the peer-reviewed publications 3, 10, 11, 12 and 18. I am co-author on the peer-reviewed publications 2, 4, 6, 7, 14, 16, 17 and 20.
I am co-first author on the preprints 15, which is currently under review. I am co-author on the preprints 1, 5, 8, 13 and 19, which are currently under review.


1. Alberto Bailoni, Constantin Pape, Steffen Wolf, Thorsten Beier, Anna Kreshuk, and Fred A Hamprecht. "A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation." In: *arXiv preprint arXiv:1906.11713* (2019).

2. Alberto Bailoni, Constantin Pape, Steffen Wolf, Anna Kreshuk, and Fred A Hamprecht. "Proposal-Free Volumetric Instance Segmentation from Latent Single-Instance Masks." In: *German Conference on Pattern Recognition* (2020).

3. Thorsten Beier, Constantin Pape, Nasim Rahaman, and Timo et al. Prange. "Multicut brings automated neurite segmentation closer to human performance." In: *Nature Methods* 14.2 (2017), pp. 101–102.

4. Mirko Cortese, Ji-Young Lee, Berati Cerikan, Christopher J Neufeldt, Viola MJ Oorschot, Sebastian Köhrer, Julian Hennies, Nicole L Schieber, Paolo Ronchi, Giulia Mizzon, et al. "Integrative imaging reveals SARS-CoV-2-induced reshaping of subcellular morphologies." In: *Cell host & microbe* 28.6 (2020), pp. 853–866.

5. Senthilkumar Deivasigamani et al. "Microglia complement signaling promotes neuronal elimination and normal brain functional connectivity." In: *bioRxiv* (2021).

6. Larissa Heinrich, Jan Funke, Constantin Pape, Juan Nunez-Iglesias, and Stephan Saalfeld. "Synaptic cleft segmentation in non-isotropic volume electron microscopy of the complete drosophila brain." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 317–325.

7. Artem Lukoyanov, Isabella Haberbosch, Constantin Pape, Alwin Krämer, Yannick Schwab, and Anna Kreshuk. “Synthetic patches, real images: screening for centrosome aberrations in EM images of human cancer cells.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 523–531.

8. Jacob M Musser, Klaske J Schippers, Michael Nickel, Giulia Mizzon, Andrea B Kohn, Constantin Pape, Jorg U Hammel, Florian Wolf, Cong Liang, Ana Hernandez-Plaza, et al. “Profiling cellular diversity in sponges informs animal cell type and nervous system evolution.” In: *BioRxiv* (2019), p. 758276.

9. Felix Pahmeier, Christopher J Neufeldt, Berati Cerikan, Vibhu Prasad, Costantin Pape, Vibor Laketa, Alessia Ruggieri, Ralf Bartenschlager, and Mirko Cortese. “A Versatile Reporter System To Monitor Virus-Infected Cells and Its Application to Dengue Virus and SARS-CoV-2.” In: *Journal of virology* 95.4 (2021).

10. Constantin Pape, Thorsten Beier, Peter Li, Viren Jain, Davi D Bock, and Anna Kreshuk. “Solving Large Multicut Problems for Connectomics via Domain Decomposition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1–10.

11. Constantin Pape, Alex Matskevych, Adrian Wolny, Julian Hennies, Giulia Mizzon, Marion Louveaux, Jacob Musser, Alexis Maizel, Detlev Arendt, and Anna Kreshuk. “Leveraging domain knowledge to improve microscopy image segmentation with lifted multicuts.” In: *Frontiers in Computer Science* 1 (2019), p. 6.

12. Constantin Pape, Roman Remme, Adrian Wolny, Sylvia Olberg, Steffen Wolf, Lorenzo Cerrone, Mirko Cortese, Severina Klaus, Bojana Lucic, Stephanie Ullrich, et al. “Microscopy-based assay for semi-quantitative detection of SARS-CoV-2 specific antibodies in human sera: A semi-quantitative, high throughput, microscopy-based assay expands existing approaches to measure SARS-CoV-2 specific antibody levels in human sera.” In: *BioEssays* 43.3 (2021), p. 2000257.

13. Akhmedkhan Shabanov, Daja Schichler, Constantin Pape, Sara Cuylen-Haering, and Anna Kreshuk. “Unsupervised temporal consistency improvement for microscopy video segmentation with Siamese networks.” In: *bioRxiv* (2021).

14. Burkhard Tönshoff et al. “Prevalence of SARS-CoV-2 Infection in Children and Their Parents in Southwest Germany.” In: *JAMA Pediatrics* (Jan. 2021).

15. Hernando M Vergara, Constantin Pape, Kimberly Meechan, Valentyna Zinchenko, Christel Genoud, Adrian A Wanner, Benjamin Titze, Rachel Templin, Paola Yanina Bertucci, Oleg Simakov, et al. “Whole-body integration of gene expression and single-cell morphology.” In: *bioRxiv* (2020).

16. Steffen Wolf, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A Hamprecht. “The Mutex Watershed and its Objective: Efficient, Parameter-Free Graph Partitioning.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

17. Steffen Wolf, Yuyan Li, Constantin Pape, Alberto Bailoni, Anna Kreshuk, and Fred A Hamprecht. “The Semantic Mutex Watershed for Efficient Bottom-Up Semantic Instance Segmentation.” In: *Proc. ECCV’2020* (2020).

18. Steffen Wolf, Constantin Pape, Alberto Bailoni, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred Hamprecht. “The Mutex Watershed: Efficient, Parameter-Free Image Partitioning.” In: *Proc. ECCV’18* (2018).

19. Adrian Wolny, Qin Yu, Constantin Pape, and Anna Kreshuk. “Sparse Object-level Supervision for Instance Segmentation with Pixel Embeddings.” In: *arXiv preprint arXiv:2103.14572* (2021).

20. Adrian Wolny et al. “Accurate and versatile 3D segmentation of plant tissues at cellular resolution.” In: *eLife* 9 (July 2020). Ed. by Christian S Hardtke, Dominique C Bergmann, Dominique C Bergmann, and Moritz Graeff, e57613.

# Bibliography


[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. "SLIC superpixels compared to state-of-the-art superpixel methods." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (2012), pp. 2274–2282.

[2] Amir Alush and Jacob Goldberger. "Break and conquer: Efficient correlation clustering for image segmentation." In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2013, pp. 134–147.

[3] Fernando Amat, William Lemon, Daniel P Mossing, Katie McDole, Yinan Wan, Kristin Branson, Eugene W Myers, and Philipp J Keller. "Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data." In: *Nature Methods* 11 (July 2014), p. 951.

[4] James R. Anderson et al. "Exploring the retinal connectome." In: *Mol. Vis.* 17 (Feb. 2011), pp. 355–379.

[5] Bjoern Andres, Jorg H Kappes, Thorsten Beier, Ullrich Kothe, and Fred A Hamprecht. "Probabilistic Image Segmentation with Closedness Constraints." In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2611–2618.

[6] Bjoern Andres, Thorben Kroeger, Kevin L Briggman, Winfried Denk, Graham Knott, Ullrich Koethe, and Fred A Hamprecht. "Globally Optimal Closed-surface Segmentation for Connectomics." In: *Proc. Europ. Conf. Comp. Vision* (2012), pp. 1–14.

