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

M.Sc. Alberto Bailoni

Born in: Trento, Italy

Oral examination: 19-07-2021

Deep Learning for Graph-Based Image Instance Segmentation

Referees: Prof. Dr. Fred A. Hamprecht

Prof. Dr. Carsten Rother

Abstract

Neuroscientists have been developing new electron microscopy imaging techniques and generating large volumes of data to reconstruct the complete neural wiring diagram of an organism's central nervous system. The sheer size of these volumes makes manual analysis infeasible. A fundamental step towards this goal is the automated segmentation of neural tissue images. This thesis presents new efficient deep learning methods for image instance segmentation and their applications to neuron segmentation.

Related work on instance segmentation focuses on training an accurate edge detector (represented by a deep learning model) to predict transitions between different object instances in an image. In this thesis, we propose novel graph partitioning algorithms that can efficiently process these edge predictions and produce an instance segmentation. We specifically focus on partitioning algorithms for signed graphs with both positive and negative edge weights. By using signed graphs, the partitioning algorithm can find a previously unspecified number of instances without requiring the user to manually specify additional parameters (e.g., a tunable threshold).

In this thesis, we introduce a simple and efficient graph partitioning algorithm, the *Mutex Watershed*, and prove its relation to the NP-hard multicut/correlation clustering optimization problem. We then propose a generalized framework for agglomerative graph clustering algorithms, called *GASP*, and prove that the Mutex Watershed is one of the algorithms covered by it. This unifying framework allows us to conveniently study both theoretical and empirical properties of the algorithms it describes. When combined with the predictions of a deep neural network, some of the algorithms in the framework constitute a segmentation pipeline that achieves state-of-the-art accuracy on the CREMI neuron segmentation challenge without requiring to tune domain-specific hyper-parameters.

Finally, this thesis proposes a new bottom-up instance segmentation method for large-scale volumetric images. The approach predicts single-instance segmentation masks across the entire image, one for each pixel, in a sliding window style. All masks are decoded from a low dimensional latent representation, which results in a memory-efficient pipeline. The method achieves competitive results on the CREMI neuron segmentation challenge and is considerably robust to noise due to prioritizing predictions with the highest consensus across overlapping masks.

Zusammenfassung

Neue Elektronenmikroskopiemethoden erlauben es Neurowissenschaftlern riesige Datenvolumen zu akquirieren, um das neuronale Verknüfungsmuster des zentralen Nervensystems vollständig zu rekonstruieren. Aufgrund der schieren Größe dieser Datensätze ist eine manuelle Analyse kaum möglich. Deswegen sind automatisierte Segmentierungsmethoden von Gehirngewebebildern unerlässlich. Diese Doktorarbeit entwickelt neue effiziente Deep Learning gestützte Instanzsegementierungsmethoden und deren Anwendung auf Neuronengewebebilder.

Bisherige Instanzsegmentierungsansätze konzentrieren sich auf die Entwicklung genauer Kantendetektoren (meist in der Form eines tiefen neuronalen Netzes), um Grenzen zwischen den verschiedenen Objektinstanzen eines Bilds zu bestimmen. Darauf aufbauend schlagen wir in dieser Arbeit graphenbasierte Partitionsalgorithmen vor, welche die prognostizierten Objektgrenzen nutzen, um Bildinstanzen zu bestimmen. Insbesondere betrachten wir Partitionierungsalgorithmen für Graphen mit sowohl positiven als auch negativen Kantengewichten. In solchen Graphen können Partitionierungsalgorithmen eine zuvor nicht spezifizierte Anzahl von Objektinstanzen finden, ohne auf händisch angepasste Parameter, zum Beispiel Schwellenwerte, zurückzugreifen.

In dieser Arbeit führen wir den einfachen und effizienten Graphpartitionierungsalgorithmus Mutex Watershed ein und zeigen seine Verbindung zum NP-schweren Multicut / Korrelationsclustering Optimierungsproblem. Anschließend entwickeln wir eine Systematik von agglomerativen Graphclusteringsalgorithmen, GASP, und ordnen den Mutex Watershed in diese Systematik ein. Die GASP Systematik vereinfacht unsere Analyse der theoretischen und praktischen Eigenschaften der beschriebenen Algorithmen. Kombiniert mit den eingangs erwähnten Kantenvorhersagen eines tiefen neuronalen Netzes, gehören einige der beschriebenen Algorithmen zu den besten Beiträgen im CREMI Neuronensegementierungswettbewerb, ohne aufwendige Optimierung von anwendungsspezifischen Hyperparametern zu benötigen.

Weiterhin schlagen wir in dieser Arbeit eine neue Instanzsegmentierungsmethode für große dreidimensionale Bilddaten vor. Bei diesem Ansatz werden Instanzen von unten nach oben bestimmt, indem für jeden Voxel basierend auf dessen Umgebung eine Maske der ihn enthaltenden Instanz vorhergesagt wird. Diese Masken werden aus einer niedrigdimensionalen Darstellung dekodiert, sodass der Speicherbedarf gering gehalten wird. Auch diese Methode produziert kompetitive Ergebnisse im CREMI Neuronensegementierungswettbewerb. Sie ist außerdem besonders widerstandsfähig gegenüber Bildstörungen, da die vorhergesagten Instanzen aufgrund von hoher Übereinstimmung zwischen den zahlreichen Instanzmasken, die sich in einem Voxel überlappen, ausgewählt werden.

Acknowledgments

First of all, I would like to thank my Ph.D. advisor Professor Fred Hamprecht, whose extensive knowledge and insight into the subject guided me through this project. He constantly challenged me to go deeper into the topic, and I much appreciated his support and encouragement both in positive and more challenging moments.

It has been a pleasure to collaborate with so many friendly, bright, and supportive people in the Image Analysis and Learning group. I especially want to thank Steffen Wolf for all the inspiring discussions about science and the many hours of fun work invested in joint research projects related to the Mutex Watershed algorithm. I also want to thank Constantin Pape for our close collaboration and the numerous brainstorming sessions on automatic segmentation of 3D EM images. Nasim Rahaman, Constantin Pape, and Steffen Wolf were a constant source of inspiration for me for developing better coding skills. I like to thank Sebastian Damrich for exciting and engaging discussions on science and beyond. I also want to thank my great officemate, Lorenzo Cerrone, for all the fun conversations that made my working days much more enjoyable. I would also like to thank present and previous lab members Steffen Wolf, Lorenzo Cerrone, Roman Remme, Sebastian Damrich, Carsten Haubold, and Elke Kirschbaum for the helpful input and for creating such an excellent and friendly working environment.

Particular thanks go to Anna Kreshuk and Ullrich Köthe for the valuable suggestions and fruitful discussions on image segmentation, algorithms, and other exciting topics. I would also like to thank Thomas Carraro for the collaboration on the project on segmentation of EM images of batteries; and Carsten Rother for being the second referee of this work. Special thanks to Barbara Werner, always so helpful and ready to fix any unexpected bureaucratic issue.

Finally, my most special thanks go to Katya, my dearest and irreplaceable friend without whom I would not be where and who I am today. Thanks to her, the past years have been so unique and magical. Lastly, I would like to thank my sister and my wonderful parents for their unconditional love and support throughout all the steps of my life.

Contents

Chapter 1

Introduction

1.1 Image Segmentation

The human brain is able to process information captured by our eyes and learn a remarkably rich representation of the world around us. However, developing an artificial system that can achieve the same performance and robustness has long challenged researchers from very diverse fields like psychology, physiology, engineering, computer science, and artificial intelligence.

Computer vision is a multidisciplinary field of science that enables computers to gain high-level understanding from digital images, videos, and other visual inputs. The idea of "understanding digital images" usually means deriving a compressed and meaningful description of the images that can then be used for further analysis. A typical example is the task of image classification, where an image is assigned to exactly one label from a fixed set of classes such as {cat, dog, person}. Another important example is *image segmentation*, which is the process of partitioning an image into meaningful segments or sets of pixels.

There exist two types of image segmentation: *Semantic segmentation* assigns each pixel of the image to a label from a defined set of classes (e.g. person, car, tree, sky); *Instance segmentation* assigns each pixel not only to a class but also to a unique instance id so that different object instances belonging to the same class are distinguished (e.g. car-1, car-2, person-1). In this thesis, we focus on instance segmentation. An example of an application is the segmentation of neuronal tissue images (see Fig. 1.2 and 1.3). In this task, there is only one class of objects (neuron cells), and each pixel has to be assigned to its corresponding neuron cell so that all pixels belonging to the same neuron cell are grouped together.

In the following sections, we will introduce the methods and graph partitioning algorithms that are studied in this thesis, and show how they can be used to solve instance segmentation tasks such as neuron segmentation.

1.2 Deep Learning and Graph-Based Instance Segmentation

In the last decade, computer vision experienced incredible progress. This was due in big part to deep learning, which is now omnipresent in the field of image analysis. As a machine learning tool, fully-connected neural networks (also known as multilayer perceptrons) encode the input through a number of non-linear fully-connected layers, where each neuron in one layer is connected to all neurons in the next layer. The neural network model's architecture is defined by the arrangement of these layers. Convolutional neural networks (CNNs) represent one of the most relevant classes of neural network architectures used in computer vision and digital image analysis. A convolutional layer of a CNN convolves its input and passes the result to the next layer. Convolution kernels shift over input features and provide in this way translation equivariant responses. As compared to fully-connected neurons, a convolutional neuron processes data only for its receptive field, remarkably reducing the number of parameters in the network and making it less prone to overfitting data. Thanks to these properties, CNNs recently demonstrated outstanding performances at image-level understanding and recognition capabilities.

A specific type of CNN, called fully convolutional network (FCN) [98], proved to be particularly good at solving image segmentation tasks. An FCN does not include any fully-connected layer and transforms the height and width of the intermediate layers back to the input image's size. In this way, when applied to a semantic segmentation task, the pixel-wise class predictions of an FCN have a one-to-one correspondence with the input image in the spatial dimension. Few years ago, an improved version of FCN was proposed, called U-Net [131], which is based on an encoder-decoder structure along with long skip connections. These skip connections proved to be very successful at recovering fine-grained details in the output predictions and, recently, the U-Net model became a prevalent choice for solving image segmentation tasks in biological applications [90, 131]. In the following chapters of this thesis, we will frequently make use of the U-Net model.

FCNs and CNNs have also been applied to instance segmentation tasks. There are two main types of deep learning approaches to instance segmentation: *proposal-based* and *proposal-free* methods. *Proposal-based* methods first perform object detection, for example by predicting anchor boxes [128], and then assign a class and a binary segmentation mask to each detected bounding box [61, 125]. However, the bounding boxes predicted by these methods may overlap, which is not always desirable. In this thesis, instead, we study *proposal-free* methods, which do not require object detection and directly group pixels into instances. These types of methods are preferred in imagery with object instances that cannot be approximated by bounding boxes and are much larger than the field of view of the model.

This thesis focuses on graph-based proposal-free segmentation methods, which solve the instance segmentation task by formalizing the input image as a graph. Nodes in the graph usually represent pixels or sets of pixels (commonly named *superpixels*), whereas graph edges express neighboring relationships between pixels. In these graph-based instance segmentation approaches, an FCN is trained to predict the graph's edge weights. Then, a more or less complex graph-partitioning algorithm is employed to output the final instance segmentation. The segmentation methods that are mostly related to this thesis predict edge weights in the form of pixel-pair affinities [56, 90, 97], which represent how likely it is for a pair of pixels to be in the same object instance. These approaches not only predict affinities for pairs of direct-neighboring pixels in the image, but they

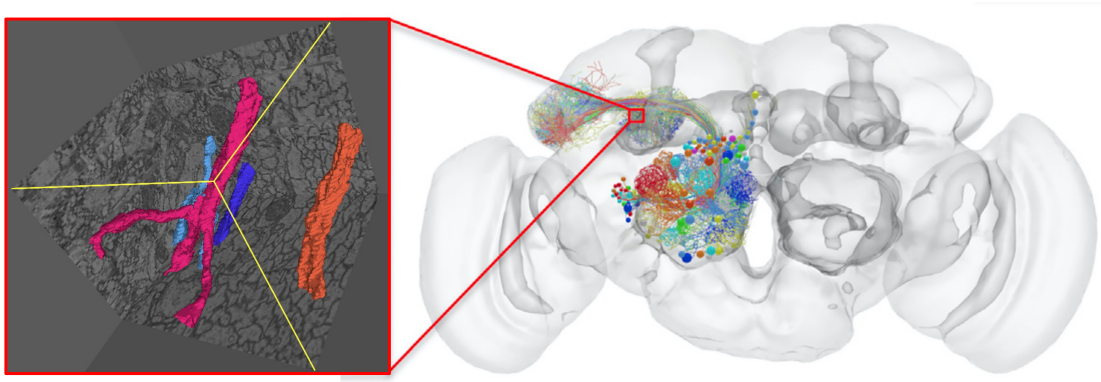


Figure 1.1. Reconstruction of the neural wiring diagram of the fruit-fly *Drosophila melanogaster*. The neuronal processes are reconstructed from serial section transmission EM (TEM) volumes (shown on the left) of the organism's neural tissue. Four automatically reconstructed neuron instances are displayed in different colors. On the right, the ventral view of the brain is shown (image taken from [165]).

also learn long-range pixel-pair affinities, since this proved to improve training and help the model to use large-scale features in images [90].

In the following section, we will review the task of neuron segmentation in the field of connectomics, which has often been tackled by using graph-based instance segmentations methods. Then, in Section 1.3, we will introduce the main graph partitioning algorithms studied in this thesis.

1.2.1 Neuron Segmentation in Connectomics

Connectomics is a field of neuroscience with the goal of reconstructing the complete neural wiring diagram of an organism's central nervous system. This comprehensive map of neural connections within a brain is known as the *connectome*. A fundamental step towards the reconstruction of the connectome is the segmentation of neural tissue images, which are commonly acquired using electron microscopy techniques (e.g., serial section transmission EM) yielding 3D image volumes.

In 2017, the whole brain of an adult Drosophila melanogaster, with a volume of approximately $8{\cdot}10^7 \mu m^3$ and comprising ∼100,000 neurons has been imaged with nanometer resolution producing a dataset of 106 TB [165]. Similar projects are generating petabytes of data [162], and a mouse brain of 500 mm^3, at a typical isotropic resolution of 8 nm, would require almost 1000 petabytes of data. Despite progress in collaborative annotation [74], the sheer size of these volumes makes manual analysis infeasible. Thus, automated processing, and especially automated neuron segmentation, is needed to reconstruct the complete organism's neural wiring diagram.

The unique features of this segmentation problem, involving large volume datasets and narrow elongated neuron segments that can span big portions of an organism's brain, encouraged the development of new proposal-free instance segmentation methods. Flood-filling networks [64] and MaskExtend [101] use a CNN to iteratively extend one neuron at a time. By using flood-filling networks, the dense connectome of half the central brain of Drosophila melanogaster, containing around 25,000 neurons, was recently reconstructed and publicly released [159]. In order to yield sufficient accuracy for correct circuit analysis, this impressive reconstruction involved significant proof-reading efforts by expert

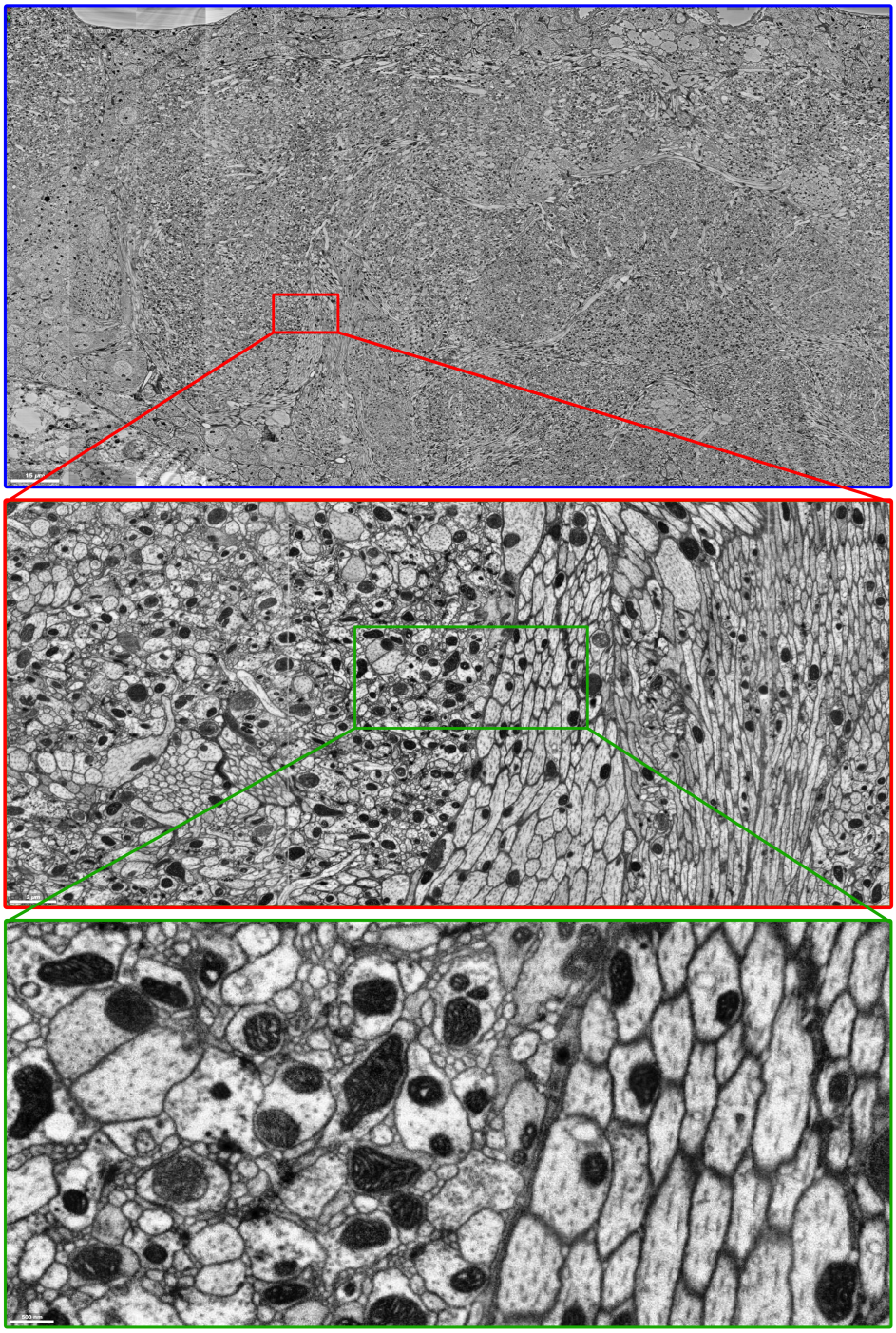


Figure 1.2. Example of electron microscopy volume image of neuronal tissue with isotropic resolution of $8 \times 8 \times 8$ nm^3 shown at three different scales. Data is taken from the hemibrain [160]: the dataset covers a large portion of the central brain of the fruit fly *Drosophila melanogaster*, including the mushroom body and central complex circuits critical for associative learning and fly navigation. Data is three dimensional. Only one 2D image from the stack is shown.

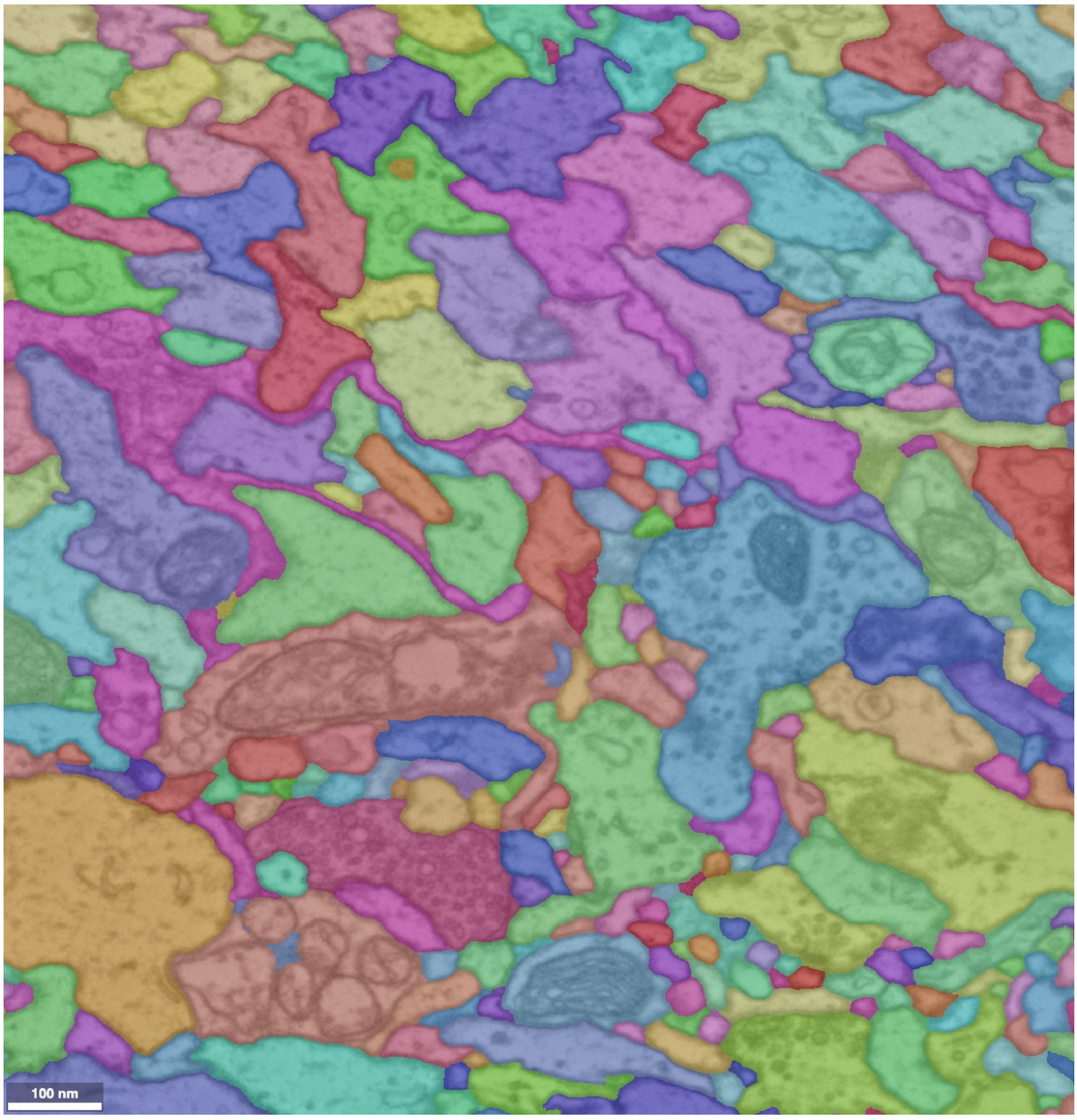


Figure 1.3. Neuron segmentation data overlaid with the dense instance segmentation obtained with the method proposed in Chapter 3 (*GASP* with average linkage). Colors are randomly assigned. Note that the data is 3D, hence the same color could be assigned to parts of segments that appear disconnected in 2D. Only one 2D image from the stack is shown here. Data is taken from the test set of the CREMI neuron segmentation challenge [40].

annotators, showing that further progress is still required in automated reconstruction methods.

Another class of instance segmentation methods that proved to be particularly useful in neuron segmentation is given by graph-based segmentation approaches. Originally, many proposed methods predicted a boundary evidence map indicating how probable it is for every voxel to be on the boundary between two neurons. After generating superpixels and building a graph based on these boundary predictions, the final segmentation was then determined as optimal cuts on this graph [6, 7, 20, 54, 102, 145]. Now, all of the top submissions of the CREMI [40], SNEMI3D [140], and ISBI neuron segmentation challenges employ convolutional neural networks [15, 90, 99]. Instead of predicting a neuron-boundary probability map, most of these methods predict affinities between voxels and directly use them to compute the edge weights of the graph [90, 99, 117, 118, 154]. An alternative approach to the direct prediction of affinities was proposed by [89], who instead learn dense voxel embeddings via deep metric learning and derive affinities in the embedded space.

In the next section, we will review some of the graph partitioning algorithms used in these graph-based segmentation pipelines and studied in this thesis.

1.3 Graph Partitioning Algorithms

In the previous sections, we have seen that graph-based instance segmentation methods have been successfully applied to neuron segmentation. In this section, we review the basic ideas and notation of graph-based segmentation algorithms studied in this thesis.

Usually, graph-based image segmentation methods represent the image as a graph $\mathcal{G} = (V, E)$. Each node $u \in V$ corresponds to a pixel in the image and nodes are connected by edges $(u, v) \in E$. A weight w_e is associated with each edge $e \in E$ based on some property of the pixels that it connects, such as their image intensities, gradients or the output of an edge classifier (usually a neural network). Depending on the method and application, the graph might be only sparsely connected, for example as a grid graph (where every pixel is connected only its direct neighboring pixels) or a graph with limited local neighborhood connectivity.

1.3.1 Agglomerative Hierarchical Clustering

The majority of graph clustering methods work with positive edge weights representing similarities or distances between the nodes. These methods require the user to specify the desired numbers of segments or a termination criterion (as in spectral clustering or iterated normalized cuts) or even a stronger version of supervision by adding a seed for each object (e.g. in seeded watershed or random walker).

Hierarchical clustering (HC) is another popular graph clustering method, which creates a hierarchy of clusters. Agglomerative HC is a bottom-up approach starting with each node assigned to its own cluster and incrementally merging clusters starting from those with the highest edge weight [86]. As compared to other divisive partitioning algorithms (e.g. normalized graph cut), agglomerative clustering is a much more efficient method that, with a heap data-structure, has a time complexity of $\mathcal{O}(m^2 \log m)$, where m is the number of edges in the graph.

In Algorithm 1, we show a simplified pseudo-code describing this simple partitioning algorithm. Apart from the weighted graph, another input of the algorithm is the so-called

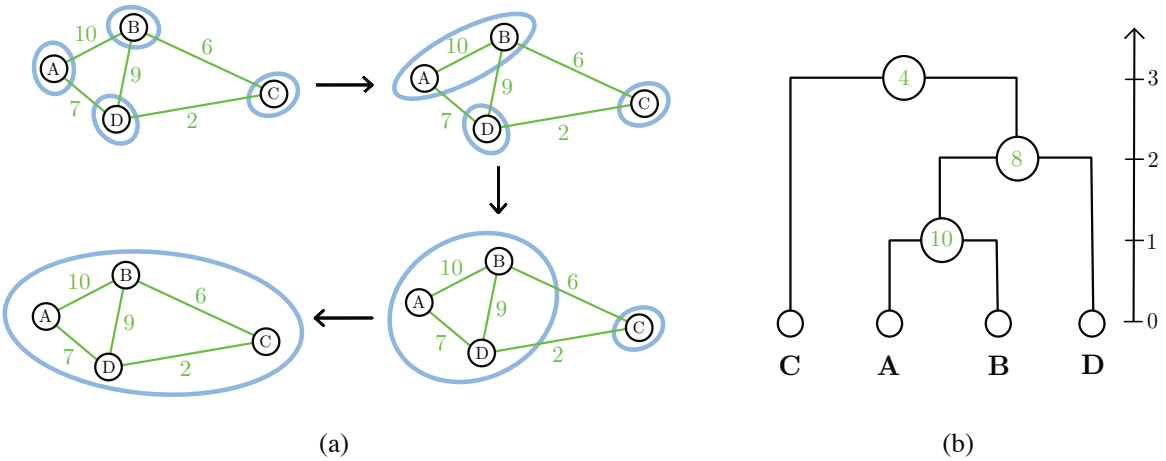


Figure 1.4. Agglomerative Hierarchical Clustering (with Average linkage criterion) on a toy graph. **(a)** On the left, the agglomeration steps of the algorithms are demonstrated on a small graph with four nodes. Edge weights are shown in green. At every step, the pair of clusters with highest interaction (according to average linkage) is merged. **(b)** On the right, the resulting dendrogram is shown, representing the merging hierarchy.

Algorithm 1 Agglomerative Hierarchical Clustering

Input: Graph $\mathcal{G}(V, E, w)$ with affinity weights $w : E \to \mathbb{R}^+$; linkage criterion $\mathcal{W}$

1: Initialize clustering $\{\{v_1\}, \ldots, \{v_{|V|}\}\}$, where each node is in its own cluster
2: Initialize empty dendrogram T
3: **repeat**
4: Select the two clusters S and S' with highest interaction
5: Merge the two selected clusters S and S', and update the dendrogram T accordingly
6: Update interactions between the new merged cluster $S \cup S'$ and its neighboring clusters, based on the linkage criterion $\mathcal{W}$
7: **until** All nodes have been merged into a single cluster
8: **return** Dendrogram representing the merging order of clusters

linkage criterion, which defines the interaction between two clusters containing multiple nodes. For example, average linkage criterion is a very common one, where the interaction between two clusters is given by the average of all the edge weights connecting them. Each time two clusters are merged by the algorithm at line 5, the interaction between the newly formed merged cluster and its neighbors has to be updated by using the given linkage criterion. Finally, the algorithm returns the hierarchy of clusters in the form of a *dendrogram*, which is a rooted tree representing the order in which clusters were merged (see toy example in Fig. 1.4).

A downside of hierarchical agglomerative algorithms is that, after running the algorithm, the user has to choose a level in the cluster hierarchy in order to define the desired output clustering. In Chapter 3, we will define a framework for agglomerative algorithms that are parameter-free because they are based on graphs with both positive and negative edge weights.

1.3.2 Signed Graph Partitioning

In contrast to the class of algorithms introduced in the previous section, this thesis will investigate algorithms that can deal with both positive and negative edge weights, corresponding to attraction and repulsion between nodes. Such a graph with positive and negative edge weights is commonly known as *signed weighted graph*. The advantage of using signed graphs is that balancing attraction and repulsion allows to perform the clustering without defining additional parameters like the number of final clusters, making this problem's formalization particularly suited for applications where the number of clusters is a-priori unknown. Balancing attraction and repulsion in the graph can be done optimally by solving the so-called *multicut optimization problem* or *correlation clustering* [32,68], which minimizes the sum of weights between clusters and can be formally defined by the following integer linear program (for a more specific definition, see Section 2.4.1):

$$\min_{\Pi} \sum_{e \in E} w_e x_e^{\Pi}, \qquad \text{where} \quad x_e^{\Pi} = \begin{cases} 1 & \text{if } e \in E_{\Pi}^1 \\ 0 & \text{otherwise,} \end{cases} \tag{1.1}$$

where Π denotes one of the possible clusterings of the graph $\mathcal{G}(V, E, w)$ with signed edge weights $w : E \to \mathbb{R}$; and $E_{\Pi}^1 \subseteq E$ is the set of edges "on cut" (i.e. all edges that link nodes belonging to distinct clusters of Π). In general, solving the minimum multicut problem is known to be NP-hard. Therefore any exact solver will fail to scale to large graphs. However, in neuron segmentation and connectomics many approximate solvers have been proposed [18,87,117,161] together with greedy agglomerative clustering algorithms [69, 72, 92].

This thesis studies novel partitioning algorithms that are both parameter-free and efficient. In Chapter 2, we introduce a new partitioning algorithm, the Mutex Watershed, and prove that it optimally optimizes an objective closely related to the multicut's objective in Eq. 1.1. Then, in Chapter 3, we show that the Mutex Watershed is actually only one specific element of a larger class of agglomerative algorithms for signed graph partitioning.

1.4 Contribution and Overview of this Thesis

Throughout the past few years, proposal-free instance segmentation methods have been successfully applied to biological imagery and achieved higher and higher accuracy, primarily thanks to novel training strategies [90, 106]. Several of these methods train an edge classifier (namely, a fully convolutional neural network) that can accurately predict transitions between object instances. This thesis proposes novel graph partitioning algorithms that can process these edge predictions and output an instance segmentation. The proposed algorithms are efficient and achieve state-of-the-art segmentation results without requiring the user to spend much time tuning complex dataset-dependent hyperparameters. The centerpiece of this thesis is a unifying framework, named GASP, for agglomerative clustering algorithms of graphs with both attractive and repulsive interactions between the nodes (Chapter 3). In the following, we give a brief overview of each chapter's content.

Chapter 2: We propose an efficient algorithm for graph partitioning, the "Mutex Watershed", and relate it to the multicut objective. The algorithm is deterministic, very simple to implement, and has empirically linearithmic complexity. Unlike seeded watershed, the Mutex Watershed algorithm can accommodate not only attractive but also repulsive cues, allowing it to find a previously *unspecified* number of segments without the need for explicit seeds or a tunable threshold. We also prove that this simple algorithm solves to global optimality an objective function that is closely related to the multicut's objective.

Chapter 3: We introduce a review framework for graph clustering, named GASP, that represents a generalization of agglomerative hierarchical clustering to graphs with both attractive and repulsive edge weights. Thanks to this framework, we explore many combinations of different linkage criteria, and study both theoretical and empirical properties of these combinations. We show that various existing partitioning algorithms can be reformulated in our framework and introduce new algorithms for combinations that have not been studied before. Finally, we conduct a comprehensive comparison of GASP instantiations on a large variety of both synthetic and existing signed clustering problems, in terms of accuracy but also efficiency and robustness to noise.

Chapter 4: We propose a new proposal-free instance segmentation method that is based on single-instance segmentation masks predicted across the entire image in a sliding window style. This method concurrently predicts all masks, one for each pixel, and thus resolves any conflict jointly across the entire image. Masks are decoded from a low dimensional latent representation, which results in great memory savings required for applications to large volumetric images. Finally, predictions from overlapping masks are combined to obtain all final instances concurrently. The result is a parameter-free method that is strongly robust to noise and prioritizes predictions with the highest consensus across overlapping masks.