[7] Björn Andres, Ullrich Köthe, Moritz Helmstaedter, Winfried Denk, and Fred A Hamprecht. "Segmentation of SBFSEM volume data of neural tissue by hierarchical classification." In: *Pattern Recogn.* (2008), pp. 142–152.

[8] Jesús Angulo and Dominique Jeulin. "Stochastic watershed segmentation." In: *Proc. ISMM Rio de Janeiro* 1 (2007), pp. 265–276.

[9] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour Detection and Hierarchical Image Segmentation." In: *IEEE Trans. Patt. Anal. Mach. Intell.* 33.5 (2011), pp. 898–916.

[10] Pablo Arbelaez. "Boundary Extraction in Natural Images Using Ultrametric Contour Maps." In: *Proc. CVPRW* (2006), pp. 182–182.

[11] Detlev Arendt, Jacob M Musser, Clare VH Baker, Aviv Bergman, Connie Cepko, Douglas H Erwin, Mihaela Pavlicev, Gerhard Schlosser, Stefanie Widder, Manfred D Laubichler, et al. "The origin and evolution of cell types." In: *Nature Reviews Genetics* 17.12 (2016), pp. 744–757.

[12] Ignacio Arganda-Carreras, Sebastian Seung, Ashwin Vishwanathan, and Daniel R Berger. *SNEMI3D: 3D Segmentation of neurites in EM images*. URL: http://brainiac2.mit.edu/SNEMI3D/ (visited on 04/04/2016).

[13] Ignacio Arganda-Carreras et al. "Crowdsourcing the creation of image segmentation algorithms for connectomics." In: *Front. Neuroanat.* 9.November (2015), pp. 1–13.

[14] Shai Bagon and Meirav Galun. "Optimizing Large Scale Correlation Clustering." In: *arXiv preprint arXiv:1112.2903* 3 (2011).

[15] Min Bai and Raquel Urtasun. "Deep Watershed Transform for Instance Segmentation." In: *arXiv:1611.08303* (2016).

[16] Alberto Bailoni, Constantin Pape, Steffen Wolf, Thorsten Beier, Anna Kreshuk, and Fred A Hamprecht. "A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation." In: *arXiv preprint arXiv:1906.11713* (2019).

[17] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation clustering." In: *Mach. Learn.* 56.1-3 (2004), pp. 89–113.

[18] Thorsten Beier, Björn Andres, Ullrich Köthe, and Fred A Hamprecht. "An efficient fusion move algorithm for the minimum cost lifted multicut problem." In: *European Conference on Computer Vision*. Springer. 2016, pp. 715–730.

[19] Thorsten Beier, Fred A. Hamprecht, and Jörg H. Kappes. "Fusion Moves for Correlation Clustering." In: *Proc. CVPR* (2015). 1, pp. 3507–3516.

[20] Thorsten Beier, Thorben Kroeger, Jorg H Kappes, Ullrich Kothe, and Fred A Hamprecht. "Cut, glue & cut: A fast, approximate solver for multicut partitioning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 73–80.

[21] Thorsten Beier, Constantin Pape, Nasim Rahaman, and Timo et al. Prange. "Multicut brings automated neurite segmentation closer to human performance." In: *Nature Methods* 14.2 (2017), pp. 101–102.

[22] Stuart Berg, Dominik Kutra, Thorben Kroeger, Christoph N Straehle, Bernhard X Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, et al. "Ilastik: interactive machine learning for (bio) image analysis." In: *Nature Methods* (2019), pp. 1–7.

[23] Manuel Berning, Kevin M. Boergens, and Moritz Helmstaedter. “SegEM: Efficient Image Analysis for High-Resolution Connectomics.” In: *Neuron* 87.6 (2015), pp. 1193–1206.

[24] Serge Beucher. “Watershed, Hierarchical Segmentation and Waterfall Algorithm.” In: *Proc. ISMM’94*. Vol. 94. 1994, pp. 69–76.

[25] Serge Beucher and Fernand Meyer. “The morphological approach to segmentation: the watershed transformation.” In: *Optical Engineering-New York-Marcel Dekker Incorporated* 34 (1992), pp. 433–433.

[26] Vincent D Blondel, Jean-Lou Guillaume, Renaud Lambiotte, and Étienne Lefebvre. “The Louvain method for community detection in large networks.” In: *J of Statistical Mechanics: Theory and Experiment* 10 (2011), P10008.

[27] Davi D. Bock, Wei-Chung Allen Lee, Aaron M. Kerlin, Mark L. Andermann, Greg Hood, Arthur W. Wetzel, Sergey Yurgenson, Edward R. Soucy, Hyon Suk Kim, and R. Clay Reid. “Network anatomy and in vivo physiology of visual cortical neurons.” In: *Nature* 471.7337 (Mar. 2011), pp. 177–182.

[28] Kevin M Boergens, Manuel Berning, Tom Bocklisch, Dominic Bräunlein, Florian Drawitsch, Johannes Frohnhofen, Tom Herold, Philipp Otto, Norman Rzepka, Thomas Werkmeister, et al. “webKnossos: efficient online 3D data annotation for connectomics.” In: *Nature Methods* 14.7 (2017), pp. 691–694.

[29] Johm Bogovic, Igor Pisarev, Philipp Hanslovsky, Neil Thistlethwaite, and Stephan Saalfeld. 2020. URL: https://github.com/saalfeldlab/n5.

[30] Kevin L Briggman and Davi D Bock. “Volume electron microscopy for neuronal circuit reconstruction.” In: *Current opinion in neurobiology* 22.1 (2012), pp. 154–161.

[31] Randal Burns, Joshua?T Vogelstein, and Alexander?S Szalay. “From Cosmos to Connectomes: The Evolution of Data-Intensive Science.” In: *Neuron* 83.6 (Sept. 2014), pp. 1249–1252.

[32] Jinzheng Cai, Le Lu, Zizhao Zhang, Fuyong Xing, Lin Yang, and Qian Yin. “Pancreas Segmentation in MRI Using Graph-Based Decision Fusion on Convolutional Neural Networks.” In: *Proc. MICCAI*. 2016.

[33] Juan C Caicedo, Allen Goodman, Kyle W Karhohs, Beth A Cimini, Jeanelle Ackerman, Marzieh Haghighi, CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, et al. “Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl.” In: *Nature methods* 16.12 (2019), pp. 1247–1253.

[34] 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." In: *PLoS Biol.* 8.10 (Oct. 2010), e1000502.

[35] Ozgun Cciccek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. "3D U-Net: learning dense volumetric segmentation from sparse annotation." In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[36] Paul KS Chan, King-Cheung Ng, Rickjason CW Chan, Rebecca KY Lam, Viola CY Chow, Mamie Hui, Alan Wu, Nelson Lee, Florence HY Yap, Frankie WT Cheng, et al. "Immunofluorescence assay for serologic diagnosis of SARS." In: *Emerging infectious diseases* 10.3 (2004), p. 530.

[37] Thomas F Chartier, Joran Deschamps, Wiebke Dürichen, Gáspár Jékely, and Detlev Arendt. "Whole-head recording of chemosensory activity in the marine annelid Platynereis dumerilii." In: *Royal Society Open Biology* 8.10 (2018), p. 180139.

[38] Hao Chen, Xiao Qi, Jie Cheng, and Pheng Heng. "Deep contextual networks for neuronal structure segmentation." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[39] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[40] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[41] Sunil Chopra and Mendu R Rao. "The partition problem." In: *Mathematical Programming* 59.1-3 (1993), pp. 87–115.