Chapter 2

The Mutex Watershed Algorithm and its Objective

In this chapter, we propose an efficient algorithm for graph partitioning, the "Mutex Watershed", and relate it to the multicut problem. Most prior work either requires seeds, one per segment; or a threshold; or formulates the task as a multicut problem. Unlike seeded watershed, the Mutex Watershed algorithm can accommodate not only attractive but also repulsive cues, allowing it to find a previously *unspecified* number of segments without the need for explicit seeds or a tunable threshold. We also prove that this simple algorithm solves to global optimality an objective function that is intimately related to the multicut's integer linear programming formulation. The algorithm is deterministic, very simple to implement, and has empirically linearithmic complexity. When presented with short-range attractive and long-range repulsive cues from a deep neural network, the Mutex Watershed gives state-of-the-art results in the ISBI 2012 EM segmentation benchmark.

2.1 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 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 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.

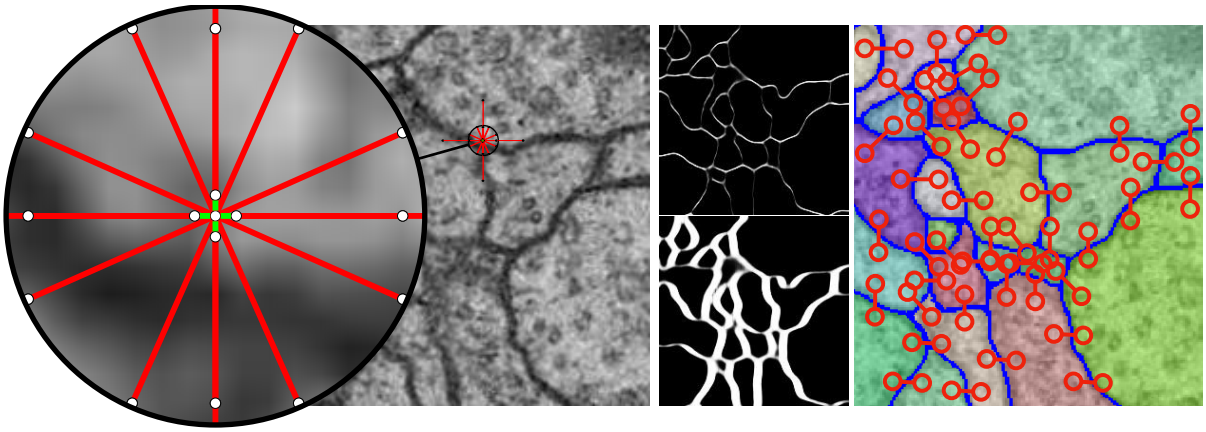


Figure 2.1. Left: Overlay of raw data from the ISBI 2012 EM segmentation challenge and the edges for which attractive (green) or repulsive (red) interactions are estimated for each pixel using a CNN. Middle: vertical / horizontal repulsive interactions at intermediate / long range are shown in the top / bottom half. Right: Active mutual exclusion (mutex) constraints that the proposed algorithm invokes during the segmentation process.

The main insight of this chapter 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 or 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 Fig. 2.1 for a visual abstract.

In summary, the principal contributions of this chapter are, first, a fast deterministic algorithm for graph partitioning with both positive and negative edge weights that does not need prior specification of the number of clusters (section 2.4); and second, its theoretical characterization, including proof that it globally optimizes an objective related to the multicut correlation clustering objective (2.4).

Combined with a deep net, the algorithm also happens to define the state-of-the-art in a competitive neuron segmentation challenge (Section 2.5).

2.2 Related Work

In the original watershed algorithm [22, 149], 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 [23]: 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, many automatic methods for seed selection have been proposed in the last decades with applications in the fields of medicine and biology. Many of these approaches rely on edge feature extraction and edge detection like gradient calculation [4, 123]. Other types of methods generate seeds by first performing feature extraction [124, 156], whereas others first extract region of interests and then place seeds

inside these regions by using thresholding [3], binarization [138], k-means [109] or other strategies [1, 2].

In applications where the number of regions is hard to estimate, simple automatic seed selection methods, e.g. defining seeds by connected regions of low boundary probability, don't work: The segmentation quality is usually insufficient because multiple seeds are in the same region and/or seeds leak through the boundary. Thus, in these cases seed selection may be biased 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 postprocessing. This works better than using watersheds alone because it exploits the larger context afforded by superpixel adjacency graphs. Many criteria have been proposed to identify the regions to be preserved during merging, e.g. region dynamics [57], the waterfall transform [21], extinction values [148], region saliency [112], and (α, ω)-connected components [142]. 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 [122].

These results highlight the close connection of watersheds to hierarchical clustering and minimum spanning trees/forests [105, 110], which inspired novel merging strategies and termination criteria. For example, [134] simply terminated hierarchical merging by fixing the number of surviving regions beforehand. [100] incorporate predefined sets of generalized merge constraints into the clustering algorithm. Graph-based segmentation according to [50] 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 [60, 78], 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 given by partitioning of graphs with both attractive and repulsive edges [73]. 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. Fortunately, modern integer linear programming solvers in combination with incremental constraint generation can solve problem instances of considerable size [8], and good approximations exist for even larger problems [117, 161] Reminiscent of strict minimizers [91] with minimal L_∞-norm solution, our work solves the multicut objective optimally when all graph weights are raised to a large power.

Related to the proposed method, the greedy additive edge contraction (GAEC) [72] heuristic for the multicut also sequentially merges regions, but we handle attractive and repulsive interactions separately and define edge strength between clusters by a maximum instead of an additive rule. The greedy fixation algorithm introduced in [92] is closely related to the proposed method; it sorts attractive and repulsive edges by their absolute weight, merges nodes connected by attractive edges and introduces no-merge constraints for repulsive edges. However, similar to GAEC, it defines edge strength by an additive rule, which increases the algorithm's runtime complexity compared to the presented Mutex Watershed. Also, it is not yet known what objective the algorithm optimizes globally,

if any.

Another beneficial extension is the introduction of additional long-range edges. The strength of such edges can often be estimated with greater certainty than is achievable for the local edges used by watersheds on standard 4- or 8-connected pixel graphs. Such repulsive long-range edges have been used in [164] to represent object diameter constraints, 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) [62]. Realistic problem sizes can only be solved approximately [18, 72], but watershed superpixels followed by LMC postprocessing achieve state-of-the-art results on important benchmarks [20]. Long-range edges are also used in [90], as side losses for the boundary detection convolutional neural network (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 neurosegmentation for connectomics, an important field we also address in the experimental section. CNNs were introduced to this application in [63] and became, in much refined form [36], the winning entry of the ISBI 2012 Neuro-Segmentation Challenge [11]. Boundary maps and superpixels were further improved by progress in CNN architectures and data augmentation methods, using U-Nets [132], FusionNets [126] or inception modules [20]. Subsequent postprocessing with the GALA algorithm [80,115], conditional random fields [146] or the lifted multicut [20] pushed the envelope of final segmentation quality. MaskExtend [101] applied CNNs to both boundary map prediction and superpixel merging, while flood-filling networks [65] eliminated superpixels altogether by training a recurrent neural network to perform region growing one region 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 [119, 166] using edge weights generated with a CNN [26,145]. Tayloring the learning objective to the needs of the watershed algorithm by penalizing critical edges along minimax paths [26] or end-to-end training of edge weights and region growing [155] improved results yet again.

Outside of connectomics, [13] obtained superior boundary maps from CNNs by learning not just boundary strength, but also its gradient direction. Holistically-nested edge detection [81, 158] couples the CNN loss at multiple resolutions using deep supervision and is successfully used as a basis for watershed segmentation of medical images in [27].

We adopt important ideas from this prior work (hierarchical single-linkage clustering, attractive and repulsive interactions, long-range edges, and CNN-based learning). The proposed efficient segmentation framework can be interpreted as a generalization of [100], because we also allow for soft repulsive interactions (which can be overridden by strong attractive edges), and constraints are generated on-the-fly.

2.3 The Mutex Watershed Algorithm as an Extension of Seeded Watershed

In this section we introduce the Mutex Watershed Algorithm, an efficient graph clustering algorithm that can ingest both attractive and repulsive cues. We first reformulate seeded watershed as a graph partitioning with infinitely repulsive edges and then derive the generalized algorithm for finitely repulsive edges, which obviates the need for seeds.

Algorithm 2 Mutex version of seeded watershed algorithm

procedure SEEDEDWATERSHED($\mathcal{G}(V, E)$, pos. weights $w : E \to \mathbb{R}^+$, seeds $S \subseteq V$)
  $A^+ \leftarrow \emptyset$
  $A^- \leftarrow \{(s, t) \in S \times S \,|\, s \neq t\}$
  for $(i, j) = e \in E$ in descending order of w_e **do**
    if not connected$(i, j; A^+)$ **and not** mutex$(i, j; A^+, A^-)$ **then**
      $A^+ \leftarrow A^+ \cup e$ ▷ Merge i, j and inherit mutex constraints of the parent clusters
  return $A^+ \cup A^-$

The output clustering is defined by the connected components of the final attractive active set A^+.

2.3.1 Definitions and notation

Let $\mathcal{G} = (V, E, w)$ be a weighted graph. The scalar attribute $w : E \to \mathbb{R}$ associated with each edge is a merge affinity: the higher this number, the higher the inclination of the two incident vertices to be assigned to the same cluster. Conversely, large negative affinity indicates a greater desire of the incident vertices to be in different clusters. In our application, each vertex corresponds to one pixel in the image to be segmented. We call an edge $e \in E$ repulsive if $w_e < 0$ and we call it attractive if $w_e > 0$ and collect them in $E^- = \{e \in E \,|\, w_e < 0\}$ and $E^+ = \{e \in E \mid w_e > 0\}$ respectively.

In our application, each vertex corresponds to one pixel in the image to be segmented. The Mutex Watershed algorithm, defined in Section 2.3.3, 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:

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

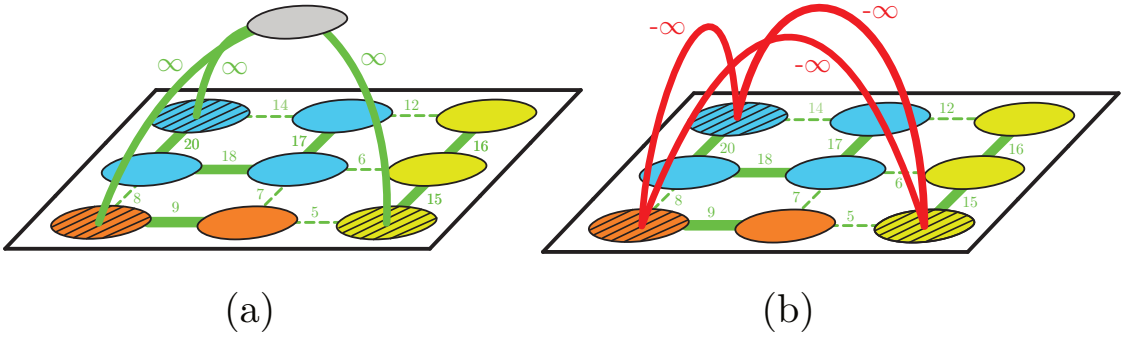
Conversely, the active subset $A^- \subseteq E^-$ of repulsive edges defines mutual exclusion relations by using the following predicate:

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

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: mutex$(i, j; A^+, A^-) \Rightarrow$ not connected$(i, j; A^+)$. The "connected" and "mutex" predicates can be efficiently evaluated using a union find data structure.

2.3.2 Seeded watershed from a mutex perspective

One interpretation of the proposed method is in terms of a generalization of the edge-based watershed algorithm [103–105] or image foresting transform [48]. This algorithm can only ingest a graph with purely attractive interactions, $E^- = \emptyset$. Without further constraints, the algorithm would yield only the trivial result of a single cluster comprising



(a) (b)

Figure 2.2. Two equivalent representations of the seeded watershed clustering obtained using (a) a maximum spanning tree computation or (b) Algorithm 2. 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.

all vertices. To obtain more interesting output, an oracle needs to provide seeds (e.g. one node per cluster). These seed vertices are all connected to an auxiliary node (see Fig. 2.2 (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 [48, 104, 105].

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 (Fig. 2.2 (b)). Algorithm 2 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.

2.3.3 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. Neither seeds nor hyperparameters that implicitly determine the number of resulting clusters are required.

The Mutex Watershed, Algorithm 3, proceeds as follows. Given a graph $\mathcal{G} = (V, E)$

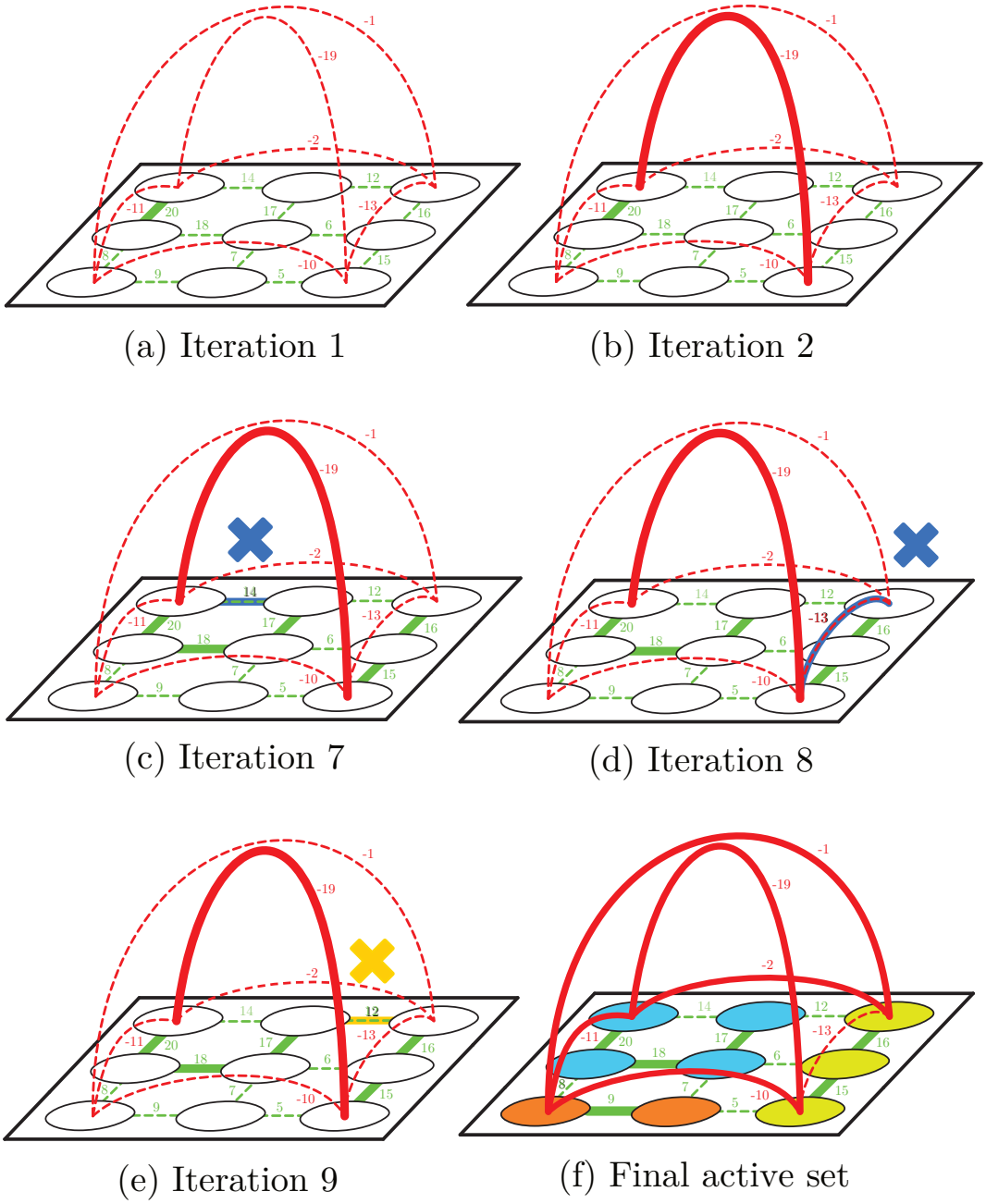


Figure 2.3. Some iterations of the Mutex Watershed Algorithm 3 applied to a graph with weighted attractive (green) and repulsive (red) edges. Edges accumulated in the active set A after a given number of iterations are shown in bold. The connect_all parameter of the algorithm is set to False, so that only the positive edges belonging to the maximum spanning tree of each cluster are added to the active set. Once the algorithm terminates, the final active set (f) defines the final clustering (indicated using arbitrary node colors). Some edges are not added to the active set because they are mutex constrained (yellow highlight) or because the associated nodes are already connected and in the same cluster (blue highlight).

Algorithm 3 Mutex Watershed Algorithm

procedure MUTEXWATERSHED($\mathcal{G}(V, E)$, $w : E \to \mathbb{R}$, boolean connect_all)
  $A^+ \leftarrow \emptyset$; $A^- \leftarrow \emptyset$
  for $(i, j) = e \in E$ in descending order of $|w_e|$ **do**
    if $e \in E^+$ **then**
      if not mutex$(i, j; A^+, A^-)$ **then**
        if not connected$(i, j; A^+)$ **or** connect_all **then**
          merge(i, j): $A^+ \leftarrow A^+ \cup e$ ▷ Merge i, j and inherit constraints of parent clusters
    else
      if not connected$(i, j; A^+)$ **then**
        addmutex(i, j): $A^- \leftarrow A^- \cup e$ ▷ Add mutex constraint between i and j
  return $A^+ \cup A^-$

The output clustering is defined by the connected components of the final attractive active set A^+. The connect_all parameter changes the internal cluster connectedness from trees to fully connected, but does not change the output clustering. The connected predicate can be efficiently evaluated using union find data structures.

with signed weights $w : E \to \mathbb{R}$, do the following: sort all edges E, attractive or repulsive, by their absolute 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, assuming connect_all is False, 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 output clustering is defined by the connected components of the final attractive active set A^+.

The crucial difference to Algorithm 2 is that mutex constraints are no longer predefined, 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.

The boolean value of the connect_all input parameter of the algorithm does not influence the final output clustering, but defines the internal cluster connectedness: when it is set to True, the algorithm adds all attractive intra-cluster edges to the active set A^+. When it is set to False, then a maximum spanning tree is built for each cluster similarly to the seeded watershed. This variant of the algorithm will be helpful in the next section 2.4 to highlight the relation between the Mutex Watershed and the multicut problem.

Fig. 2.3 illustrates the proposed algorithm: Fig. 2.3a and Fig. 2.3b show examples of an unconstrained merge and an added mutex constraint, respectively; Fig. 2.3c and Fig. 2.3d show, respectively, an example of an attractive edge ($w_e = 14$) and repulsive edge ($w_e = -13$) that are not added to the active set because their incident vertices are already "connected" and belong to the same tree of the forest A^+; finally, Fig. 2.3e shows an attractive edge ($w_e = 12$) that is ruled out by a previously introduced mutual exclusion relation.

2.3.4 Time Complexity Analysis

Before analyzing the time complexity of algorithm 3 we first review the complexity of Kruskal's algorithm. Using a union-find data structure (with path compression and union by rank) the time complexity of merge$(i,\ j)$ and connected$(i,\ j)$ is $\mathcal{O}(\alpha(V))$, where α is the slowly growing inverse Ackerman function, and the total runtime complexity is dominated by the initial sorting of the edges $\mathcal{O}(E \log E)$ [38].
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 $C_i = \text{cluster}(i)$ using hash tables, where insertion of new mutex edges (i.e. 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 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 mutex$(i,\ j)$ has an average complexity of $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$. Similarly, during 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 $\mathcal{O}(\max(|E^+| \cdot \alpha(V), |E^+| \cdot M))$ (checking mutex constraints and possibly merging) and repulsive edges $\mathcal{O}(\max(|E^-| \cdot \alpha(V), |E^-|))$ (insertion of one mutex edge) result in a total average runtime complexity of algorithm 3:

$$\mathcal{O}(\max(E \log E\ ,\ EM)). \tag{2.1}$$

where M is the expected value of $\min(|M[C_i]|, |M[C_j]|)$ and $\alpha(V) \in \mathcal{O}(\log V) \in \mathcal{O}(\log E)$ [1].
In the worst case $\mathcal{O}(M) \in \mathcal{O}(E)$, 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 2.4), leading to a

$$\text{Empirical Mutex Watershed Complexity: } \mathcal{O}(E \log E) \tag{2.2}$$

2.4 Theoretical characterization

Towards the Multicut framework. In section 2.3.3, we have introduced the Mutex Watershed (MWS) algorithm as a generalization of seeded watersheds and the Kruskal algorithm in particular. However, since we are considering graphs with negative edge weights, the MWS is conceptually closer to the multicut problem and related heuristics such as GAEC and GF [92]. Fortunately, due to the structure of the MWS it can be analyzed using dynamic programming. This section summarizes our second contribution, i.e. the proof that the Mutex Watershed Algorithm globally optimizes a precise objective related to the multicut.

2.4.1 Review of the Multicut problem and its objective

In the following, we will review the multicut problem not in its standard formulation but in the *Cycle Covering Formulation* introduced in [88], which is similar to the MWS

[1] In the worst case G is a fully connected graph, with $|E| = |V|^2$, hence $\log |V| = \frac{1}{2} \log |E|$.

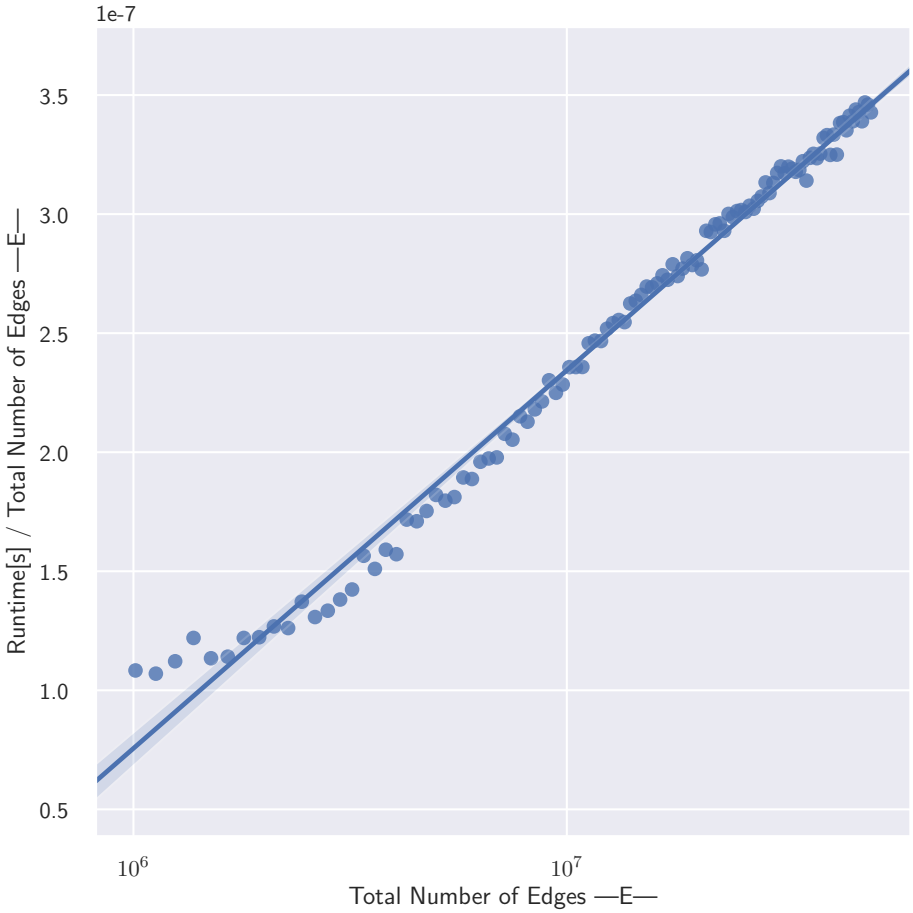


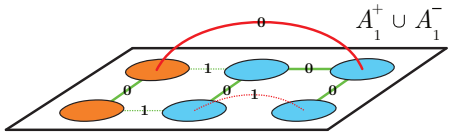
Figure 2.4. Runtime T of Mutex Watershed (without sorting of edges) measured on sub-volumes of the ISBI challenge of different sizes (thereby varying the total number of edges E). We plot $\frac{T}{|E|}$ over $|E|$ in a logarithmic plot, which makes $T \sim |E|log(|E|)$ appear as straight line. A logarithmic function (blue line) is fitted to the measured $\frac{T}{|E|}$ (blue circles) with ($R^2 = 0.9896$). The good fit suggests that empirically $T \approx \mathcal{O}(E \log E)$.

formulation as it also considers the set of *attractive* and *repulsive* edges separately. Previously, in Sec. 2.3.1, we defined a clustering by introducing the concept of an active set of edges $A = A^+ \cup A^- \subseteq E$ and the connected/mutex predicates. In particular, an active set describes a valid clustering if it does not include **both** a path of only attractive edges **and** a path with exactly one repulsive edge connecting any two nodes $i, j \in V$:

$$\text{connected}(i, j; A^+) \quad \Longrightarrow \quad \textbf{not } \text{mutex}(i, j; A^+, A^-). \tag{2.3}$$

In other words, an active set is *consistent* and describes a clustering if it does not contain any cycle with exactly one repulsive edge (known as conflicted cycles).

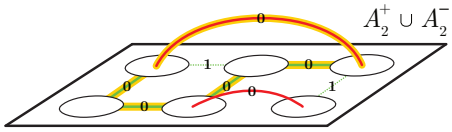
Definition 2.4.1. ***Conflicted cycles*** – *We call a cycle of $\mathcal{G}$ conflicted w.r.t. $(\mathcal{G}, w)$ if it contains precisely one repulsive edge $e \in E^-$, s.t. $w_e < 0$. We denote by $\mathcal{C}^-(\mathcal{G}, w) \subseteq \mathcal{C}(\mathcal{G}, w)$ the set of all conflicted cycles. Furthermore, given a set of edges $A \subseteq E$, we denote by $\mathcal{C}^-(A, \mathcal{G}, w) \subseteq \mathcal{C}^-(\mathcal{G}, w)$ the set of conflicted cycles involving only edges in A.*



$\forall i,j \in V$ connected$(i,j;A_1^+) \Rightarrow$ **not** mutex$(i,j;A_1^+,A_1^-)$

$\mathcal{C}^-(A_1,\mathcal{G},w) = \emptyset$

$x^{A_1} \in \mathsf{SC}(\mathcal{G},w)$



$\exists\, i,j \in V$ connected$(i,j;A_2^+)$ and mutex$(i,j;A_2^+,A_2^-)$

$\mathcal{C}^-(A_2,\mathcal{G},w) \neq \emptyset$

$x^{A_2} \notin \mathsf{SC}(\mathcal{G},w)$

Figure 2.5. Consistent and inconsistent active sets – Two different active edge sets $A_1 \subseteq E$ (on the left) and $A_2 \subseteq E$ (on the right) on identical toy graphs with six nodes, attractive (green) and repulsive (red) edges. The value of the edge indicator $x^A \in \{0,1\}^{|E|}$ defined in Eq. 2.4 is shown for every edge. Members of the active sets are shown as solid lines. **On the left**, the active set A_1 is *consistent*, i.e. does not include any conflicted cycle $\mathcal{C}^-(\mathcal{G},w)$ (see Def. 2.4.1): Therefore, it is associated with a clustering (represented by arbitrary node colors). **On the right**, the active set A_2 is not consistent and includes at least one conflicted cycle (highlighted in yellow), thus it cannot be associated with a node clustering.

From now on, in order to describe different clustering solutions in the framework of (integer) linear programs, we associate each active set A with the following edge indicator x^A

$$x^A := \mathbb{1}\{e \notin A)\} \in \{0,1\}^{|E|}. \tag{2.4}$$

In this way, the cycle-free property $\mathcal{C}^-(A,\mathcal{G},w) = \emptyset$ of an active set can be reformulated in terms of linear inequalities:

$$\forall C \in \mathcal{C}^-(\mathcal{G},w) : \sum_{e \in E_C} x_e^A \geq 1 \quad \Longleftrightarrow \quad \mathcal{C}^-(A,\mathcal{G},w) = \emptyset. \tag{2.5}$$

In words, the active set cannot contain conflicted cycles; or vice versa, every conflicted cycle must contain at least one edge that is not part of the active set. Following [88], via this property we describe the space of all possible clustering solutions by defining the convex hull $\mathsf{SC}(\mathcal{G},w)$ of all edge indicators corresponding to valid clusterings of $(\mathcal{G},w)$:

Definition 2.4.2. *Let* $\mathsf{SC}(\mathcal{G},w)$ *denote the convex hull of all edge indicators* $x \in \{0,1\}^{|E|}$ *satisfying the following system of inequalities:*

$$\forall C \in \mathcal{C}^-(\mathcal{G},w) : \quad \sum_{e \in E_C} x_e \geq 1. \tag{2.6}$$

That is, $\mathsf{SC}(\mathcal{G},w)$ *contains all edge labelings for which every conflicted cycle is broken at least once. We call* $\mathsf{SC}(\mathcal{G},w)$ *the set covering polyhedron with respect to conflicted cycles, similarly to [88].*

Fig. 2.5 summarizes these definitions and provides an example of consistent and inconsistent active sets with their associated clusterings and edge indicators.

As shown in [88], the *multicut optimization problem* can be formulated with constraints over conflicted cycles in terms of the following integer linear program (ILP), which is NP-hard:

$$\min_{x \in \mathsf{SC}(\mathcal{G},w)} \sum_{e \in E} |w_e| x_e. \tag{2.7}$$

The solution of the multicut problem is given by the clustering associated to the connected components of the active set $\hat{A}^+ = \{e \in E^+ | \hat{x}_e = 0\}$, where $\hat{x} \in \{0,1\}^{|E|}$ is the solution of (2.7).

2.4.2 Mutex Watershed Objective

We now define the Mutex Watershed objective that is minimized by the Mutex Watershed Algorithm (proof in subsection 2.4.3) and show how it is closely related to the multicut problem defined in Eq. (2.7). Lange et al. [88] introduce the concept of dominant edges in a graph. For example, an attractive edge $f \in E^+$ is called dominant if there exists a cut B with $f \in E_B$ such that $|w_f| \geq \sum_{e \in E_B \setminus \{f\}} |w_e|$. These highlight an aspect of the multicut problem that can be used to search for optimal solutions more efficiently. Not all weighted graphs contain dominant edges; but if, assuming no ties, we raise all graph weights to a large enough power a similar property emerges.

Definition 2.4.3. ***Dominant power:*** *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, with unique weights $w : E \to \mathbb{R}$. We call $p \in \mathbb{N}^+$ a dominant power if:*

$$|w_e|^p > \sum_{t \in E,\, w_t < w_e} |w_t|^p \qquad \forall e \in E, \tag{2.8}$$

In contrast to dominant edges [88], we do not consider edges on a cut but rather all edges with smaller absolute weight. Note that there exists a dominant power for any finite set of edges, since for any $e \in E$ we can divide (2.8) by $|w_e|^p$ and observe that the normalized weights $|w_t|^p / |w_e|^p$ (and any finite sum of these weights) converges to 0 when p tends to infinity.

By considering the multicut problem in Eq. (2.7) and raising the weights $|w_e|$ to a dominant power p, we fundamentally change the problem structure:

Definition 2.4.4. ***Mutex Watershed Objective:*** *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, with unique weights $w : E \to \mathbb{R}$ and $p \in \mathbb{N}^+$ a dominant power. Then the Mutex Watershed Objective is defined as the integer linear program*

$$\min_{x \in \mathsf{SC}(\mathcal{G}, w)} \quad \sum_{e \in E} |w_e|^p \, x_e \tag{2.9}$$

where $\mathsf{SC}(\mathcal{G}, w)$ is the convex hull defined in Def. 2.4.2.

In the following section, we will prove that this modified version of the multicut objective, which we call Mutex Watershed Objective, is indeed optimized by the Mutex Watershed Algorithm:

Theorem 2.4.1. *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, with unique weights $w : E \to \mathbb{R}$ and $p \in \mathbb{N}^+$ a dominant power. Then the edge indicator given by the Mutex Watershed Algorithm 3*

$$x^{\mathbf{MWS}} := \mathbb{1}\Big\{ e \notin \mathbf{MWS}\Big(\mathcal{G}, w, \mathit{connect\_all}{=}\mathit{True}\Big) \Big\}$$

minimizes the Mutex Watershed Objective in Eq. (2.9).

Algorithm 4 Conflicted-Cycles Mutex Watershed

procedure ConflictedCyclesMWS($\mathcal{G}(V, E), w : E \to \mathbb{R}$)
  $A \leftarrow \emptyset$
  for $(i, j) = e \in E$ in descending order of $|w_e|$ **do**
    if $\mathcal{C}^-(A \cup \{e\}, \mathcal{G}, w) = \emptyset$ **then**
      $A \leftarrow A \cup e$
  return A

Equivalent formulation of the Mutex Watershed Algorithm 3, with input parameter connect_all=True. The set of conflicted cycles $\mathcal{C}^-(A, \mathcal{G}, w)$ is defined in Def. 2.4.1. The output clustering is defined by the connected components of the final attractive active set $A^+ = A \cap E^+$.

2.4.3 Proof of optimality via dynamic programming

In this section we prove Theorem 2.4.1, i.e. that the Mutex Watershed Objective defined in 2.4.4 is solved to optimality by the Mutex Watershed Algorithm 4. Particularly, in the following Sec. 2.4.3 we show that the edge indicator associated to the solution of the MWS algorithm lies in $\mathsf{SC}(\mathcal{G}, w)$, whereas in Sec. 2.4.3 we prove that it solves Eq. 2.9 to optimality.