[42] Dan C Ciresan, Alessandro Giusti, Luca M Gambardella, and Jurgen Schmidhuber. "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images." In: *Proc. NIPS* (2012).

[43] Steven J Cook, Charles M Crouse, Eviatar Yemini, David H Hall, Scott W Emmons, and Oliver Hobert. "The connectome of the Caenorhabditis elegans pharynx." In: *BioRxiv* (2019), p. 868513.

[44] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[45] Victor M Corman, Olfert Landt, Marco Kaiser, Richard Molenkamp, Adam Meijer, Daniel KW Chu, Tobias Bleicker, Sebastian Brünink, Julia Schneider, Marie Luisa Schmidt, et al. "Detection of 2019 novel coronavirus (2019-nCoV) by real-time RT-PCR." In: *Eurosurveillance* 25.3 (2020), p. 2000045.

[46] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[47] Mirko Cortese, Ji-Young Lee, Berati Cerikan, Christopher J Neufeldt, Viola MJ Oorschot, Sebastian Köhrer, Julian Hennies, Nicole L Schieber, Paolo Ronchi, Giulia Mizzon, et al. "Integrative imaging reveals SARS-CoV-2-induced reshaping of subcellular morphologies." In: *Cell host & microbe* 28.6 (2020), pp. 853–866.

[48] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. "Watershed cuts: Minimum spanning forests and the drop of water principle." In: *IEEE transactions on pattern analysis and machine intelligence* 31.8 (2008), pp. 1362–1374.

[49] Bert De Brabandere, Davy Neven, and Luc Van Gool. "Semantic instance segmentation with a discriminative loss function." In: *arXiv preprint arXiv:1708.02551* (2017).

[50] Senthilkumar Deivasigamani et al. "Microglia complement signaling promotes neuronal elimination and normal brain functional connectivity." In: *bioRxiv* (2021).

[51] Winfried Denk and Heinz Horstmann. "Serial Block-Face Scanning Electron Microscopy to Reconstruct Three-Dimensional Tissue Nanostructure." In: *PLOS Biology* 2.11 (Oct. 2004).

[52] Lee R Dice. "Measures of the amount of ecologic association between species." In: *Ecology* 26.3 (1945), pp. 297–302.

[53] Konstantin Dmitriev, Toufiq Parag, Brian Matejek, Arie E Kaufman12, and Hanspeter Pfister. "Efficient correction for em connectomics with skeletal representation." In: *British Machine Vision Conferemce (BMVC)* (2018).

[54] J Dubochet and AW McDowall. "Vitrification of pure water for electron microscopy." In: *Journal of Microscopy* 124.3 (1981), pp. 3–4.

[55] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[56] AL Eberle, S Mikula, R Schalek, J Lichtman, ML Knothe Tate, and D Zeidler. "High-resolution, high-throughput imaging with a multibeam scanning electron microscope." In: *Journal of microscopy* 259.2 (2015), pp. 114–120.

[57] Katharina Eichler, Feng Li, Ashok Litwin-Kumar, Youngser Park, Ingrid Andrade, Casey M Schneider-Mizell, Timo Saumweber, Annina Huser, Claire Eschbach, Bertram Gerber, et al. "The complete connectome of a learning and memory centre in an insect brain." In: *Nature* 548.7666 (2017), pp. 175–182.

[58] Eva Engvall and Peter Perlmann. "Enzyme-Linked Immunosorbent Assay, Elisa." In: *The Journal of Immunology* 109.1 (1972), pp. 129–135.

[59] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[60] Alexandre X Falcão, Jorge Stolfi, and Roberto de Alencar Lotufo. "The image foresting transform: Theory, algorithms, and applications." In: *IEEE Trans. Patt. Anal. Mach. Intell.* 26.1 (2004), pp. 19–29.

[61] Jingtao Fan, Jinli Suo, Jiamin Wu, Hao Xie, Yibing Shen, Feng Chen, Guijin Wang, Liangcai Cao, Guofan Jin, Quansheng He, et al. "Video-rate imaging of biological dynamics at centimetre scale and micrometre resolution." In: *Nature Photonics* 13.11 (2019), pp. 809–816.

[62] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Efficient Graph-Based Image Segmentation." In: *Int. J. Comput. Vision* 59.2 (2004), pp. 167–181.

[63] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. "An overview of the HDF5 technology suite and its applications." In: *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*. 2011, pp. 36–47.

[64] Jan Funke, Bjorn Andres, Fred A. Hamprecht, Albert Cardona, and Matthew Cook. "Efficient automatic 3D-reconstruction of branching neurons from EM data." In: *Proc. CVPR* (June 2012), pp. 1004–1011.

[65] Jan Funke, Lisa Mais, Andrew Champion, Natalie Dye, and Dagmar Kainmueller. "A Benchmark for Epithelial Cell Tracking." In: *The European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

[66] Jan Funke, Stephan Saalfeld, Davi Bock, Srini Turaga, and Eric Perlman. *Circuit Reconstruction from Electron Microscopy Images*. URL: https://cremi.org/ (visited on 08/27/2020).

[67] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. “Large scale image segmentation with structured loss based deep learning for connectome reconstruction.” In: *IEEE transactions on pattern analysis and machine intelligence* 41.7 (2018), pp. 1669–1680.

[68] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[69] HA Goossens, MK Nohlmans, and AE Van Den Bogaard. “Epstein-Barr virus and cytomegalovirus infections cause false-positive results in IgM two-test protocol for early Lyme borreliosis.” In: *Infection* 27.3 (1999), pp. 231–231.

[70] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. “Self-supervised pretraining of visual features in the wild.” In: *arXiv preprint arXiv:2103.01988* (2021).

[71] M. Grimaud. “New measure of contrast: the dynamics.” In: *Proc. Image Algebra and Morphological Processing*. Ed. by P. D. Gader, E. R. Dougherty, & J. C. Serra. Vol. 1769. SPIE Conf. Series. 1992, pp. 292–305.

[72] Philipp Hanslovsky et al. *saalfeldlab/paintera: Paintera 0.27.0*. Version paintera-0.27.0. Mar. 2021. URL: https://doi.org/10.5281/zenodo.4652869.

[73] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. “Array programming with NumPy.” In: *Nature* 585.7825 (2020), pp. 357–362.

[74] Christian Hasse, Nicole Rebscher, Wencke Reiher, Kathrin Sobjinski, Erhard Moerschel, Lothar Beck, Kristin Tessmar-Raible, Detlev Arendt, and Monika Hassel. “Three consecutive generations of nephridia occur during development of Platynereis dumerilii (Annelida, Polychaeta).” In: *Developmental dynamics* 239.7 (2010), pp. 1967–1976.

[75] Kenneth J Hayworth, David Peale, Zhiyuan Lu, C Shan Xu, and Harald F Hess. “Serial Thick Section Gas Cluster Ion Beam Scanning Electron Microscopy.” In: *Microscopy and Microanalysis* 24.S1 (2018), pp. 1444–1445.

[76] Kenneth J Hayworth, C Shan Xu, Zhiyuan Lu, Graham W Knott, Richard D Fetter, Juan Carlos Tapia, Jeff W Lichtman, and Harald F Hess. “Ultrastructurally smooth thick partitioning and volume stitching for large-scale connectomics.” In: *Nature methods* 12.4 (2015), p. 319.