Cycle consistency

The Mutex Watershed algorithm introduced in Sec. 2.3 iteratively builds an active set $A = A^+ \cup A^-$ such that nodes engaged in a mutual exclusion constraint (encoded by edges in A^-) are never part of the same cluster. In other words, this means that the active set built by the Mutex Watershed at every iteration does never include a *conflicted cycle* and is always *consistent*. In particular, for any attractive edge $(i, j) = e^+ \in E^+$ and any consistent set A that fulfills $\mathcal{C}^-(A, \mathcal{G}, w) = \emptyset$:

$$\textbf{not } \text{mutex}(i, j, A^+, A^-) \quad \Leftrightarrow \quad \mathcal{C}^-(A \cup \{e^+\}, \mathcal{G}, w) = \emptyset$$

Similarly, for any repulsive edge $(s, t) = e^- \in E^-$:

$$\textbf{not } \text{connected}(s, t, A^+) \quad \Leftrightarrow \quad \mathcal{C}^-(A \cup \{e^-\}, \mathcal{G}, w) = \emptyset$$

Therefore, we can rewrite Algorithm 3 in the form of Algorithm 4. This new formulation makes it clear that

$$\mathcal{C}^-\Big(\textbf{MWS}\big(\mathcal{G}, w, \text{connect\_all=True}\big)\Big) = \emptyset. \tag{2.10}$$

Thus, thanks to Eq. 2.5 and definition 2.4.2, it follows that the MWS edge indicator $x^{\textbf{MWS}}$ defined in 2.4.1 lies in $\mathsf{SC}(\mathcal{G}, w)$:

$$x^{\textbf{MWS}} \in \mathsf{SC}(\mathcal{G}, w). \tag{2.11}$$

Optimality

We first note that the Mutex Watershed Objective 2.4.4 and Theorem 2.4.1 can easily be reformulated in terms of active sets to minimize

$$\underset{A \subseteq E}{\arg\min} \quad -\sum_{e \in A} |w_e|^p \qquad \text{s.t.} \quad \mathcal{C}^-(A, \mathcal{G}, w) = \emptyset. \tag{2.12}$$

We now generalize the Mutex Watershed (see Algorithm 5) and the objective such that an initial consistent set of active edges $\tilde{A} \subseteq E$ is supplied:

Algorithm 5 Initialized Mutex Watershed

```
procedure InitializedMWS((G(V, E), w : E → ℝ, initial active set Ã)
  A ← ∅
  for e ∈ E \ Ã in descending order of weight do
    if C⁻(A ∪ Ã ∪ {e}, G, w) = ∅ then
      A ← A ∪ e
  return A
```

Mutex Watershed algorithm starting from initial active set $\tilde{A}$. An initial set $\tilde{A}$ of active edges is given as additional input and the final active set is such that $A \subseteq E \setminus \tilde{A}$. Note that Algorithm 4 is a special case of this algorithm when $\tilde{A} = \emptyset$. Differences with Algorithm 4 are highlighted in blue.

Definition 2.4.5. ***Energy optimization subproblem.*** *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph. Define the optimal solution of the subproblem as*

$$S(\mathcal{G}, \tilde{A}) := \underset{A \subseteq (E \setminus \tilde{A})}{\operatorname{argmin}} T(A) \qquad \text{with} \quad T(A) := -\sum_{e \in A} |w_e|^p, \tag{2.13}$$

$$s.t. \quad \mathcal{C}^-(A \cup \tilde{A}, \mathcal{G}, w) = \emptyset, \tag{2.14}$$

where $\tilde{A} \subseteq E$ is a set of initially activated edges such that $\mathcal{C}^-(\tilde{A}, \mathcal{G}, w) = \emptyset$.

We note that for $\tilde{A} = \emptyset$, the optimal solution $S(\mathcal{G}, \emptyset)$ is equivalent to the solution minimizing the Mutex Watershed Objective and Eq. (2.12).

Definition 2.4.6. ***Incomplete, consistent initial set:*** *For an edge-weighted graph $\mathcal{G} = (V, E, w)$ a set of edges $\tilde{A} \subseteq E$ is consistent if*

$$\mathcal{C}^-(\tilde{A}, \mathcal{G}, w) = \emptyset. \tag{2.15}$$

$\tilde{A}$ is incomplete if it is not the final solution and there exists a consistent edge $\tilde{e}$ that can be added to $\tilde{A}$ without violating the constraints.

$$\exists\, \tilde{e} \in E \setminus \tilde{A} \quad s.t. \quad \mathcal{C}^-(\tilde{A} \cup \{\tilde{e}\}, \mathcal{G}, w) = \emptyset \tag{2.16}$$

Definition 2.4.7. ***First greedy step:*** *Let us consider an incomplete, consistent initial active set $\tilde{A} \subseteq E$ on $\mathcal{G} = (V, E, w)$. We define*

$$g := \underset{e \in (E \setminus \tilde{A})}{\operatorname{argmax}} |w(e)| \quad s.t. \quad \mathcal{C}^-(\tilde{A} \cup \{e\}, \mathcal{G}, w) = \emptyset. \tag{2.17}$$

as the feasible edge with the highest weight, which is always the first greedy step of Algorithm 5.

In the following two lemmas, we prove that the Mutex Watershed problem has an *optimal substructure property* and a *greedy choice property* [38], which are sufficient to prove that the Mutex Watershed algorithm finds the optimum of the Mutex Watershed Objective.

Lemma 2.4.2. ***Greedy-choice property.*** *For an incomplete, consistent initial active set $\tilde{A}$ of the Mutex Watershed, the first greedy step g is always part of the optimal solution*

$$g \in S(\mathcal{G}, \tilde{A}).$$

Proof. We will prove the theorem by contradiction by assuming that the first greedy choice is not part of the optimal solution, i.e. $g \notin S(\mathcal{G}, \tilde{A})$. Since g is by definition the feasible edge with highest weight, it follows that:

$$|w(e)| < |w(g)| \quad \forall e \in S(\mathcal{G}, \tilde{A}). \tag{2.18}$$

We now consider the alternative active set $A' = \{g\}$, that is a consistent solution, with

$$T(A') = -|w_g|^p \overset{(2.8)}{<} -\sum_{t \in S(\mathcal{G}, \tilde{A})} |w_t|^p = T\Big(S(\mathcal{G}, \tilde{A})\Big) \tag{2.19}$$

which contradicts the optimality of $S(\mathcal{G}, \tilde{A})$. □

Lemma 2.4.3. ***Optimal substructure property.*** *Let us consider an initial active set $\tilde{A}$, the optimization problem defined in Equation 2.13, and assume to have an incomplete, consistent problem (see Def. 2.4.6). Then it follows that:*

1. *After making the first greedy choice g, we are left with a subproblem that can be seen as a new optimization problem of the same structure;*
2. *The optimal solution $S(\mathcal{G}, \tilde{A})$ is always given by the combination of the first greedy choice and the optimal solution of the remaining subproblem.*

Proof. After making the first greedy choice and selecting the first feasible edge g defined in Equation 2.17, we are clearly left with a new optimization problem of the same structure that has the following optimal solution: $S(\mathcal{G}, \tilde{A} \cup \{g\})$.
In order to prove the second point of the theorem, we now show that:

$$S(\mathcal{G}, \tilde{A}) = \{g\} \cup S(\mathcal{G}, \tilde{A} \cup \{g\}). \tag{2.20}$$

Since algorithm 5 fulfills the greedy-choice property, $g \in S(\mathcal{G}, \tilde{A})$ and we can add the edge g as an additional constraint to the optimal solution:

$$\begin{aligned} S(\mathcal{G}, \tilde{A}) = &\operatorname*{argmin}_{A \subseteq (E \setminus \tilde{A})} \quad T(A) \\ &\text{s. t.} \quad \mathcal{C}^-(A \cup \tilde{A}, \mathcal{G}, w) = \emptyset; \quad g \in A \end{aligned} \tag{2.21}$$

Then it follows that:

$$\begin{aligned} S(\mathcal{G}, \tilde{A}) = \{g\} \cup &\operatorname*{argmin}_{A \subseteq\, E \setminus (\tilde{A} \cup \{g\})} \quad T(A) \\ &\text{s. t.} \quad \mathcal{C}^-\Big(A \cup \{g\} \cup \tilde{A}, \mathcal{G}, w\Big) = \emptyset \end{aligned} \tag{2.22}$$

which is equivalent to Equation 2.20. □

Proof of Theorems 2.4.1. In Lemmas 2.4.2 and 2.4.3 we have proven that the optimization problem defined in 2.12 has the optimal substructure and a greedy choice property. It follows through induction that the final active set $\mathbf{MWS}\big(\mathcal{G}, w, \text{connect\_all=True}\big)$ found by the Mutex Watershed Algorithm 4 is the optimal solution for the Mutex Watershed objective (2.12) [38]. □

Algorithm 6 Generic hierarchical optimization

procedure GHO($Q_0, \dots, Q_{t-1}$)
  $M_0 = \arg\min_{x \in \mathbb{R}^m} Q_0(x)$
  for $k \in 1, \dots, t-1$ **do**
    $M_k = \arg\min_{x \in M_{k-1}} Q_k(x)$
  Return: some $x^* \in M_{t-1}$
  return some $x^* \in M_{t-1}$

Generic hierarchical optimization algorithm introduced in [111]. The sequence of continuous functions $Q_k : \mathbb{R}^m \to \mathbb{R}$ is sorted according to the associated scales λ_k (Eq. 2.23).

2.4.4 Relation to the extended Power Watershed framework

The Power Watershed [39] is an important framework for graph-based image segmentation that includes several algorithms like seeded watershed, random walker and graph cuts. Recently, [111] extended the framework to even more general types of hierarchical optimization algorithms thanks to the use of Γ-theory and Γ-convergence [24, 43]. In this section, we show how the Mutex Watershed algorithm can also be included in this extended framework[2] and how the framework suggests an optimization problem that is solved by the Mutex Watershed.

Mutex Watershed as hierarchical optimization algorithm

We first start by introducing the extended Power Watershed framework and restating the main theorem from [111]:

Theorem 2.4.4. ***[111] Extended Power Watershed Framework.*** *Consider three strictly positive integers $p, m, t \in \mathbb{N}^+$ and t real numbers*

$$1 \geq \lambda_0 > \lambda_1 > \dots \lambda_{t-1} > 0 \tag{2.23}$$

Given t continuous functions $Q_k : \mathbb{R}^m \to \mathbb{R}$ with $0 \leq k < t$, define the function

$$Q^p(x) := \sum_{0 \leq k < t} \lambda_k^p Q_k(x). \tag{2.24}$$

Then, if any sequence $(x_p)_{p>0}$ of minimizers x_p of $Q^p(x)$ is bounded (i.e. there exists $C > 0$ such that for all $p > 0$, $||x_p||_\infty \leq C$), the sequence is convergent, up to taking a subsequence, toward a point of M_{t-1}, which is the set of minimizers recursively defined in Algorithm 6.

Proof. See [111] (Theorem 3.3). □

We now show that the Mutex Watershed algorithm can be seen as a special case of the generic hierarchical Algorithm 6, for a specific choice of scales λ_k and functions $Q_k(x) : \mathbb{R}^m \to \mathbb{R}$ (see definitions (2.25, 2.26) below) .

Scales λ_k: Let $\tilde{w}_k$ be the signed edge weights $w : E \to \mathbb{R}$ ordered by decreasing absolute value $|\tilde{w}_1| > |\tilde{w}_2| > \dots > |\tilde{w}_{t-1}|$. If two edges share the same weight, then the weight

[2] The connection between the Mutex Watershed and the extended Power Watershed framework was kindly pointed out by an anonymous reviewer.

Algorithm 7 PowerWatershed Mutex Watershed

procedure PWSMWS($Q_0, \dots, Q_{t-1}$)
  $M_0 = \arg\min_{x \in \mathbb{R}^{|E|}} Q_0(x) = \mathsf{ISC}(\mathcal{G}, w)$
  for $k \in 1, \dots, t-1$ **do**
    $M_k = \arg\min_{x \in M_{k-1}} \sum_{e \in E_k} x_e$
  return some $x^* \in M_{t-1}$

Special case of the general hierarchical Algorithm 6 obtained by substituting Def. (2.25) and (2.26). With the additional assumption of unique signed weights $w : E \to \mathbb{R}$, this algorithm is equivalent to the Mutex Watershed Algorithm 4. The sequence of functions $Q_k : \mathbb{R}^m \to \mathbb{R}$ defined in Eq. 2.26 is sorted according to the associated scales λ_k in Eq. 2.25. $\mathsf{ISC}(\mathcal{G}, w)$ is defined in Eq. 2.27

is called $\tilde{w}_k$ for both and $E_k \subseteq E$ denotes the set of all edges with weight $\tilde{w}_k$. We then define *the scales* λ_k as

$$\lambda_k := \begin{cases} 1 & \text{if } k = 0 \\ \left| \frac{\tilde{w}_k}{2\tilde{w}_1} \right| & \text{otherwise.} \end{cases} \tag{2.25}$$

The continuous functions $Q_k(x) : \mathbb{R}^{|E|} \to \mathbb{R}$ are defined as follows

$$Q_k(x) := \begin{cases} |E| \cdot \min_{x' \in \mathsf{ISC}(\mathcal{G}, w)} ||x' - x|| & \text{if } k = 0 \\ \sum_{e \in E_k} x_e & \text{otherwise,} \end{cases} \tag{2.26}$$

where $\mathsf{ISC}(\mathcal{G}, w)$ is defined as:

$$\mathsf{ISC}(\mathcal{G}, w) := \mathsf{SC}(\mathcal{G}, w) \cap \{0, 1\}^{|E|}. \tag{2.27}$$

In words, $Q_0(x)$ is proportional to the distance between x and the closest point on the set $\mathsf{ISC}(\mathcal{G}, w)$, whereas $Q_k(x)$ depends only on the indicators x_e of edges in E_k, for $k > 0$.

Algorithm 7 is obtained by substituting the scales λ_k and functions $Q_k(x)$ (respectively defined in Eq. (2.25) and (2.26)) into Algorithm 6 . The algorithm starts by setting M_0 to $\mathsf{ISC}(\mathcal{G}, w)$, i.e. by restricting the space of the solutions only to integer edge labelings x that do not include any conflicted cycles. Then, in the following iterations $k \in 1, \dots, t-1$, the algorithm solves a series of minimization sub-problems that in the most general case are NP-hard, even though they involve a smaller set of edges $E_k \subseteq E$. Nevertheless, if we assume that all weights are distinct, then $|E_k| = 1$ for all k and the solution to the sub-problems amounts to checking if the new edge can be labeled with $x_e = 0$ without introducing any conflicted cycles. This procedure is identical to Algorithm 3: at every iteration, the Mutex Watershed tries to add an edge to the active set A, provided that no mutual exclusion constraints are violated.

In summary, the framework in [111] provides a new formulation of the Mutex Watershed Algorithm that is even applicable to graphs with tied edge weights. In practice, when edge weights are estimated by a CNN, we do not expect tied edge weights.

Convergence of the sequence of minimizers

In this section, we see how Theorem 2.4.4 also suggests a minimization problem that is solved by the Mutex Watershed algorithm. A short summary is given in the final paragraph of the section.
First, we make sure that the conditions of Theorem 2.4.4 are satisfied when we apply it to Algorithm 7:

Lemma 2.4.5. *Let us consider the scales λ_k and continuous functions $Q_k(x) : \mathbb{R}^{|E|} \to \mathbb{R}$ respectively defined in Eq. (2.25) and (2.26). For any value of $p \in \mathbb{N}^+$, let $x_p \in \mathbb{R}^{|E|}$ be a minimizer of the function $Q^p(x)$ defined in Eq. (2.24). Then, the minimizer x_p lies in the set $\mathsf{ISC}(\mathcal{G}, w)$. From this, it follows that any sequence of minimizers $(x_p)_{p>0}$ is bounded and the conditions of Theorem 2.4.4 are satisfied.*

Proof. The function $Q^p(x)$ can be explicitly written as (see Eq. 2.24, 2.25 and 2.26):

$$Q^p(x) = \sum_{0 \le k < t} \lambda_k^p Q_k(x) \tag{2.28}$$

$$= |E| \min_{x' \in \mathsf{ISC}(\mathcal{G},w)} ||x - x'|| + \sum_{1 \le k < t} \left| \frac{\tilde{w}_k}{2\tilde{w}_1} \right|^p \sum_{e \in E_k} x_e \tag{2.29}$$

$$= |E| \min_{x' \in \mathsf{ISC}(\mathcal{G},w)} ||x - x'|| + \sum_{e \in E} \left| \frac{w_e}{2\tilde{w}_1} \right|^p x_e. \tag{2.30}$$

We then denote these two terms by:

$$Q^p_{\mathrm{A}}(x) := |E| \min_{x' \in \mathsf{ISC}(\mathcal{G},w)} ||x - x'||, \tag{2.31}$$

$$Q^p_{\mathrm{B}}(x) := \sum_{e \in E} \left| \frac{w_e}{2\tilde{w}_1} \right|^p x_e. \tag{2.32}$$

Intuitively, we now prove that the minimizer x_p of $Q^p(x)$ lies in $\mathsf{ISC}(\mathcal{G}, w)$ by showing that the first term $Q^p_{\mathrm{A}}(x)$ is always "dominant" as compared to $Q^p_{\mathrm{B}}(x)$.
First, we note that the gradient of the first term $Q^p_{\mathrm{A}}(x)$ has always norm equal to $|E|$ and points in the direction of the closest point $x' \in \mathsf{ISC}(\mathcal{G}, w)$. Given a generic point $y \in \mathbb{R}^{|E|}$, the only two cases when the gradient $\nabla_x Q^p_{\mathrm{A}}(x)$ does not exists are: i) if $y \in \mathsf{ISC}(\mathcal{G}, w)$; ii) if there are at least two points $x'', x''' \in \mathsf{ISC}(\mathcal{G}, w)$ such that $||y - x''|| = ||y - x'''||$. Clearly, $Q^p_{\mathrm{A}}(x)$ presents minima only in the first case, when $y \in \mathsf{ISC}(\mathcal{G}, w)$.
On the other hand, the second term $Q^p_{\mathrm{B}}(x)$ is always differentiable and the norm of its gradient is never greater than $\sqrt{|E|}$:

$$||\nabla_x Q^p_{\mathrm{B}}(x)|| < \left|\left| \nabla_x \left(\sum_{e \in E} x_e \right) \right|\right| = \sqrt{|E|} \tag{2.33}$$

where we used the fact that $\tilde{w}_k / 2\tilde{w}_1 < 1$ for every $1 \le k < t$. Thus, the magnitude of the gradient given by the first term is always larger compared to the one given by the second term. We then conclude that the objective can always be reduced unless x_p is a point of $\mathsf{ISC}(\mathcal{G}, w)$. □

Then, given any $p \in \mathbb{N}^+$ and the Def. (2.25, 2.26), we have that the minimization of the function $Q^p(x)$ defined in Eq. (2.24) is given by the following problem:

$$\arg\min_{x \in \mathbb{R}^m} Q^p(x) = \arg\min_{x \in \mathbb{R}^m} \sum_{0 \le k < t} \lambda_k^p Q_k(x) \tag{2.34}$$

$$= \arg\min_{x \in \mathsf{ISC}(\mathcal{G},w)} \sum_{1 \le k < t} \left| \frac{\tilde{w}_k}{2\tilde{w}_1} \right|^p \sum_{e \in E_k} x_e \tag{2.35}$$

$$= \arg\min_{x \in \mathsf{ISC}(\mathcal{G},w)} \frac{1}{|2\tilde{w}_1|^p} \sum_{e \in E} |w_e|^p \, x_e \tag{2.36}$$

where we used Lemma 2.4.5 and restricted the domain of the arg min operation to $\mathsf{ISC}(\mathcal{G}, w)$, so that $Q_0(x) = 0$ for all $x \in \mathsf{ISC}(\mathcal{G}, w)$.

It follows from Lemma 2.4.5 and Theorem 2.4.4 that a sequence of minimizers $(x_p)_{p>0}$ of the problem (2.36) converge, up to taking a subsequence, to the solution x^* returned by Algorithm 7. More specifically, we know that any minimizer x_p of (2.36) is in the discrete set $\mathsf{ISC}(\mathcal{G}, w)$. Hence, the convergent sequence of minimizers $(x_p)_{p>0}$ eventually becomes constant and there exists a $p' \in \mathbb{N}^+$ large enough such that $x_p = x^*$ for all $p \geq p'$. In other words, in the case of unique weights and $p \geq p'$ large enough, the solution x^* of the Mutex Watershed Algorithm 7 solves the problem (2.36), which is just a rescaled version of the Mutex Watershed Objective we introduced in Sec. 2.4.2.

To summarize, we used the extended Power Watershed framework to show that the Mutex Watershed provides a solution to the minimization problem in Eq. (2.36) for p large enough. In particular, this problem suggested by the Power Watershed framework is the same one previously derived in Sec. 2.4.2 by linking the Mutex Watershed Algorithm to the multicut optimization problem.

2.5 Experiments

We evaluate the Mutex Watershed on the challenging task of neuron segmentation in electron microscopy (EM) image volumes. This application is of key interest in connectomics, a field of neuro-science that strives to reconstruct neural wiring digrams spanning complete central nervous systems. The task requires segmentation of neurons from electron microscopy images of neural tissue – 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 need to be preserved in the segmentation to arrive at the correct connectivity graph. While a lot of progress is being made, currently only manual tracing or proof-reading yields sufficient accuracy for correct circuit reconstruction [136].

We validate the Mutex Watershed algorithm on the most popular neural segmentation challenge: ISBI2012 [11]. We estimate the edge weights using a CNN as described in Section 2.5.1 and compare with other entries in the leaderboard as well as with other popular post-processing methods for the same network predictions in Section 2.5.2.

2.5.1 Estimating edge weights with a CNN

The common first step to EM segmentation is to predict which pixels belong to a cell membrane using a CNN. Different post-processing methods are then used to obtain a segmentation, see section 2.2 for an overview of such methods. The CNN can either be trained to predict boundary pixels [20, 36] or undirected affinities [54, 90] which express how likely it is for a pixel to belong to a different cell than its neighbors in the 6-neighborhood. 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, [90] have shown that introduction of long-range affinities is beneficial for the final segmentation even if they are only used to train the network. Building on the work of [90], we train a CNN to predict short- and long-range affinities and then use those directly as weights for the Mutex Watershed algorithm.

We estimate the affinities / edge weights for the neighborhood structure shown in Figure 2.6. To that end, we define local attractive and long-range repulsive edges. When attractive edges are only short-range, the solution will consist of spatially connected segments that cannot comprise "air bridges". This holds true for both (lifted) multicut and for Mutex Watershed. We use a different pattern for in-plane and between-plane edges due to the great anisotropy of the data set. In more detail, we pick a sparse ring of in-plane repulsive edges and additional longer-range in-plane edges which are necessary to split regions reliably (see Figure 2.6a). We also added connections to the indirect neighbors in the lower adjacent slice to ensure correct 3D connectivity (see Figure 2.6b). 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 the stride is not applied to local (attractive) edges, but only to long-range (repulsive) edges. The particular pattern used was selected after inspecting the size of typical regions. The specific pattern is the only one we have tried and was *not* optimized over.

In total, C^+ attractive and C^- repulsive edges are defined for each pixel, resulting in $C^+ + C^-$ 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: $E^+ = \bigcup_{c=1}^{C^+} H_c^+$ for attractive edges, with H^- defined 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 $[0, 1]$.

Network architecture and training

We use the 3D U-Net [34, 132] architecture, as proposed in [54]. Our training targets for attractive / repulsive edges $\overset{*}{w}^{\pm}$ can be derived from a groundtruth label image $\overset{*}{L}$ according to

$$\overset{*}{w}{}^{+}_{e=(i,j)} = \begin{cases} 1, & \text{if } \overset{*}{L}_i = \overset{*}{L}_j \\ 0, & \text{otherwise} \end{cases} \tag{2.37}$$

$$\overset{*}{w}{}^{-}_{e=(i,j)} = \begin{cases} 0, & \text{if } \overset{*}{L}_i = \overset{*}{L}_j \\ 1, & \text{otherwise} \end{cases} \tag{2.38}$$

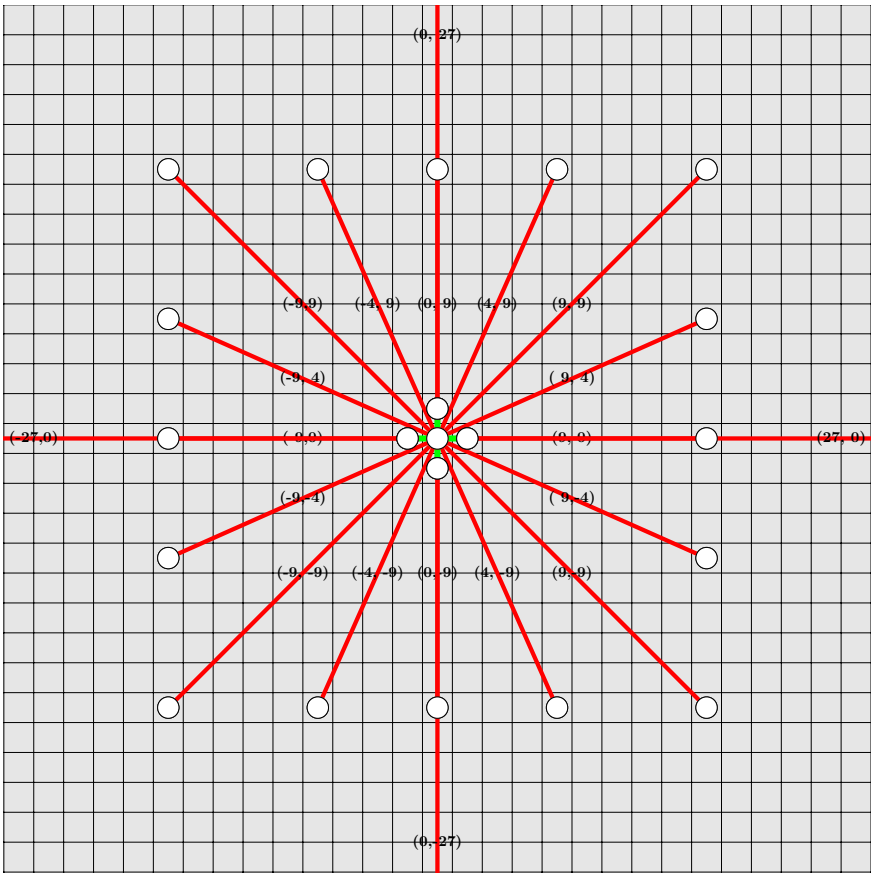
Here, i and j are the indices of vertices / image pixels. Next, we define the loss terms:

$$\mathcal{J}_c^+ = -\frac{\sum_{e \in H_c^+} (1 - w_e^+)(1 - \overset{*}{w}{}_e^+)}{\sum_{e \in H_c^+} ((1 - w_e^+)^2 + (1 - \overset{*}{w}{}_e^+)^2)} \tag{2.39}$$

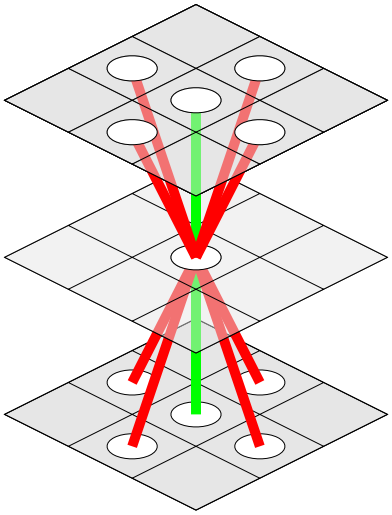
$$\mathcal{J}_c^- = -\frac{\sum_{e \in H_c^-} w_e^- \overset{*}{w}{}_e^-}{\sum_{e \in H_c^-} ((w_e^-)^2 + (\overset{*}{w}{}_e^-)^2)} \tag{2.40}$$

for attractive edges (i.e. channels) and repulsive edges (i.e. channels).

Equation 2.39 is the Sørensen-Dice coefficient [47, 143] formulated for fuzzy set membership values. 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 locally two-dimensional and thin, they occupy very few pixels in three-dimensional the volume. More precisely, if w_e^+ or $\overset{*}{w}{}_e^+$ (or both) are sparse, we can expect the denominator $\sum_e ((w_e^+)^2 + (\overset{*}{w}{}_e^+)^2)$ to be small, which has the effect that the numerator is adaptively weighted higher. In this



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



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

Figure 2.6. Local neighborhood structure of attractive (green) and repulsive (red) edges in the Mutex Watershed graph.

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 [75] and additionally condition learning rate decay on the Adapted Rand Score [11] 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.

2.5.2 ISBI Challenge

The ISBI 2012 EM Segmentation Challenge [11] 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 and has 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, but not 3D segmentation. To train a 3D model, we follow the procedure described in [20].

The test volume has private groundtruth; results can be submitted to the leaderboard. They are evaluated based on the Adapted Rand Score (Rand-Score) and the Variation of Information Score (VI-Score) [11].

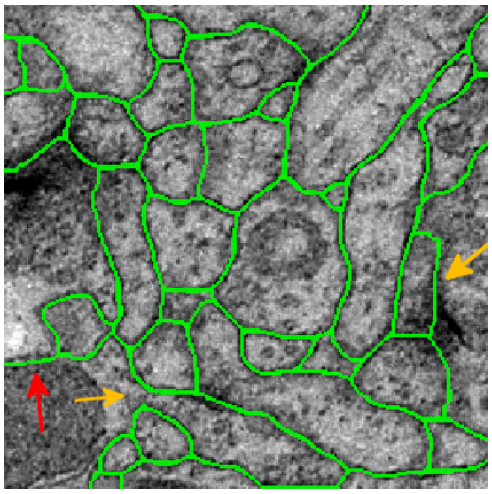
Our method holds the top entry in the challenge's leader board[3] at the time of submission, see Table 2.1. This is especially remarkable insofar as it is simpler than the methods holding the other top entries. Three out of four rely on a CNN to predict boundary locations and postprocess its output with the complex pipeline described in [20]. This post-processing first generates superpixels via distance transform watersheds. Then it computes a merge cost for local and long-range connections between superpixels. Based on this, it defines a lifted multicut partioning problem that is solved approximately. In contrast, our method finds an optimal solution of its objective purely on the pixel level.

Comparison with other segmentation methods

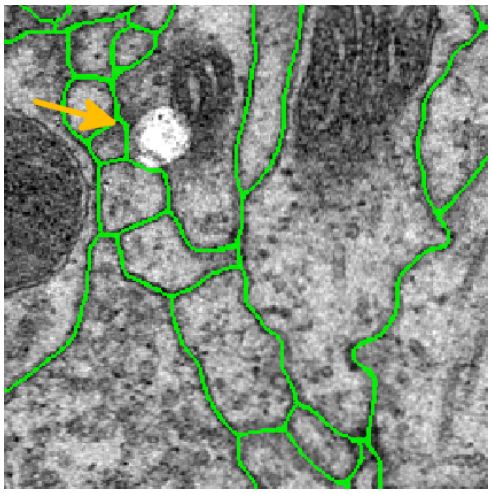
The weights predicted by the CNN described above can be post-processed directly by the Mutex Watershed algorithm. To ensure a fair comparison, we transform the same CNN predictions into a segmentation using basic and state-of-the-art post-processing methods. We start from simple thresholding (THRESH) and seeded watershed. Since these cannot take long-range repulsions into account, we generate a boundary map by taking the maximum[4] values over the attractive 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 optimized 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 2.2. The methods were applied only in 2D, because the high degree of anisotropy leads to inferior results when applied in 3D. In

[3] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new

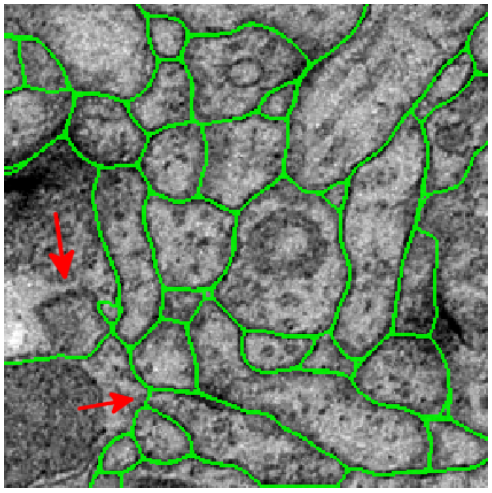
[4] The maximum is chosen to preserve boundaries.



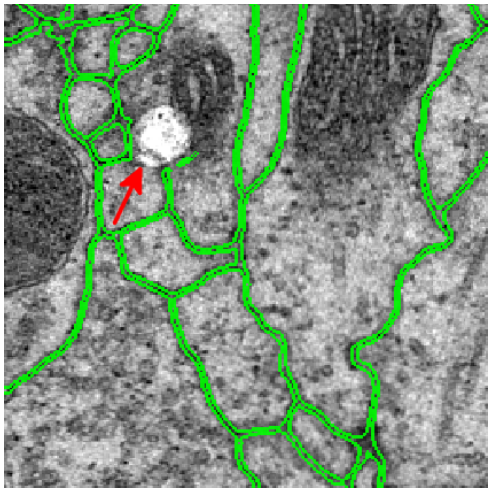
(a) Mutex Watershed



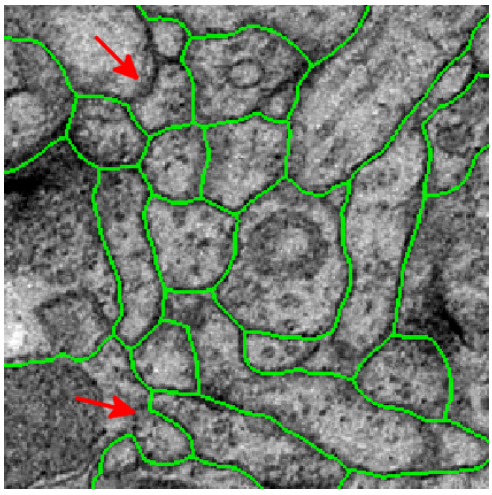
(b) Mutex Watershed



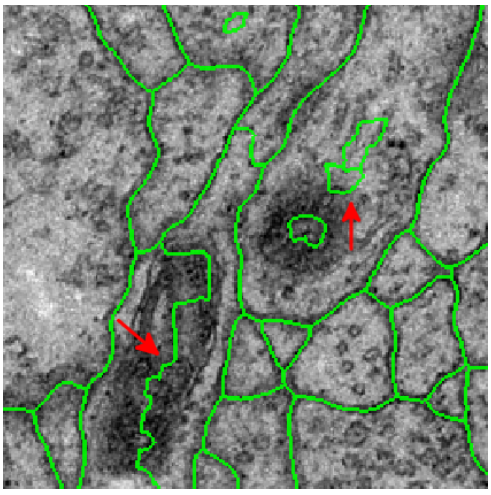
(c) Multicut partitioning based segmentation (MC-FULL)



(d) Thresholding of local boundary maps (THRESH)



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



(f) Distance Transform Watershed (WSDT)

Figure 2.7. 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.

Method	Rand-Score	VI-Score
UNet + MWS	**0.98792**	**0.99183**
ResNet + LMC [157]	0.98788	0.99072
SCN + LMC [151]	0.98680	0.99144
M2FCN-MFA [139]	0.98383	0.98981
FusionNet + LMC [126]	0.98365	0.99130

Table 2.1. Top five entries at time of submission on the ISBI 2012 EM Segmentation Challenge. Our Mutex Watershed (MWS) is state-of-the-art without relying on the complex lifted multicut postprocessing used by most other top entries.

Method	Rand-Score	VI-Score	Time [s]
MWS	**0.98792**	**0.99183**	43.3
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

Table 2.2. Comparison on the ISBI 2012 EM Segmentation Challenge to other segmentation strategies, all of which are based on our CNN. Runtimes were measured on a single thread of a Intel Xeon CPU E5-2650 v3 @ 2.30GHz.

contrast, the Mutex Watershed can be applied in 3D out of the box and yields significantly better 2D segmentation scores.

Qualitatively, we show patches of results in Figure 2.7. The major failure case for WS (Figure 2.7e) and WSDT (Figure 2.7f) is over-segmentation caused by over-seeding a region. The major failure case for THRESH is under-segmentation due to week boundary evidence (see Figure 2.7d). In contrast, the Mutex Watershed produces a better segmentation, only causing minor over-segmentation (see Figure 2.7a, Figure 2.7b).

Note that, in contrast to most pixel-based postprocessing 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 [8] and [62]. As proposed in [7], we compute costs corresponding to edge cuts from the affinities estimated by the CNN via:

$$s_e = \begin{cases} \log \frac{w_e^+}{1-w_e^+}, & \text{if } e \in E^+ \\ \log \frac{1-w_e^-}{w_e^-}, & \text{otherwise,} \end{cases} \tag{2.41}$$

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 bridges", 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 [71, 72] 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 2.2. 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. Somewhat surprisingly, the Mutex Watershed yields a better segmentation still, despite being much cheaper in inference. We note that both MC-FULL, LMC and the Mutex Watershed are evaluated on the same long-range affinity maps (i.e. generated by the same CNN with the same set of weights).

2.6 Conclusion

We have presented a fast algorithm for the clustering of graphs with both attractive and repulsive edges. The ability to consider both gives a valid alternative to other popular graph partitioning algorithms that rely on a stopping criterion or seeds. The proposed method has low computational complexity in imitation of its close relative, Kruskal's algorithm. We have shown which objective this algorithm optimizes exactly, and that this objective emerges as a specific case of the multicut objective. It is possible that recent interesting work [88] on partial optimal solutions may open an avenue for an alternative proof. Finally, we have found that the proposed algorithm, when presented with informative edge costs from a good neural network, outperforms all known methods on a competitive bioimage partitioning benchmark, including methods that operate on the very same network predictions.

Chapter 3

GASP: Generalized Agglomerative Algorithm for Signed Graph Partitioning

In this chapter, we propose a theoretical framework that generalizes simple and fast algorithms for hierarchical agglomerative clustering to weighted graphs with both attractive and repulsive interactions between the nodes. This framework defines GASP, a Generalized Algorithm for Signed graph Partitioning, and allows us to explore many combinations of different linkage criteria and cannot-link constraints. We prove the equivalence of existing clustering methods to some of those combinations and introduce new algorithms for combinations that have not been studied before. We study both theoretical and empirical properties of these combinations and prove that some of these define an ultrametric on the graph. We conduct a systematic comparison of various instantiations of GASP on a large variety of both synthetic and existing signed clustering problems, in terms of accuracy but also efficiency and robustness to noise. Lastly, we show that some of the algorithms included in our framework, when combined with the predictions from a CNN model, result in a simple bottom-up instance segmentation pipeline. Going all the way from pixels to final segments with a simple procedure, we achieve state-of-the-art accuracy on the CREMI 2016 EM segmentation benchmark without requiring domain-specific superpixels.

3.1 Introduction

In computer vision, the partitioning of weighted graphs has been successfully applied to tasks as diverse as image segmentation, object tracking and pose estimation. Most graph clustering methods work with positive edge weights only, which can be interpreted as similarities or distances between the nodes. These methods require users to specify the desired numbers of clusters (as in spectral clustering) or a termination criterion (e.g. in iterated normalized cuts) or even to add a seed for each object (e.g. seeded watershed or random walker).

Other graph clustering methods work with so-called *signed graphs*, which feature both positive and negative edge weights corresponding to attraction and repulsion between nodes. The advantage of signed graphs over unsigned graphs is that balancing attraction and repulsion allows us to obtain a clustering without defining additional parameters. A canonical formulation of the signed graph partitioning problem is the *multicut* or *correlation clustering* problem [32, 68]. This problem is NP-hard, though many approximate solvers have been proposed [18, 87, 117, 161] together with greedy agglomerative clustering

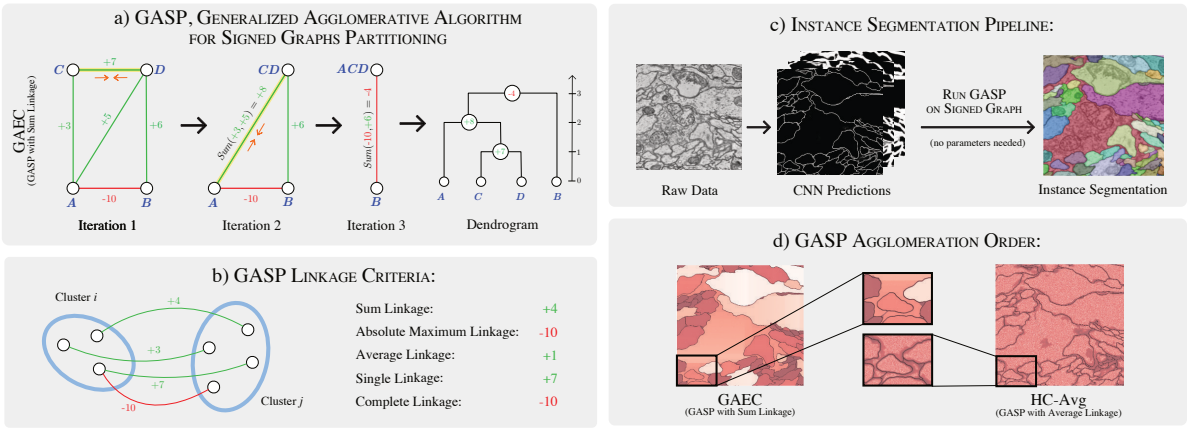


Figure 3.1. **(a)** Some iterations of GASP on a graph with attractive (green) and repulsive (red) interactions. At each iteration, the yellow edge with highest weight is contracted (example with sum linkage criterion is shown). **(b)** Linkage criteria demonstrated on two small clusters (see definitions in Table 3.1 below). **(c)** Application of GASP to instance segmentation: we show raw data from the CREMI neuron-segmentation challenge and some predictions of our CNN model, where white pixels represent boundary evidence. **(d)** Seemingly similar linkage criteria can result in very different clustering dynamics, as shown in this example: color coded sequence of merges from early (white) via late (brown) to never (black).

algorithms [69, 72, 92, 154]. Agglomerative clustering algorithms for signed graphs have clear advantages: they are parameter-free and efficient. Despite the fact that a variety of these algorithms exist, no overarching study has so far been conducted to compare their robustness and efficiency or to provide guidelines for matching an algorithm to the partitioning problem at hand.

The first contribution presented in this chapter is a simple theoretical framework that generalizes over agglomerative algorithms for signed graphs by linking them to hierarchical clustering (HC) on unsigned graphs (Section 3.3.2). This framework defines an underlying basic algorithm and allows us to explore its combinations with different linkage criteria and *cannot-link constraints* (see Fig. 3.1a, 3.1b, and Table 3.1). As second contribution, in Section 3.3.3, we formally prove that some of the combinations correspond to existing clustering algorithms, and introduce new algorithms for combinations which have not been explored before. By analyzing their theoretical properties, we also show that some of them define an ultrametric on the graph (see Table 3.1).

Third, we evaluate the algorithms on a large variety of both existing and synthetically generated signed graph clustering problems (Section 3.4). Fourth and finally, we also test the algorithms on *instance segmentation* – a computer vision task consisting of assigning each pixel of an image to an object instance – by partitioning graphs whose edge weights are estimated by a CNN (see Fig. 3.1c and Section 3.4.3). Our experiments show that the choice of linkage criterion markedly influences how clusters are grown by the agglomerative algorithms (Fig. 3.1d), making some linkage methods more suited for certain types of clustering problems. We benchmark the clustering algorithms by focusing on their efficiency, robustness and tendency to over- or under-cluster. On instance segmentation, we show that the tested agglomerative algorithms strongly outperform recently proposed spectral clustering methods, and that average-linkage based agglomerative algorithms

GASP	Sum Linkage $\sum_{e \in E_{ij}} w_e$	Absolute Maximum Linkage w_e with $e = \arg\max_{t \in E_{ij}} \lvert w_t \rvert$	Average Linkage $\sum_{e \in E_{ij}} w_e \Big/ \lvert E_{ij} \rvert$	Single Linkage $\max_{e \in E_{ij}} w_e$	Complete Linkage $\min_{e \in E_{ij}} w_e$
Unsigned graphs	-	**HC-Single**	**HC-Avg**	**HC-Single**	**HC-Complete**
Signed graphs	GAEC [72]	**Mutex Watershed** [154]	**HC-Avg**	**HC-Single**	**HC-Complete**
Signed graphs + cannot-link-constr	HCC-Sum	**Mutex Watershed** [154]	HCC-Avg	HCC-Single	**HC-Complete**

Table 3.1. Conceptual contribution: Properties of clustering algorithms included in the proposed GASP framework, given a linkage criterion, a type of graph (signed or unsigned) and the optional use of cannot-link constraints. New constrained hierarchical clustering algorithms (HCC) proposed in this thesis are highlighted in yellow. For algorithms typeset in bold font we prove that they define an ultrametric on the graph (Eq. 3.3). For algorithms in the green box we show that they are weight-shift invariant (Prop. 3.3.2). Notation: $E_{ij} = (S_i \times S_j) \cap E$ denotes the set of edges connecting two clusters $S_i, S_j \subseteq V$.

achieve state of the art results on the CREMI 2016 challenge for neuron segmentation of 3D electron microscopy image volumes of brain tissue.

3.2 Related work

Proposal-free instance segmentation methods adopt a bottom-up approach by directly grouping pixels into instances. In the last years, there has been a growing interest in such methods that do not involve object detection because, in certain types of data, object instances cannot be approximated by bounding boxes [14, 79]. Some use metric learning to predict high-dimensional associative pixel embeddings that map pixels of the same instance close to each other [45, 49, 89, 114] and then retrieve final instances by applying a clustering algorithm [82]. Other recent methods let the model predict the relative coordinates of the instance center [30, 113] or, given a pixel (x, y), they train a model to generate the mask of the instance located at (x, y) [141].

Edge detection also experienced recent progress thanks to deep learning, both on natural images [56,81,97,158] and biological data [35,90,101,137]. In neuron segmentation for connectomics, a field of neuroscience we also address in our experiments, boundaries are converted to final instances with subsequent postprocessing and superpixel-merging: some use a combinatorial framework [20], others use loopy graphs [70, 83] or trees [53, 94, 96, 101, 147] to represent the region merging hierarchy. Flood-filling networks [64] and MaskExtend [101] used a CNN to iteratively grow one region/neuron at the time. A structured learning approach was also proposed in [54, 144].

Agglomerative graph clustering has often been applied to instance segmentation [10, 95, 129, 135] because of its efficiency as compared to other divisive approaches like graph cuts. Novel termination criteria and merging strategies have often been proposed: the agglomeration in [100] deploys fixed sets of merge constraints; the popular graph-based method [51] stops the agglomeration when the merge costs exceed a measure of quality for the current clusters. The optimization approach in [77] performs greedy merge decisions that minimize a certain energy, while other pipelines use classical linkage criteria, e.g. average linkage [90, 97], median [54] or a linkage learned by a random forest

classifier [80, 116].

Clustering of signed graphs has the goal of partitioning a graph with both attractive and repulsive cues. Finding an optimally balanced partitioning has a long history in combinatorial optimization [33,58,59]. NP-hardness of the *correlation clustering* problem was shown in [17], while the connection with graph multicuts was made by [46]. Modern integer linear programming solvers can tackle problems of considerable size [7], but accurate approximations [18,117,161], greedy agglomerative algorithms [69,72,92,153] and persistence criteria [87,88] have been proposed for even larger graphs. Another line of research is given by spectral clustering methods that, on the other hand, require the user to specify the number of clusters in advance. Recently, some of these methods have been generalized to graphs with signed weights [31,41,85], whereas others let the user specify must-link and cannot-link constraints between clusters [42,127,150].

This chapter reformulates the clustering algorithms of [72, 92, 154] in a generalized framework and adopts ideas from the proposal-free instance segmentation methods [90, 97, 154] to predict edge weights of a graph.

3.3 Generalized framework for agglomerative clustering of signed graphs

3.3.1 Notation

Graph formalism We consider an undirected simple edge-weighted graph $\mathcal{G}(V, E, w^+, w^-)$ with both attractive and repulsive edge attributes. The weight function $w^+ : E \to \mathbb{R}^+$ associates to every edge a positive scalar attribute $w_e^+ \in \mathbb{R}^+$ representing a merge affinity or a similarity measure. On the other hand, $w^- : E \to \mathbb{R}^+$ associates to each edge a split tendency $w_e^- \in \mathbb{R}^+$. Graphs of the type $\mathcal{G}(V, E, w^+, w^-)$ are often defined as *signed graphs* $\mathcal{G}(V, E, w)$, featuring positive and negative edge weights $w_e \in \mathbb{R}$. Following the theoretical considerations in [88], we define signed weights as $w_e = w_e^+ - w_e^-$.

Multicut objective We call the set $\Pi = \{S_1, \dots, S_K\}$ a *clustering* or *partitioning* if $V = \cup_{S\in\Pi} S$, $S \cap S' = \emptyset$ for different clusters S, S' and every cluster $S \in \Pi$ induces a connected subgraph of $\mathcal{G}$. For any clustering Π of $\mathcal{G}$, we denote as $E_\Pi^0 = \{e_{uv} \in E \,|\, \exists S \in \Pi : u, v \in S\}$ the set of edges linking nodes in the same cluster. Its complementary set $E_\Pi^1 = E \setminus E_\Pi^0$ of edges linking nodes belonging to distinct clusters, is known as the *multicut* of $\mathcal{G}$ associated to clustering Π. The instance of the NP-hard *minimum cost multicut problem* w.r.t. $\mathcal{G}(V, E, w_e)$ is the task of finding a clustering that optimally balances the attraction and repulsion in the graph and is given by the following binary integer program:

$$\min_{\Pi} \sum_{e\in E} w_e x_e^\Pi, \qquad \text{where} \quad x_e^\Pi = \begin{cases} 1 & \text{if } e \in E_\Pi^1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Linkage criteria and hierarchical trees Let the interaction $\mathcal{W}(S, S') \in \mathbb{R}$ between two clusters S, S' be defined as a function, named *linkage criterion*, depending on the weights of *all* edges connecting clusters S and S', i.e. $(S \times S') \cap E$ where $\times$ denotes the outer product. The linkage criteria tested in this chapter are listed and defined in Table 3.1. A

dendrogram T is a rooted binary tree[1] representing the merging order of an agglomerative algorithm, such that the leaves of the tree are in one-to-one correspondence with V and each node of the tree represents a merge between two clusters. Let $T_{\mathrm{R}}, T_{\mathrm{L}} \subset T$ denote the subtrees rooted at the two children of the root node in T. For any two leaves $u, v \in V$, let $T[u \vee v]$ be the subtree rooted at the least common ancestor $(u \vee v) \in T$ of nodes u and v (furthest from the root), and let $\texttt{leaves}(T[u \vee v]) \subseteq V$ be the set of leaves of this subtree. Given an agglomerative algorithm with merging tree T, let $h_T : V \times V \to \mathbb{N}$ denote the *dendrogram-height* of each $(u \vee v) \in T$, which is defined as the iteration number at which nodes $u, v \in V$ were merged by the algorithm (see example in Fig. 3.1a). We also define $\mathcal{W}_T(u, v)$ as the signed interaction $\mathcal{W}(S, S')$ between the two clusters S, S' that were merged at iteration $h_T(u, v)$:

$$\mathcal{W}_T(u, v) \equiv \mathcal{W}\big(\texttt{leaves}(T_{\mathrm{R}}[u \vee v]), \texttt{leaves}(T_{\mathrm{L}}[u \vee v])\big) \tag{3.2}$$

3.3.2 The GASP algorithm

Our main contribution is a generalized agglomerative algorithm for signed graph partitioning (GASP) that generalizes hierarchical clustering (HC) to signed graphs. The framework, defined in the following, encompasses several known and new agglomerative algorithms on display in Table 3.1, which are differentiated by the linkage criterion employed, similarly to HC.

In Algorithm 8, we provide simplified pseudo-code for the proposed GASP algorithm. GASP implements a bottom-up approach that starts by assigning each node to its own cluster and then iteratively merges pairs of adjacent clusters. The algorithm proceeds in three phases.

In phase one, GASP selects the pair of clusters with the highest absolute interaction $|\mathcal{W}(S, S')|$, so that the most attractive and the most repulsive pairs are analyzed first. If the interaction is repulsive and the algorithm option *addCannotLinkConstraints* is `True`, then the two clusters are constrained so that their members can never merge in subsequent steps of phase one. If the interaction is attractive, then the clusters are merged, provided that they were not previously constrained. After each merge, the interaction between the merged cluster and its neighbors is updated according to one of the linkage criteria $\mathcal{W}(S, S')$ listed in Table 3.1. Phase one terminates when all the remaining clusters are either constrained or share repulsive interactions. Note that, on unsigned graphs, in phase one all nodes are merged into a single cluster and GASP is then equivalent to a standard hierarchical clustering algorithm.

Phase two: Now that the clusters have grown in size, the algorithm removes the constraints previously introduced in phase one and merges all the clusters that still share an attractive interaction, merging the most attractive one first[2]. The final clustering Π^* returned by GASP is found at the end of phase two and it is then composed of clusters sharing only mutual repulsive interactions.

Finally, in phase three, the algorithm keeps merging all clusters until only a single one is left and then returns the hierarchical tree T^* representing the full sequence of merging steps. The algorithm was implemented using a standard HC implementation with computational complexity $\mathcal{O}(N^2 \log N)$ (details left in Appendix A.1).

[1] In general, one could look at trees that are not binary. However, the algorithms discussed in this chapter always generate binary hierarchical trees, so nothing would be gained by this generalization.

[2] Note that in the version of GASP without *cannotLinkConstraints*, nothing happens in phase two because all remaining interactions are repulsive.

Algorithm 8 GASP

Input: Graph $\mathcal{G}(V, E, w^+, w^-)$; linkage criterion $\mathcal{W}$;
boolean addCannotLinkConstraints
Output: Final clustering Π^*, rooted binary hierarchical tree T^*

Initial clustering: $\Pi = \{\{v_1\}, \dots, \{v_{|V|}\}\}$
Initialize hierarchical tree T^* with leaf nodes $V = \{v_1, \dots, v_{|V|}\}$
Initialize cluster interactions with $w_e = w_e^+ - w_e^-$, $\forall e \in E$
// Phase 1: Merge positive interactions (possibly using constraints)
Push incident nodes of every edge $e \in E$ to priority queue (PQ) with priority $|w_e|$
repeat
  Pop $S, S' \in \Pi$ with highest interaction $|\mathcal{W}(S, S')|$ from PQ
  if $[\mathcal{W}(S, S') > 0]$ **and** $[S, S'$ **not** constrained$]$ **then**
    Merge clusters S, S' and update hierarchical tree T^*
    Update interactions & constraints with neighboring clusters
  else if addCannotLinkConstr **and** $[\mathcal{W}(S, S') \leq 0]$ **then**
    Add CannotLink Constraint between S and S'
until $[PQ$ is empty$]$
// Phase 2: Remove constraints & merge all positive interactions
Push signed interactions $\mathcal{W}(S, S')$ to PQ, $\forall S, S' \in \Pi$
repeat
  Pop $S, S' \in \Pi$ with highest interaction $\mathcal{W}(S, S')$ from PQ
  if $[\mathcal{W}(S, S') > 0]$ **then**
    Merge clusters S, S' and update hierarchical tree T^*
    Update interactions with neighboring clusters
until $[\mathcal{W}(S, S') \leq 0]$
Save the final clustering $\Pi^* \leftarrow \Pi$
// Phase 3: Merge negative interactions until one single cluster is left
repeat
  Pop $S, S' \in \Pi$ with highest interaction $\mathcal{W}(S, S')$ from PQ
  Merge clusters S, S' and update hierarchical tree T^*
  Update interactions with neighboring clusters
until [Only one cluster is left in Π]
return Π^*, T^*

3.3.3 GASP: New and existing algorithms

Here, we focus on five linkage methods (see columns of Table 3.1). Many more linkage criteria have been applied to unsigned graphs [51,54,116], involving median-based or size-regularized methods, but we decided to focus on those five criteria because they represent the most popular choices.

Sum Linkage On signed graphs, the sum of two attractive (or repulsive) interactions is still attractive (repulsive). On the other hand, on unsigned graphs, a strong attractive interaction could be obtained by summing many weak interactions, which depending on the application could be undesirable. This explains why, to our knowledge, an agglomerative algorithm with sum linkage has never been used on unsigned graphs. On signed graphs, such an algorithm was pioneered in [72, 92] and was named Greedy Agglomerative Edge Contraction (GAEC)[3]. All variations of GASP decrease the multicut objective defined in Eq. 3.1 each time two clusters with positive interaction are merged. But only

[3] An algorithm equivalent to GAEC was recently independently re-proposed in [29].

GAEC always makes the *greedy choice* that most decreases the multicut objective at each iteration. The authors of [92] propose an algorithm named *GreedyFixation*, which, in our framework, is equivalent to phase one of GASP using cannot-link-constraints and a sum linkage. However, running both phase one and two of GASP with sum linkage (algorithm named HCC-Sum in this chapter) performed better than GreedyFixation in our experiments.

AbsMax Linkage This linkage method is also specific to signed graphs, since on unsigned graphs it would be equivalent to single linkage. Here, we prove that the Mutex Watershed Algorithm [154] introduced in Chapter 2 can be seen as an agglomerative algorithm with AbsMax linkage (proofs of the following three propositions are given in Appendix A.2):

Proposition 3.3.1. *The GASP Algorithm 8 with AbsMax linkage, with or without cannot link constraints, returns the same final clustering* Π^*_{AbsMax} *also returned by the Mutex Watershed Algorithm (MWS) [154] (see Chapter 2), which has empirical complexity* $\mathcal{O}(N \log N)$.

Average, Single, and Complete Linkage These three linkage criteria have been thoroughly studied on unsigned graphs, but never - until very recently - on signed graphs. In concurrent independent related work [29], the authors prove that applying these three linkage methods to a signed graph is equivalent to applying them to the unsigned graph obtained by shifting all edge weights by a constant. Here, we prove which of the algorithms studied here are "intrinsically signed" and do not have this invariance-property:

Proposition 3.3.2. *We call an agglomerative algorithm "weight-shift invariant" if the dendrogram* T *returned by the algorithm is invariant w.r.t. a shift of all edge weights* w_e *by a constant* $\alpha \in \mathbb{R}$. *Among the variations of GASP, only hierarchical clustering with Average (HC-Avg), Single (HC-Single), and Complete linkage (HC-Complete) are weight-shift-invariant (see green box in Table 3.1).*

Although average and single linkage methods have been largely studied on unsigned graphs, to our knowledge, they have never been combined with cannot-link constraints on signed graphs[4], so we name these algorithms *HCC-Avg* and *HCC-Single*.

Algorithms defining an ultrametric The connection between agglomerative algorithms and ultrametrics[5] is well known. Usually, ultrametrics are associated to strictly positive *similarity* or *dissimilarity* measures on a graph. In our framework, a trivial ultrametric is always given by the height h_T of the dendrogram. However, for some of the GASP variations, we now define an ultrametric based on the edge weights and the signed interactions between clusters, generalizing what has been done for HC on unsigned graphs [66,107]. To define this measure and prove its ultrametric property, we first map the signed interaction

[4]Note that Complete linkage methods return the same clustering whether constraints are enforced or not (proof in Lemma A.2.3, in Appendix).

[5]A metric space (X, d) is an *ultrametric* if, for every $x, y, z \in X$, $d(x, y) \leq \max\{d(x, z), d(y, z)\}$.

$\mathcal{W}_T$ defined in Eq. 3.2 to positive "pseudo-distances" $d_T : V \times V \to \mathbb{R}^+$:

$$d_T(u,v) \equiv \begin{cases} 0 & \text{if } u = v \\ M - \mathcal{W}_T(u,v) & \text{if } u \neq v \end{cases} \quad \forall u, v \in V \tag{3.3}$$

$$\text{where} \quad M \equiv \epsilon + \max_{u',v' \in V,\, u' \neq v'} \mathcal{W}_T(u',v') \tag{3.4}$$

and where $\epsilon > 0$. We then prove the following proposition:

Proposition 3.3.3. *Among the algorithms included in the GASP framework (see Table 3.1), only Mutex Watershed and hierarchical clustering with Average (HC-Avg), Single (HC-Single) and Complete linkage (HC-Complete) define an ultrametric (V, d_{T^*}), where d_{T^*} is defined in Eq. 3.3 and T^* is the tree returned by the GASP Algorithm 8.*

In summary, in this section we have extended the family of HC algorithms [66, 107] with "weight-based ultrametrics" to signed graphs. Next, we move to their empirical evaluation.

3.4 Experiments

3.4.1 Signed graph clustering problems

We evaluate the agglomerative clustering algorithms included in our framework on a large collection of both synthetic and real-world graphs with very different structures. The size of the graphs ranges from a few hundred to hundreds of millions of edges.

Synthetic SSBM graphs We first consider synthetic graphs generated by a signed stochastic block model (SSBM). We use an Erdős-Rényi random graph model $\mathcal{G}(N,p)$ with N vertices and edge probability p. Following the approach in [41], we partitioned the graph into k *ground-truth* clusters, such that edges connecting vertices belonging to the same cluster (different clusters, respectively) have Gaussian distributed edge weights centered at $\mu = 1$ ($\mu = -1$, respectively) and with standard deviation $\sigma = 0.1$. To model noise, the sign of the edge weights is flipped independently with probability η.

Existing signed graphs We use clustering instances from the OpenGM benchmark [67] as well as biomedical segmentation instances [117]. The dataset *Image Segmentation* contains planar region-adjacency-graphs (RAG) that are constructed from superpixel

Clustering problem	Graph Type	$\#I$	$\lvert V\rvert$	$\lvert E\rvert$
Modularity Clustering [25]	*complete*	6	34-115	561-6555
Image Segmentation [5]	*RAG*	100	156-3764	439-10970
Knott-3D (150-300-450) [7]	*3D-RAG*	24	572-17k	3381-107k
CREMI-3D-RAG (OurCNN)	*3D-RAG*	3	134k-157k	928k-1065k
Fruit-Fly Level 1-4 [117]	*3D-RAG*	4	5m-11m	28m-72m
CREMI-gridGraph (OurCNN)	*gridGraph*	15	39m	140m
Fruit-Fly Level Global [117]	*3D-RAG*	1	90m	650m

Table 3.2. List of compared signed graph clustering problems: for each, we specify the number of instances $\#I$, number of nodes $|V|$, and number of edges $|E|$ per instance.

adjacencies of photographs. The *Knott-3D* datasets contains 3D-RAGs arising from volume images acquired by electron microscopy (EM). The set *Modularity Clustering* contains complete graphs constructed from clustering problems on small social networks. The *Fruit-Fly* 3D-RAG instances were generated from volume image scans of fruit fly brain matter. Instances *Level 1-4* are progressively simplified versions of the global problem obtained via block-wise domain decomposition [117].

Grid-graphs from CNN predictions We also evaluate the clustering methods on the task of neuron segmentation in EM image volumes using training data from the CREMI 2016 EM Segmentation Challenge [40]. We train a 3D U-Net [34, 131] using the same architecture as [54] and predict long-and-short range affinities as described in [90]. The predicted affinities $a_e \in [0, 1]$, which represent how likely it is for a pair of pixels to belong to the same neuron segment, are then mapped to signed edge weights $w_e = a_e - 0.5$, resulting in a 3D grid-graph having a node for each pixel/voxel of the image[6]. We divided the three CREMI training samples, consisting of ∼196 million voxels each, into five sub-blocks for a total of 15 clustering problems (named *CREMI-gridGraph* in Table 3.2). See the following Section 3.4.2 for extended details about training, data augmentation, and how we remove tiny clusters left after running GASP on the *CREMI-gridGraph* clustering problems.

3D-RAG from CNN-predictions Lastly, we use the predictions of our CNN model to generate three graph instances (one for each CREMI training sample, named *CREMI-3D-RAG* in Table 3.2), which have very similar structure to the *Knott-3D* and *Fruit-Fly* instances. We obtain these problems by using a pipeline that is very common in neuron segmentation: a watershed algorithm generates superpixels and from those a 3D region-adjacency graph is built, where edge weights are given by the CNN predictions averaged over the boundaries of adjacent superpixels (details in the following Section 3.4.2).

3.4.2 Details on neuron segmentation graph instances

Training and data augmentation The data from the CREMI challenge is highly anisotropic and contains artifacts like missing sections, staining precipitations and support film folds. 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 [133]. In addition to the standard data augmentation techniques of random rotations, random flips and elastic deformations, we simulate data artifacts. We randomly zero-out slices, decrease the contrast of slices, simulate tears, introduce alignment jitter and paste artifacts extracted from the training data. Both [54] and [90] have shown that these kinds of augmentations can help to alleviate issues caused by EM-imaging artifacts. We use L2 loss and Adam optimizer to train the network. The model was trained on all three samples with available ground truth labels.

CREMI-gridRag instances Our 3D UNet model predicts the same set of 12 long-and-short range affinities as described in [90]. When building the pixel-grid graph, we add both direct neighbors connections and the long-range connections predicted by our model

[6] To map affinities to signed weights, we also tested the *logarithmic mapping* proposed in [7, 52], but it performed worse in our experiments.

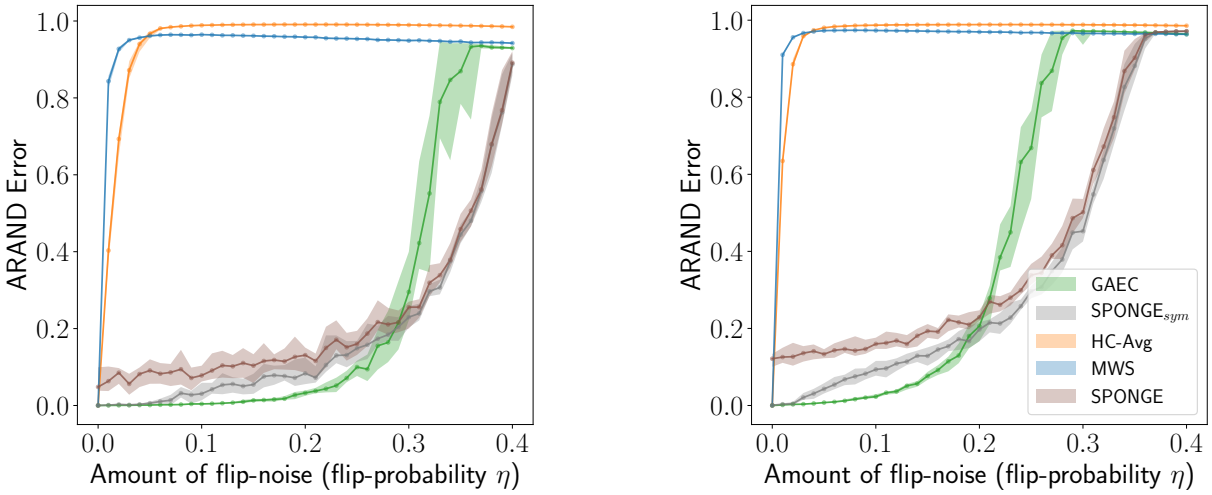
(every voxel is connected to other six voxels via direct connections and other 18 voxels via long-range edges). Empirically, when long-range predictions of the CNN are added as long-range connections in the graph, GASP achieves better scores as compared to when only direct-neighbors predictions are used. Our intuitive explanation of this is that, where there is a clear boundary evidence between two segments, the long-range predictions of the CNN model are more certain than the direct-neighbor ones, because it is often impossible to estimate the exact ground-truth label transition for pixels that are very close to a boundary evidence. However, empirically, we also find that GASP achieves the best scores when only 10% of the long-range connections are randomly sampled and added to the grid-graph. When all the long-range connections predicted by the CNN are added to the graph (18 connections for every voxel), all versions of GASP tend to perform more over-clustering errors. In practice, we explain this by observing that many challenging parts of the studied neuron segmentation data involve thin and elongated segments, and our model sometimes fails to connect distant pairs of pixels that, according to the ground-truth labels, should belong to the same segment (even though, in this case, the direct neighboring predictions are correct). To sum up, the scores we report in Tables 3.5a are obtained by using only 10% of the long-range predictions, since this was the setup that performed the best. After running GASP, we use a simple post-processing step to delete small segments on the boundaries, most of which are given by single-voxel clusters. On the neuron segmentation predictions, we deleted all regions with less than 200 voxels and used a seeded watershed algorithm to expand the bigger segments.

CREMI-3D-rag instances We build these clustering problems by generating superpixels and then building a 3D region adjacency graph. Due to the anisotropy of the data, we generate 2D superpixels by considering each 2D image in the stack singularly. First, we generate a boundary-evidence map by taking an average over the two direct-neighbor predictions of the CNN model (one for each direction in the 2D image of the stack) and applying some additional smoothing. Then, we threshold the boundary map, compute a distance transform, and run a watershed algorithm seeded at the maxima of the distance transform (WSDT). The degree of smoothing was optimized such that each region receives as few seeds as possible, without however causing severe under-segmentation. The computed 2D superpixels are then used to build a 3D region-adjacency graph (3D-rag). The weights of the edges are given by averaging the CNN affinities over the boundaries of adjacent superpixels.