[77] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. “Momentum contrast for unsupervised visual representation learning.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.

[78] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask r-cnn.” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[80] Larissa Heinrich, Jan Funke, Constantin Pape, Juan Nunez-Iglesias, and Stephan Saalfeld. “Synaptic cleft segmentation in non-isotropic volume electron microscopy of the complete drosophila brain.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 317–325.

[81] Fritjof Helmchen and Winfried Denk. “Deep tissue two-photon microscopy.” In: *Nature methods* 2.12 (2005), pp. 932–940.

[82] Moritz Helmstaedter. “Cellular-resolution connectomics: challenges of dense neural circuit reconstruction.” In: *Nat. Methods* 10.6 (May 2013), pp. 501–507.

[83] Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, H Sebastian Seung, and Winfried Denk. “Connectomic reconstruction of the inner plexiform layer in the mouse retina.” In: *Nature* 500.7461 (2013), pp. 168–174.

[84] Olivier J Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron van den Oord, Oriol Vinyals, and João Carreira. “Efficient Visual Pretraining with Contrastive Detection.” In: *arXiv preprint arXiv:2103.10957* (2021).

[85] David Grant Colburn Hildebrand, Marcelo Cicconet, Russel Miguel Torres, Woohyuk Choi, Tran Minh Quan, Jungmin Moon, Arthur Willis Wetzel, Andrew Scott Champion, Brett Jesse Graham, Owen Randlett, et al. “Whole-brain serial-section electron microscopy in larval zebrafish.” In: *Nature* 545.7654 (2017), pp. 345–349.

[86] Reka Hollandi, Abel Szkalisity, Timea Toth, Ervin Tasnadi, Csaba Molnar, Botond Mathe, Istvan Grexa, Jozsef Molnar, Arpad Balind, Mate Gorbe, et al. “nucleAIzer: A parameter-free deep learning framework for nucleus segmentation using image style transfer.” In: *Cell Systems* (2020).

[87] Andrea Horňáková, Jan-Hendrik Lange, and Bjoern Andres. “Analysis and optimization of graph decompositions by lifted multicuts.” In: *International Conference on Machine Learning*. 2017, pp. 1539–1548.

[88] Jan Huisken, Jim Swoger, Filippo Del Bene, Joachim Wittbrodt, and Ernst HK Stelzer. "Optical sectioning deep inside live embryos by selective plane illumination microscopy." In: *Science* 305.5686 (2004), pp. 1007–1009.

[89] Coronaviridae Study Group of the International et al. "The species Severe acute respiratory syndrome-related coronavirus: classifying 2019-nCoV and naming it SARS-CoV-2." In: *Nature microbiology* 5.4 (2020), p. 536.

[90] Viren Jain, Joseph F Murray, Fabian Roth, Srinivas Turaga, Valentin Zhigulin, Kevin L Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. "Supervised learning of image restoration with convolutional networks." In: *Proc. Intl. Conf. Comp. Vision* (2007), pp. 1–8.

[91] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. "High-precision automated reconstruction of neurons with flood-filling networks." In: *Nature methods* 15.8 (2018), p. 605.

[92] Cory Jones, Ting Liu, Nathaniel Wood Cohan, Mark Ellisman, and Tolga Tasdizen. "Efficient semi-automatic 3D segmentation for neuron tracing in electron microscopy images." In: *J. Neurosci. Methods* 246 (May 2015), pp. 13–21.

[93] Elizabeth Jurrus, Antonio RC Paiva, Shigeki Watanabe, James R Anderson, Bryan W Jones, Ross T Whitaker, Erik M Jorgensen, Robert E Marc, and Tolga Tasdizen. "Detection of neuron membranes in electron microscopy images using a serial neural network architecture." In: *Med. Image Anal.* 14.6 (2010), pp. 770–783.

[94] Jörg Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schn. "Globally optimal image partitioning by multicuts." In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer. 2011, pp. 31–44.

[95] Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. "Higher-order segmentation via multicuts." In: *Computer Vision and Image Understanding* 143 (2016), pp. 104–119.

[96] Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. "Saturated reconstruction of a volume of neocortex." In: *Cell* 162.3 (2015), pp. 648–661.

[97] Verena Kaynig, Thomas Fuchs, and Joachim M Buhmann. "Neuron geometry extraction by perceptual grouping in sstem images." In: *Proc. CVPR* (2010), pp. 2902–2909.

[98] Verena Kaynig, Amelio Vazquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R. Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister. "Large-scale automatic reconstruction of neuronal processes from electron microscopy images." In: *Med. Image Anal.* 22.1 (2015), pp. 77–88.

[99] Philipp J. Keller, Annette D. Schmidt, Joachim Wittbrodt, and Ernst H.K. Stelzer. "Reconstruction of Zebrafish Early Embryonic Development by Scanned Light Sheet Microscopy." In: *Science* 322.5904 (2008). bibtex*[publisher=American Association for the Advancement of Science;eprint=https://science.sciencemag.org/content/322/5904/1065.full.pdf], pp. 1065–1069.

[100] Brian W Kernighan and Shen Lin. "An efficient heuristic procedure for partitioning graphs." In: *The Bell system technical journal* 49.2 (1970), pp. 291–307.

[101] Margret Keuper. "Higher-order minimum cost lifted multicuts for motion segmentation." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4242–4250.

[102] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. "Efficient decomposition of image and mesh graphs by lifted multicuts." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1751–1759.

[103] Jinseop S Kim et al. "Space-time wiring specificity supports direction selectivity in the retina." In: *Nature* 509.7500 (May 2014), pp. 331–336.

[104] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *Proc. ICLR* (2014).

[105] B. Ravi Kiran and Jean Serra. "Global–local optimizations by hierarchical cuts and climbing energies." In: *Pattern Recognition* 47.1 (2014), pp. 12–24.

[106] Stefan Klein, Marius Staring, Keelin Murphy, Max A Viergever, and Josien PW Pluim. "Elastix: a toolbox for intensity-based medical image registration." In: *IEEE transactions on medical imaging* 29.1 (2009), pp. 196–205.

[107] Seymour Knowles-Barley, Verena Kaynig, Thouis Ray Jones, Alyssa Wilson, Joshua Morgan, Dongil Lee, Daniel Berger, Narayanan Kasthuri, Jeff W Lichtman, and Hanspeter Pfister. "RhoanaNet Pipeline: Dense Automatic Neural Annotation." In: *arXiv:1611.06973* (2016).

[108] Iasonas Kokkinos. "Pushing the boundaries of boundary detection using deep learning." In: *arXiv:1511.07386* (2015).

[109] Ullrich Köthe. *Generische Programmierung für die Bildverarbeitung*. BoD–Books on Demand, 2000.

[110] NE Krasowski, Thorsten Beier, GW Knott, Ullrich Köthe, Fred A Hamprecht, and Anna Kreshuk. "Neuron segmentation with high-level biological priors." In: *IEEE transactions on medical imaging* 37.4 (2017), pp. 829–839.

[111] Anna Kreshuk, Robert Walecki, Ullrich Koethe, Mortimer Gierthmuehlen, Dennis Plachta, Christel Genoud, Kristin Haastert-Talini, and Fred A. Hamprecht. "Automated Tracing of Myelinated Axons and Detection of the Nodes of Ranvier in Serial Images of Peripheral Nerves." In: *J. Microsc.* 259 (2) (2015), pp. 143–154.