3.4.3 Comparison of results and discussion

Multicut objective values In Table 3.3, we report the values of the multicut objective obtained for clustering with different GASP algorithms[7]. Although many heuristics were proposed to better optimize this objective [18, 19, 71], these methods are out of the scope of this work, since they do not scale to the largest graph instances considered here. By looking at results in Table 3.3, we observe that GAEC almost always achieves the lowest objective values, expect in the *CREMI-gridGraph* instances. Despite this, on graphs where a ground truth clustering is known, GAEC does not achieve the lowest ARAND errors (see Tables 3.5a and 3.5b).

[7]Objective values achieved by Single and Complete linkage methods are much worse compared to other algorithms and are reported in Table **??**, in Appendix.



(a) $k = 20$, $p = 0.1$ **(b)** $k = 50$, $p = 0.2$

Figure 3.2. ARAND errors (median values over 20 experiments, lower is better) on synthetic graphs generated with SSBM. We consider k ground truth communities of random size. Graphs have $N = 10000$ nodes and edges are randomly added with probability p.

Size of growing clusters: Sum vs Avg linkage In all the studied clustering problems, we empirically observe that sum-linkage algorithms like GAEC grow clusters one after the other, as shown in Fig. 3.1d and Fig. 3.3 by the agglomeration order of GAEC[8]. This is intuitively explained by the following: initially, many of the most attractive edge weights have very similar values; when the two nodes u, v with the highest attraction are merged, there is a high chance that they will have a common neighboring node t belonging to the same cluster; thus, the interaction between the merged nodes uv and t is likely assigned to the highest priority, because it is given by the sum of two highly attractive edge weights. This will then start a "chain reaction" where only a single cluster is agglomerated at the time. In the following, we will show how this unique *flooding strategy* of the sum-linkage methods can be both an advantage or a disadvantage, depending on the type of clustering problem.

Comparison to spectral clustering The spectral clustering methods for signed graphs SPONGE$_{sym}$ and SPONGE proposed by [41] achieved state of the art performances on SSBM synthetic graphs. Their competitive performances are also confirmed by our experiments in Fig. 3.2. However, these methods do not scale up to the large graph instances considered here and they also require the user to specify the true number of clusters in advance, which is not known for other graph instances tested in this chapter. In Table 3.4, we report the scores achieved by these methods on a much smaller sub-instance of the *CREMI-gridGraph* problem: even when the true number of clusters is specified in advance for the spectral methods, they perform much worse than other GASP algorithms, with an accuracy penalty of almost 50%. For these reasons, we exclude them from our other comparison experiments.

[8]This *flooding agglomeration-strategy* of GAEC was also observed in [69].

Clustering problem	Multicut objective values (average across instances, lower is better) GAEC [72]	HCC-Sum	MWS [154]	HC-Avg	HCC-Avg	HC-Single	HC-Complete
Modularity Clustering	-0.457	-0.453	-0.073	**-0.467**	**-0.467**	0.000	-0.201
Image Segmentation	**-2,955**	-2,953	-2,901	-2,903	-2,896	-1,384	-2,102
Knott-3D (150-300-450)	**-36,667**	-36,652	-35,200	-35,957	-35,631	-2,522	30,629
CREMI-3D-rag	**-1,112,287**	-1,112,286	-1,109,731	-1,112,177	-1,112,100	-1,038,709	-748,734,869
Fruit-Fly Level 1-4	**-151,022**	-151,017	-150,879	-150,909	-150,876	-71,477	-128,733
CREMI-gridGraph	-73,317,601	-73,328,867	-73,330,568	**-73,502,947**	-73,474,856	-45,194,180	311,598,700
Fruit-Fly Level Global	**-151,688**	-151,596	-146,315	-150,466	-150,171	-4,422	6,876

Table 3.3. We compare algorithms in the GASP framework by their value of the multicut objective defined in Eq. 3.1 (lower is better).

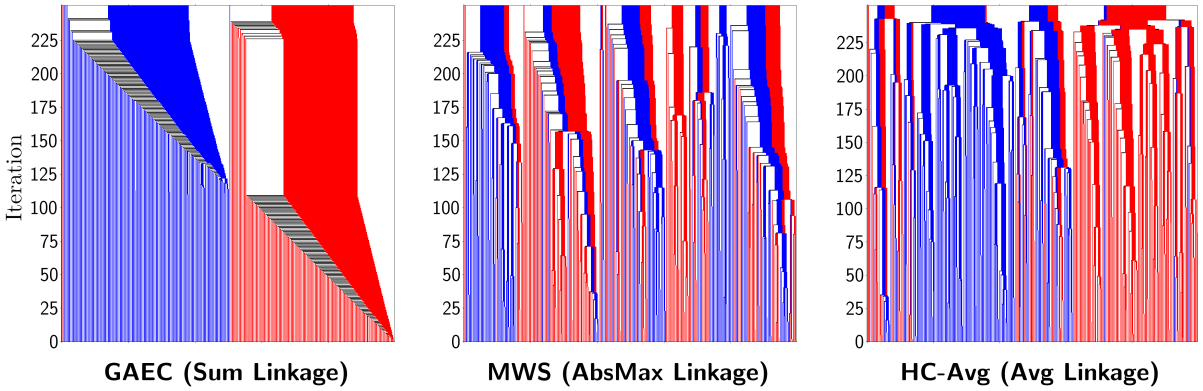


Figure 3.3. Clustering dynamics and accuracy of GASP variations on stochastic block models. The dendrograms result from three versions of GASP on a synthetic graph generated with SSBM (250 nodes, edge probability $p = 0.05$, flipping probability $\eta = 0.1$). Red and blue colors show which of the two equal-sized ground-truth communities each node belongs to. At the top, dendrograms are truncated at the level of the final clustering Π^* returned by GASP.

GASP on synthetic SSBM graphs

GASP algorithms using cannotLinkConstraints are not expected to perform well on these graphs, because of the type of employed sign noise, so we focus our comparison only on the GAEC, HC-Avg and MWS algorithms (using Sum, Average, and AbsMax linkage methods, respectively). Empirically, we observe that GAEC is the agglomerative algorithm performing best on SSBM graphs, on par with spectral method SPONGE$_{sym}$ (see Fig. 3.2). Given the simple properties of SSBM graphs, we can now give a detailed explanation of these empirical results. In SSBM graphs, the number of edges $E_{ij} \equiv (S_i \times S_j) \cap E$ connecting two clusters S_i, S_j is proportional to the product $|S_i| \cdot |S_j|$ of cluster sizes. With Sum or Avg linkage methods, due to the law of large numbers, the flipping noise is "averaged out" as soon as the set E_{ij} becomes larger and clusters grow in size. On the other hand, when clusters are small, it can happen that, for few clusters, several of their edges in E_{ij} are flipped and the algorithm makes a mistake by merging two clusters belonging to different ground truth communities. From this observation, it follows that the *flooding strategy* of the sum-linkage algorithm GAEC is a very good strategy on these types of graphs, because clusters are immediately grown in size (see dendrograms in Fig. 3.3). Average linkage method HC-Avg instead performs much worse on these graphs because it grows small equally-sized clusters and makes several wrong merge-decisions at the beginning. Lastly, the MWS algorithm is not expected to perform well on these graphs because of the high sensitivity of the AbsMax linkage to flipping noise. In the following Proposition 3.4.1, we prove that, at every iteration, the MWS algorithm makes a mistake

Method	ARAND Error
HC-Avg (GASP with Avg Linkage)	**0.1034**
GAEC [72] (GASP with Sum Linkage)	0.1035
MWS [154] (GASP with AbsMax linkage)	0.1068
SPONGE$_{sym}$ [41]	0.4161
L_{sym} [85]	0.8069
SPONGE [41]	0.9211
BNC [31]	0.9926

Table 3.4. GASP compared to spectral clustering methods on a small crop of the CREMI neuron segmentation dataset. Since spectral methods cannot scale to the full CREMI dataset, we evaluated them on a smaller $10 \times 100 \times 100$ sub-volume of CREMI training sample B. Despite the fact that the true number of ground truth clusters was given as an input to the spectral methods, GASP significantly outperformed them.

with at least probability η, independently on the sizes of the two clusters that are popped from priority queue. In summary, for the SSBM, we can obtain a deep understanding of the dynamics induced by various linkage criteria, and find that GAEC gives highest accuracy by a large margin.

Proposition 3.4.1. *Consider a graph generated by an Erdős-Rényi signed stochastic block model (SSBM) as described in Section 3.4.1, with N nodes, edges added with probability p, sign-flip probability $\eta < 0.5$, k ground-truth clusters, and edge weights Gaussian-distributed with standard deviation σ. Then, at every iteration, GASP with Absolute Maximum linkage (or, in other words, the Mutex Watershed algorithm) always makes a mistake with at least probability η.*

Proof. Thanks to Lemma A.2.3 we know that GASP with Absolute Maximum linkage returns the same clustering whether or not cannot-link-constraints are used. Thus, in the following, we prove the proposition considering the version enforcing constraints. Let us consider a generic iteration of the algorithm, where two clusters S_α and S_β have the highest priority and are popped from priority queue. Then, the MWS algorithm will either merge or constrain them depending on the fact that their interaction $\mathcal{W}_{\text{AbsMax}}(S_\alpha, S_\beta)$ is positive or negative (note that, with AbsMax linkage, an interaction can never be positive and constrained, as shown in Lemma A.2.3). By construction of the SSBM, every edge $e \in E$ in the graph has a absolute weight distributed as $|w_e| \sim \mathcal{N}(1, \sigma^2)$. Thus, every edge $e' \in (S_\alpha \times S_\beta) \cap E$ connecting the two clusters has the same probability to have the highest absolute weight, and the sign of the interaction $\mathcal{W}_{\text{AbsMax}}(S_\alpha, S_\beta)$ will only depend on the sign of this highest edge. Therefore, the probability that the MWS merges two clusters is simply given by the fraction of positive weighted edges connecting them.

Let $\tilde{\Pi} = \{\tilde{S}_1, \dots, \tilde{S}_k\}$ denote the ground truth clustering, and $\tilde{S}_{\alpha i} = S_\alpha \cap \tilde{S}_i$ denote the intersection between cluster S_α and a ground-truth cluster $\tilde{S}_i$. If the generated graph is dense, i.e. $p = 1$, then the total number of edges connecting clusters S_α and S_β that have a true attractive or repulsive weight is (according to the ground truth labels)

$$\Gamma^+ = \sum_{i=1}^{k} |\tilde{S}_{\alpha i}||\tilde{S}_{\beta i}|, \qquad \Gamma^- = \sum_{i=1}^{k} \sum_{j=1, j\neq i}^{k} |\tilde{S}_{\alpha i}||\tilde{S}_{\beta j}|. \tag{3.5}$$

When the edges in the graph are randomly added with a probability p, then the actual number of true attractive and repulsive interactions connecting the two clusters is

(according to the ground truth labels):

$$\gamma^+ \sim \mathcal{B}(\Gamma^+, p), \qquad \gamma^- \sim \mathcal{B}(\Gamma^-, p), \tag{3.6}$$

where $\mathcal{B}(\Gamma, p)$ is the binomial distribution:

$$\mathcal{B}(\gamma; \Gamma, p) = \frac{\Gamma!}{\gamma!(\Gamma-\gamma)!} p^\gamma (1-p)^{\Gamma-\gamma}. \tag{3.7}$$

Here, we only assume that $\gamma^+ + \gamma^- > 0$, i.e. there is at least one edge connecting the two clusters (otherwise their interaction would be zero and the MWS would not have popped them from priority queue).

So far we have been talking about attractive and repulsive connections according to the ground truth labels. In our SSBM however every edge has a uniform probability η to have its sign flip, so the actual number of attractive interactions connecting the two clusters will be instead given by the sum of the true attractive interactions $\gamma^+_{\text{nf}} \sim \mathcal{B}(\gamma^+, 1-\eta)$ that have not been flipped, plus the true negative interactions $\gamma^-_{\text{f}} \sim \mathcal{B}(\gamma^-, \eta)$ that have been flipped. Putting everything together, given two clusters with γ^+ true attractive interactions and γ^- true negative ones, the highest-absolute-weight edge connecting them has the following probability to be positive:

$$\begin{aligned}\mathbb{P}[\mathcal{W}_{\text{AbsMax}}(S_\alpha, S_\beta) > 0; \gamma^+, \gamma^-] &= \sum_{\gamma^+_{\text{nf}}=0}^{\gamma^+} \sum_{\gamma^-_{\text{f}}=0}^{\gamma^-} \mathcal{B}(\gamma^-_{\text{f}}; \gamma^-, \eta)\mathcal{B}(\gamma^+_{\text{nf}}; \gamma^+, 1-\eta) \cdot \left(\frac{\gamma^+_{\text{nf}} + \gamma^-_{\text{f}}}{\gamma^+ + \gamma^-}\right) \\ &\overset{(*)}{=} \frac{\gamma^+(1-\eta) + \gamma^-\eta}{\gamma^+ + \gamma^-}\end{aligned} \tag{3.8}$$

where in $(*)$ we used the fact that the expected value of a binomial distribution $\mathcal{B}(\gamma, \eta)$ is $\gamma\eta$.

Now we note that this probability is bounded in the interval $[\eta, 1-\eta]$. So, regardless of whether the two clusters S_α and S_β should be merged or constraint according to ground truth labels, the probability not to make the correct decision is always at least η. Remarkably, while the exact probability in Eq. 3.8 depends on the number of edges connecting the two clusters $\gamma^+ + \gamma^-$ and thus on the cluster sizes, the bounds do not. Thus, this result shows that, unlike Sum or Avg linkage methods, the MWS algorithm is unable to reliably correct for the sign flip noise even for big clusters linked by many edges. □

GASP on neuron segmentation graph instances

SSBM graphs are *non-planar*, and every edge has the same probability to be present in the graph. On the other hand, the *gridGraph* and *3D-RAG* graphs of Table 3.2 are sparse and have a very regular structure: regardless of whether a node represents a pixel or a superpixel, it will only have edge connections with its neighbors in the image (up to a certain hop distance). Tables 3.5a-3.5b show that average linkage methods (HC-Avg, HCC-Avg) strongly outperform other methods on *CREMI-gridGraph* instances and also achieve the best scores on *CREMI-3D-rag* graphs. Sum-based linkage methods (GAEC, HCC-Sum) have a two times higher ARAND error on grid-graphs and often return under-clustered segments (see failure cases in Fig. 3.4). This suggests that the *flooding strategy*

	ARAND Error	VOI split	VOI merge	Runtime (s)
HC-Avg	**0.0487**	0.387	0.258	2344
HCC-Avg	0.0492	0.389	0.259	2892
MWS [154]	0.0554	0.440	0.249	688
GAEC [72]	0.0856	0.356	0.338	4717
HCC-Sum	0.0872	0.365	0.337	4970
HC-Complete	0.9211	4.536	**0.211**	1020
HC-Single	0.9264	**0.060**	4.887	**312**
HCC-Single	0.9264	**0.060**	4.887	6440

(a) CREMI-gridGraph (OurCNN)

	ARAND Error	VOI split	VOI merge	Runtime (s)
HC-Avg	**0.0896**	0.603	0.323	86
HCC-Avg	0.0898	0.600	0.325	87
GAEC [72]	0.0905	0.606	0.323	89
HCC-Sum	0.0910	0.608	0.323	**85**
MWS [154]	0.1145	0.825	0.295	86
HCC-Single	0.5282	**0.437**	1.367	88
HC-Single	0.5282	**0.437**	1.367	**85**
HC-Complete	0.5654	2.253	**0.249**	86

(b) CREMI-3D-RAG (OurCNN)

	Needs superpixels?	CREMI Score	ARAND Error	VOI split	VOI merge
OurCNN: 3D-RAG + LiftedMulticut	☒	**0.221**	**0.108**	**0.339**	0.115
GASP: OurCNN + gridGraph + HCC-Avg	☐	0.224	0.113	0.361	0.085
GASP: OurCNN + gridGraph + HC-Avg	☐	0.224	0.114	0.364	0.083
PNI CNN [90]	☒	0.228	0.116	0.345	0.106
LSI-Masks [16]	☐	0.246	0.125	0.383	0.107
GASP: OurCNN + 3D-RAG + HCC-Avg	☒	0.257	0.132	0.438	**0.063**
GASP: OurCNN + 3D-RAG + HC-Avg	☒	0.262	0.135	0.448	**0.063**
MALA CNN + MC [54]	☒	0.276	0.132	0.490	0.089
CRU-Net [163]	☒	0.566	0.229	1.081	0.389

(c) CREMI Challenge leader-board

Table 3.5. (a-b): Scores and run times of algorithms in the GASP framework on the *CREMI-gridGraph* and *CREMI-3D-RAG* clustering problems: average linkage methods achieved the best accuracy. Measures shown are: Adapted-Rand error (ARAND, lower is better); Variation of Information (VOI) [12] (VOI-merge for under-clustering error and VOI-split for over-clustering error, lower values are better). **(c)**: Current leading entries in the CREMI challenge leaderboard. CREMI-score is given by the geometric mean of (VOI-split + VOI-merge) and ARAND error (lower is better).

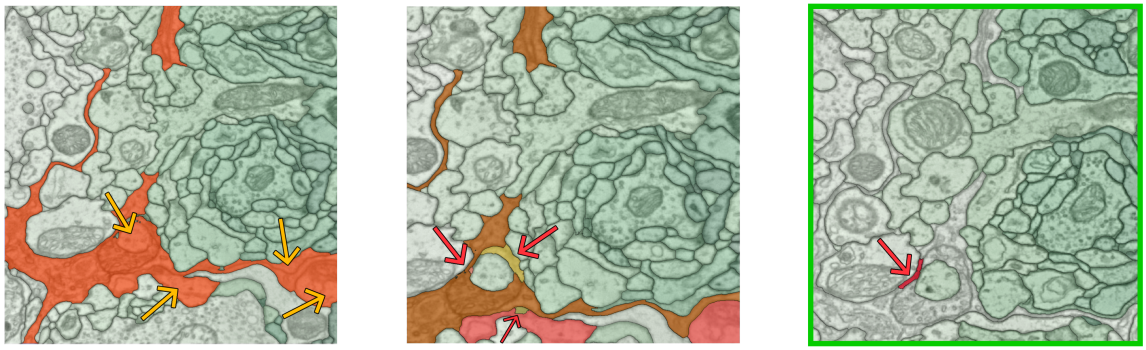


Figure 3.4. Failure cases of three versions of GASP applied to neuron segmentation. Only *wrongly* segmented regions are highlighted in different warm colors. Red arrows point to wrongly split regions; yellow arrows point to false merge errors. HC-Avg returned the best segmentation. Data is 3D, hence the same color could be assigned to parts of segments that appear disconnected in 2D.

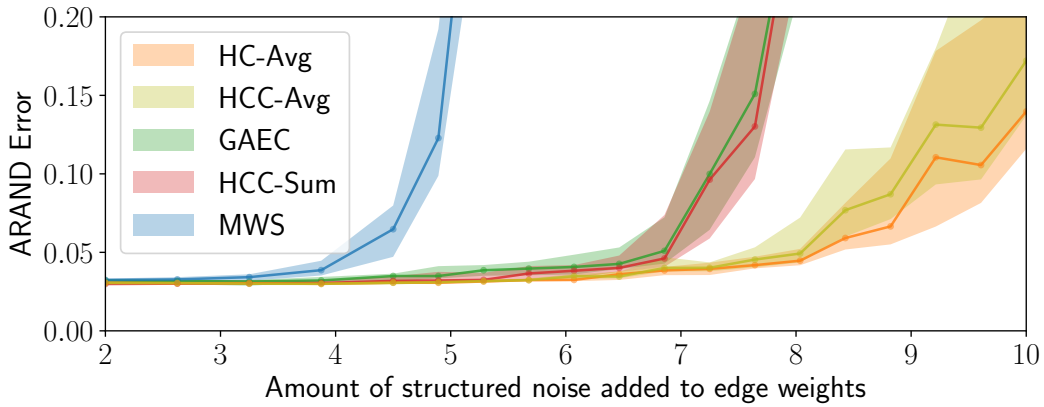


Figure 3.5. ARAND errors (median values over 20 experiments, lower is better) on *CREMI-gridGraph* clustering problems perturbed with structured noise. Average-linkage algorithms proved to be the most robust.

observed previously in the sum-linkage methods does not work on grid-graphs, because in this setup edge weights are predicted by a CNN and noise is strongly spatially-correlated [9]. To fully test this hypothesis, we conduct a set of experiments where the CNN predictions are perturbed by adding structured noise and simulating additional artifacts like "holes" in the boundary evidence[10]. The plot in Fig. 3.5 confirms that HC-Avg and HCC-Avg are very robust algorithms on this data, followed by Sum-linkage algorithms and the Mutex Watershed algorithm (MWS). It is not a surprise that the AbsMax linkage used by MWS is not robust to this type of structured noise. However, the scores and runtimes in Table 3.5a prove how MWS can achieve high accuracy with 70% lower runtime compared to HC-Avg.

Complete and Single Linkage We use these two linkage methods as baselines to highlight the difficulty of the studied graph clustering problems listed in Table 3.2. Scores in Tables 3.5a-3.5b show their poor performance: Single linkage hierarchical clustering (HC-Single), which here is equivalent to thresholding the edge weights at $w_e = 0$ and computing connected components in the graph, often returned few big under-segmented clusters. HC-Complete returned instead a lot of over-segmented clusters.

Results on CREMI challenge Table 3.5c shows that the HCC-Avg and HC-Avg clustering algorithms achieve state-of-the-art accuracy on the CREMI challenge, when combined with predictions of our CNN. Most of the other entries (apart from *LSI-Masks* [16]) employ super-pixels based post-processing pipelines and cluster 3D-region-adjacency graphs. As we show in Table 3.5b, using superpixels considerably reduces the size of the clustering problem and, consequently, the post-processing time. However, our method operating directly on pixels (*gridGraph + HCC-Avg*) achieves better performances than superpixel-based methods (*3D-RAG + HCC-Avg*) and does not require the parameter tuning necessary to obtain good super-pixels, which is usually highly dataset dependent. To scale up our method operating on pixels, we divided each test-volume into four sub-blocks, and then combined the resulting clusterings by running the algorithms again on the combined graph. The method *3D-RAG + LiftedMulticut* based on the lifted multicut approximation of [20] achieves the best scores overall, but it takes into account different information through the lifted edge weights that also depend on additional raw-data and shape information from highly engineered super-pixels.

3.5 Conclusion

We have presented a unifying framework for agglomerative clustering of graphs with both positive and negative edge weights. This framework allowed us to explore new combinations of constraints and linkage criteria and to perform a consistent evaluation of all algorithms in it. We have then analyzed several theoretical and empirical properties of these algorithms. On instance segmentation, algorithms based on an average linkage criterion outperformed all the others: they proved to be simple and robust approaches to

[9] This effect is not as strong on *3D-RAG* graphs, because edge weights are computed by averaging CNN predictions (and noise) over the boundaries of adjacent supervoxels.

[10] See Appendix A.3 for details about how we perturbed the *CREMI-gridGraph* problems by using Perlin noise [120, 121], which is one of the most common gradient noises used in procedural pattern generation.

process short- and long-range predictions of a CNN. On biological images, these simple average agglomeration algorithms achieve state-of-the-art results without requiring the user to spend much time tuning complex task-dependent pipelines based on super-pixels.

Chapter 4

Predicting Latent Single-Instance Masks

In this chapter we introduce a new proposal-free instance segmentation method that builds on single-instance segmentation masks predicted across the entire image in a sliding window style. In contrast to related approaches, this method concurrently predicts all masks, one for each pixel, and thus resolves any conflict jointly across the entire image. Specifically, predictions from overlapping masks are combined into edge weights of a signed graph that is subsequently partitioned to obtain all final instances concurrently. The result is a parameter-free method that is strongly robust to noise and prioritizes predictions with the highest consensus across overlapping masks. All masks are decoded from a low dimensional latent representation, which results in great memory savings strictly required for applications to large volumetric images. We test our method on the challenging CREMI 2016 neuron segmentation benchmark where it achieves competitive scores.

4.1 Introduction

Instance segmentation is the computer vision task of assigning each pixel of an image to an instance, such as individual car, person or biological cell. There are two main types of successful deep learning approaches to instance segmentation: *proposal-based* and *proposal-free* methods. Recently, there has been a growing interest in the latter. Proposal-free methods do not require object detection and are preferred in imagery as studied here, in which object instances cannot be approximated by bounding boxes and are much larger than the field of view of the model.

Some recent successful proposal-free approaches [64, 93, 101] tackle instance segmentation by predicting, for a given patch of the input image, whether or not each pixel in the patch is part of the instance that covers the central pixel of the patch. This results a probability mask, which from now on we call *central instance mask*. These masks are then repeatedly predicted across the entire image, either in a sliding window style or by starting from a seed and then shifting the field of view depending on the previously predicted masks. Final object-instances are then obtained by aggregating predictions from overlapping masks which is in itself a nontrivial and interesting problem.

In this chapter, we introduce a new proposal-free segmentation method that is also

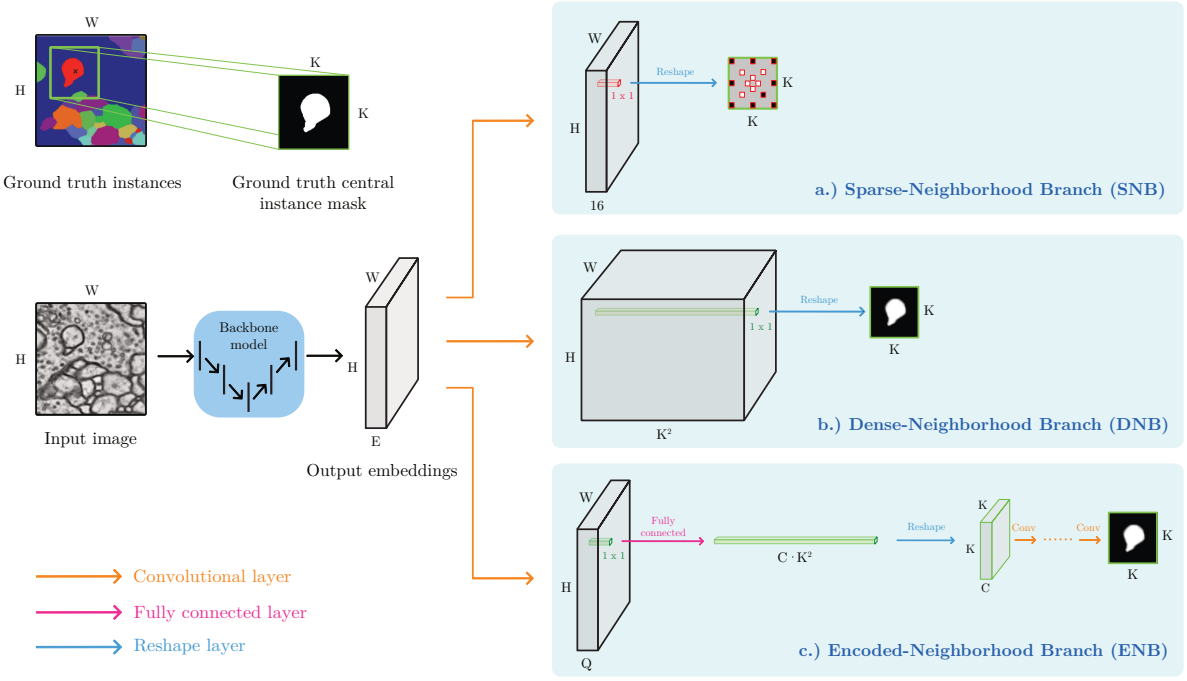


Figure 4.1. Comparison between the proposed method and the strong baseline representing the current state-of-the-art. **Left**: At the top-left corner, an example of binary central instance mask for a given ground truth label image; below, the backbone model predicts feature maps with the spatial dimensions of the input image. **Right**: a.) *Sparse-neighborhood branch* used in the baseline model to predict affinities for a given sparse neighborhood structure; b) Simple generalization of the *sparse-neighborhood branch* to predict dense central instance masks; c) Proposed *encoded-neighborhood branch* predicting central instance masks in a low-dimensional latent space.

based on predicting central instance masks[1]. However, our approach comes with four main advantages compared to previous methods. Firstly, our model concurrently predicts all central instance masks, one for each pixel, by using a fully-convolutional approach with much smaller computational footprint than previous methods, which iteratively predict one instance at the time, one mask after the other [64, 101]. Secondly, our approach predicts central instance masks in a low dimensional latent representation (see Fig. 4.1c), which results in great memory savings that are strictly required to apply the method to large volumetric images. Thirdly, the proposed approach aggregates predictions from overlapping central instance masks without the need for any extra parameter or threshold and outputs predictions with associated uncertainty; and, finally, all final object-instances are obtained concurrently, as opposed to previous methods predicting them one-by-one with subsequent conflict resolution.

Additionally, we systematically compare the proposed model with the current state-of-the-art proposal-free method both on natural and biological images [15, 56, 90, 97, 154]. This strong baseline consists of a fully-convolutional network predicting, for each pixel, an arbitrary predefined set of short- and long-range affinities, i.e. neighborhood relations representing how likely it is for a pair of pixels to belong to the same object instance (see Fig. 4.1a).

[1]For interesting, closely related but independent work, see [99].

Our method achieves competitive scores on the challenging CREMI 2016 neuron segmentation benchmark. In our set of validation experiments, we show how predicting encoded central instance masks always improves accuracy. Moreover, when predictions from overlapping masks are combined into edge weights of a graph that is subsequently partitioned, the result is a method that is strongly robust to noise and gives priority to predictions sharing the highest consensus across predicted masks. This parameter-free algorithm, for the first time, outperforms super-pixel based methods, which have so far been the default choice on the challenging data from the CREMI competition challenge.

4.2 Related Work

Many of the recent successful instance segmentation methods on natural images are *proposal-based*: they first perform object detection, for example by predicting anchor boxes [128], and then assign a class and a binary segmentation mask to each detected bounding box [61, 125]. Proposal-Free methods on the other hand directly group pixels into instances. Recent approaches use metric learning to predict high-dimensional associative pixel embeddings that map pixels of the same instance close to each other, while mapping pixels belonging to different instances further apart, e.g. [82, 89]. Final instances are then retrieved by applying a clustering algorithm. A post-processing step is needed to merge instances that are larger then the field of view of the network.

Aggregating Central Instance Masks The line of research closest to ours predicts overlapping central instance masks in a sliding window style across the entire image. The work of [93] aggregates overlapping masks and computes intersection over union scores between them. In neuron segmentation, flood-filling networks [64] and MaskExtend [101] use a CNN to iteratively grow one instance/neuron at a time, merging one mask after the other. Recently, the work of [102] made the process more efficient by employing a combinatorial encoding of the segmentation, but the method remains orders of magnitude slower as compared to the convolutional one proposed here, since in our case all masks are predicted at the same time and for all instances at once. The most closely related work to ours is the independent preprint [99], where a very similar model is applied to the BBBC010 benchmark microscopy dataset of *C. elegans* worms. However, here we propose a more efficient model that scales to 3D data, and we provide an extensive comparison to related models predicting long-range pixel-pair affinities.

Predicting Pixel-Pair Affinities Instance-aware edge detection has experienced recent progress due to deep learning, both on natural images and biological data [15, 56, 90, 97, 119, 137, 154, 163]. Among these methods, the most recent ones also predict long-range affinities between pixels and not only direct-neighbor relationships [56, 90, 97]. Other related work [54, 144] approach boundary detection via a structured learning approach. In neuron segmentation, boundaries predicted by a CNN are converted to final instances with subsequent postprocessing and superpixel-merging. Some methods define a graph with both positive and negative weights and formulate the problem in a combinatorial framework, known as multicut or correlation clustering problem [32]. In neuron segmentation and connectomics, exact solvers can tackle problems of considerable size [7], but accurate approximations [117, 161] and greedy agglomerative algorithms [15, 92, 153] are required on larger problems.

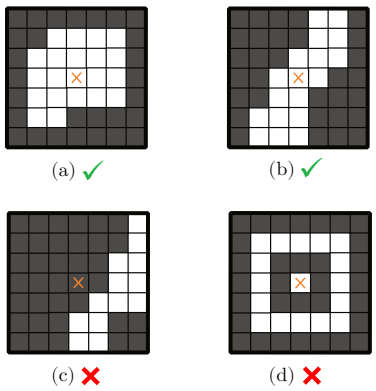


Figure 4.2. Examples of expected (**a-b**) and not expected (**c-d**) binary 2D central instance masks.

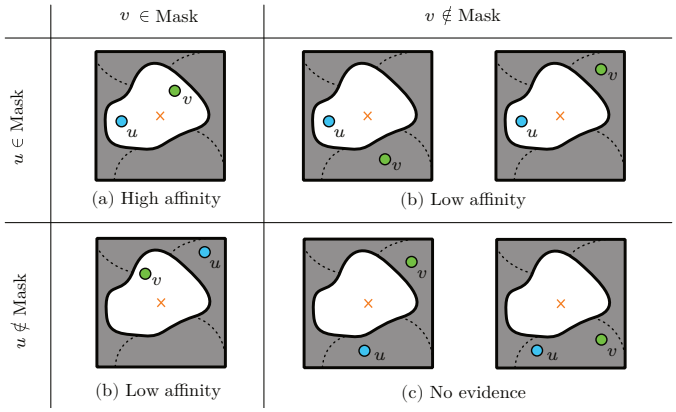


Figure 4.3. Computing instance-aware affinity between pixels u and v from instance masks associated to the central pixel in the patch (orange cross).

4.3 Model and Training Strategy

In this section, we first define central instance masks in Sec. 4.3.1. Then, in Sec. 4.3.2, we present our first main contribution, a model trained end-to-end to predict encoded central instance masks, one for each pixel of the input image.

4.3.1 Local Central Instance Masks

The method presented here proposes to distinguish between different object instances based on instance-aware pixel-pair affinities in the interval $[0, 1]$, which specify whether or not two pixels belong to the same instance or not. Given a pixel of the input image with coordinates $\vec{u} = (u_x, u_y)$, a set of affinities to neighboring pixels within a $K \times K$ window is learned, where K is an odd number. We define the $K \times K$-neighborhood of a pixel as: $\mathcal{N}_{K\times K} \equiv \mathcal{N}_K \times \mathcal{N}_K$, where $\mathcal{N}_K \equiv \left\{-\frac{K-1}{2}, \ldots, \frac{K-1}{2}\right\}$ and represent the affinities relative to pixel $\vec{u}$ as a central instance mask $\mathcal{M}_{\vec{u}} : \mathcal{N}_{K\times K} \to [0, 1]$.

We represent the associated training targets as binary ground-truth masks $\hat{\mathcal{M}}_{\vec{u}} : \mathcal{N}_{K\times K} \to \{0, 1\}$, which can be derived from a ground-truth instance label image $\hat{L} : H \times W \to \mathbb{N}$ with dimension $H \times W$:

$$\forall\, \vec{u} \in H \times W, \quad \forall\, \vec{n} \in \mathcal{N}_{K\times K} \qquad \hat{\mathcal{M}}_{\vec{u}}(\vec{n}) = \begin{cases} 1, & \text{if } \hat{L}(\vec{u}) = \hat{L}(\vec{u} + \vec{n}) \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

We actually use similar definitions in 3D, but use 2D notation here for simplicity.

4.3.2 Training Encoded Central Instance Masks End-To-End

In several related work approaches [15, 56, 90, 97, 154], affinities between pairs of pixels are predicted for a predefined sparse stencil representing a set of N short- and long-range neighborhood relations for each pixel ($N = 8$ *sparse-neighborhood branch* of Fig. 4.1a). The N output feature maps are then trained with a binary classification loss.

On paper, this training method can be easily generalized to output a feature map of size $K^2 \times H \times W$ and thus predict a full $K \times K$ central instance mask for each

pixel of the input image (see *dense-neighborhood branch* in Fig. 4.1b). Nevertheless, in practice, this model has prohibitively large memory requirements for meaningful values of K, precluding application to 3D data of interest here.

However, among the $2^{K \cdot K}$ conceivable binary masks $\hat{\mathcal{M}}_{\vec{u}} : \mathcal{N}_{K^2} \rightarrow \{0, 1\}$, in practice only a tiny fraction corresponds to meaningful instance masks (see some examples in Fig. 4.2). This suggests that it is possible to find a compact representation that spans the manifold of expected instance shapes.

As our first main contribution, we test this assumption by training a model end-to-end to predict, for each pixel $\vec{u} \in H \times W$ of the input image, a latent vector $z_{\vec{u}} \in \mathbb{R}^Q$ encoding the $K \times K$ central instance mask $\mathcal{M}_{\vec{u}}$ centered at pixel $\vec{u}$ (see *encoded-neighborhood branch* in Fig. 4.1c). The backbone model is first trained to output a more compact $Q \times H \times W$ feature map and then a tiny convolutional decoder network is applied to each pixel of the feature map to decode masks. During training, decoding one mask for each pixel in the image would be too memory consuming. Thus, we randomly sample R pixels with coordinates $\vec{u}_1, \ldots, \vec{u}_R$ and only decode the associated masks $\mathcal{M}_{\vec{u}_1}, \ldots, \mathcal{M}_{\vec{u}_R}$. Given the ground-truth central instance masks $\hat{\mathcal{M}}_{\vec{u}_i}$ defined in Eq. 4.1, the training loss is then defined according to the Sørensen-Dice coefficient formulated for fuzzy set membership values, similarly to what was done in [154]. Ground-truth labels are not always pixel-precise and it is often impossible to estimate the correct label for pixels that are close to a ground-truth label transition. Thus, in order to avoid noise during training, we predict completely empty masks for pixels that are less than two pixels away from a label transition, so that the model is trained to predict single-pixel clusters along the ground-truth boundaries. In our experiments, this approach performed better than masking the training loss along the boundaries.

4.3.3 Predicting Multi-Scale Central Instance Masks

Previous related work [56, 90, 97] shows that predicting long-range affinities between distant pixels improves accuracy as compared to predicting only short-range ones. However, predicting large central instance masks would translate to a bigger model that, on 3D data, would have to be trained on a small 3D input field of view. This, in practice, usually decreases accuracy because of the reduced 3D context available to the network. Thus, we instead predict multiple central instance masks of the same window size $7 \times 7 \times 5$ but at different resolutions, so that the lower the resolution the larger the size of the associated patch in the input image. These multiple masks at different resolutions are predicted by adding several *encoded-neighborhood branches* along the hierarchy of the decoder in the backbone model, which in our case is a 3D U-Net [34, 131] (see Fig. 4.5). In this way, the encoded central instance masks at higher and lower resolutions can be effectively learned at different levels in the feature pyramid of the U-Net.

4.4 Affinities with Uncertainty from Aggregated Masks

In order to obtain an instance segmentation from the predictions of the model presented in Sec. 4.3, we now compute instance-aware pixel-pair affinities for a given sparse N-neighborhood structure (see Table 4.3 in supplementary material for details about the structure) and use them as edge weights of a pixel grid-graph $\mathcal{G}(V, E)$, such that each node represents a pixel / voxel of the image. The graph is then partitioned to obtain object instances.

Algorithm 9 Affinities from Aggregated Central Instance Masks

Input: Graph $\mathcal{G}(V, E)$; central instance masks $\mathcal{M}_{\vec{u}} : \mathcal{N}_{K\times K} \rightarrow [0, 1]$
Output: Affinities $\bar{a}_e \in [0, 1]$ with variance σ_e^2 for all edges $e \in E$

1: **for** each edge $e = (\vec{u}, \vec{v}) \in E$ in graph $\mathcal{G}$ **do**
2: Get coordinates $\vec{u} = (u_x, u_y)$ and $\vec{v} = (v_x, v_y)$ of pixels linked by edge e
3: Collect all T masks $\mathcal{M}_{\vec{c}_1}, \ldots, \mathcal{M}_{\vec{c}_T}$ including both pixel $\vec{u}$ and pixel $\vec{v}$
4: Init. vectors $[a_1, \ldots, a_T] = [w_1, \ldots, w_T] = 0$ for affinities and evidence weights
5: **for** $i \in 1, \ldots, T$ **do**
6: Get relative coords. of $\vec{u}$ and $\vec{v}$ with respect to the central pixel $\vec{c}_i$
7: $a_i \leftarrow \min\left(\mathcal{M}_{\vec{c}_i}(\vec{u} - \vec{c}_i), \mathcal{M}_{\vec{c}_i}(\vec{v} - \vec{c}_i)\right)$ ▷ Fuzzy-AND: both values active
8: $w_i \leftarrow \max\left(\mathcal{M}_{\vec{c}_i}(\vec{u} - \vec{c}_i), \mathcal{M}_{\vec{c}_i}(\vec{v} - \vec{c}_i)\right)$ ▷ Fuzzy-OR: at least one value active
9: Get weighted affinity average $\bar{a}_e = \sum_i a_i w_i \,/\, \sum_i w_i$
10: Get weighted affinity variance $\sigma_e^2 = \sum_i w_i (a_i - \bar{a}_e)^2 \,/\, \sum_i w_i$
11: **return** $\bar{a}_e, \sigma_e^2$ for each $e \in E$

In this section, we propose an algorithm that, without the need of any threshold parameter, aggregates predictions from overlapping central instance masks and outputs edge weights with associated uncertainty. Related work either thresholds the predicted central instance masks [64, 99, 101] or computes Intersection over Union (IoU) scores for overlapping patches [93]. However, an advantage of predicting pixel-pair affinities / pseudo-probabilities as compared to IoU scores is that affinities can easily be translated into attractive and repulsive interactions in the grid-graph and a parameter-free partitioning algorithm can be employed to yield instances.

Here, we propose a simple algorithm to aggregate predictions from multiple patches: Fig. 4.4 shows a simplified example of how Algorithm 9 computes the affinity for an edge e linking a pair of pixels $\vec{u}$ and $\vec{v}$. As a first step, the algorithm loops over all predicted central instance masks including both $\vec{u}$ and $\vec{v}$. However, not all these masks are informative, as we visually explain in Fig. 4.3: a mask $\mathcal{M}_{\vec{c}_i}$ centered at pixel $\vec{c}_i$ provides any evidence about the affinity between pixels $\vec{u}$ and $\vec{v}$ only if at least one of the two pixels belongs to the mask (fuzzy OR operator at line 8 in Alg. 9). If both pixels do not belong to it, we cannot say anything about whether they belong to the same instance (see Fig. 4.3c). We model this with an evidence weight $w_i \in [0, 1]$, which is low when both pixels do not belong to the mask. On the other hand, when at least one of the two pixels belongs to the mask, we distinguish two cases (fuzzy AND operator at line 7 in Alg. 9): i) both pixels belong to the mask (case in Fig. 4.3a), so by transitivity we conclude they should be in the same instance and their affinity a_i should tend to one; ii) only one pixel belongs to the mask (case in Fig. 4.3b), so that according to this mask they are in different instances and their affinity should tend to zero.

At the end, we compute a weighted average $\bar{a}_e$ and variance σ_e^2 of the collected affinities from all overlapping masks, such that masks with more evidence will contribute more on average, and the obtained variance is a measure of how consistent were the predictions across masks. The algorithm was implemented on GPU using PyTorch and the variance was computed via Welford's online stable algorithm [152].

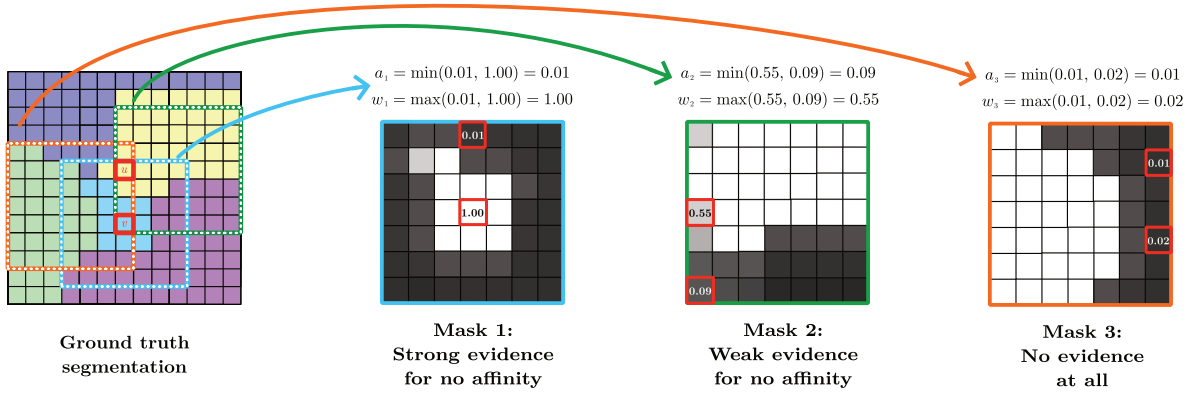


Figure 4.4. Proposed method to average overlapping masks and compute the affinity between pixel u and pixel v (highlighted in red in the ground-truth segmentation on the left). For simplicity, we only consider three masks among all the ones including both pixels u and v. In *Mask 1*, only v is part of the mask, so there is a strong evidence for no affinity between u and v; in *Mask 2*, u is predicted to be part of the mask only with a low confidence, so the contribution of this mask in the final average will be weak; in *Mask 3*, both pixels are not part of the central instance mask, so there is no evidence about their affinity. The final affinity value of edge (u, v) is given by the weighted average of the collected affinities a_i weighted with the evidence weights w_i: $\bar{a}_e = \sum_{i=1}^{3} a_i w_i \,/\, \sum_i w_i$

4.5 Experiments on Neuron Segmentation

We evaluate and compare our method on the task of neuron segmentation in electron microscopy (EM) image volumes. This application is of key interest in connectomics, a field of neuro-science with the goal of reconstructing neural wiring diagrams spanning complete central nervous systems. We test our method on the competitive CREMI 2016 EM Segmentation Challenge [40] that is currently the neuron segmentation challenge with the largest amount of training data available. The dataset comes from serial section EM of *Drosophila* fruit-fly brain tissue and consists of 6 volumes of $1250 \times 1250 \times 125$ voxels at resolution $4 \times 4 \times 40$ nm^3, three of which come with publicly available training ground truth. We achieved the best scores by downscaling the resolution of the EM data by a factor $(\frac{1}{2}, \frac{1}{2}, 1)$, since this helped increasing the 3D context provided as input to the model. We use the second half of CREMI sample C as validation set for our comparison experiments in Table 4.1 and then we train a final model on all the three samples with available ground truth labels to submit results to the leader-board in Tab. 4.2. Results are evaluated using the CREMI score, which is given by the geometric mean of Variation of Information Score (VOI split + VOI merge) and Adapted Rand-Score (Rand-Score) [12]. See Sec. 3.4.2 for more details on data augmentation, strongly inspired by related work.

4.5.1 Architecture details of the tested models

As a backbone model we use a 3D U-Net consisting of a hierarchy of four feature maps with anisotropic downscaling factors $(\frac{1}{2}, \frac{1}{2}, 1)$, similarly to [89,90,154]. Models are trained with the Adam optimizer and a batch size equal to one. Before applying the loss, we slightly crop the predictions to prevent training on borders where not enough surrounding context is provided. Fig. 4.5 shows the details on the 3D-UNet architecture and Table 4.3 lists the sparse neighborhood structures predicted by the *sparse-neighborhood branches*.

Only the outputs at the highest resolution (given by branches SNB_1, ENB_1 and ENB_2) are used to compute edge weights in the pixel grid-graph. A visualization of the predicted single-instance mask latent spaces is given in Fig. 4.7.

The input volume has shape $272 \times 272 \times 12$ which is equivalent to a volume of $544 \times 544 \times 12$ voxels in the original resolution $4 \times 4 \times 40$ nm^3. Before to apply the loss, we crop the predictions to a shape $224 \times 224 \times 9$ in order to avoid border artifacts[2]. The final model trained on all available ground truth labels is trained with a slightly larger input volume of $288 \times 288 \times 14$.

Baseline Model (SNB) As a strong baseline, we re-implement the current state-of-the-art and train a model to predict affinities for a sparse neighborhood structure (Fig. 4.1a). We perform deep supervision by attaching three *sparse-neighborhood branches* (SNB) at different levels in the hierarchy of the UNet decoder and train the coarser feature maps to predict longer range affinities. Details about the used neighborhood structures and the architecture can be found in Table 4.3 and Fig. 4.5.

Proposed Model (ENB) We then train a model to predict encoded central instance masks (Fig. 4.1c). Similarly to the baseline model, we provide deep supervision by attaching four *encoded-neighborhood branches* (ENB) to the backbone U-Net. As explained in Sec. 4.3.3, all branches predict 3D masks of shape $7 \times 7 \times 5$, but at different resolutions $(1, 1, 1)$, $(\frac{1}{4}, \frac{1}{4}, 1)$ and $(\frac{1}{8}, \frac{1}{8}, 1)$, as we show in the architecture in Fig. 4.5. A visualization of the learned latent spaces is given in Fig. 4.7.

Combined Model (SNB+ENB) Finally, we also train a combined model to predict both central instance masks and a sparse neighborhood of affinities, by providing deep supervision both via *encoded-neighborhood* and *sparse-neighborhood branches*. The backbone of this model is then trained with a total of seven branches: three branches equivalent to the ones used in the baseline model SNB, plus four additional ones like those in the ENB model (see Fig. 4.5).

Efficient Affinities for any Sparse Neighborhood An advantage of training dense central instance masks is that the graph N-neighborhood structure can be defined at prediction time, after the model has been trained. As an alternative to the method presented in Sec. 4.4 that aggregates overlapping masks, here we propose the following efficient approach to predict affinities for a sparse neighborhood structure: Given a model that has been already trained end-to-end to predict encoded central instance masks, we stack few additional convolutional layers that are trained to convert the Q-dimensional latent mask space to N output feature maps representing affinities for the chosen sparse neighborhood structure. These last layers are not trained jointly with the full model, so in practice they are very quick and easy to train with a binary classification loss. By using this method, we avoid to decode all masks explicitly (one for each pixel) and achieve great time and memory savings. As a result, we obtain a model that at inference time is no more memory-consuming than the current state-of-the-art approach predicting affinities only for a specific sparse neighborhood structure.

[2]Instead of cropping directly the final predictions, we perform several crops in the decoder part of the UNet model (see Upsample + Crop connections in Fig. 4.5) in order to optimize GPU-memory usage.

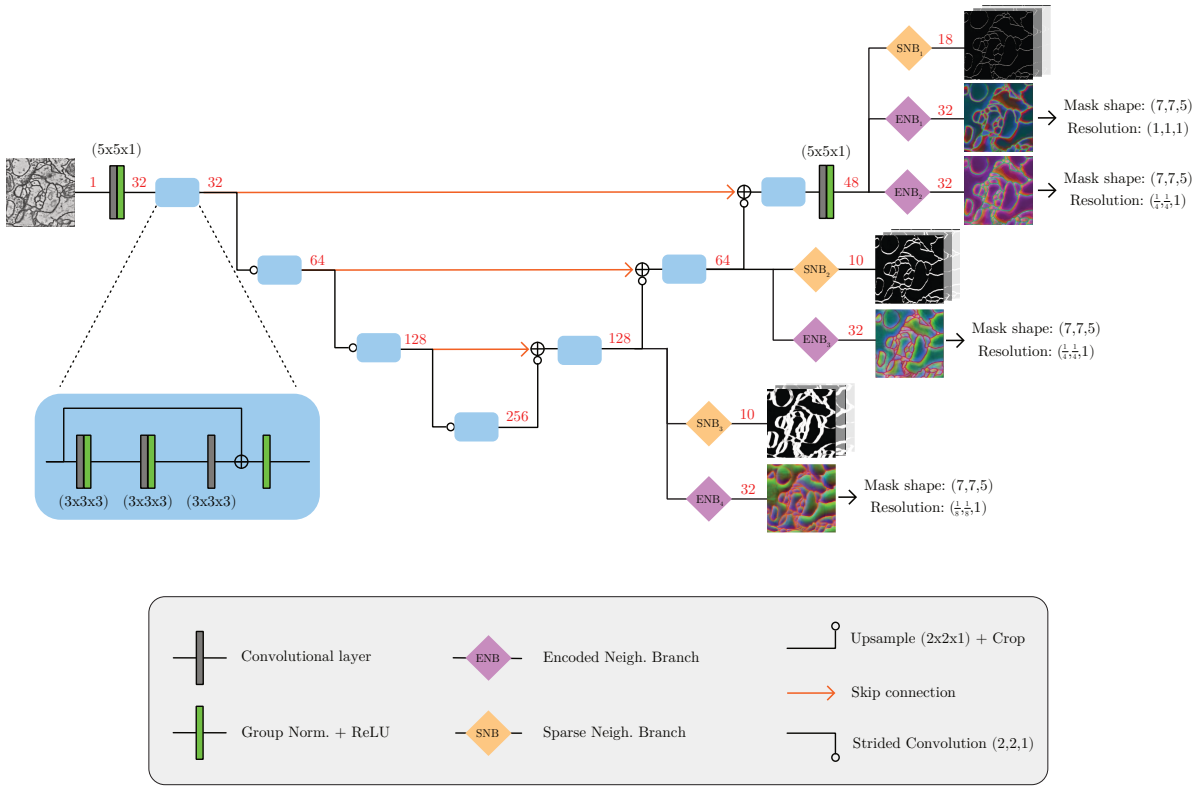


Figure 4.5. The architecture of the model, which is strongly inspired by the 3D-UNet models proposed in [54, 90]. Red numbers indicate the number of used feature maps. As we explain in Sec. 4.5.1, in this work we consider three models: i) a baseline model based on the three *sparse-neighborhood branches* $\text{SNB}_{i=1,2,3}$, shown in the figure; ii) another model based on the four *encoded-neighborhood branches* $\text{ENB}_{i=1,2,3,4}$; iii) and, finally, a combined model trained with all seven branches shown in the Figure. Even though the input of the model is a 3D volume, here, for simplicity, we show an example of 2D input image taken from the stack. As output of the *sparse-neighborhood branches* $\text{SNB}_{i=1,2,3}$, we show few channels representing some of the predicted affinities (see Table 4.3 for details on the sparse neighborhood structures predicted by each branch $\text{SNB}_{i=1,2,3}$). We also show the first three principal components of the encoded masks predicted by the *encoded-neighborhood branches* $\text{ENB}_{i=1,2,3,4}$. All branches $\text{ENB}_{i=1,2,3,4}$ predict central instance masks of the same window size $7 \times 7 \times 5$, but at different resolutions.

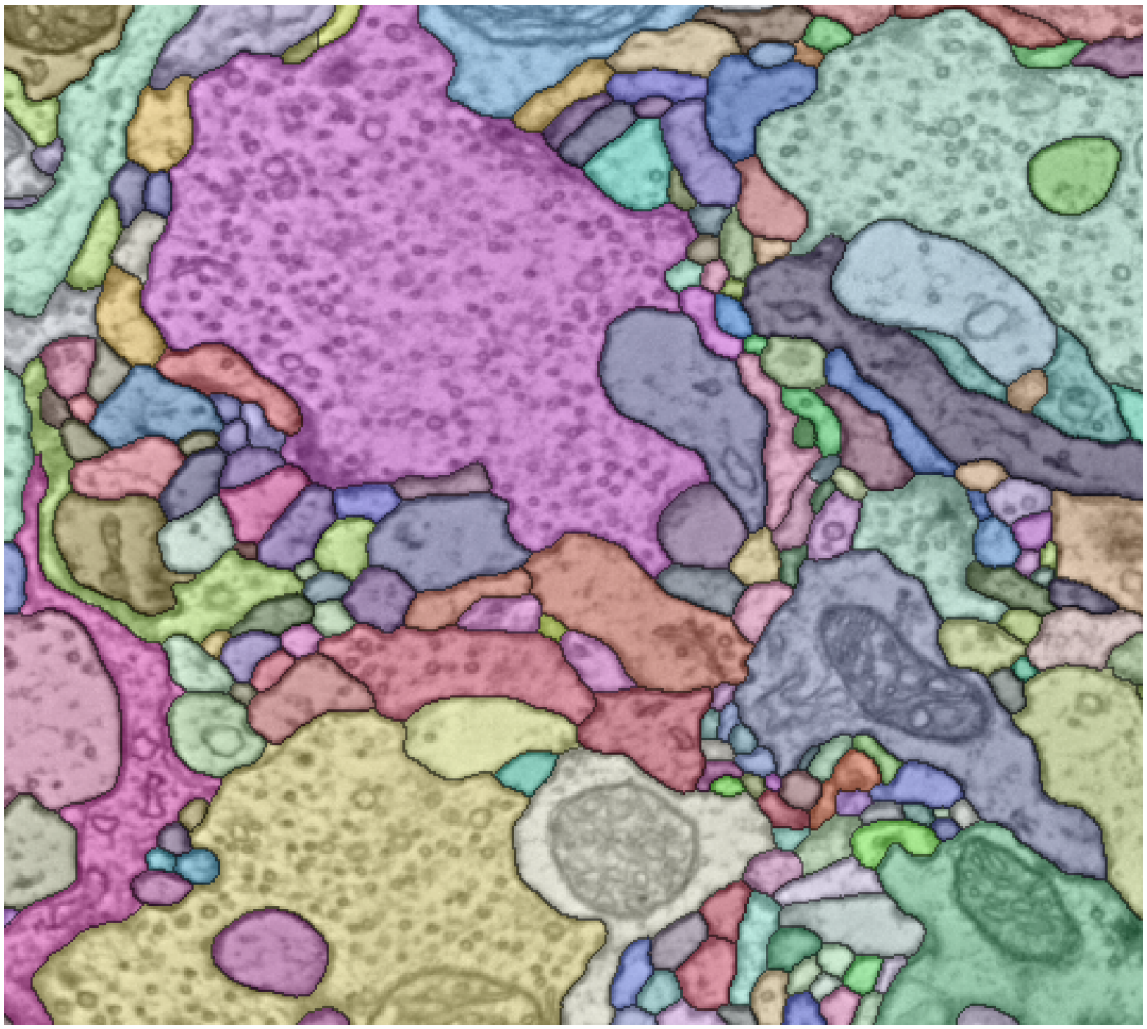


Figure 4.6. Raw data from the validation set overlaid with the final instance segmentation obtained with our method: affinities are computed by averaging overlapping masks (MaskAggr); the final segmentation is achieved by running the Mutex Watershed algorithm on the obtained graph with positive and negative edge weights. Note that the data is 3D, hence the same color could be assigned to parts of segments that appear disconnected in 2D.

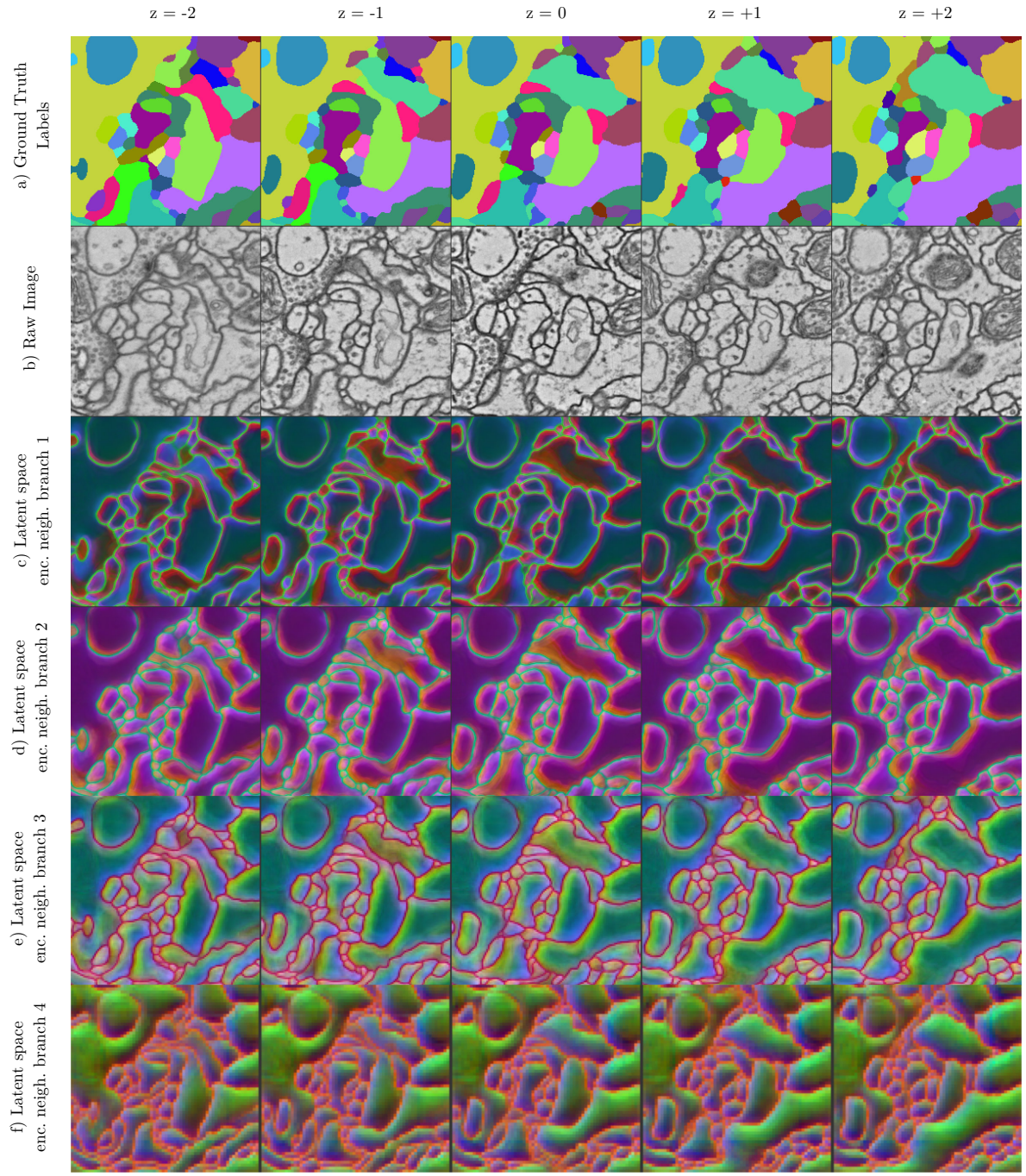


Figure 4.7. Visualization of the predicted single-instance mask latent spaces – Each column represents a 2D image from the 3D stack (only five are shown here). **(a)** Ground-truth labels. **(b)** Raw image given as input to the model. **(c-d-e-f)** Visualization of the first three principal components of the 32-dimensional mask latent spaces predicted by the encoded-neighborhood branches $\text{ENB}_{i=1,2,3,4}$ in our model. Note how latent spaces learned at different levels of the U-Net pyramid show different feature-scales, because they encode central instance masks at different resolutions.

4.5.2 Graph Partitioning Methods

Given the predicted encoded central instance masks, we compute affinities a_e either with the average aggregation method introduced in Sec. 4.4 (**MaskAggr**) or the efficient approach described in the previous section 4.5.1. The result of either is a signed pixel grid-graph, i.e. a graph with positive and negative edge weights that needs to be partitioned into instances. The used neighborhood connectivity of the graph, which is very similar to the one used in related work [90, 154], is given in Table 4.3. Positive and negative edge weights w_e are computed by applying the additive transformation $w_e = a_e - 0.5$ to the predicted affinities.

To obtain final instances, we test different partitioning algorithms. The Mutex Watershed (**MWS**) [154] is an efficient algorithm to partition graphs with both attractive and repulsive weights without the need for extra parameters. It can easily handle the large graphs considered here with up to 10^8 nodes/voxels and 10^9 edges[3]. In Fig. 4.6, we show the resulting instance segmentation obtained by computing affinities from central instance masks and then running the Mutex Watershed algorithm on the obtained graph with positive and negative edge weights.

Then, we also test another graph partitioning pipeline that has often been applied to neuron segmentation because of its robustness. This method first generates a 2D super-pixel over-segmentation from the model predictions and then partitions the associated region-adjacency graph to obtain final instances. Super-pixels are computed with the following procedure: First, the predicted direct-neighbor affinities are averaged over the two isotropic directions to obtain a 2D neuron-membrane probability map; then, for each single 2D image in the stack, super-pixels are generated by running a watershed algorithm seeded at the maxima of the boundary-map distance transform (**WSDT**). Given this initial over-segmentation, a 3D region-adjacency graph is built, so that each super-pixel is represented by a node in the graph. Edge weights of this graph are computed by averaging short- and long-range affinities over the boundaries of neighboring super-pixels. Finally, the graph is partitioned by applying the average agglomeration algorithm proposed in [15] (**GaspAvg**).

After running the graph partitioning method, we use a simple post-processing step to delete small segments on the boundaries, most of which are given by single-voxel clusters. On the neuron segmentation predictions, we deleted all regions with less than 200 voxels and used a seeded watershed algorithm to expand the bigger segments.

4.5.3 Results and Discussion

Pre-Training of the Encoded Space The proposed model based on an *encoded-neighborhood branch* can be properly trained only if the dimension Q of the latent space is large enough to accommodate all possible occurring neighborhood patterns. To find a small but sufficiently large Q, we trained a convolutional Variational Auto-encoder (VAE) [76, 130] to compress binary ground-truth central instance masks $\hat{\mathcal{M}}_{\vec{u}}$ into latent variables $z_{\vec{u}} \in \mathbb{R}^Q$ and evaluated the quality of the reconstructed binary masks via the reconstruction loss. We concluded that $Q = 32$ is large enough to compress the masks considered here consisting of $7 \times 7 \times 5 = 245$ pixels. As a first experiment, we tried to make use of this VAE-pretrained latent space to train the proposed *encoded-neighborhood branch* and pre-

[3] Among all edges given by the chosen neighborhood structure, we add only 10% of the long-range ones, since the Mutex Watershed was shown to perform optimally in this setup [15, 154].

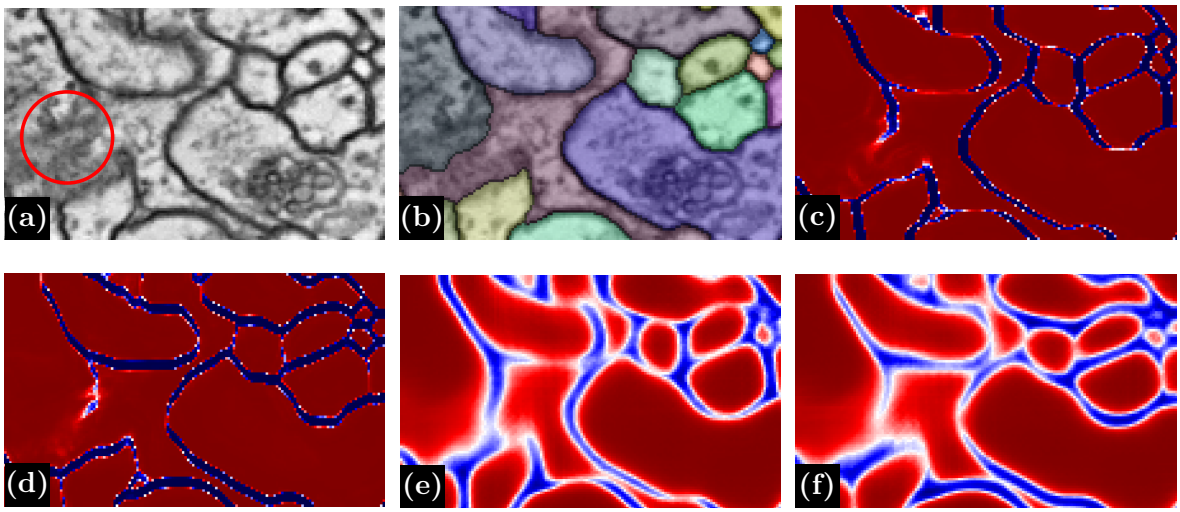


Figure 4.8. Comparison between different affinities and their robustness to noise. **(a-b)** Raw data and ground-truth labels. **(c-d)** Affinities predicted by the *sparse-neighborhood branch*, which is trained with a dense binary classification loss (high affinities are red). **(e-f)** Affinities computed by averaging overlapping masks as explained in Sec. 4.4 (MaskAggr). Affinities from averaged masks are smoother and present a more consistent boundary evidence in the noisy region highlighted by the red circle in (a). Here we show affinities along the horizontal (-4, 0, 0) and vertical (0, -4, 0) directions.

dict encoded masks directly in this space by using an L2 loss on the encoded vectors. However, similarly to the findings of [99], this approach performed worse than directly training the full model end-to-end as described in Sec. 4.3.2.