[112] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[113] Thorben Kroeger, Jörg H Kappes, Thorsten Beier, Ullrich Koethe, and Fred A Hamprecht. "Asymmetric cuts: Joint image labeling and partitioning." In: *German Conference on Pattern Recognition*. Springer. 2014, pp. 199–211.

[114] Thorben Kroeger, Shawn Mikula, Winfried Denk, Ullrich Koethe, and Fred A Hamprecht. "Learning to segment neurons with non-local quality measures." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2013, pp. 419–427.

[115] Uros Krzic, Stefan Gunther, Timothy E Saunders, Sebastian J Streichan, and Lars Hufnagel. "Multiview light-sheet microscope for rapid in toto imaging." In: *Nature Methods* 9 (June 2012), p. 730.

[116] Barry Ladizinski and Christopher Sankey. "Acute Epstein-Barr virus infection and human immunodeficiency virus antibody cross-reactivity." In: *The American journal of medicine* 127.6 (2014), e9–e10.

[117] Jan-Hendrik Lange, Andreas Karrenbauer, and Bjoern Andres. "Partial optimality and fast lower bounds for weighted correlation clustering." In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2892–2901.

[118] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. "Building skeleton models via 3-D medial surface axis thinning algorithms." In: *CVGIP: Graphical Models and Image Processing* 56.6 (1994), pp. 462–478.

[119] Kisuk Lee, Ran Lu, Kyle Luther, and H Sebastian Seung. "Learning Dense Voxel Embeddings for 3D Neuron Reconstruction." In: *arXiv preprint arXiv:1909.09872* (2019).

[120] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. "Superhuman accuracy on the SNEMI3D connectomics challenge." In: *arXiv preprint arXiv:1706.00120* (2017).

[121] Wei-Chung Allen Lee, Vincent Bonin, Michael Reed, Brett J. Graham, Greg Hood, Katie Glattfelder, and R. Clay Reid. "Anatomy and function of an excitatory network in the visual cortex." In: *Nature* 532.7599 (Apr. 2016), pp. 370–374.

[122] Evgeny Levinkov, Alexander Kirillov, and Bjoern Andres. "A comparative study of local search algorithms for correlation clustering." In: *German Conference on Pattern Recognition*. Springer. 2017, pp. 103–114.

[123] Evgeny Levinkov, Jonas Uhrig, Siyu Tang, Mohamed Omran, Eldar Insafutdinov, Alexander Kirillov, Carsten Rother, Thomas Brox, Bernt Schiele, and Bjoern Andres. "Joint graph decomposition & node labeling: Problem, algorithms, applications." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6012–6020.

[124] Peter H Li, Larry F Lindsey, Michał Januszewski, Mike Tyka, Jeremy Maitin-Shepard, Tim Blakely, and Viren Jain. "Automated reconstruction of a serial-section EM Drosophila brain with flood-filling networks and local realignment." In: *Microscopy and Microanalysis* 25.S2 (2019), pp. 1364–1365.

[125] Jeff W. Lichtman, Hanspeter Pfister, and Nir Shavit. "The big data challenges of connectomics." In: *Nat. Neurosci.* 17.11 (Nov. 2014). Perspective, pp. 1448–1454.

[126] Benjamin J Liebeskind, Hans A Hofmann, David M Hillis, and Harold H Zakon. "Evolution of animal neural systems." In: *Annual review of ecology, evolution, and systematics* 48 (2017), pp. 377–398.

[127] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[128] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context." In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[129] Ting Liu, Cory Jones, Mojtaba Seyedhosseini, and Tolga Tasdizen. "A modular hierarchical approach to 3D electron microscopy image segmentation." In: *J. Neurosci. Methods* 226 (Apr. 2014), pp. 88–102.

[130] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[131] Quan-Xin Long, Bai-Zhong Liu, Hai-Jun Deng, Gui-Cheng Wu, Kun Deng, Yao-Kai Chen, Pu Liao, Jing-Fu Qiu, Yong Lin, Xue-Fei Cai, et al. "Antibody responses to SARS-CoV-2 in patients with COVID-19." In: *Nature medicine* 26.6 (2020), pp. 845–848.

[132] Filip Lux and Petr Matula. "Cell Segmentation by Combining Marker-Controlled Watershed and Deep Learning." In: *arXiv preprint arXiv:2004.01607* (2020).

[133] Matthew Madany, Kyle Marcus, Steven Peltier, Mark H Ellisman, and Ilkay Altintas. "NeuroKube: An Automated and Autoscaling Neuroimaging Reconstruction Framework using Cloud Native Computing and AI." In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE Computer Society. 2020, pp. 320–330.

[134] L Mais, P Hirsch, and D Kainmueller. "PatchPerPix for instance segmentation." In: *Lecture Notes in Computer Science* 12370 (2020), pp. 288–304.

[135] Filip Malmberg, Robin Strand, and Ingela Nyström. "Generalized hard constraints for graph segmentation." In: *Scandinavian Conference on Image Analysis*. Springer. 2011, pp. 36–47.

[136] Fredrik Manne and Md Mostofa Ali Patwary. "A scalable parallel union-find algorithm for distributed memory computers." In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2009, pp. 186–195.

[137] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics." In: *Proc. 8th Int'l Conf. Computer Vision*. Vol. 2. July 2001, pp. 416–423.

[138] Brian Matejek, Daniel Haehn, Haidong Zhu, Donglai Wei, Toufiq Parag, and Hanspeter Pfister. "Biologically-Constrained Graphs for Global Connectomics Reconstruction." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2089–2098.

[139] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction." In: *arXiv preprint arXiv:1802.03426* (2018).

[140] Marina Meilua. "Comparing clusterings by the variation of information." In: *Learning theory and kernel machines*. Springer, 2003, pp. 173–187.

[141] Yaron Meirovitch, Alexander Matveev, Hayk Saribekyan, David Budden, David Rolnick, Gergely Odor, Seymour Knowles-Barley Thouis Raymond Jones, Hanspeter Pfister, Jeff William Lichtman, and Nir Shavit. "A Multi-Pass Approach to Large-Scale Connectomics." In: *arXiv preprint arXiv:1612.02120* (2016).

[142] Yaron Meirovitch, Lu Mi, Hayk Saribekyan, Alexander Matveev, David Rolnick, and Nir Shavit. "Cross-Classification Clustering: An Efficient Multi-Object Tracking Technique for 3-D Instance Segmentation in Connectomics." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8425–8435.

[143] Benjamin Meyer, Christian Drosten, and Marcel A Müller. "Serological assays for emerging coronaviruses: challenges and pitfalls." In: *Virus research* 194 (2014), pp. 175–183.

[144] Fernand Meyer. "Minimum spanning forests for morphological segmentation." In: *Mathematical morphology and its applications to image processing*. 1994, pp. 77–84.

[145] Fernand Meyer. "Topographic distance and watershed lines." In: *Signal processing* 38.1 (1994), pp. 113–125.

[146] Fernand Meyer. "Morphological multiscale and interactive segmentation." In: *WS on Nonlinear Signal and Image Processing*. 1999, pp. 369–377.

[147] Alistair Miles et al. *zarr-developers/zarr-python: v2.4.0*. Version v2.4.0. Jan. 2020.

[148] Josh Moore et al. "OME-NGFF: scalable format strategies for interoperable bioimaging data." In: *bioRxiv* (2021).