Training Encoded Masks As we show in our validation experiments in Tab. 4.1, models trained to predict encoded central instance masks (ENB) achieved better scores than the current state-of-the-art method predicting affinities for a sparse neighborhood structure (SNB). Our interpretation of this result is that using the encoding process to predict central instance masks encourages the model to predict segment shapes that are consistent in a larger neighborhood, which can be helpful to correctly segment the most difficult regions of the data.

Aggregating Overlapping Masks In our validation experiments of Tab. 4.1, we also test the affinities computed by averaging over overlapping masks (MaskAggr), as described in Sec. 4.4. We then partition the resulting signed graph by using the Mutex Watershed, which has empirical linearithmic complexity in the number of edges. Our experiments show that, for the first time on this type of more challenging neuron segmentation data, the Mutex Watershed (MWS) achieves better scores than the super-pixel-based methods (WSDT+GaspAvg), which have so far been known to be more robust to noise but also require the user to tune more hyper-parameters.

We also note that the MWS achieves competitive scores only with affinities computed from aggregating overlapping masks (MaskAggr). This shows that the MWS algorithm can take full advantage of the central instance aggregation process by assigning the highest priority to the edges with largest attractive and repulsive weights that were consistently predicted across overlapping masks.

On the other hand, most of the affinities trained with the *sparse-neighborhood*

Train Sparse Neighbor. (SNB)	Train Encoded Masks (ENB)	Aggregate Overlapping Masks (MaskAggr)	Partitioning algorithm	No superpixels required	CREMI-Score (lower is better)	VI-merge (lower is better)
✗	✗	✗	MWS	✗	**0.153**	0.272
□	✗	✗	MWS	✗	0.184	0.273
□	✗	□	MWS	✗	0.419	0.302
✗	✗	□	MWS	✗	0.532	0.447
✗	□	□	MWS	✗	1.155	0.874
□	✗	□	WSDT+GaspAvg	□	0.173	**0.234**
✗	✗	□	WSDT+GaspAvg	□	0.237	0.331
✗	□	□	WSDT+GaspAvg	□	0.254	0.355
✗	✗	✗	WSDT+GaspAvg	□	0.334	0.388
□	✗	✗	WSDT+GaspAvg	□	0.357	0.391

Table 4.1. Comparison experiments on our CREMI validation set. Training encoded central instance masks (ENB) achieved better scores than the current state-of-the-art approach training only affinities for a sparse neighborhood (SNB). The model that performed best was the one using the method proposed in Sec. 4.4 to average overlapping masks (MaskAggr).

branch and a dense binary classification loss are almost binary, i.e. they present values either really close to zero or really close one (see comparison between different types of affinities in Fig. 4.8). This is not an ideal setup for the MWS, which is a greedy algorithm merging and constraining clusters according to the most attractive and repulsive weights in the graph. In fact, in this setting the MWS can often lead to over-segmentation and under-segmentation artifacts like those observed in the output segmentations of the (SNB+ENB+MWS) and (SNB+MWS) models. Common causes of these mistakes can be for example inconsistent predictions from the model and partially missing boundary evidence, which are very common in this type of challenging application (see Fig. 4.8 for an example).

Finally, we also note that superpixel-based methods did not perform equally well on affinities computed from aggregated masks and the reason is that these methods were particularly tailored to perform well with the more *binary-like* classification output of the *sparse-neighborhood branch*.

Training Both Masks and a Sparse Neighborhood In our validation experiments, the combined model, which was trained to predict both a sparse neighborhood (SNB) and encoded central instance masks (ENB), achieved the best scores and yielded sharper and more accurate mask predictions. In general, providing losses for multiple tasks simultaneously has often been proven beneficial in a supervised learning setting. Moreover, the dense gradient of the *encoded-neighborhood branch*, which focuses on locally correct predictions, nicely complements the sparse gradient[4] of the *encoded-neighborhood branch*, which focuses on predictions that are consistent in a larger neighborhood. We expect this to be another reason why the combination of affinities and central instance masks performed best in our experiments.

Results on Test Samples The evaluation on the three test samples presented in Tab. 4.2 confirms our findings from the validation experiments: among the methods tested

[4]The gradient of the *encoded-neighborhood branch* is sparse, due to GPU-memory constraints as explained in Sec. 4.3.2.

Model	Train Sparse Neighbor. (SNB)	Train Encoded Masks (ENB)	Aggregate Overlapping Masks (MaskAggr)	Partitioning algorithm	No superpixels required	CREMI-Score (lower is better)
GaspUNet [15]	✗	□	□	WSDT+LMulticut	□	**0.221**
GaspUNet [15]	✗	□	□	GaspAvg	✗	0.224
PNIUNet [90]	✗	□	□	Z-Watershed+Agglo	□	0.228
OurUNet	✗	✗	✗	MWS	✗	0.246
OurUNet	□	✗	□	WSDT+GaspAvg	□	0.268
MALAUNet [54]	✗	□	□	WSDT+Multicut	□	0.276
OurUNet	✗	✗	□	WSDT+GaspAvg	□	0.280
CRUNet [163]	□	□	□	3D-Watershed	□	0.566
LFC [119]	✗	□	□	Z-Watershed+Agglo	□	0.616

Table 4.2. Representative excerpt of the published methods currently part of the CREMI leaderboard [40]. The best method proposed here achieves competitive scores and is based on an efficient parameter-free algorithm that does not rely on superpixels. For more details about the partitioning algorithms used by related work, see references in the first column.

here, the best scores are achieved by the combined model (ENB+SNB) and by using the Mutex Watershed algorithm (MWS) on affinities averaged over overlapping masks (MaskAggr). Our method achieves comparable scores to the only other method in the leader-board that does not rely on super-pixels (line 3 in Table 4.2). This method uses the average agglomeration algorithm GaspAvg proposed in [15] instead of the MWS. GaspAvg has been shown to be more robust to noise than Mutex Watershed, however it is also considerably more computationally expensive to run on large graphs like the ones considered here.

4.6 Conclusions

We have presented a new proposal-free method predicting encoded central instance masks in a sliding window style, one for each pixel of the input image, and introduced a parameter-free approach to aggregate predictions from overlapping masks and obtain all instances concurrently. When applied to large volumetric biological images, the resulting method proved to be strongly robust to noise and compared favorably to competing methods that need super-pixels and hence more hyper parameters.

Graph neighborhood structure (16 neighbors)	SNB_1 (18 neighbors)	SNB_2 (10 neighbors)	SNB_3 (10 neighbors)
(0, 0, -1)	(0, 0, -1)	(0, 0, -1)	(0, 0, -1)
(-1, 0, 0)	(-1, 0, 0)	(-4, 0, 0)	(-4, 0, 0)
(0, -1, 0)	(0, -1, 0)	(0, -4, 0)	(0, -4, 0)
(-4, 0, 0)	(-4, 0, 0)	(0, 0, -2)	(0, 0, -2)
(0, -4, 0)	(0, -4, 0)	(0, 0, -3)	(0, 0, -3)
(-4, -4, 0)	(-4, -4, 0)	(0, 0, -4)	(0, 0, -4)
(4, -4, 0)	(4, -4, 0)	(-14, 0, 0)	(-12, 0, 0)
(-4, 0, -1)	(-4, 0, -1)	(0, -14, 0)	(0, -12, 0)
(0, -4, -1)	(0, -4, -1)	(-14, -14, 0)	(-12, -12, 0)
(-4, -4, -1)	(-4, -4, -1)	(14, -14, 0)	(12, -12, 0)
(4, -4, -1)	(4, -4, -1)	-	-
(0, 0, -2)	(0, 0, -2)	-	-
(-8, -8, 0)	(0, 0, -3)	-	-
(8, -8, 0)	(0, 0, -4)	-	-
(-12, 0, 0)	(-8, -8, 0)	-	-
(0, -12, 0)	(8, -8, 0)	-	-
-	(-12, 0, 0)	-	-
-	(0, -12, 0)	-	-

Table 4.3. Sparse neighborhood structures – In this table, we represent sparse neighborhood structures (see for example the one shown in Fig. 4.1a) as lists of offsets $(\delta_x, \delta_y, \delta_z)$ indicating the relative coordinates of neighboring pixels with respect to the central pixel. The first column shows the neighborhood structure of the pixel grid-graph, such that each pixel / node is connected to 16 neighbors. In the following columns, we provide the neighborhood structures predicted by the three *sparse-neighborhood branches* SNB_i used in our model (see architecture in Fig. 4.5). These neighborhood structures were inspired by the ones used in [90,154] but were adapted to our version of the CREMI data that is downscaled by a factor $(\frac{1}{2}, \frac{1}{2}, 1)$. Note that the offsets provided here are given in the downscaled resolution.

Conclusions

In this thesis, we have introduced new graph-based instance segmentation methods and applied them to automated segmentation of neural tissue image volumes. Until a few years ago, most of the instance segmentation pipelines employed in neuron segmentation consisted of several post-processing steps [20, 54, 55, 90, 96, 101, 117]. Many of these pipelines included a fully convolutional neural network trained to delineate boundaries between different neuronal processes in the image. These boundary predictions were first processed by a superpixel-generation algorithm, which would then return an over-segmentation, i.e., a segmentation where all pixels in an image's segment belong to the same neuronal process, but there can be many segments associated with one neuron. This initial over-segmentation was then converted into a graph whose nodes represented superpixels, whereas edges expressed neighboring relationships. Finally, a graph partitioning algorithm returned the final instance segmentation.

These pipelines were rather complex and required the user to spend time tuning task-dependent hyper-parameters related to the superpixel generation algorithm. However, they used to be necessary to achieve state-of-the-art results because, at the time, deep learning models would often predict inconsistent boundaries, including holes and gaps. A way to fix this problem was then to generate superpixels and accumulate the model's predictions along boundaries separating the segments so that noise would be average out.

Recently, the performances of fully convolutional models have dramatically improved, primarily thanks to novel training strategies [90, 106]. Many instance segmentation methods now predict pixel-pair affinities [56, 90, 97], representing how likely it is for a pair of pixels to be in the same object instance. These approaches do not only predict affinities for pairs of direct-neighboring pixels: they also learn affinities between distant pixels. Predicting such long-range relations proved to be very beneficial during training because the model learns to more effectively detect large-scale features in images [90].

In this thesis, we have developed several new graph partitioning algorithms that are simple and parameter-free. By combining these algorithms with short- and long-range affinities from a deep convolutional neural network, we have shown that it is now possible to achieve state-of-the-art accuracies on neuron segmentation without relying on superpixels or other dataset-dependent post-processing pipelines.

In Chapter 2, we have introduced a new efficient algorithm, the Mutex Watershed, to cluster graphs with both attractive and repulsive edge weights. The proposed algorithm has low computational complexity and is closely related to Kruskal's algorithm for minimum spanning tree computation [84]. We prove that this algorithm finds the global optimum of an objective function and that this objective is closely related to the multicut optimization problem and the power watershed framework. The Mutex Watershed algorithm achieved state-of-the-art results on the ISBI neuron segmentation challenge. After our results were published, it also became part of two other state-of-the-art segmentation pipelines used for neuron segmentation [89, 99].

In Chapter 3, we have then proved that the Mutex Watershed algorithm is part of a larger generalized framework, named GASP, for agglomerative clustering of graphs with both positive and negative edge weights. We have analyzed several theoretical and empirical properties of the algorithms in it, by exploring new and existing combinations of linkage criteria and applying them to different types of graphs. Algorithms based on an average linkage criterion proved to be simple and robust approaches when applied to an instance segmentation task. On large volume images, these simple average agglomeration algorithms achieved state-of-the-art results on the competitive CREMI neuron segmentation challenge.

Finally, in Chapter 4, we introduced a novel proposal-free method predicting encoded single-instance masks in a sliding window style, one for each pixel of the input image, and introduced a parameter-free approach to aggregate predictions from overlapping masks and obtain all instances concurrently. The resulting pipeline proved to be strongly robust to noise when applied to large neuron segmentation volumetric images. The method also endows its predictions with an uncertainty measure, depending on the consensus of overlapping single-instance masks. Future work could use these uncertainty measures to estimate the confidence of individual instances, which may help facilitate the subsequent proof-reading step still needed in neuron segmentation.

The proposed unifying framework for agglomerative clustering algorithms also opens several opportunities for further research. Future work could investigate which objectives are optimized by the algorithms included in the GASP framework and study how they relate to the multicut / correlation clustering problem. A generalized GASP objective function could also be linked to existing cost functions for hierarchical clustering on unsigned graphs with positive edge weights [37, 44, 108]. Another research direction is the structured learning of graph edge weights, which optimizes the segmentation performance directly. The fact that this approach has already been successfully applied to instance segmentation and other graph partitioning algorithms [28, 54, 82, 155] suggests that it could be similarly used to achieve an end-to-end training of the segmentation pipelines presented in this thesis that rely on agglomerative algorithms in the GASP framework.

Appendices

Appendix A

GASP

A.1 Implementation and complexity of GASP

A.1.1 Update rules

During the agglomerative process, the interaction between adjacent clusters has to be properly updated and recomputed, as shown in Algorithm 8. An efficient way of implementing these updates can be achieved by representing the agglomeration as a sequence of *edge contractions* in the graph. Given a graph $\mathcal{G}(V, E, w)$ and a clustering Π, we define the associated *contracted graph* $\tilde{\mathcal{G}}_\Pi(\tilde{V}, \tilde{E}, \tilde{w})$, such that there exists exactly one representative node $|\tilde{V} \cap S| = 1$ for every cluster $S \in \Pi$. Edges in $\tilde{E}$ represent adjacency-relationships between clusters and the signed edge weights $\tilde{w}_e$ are given by inter-cluster interactions $\tilde{w}(e_{uv}) = \mathcal{W}_{S_u,S_v}$, where S_u denotes the clustering including node u. For the linkage criteria tested in this article, when two clusters S_u and S_v are merged, the interactions between the new cluster $S_u \cup S_v$ and each of its neighbors depend only on the previous interactions involving S_u and S_v. Thus, we can recompute these interactions by using an *update rule* f that does not involve any loop over the edges of the original graph $\mathcal{G}$:

$$\mathcal{W}(S_u \cup S_v, S_t) = f\Big[\mathcal{W}(S_u, S_t), \mathcal{W}(S_v, S_t)\Big] \tag{A.1}$$

$$= f(\tilde{w}(e_{ut}), \tilde{w}(e_{vt})) \tag{A.2}$$

In Fig. A.1 we show an example of edge contraction and in Table A.1 we list the update rules associated to the linkage criteria we introduced in Table 3.1.

A.1.2 Implementation

Our implementation of GASP is based on an union-find data structure and a heap allowing deletion of its elements. In Phases 2 and 3, GASP is equivalent to a standard hierarchical agglomerative clustering algorithm with complexity $\mathcal{O}(N^2 \log N)$. In Algorithm 10, we show our implementation of phase 1, involving cannot-link constraints. In phase 1, the algorithm starts with each node assigned to its own cluster and sorts all edges $e \in E$ in a heap/priority queue (PQ) by their absolute weight $|w_e| = |w_e^+ - w_e^-|$ in descending order, so that the most attractive and the most repulsive interactions are processed first. It then iteratively pops one edge e_{uv} from PQ and, depending on the priority $\tilde{w}_{uv}$, does the following: in case of attractive interaction $\tilde{w}_{uv} > 0$, provided that e_{uv} was not flagged as a cannot-link constraint, merge the connected clusters, perform an edge contraction of e_{uv} in

Algorithm 10 Implementation of GASP - Phase 1

Input: $\mathcal{G}(V, E, w^+, w^-)$ with N nodes and M edges; boolean `addCannotLinkConstraints`
Output: Final clustering

$\tilde{\mathcal{G}}(\tilde{V}, \tilde{E}) \leftarrow \mathcal{G}(V, E, w^+, w^-)$ ▷ Init. contracted graph
`UF` $\leftarrow$ initUnionFind(V) ▷ Init. data structure representing clustering
PQ.push($|w_e|, e$) $\forall e \in E$ ▷ Init. priority queue in desc. order of $|w_e| = |w_e^+ - w_e^-|$, $\mathcal{O}(|E|)$
`canBeMerged`[e] $\leftarrow$ `True` $\forall e \in E$ ▷ Init. cannot-link constraints

while PQ is **not** empty **do**
  $\tilde{w}, e_{uv} \leftarrow$ PQ.popHighest() ▷ $\mathcal{O}(\log |E|)$
  assert `UF`.find(u) $\neq$ `UF`.find(v) ▷ Edges in PQ always link nodes in different clusters
  if ($\tilde{w} > 0$) **and** `canBeMerged`[e_{uv}] **then**
    PQ, `canBeMerged`, $\tilde{E} \leftarrow$ UpdateNeighbors(u, v)
    $\tilde{V} \leftarrow \tilde{V} \setminus \{v\}$, $\tilde{E} \leftarrow \tilde{E} \setminus \{e_{uv}\}$ ▷ Update contracted graph
    `UF`.merge(u, v) ▷ Merge clusters, $\mathcal{O}(\alpha(|E|))$
  else if ($\tilde{w} \leq 0$) **and** `addCannotLinkConstraints` **then**
    `canBeMerged`[e_{uv}] $\leftarrow$ `False` ▷ Constrain the two clusters
return Final clustering given by union-find data structure `UF`

function UpdateNeighbors(u, v)
  $\mathcal{N}_u = \{t \in \tilde{V} | e_{ut} \in \tilde{E}\}$
  $\mathcal{N}_v = \{t \in \tilde{V} | e_{vt} \in \tilde{E}\}$
  for $t \in \mathcal{N}_v$ **do** ▷ Loop over neighbors in $\tilde{\mathcal{G}}$ of deleted node v
    $\tilde{E} \leftarrow \tilde{E} \setminus \{e_{vt}\}$
    $\tilde{w}_{vt} \leftarrow$ PQ.delete(e_{vt}) ▷ Delete edge e_{vt} from PQ and get the old edge weight, $\mathcal{O}(\log |E|)$
    `canBeMerged`[e_{ut}] $\leftarrow$ `canBeMerged`[e_{ut}] **and** `canBeMerged`[e_{vt}]
    if $t \in \mathcal{N}_u$ **then** ▷ Check if t is a common neighbor of u and v
      $\tilde{w}_{ut} \leftarrow$ PQ.delete(e_{ut}) ▷ $\mathcal{O}(\log |E|)$
      PQ.push($|f(\tilde{w}_{ut}, \tilde{w}_{vt})|, e_{ut}$) ▷ $\mathcal{O}(\log |E|)$
    else
      $\tilde{E} \leftarrow \tilde{E} \cup \{e_{ut}\}$
      PQ.push($|\tilde{w}_{vt}|, e_{ut}$) ▷ $\mathcal{O}(\log |E|)$
  return PQ, `canBeMerged`, $\tilde{E}$

Algorithm 11 Mutex Watershed Algorithm proposed in Chapter 2

Input: $\mathcal{G}(V, E, w^+, w^-)$ with N nodes and M edges
Output: Final clustering

`UF` $\leftarrow$ initUnionFind(V)
for $(u, v) = e \in E$ in descending order of $|w_e| = |w_e^+ - w_e^-|$ **do**
  if `UF`.find(u) $\neq$ `UF`.find(v) **then** ▷ Check if u, v are already in the same cluster
    if ($w_e > 0$) **and** `canBeMerged`(u, v) **then** ▷ Check for cannot-link constraints
      `UF`.merge(u, v) and inherit constraints of parent clusters
    else if ($w_e \leq 0$) **then**
      Add cannot-link constraints between parent clusters of u, v
return Final clustering given by union-find data structure `UF`

Linkage criteria	Update rule f
Sum:	$f(\tilde{w}_1, \tilde{w}_2) = \tilde{w}_1 + \tilde{w}_2$
Absolute Maximum:	$f(\tilde{w}_1, \tilde{w}_2) = \begin{cases} \tilde{w}_1 & \text{if } \lvert\tilde{w}_1\rvert > \lvert\tilde{w}_2\rvert \\ \tilde{w}_2 & \text{otherwise} \end{cases}$
Average:	$f(\tilde{w}_1, \tilde{w}_2) = \text{weightAvg}\{\tilde{w}_1, \tilde{w}_2\}$
Single:	$f(\tilde{w}_1, \tilde{w}_2) = \max\{\tilde{w}_1, \tilde{w}_2\}$
Complete:	$f(\tilde{w}_1, \tilde{w}_2) = \min\{\tilde{w}_1, \tilde{w}_2\}$

Table A.1. The table lists the update rules $f(\tilde{w}_1, \tilde{w}_2)$ associated to the linkage criteria of Table 3.1 and that are used to efficiently update the interactions between clusters.

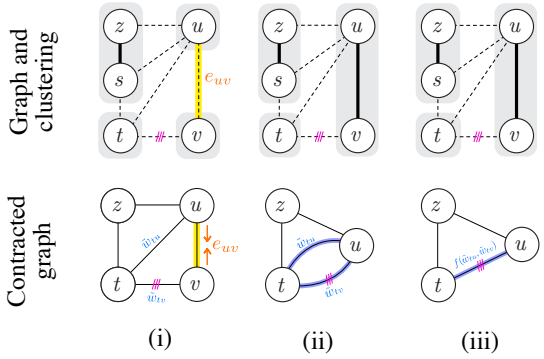


Figure A.1. Example of edge contraction. First row: original graph $\mathcal{G}$; clustering Π (gray shaded areas) with dashed edges on cut; cannot-link constraints (violet bars). Second row: contracted graph $\tilde{\mathcal{G}}_\Pi$. In step ii), edge e_{uv} is contracted and node v deleted from $\tilde{\mathcal{G}}_\Pi$. In step iii), double edges e_{tu} and e_{tv} resulting from the edge contraction are replaced by a single edge with updated interaction.

$\tilde{\mathcal{G}}_\Pi$ and update the priorities of new double edges as explained in Fig. A.1. If, on the other hand, the interaction is repulsive ($\tilde{w}_{uv} \leq 0$) and the option `addCannotLinkContraints` of Alg. 10 is `True`, then the edge e_{uv} is flagged as cannot-link constraint.

A.1.3 Complexity

In the main loop of Phase 1, the algorithm iterates over all edges, but the only iterations presenting a complexity different from $\mathcal{O}(1)$ are the ones involving a merge of two clusters, which are at most $N-1$. By using a union-find data structure (with path compression and union by rank) the time complexity of `merge`(u, v) and `find`(u) operations is $\mathcal{O}(\alpha(N))$, where α is the slowly growing inverse Ackerman function. The algorithm then iterates over the neighbors of the merged cluster (at most N) and updates/deletes values in the priority queue ($\mathcal{O}(\log |E|)$). Therefore, similarly to a heap-based implementation of hierarchical agglomerative clustering, our implementation of GASP - Phase 1 has a complexity of $\mathcal{O}(N^2 \log N)$. In the worst case, when the graph is dense and $|E| = N^2$, the algorithm requires $\mathcal{O}(N^2)$ memory. Nevertheless, in our practical applications the graph is much sparser, so $\mathcal{O}(|E|) = \mathcal{O}(N)$. With a single-linkage, corresponding to the

choice of the *Maximum* update rule in our framework, the algorithm can be implemented by using the more efficient Kruskal's Minimum Spanning Tree algorithm with complexity $\mathcal{O}(N \log N)$, but only when cannotLinkConstraints are not used. Moreover, GASP with *Absolute Maximum* linkage can be implemented more efficiently (see next section).

A.2 Proofs of Propositions in Section 3.3.3

Lemma A.2.1. *If GASP Algorithm 8 with* ***Complete linkage criteria*** *enforces a constraint between two clusters in* Phase 1*, then the interaction between the clusters will never become positive over the course of the following agglomeration steps.*

Proof. Two clusters are constrained in *Phase 1* only if their interaction is repulsive and, with complete linkage, the signed interaction between two clusters can only decrease over the course of the agglomeration. Thus, if two clusters are constrained by the algorithm, their negative interaction cannot increase and become positive later on in the agglomeration process. □

Lemma A.2.2. *If GASP Algorithm 8 with* ***AbsMax linkage criteria*** *enforces a constraint between two clusters in* Phase 1*, then the interaction between the clusters will never become positive over the course of the following agglomeration steps.*

Proof. During the agglomeration the interaction between two clusters can only increase in absolute value. Thus, the negative interaction $\mathcal{W}(S_i, S_j) < 0$ between two constrained clusters can possibly become positive over the course of next agglomeration steps only if there is at least another pair of clusters in the graph that has a positive interaction $\mathcal{W}(S_l, S_t) > 0$ higher in absolute value: $|\mathcal{W}(S_l, S_t)| > |\mathcal{W}(S_i, S_j)|$. If such clusters S_l, S_t with positive interaction exist, we note that they must also be constrained (in the opposite case, the algorithm would have already merged them before to constrain S_i and S_j, because their priority is higher). In other words, a constrained negative interaction can become positive only if there is already another positive constrained interaction: but this can never be the case because initially all constrained interactions are negative. □

Lemma A.2.3. *In the GASP Algorithm 8 with AbsMax or Complete linkage criteria (see linkage definition in Table 3.1), the same final clustering is returned whether or not cannot-link constraints are enforced.*

Proof. In phase 1 of Algorithm 10, two clusters are merged only if the condition at line 9 is satisfied (i.e. when an interaction is both positive and not constrained). From Lemma A.2.1 and Lemma A.2.2 follows that with Complete and AbsMax linkage an interaction can never be both positive and constrained at the same time, so we directly conclude that the constrained and unconstrained versions of the algorithm will perform precisely the same agglomeration steps in phase 1. In phase 2 (after constraints have been removed) no clusters are merged because all interactions are already negative (whether they previously constrained or not). Thus, both constrained and unconstrained versions of GASP return the same clustering Π^*. □

Proposition 3.3.1. *The GASP Algorithm 8 with AbsMax linkage, with or without cannot link constraints, returns the same final clustering* Π^*_{AbsMax} *also returned by the Mutex Watershed Algorithm (MWS) [154] (see Chapter 2), which has empirical complexity* $\mathcal{O}(N \log N)$.

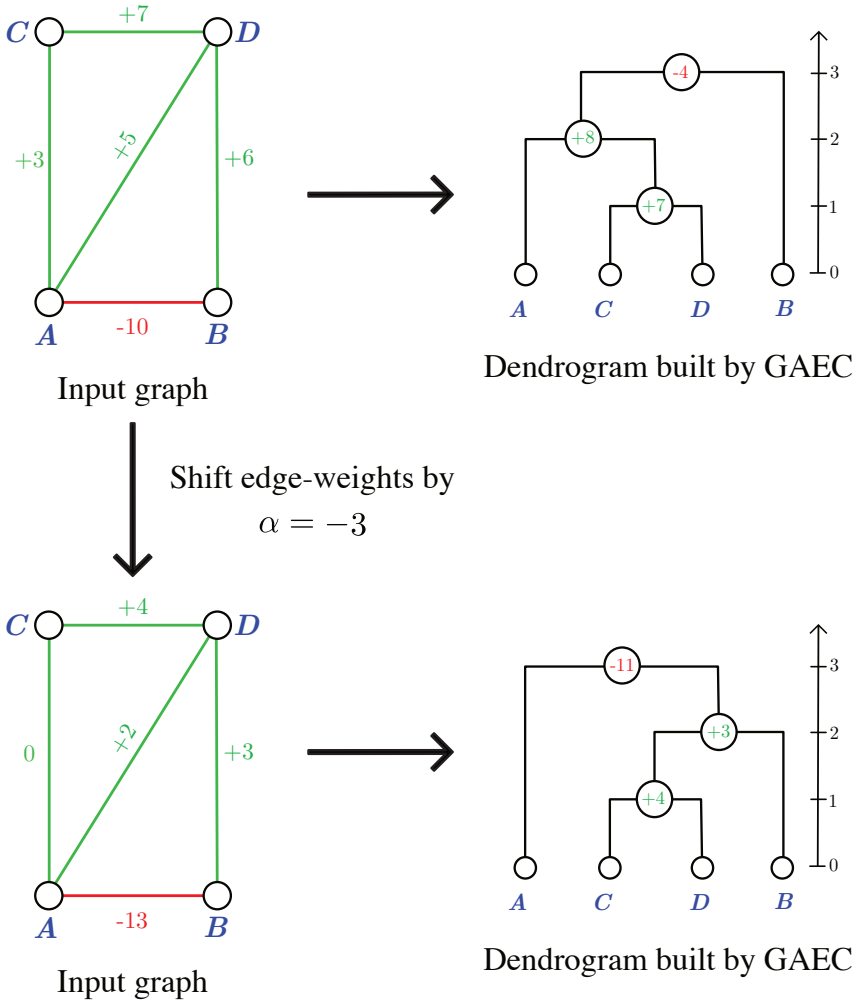


Figure A.2. Counter-example showing that GAEC is not weight-shift invariant.

Proof. From Lemma A.2.3 it directly follows that GASP with AbsMax linkage criterion returns the same final clustering whether or not cannot-link constraints are enforced. In the following, we prove that MWS (see pseudocode 11) and the constrained AbsMax version of GASP also return the same clustering. Both algorithms sort edges in descending order of the absolute interactions $|w_e|$ and then iterate over all of them. The only difference is that MWS, after merging two clusters, does not update the interactions between the new cluster and its neighbors. However, since with an Abs. Max. linkage the interaction between clusters is simply given by the edge with highest absolute weight $|w_e|$, the order by which edges are iterated over in GASP is never updated. Thus, both algorithms perform precisely the same steps and return the same clustering. □

Proposition 3.3.2. *We call an agglomerative algorithm "weight-shift invariant" if the dendrogram T returned by the algorithm is invariant w.r.t. a shift of all edge weights w_e by a constant $\alpha \in \mathbb{R}$. Among the variations of GASP, only hierarchical clustering with Average (HC-Avg), Single (HC-Single), and Complete linkage (HC-Complete) are weight-shift-invariant (see green box in Table 3.1).*

Proof. Theorem 1 in [29] proves that hierarchical clustering with Average (HC-Avg),

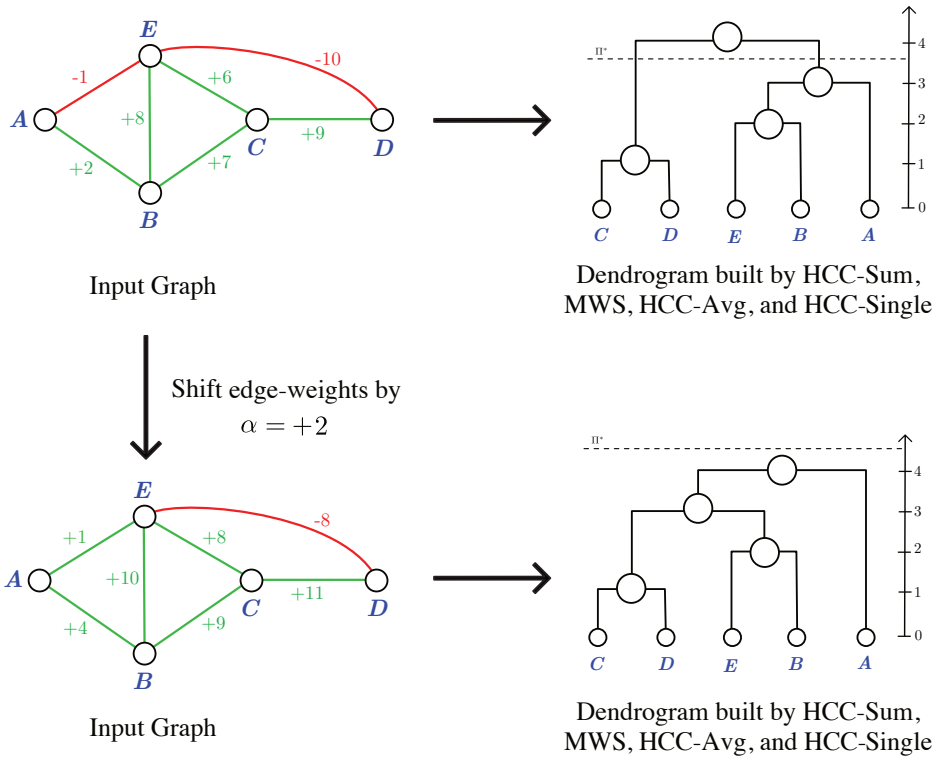


Figure A.3. Counter-example showing that HCC-Sum, MWS, HCC-Avg, and HCC-Single are not weight-shift invariant.

Single (HC-Single), and Complete linkage (HC-Complete) are weight-shift invariant.

The same is not true for GASP with Sum linkage criteria (GAEC and HCC-Sum), because by adding a constant α to all edge weights w_e, the interaction between two clusters S_i and S_j is increased by a factor $\alpha|E_{ij}|$, which depends on the number of edges $|E_{ij}|$ connecting the two clusters. Thus, when all edge weighs of the graph are shifted, the agglomeration order may change. For a simple example of this, it is enough to consider the toy graph in Fig. 3.1a and shift the weights of the graph by $\alpha = -3$ (see Fig. A.2).