[149] Josh Lyskowski Morgan, Daniel Raimund Berger, Arthur Willis Wetzel, and Jeff William Lichtman. "The fuzzy logic of network connectivity in mouse visual thalamus." In: *Cell* 165.1 (2016), pp. 192–206.

[150] Alessandro Motta, Manuel Berning, Kevin M Boergens, Benedikt Staffler, Marcel Beining, Sahil Loomba, Philipp Hennig, Heiko Wissler, and Moritz Helmstaedter. "Dense connectomic reconstruction in layer 4 of the somatosensory cortex." In: *Science* 366.6469 (2019).

[151] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. "The Role of Context for Object Detection and Semantic Segmentation in the Wild." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[152] Andreas Müller, Deborah Schmidt, C Shan Xu, Song Pang, Joyson Verner D'Costa, Susanne Kretschmar, Carla Münster, Thomas Kurth, Florian Jug, Martin Weigert, et al. "3D FIB-SEM reconstruction of microtubule–organelle interaction in whole primary mouse $\beta$ cells." In: *Journal of Cell Biology* 220.2 (2021).

[153] Jacob M Musser, Klaske J Schippers, Michael Nickel, Giulia Mizzon, Andrea B Kohn, Constantin Pape, Jorg U Hammel, Florian Wolf, Cong Liang, Ana Hernandez-Plaza, et al. "Profiling cellular diversity in sponges informs animal cell type and nervous system evolution." In: *BioRxiv* (2019), p. 758276.

[154] Laurent Najman. “On the equivalence between hierarchical segmentations and ultrametric watersheds.” In: *J. of Mathematical Imaging and Vision* 40.3 (2011), pp. 231–247.

[155] Laurent Najman and Michel Schmitt. “Geodesic saliency of watershed contours and hierarchical segmentation.” In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 18.12 (1996), pp. 1163–1173.

[156] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. “Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8837–8845.

[157] Adrian Pascal Nievergelt, Gonzalo Alvarez Viar, and Gaia Pigino. “Towards a mechanistic understanding of cellular processes by cryoEM.” In: *Current opinion in structural biology* 58 (2019), pp. 149–158.

[158] Jonathon Nixon-Abell, Christopher J Obara, Aubrey V Weigel, Dong Li, Wesley R Legant, C Shan Xu, H Amalia Pasolli, Kirsten Harvey, Harald F Hess, Eric Betzig, et al. “Increased spatiotemporal resolution reveals highly dynamic dense tubular matrices in the peripheral ER.” In: *Science* 354.6311 (2016), aaf3928.

[159] Sebastian Nowozin and Stefanie Jegelka. “Solution stability in linear programming relaxations: Graph partitioning and unsupervised learning.” In: *Proc. ICML* (2009), pp. 769–776.

[160] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. “Machine learning of hierarchical clustering to segment 2D and 3D images.” In: *PloS ONE* 8.8 (2013), e71715.

[161] Juan Nunez-Iglesias, Ryan Kennedy, Stephen M Plaza, Anirban Chakraborty, and William T Katz. “Graph-based active learning of agglomeration (GALA): a Python library to segment 2D and 3D neuroimages.” In: *Frontiers in neuroinformatics* 8 (2014), p. 34.

[162] Natacha S Ogando, Tim J Dalebout, Jessika C Zevenhoven-Dobbe, Ronald WAL Limpens, Yvonne van der Meer, Leon Caly, Julian Druce, Jutte JC de Vries, Marjolein Kikkert, Montserrat Bárcena, et al. “SARS-coronavirus-2 replication in Vero E6 cells: replication kinetics, rapid adaptation and cytopathology.” In: *The Journal of general virology* 101.9 (2020), p. 925.

[163] Nisreen MA Okba, Marcel A Müller, Wentao Li, Chunyan Wang, Corine H GeurtsvanKessel, Victor M Corman, Mart M Lamers, Reina S Sikkema, Erwin de Bruin, Felicity D Chandler, et al. "Severe acute respiratory syndrome coronavirus 2-specific antibody responses in coronavirus disease patients." In: *Emerging infectious diseases* 26.7 (2020), pp. 1478–1488.

[164] Shotaro Otsuka, Anna M Steyer, Martin Schorb, Jean-Karim Heriche, M Julius Hossain, Suruchi Sethi, Moritz Kueblbeck, Yannick Schwab, Martin Beck, and Jan Ellenberg. "Postmitotic nuclear pore assembly proceeds by radial dilation of small membrane openings." In: *Nature structural & molecular biology* 25.1 (2018), p. 21.

[165] Felix Pahmeier, Christopher J Neufeldt, Berati Cerikan, Vibhu Prasad, Costantin Pape, Vibor Laketa, Alessia Ruggieri, Ralf Bartenschlager, and Mirko Cortese. "A Versatile Reporter System To Monitor Virus-Infected Cells and Its Application to Dengue Virus and SARS-CoV-2." In: *Journal of virology* 95.4 (2021).

[166] Constantin Pape, Thorsten Beier, Peter Li, Viren Jain, Davi D Bock, and Anna Kreshuk. "Solving Large Multicut Problems for Connectomics via Domain Decomposition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1–10.

[167] Constantin Pape, Alex Matskevych, Adrian Wolny, Julian Hennies, Giulia Mizzon, Marion Louveaux, Jacob Musser, Alexis Maizel, Detlev Arendt, and Anna Kreshuk. "Leveraging domain knowledge to improve microscopy image segmentation with lifted multicuts." In: *Frontiers in Computer Science* 1 (2019), p. 6.

[168] Constantin Pape, Roman Remme, Adrian Wolny, Sylvia Olberg, Steffen Wolf, Lorenzo Cerrone, Mirko Cortese, Severina Klaus, Bojana Lucic, Stephanie Ullrich, et al. "Microscopy-based assay for semi-quantitative detection of SARS-CoV-2 specific antibodies in human sera: A semi-quantitative, high throughput, microscopy-based assay expands existing approaches to measure SARS-CoV-2 specific antibody levels in human sera." In: *BioEssays* 43.3 (2021), p. 2000257.

[169] Toufiq Parag, Fabian Tschopp, William Grisaitis, Srinivas C Turaga, Xuewen Zhang, Brian Matejek, Lee Kamentsky, Jeff W Lichtman, and Hanspeter Pfister. "Anisotropic EM segmentation by 3D affinity learning and agglomeration." In: *arXiv preprint arXiv:1707.08935* (2017).

[170] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library." In: *arXiv preprint arXiv:1912.01703* (2019).

[171] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python." In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[172] André F Pereira, Daniel J Hageman, Tomasz Garbowski, Christof Riedesel, Ulf Knothe, Dirk Zeidler, and Melissa L Knothe Tate. "Creating high-resolution multiscale maps of human tissue using multi-beam SEM." In: *PLoS computational biology* 12.11 (2016), e1005217.

[173] Benjamin Perret, Jean Cousty, Silvio J.F. Guimaraes, and Deise S Maia. "Evaluation of hierarchical watersheds." In: (2017). HAL preprint 01430865.

[174] Tobias Pietzsch, Stephan Saalfeld, Stephan Preibisch, and Pavel Tomancak. "BigDataViewer: visualization and processing for large image data sets." In: *Nature methods* 12.6 (2015), pp. 481–483.

[175] Stephen M Plaza. "Focused proofreading: efficiently extracting connectomes from segmented EM images." In: *arXiv preprint arXiv:1409.1199* (2014).

[176] Stephen M Plaza and Stuart E Berg. "Large-Scale Electron Microscopy Image Segmentation in Spark." In: *arXiv preprint arXiv:1604.00385* (2016).

[177] Tran Minh Quan, David GC Hilderbrand, and Won-Ki Jeong. "FusionNet: A deep fully residual convolutional neural network for image segmentation in connectomics." In: *arXiv:1612.05360* (2016).

[178] David Rolnick, Yaron Meirovitch, Toufiq Parag, Hanspeter Pfister, Viren Jain, Jeff W Lichtman, Edward S Boyden, and Nir Shavit. "Morphological error detection in 3d segmentations." In: *arXiv preprint arXiv:1705.10882* (2017).

[179] William R Gray Roncal, Dean M Kleissas, Joshua T Vogelstein, Priya Manavalan, Kunal Lillaney, Michael Pekala, Randal Burns, R Jacob Vogelstein, Carey E Priebe, Mark A Chevillet, et al. "An automated images-to-graphs framework for high resolution connectomics." In: *Frontiers in neuroinformatics* 9 (2015).

[180] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Proc. MICCAI* (2015), pp. 234–241.

[181] Loïc A Royer, William C Lemon, Raghav K Chhetri, Yinan Wan, Michael Coleman, Eugene W Myers, and Philipp J Keller. "Adaptive light-sheet microscopy for long-term, high-resolution imaging in living organisms." In: *Nature Biotechnology* 34 (Oct. 2016), p. 1267.

[182] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." In: *nature* 323.6088 (1986), pp. 533–536.

[183] Matthew RG Russell, Thomas R Lerner, Jemima J Burden, David O Nkwe, Annegret Pelchen-Matthews, Marie-Charlotte Domart, Joanne Durgan, Anne Weston, Martin L Jones, Christopher J Peddie, et al. "3D correlative light and electron microscopy of cultured cells using serial blockface scanning electron microscopy." In: *J Cell Sci* 130.1 (2017), pp. 278–291.

[184] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomančák. "CATMAID: collaborative annotation toolkit for massive amounts of image data." In: *Bioinformatics* 25.15 (2009), pp. 1984–1986.

[185] Stephan Saalfeld, Richard Fetter, Albert Cardona, and Pavel Tomancak. "Elastic volume reconstruction from series of ultra-thin microscopy sections." In: *Nature methods* 9.7 (2012), p. 717.

[186] Philippe Salembier and Luis Garrido. "Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and Information Retrieval." In: *IEEE Trans. Image Proc.* 9 (2000), pp. 561–576.

[187] Louis K Scheffer, C Shan Xu, Michal Januszewski, Zhiyuan Lu, Shin-ya Takemura, Kenneth J Hayworth, Gary B Huang, Kazunori Shinomiya, Jeremy Maitlin-Shepard, Stuart Berg, et al. "A connectome and analysis of the adult Drosophila central brain." In: *Elife* 9 (2020), e57443.

[188] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. "Fiji: an open-source platform for biological-image analysis." In: *Nature methods* 9.7 (2012), pp. 676–682.

[189] Philipp Schlegel, Marta Costa, and Gregory SXE Jefferis. "Learning from connectomics on the fly." In: *Current opinion in insect science* (2017).

[190] Benjamin Schmid, Johannes Schindelin, Albert Cardona, Mark Longair, and Martin Heisenberg. "A high-level 3D visualization API for Java and ImageJ." In: *BMC bioinformatics* 11.1 (2010), pp. 1–7.

[191] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. "Cell detection with star-convex polygons." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 265–273.

[192] Wei Shen, Bin Wang, Yuan Jiang, Yan Wang, and Alan Yuille. "Multi-stage Multi-recursive-input Fully Convolutional Networks for Neuronal Boundary Detection." In: *arXiv preprint arXiv:1703.08493* (2017).

[193] Paul Simion, Hervé Philippe, Denis Baurain, Muriel Jager, Daniel J Richter, Arnaud Di Franco, Béatrice Roure, Nori Satoh, Éric Quéinnec, Alexander Ereskovsky, et al. "A large and consistent phylogenomic dataset supports sponges as the sister group to all other animals." In: *Current Biology* 27.7 (2017), pp. 958–967.

[194] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *CoRR* abs/1409.1556 (2014).

[195] Natcha Simsiri, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. "Work-efficient parallel union-find." In: *Concurrency and Computation: Practice and Experience* 30.4 (2018), e4333.

[196] Nicholas Sofroniew et al. *napari/napari: 0.4.7rc1*. Version v0.4.7rc1. Mar. 2021.

[197] P. Soille. "Constrained Connectivity for Hierarchical Image Decomposition and Simplification." In: *IEEE Trans. Patt. Anal. Mach. Intell.* 30.7 (2008), pp. 1132–1145.

[198] Thorvald Sørensen. "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons." In: *Biol. Skr.* 5 (1948), pp. 1–34.

[199] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958.

[200] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. "Cellpose: a generalist algorithm for cellular segmentation." In: *Nature Methods* 18.1 (2021), pp. 100–106.

[201] Paul Swoboda and Bjoern Andres. "A Message Passing Algorithm for the Minimum Cost Multicut Problem." In: *arXiv preprint arXiv:1612.05441* (2016).

[202] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In: *Proc. CVPR* (2015), pp. 1–9.

[203] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In: *arXiv preprint arXiv:1512.00567* (2015).

[204] 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, et al. "A visual motion detection circuit suggested by Drosophila connectomics." In: *Nature* 500.7461 (2013), pp. 175–181.

[205] Shin-Ya Takemura et al. “Synaptic circuits and their variations within different columns in the visual system of Drosophila.” In: *Proc. Natl. Acad. Sci. USA* 112.44 (2015).

[206] Benjamin Titze, Christel Genoud, and Rainer W Friedrich. “SBEMimage: versatile acquisition control software for serial block-face electron microscopy.” In: *Frontiers in neural circuits* 12 (2018), p. 54.

[207] Dejan Tomaževič, Bostjan Likar, and Franjo Pernuš. “Comparative evaluation of retrospective shading correction methods.” In: *Journal of microscopy* 208.3 (2002), pp. 212–223.

[208] Burkhard Tönshoff et al. “Prevalence of SARS-CoV-2 Infection in Children and Their Parents in Southwest Germany.” In: *JAMA Pediatrics* (Jan. 2021).

[209] Zhuowen Tu. “Auto-context and its application to high-level vision tasks.” In: *Proc. CVPR* (2008), pp. 1–8.

[210] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and H Sebastian Seung. “Maximin affinity learning of image segmentation.” In: *Proc. NIPS* (2009), pp. 1865–1873.

[211] 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.” In: *Neural computation* 22.2 (2010), pp. 511–538.

[212] Mustafa Gokhan Uzunbas, Chao Chen, and Dimitris Metaxas. “An efficient conditional random field approach for automatic and interactive neuron segmentation.” In: *Med. Image Anal.* 27 (2016), pp. 31–44.

[213] Mustafa Gökhan Uzunbaş, Chao Chen, and Dimitris Metaxas. “Optree: a learning-based adaptive watershed algorithm for neuron segmentation.” In: *Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI’14)*. 2014, pp. 97–105.

[214] Corinne Vachier and Fernand Meyer. “Extinction value: a new measurement of persistence.” In: *Worksh. Nonlinear Signal and Image Processing*. Vol. 1. 1995, pp. 254–257.