The constrained versions of GASP (HCC-Avg and HCC-Single) are also not weight-shift invariant: here, the algorithm merges or constrains clusters in a given order, depending on the absolute interactions $|\mathcal{W}(S_i, S_j)|$ between clusters; so, when edge weights are shifted by a constant α, the sorting by absolute value can change arbitrarily together with the agglomeration order, as we show in the counter-example of Fig. A.3. Similarly, the Mutex Watershed algorithm is not weight-shift invariant because it uses a linkage criterion that compares weights by their absolute values (see again counter-example in Fig. A.3. □

Proposition A.2.1. *Consider a graph $\mathcal{G}(V, E, w_e)$, a linkage criterion $\mathcal{W}$, and an agglomerative algorithm returning a binary rooted tree T with height h_T. Then, (V, d_T) defined in Eq. 3.3 is an ultrametric if and only if the following is true:*

$$\forall u, v, t \in V$$
$$h_T(u, v) < h_T(u, t) \Rightarrow \mathcal{W}_T(u, v) \geq \mathcal{W}_T(u, t) \tag{A.3}$$

In words, condition A.3 means: if the algorithm merges nodes u, v before to merge nodes u, t, then the signed interaction $\mathcal{W}_T(u, v)$ between u and v has to be higher or equal than $\mathcal{W}_T(u, t)$.

Proof. From the definition of d_T, it follows that:

$$d_T(u, u) = 0 \qquad \forall u \in V \tag{A.4}$$
$$d_T(u, v) \geq 0 \qquad \forall u, v \in V \tag{A.5}$$
$$d_T(u, v) = d_T(v, u) \qquad \forall u, v \in V. \tag{A.6}$$

In order to show that (V, d_T) is an ultrametric, we only need to prove the ultrametric property:

$$d_T(u, v) \leq \max\{d_T(u, t), d_T(v, t)\} \quad \forall u, v, t \in V. \tag{A.7}$$

When at least two of the three nodes $u, v, t \in V$ are the same, this property follows from Eq. A.4 and Eq. A.5. When nodes $u, v, t \in V$ are distinct, from the definition of d_T it follows that Eq. A.7 is equivalent to:

$$\mathcal{W}_T(u, v) \geq \min\{\mathcal{W}_T(u, t), \mathcal{W}_T(v, t)\}. \tag{A.8}$$

In the following, we prove both sides of the *if and only if* statement in the proposition. First, we prove the ($\Leftarrow$) side, i.e. that if assumption A.3 holds, then (V, d_T) is an ultrametric and A.8 holds.

Case 1: in Eq. A.8, $t \in V$ is part of the sub-tree $T[u \vee v]$. In other words, the algorithm first merges node t with either node u or v, and then u and v are merged together. Let us assume that t is first merged with u (the following proof also holds for the opposite case in which t is first merged with v):

$$h_T(u, t) < h_T(u, v) = h_T(v, t). \tag{A.9}$$

Thus, by combining the last equation with assumption (A.3), it follows that

$$\mathcal{W}_T(u, t) \geq \mathcal{W}_T(v, t) \quad \text{and} \quad \mathcal{W}_T(u, v) = \mathcal{W}_T(v, t) \tag{A.10}$$

and Eq. A.8 follows (becoming an equality in this case).

Case 2: in Eq. A.8, $t \in V$ is *not* part of the sub-tree $T[u \vee v]$. Thus, the algorithm first merges nodes u and v, and then it merges node t together with the cluster containing u and v:

$$h_T(u, v) < h_T(u, t) = h_T(v, t). \tag{A.11}$$

Thus, from assumption A.3 we have that

$$\mathcal{W}_T(u, v) \geq \mathcal{W}_T(u, t) \quad \text{and} \quad \mathcal{W}_T(u, v) \geq \mathcal{W}_T(v, t), \tag{A.12}$$

so also in this case Eq. A.8 follows.

Next, we are left to prove the ($\Rightarrow$) side of the *if and only if* statement: if (V, d_T) is an ultrametric, then assumption A.3 holds. To prove this statement, we first rephrase it in the following equivalent form: if assumption A.3 does not hold, then (V, d_T) is not an ultrametric and A.8 does not hold. If we negate assumption A.3, there must be at least three $u, v, t \in V$ such that:

$$h_T(u, v) < h_T(u, t) \quad \text{and} \quad \mathcal{W}_T(u, v) < \mathcal{W}_T(u, t). \tag{A.13}$$

The first condition, in words, is again assuming that the algorithm first merges nodes u and v, and later it also merges node t with the cluster containing u and v. Thus, we can rephrase this assumption as:

$$\mathcal{W}_T(u,v) < \mathcal{W}_T(u,t) = \mathcal{W}_T(v,t). \tag{A.14}$$

From this, it follows that

$$\mathcal{W}_T(u,v) < \min\{\mathcal{W}_T(u,t), \mathcal{W}_T(v,t)\}, \tag{A.15}$$

which is exactly the negation of the ultrametric property A.8. □

Proposition 3.3.3. *Among the algorithms included in the GASP framework (see Table 3.1), only Mutex Watershed and hierarchical clustering with Average (HC-Avg), Single (HC-Single) and Complete linkage (HC-Complete) define an ultrametric (V, d_{T^*}), where d_{T^*} is defined in Eq. 3.3 and T^* is the tree returned by the GASP Algorithm 8.*

Proof. Thanks to Prop. A.2.1, we know that (V, d_{T^*}) is an ultrametric if and only if assumption A.3 holds. Thus, in the following, we will prove which variations of the GASP Algorithm 8 satisfy assumption A.3. In other words, we need to prove in which cases GASP merges clusters according to a monotonously decreasing order of signed interactions $\mathcal{W}$.

GASP puts clusters in a priority queue (Algorithm 8, lines 5 and 15) and merges them starting from those with the highest interaction (lines 9, 19, and 26). However, the priority queue is updated each time two clusters are merged (lines 10, 20, and 27). Thus, to ensure a monotonously decreasing merging order, updated interactions involving a merged cluster should always be lower or equal than previously existing interactions (**condition 1**):

$$\begin{gathered}\forall S_i \in \Pi \setminus \{S_1, S_2\}, \\ \mathcal{W}(S_1 \cup S_2, S_i) \leq \max\{\mathcal{W}(S_1, S_i), \mathcal{W}(S_2, S_i)\}\end{gathered} \tag{A.16}$$

where Π is a clustering, $\mathcal{W}$ is a linkage criteria, and $S_1, S_2 \in \Pi$ are two clusters merged by the algorithm at a given iteration. If this condition is true then, in the following iterations, GASP can only merge clusters with lower (or equal) interaction values.

We also note that, in phase 1, the algorithm skips interactions that are both positive and constraint (condition at line 8 in Algorithm 8) and merges them only later in phase 2 (line 19), when constraints are removed. Clearly, whenever this happens, a decreasing merging order is no longer ensured. Thus, on top of condition 1, we also have that no merging decisions should be "delayed" from phase 1 to phase 2 (**condition 2**).

Condition 1 always holds for Average, Single, Complete, and AbsMax linkage criteria, but not for a Sum linkage criteria, because the sum of two positive numbers a, b is always higher than $\max\{a, b\}$. This is also demonstrated in the toy example of Fig. 3.1a, proving that, in general, Sum-linkage algorithms like GAEC or HCC-Sum do not define an ultrametric on the graph.

Thanks to Lemma A.2.3, we have that condition 2 always holds for algorithms based on AbsMax and Complete linkage, proving that the Mutex Watershed and HC-Complete algorithms define an ultra-metric (whether or not cannot-link-constraints are enforced). On the other hand, condition 2 does not hold for other variations of GASP involving cannot-link-constraints (HCC-Sum, HCC-Avg, and HCC-Single), which do not then define an ultrametric.

Finally, the remaining not constrained versions of GASP (HC-Avg, HC-Single, and HC-Complete) satisfy both conditions, so they define an ultrametric, confirming the well-known results of related work in hierarchical clustering on unsigned graphs [66, 107].

□

A.3 Adding structured noise to CNN predictions

Additionally to the comparison on the full training dataset, we performed more experiments on a crop of the more challenging CREMI training sample B, where we perturbed the predictions of the CNN with noise and we introduced additional artifacts like missing boundary evidences.

In the field of image processing there are several ways of adding noise to an image, among which the most common are Gaussian noise or Poisson shot noise. In these cases, the noise of one pixel does not correlate with its neighboring noise values. On the other hand, predictions of a CNN are known to be spatially correlated. Thus, we used Perlin noise[1], one of the most common gradient noises used in procedural pattern generation. This type of noise $n(x) \in [0, 1]$ generates spatial random patterns that are locally smooth but have large and diverse variations on bigger scales. We then combined it with the CNN predictions $p(x)$ in the following way:

$$\tilde{F}(x; \mathcal{K}) = F(x) + \mathcal{K} \cdot \max\left(N(x), 0\right), \tag{A.17}$$

where $N(x) = \text{Logit}[n(x)]$; $F(x) = \text{Logit}[p(x)]$ and $\mathcal{K} \in \mathbb{R}^+$ is a positive factor representing the amount of added noise. The resulting perturbed predictions $\tilde{F}(x; \mathcal{K})$ are then under-clustering biased, such that the probability for two pixels to be in the same cluster is increased only if $N(x) > 0$ (see Fig. A.4b and A.4c). Note that in these experiments we focused only on predictions perturbed with under-clustering biased noise (and not over-clustering biased noise). The reason is that generating realistic over-clustering biased CNN predictions is more complex and cannot be simply done by adding Perlin noise: as we show in Fig. A.4c, by adding Perlin noise we can easily "remove" parts of a boundary evidence, but it is not possible to generate random new realistic boundary evidence.

In our experiments, each pixel is represented by a node in the grid-graph and it is linked to n_{nb} other nodes by short- and long-range edges. Thus, the output volume of our CNN model is a four-dimensional tensor with n_{nb} channels: for each pixel / voxel, the model outputs n_{nb} values representing affinities of different edge connections. We then generated a 4-dimensional Perlin noise tensor that matches the dimension of the CNN output. The data is highly anisotropic, i.e. it has a lower resolution in one of the dimensions. Due to this fact, we chose different smoothing parameters to generate the noise in different directions.

[1] In our experiments, we used an open-source implementation of simplex noise [121], which is an improved version of Perlin noise [120]

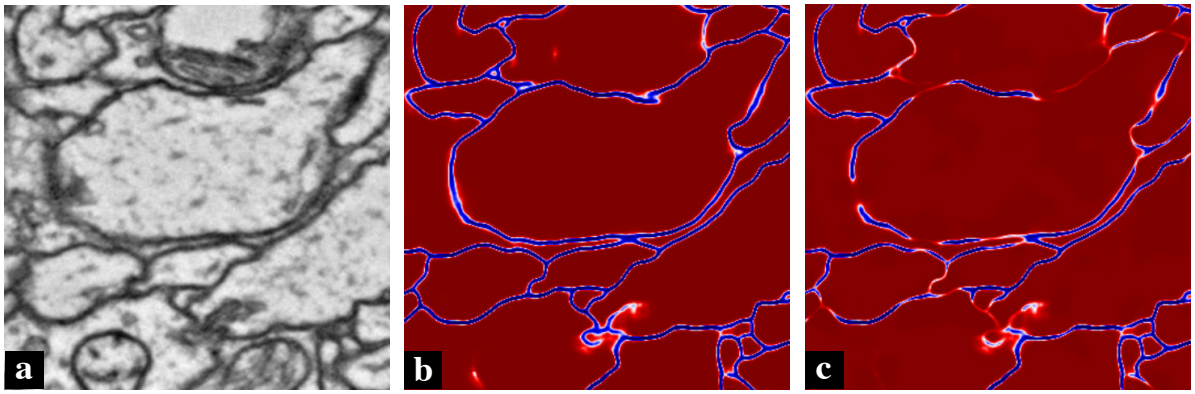


Figure A.4. CNN predictions on a slice of the CREMI neuron segmentation challenge with and without additional spatially-correlated noise. (**a**) Raw data (**b**) Original CNN predictions $F(x)$, where blue pixels represent boundary evidence (**c**) Strongly perturbed version $\tilde{F}(x;\mathcal{K})$ of the predictions defined in Eq. A.17 with $\mathcal{K} = 8$. Long-range predictions are not shown.

List of Publications

I contributed to the following peer reviewed publications:

- **Alberto Bailoni**[2], Steffen Wolf[2], Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A. Hamprecht. "The Mutex Watershed and Its Objective: Efficient, Parameter-Free Image Partitioning." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 2020.

- **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 (GCPR)*. 2020.

- Steffen Wolf, Constantin Pape, **Alberto Bailoni**, Nasim Rahaman, Anna Kreshuk, Ullrich Kothe, and Fred A. Hamprecht. "The mutex watershed: efficient, parameter-free image partitioning." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

- Elke Kirschbaum, **Alberto Bailoni**, and Fred A. Hamprecht. "DISCo: Deep learning, Instance Segmentation, and Correlations for cell segmentation in calcium imaging". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (ISBI)*. 2020.

- 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: *Proceedings of the European Conference on Computer Vision (ECCV)* (2020).

- Adrian Wolny, Lorenzo Cerrone, Athul Vijayan, Rachele Tofanelli, Amaya Vilches Barro, Marion Louveaux, Christian Wenzl, Sören Strauss, David Wilson-Sánchez, Rena Lymbouridou, Susanne S Steigleder, Constantin Pape, **Alberto Bailoni**, Salva Duran-Nebreda, George W Bassel, Jan U Lohmann, Miltos Tsiantis, Fred A Hamprecht, Kay Schneitz, Alexis Maizel, and Anna Kreshuk. "Accurate and versatile 3D segmentation of plant tissues at cellular resolution". In: *Elife, 9, e57613.* 2020.

The following publications are currently under peer review:

- **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". *arXiv preprint arXiv:1906.11713.* (2021)

[2]Both authors contributed equally

Bibliography

[1] M. Abdelsamea. An enhancement neighborhood connected segmentation for 2D-Cellular Image. *International Journal of Bioscience, Biochemistry and Bioinformatics*, 1(4), 2011.

[2] A. Q. Al-Faris, U. K. Ngah, N. A. M. Isa, and I. L. Shuaib. Breast MRI tumour segmentation using modified automatic seeded region growing based on particle swarm optimization image clustering. In *Soft Computing in Industrial Applications*, pages 49–60. Springer, 2014.

[3] A. Q. Al-Faris, U. K. Ngah, N. A. M. Isa, and I. L. Shuaib. Computer-aided segmentation system for breast MRI tumour using modified automatic seeded region growing (BMRI-MASRG). *Journal of digital imaging*, 27(1):133–144, 2014.

[4] M. A. Alattar, N. F. Osman, and A. S. Fahmy. Myocardial segmentation using constrained multi-seeded region growing. In *International Conference Image Analysis and Recognition*, pages 89–98. Springer, 2010.

[5] B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *2011 International Conference on Computer Vision*, pages 2611–2618. IEEE, 2011.

[6] B. Andres, U. Koethe, T. Kroeger, M. Helmstaedter, K. L. Briggman, W. Denk, and F. A. Hamprecht. 3d segmentation of sbfsem images of neuropil by a graphical model over supervoxel boundaries. *Medical image analysis*, 16(4):796–805, 2012.

[7] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht. Globally optimal closed-surface segmentation for connectomics. In *European Conference on Computer Vision*, pages 778–791. Springer, 2012.

[8] B. Andres, T. Kröger, K. L. Briggmann, W. Denk, N. Norogod, G. Knott, U. Köthe, and F. A. Hamprecht. Globally optimal closed-surface segmentation for connectomics. In *Proc. ECCV'12, part 2*, number 7574, pages 778–791, 2012.

[9] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 33(5):898–916, 2011.

[10] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[11] I. Arganda-Carreras, S. Turaga, D. Berger, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Front. Neuroanatomy*, 9:142, 2015.

[12] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:142, 2015.

[13] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2858–2866. IEEE, 2017.

[14] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[15] A. Bailoni, C. Pape, S. Wolf, T. Beier, A. Kreshuk, and F. A. Hamprecht. A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation. *arXiv preprint arXiv:1906.11713*, 2019.

[16] A. Bailoni, C. Pape, S. Wolf, A. Kreshuk, and F. A. Hamprecht. Proposal-free volumetric instance segmentation from latent single-instance masks. *arXiv preprint arXiv:2009.04998*, 2020.

[17] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

[18] T. Beier, B. Andres, U. Köthe, and F. A. Hamprecht. An efficient fusion move algorithm for the minimum cost lifted multicut problem. In *European Conference on Computer Vision*, pages 715–730. Springer, 2016.

[19] T. Beier, T. Kroeger, J. H. Kappes, U. Kothe, and F. A. Hamprecht. Cut, glue & cut: A fast, approximate solver for multicut partitioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2014.

[20] T. Beier, C. Pape, N. Rahaman, and T. e. a. Prange. Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*, 14(2):101–102, 2017.

[21] S. Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In *Proc. ISMM'94*, volume 94, pages 69–76, 1994.

[22] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Int. Workshop on Image Processing*, Rennes, France, Sept. 1979. CCETT/IRISA.

[23] S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. *Optical Engineering*, 34:433–433, 1992.

[24] A. Braides. A handbook of γ-convergence. In *Handbook of Differential Equations: stationary partial differential equations*, volume 3, pages 101–213. Elsevier, 2006.

[25] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2):172–188, 2007.

[26] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga. Maximin affinity learning of image segmentation. In *Advances in Neural Information Processing Systems*, pages 1865–1873, 2009.

[27] J. Cai, L. Lu, Z. Zhang, F. Xing, L. Yang, and Q. Yin. Pancreas segmentation in MRI using graph-based decision fusion on convolutional neural networks. In *Proc. MICCAI*, 2016.

[28] L. Cerrone, A. Zeilmann, and F. A. Hamprecht. End-to-end learned random walker for seeded image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12559–12568, 2019.

[29] M. H. Chehreghani. Hierarchical correlation clustering and tree preserving embedding. *arXiv preprint arXiv:2002.07756*, 2020.

[30] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-DeepLab. *arXiv preprint arXiv:1910.04751*, 2019.

[31] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon. Scalable clustering of signed networks using balance normalized cut. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 615–624. ACM, 2012.

[32] S. Chopra and M. R. Rao. On the multiway cut polyhedron. *Networks*, 21(1):51–89, 1991.

[33] S. Chopra and M. R. Rao. The partition problem. *Mathematical Programming*, 59(1-3):87–115, 1993.

[34] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[35] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[36] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. *Proc. NIPS'12*, 2012.

[37] V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, and C. Mathieu. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, 66(4):1–42, 2019.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[39] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Patt. Anal. Mach. Intell.*, 33(7), 2011.

[40] C. CREMI. Miccai challenge on circuit reconstruction from electron microscopy images, 2017.

[41] M. Cucuringu, P. Davies, A. Glielmo, and H. Tyagi. SPONGE: A generalized eigenproblem for clustering signed networks. In *AISTATS*, 2019.

[42] M. Cucuringu, I. Koutis, S. Chawla, G. Miller, and R. Peng. Simple and scalable constrained clustering: a generalized spectral method. In *Artificial Intelligence and Statistics*, pages 445–454, 2016.

[43] G. Dal Maso. *An introduction to* Γ*-convergence*, volume 8. Springer Science & Business Media, 2012.

[44] S. Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127, 2016.

[45] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[46] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

[47] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[48] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Trans. Patt. Anal. Mach. Intell.*, 26(1):19–29, 2004.

[49] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[50] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

[51] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[52] J. R. Finkel and C. D. Manning. Enforcing transitivity in coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 45–48. Association for Computational Linguistics, 2008.

[53] J. Funke, F. A. Hamprecht, and C. Zhang. Learning to segment: training hierarchical segmentation under a topological loss. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 268–275. Springer, 2015.

[54] J. Funke, F. D. Tschopp, W. Grisaitis, A. Sheridan, C. Singh, S. Saalfeld, and S. C. Turaga. Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[55] J. Funke, C. Zhang, T. Pietzsch, M. A. G. Ballester, and S. Saalfeld. The candidate multi-cut for cell segmentation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 649–653. IEEE, 2018.

[56] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang. SSAP: Single-shot instance segmentation with affinity pyramid. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[57] M. Grimaud. New measure of contrast: the dynamics. In P. D. Gader, E. R. Dougherty, & J. C. Serra, editor, *Proc. Image Algebra and Morphological Processing*, volume 1769 of *SPIE Conf. Series*, pages 292–305, 1992.

[58] M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1-3):59–96, 1989.

[59] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1-3):367–387, 1990.

[60] L. Guigues, J. P. Cocquerez, and H. Le Men. Scale-sets image analysis. *International Journal of Computer Vision*, 68(3):289–317, 2006.

[61] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[62] A. Horňáková, J.-H. Lange, and B. Andres. Analysis and optimization of graph decompositions by lifted multicuts. In *International Conference on Machine Learning*, pages 1539–1548, 2017.

[63] V. Jain, J. F. Murray, F. Roth, S. Turaga, V. Zhigulin, K. L. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung. Supervised learning of image restoration with convolutional networks. *Proc. ICCV'07*, pages 1–8, 2007.

[64] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitin-Shepard, M. Tyka, W. Denk, and V. Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8):605, 2018.

[65] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitin-Shepard, M. Tyka, W. Denk, and V. Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, page 1, 2018.

[66] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[67] J. Kappes, B. Andres, F. Hamprecht, C. Schnorr, S. Nowozin, D. Batra, S. Kim, B. Kausler, J. Lellmann, N. Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1335, 2013.

[68] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schn. Globally optimal image partitioning by multicuts. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 31–44. Springer, 2011.

[69] A. Kardoost and M. Keuper. Solving minimum cost lifted multicut problems by node agglomeration. In *ACCV 2018, 14th Asian Conference on Computer Vision*, Perth, Australia, 2018.

[70] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *Medical image analysis*, 22(1):77–88, 2015.

[71] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[72] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proc. ICCV'15*, pages 1751–1759, 2015.

[73] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects. *arXiv preprint arXiv:1607.06317*, 2016.

[74] J. S. Kim, M. J. Greene, A. Zlateski, K. Lee, M. Richardson, S. C. Turaga, M. Purcaro, M. Balkam, A. Robinson, B. F. Behabadi, et al. Space–time wiring specificity supports direction selectivity in the retina. *Nature*, 509(7500):331–336, 2014.

[75] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2015.

[76] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[77] B. R. Kiran and J. Serra. Global–local optimizations by hierarchical cuts and climbing energies. *Pattern Recognition*, 47(1):12–24, 2014.

[78] B. R. Kiran and J. Serra. Globallocal optimizations by hierarchical cuts and climbing energies. *Pattern Recognition*, 47(1):12–24, 2014.

[79] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017.

[80] S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, J. Morgan, D. Lee, D. Berger, N. Kasthuri, J. W. Lichtman, and H. Pfister. RhoanaNet pipeline: Dense automatic neural annotation. *arXiv:1611.06973*, 2016.

[81] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. *arXiv:1511.07386*, 2015.

[82] S. Kong and C. C. Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9018–9028, 2018.

[83] N. Krasowski, T. Beier, G. W. Knott, U. Koethe, F. A. Hamprecht, and A. Kreshuk. Improving 3D EM data segmentation by joint optimization over boundary evidence and biological priors. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 536–539. IEEE, 2015.

[84] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[85] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak. Spectral analysis of signed graphs for clustering, prediction and visualization. SIAM, 2010.

[86] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies: 1. Hierarchical systems. *The computer journal*, 9(4):373–380, 1967.

[87] J.-H. Lange, B. Andres, and P. Swoboda. Combinatorial persistency criteria for multicut and max-cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6093–6102, 2019.

[88] J.-H. Lange, A. Karrenbauer, and B. Andres. Partial optimality and fast lower bounds for weighted correlation clustering. In *International Conference on Machine Learning*, pages 2898–2907, 2018.

[89] K. Lee, R. Lu, K. Luther, and H. S. Seung. Learning dense voxel embeddings for 3d neuron reconstruction. *arXiv preprint arXiv:1909.09872*, 2019.

[90] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung. Superhuman accuracy on the SNEMI3D connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[91] Z. Levi and D. Zorin. Strict minimizers for geometric optimization. *ACM Trans. Graph.*, 33(6):185:1–185:14, Nov. 2014.

[92] E. Levinkov, A. Kirillov, and B. Andres. A comparative study of local search algorithms for correlation clustering. In *German Conference on Pattern Recognition*, pages 103–114. Springer, 2017.

[93] S. Liu, X. Qi, J. Shi, H. Zhang, and J. Jia. Multi-scale patch aggregation (MPA) for simultaneous detection and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3141–3149, 2016.

[94] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen. A modular hierarchical approach to 3D electron microscopy image segmentation. *Journal of neuroscience methods*, 226:88–102, 2014.

[95] T. Liu, M. Seyedhosseini, and T. Tasdizen. Image segmentation using hierarchical merge tree. *IEEE transactions on image processing*, 25(10):4596–4607, 2016.

[96] T. Liu, M. Zhang, M. Javanmardi, N. Ramesh, and T. Tasdizen. SSHMT: Semi-supervised hierarchical merge tree for electron microscopy image segmentation. In *European Conference on Computer Vision*, pages 144–159. Springer, 2016.

[97] Y. Liu, S. Yang, B. Li, W. Zhou, J. Xu, H. Li, and Y. Lu. Affinity derivation and graph merge for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 686–703, 2018.

[98] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[99] L. Mais, P. Hirsch, and D. Kainmueller. Patchperpix for instance segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 288–304. Springer International Publishing, 2020.

[100] F. Malmberg, R. Strand, and I. Nyström. Generalized hard constraints for graph segmentation. In *Scandinavian Conference on Image Analysis*, pages 36–47. Springer, 2011.

[101] Y. Meirovitch, A. Matveev, H. Saribekyan, D. Budden, D. Rolnick, G. Odor, S. K.-B. T. R. Jones, H. Pfister, J. W. Lichtman, and N. Shavit. A multi-pass approach to large-scale connectomics. *arXiv preprint:1612.02120*, 2016.

[102] Y. Meirovitch, L. Mi, H. Saribekyan, A. Matveev, D. Rolnick, and N. 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*, pages 8425–8435, 2019.

[103] F. Meyer. Minimum spanning forests for morphological segmentation. In *Mathematical morphology and its applications to image processing*, pages 77–84. 1994.

[104] F. Meyer. Topographic distance and watershed lines. *Signal processing*, 38(1):113–125, 1994.

[105] F. Meyer. Morphological multiscale and interactive segmentation. In *NSIP*, pages 369–377, 1999.

[106] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[107] G. W. Milligan. Ultrametric hierarchical clustering algorithms. *Psychometrika*, 44(3):343–346, 1979.

[108] B. Moseley and J. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *Advances in neural information processing systems*, 30:3094–3103, 2017.

[109] D. M. N. Mubarak, M. M. Sathik, S. Z. Beevi, and K. Revathy. A hybrid region growing algorithm for medical image segmentation. *International Journal of Computer Science & Information Technology*, 4(3):61, 2012.

[110] L. Najman. On the equivalence between hierarchical segmentations and ultrametric watersheds. *J. of Mathematical Imaging and Vision*, 40(3):231–247, 2011.

[111] L. Najman. Extending the power watershed framework thanks to Γ-convergence. *SIAM Journal on Imaging Sciences*, 10(4):2275–2292, 2017.

[112] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(12):1163–1173, 1996.

[113] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019.

[114] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[115] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. Chklovskii. Machine learning of hierarchical clustering to segment 2D and 3D images. *PLoS one*, 8:e71715, 2013.

[116] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment 2D and 3D images. *PloS one*, 8(8):e71715, 2013.

[117] C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk. Solving large multicut problems for connectomics via domain decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2017.

[118] C. Pape, A. Matskevych, A. Wolny, J. Hennies, G. Mizzon, M. Louveaux, J. Musser, A. Maizel, D. Arendt, and A. Kreshuk. Leveraging domain knowledge to improve microscopy image segmentation with lifted multicuts. *Frontiers in Computer Science*, 1:6, 2019.

[119] T. Parag, F. Tschopp, W. Grisaitis, S. C. Turaga, X. Zhang, B. Matejek, L. Kamentsky, J. W. Lichtman, and H. Pfister. Anisotropic EM segmentation by 3d affinity learning and agglomeration. *arXiv preprint 1707.08935*, 2017.

[120] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[121] K. Perlin. Noise hardware. *Real-Time Shading SIGGRAPH Course Notes*, 2001.

[122] B. Perret, J. Cousty, S. J. F. Guimaraes, and D. S. Maia. Evaluation of hierarchical watersheds. *IEEE Transactions on Image Processing*, 27(4):1676–1688, 2018.

[123] R. Pohle and K. D. Toennies. Segmentation of medical images using adaptive region growing. In *Medical Imaging 2001: Image Processing*, volume 4322, pages 1337–1346. International Society for Optics and Photonics, 2001.

[124] S. Poonguzhali and G. Ravindran. A complete automatic region growing method for segmentation of masses on ultrasound images. In *2006 International Conference on Biomedical and Pharmaceutical Engineering*, pages 88–92. IEEE, 2006.

[125] L. Porzi, S. R. Bulo, A. Colovic, and P. Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019.

[126] T. M. Quan, D. G. Hilderbrand, and W.-K. Jeong. FusionNet: a deep fully residual convolutional neural network for image segmentation in connectomics. *arXiv:1612.05360*, 2016.

[127] S. S. Rangapuram and M. Hein. Constrained 1-spectral clustering. In *AISTATS*, volume 30, page 90, 2012.

[128] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[129] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2013.

[130] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[131] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[132] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. *Proc. MICCAI'15*, pages 234–241, 2015.

[133] S. Saalfeld, R. Fetter, A. Cardona, and P. Tomancak. Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature methods*, 9(7):717, 2012.

[134] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Trans. Image Proc.*, 9:561–576, 2000.

[135] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE transactions on Image Processing*, 9(4):561–576, 2000.

[136] P. Schlegel, M. Costa, and G. S. X. E. Jefferis. Learning from connectomics on the fly. *Current opinion in insect science*, 24:96–105, 2017.

[137] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers. Cell detection with star-convex polygons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 265–273. Springer, 2018.

[138] J. Shan, H.-D. Cheng, and Y. Wang. A novel automatic seed point selection algorithm for breast ultrasound images. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[139] W. Shen, B. Wang, Y. Jiang, Y. Wang, and A. L. Yuille. Multi-stage multi-recursive-input fully convolutional networks for neuronal boundary detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2410–2419, 2017.

[140] C. SNEMI3D. Isbi 2013 challenge: 3d segmentation of neurites in em images, 2013.

[141] K. Sofiiuk, O. Barinova, and A. Konushin. AdaptIS: Adaptive instance selection network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7355–7363, 2019.

[142] P. Soille. Constrained connectivity for hierarchical image decomposition and simplification. *IEEE Trans. Patt. Anal. Mach. Intell.*, 30(7):1132–1145, 2008.

[143] T. 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. *Biol. Skr.*, 5:1–34, 1948.

[144] S. C. Turaga, K. L. Briggman, M. Helmstaedter, W. Denk, and H. S. Seung. Maximin affinity learning of image segmentation. pages 1865–1873, 2009.

[145] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

[146] M. G. Uzunbaş, C. Chen, and D. Metaxas. Optree: a learning-based adaptive watershed algorithm for neuron segmentation. In *Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'14)*, pages 97–105, 2014.

[147] M. G. Uzunbas, C. Chen, and D. Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical image analysis*, 27:31–44, 2016.

[148] C. Vachier and F. Meyer. Extinction value: a new measurement of persistence. In *Worksh. Nonlinear Signal and Image Processing*, volume 1, pages 254–257, 1995.

[149] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Analysis Machine Intelligence*, (6):583–598, 1991.

[150] X. Wang, B. Qian, and I. Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30, 2014.

[151] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant CNNs. pages 849–858, 2018.

[152] B. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[153] S. Wolf, A. Bailoni, C. Pape, N. Rahaman, A. Kreshuk, U. Köthe, and F. A. Hamprecht. The mutex watershed and its objective: Efficient, parameter-free image partitioning. *arXiv preprint arXiv:1904.12654*, 2019.

[154] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Köthe, and F. Hamprecht. The mutex watershed: Efficient, parameter-free image partitioning. *Proc. ECCV'18*, 2018.

[155] S. Wolf, L. Schott, U. Köthe, and F. Hamprecht. Learned watershed: End-to-end learning of seeded segmentation. *Proc. ICCV'17*, 2017.

[156] J. Wu, S. Poehlman, M. D. Noseworthy, and M. V. Kamath. Texture feature based automated seeded region growing in abdominal MRI segmentation. In *2008 International Conference on BioMedical Engineering and Informatics*, volume 2, pages 263–267. IEEE, 2008.

[157] C. Xiao, J. Liu, X. Chen, H. Han, C. Shu, and Q. Xie. Deep contextual residual network for electron microscopy image segmentation in connectomics. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 378–381. IEEE, 2018.

[158] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proc. ICCV'15*, pages 1395–1403, 2015.

[159] C. S. Xu, M. Januszewski, Z. Lu, S.-y. Takemura, K. Hayworth, G. Huang, K. Shinomiya, J. Maitin-Shepard, D. Ackerman, S. Berg, et al. A connectome of the adult drosophila central brain. *BioRxiv*, 2020.

[160] C. S. Xu, M. Januszewski, Z. Lu, S.-y. Takemura, K. J. Hayworth, et al. A connectome of the adult drosophila central brain. *bioRxiv*, 2020.

[161] J. Yarkony, A. Ihler, and C. C. Fowlkes. Fast planar correlation clustering for image segmentation. In *Proc. ECCV'12*, pages 568–581, 2012.

[162] W. Yin, D. Brittain, J. Borseth, M. E. Scott, D. Williams, J. Perkins, C. S. Own, M. Murfitt, R. M. Torres, D. Kapner, et al. A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nature communications*, 11(1):1–12, 2020.

[163] T. Zeng, B. Wu, and S. Ji. DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation. *Bioinformatics*, 33(16):2555–2562, 2017.

[164] C. Zhang, J. Yarkony, and F. A. Hamprecht. Cell detection and segmentation using correlation clustering. In *Proc. MICCAI'14*, pages 9–16, 2014.

[165] Z. Zheng, J. S. Lauritzen, E. Perlman, C. G. Robinson, M. Nichols, D. Milkie, O. Torrens, J. Price, C. B. Fisher, N. Sharifi, et al. A complete electron microscopy volume of the brain of adult drosophila melanogaster. *Cell*, 174(3):730–743, 2018.

[166] A. Zlateski and H. S. Seung. Image segmentation by size-dependent single linkage clustering of a watershed basin graph. *arXiv:1505.00249*, 2015.

List of Figures

List of Tables

List of Algorithms