[215] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. “scikit-image: image processing in Python.” In: *PeerJ* 2 (2014), e453.

[216] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. “Unsupervised semantic segmentation by contrasting object mask proposals.” In: *arXiv preprint arXiv:2102.06191* (2021).

[217] Amelio Vazquez-Reina, Michael Gelbart, Daniel Huang, Jeff Lichtman, Eric Miller, and Hanspeter Pfister. “Segmentation fusion for connectomics.” In: *Proc. Intl. Conf. Comp. Vision* (2011), pp. 177–184.

[218] Hernando M Vergara, Constantin Pape, Kimberly Meechan, Valentyna Zinchenko, Christel Genoud, Adrian A Wanner, Benjamin Titze, Rachel Templin, Paola Yanina Bertucci, Oleg Simakov, et al. “Whole-body integration of gene expression and single-cell morphology.” In: *bioRxiv* (2020).

[219] Hernando Martinez Vergara, Paola Yanina Bertucci, Peter Hantz, Maria Antonietta Tosches, Kaia Achim, Pavel Vopalensky, and Detlev Arendt. “Whole-organism cellular gene-expression atlas reveals conserved cell types in the ventral nerve cord of Platynereis dumerilii.” In: *PNAS* 114.23 (2017), pp. 5878–5885.

[220] Luc Vincent and Pierre Soille. “Watersheds in digital spaces: an efficient algorithm based on immersion simulations.” In: *IEEE Trans. Pattern Analysis Machine Intelligence* 6 (1991), pp. 583–598.

[221] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python.” In: *Nature methods* 17.3 (2020), pp. 261–272.

[222] Pavel Vopalensky, Maria Antonietta Tosches, Kaia Achim, Mette Handberg-Thorsager, and Detlev Arendt. “From spiral cleavage to bilateral symmetry: the developmental cell lineage of the annelid brain.” In: *BMC biology* 17.1 (2019), p. 81.

[223] Adrian A. Wanner, Christel Genoud, Tafheem Masudi, Lea Siksou, and Rainer W. Friedrich. “Dense EM-based reconstruction of the interglomerular projectome in the zebrafish olfactory bulb.” In: *Nat. Neurosci.* 19.6 (June 2016). Article, pp. 816–825.

[224] Joe H Ward Jr. “Hierarchical grouping to optimize an objective function.” In: *Journal of the American statistical association* 58.301 (1963), pp. 236–244.

[225] Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. “Star-convex polyhedra for 3d object detection and segmentation in microscopy.” In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 3666–3673.

[226] Maurice Weiler, Fred A Hamprecht, and Martin Storath. “Learning Steerable Filters for Rotation Equivariant CNNs.” In: *arXiv preprint arXiv:1711.07289* (2017).

[227] Nathan V Whelan, Kevin M Kocot, Tatiana P Moroz, Krishanu Mukherjee, Peter Williams, Gustav Paulay, Leonid L Moroz, and Kenneth M Halanych. “Ctenophore relationships and their placement as the sister group to all other animals.” In: *Nature ecology & evolution* 1.11 (2017), pp. 1737–1746.

[228] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. “The structure of the nervous system of the nematode Caenorhabditis elegans.” In: *Philos Trans R Soc Lond B Biol Sci* 314.1165 (1986), pp. 1–340.

[229] Jeffrey D Whitman, Joseph Hiatt, Cody T Mowery, Brian R Shy, Ruby Yu, Tori N Yamamoto, Ujjwal Rathore, Gregory M Goldgof, Caroline Whitty, Jonathan M Woo, et al. “Evaluation of SARS-CoV-2 serology assays reveals a range of test performance.” In: *Nature biotechnology* 38.10 (2020), pp. 1174–1183.

[230] Stiaan Wiehman and Hendrik de Villiers. “Semantic segmentation of bioimages using convolutional neural networks.” In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 624–631.

[231] Steffen Wolf, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A Hamprecht. “The Mutex Watershed and its Objective: Efficient, Parameter-Free Graph Partitioning.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[232] Steffen Wolf, Yuyan Li, Constantin Pape, Alberto Bailoni, Anna Kreshuk, and Fred A Hamprecht. “The Semantic Mutex Watershed for Efficient Bottom-Up Semantic Instance Segmentation.” In: *Proc. ECCV’2020* (2020).

[233] Steffen Wolf, Constantin Pape, Alberto Bailoni, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred Hamprecht. “The Mutex Watershed: Efficient, Parameter-Free Image Partitioning.” In: *Proc. ECCV’18* (2018).

[234] Steffen Wolf, Lukas Schott, Ullrich Köthe, and Fred Hamprecht. “Learned Watershed: End-to-End Learning of Seeded Segmentation.” In: *Proc. ICCV’17* (2017).

[235] Adrian Wolny, Qin Yu, Constantin Pape, and Anna Kreshuk. “Sparse Object-level Supervision for Instance Segmentation with Pixel Embeddings.” In: *arXiv preprint arXiv:2103.14572* (2021).

[236] Adrian Wolny et al. “Accurate and versatile 3D segmentation of plant tissues at cellular resolution.” In: *eLife* 9 (July 2020). Ed. by Christian S Hardtke, Dominique C Bergmann, Dominique C Bergmann, and Moritz Graeff, e57613.

[237] Fan Wu, Su Zhao, Bin Yu, Yan-Mei Chen, Wen Wang, Zhi-Gang Song, Yi Hu, Zhao-Wu Tao, Jun-Hua Tian, Yuan-Yuan Pei, et al. “A new coronavirus associated with human respiratory disease in China.” In: *Nature* 579.7798 (2020), pp. 265–269.

[238] Saining Xie and Zhuowen Tu. “Holistically-nested edge detection.” In: *Proc. ICCV’15*. 2015, pp. 1395–1403.

[239] C Shan Xu, Song Pang, Gleb Shtengel, Andreas Müller, Alex T Ritter, Huxley K Hoffman, Shin-ya Takemura, Zhiyuan Lu, H Amalia Pasolli, Nirmala Iyer, et al. "Isotropic 3D electron microscopy reference library of whole cells and tissues." In: *bioRxiv* (2020).

[240] Tao Zeng, Bian Wu, and Shuiwang Ji. "DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation." In: *Bioinformatics* 33.16 (2017), pp. 2555–2562.

[241] C. Zhang, Julian Yarkony, and Fred A. Hamprecht. "Cell detection and segmentation using correlation clustering." In: *Proc. MICCAI'14*. 2014, pp. 9–16.

[242] Zhihao Zheng, J Scott Lauritzen, Eric Perlman, Camenzind G Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B Fisher, Nadiya Sharifi, et al. "A complete electron microscopy volume of the brain of adult Drosophila melanogaster." In: *Cell* 174.3 (2018), pp. 730–743.

[243] Aleksandar Zlateski and H Sebastian Seung. "Image segmentation by size-dependent single linkage clustering of a watershed basin graph." In: *arXiv:1505.00249* (2015).

[244] Jonathan Zung, Ignacio Tartavull, Kisuk Lee, and H Sebastian Seung. "An error detection and correction framework for connectomics." In: *Advances in Neural Information Processing Systems*. 2017, pp. 6818–6829.

[245] Mark H Zweig and Gregory Campbell. "Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine." In: *Clinical chemistry* 39.4 (1993), pp. 561–577.